

Getting Started with RoboRealm

Introduction

The RoboRealm application contains all the tools you need for processing video images for use in robotic applications. It is easy enough for the casual user who wants to experiment with video processing, yet powerful enough for the serious robotic scientist who needs to integrate a vision system into their project.

The heart of the video processor is the image processing pipeline that allows multiple modules (aka image processing filters) to be added in succession to generate the desired results. Combined with the programming logic and control interfaces one can create a visual processing loop that moves a robot according to what it sees. To get an idea on how the application works please watch our [Introduction to RoboRealm](#) video.

To try RoboRealm with your own camera and see what it can do  [Click here](#) to load a slideshow configuration that will cycle through various filters.

Main RoboRealm Interface

The main interface shows the live video, available functions and the processing pipeline. Adding modules into the pipeline will immediately begin processing of the image. You can reorder the modules in the pipeline by using the up and down yellow arrow buttons.

Theory

The main purpose of RoboRealm is to translate what a camera sees into meaningful numbers in order to cause a reaction to what the image contains. This requires the processing of an image into a few or a single number that is meaningful in the current context of whatever project you have in mind. As each project has vastly different requirements with different attributes that need to be detected, RoboRealm is written in a flexible approach where the basic tools are provided to you in the form of modules in order to be combined into a relevant combination. For example, if your task is to track a red ball then the color red can be picked up by a color filter module as the first step. But if you wanted to track any type of ball then you would not use the red color as an attribute but instead a round shape detector. The reason you would ever want to use a color tracker instead of just using a shape tracker is that the color tracker may be more reliable and repeatable in some environments.

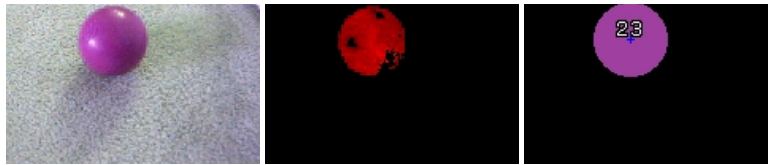
For example, if a red circle is only partially seen a color tracker would easily pick up half of the red ball but a circle detector might not due to the incomplete circle. The choice of which modules to use comes with experience and is very dependant on the environment being sampled. For this reason we regularly will ask for images in the forum as apposed to a description as images will reveal many attributes such as image quality, lighting, color, etc. in the environment.

Original Image

Red Identified

Circle Identified

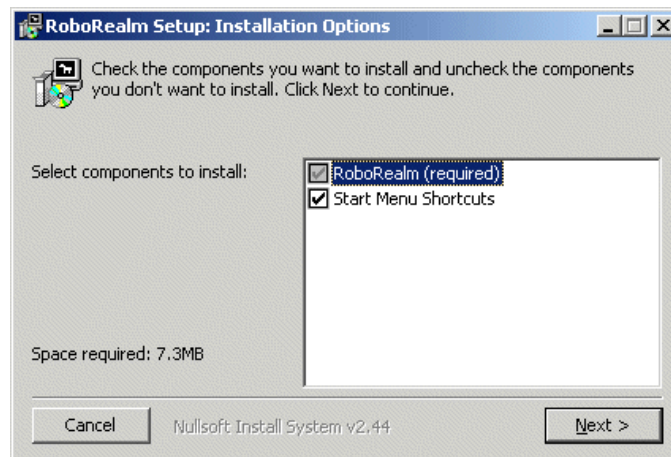




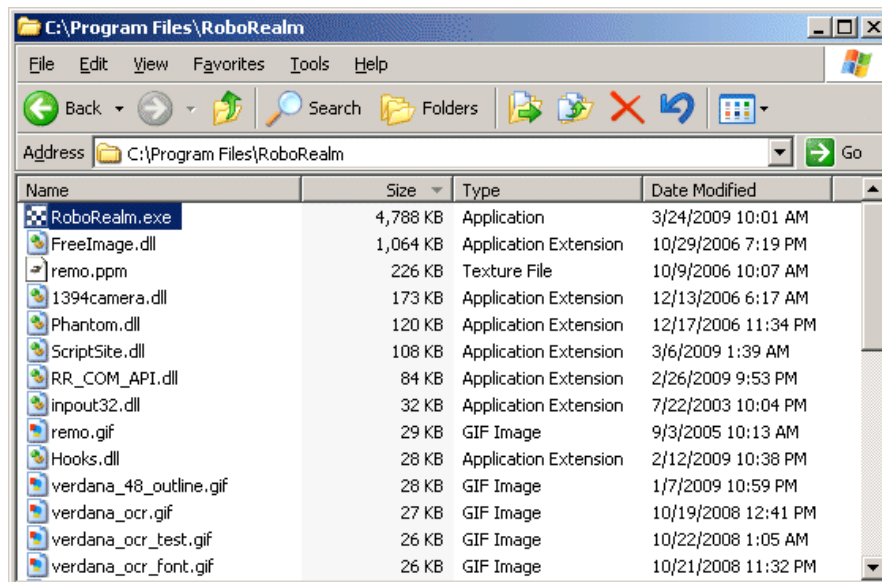
The best way to accomplish your processing task is to learn from the [tutorials](#) and post an example image in the [forum](#) where we and others can help provide meaningful modules that might accomplish the task.

Installation Instructions

1. You should have received an email containing a download link to the RoboRealm.zip application after completing the [email registration](#) process. Run the downloaded RoboRealm.exe file. This will launch the installation program.



2. Once installed Double-click on RoboRealm.exe in the installed folder or look for RoboRealm from you Window Start button, Programs menu, RoboRealm menu. Once you have done this you should see the main interface. If the application fails to execute you can try executing RoboRealm.exe while holding down the CTRL key. Holding down the CTRL key tells RoboRealm NOT to try to connect to any cameras or run any existing configurations which allows RoboRealm to start in a safe default mode.



3. If your webcam image does not appear in the main image area click on the Options Button->Video tab and select your camera in the dropdown selection. This should cause RoboRealm to start previewing the image in the main RoboRealm interface. Also be sure that the Camera button is pressed in order to activate the camera. If it is not, press the camera button and then use the Options button->Video tab to select the appropriate camera.

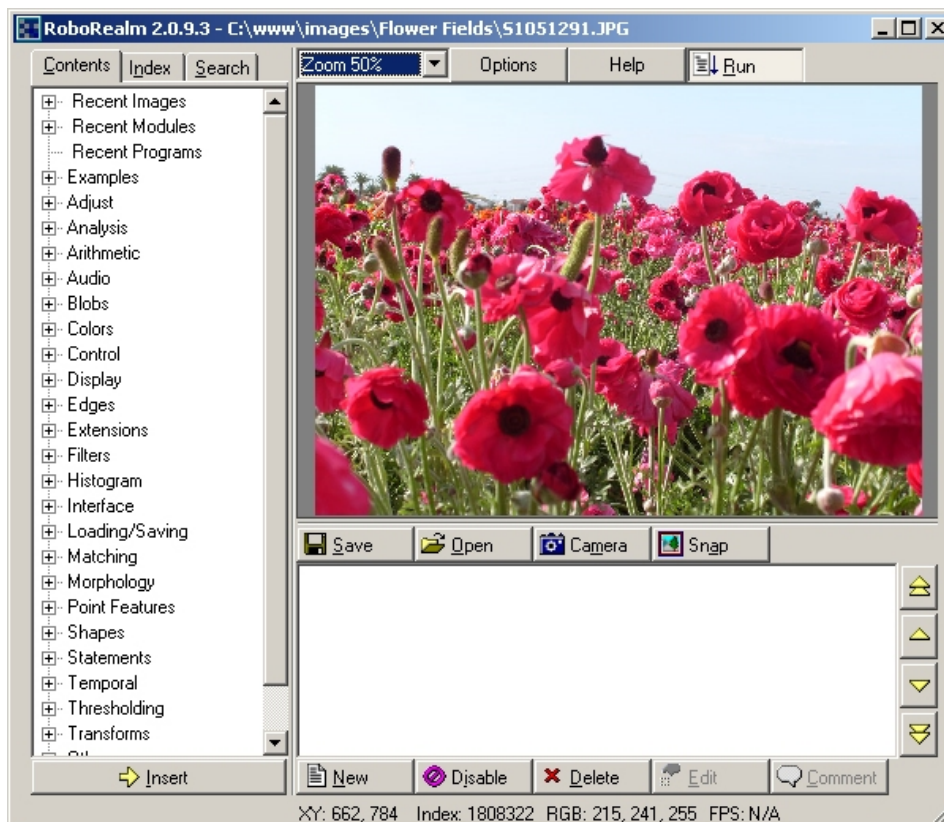
Hello World

Once RoboRealm is running and you have connected to and can see your webcam image you can now start inserting in modules that will affect the currently viewed image. You do this by using either the Contents, Index or Search tab in the upper left corner of the main interface. They all display the same modules but from a different perspective. Selecting a module and then pressing insert will insert that module into the processing pipeline.

(You can also double click on a module to automatically insert it into the pipeline). Once a module is added into the pipeline it will start to alter the image in some way to extract or highlight certain features (color, shape, texture, etc.) By adding in more modules additional attributes can be filtered

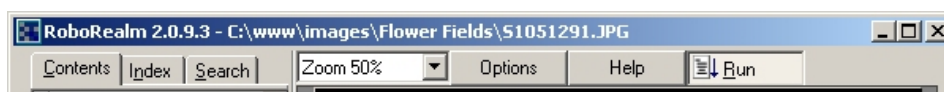
from the image more precisely in order to identify a particular object or image quality.

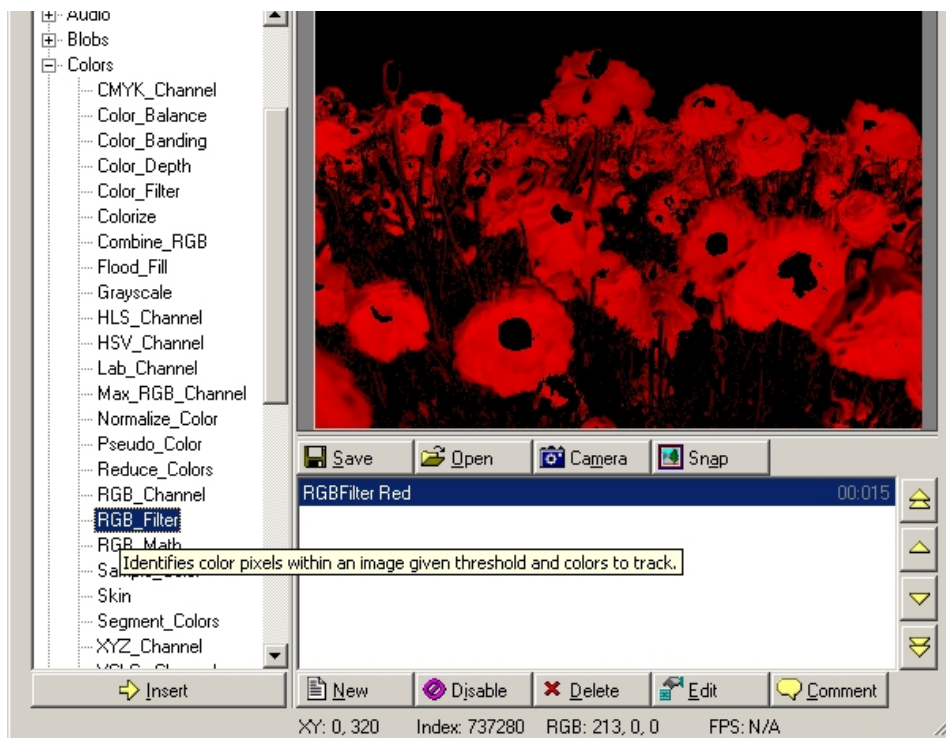
Note that you can also use [Firewire Cameras](#), [Internet Cameras](#), [Video files \(Media Reader\)](#) or just static images (you can drag static images into RoboRealm) as apposed to using a webcam.



The processing of images using the inserted modules happens as quickly as possible and is limited by the speed of the camera in use and the CPU requirements of the inserted modules. The sequence of processing an image starts with an image capture (for example a webcam) and then proceeds by passing the image to each of the inserted modules one at a time in the sequence they appear in the processing pipeline. At the completion of all modules the resulting image is displayed (the exception to this is if a module is selected in which case the image up to and including that module is displayed to aid in debugging the pipeline). Once this image is displayed the processing loop begins again with the next new image. Note that if the camera does not provide a new image to RoboRealm then RoboRealm will STOP processing and wait for a new image as reprocessing an already processed image will not add any new information into the system. Thus the pipeline with no inserted modules will run at the frame rate of the webcam. Adding a module will slow the looping speed a little depending on the processing requirements of the module. Different modules will have different CPU requirements as each module is different in terms of complexity and usage.

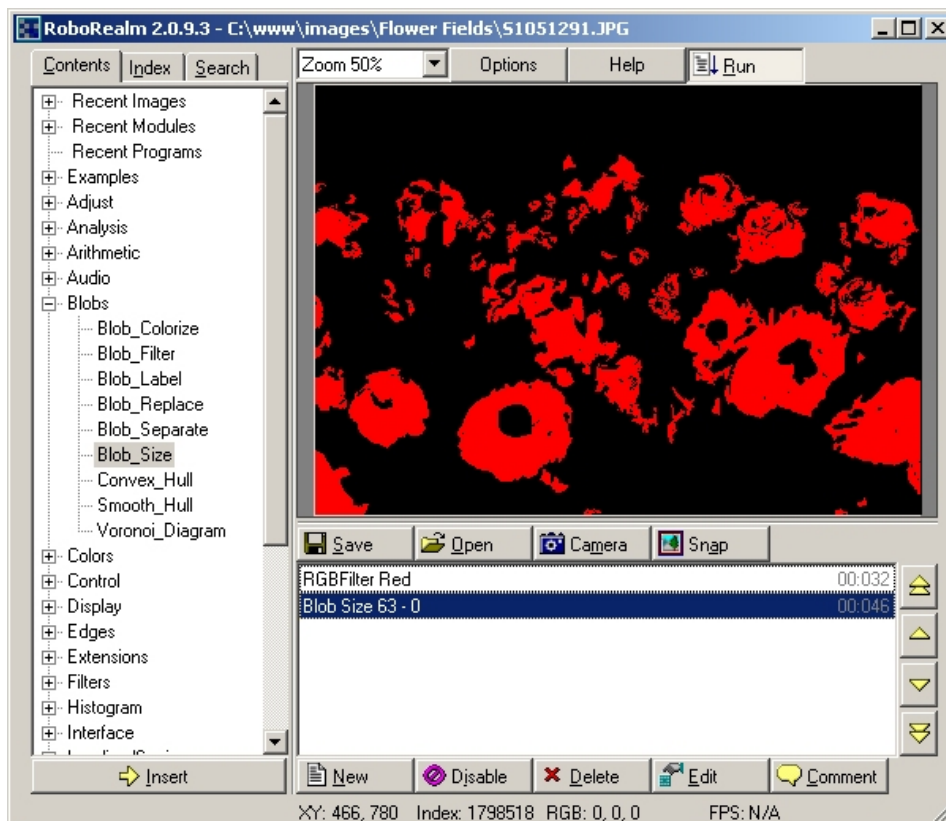
For example, adding in the [RGB Filter](#) module to the above image will cause the image to appear as






[Download .robo file](#)

and then adding a [Blob Size](#) module to remove smaller connected objects would result in



[Download .robo file](#)

Note that the order of the modules is relevant. If the [Blob Size](#) module is moved above the [RGB Filter](#) module the resulting image will not be correct as the initial image has thousands of colors that define individual pixels as apposed to connected blobs. You can explore these configurations by clicking on the  icon which will launch RoboRealm and allow you to change the configuration to see the effects.

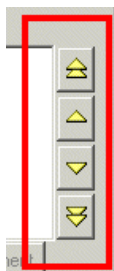
If you want to see the actual image without any modules changing the image untoggle the Run button in the upper right corner which will stop all modules from running and just show the image being captured from the imaging device.

You can continue exploring other robofile configurations by clicking on the "+Examples" branch in the Content tree view and double clicking on one of the example configurations. Ensure that you have a webcam on to see how the modules can change the image in realtime.

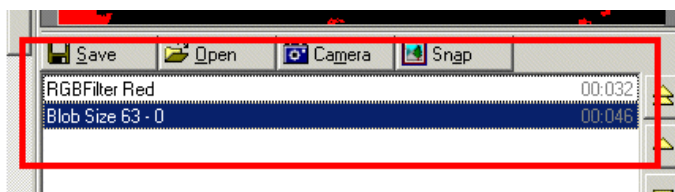
Working with modules

Most modules have an associated user interface that will popup once inserted into the pipeline. Changing values in those interfaces will immediately be reflected in the processed image. You can experiment around with changing the values to view what changes do what to the current image. In this way you can better understand what change in a value causes what change in the image.

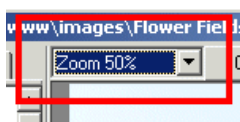
You can move the modules using the right side yellow move buttons. Note that most modules will react differently depending on where they are placed in the pipeline as the current image entering into that module will also be different.



Selecting a module will display the processing to that point in the pipeline. Using this technique you can understand what module is doing what to an image. Selecting the module again will deselect the module and show the entire processing of the pipeline.

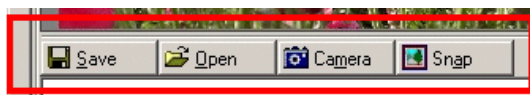


You can zoom into the image by selecting the zoom dropdown at the top of the main interface. When zoomed in you can click on the current image and drag the image to move it to another area as you would Google maps.



The Save and Open buttons allow you to save and load processing pipelines also known as RoboFiles. The same buttons serve to load and save images. You indicate which data you want to load and save by specifying an extension i.e. .robo for robofiles and .gif, .jpg, etc. for images.

The Camera button switches the webcam on and off. The Snap button will save the current image being viewed into a second window that can be further zoomed in or saved. This provides you a way to quickly snap an image for a more in depth viewing in case the camera image is changing rapidly.



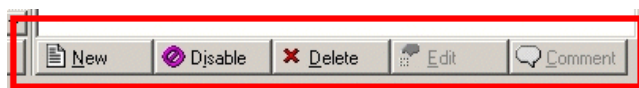
The New button removes all modules from within the pipeline and starts with a fresh empty configuration.

The Disable button will cause a specific module to be skipped for processing but remain in the pipeline. This is very useful if you are experimenting with several modules that create similar results but you are not sure which one to use and are comparing results. Disabling a module simply prevents it from changing or reacting to an image. Note that disabled modules are also saved as disabled in robofiles so be careful not to leave unintended modules disabled when saving.

The Delete button will remove the currently selected module from the processing pipeline.

The Edit button will cause a module's editable GUI to appear. This is the same action as double clicking on a module within the pipeline. Note that not all modules have an editable interface. For example the [Negative](#) module which inverts the current image does not have any parameters to edit and thus does not have an editable interface.

The Comment button allows you to associate a line of text with the currently selected module to aid in annotating the intended purpose of the module. Note that this can be used in conjunction with the Comment module which allows for longer annotations to be added into the pipeline.



Processing Pipeline

The processing pipeline displays all the modules currently active within RoboRealm. To view the original image without any processing unpress the Run button. When the Run button is activated the source image is captured, fed through each of the modules and then displayed. Once this has been accomplished the next image is captured and the process repeated. The processing pipeline can be thought of as a flow within a infinite loop of

been accomplished the next image is captured and the process repeated. The processing pipeline can be thought of as meters within a infinite loop of capturing and processing frames. Note that as more and more modules are added this loop can slow down as shown by the FPS (Frames Per Second) rate in the interface status bar.

You can always see what the a module is doing to an image by selecting that module with the mouse (1 click) which will cause the pipeline to only how the processed image to the point of the selected module. If you want to see the original image you can press the Run button which will stop all processing of any module and show what the original image looks like.

Tabs

RoboRealm has tabs above the processing pipeline which add additional capability to the application in terms of creating and reviewing processing pipelines.

The Main tab is always the first tab in the list and is ALWAYS processed/called for each image. To create a second tab press the [new] tab which will create another tab. This new tab can be used to create a new processing pipeline. Note that while this tab is selected the first Main tab will still be executing. Each tab runs in a global context in that variables are shared across all tabs (i.e. if you set a variable in one tab, another tab will see that change).

Tabs other than the Main tab are executed when:

1. You are currently in that tab (i.e. that tab is selected)
2. You've set it to executing using the tab properties (white button on the right of the tabs).
3. You are calling it from another tab using the [Call](#) module.

The Call module allows you to use the contents of a tab as if it were a function call. See the [Call](#) module for more information.

Images

During the processing of images several new images can be created and kept in memory using the [Marker](#) module. These saved images as well as other connected webcams can be accessed in certain modules (like the Marker) to restore the current image to previously saved images. In this way images can be moved into and out of the processing pipeline and saved for further processing. If you want to process more than one image at a time you can first process one image and then use the marker module to switch to another camera to process that image.

You can load in an image into RoboRealm in many ways. The most typically are using the Open button and specifying an image (or just dragging an image into the RoboRealm interface) or by pressing the Camera button and selecting an appropriate webcam. Other modules provide access to internet cameras, video files, etc.

Variables

As modules are added into the processing pipeline several RoboRealm variables are added into the variable table as needed. These variables are data units that are accessible from other modules and function as an information pipeline between modules or your custom scripts. See the **Variables** section of each module documentation page to see what variables are added by the module.

All variables currently within RoboRealm can be viewed using the [Watch Variable](#) module. Note that the placement of this module will reflect all variable values to that point. Thus the position of the Watch module may cause some variables to differ in values.

Several [default variables](#) are added once the pipeline begins execution.

Extending/Interfacing/Programming RoboRealm

RoboRealm offers many extension capabilities that allow you to utilize RoboRealm in your own applications.

[API](#) - RoboRealm offers a complete API that can be used to control RoboRealm from external applications. The API communicates over a socket based XML protocol so that it can be used to access RoboRealm on the same machine or over a network. The API can be used to acquire images, send images to RoboRealm for processing, access variables, set variables, etc. Note that while RoboRealm plugins are used to extend RoboRealm the API is used to control it. The two differ in terms of who is in "charge". With the API your application is in charge, with plugins RoboRealm is in charge.

[DLL](#) - The RoboRealm DLL specifies a shared library interface into RoboRealm modules. The DLL is meant to be incorporated into another applications that wish to directly utilize the RoboRealm modules. Note that the RoboRealm.dll is located within your RoboRealm installation folder (typically c:\Program Files\RoboRealm\)) and comes as part of the download link that you used to install the GUI application. Once you have access to this DLL you can copy it to alternative locations as needed. This is meant for advanced users that are very comfortable with low level Windows programming.

[VBScript](#) / [Python](#) / [CScript](#) - Embedded within RoboRealm as one of the native modules are the VBScript, Python and CScript modules. These

modules provide you with a simple scripting functionality to modify RoboRealm variables to perform tasks like image point to motor value mapping. These modules are used when small calculations or quick programming tasks need to be performed and is one of the easiest ways to customize RoboRealm while keeping your project contained within RoboRealm itself.

[Plugins](#) - New RoboRealm modules can be created and embedded in RoboRealm using RoboRealm Plugins which allow you to interface RoboRealm to a number of other languages like Java, C#, VB.Net, etc. in order to create new modules. These new modules can then be accessed from RoboRealm like any other module and serve to extend the filtering capabilities of RoboRealm.

[FTP](#) - RoboRealm provides an FTP module that allows you to send your images to remote machines or websites.

[HTTP Posting](#) - RoboRealm provides an HTTP module that allows you to send variable information from within RoboRealm to remote webpages. The module can also process text returned by that page back into RoboRealm variables. The HTTP tag extraction expressions can also be used to extract information from generic webpages to enable querying of generic information like stocks, weather, horoscopes, etc.

[Email](#) - RoboRealm provides an Email module that allows you to send your images and text to any email address.

[Socket Client/Socket Server](#) - RoboRealm provides a generic way of sending and receiving information over a network connection. These modules can be useful for communicating with your own socket based Clients/Server that have a specific custom protocol or one that is not supported by RoboRealm.

[OSC](#) - RoboRealm provides an Open Sound Control module that transmits RoboRealm variables to other applications via UDP. The OSC protocol is a simple data transmission protocol often used to transmit midi type signals between applications and devices.

[Modbus Slave](#) - RoboRealm provides a Modbus slave module that accepts modbus master requests over TCP/IP. The Modbus protocol is a simple data transmission protocol often used to transmit signals between PCs and PLC type devices.

[Write Image / Load Image](#) - The Write and Load Image modules can be used to save any image within RoboRealm to disk for other applications to process or load any image into RoboRealm for it to process.

[Write Variables / Read Variables](#) - The Read and Write Variable modules can be used to save any number of variables within RoboRealm to disk for other applications to process or read back information into RoboRealm from disk. (like Excel)

[Write Clipboard / Read Clipboard](#) - The Read and Write Clipboard modules can be used to read and write images from RoboRealm to the clipboard.

[Serial Port](#) - While there are many modules to communicate directly with known robotic hardware, RoboRealm also has a generic Serial module used to communicate to unknown serial devices and can transmit variables like image Center_Of_Gravity to those devices.

[Parallel Port](#) - In addition to serial communication you can also transmit information over the computer Parallel port in a bitwise manner.

[USB HID](#) - RoboRealm has a generic USB HID module that can be used to communicate to unknown USB Human Interface Devices and can transmit variables like image Center_Of_Gravity to those devices.

[Execute Command](#) - The Execute Command module can be used to execute a console based program and pass it variables to either process, save, or transmit. The Execute Command module will redirect standard input and standard output pipes normally used in console based programming to provide applications an easy way to access RoboRealm information.

[Virtual Camera](#) - Often RoboRealm is not the only program that needs access to the webcam. The virtual camera driver embedded within RoboRealm can be used to provide webcam images to more than just RoboRealm. The VCam can also be used to send other applications processed images from RoboRealm for further analysis or display in other contexts.

[Web Server](#) - RoboRealm contains an integrated WebServer that can be used to deliver images and variable values to any web service or receive HTTP requests from other systems to get and set variables.

If you do not find what you need please [contact us](#) and let us know.

RoboRealm Server API

- [Download Examples of RoboRealm API](#)
- [Starting the API Server](#)
- [Testing the API communication](#)
- [API parameters](#)
- [XML messages](#)
- [COM functions](#)
- [API Tips](#)

Introduction

INTRODUCTION

The RoboRealm Server API specifies a socket based communication protocol that can be used to remote control the RoboRealm Application. This API allows you to use your own program to command RoboRealm to load/save image, load/save programs, get/put images and perform any image analysis processing as you would using the RoboRealm application.

The RoboRealm API is different than the [RoboRealm Plugins](#) as the plugins are used to add new image processing modules into RoboRealm for use in any application/project. They also guarantee that each frame is processed by the plugin before execution is passed back to the image processing pipeline. The API is used for external applications to query the data within RoboRealm and/or manage the RoboRealm application remotely.

The RoboRealm API is used for situations where you have a main control program that is the central processor for all functions and you need to interface vision processing into parts of your program when needed. You can use the API commands to execute a sequence of image processing instructions which then return the results to your program. Your program would then continue based on those results.

There are API routines to manipulate variables and parameters. Variables are containers of information that are global within RoboRealm and can be accessed by scripting functions like the VBScript module using GetVariable function calls. You can see all current variables in RoboRealm using the [Watch](#) module. Parameters, are similar to variables but they specify a specific value within one of the GUI interfaces and are localized to that module. Thus if you want to set a value that you use in an IF statement or in the VBScript module use variables, whereas if you want to change the static number or text within the GUI interface of a module use parameters.

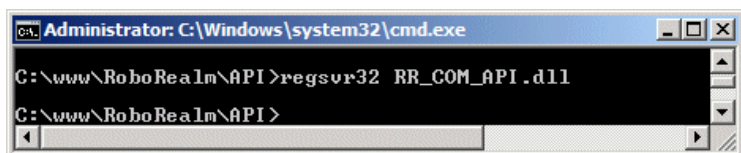
Note that some GUI modules allow for the [\[expression\]](#) format to be used in place of a number/text. In that case setting a variable will modify a GUI parameter too. This is the recommended usage when possible as variable modification is quicker than parameter changes as parameter changes need to interact with the GUI interface.

To determine what parameters to do what within a GUI dialog, insert that module within the RoboRealm GUI, save that pipeline using the save button to something.rob. Then load that something.rob into a text editor and you will see the XML tags that are the parameters to the module and what values they have. These XML names are the parameter names that can be used in the API to change those values.

[Download Examples of RoboRealm API](#)

The examples currently within the above zip download include C/C++, CSharp, Java, Python, Visual Basic (VB and .Net), Labview, Lisp, Matlab, Processing and WScript examples. While many of the examples include code in the languages native format to create a connection directly to the API server, it is recommended to use the COM object in all languages as the C/C++ and COM examples are the ONLY examples kept up to date. As new functions are added only the C/C++ and COM examples are updated to reflect these changes. Should you chose to use a native connection, you may have to refer to the [XML messages](#) to create messaging function that provides the needed functionality.

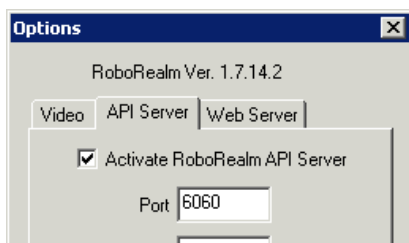
To use the COM examples **you must first register the COM object** using 'regsvr32 \path\API\RR_COM_API.dll' where path is the path you unzipped the API.zip file. This can be done by typing in the above line into the Windows Start Button->Run or by first opening a command prompt (aka cmd in the Start->Run program). Either CD to that folder or type the full path.



Special attention is needed for the Execute message as it is one of the most powerful but most generic function that accepts an XML image processing sequence string. The string will contain data that can be modified prior to sending the command sequence to be executed (using sprintf in C/C++). For example, you may want to change the filename in the example above as needed. The main difficulty of this routine is in creating the appropriate RoboRealm based XML string needed to accomplish a specific task. The easiest way to create these command strings is to run RoboRealm, configure the image processing pipeline using the provided GUI controls, save the program into a .rob file and then cut and paste the contents of the .rob file into the execute parameter. I.e. the execute function uses the same XML format as the .rob file. To determine the appropriate values for each of the modules you can use the same procedure and check what parameter values have changed.

Starting the API Server

The RoboRealm Server enabled/disabled from the Options dialog (click on the Options button from the main RoboRealm dialog window). For security reasons the RoboRealm Server is OFF by default. Clicking on the checkbox will active the socket listener and set RoboRealm ready to accept remote commands.





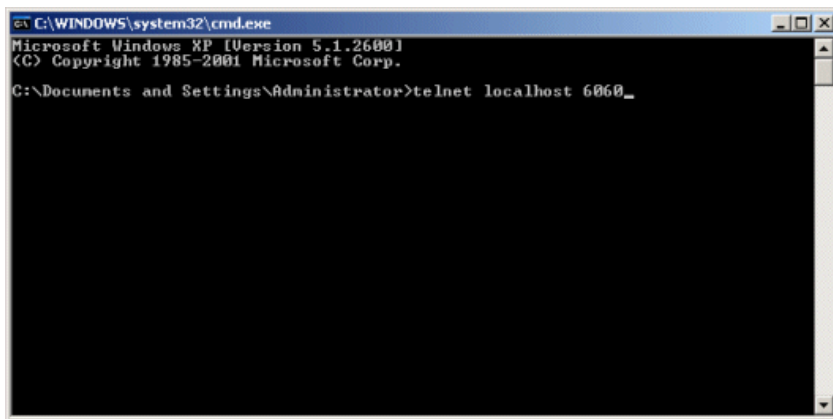
The API communicates to RoboRealm over network sockets. This means that RoboRealm can be remote controlled from a different machine located on your network. Note that this assumes that RoboRealm is already running as the API has no way of starting RoboRealm on a different machine than where the API commands are being executed. In this case you will want to place a link to RoboRealm in your windows "startup" folder.

Once you have started the RoboRealm Server (by clicking on the checkbox in the Options Button->API Server tab->Activate RoboRealm API Server checkbox) you can command RoboRealm from any socket based program given the specific commands.

Testing the API communication

For a quick test to see if your RoboRealm API Server is functioning correctly you can execute the following command from the windows command console to request the IMAGE_COUNT variable (the number of images that have been processed so far).

First open up a command console using Windows Start Button->Run->Type cmd and press OK. You should then see something like the following:



Within this window type:

```
telnet localhost 6060
```

and press enter. This should clear the screen and show nothing. Next copy the following text into your clipboard

```
<request>  
  <get_variable>IMAGE_COUNT</get_variable>  
</request>
```

and move back to the command console, right click and select paste. This will immediately send this text to the RoboRealm Server and return back an XML string with a single value of the image count (look for the value inbetween the <IMAGE_COUNT>X</IMAGE_COUNT> reply). You can then paste again to get another updated value.

For another example try pasting in the following which, assuming you have installed RoboRealm in the default location, it will load in an image.

```
<request>  
  <load_image>  
    <filename>c:\program files\RoboRealm\remo.gif</filename>  
  </load_image>  
</request>
```

Once this is done check the RoboRealm interface to see if it loaded in the specified image.

Note the usage of the "telnet" program which is a generic network connection program. The first line sets up the telnet program with the correct hostname and port number (RoboRealm Server runs on port 6060). The text you are pasting into that application is the XML command string that is interpreted by RoboRealm and executed accordingly. Note that the filename specification is relative to the machine that RoboRealm is running on and may be different that specified above. You can execute this sequence from any machine in your network AND from any operating system (including Linux) that has the telnet program installed.

The above text is a quick example of how the API can be used directly. We provide a complete source code examples written in C++, Java, C# (CSharp), VB.Net, Visual Basic, Matlab, LabView, Python, Lisp and WScript that implements the API socket communication. You can include this source into your own robot control program to add in the full vision processing capabilities of RoboRealm.

Testing Problems

If you have problems with the above example, the most likely scenario is that either the RoboRealm API is not running or you have a network restriction enabled. To check that RoboRealm is running ensure that you see the application running in your task bar AND that you have enabled the API Server (which is disabled by default) by checking the checkbox in Options Button->API Server Tab. Also be sure that the port specified in that interface is the SAME as what you are using in the example above.

If things still do not work (i.e. you see something like 'Connecting To localhost...Could not open connection to the host') then you probably have a firewall restriction that is preventing RoboRealm in opening and listening on port 6060. This is a non-known port so most likely your firewall does not like it. What you can try instead is to change the port to 80 and likewise in the API Server tab under the Port value. Changing it to 80 which is the port that browsers use may allow the firewall to agree to let information pass. Please note that if you are already running a webserver or have activated the RoboRealm Web Server this change will not be accepted. You may try port 8080 instead which may or may not work. If this fails you will have to check with your IT staff to determine how to allow your firewall to allow RoboRealm to run on a port.

API Parameters

When invoking RoboRealm.exe from your own application you have several command line options that you can use to force RoboRealm's behavior when it runs.

```
c:\RoboRealm\bin\RoboRealm.exe -api_port 1234 -new_instance -instance_2 -ini_file c:\temp\RoboRealm.ini -faceless
```

api_port - If you wish to run multiple copies of RoboRealm and use the API in each separate instance you will have to run RoboRealm with different API port numbers. Using -api_port 1234 will run a copy of RoboRealm using the API port 1234 instead of whatever is saved as the last configuration or the default 6060. Specifying the -api_port will also force the API server to execute regardless of the settings saved within RoboRealm's configuration. To force a new instance of RoboRealm each time you can use. This option can also be found in Options Button->API Tab in the GUI.

new_instance - Forces another RoboRealm instance to appear as apposed to defaulting to the already running instance. This option can also be found in Options Button->Startup Tab in the GUI.

instance_2 - Forces RoboRealm to run using the configuration of the second instance of RoboRealm. Whenever RoboRealm is run with more than one instance of the application the settings for that instance are separate from the first. This allows successive instances to run using the same settings as they did the last time they ran. Using the -instance_X command will allow you to pick which instance to run regardless of how many actual instances of RoboRealm you currently have running. Note that -instance_1, -instance_2, ... -instance_16 are valid parameters but only one is valid at a time.

ini_file - The ini_file parameter forces RoboRealm to read all configuration that is normally stored in the Registry from an ini file. This can be useful if you want to copy the configuration of RoboRealm and move it to another machine without having to reconfigure the application. Note that the instance_X parameter still applies in this case as each instance is also stored in the single ini file.

faceless - Forces RoboRealm to operate in faceless (without GUI) mode.

API Tips

- Think of RoboRealm like a server. Slow to start and stop but very fast to service requests. It is best to start RoboRealm at the beginning of your application, leave it running and then close when your application exits. This is what is done in the C# SampleApp and other continuous loops. RoboRealm is really meant to quickly process successive images so the speed is favored towards that and NOT to startup or stopping of the application.
- Load in one robofile at the start and use conditionals or tab calls to get different functionality. Loading in a robofile takes a bit and is not optimized. Again, RoboRealm does not expect the robofile to change from image to image so it is biased again to setup a new pipeline for successive image processing rather than quick loading.
- Keep in mind that CTRL-ALT-R when focused on the desktop will bring a faceless RoboRealm into view. Normally when doing

development using the API it is helpful to just open up the RoboRealm GUI and switch on the API to keep watch of what is going on via the GUI. This is helpful for debugging purposes and to catch mistakes.

- If your application starts RoboRealm but doesn't close it down correctly (i.e. you press stop on your application) it is possible for a second instance of RoboRealm to start which can cause conflicts. If you suspect communications issues, check your Task manager to ensure that only one instance of RoboRealm.exe is running.
- If you bring up the API tab in the Options button you can watch the request/response replies as they happen.

RoboRealm DLL

[Download Examples of using the RoboRealm DLL](#)

The RoboRealm DLL specifies a shared library interface into RoboRealm modules. The DLL is meant to be incorporated into another applications that wish to directly utilize the RoboRealm modules. Note that the RoboRealm.dll is located within your RoboRealm installation folder (typically c:\Program Files\RoboRealm\)) and comes as part of the download link that you used to install the GUI application. Once you have access to this DLL you can copy it to alternative locations as needed.

The RoboRealm DLL is meant for advanced users that are familiar with working with low level functions and integration of Windows applications. We do not recommend the DLL interface for those that are unfamiliar with Windows shared libraries. Instead, use either the [scripting modules or the API](#) which are simpler to utilize.

Theory

Like the RoboRealm API, the DLL specifies calls into the DLL to perform various functions. Please note, the DLL AND the API have exactly the same functions! Thus, we recommend you start using the API instead as it is easier to determine what is happening within RoboRealm with the GUI exposed. When you are ready, switching to the DLL calls is a simple change. This symmetry between the API and the DLL function provides the ability to leverage which technique is most appropriate for your application with minimal changes if you decide to switch.

Virtual Camera Driver

The RoboRealm Virtual Camera Driver provides a way to utilize the processed image results from RoboRealm in up to 5 other applications that supports frame grabbing from WebCameras or other capture devices. This allows you to interface any camera that RoboRealm can connect to as a regular webcam device to other applications.

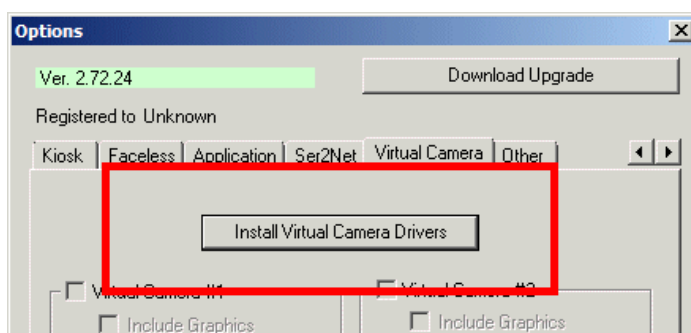
This image relaying is accomplished by installing the RoboRealm Virtual Camera (VCam) drivers that will create new webcam drivers in your system that can be connected to like a normal webcam. This setup works as follows:

1. RoboRealm is running and grabbing images from the actual webcam driver
2. RoboRealm process the image
3. RoboRealm then send its processed image to its VCam driver (ensure that the Options->Virtual Camera->Virtual Camera #1 is set)
4. Your application connects to the RoboRealm VCam driver and grabs the processed (or original) image

Note that this functionality can also be used to provide the source image from a single camera to both RoboRealm and another running application that requires a webcam. This can be accomplished by the same setup as above by ensuring that the processed image in RoboRealm is unchanged from the source. You would specify the "source" image within the Virtual Camera Options tab. Likewise, if you wish to feed a particular image to other capture applications you can accomplish this by setting a Marker module within the pipeline and specify that image in the "Use Image" dropdown menu.

Opening single static images or playing movie files in RoboRealm will also cause the VCam to update too.

Installation



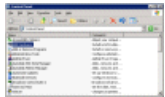


To install the Virtual Camera Drivers proceed to the Options Button->Virtual Camera tab->Install button. This will install or replace the current drivers for the Virtual Camera. Please note, this requires admin permissions and will take a while to complete ... so please be patient. Once done, you can enable the virtual cameras by selecting the checkboxes below the install button.

IF for any reason this installation fails, the install process below is provided to install the drivers manually.

Windows XP Installation

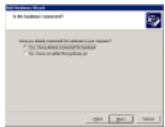
As the VCam driver is not actually attached to a physical device the installation is a more manual process than hardware based plug and play drivers. The following outlines the installation steps for Windows XP. (Windows 2000 and Vista have similar installations).



1. Click on Windows Start Button -> Select Settings -> Select Control Panel. In the Control Panel click "Add Hardware"



2. Click Next on Add Hardware Wizard.



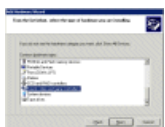
3. Select "Yes, I have already connected the hardware". Again, as no hardware is actually connected to the VCam driver you need to manually tell the system where to locate the drivers.



4. Select "Add a new hardware device". Then select Next.



5. Select "Install the hardware that I manually select from a list (Advanced)". Then select Next.



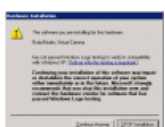
6. Select the hardware type "Sound, video and game controllers". Then select Next.



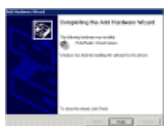
7. Click on the "Have Disk" button. Note that you will **not** find RoboRealm in the Manufacturer list nor the VCam in the Model list.



8. Once the Browse dialog pops up select the RoboRealm folder where you unzipped/installed the RoboRealm application (you can target the RoboRealm.inf file which is what the system is currently looking for). All the needed files are in that folder. Once selected you should see "RoboRealm, Virtual Camera" in the Model list. Select Next to continue.



9. You will see some form of verification screen that claims the Driver has not passed "Windows Logo testing". Click "Continue Anyway" or "Ignore" to continue.



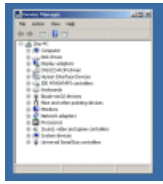
10. The system will now copy files into the appropriate location. You *may* get another popup searching for the RoboRealm.sys file. If so, again, select the same RoboRealm folder that you unzipped/installed RoboRealm in. Once all files are copied the finished installation screen will appear. Click "Finish" to end the installation. You are now ready to run the RoboRealm VCam!

Be sure to have enabled the VCam capturing (Options Button->Virtual Camera->Virtual Camera #1) and installed the VCam driver (see steps

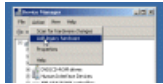
below) before attempting to use the VCam functionality. Be sure to have at least DirectX9.0 installed.

Windows 7/Windows 8 Installation

Windows 7 (especially the 64bit version) requires that all kernel drivers be signed. You will get a popup asking if you trust RoboRealm, LLC. You should accept that (it's us) to continue the driver installation.



1. Click Windows Start Button->Control Panel->Device Manager to Open up the Device Manager.



2. Within that Window click on the Computer and then on the Add Legacy hardware menu. Note that the menu option will NOT appear until you select the Computer subtree.



3. You should now see the Add Hardware GUI. Click Next.



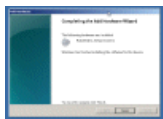
4. Select the "Install the Hardware that I manually select from a list (Advanced)".



5. Ensure that the Show All Devices option is selected and click Next.



6. Click on the Have Disk button and specify the path where you installed RoboRealm. Press OK and then Next.



7. Click on Next and finally Finish once the Driver has been added.

IF things do NOT work on Win7 and you have a yellow exclamation mark next to the VCam drivers in the device manager that's most likely because you've not updated your Windows system with a SHA256 hash fix for drivers. This update came out in Oct 2014 and can be read about [here](#). Our drivers use SHA256 (since SHA1 will be invalid starting 2016) and thus require that Win7 update to fix this issue. Note, Windows 8 already has this update and need not be updated.

Testing

1. Open AmCap or other webcam viewing software
2. Select "RoboRealm VCam" as the driver or image source. The image will appear black or gray until RoboRealm is run
3. Run RoboRealm. Ensure that the Activate VCam checkbox in the Options Button->Virtual Camera tab is set
4. Connect to your actual hardware camera using RoboRealm.
5. You should now see your webcam image in RoboRealm and AmCap.
6. For optimal speed ensure that the AmCap Capture Pin size is the same or similar as to what RoboRealm is capturing otherwise RoboRealm will resize the image before sending it to the VCam driver which can add to performance delays.

Troubleshooting Tips

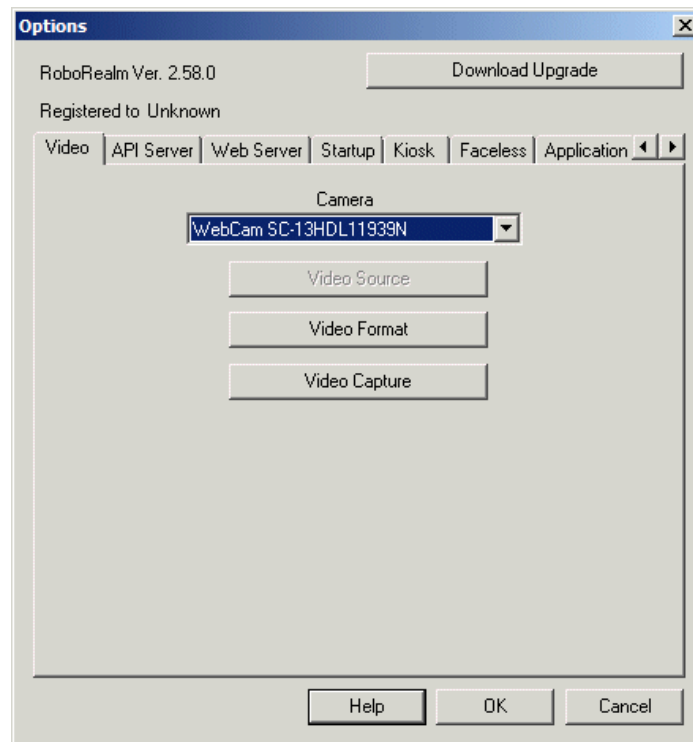
1. Did you install the VCam driver?
2. Did you click on Options Button->Other Tab->Activate RoboRealm VCam check box in the RoboRealm application?
3. Do you have at least DirectX 9.0 installed? See [Microsoft DirectX End-User Runtime](#) for installation.
4. If you are not running any hardware cameras RoboRealm may startup by connecting to its own VCam driver. In this case unclick the Camera button and drag or open an image into RoboRealm. Exit RoboRealm to ensure that on startup it will only load in the image instead of connecting to

RoboRealm Options Page

- [Video Tab](#)
- [API Server Tab](#)
- [Web Server Tab](#)
- [Startup Tab](#)
- [Application Tab](#)
- [Kiosk Tab](#)
- [Faceless Tab](#)
- [Ser2Net Tab](#)
- [Virtual Camera Tab](#)
- [Other Tab](#)

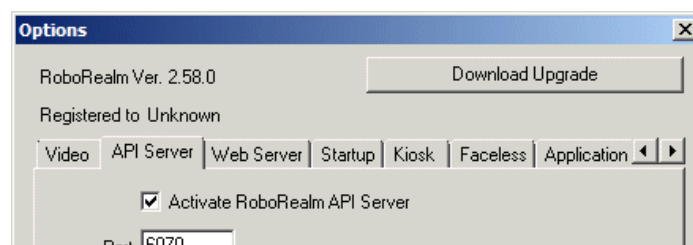
The options page provides access to the configuration information for RoboRealm and is broken into 4 tabs, Video, WebServer, API Server and Other.

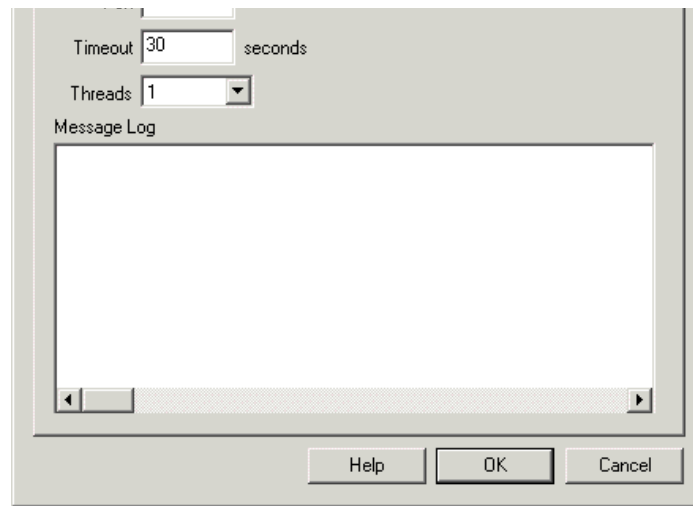
Video Tab



- Camera - Provides a list of recognized cameras currently available on your computer. You can select any of the items in that list to switch to that camera and begin processing its video. Note that this list will appear in other RoboRealm modules to allow you to access multiple cameras at a time. For Firewire cameras see the [Firewire Camera](#) module. For Internet IP cameras see the [HTTP Read](#) module or the [DLink Internet Camera](#).
- Video Source/Format/Display/etc - Once a camera is selected the appropriate buttons will be enabled that provide access directly to that cameras driver information and allows you to configure low level information (such as video format) that the camera and camera driver will produce. The most relevant is Video Format which will dictate what format (RGB32, YUV, I420, etc) that is passed back to RoboRealm. RoboRealm supports many formats but may not support your exact format. If you receive a format error message box try to change the format to RGB32 or other format that RoboRealm supports.

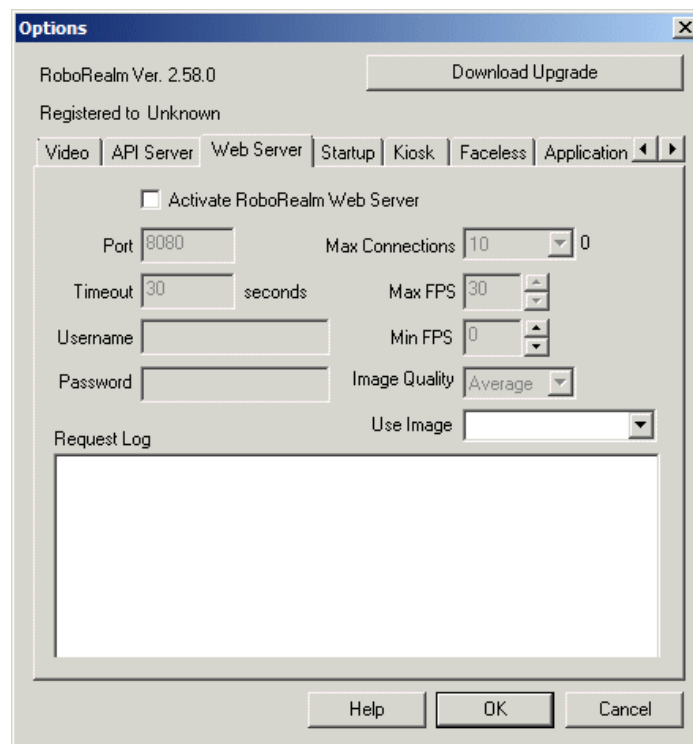
API Server Tab





- Active RoboRealm API Server - by default RoboRealm does not activate the API server port. To communicate with RoboRealm using the API this checkbox must be selected.
- Port - the network port that is used to communicate to RoboRealm. The default is 6060.
- Timeout - the connection timeout that is used to wait for new commands. If RoboRealm does not receive any commands within the specified timeout period it will automatically terminate that port connection.
- Threads - specifies how many simultaneous connections the API server will handle. If you want more than once machine to connect to RoboRealm switch to a value higher than the default 1.
- Message Log - a log of the interactions that a remote program is sending the RoboRealm API server.
- Be sure to see the [API documentation](#) page for more information about interfacing with RoboRealm.

Web Server Tab

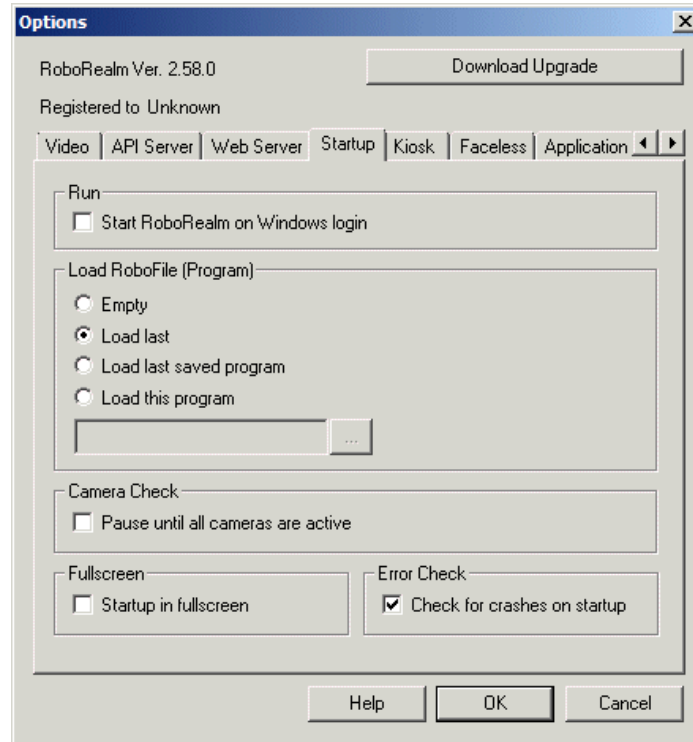


- Active RoboRealm Web Server - by default the embedded webserver in RoboRealm is no on. You need to select this checkbox to enable the webserver.
- Port - the webserver network port that RoboRealm will use. The default is 8080. Note that if you change this port you will need to also change the url used to connect to the RoboRealm webserver too. The port in the url is located inbetween the hostname and before the first path specification.
- Username - to provide secure access to the webserver you can specify a username that must be typed in (the user will be prompted) in order for the video stream to be accessible.
- Password - the password required for access. Default is blank which means anyone can view the video stream without requiring authentication.
- Max Connections - specifies how many simultaneous connections the Web server will handle. If you are expecting many users to connect via the web to a value higher than the default 5.
- Max FPS - the maximum number of frames per second that the webserver will send to clients. This setting can help you decrease your bandwidth requirements.
- Min FPS - the minimum number of frames per second that the webserver will send to clients. If your viewer requires a set number of frames

per second this can help ensure that the rate is maintained even if new images are not present (old ones are sent to maintain the connection).

- Image Quality - allows you to specify how good a quality image is sent out to connected browsers. Setting this to Worst will send the smallest image but will affect the image quality. Setting it to Best will send a very high quality image but will use a lot of bandwidth.
- Use Image - By default the webserver will send back the processed image that is seen at the end of the pipeline. Sometimes this is not desired so you can select which image or which marked image (via the [Maker](#) module) should be send via a web server HTTP request.
- Request Log - a web browser request log to monitor who is connecting to the RoboRealm webserver.
- See [RoboRealm WebServer](#) for more information.

Startup Tab



- Start when Windows starts - Select if you want RoboRealm to startup without having to first login into the computer. This is very handy when running RoboRealm automatically and it should startup when rebooted.
- Load RoboFile - by selecting a load option you can change the way RoboRealm first starts up. By default "Load last program" is set.

Empty - startup RoboRealm with a new/empty/clear pipeline program

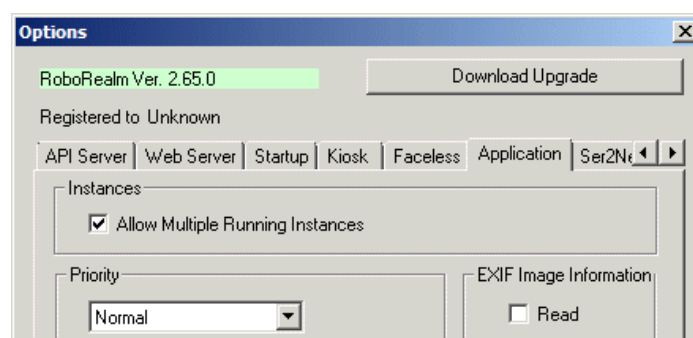
Load last program - loads in the very last pipeline program that you were working with. This configuration is saved to a working. robo file at the last time you exited RoboRealm.

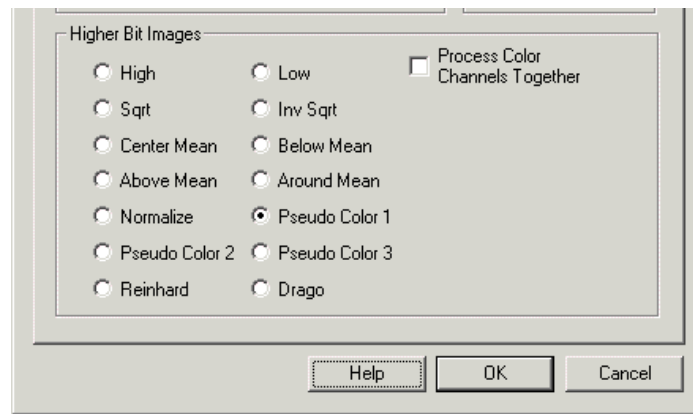
Load last saved program - loads in the last program you saved. This will overwrite any changes to the pipeline you were last working on.

Load this program - loads in the program specified in the below text box. This is a great way to ensure that regardless of what you do within RoboRealm that it always starts up with the same pipeline program.

- Camera Check - causes the execution of the processing pipeline (program) to wait until all cameras specified in the pipeline are active and producing images. A popup will occur that will indicate that a camera is missing which will then disappear when the camera is found or when the user clicks to ignore this issue. This is very useful when first starting up the operating system where RoboRealm may be run before all the cameras have had a chance to initialize. This can cause instability in the system and cause control issues on the robot during startup.

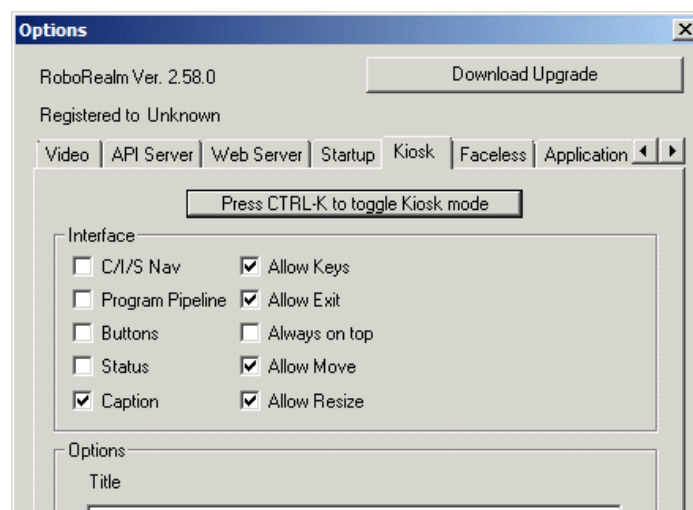
Application Tab

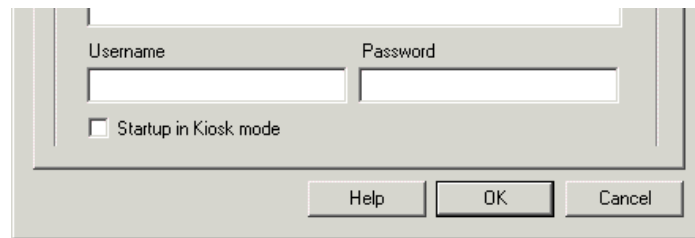




- Instances - by clicking on "Allow Multiple Running Instances" you can have more than one copy of RoboRealm active within your desktop. By default anytime you double click on a .robo file the currently running RoboRealm application will focus and execute that file. This option overrides that behavior.
- Priority - You can select what priority RoboRealm uses when running. Higher priorities will give RoboRealm more CPU time but remove it from other applications. Lower priorities will force RoboRealm only to run when there is available processing power (idle time).
- EXIF Info - Select if you want to load an image's EXIF information into the RoboRealm variable table.
- Higher Bit Images - RoboRealm operates on 24 bit RGB for performance reasons. When loading in images with higher bit depths RoboRealm needs to know how to process the image into a 24 bit image. There are many ways this can be accomplished:
 - High - Uses the upper 8 bits of the image
 - Low - Uses the lower 8 bits of the image
 - Sqrt - Square root's the image pixel to the 8 bit range
 - Inv Sqrt - The Sqrt function will favor darker pixels, the Inv Sqrt favors lighter
 - Center Mean - Forces the high bit range to be centered at the image mean
 - Below Mean - Shows only pixel below the image mean compressed into 24 bits
 - Above Mean - Shows only pixel above the image mean compressed into 24 bits
 - Around Mean - Similar to Center Mean but thresholds values on either side to improve contrast
 - Normalize - Determines image low and high values and scales to 24 bit
 - Pseudo X - Translates image intensity into a higher color range for improved visibility
 - Reinhard/Drago - High Dynamic Range reduction techniques as specified in the FreeImage.dll
 - Logarithm - Takes the logarithm of the image pixel and then stretches to 8 bit range
- Process Color Channels Together - Specifies that color channels will be considered a single channel such that the relative color amounts will not change. Unselected, each color channel will be processed independently which can improve or worsen an images overall appearance.

Kiosk Tab

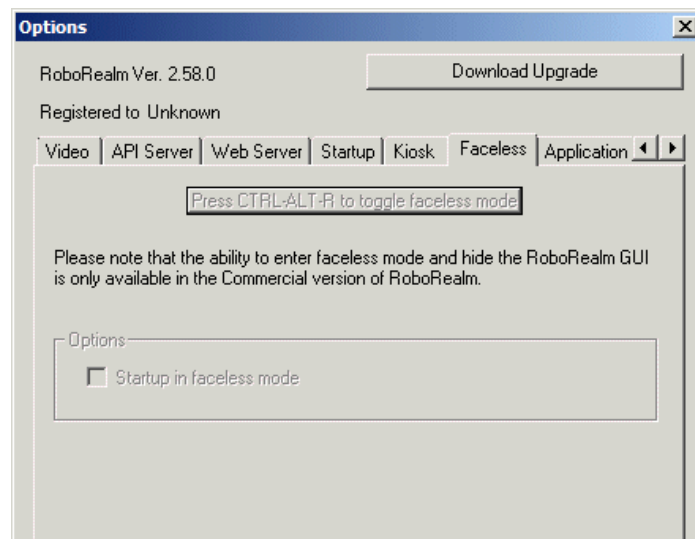


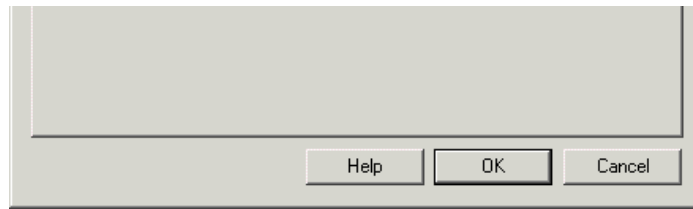


The Kiosk mode provides a way for you to run RoboRealm without all the controls that are used to modify and create RoboRealm programs. The Kiosk mode provides a customized view (can be as simple as just the video feed) to allow just those parts of the program to be available to the end user. This is very useful when running RoboRealm in environments where you would prefer no one has access to reconfigure the running program. It is worth noting that for an insecure system the Kiosk mode username/password can be broken by deleting the regedit entry for the RoboRealm program. To ensure a fully secure system you would have to disable user editing of the registry.

- "Press CTRL-K to toggle Kiosk mode" - by clicking this button you will immediately enter into the Kiosk mode of RoboRealm. Note that at any time you can toggle between Program and Kiosk mode by pressing CTRL-K.
- Contents/Index/Search - to show/hide the module navigation on the left side of the main RoboRealm interface toggle this checkbox
- Program Pipeline - to show/hide the program listing seen below the video image and all the edit buttons that are related to the program mode toggle this checkbox
- Buttons - to show/hide the zoom, option, help and run buttons toggle this checkbox
- Status - to show/hide the status bar seen at the bottom of the interface that includes the current mouse location, pixel color at that location and the current fps toggle this checkbox
- Caption - to remove the blue title caption seen at the top of the interface toggle this checkbox. Note that without the caption users will not be able to move, resize or quit using the upper rightmost buttons.
- Allow Keys - if this checkbox is unchecked the user will not be able to use any keyboard shortcuts like Alt-O to open files, etc. Normally if you disable the user from using buttons you will probably want to prevent them from using keyboard shortcuts too.
- Allow Exit - if this checkbox is unchecked the user will not be able to exit the RoboRealm application by clicking on the upper rightmost X button, using ALT F4, or by right clicking on the caption bar.
- Always on top - when enabled this checkbox will ensure that the RoboRealm window is always on top of other windows.
- Allow Move - allows the user to move the window by dragging the caption area
- Allow Resize - allows the user to resize the application interface using the corner stretching and shrinking. Note that when disabled this mode also disables minimization.
- Fullscreen - shows just the image area in fullscreen mode. Note, in this mode none of the GUI window is shown.
- Authentication - to prevent users from breaking out of Kiosk mode by pressing CTRL-K you can specify a username and password that are required in order for Kiosk mode to be broken. When these are set a Login prompt will appear when exiting Kiosk mode to ensure that the current user is knowledgeable enough to do so. Note that if you forget your username/password you will have to remove the RoboRealm registry entry. See the [FAQ](#) for how to do so.
- Startup - select if you want RoboRealm to jump into Kiosk mode once the program is activated. Note that if you start RoboRealm in Kiosk mode, have a username/password to break out of Kiosk mode AND are holding down the CTRL key you will be requested for that login before the system resets to the default settings. This prevents anyone without the username/ password from resetting the system without that knowledge.

Faceless Tab





The faceless mode of RoboRealm provides a way for the RoboRealm application to function while being invisible to the user. More can be read about this mode in the [FAQ](#). This faceless mode can be entered using several techniques:

1. Specify the command line argument "-faceless" (without the quotes like -api_port) when running RoboRealm. This can be added either when running RoboRealm from your desktop or included in the startup process as seen in the C++ API. Included in the RoboRealm download is an example file called RoboRealm_Faceless.bat that shows what this technique looks like.

```
c:\Program File (x86)\RoboRealm\RoboRealm.exe -faceless
```

2. The Options button -> Faceless tab interface can be used to enter into the faceless mode (by pressing the top button) and also set the faceless mode on startup. When specifying the faceless mode on startup RoboRealm will run without an interface unless the "-display" command is provided in the command line (the -faceless command is not needed in this mode). This startup checkbox will ensure that each time RoboRealm is started it will not show any GUI interface.
3. You can send the API request [faceless](#) to hide the [display](#) request to show it.
4. Press the key combination of CTRL-ALT-R to toggle the interface display/faceless on and off.

Ser2Net

The ser2net tab provides a way to configure remote serial ports (hostname and port) that exist on another machine but can be accessed as if local on a machine running RoboRealm. This provides a flexible way to control remote serial devices on embedded systems that are not capable of running RoboRealm.

Why is this a big deal? If you happen to have any of the recent embedded processors like the Raspberry PI, BeagleBone Black, PCduino, TP-Link Router, etc. you can use these to provide a wireless connection to your laptop and control a serial device from RoboRealm.

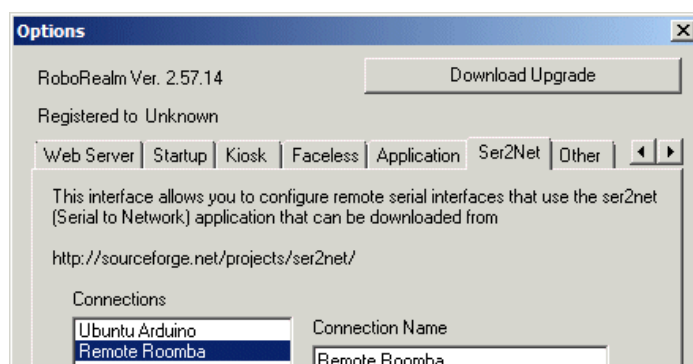
ser2net NEEDS to be installed on the remote machine and configured as appropriate. ser2net can be [downloaded here](#). Once installed you need to configure it to access the local port. Note that all parameters (baud, stop bits, etc.) MUST be specified on that machine. For example, the following will run ser2net with a configuration appropriate for an Ubuntu installation to access an Arduino microprocessor. In this case, it is a TP-Link router that connected to an Arduino using a USB cable.

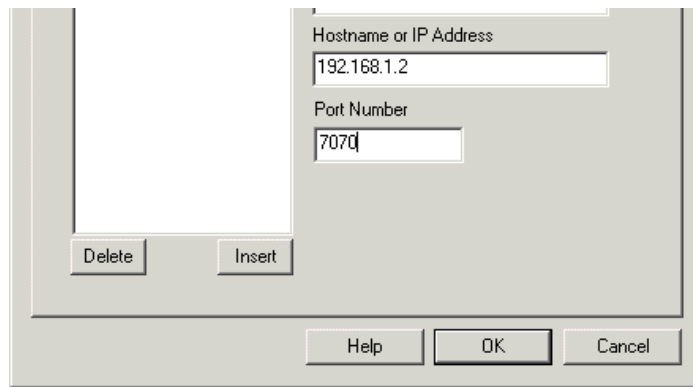
```
sudo ./ser2net -C "7070:raw:1000:/dev/ttyACM0:115200 NONE 1STOPBIT 8DATABITS XONXOFF LOCAL"
```

Note the use of sudo which ensures that ser2net has access to your serial port (if you know your current user has access you can remove the sudo command). Also note that the above example uses /dev/ttyACM0 as the serial port. This may be different on your machine. Finally the baud rate is set to 115200 which is what the default Arduino sketch used for the Sparkfun_Arduino module expects to use.

Connecting your device to an embedded processor STILL requires the needed drivers to be loaded by the device. In most cases, a generic serial connection will suffice but it is very possible that your embedded processor fails to load the needed drivers which would normally load without issue within a Windows environment. The only option you have at that point is to contact the vendor of the peripheral device and request drivers for your appropriate embedded device.

Once configured, the name that you specify will appear in the COM Port dropdown in those modules that require a serial interface to operate. By selecting this name, the connection will automatically connect to the remote processor and appear as if the connection were done locally. Thus by simply changing the port specification in a module you can now operate the device remotely (assuming ser2net has been configured on the remote processor).



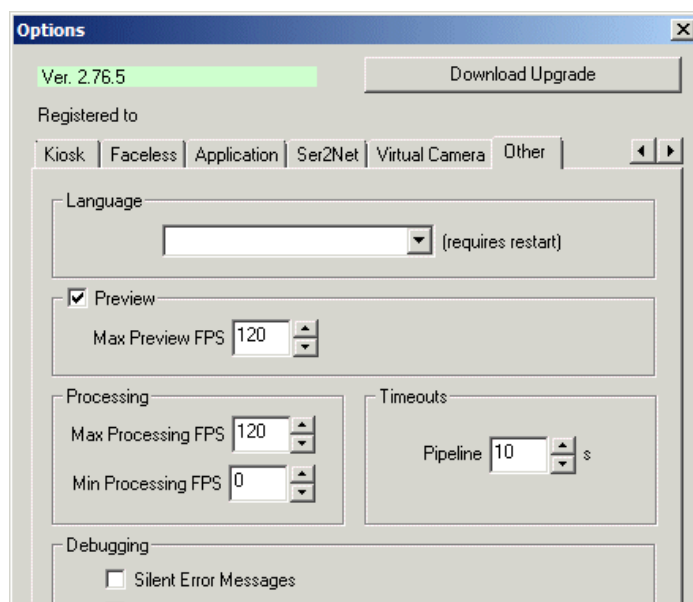


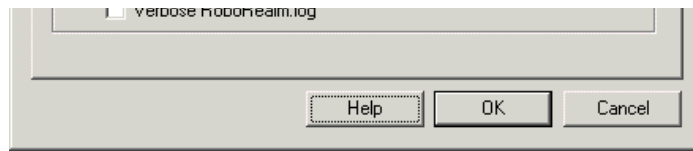
- Connections - Shows the list of currently configured remote connections. Clicking on a connection name will populate the hostname and port number to the right of that list.
- Connection Name - The name you chose to call the specific configuration. Its best to use something descriptive.
- Hostname or IP Address - The hostname (i.e. www.roborealm.com) or IP Address (i.e. 192.168.0.2) of the REMOTE machine (i.e. Raspberry PI, BeagleBone, TP-Link, etc.) that RoboRealm will connect to in order to use its local serial connection.
- Port Number - The port number of the TCP connection to the above hostname that is waiting for a connection that is tied to a serial port. Default is 7070 which is specified in the above configuration line for ser2net. As long as your configure this number in both ser2net and in this RoboRealm tab, you can use any number you'd like. Note that ports below 1000 in most Unix systems require root privileges.
- Delete / Insert buttons - Delete currently select Connection or Insert a new one.

Virtual Camera

To read more about the Virtual Camera capabilities within RoboRealm please see [Virtual Camera Driver \(VCam\)](#) page for more information.

Other Tab

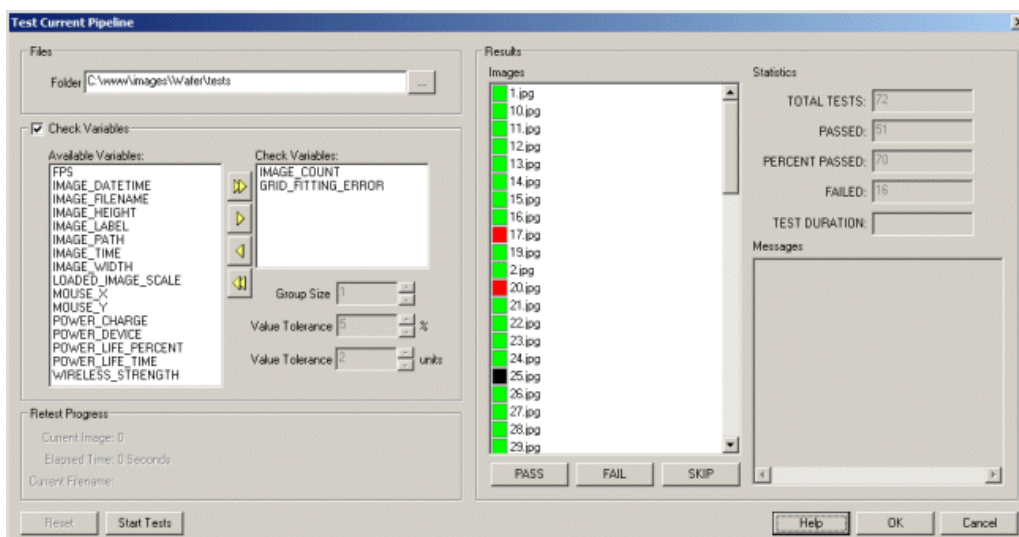




- Language - Select which language you would like the main RoboRealm interface to use. When you restart the application, your selected language will be used for the main interface text AND module tooltips.
- Preview Checkbox - to disable viewing the processed video unselect the preview checkbox. This will allow RoboRealm to continue to process and react to visual images BUT will stop display of those images on your computer screen. This can help to increase processing performance as the rapid display of images using a slow graphics card or over the network can limit the total processing speed of the system.
- Max Preview FPS - instead of completely disabling preview you can also modify the FPS to slow the preview rate down. For example, typing in 1 will limit the preview of processed images to a maximum of 1 per second. If you type in 0.5 that will limit the update rate to 1 frame every 2 seconds and so on. Note that this does not limit the number of frames that are being processed but just how many of those processed frames you are viewing.
- Max Processing FPS - if you want to slow the processing of images down modify the FPS to something slower than the typical 30 fps. For example, if you specify 5 then RoboRealm will process a maximum of 5 frames every second or 1 frame every 0.2 seconds. Note that your preview rate will also be affected as the preview rate will decrease with a slower processing rate. You can also use decimal numbers such as 0.25 which will cause RoboRealm to process 1 frame every 4 seconds.
- Min Processing FPS - if you want to ensure that the processing of images happens even if a new image is not available you can set the Min Processing FPS to some number above zero. This will ensure that RoboRealm does not wait too long for a new image and instead run the pipeline again with the same image. This is useful if you have some form of polling occurring in the pipeline that needs to run regardless of if any new images are made available.
- Silent Errors - As RoboRealm is often used as a server process it is not desirable for error messages to cause a popup that requires user interaction in order for the application to continue functioning. Once this checkbox is selected all errors are saved in RoboRealm.log instead of causing a popup to occur. The RoboRealm.log file will be created in the current folder where RoboRealm is being executed. To warn of this event a red square text area in the main GUI window will appear for 15 seconds to remind you that errors are being logged instead of alerting via popups. Note that this red square will NOT be part of the image being processed and is only displayed as a reminder.
- Verbose RoboRealm.log - Often if you have a particular problem when using RoboRealm we will ask you to turn on extra debugging that helps us understand where the application is failing. When using the Verbose RoboRealm.log at lot more statements are added to the RoboRealm.log file in order to help us determine where things are failing. You need not use this checkbox unless directed by us. Please note that this checkbox is cleared each time the application is restarted.

RoboRealm Test Page

The RoboRealm Test page provides a way to test multiple static images in a pipeline while reporting any differences from previous tests. This procedure allows for an automated way of regression testing against a large number of images without having to manual ensure every image results.



The process for this interface is to first select a folder that contains many images. Once specified these images are listed in the clickable interface on the right. You then specify what variable needs to be checked for each image. This variable would be generated by your current pipeline, i.e. maybe

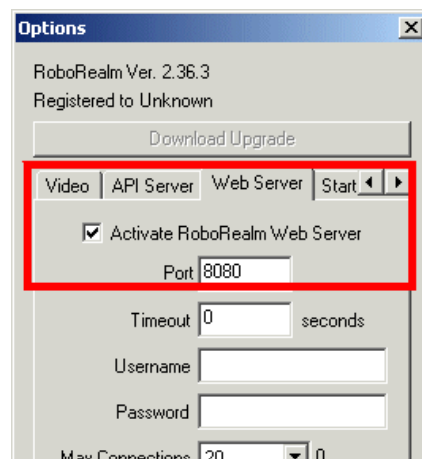
a variable that holds the amount of Red color in an image. The specific variable and value within would be dependant on your current pipeline. You then proceed through all the images starting with the first by pressing PASS or FAIL. These buttons are located at the bottom of the image list. Pressing PASS will record the variable results as being positive, FAIL as negative and SKIP as ignored. Once the entire list has been manually reviewed once, you can then press the Start Tests button to automatically reprocess the Result Images in the current pipeline while comparing the variable results to the previous recorded results.

For example, if you used the [RGB_Filter](#) module and selected the RGB_FILTER_RED_COUNT variable in the Test interface the PASS and FAIL buttons would record this red count as valid (PASS) or invalid (FAIL) amounts. If you then eliminated this module from the pipeline and reran all tests everything would appear as FAILED since the variable would no longer be created based on the image content in the pipeline. In this manner you can validate that changes to the pipeline either change or do not change previously desired results.

- Folder - Specifies the folder that contains all the images to test. This will also be the location of where RoboRealm will store a roborealms_test.dat file that contains the following configuration.
- Results - Once you specify the Folder, the Results list will immediately list all the images in that folder. By clicking on each image that will cause that image to be loaded into RoboRealm.
- Check Variables - Check if the test should verify a variables value each time a test is performed. Select that variable into the Check Variables list and specify what Tolerance value is allowed.
- Group Size - If the variable contains a list of values, the group size specifies how long that array/list should be and that all values should also be checked.
- Value Tolerance Percent - Specifies that if a variable changes as a percent of the correct value this will trigger a negative test.
- Value Tolerance Units - Specifies that if a variable changes +/- the specified number that will trigger a negative test.
- Messages - Provides some feedback on variable differences between current and recorded.
- Total Tests - The number of images tested
- Passed - The number of correct tests as compared with previous values
- Percent Passed - Radio of passed tests relative to total tests.
- Failed - The number of failed tests
- Test Duration - Number of seconds the total tests took. This can be used as a measure of performance.

WebServer

The WebServer module allows you to view images being processed by RoboRealm over the web using a regular web browser. The webserver within RoboRealm is off by default and needs to be turned on before becoming active. To activate the RoboRealm WebServer select the checkbox in the options dialog as seen below. You can view this interface by clicking on the "Options" button in the main RoboRealm dialog.



Activating the WebServer RoboRealm will transmit images over a TCP/IP network using port 8080. To connect to these images (after activating the webserver) point your browser at <http://localhost:8080/> or use your machine name in place of localhost if you are accessing the images remotely.

Images are transmitted using an MJPEG encoding to a Java applet running within your web browser. Alternatively, if you are using FireFox or other MJPEG compliant browser (this does NOT include IE) you can access the image stream directly using <http://localhost:8080/mjpeg.cgi> This allows

you to connect the video stream to other streaming systems that are typically used with Internet WebCams that stream their video also using an MJPEG format.

The webserver is configured to only return a few types of files. Most of those files are located in the RoboRealm/webroot directory that you unzipped RoboRealm into. The following are the files returned and their usage:

- index.html - contains the HTML for the Java applet and is returned on initial contact to the webserver. The page is very simple and only contains the needed Java applet code to configure the browser to contact RoboRealm and start streaming video.
- favicon.ico - the favicon is the icon used when bookmarking or indicating the link type in many web browsers.
- RoboRealm.class - the actual Java applet class returned to the web browser to start streaming.
- mjpeg.cgi - this is a virtual filename that does not exist on the file system but instead signals to the webserver to start streaming.

You can specify different variables to be communicated back to the RoboRealm Webserver based on HTML buttons/checkboxes/etc. If you look at the HTML code for the buttons you will see the onclick specified as

```
onclick="setVariable('move=1')"
```

in each of the buttons in the provided index.html file. What that does is call a Javascript routine called setVariable (also in the same HTML page) which will execute a background HTTP call to the RR webserver and tell it to create a variable called 'move' and set the value to 1. In this way any button can be created to set a variable to a specific value back in RoboRealm.

To then react to this variable (for example in driving a robot) you will need to hook that variable into whatever module you are using (typically seen as a variable dropdown). For example, suppose you are using one of the Servo modules (like the SSC) to control a servo. The variable you used in that module's variable dropdown is called "move_servo" and ranges from 0 to 255. You would then ensure that this variable is in the SSC module under the variable dropdown (either select it if already existing or type it in). Then edit your HTML page and add a new HTML button with an onclick attribute in it like:

```
<input type="button" value="GO" onclick="setVariable('move_servo=255')">
```

which would set the servo to 255 if that button is pressed. You can add more buttons to stop it, something like

```
<input type="button" value="STOP" onclick="setVariable('move_servo=128')">
```

would set the servo back to neutral 128. Using this technique of communicating back values to RoboRealm and interfacing them with various modules can provide a easy way to web enable the control of many devices.

Note that the RoboRealm WebServer is NOT intended as a replacement for industry standard web servers such as Apache or IIS and therefore does not have any additional functionality other than basic video streaming.

For security reasons, RoboRealm does NOT allow access to other folders other than the RoboRealm folder. If you wish to put images into a /images folder you will need to do so using another WebServer with the appropriate reference. Adding images to the RoboRealm installation folder (typically c:\program files\RoboRealm) WILL be accessible to the browser and served by the webserver.

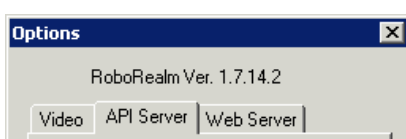
RoboRealm setup for Microsoft Robotics Developer Studio (MSRDS)

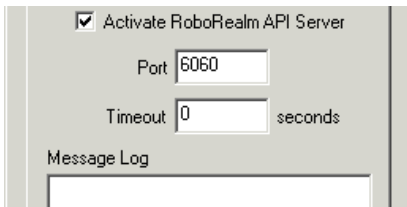
The Microsoft Robotics Studio provides an application framework for working with complex robotic architectures. Within this platform RoboRealm can provide a smart camera interface that allows you to include machine vision within the MSRS platform. The MSRS interface can either provide the resulting processed image from RoboRealm and/or include any variables generated by RoboRealm for use in further processing. In this way your webcam can be processed for specific features at the pixel level and report any findings to the MSRS orchestrating system to determine the appropriate action.

The connection between the MSRS system and RoboRealm is similar to the webcam interface currently provided with the MSRS system.

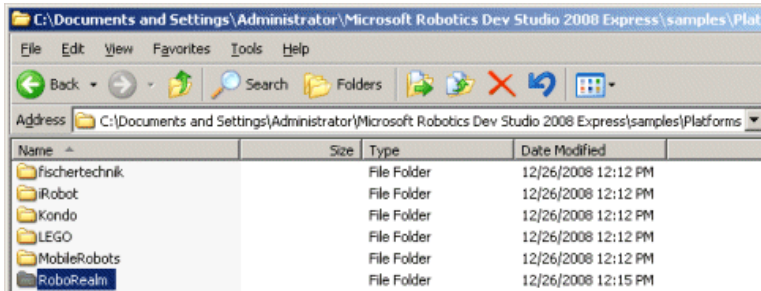
Instructions

1. [Download](#) and run RoboRealm if you have not already. Be sure to have selected the "Active RoboRealm API Server" checkbox in the Options Dialog. This ensures that the API server is running which is needed for the MSRS interface.

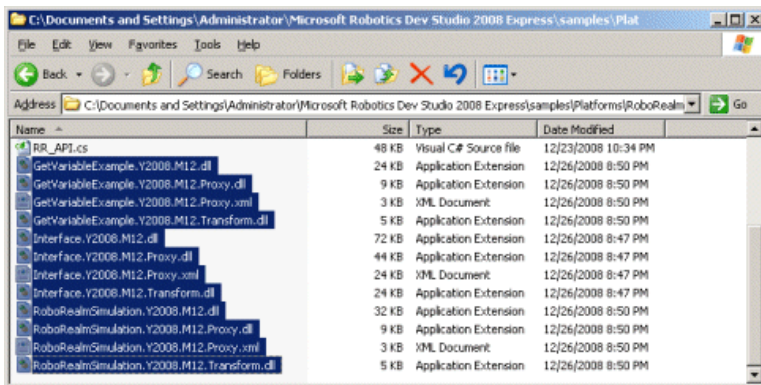




2. If you have not already done so [download](#) and install Microsoft Robotics Studio 1.5 in C:\
3. Download the [2008 Express](#) or [2010 Express R3](#) or [RDS 4 Beta 2](#) or [MSRDS 4](#) (latest) of the RoboRealm MSRS interface code.
4. **Unzip into** C:\Documents and Settings\Administrator\Microsoft Robotics Dev Studio 2008 R3\samples\Platforms or C:\Users\Administrator\Microsoft Robotics Dev Studio 4 Beta 2\samples\Platforms for Win7 users.



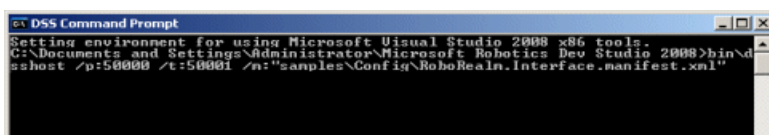
5. Copy *Interface.manifest.xml* (or *RoboRealm.Interface.manifest.xml* for Studio version) within the Platforms/RoboRealm folder to C:\Documents and Settings\Administrator\Microsoft Robotics Dev Studio 2008 R3\samples\Config\
6. Copy *.dll (all the dll files) within the Platforms/RoboRealm folder to C:\Documents and Settings\Administrator\Microsoft Robotics Dev Studio 2008 R3\bin



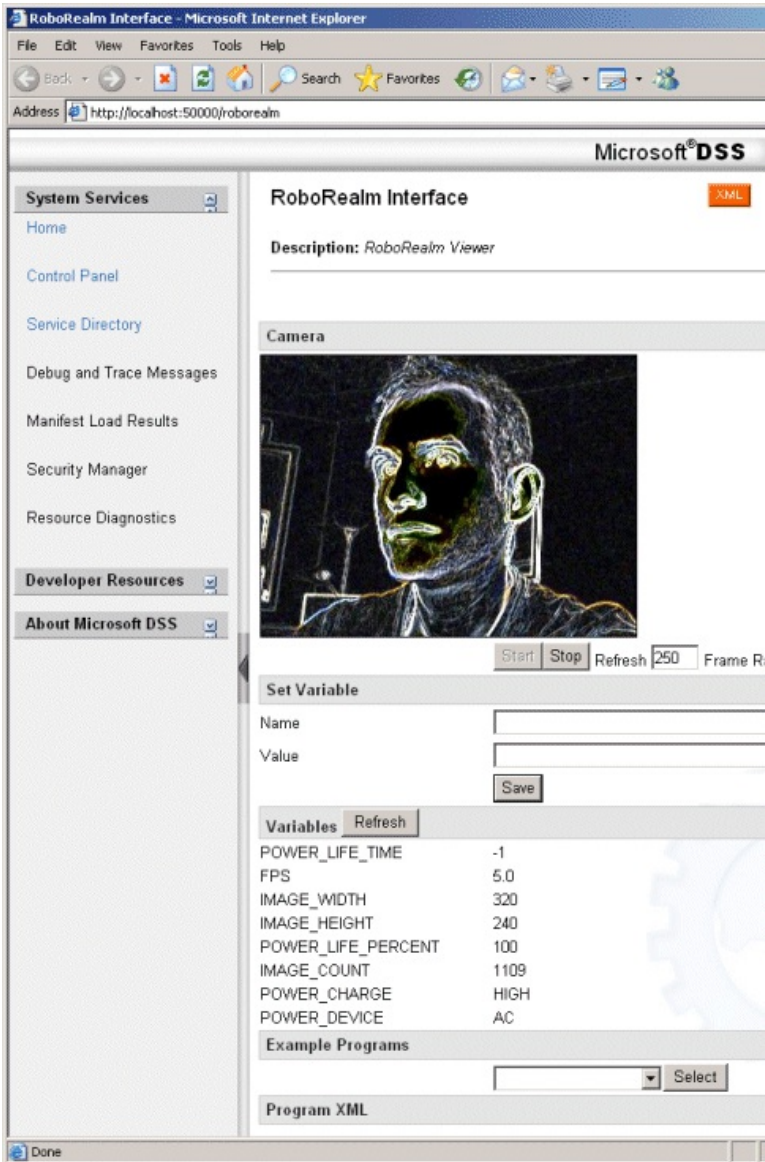
7. Start MSRS by selecting your *Start* button, select *Microsoft Robotics Developer Studio 2008 R3*, and then select *DSS Command Prompt*



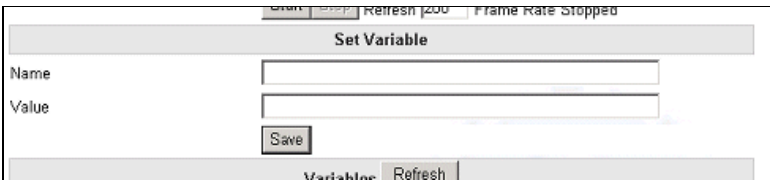
8. Copy *.xml into the Microsoft Robotics Developer Studio 2008 R3/samples/Config folder.
9. This will open a command console within the MSRS folder. Within that console type `bin\dsshost /p:50000 /t:50001 /m:"samples\Config\RoboRealm.Interface.manifest.xml"` which will start the MSRS system and register the RoboRealm interface.



- Enter the MSRS system by accessing the following url `http://localhost:50000/roborealm` using Internet Explorer (IE).
- You should now see a web interface to RoboRealm. Click on the Start button under the image to start streaming the processed video from RoboRealm

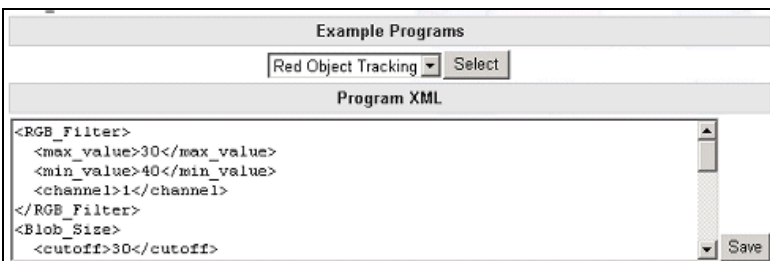


- Below the start buttons are the variable values. To change the variable values you can enter a new/existing variable into the HTML text boxes and press save to make the changes. The current variables existing within RoboRealm are listed at the bottom of the page.



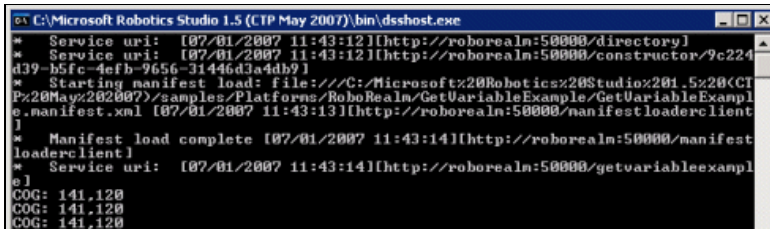
- What's Next??**

You can play with the examples programs provided to view different effects. You will notice the configuration xml in the textarea at the bottom of the page updating based on which example program you select. Try selecting the Red Object Tracking and change the `<channel>1</channel>` to 2. This changes the tracking to the color green.



14. GetVariableExample

A sample orchestration program called GetVariableExample is included in the RoboRealm MSRS examples download code. This program shows you how to write other programs that would interface to RoboRealm to control its actions and get variables from RoboRealm via the MSRS RoboRealm interface code. This example in particular shows how to grab the COG_X and COG_Y variables from RoboRealm and write their values to the command console. Use this application as a basis for how to incorporate variable access within your own orchestration programs.



```
C:\Microsoft Robotics Studio 1.5 (CIP May 2007)\bin\dsshost.exe
* Service uri: [07/01/2007 11:43:12][http://roborealm:50000/directory]
* Service uri: [07/01/2007 11:43:12][http://roborealm:50000/constructor/9c224d39-b5fc-4efb-9656-31446d3a4db9]
* Starting manifest load: file:///C:/Microsoft%20Robotics%20Studio%201.5%20(CIP%20May%202007)/samples/Platforms/RoboRealm/GetVariableExample/GetVariableExample.manifest.xml [07/01/2007 11:43:13][http://roborealm:50000/manifestloaderclient]
* Manifest load complete [07/01/2007 11:43:14][http://roborealm:50000/manifestloaderclient]
* Service uri: [07/01/2007 11:43:14][http://roborealm:50000/getvariableexample]
COG: 141.120
COG: 141.120
COG: 141.120
```

15. SimulationExample

A sample orchestration program called RoboRealmSimulation is included in the RoboRealm MSRS example download code. This program shows you how to write a program that interfaces RoboRealm to the MSRS simulator to practice your vision procedures in a purely virtual world. This example in particular shows how to track a yellow cone, approach it and then turn away for a period of time. You will need the [Simulation. robo](#) file loaded into RoboRealm in order to run the simulation correctly.

Camera Properties

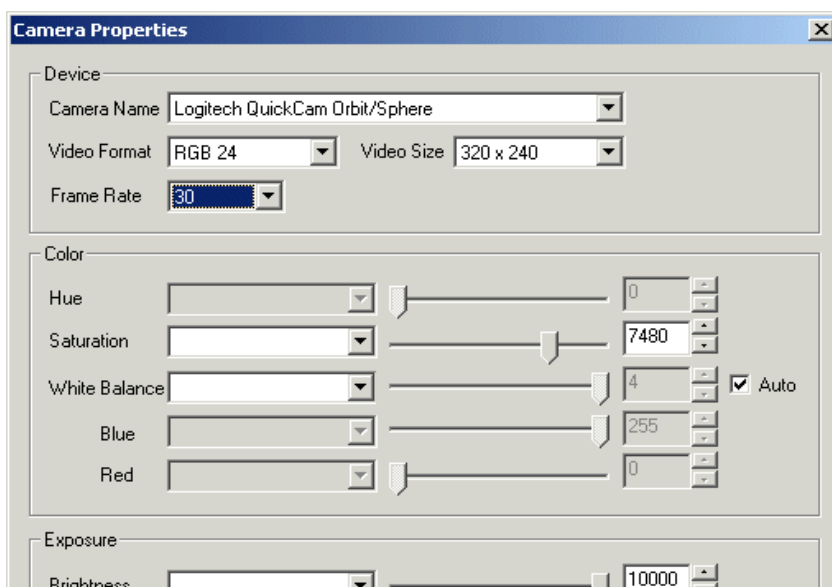
The Camera Properties module provides a way to alter the exposed DirectX camera properties built into the camera driver. This differs from camera to camera but can include exposure, brightness, gamma, etc. Changing these properties is different from other modules (such as the [brightness/intensity](#) module) in that the attribute change may be sent directly to the camera which can alter the physical capture of the image rather than just manipulating the image bytes which the other RoboRealm modules do. This has advantages since changing the capture method will always give more range to the incoming pixels than just byte mapping alone. For example, an over-exposed image cannot be digitally corrected since all the bytes might be condensed into a small numeric range whereas adjusting the exposure on the camera will allow that range to cover the full intensity range (normally 0-255) of a pixel.

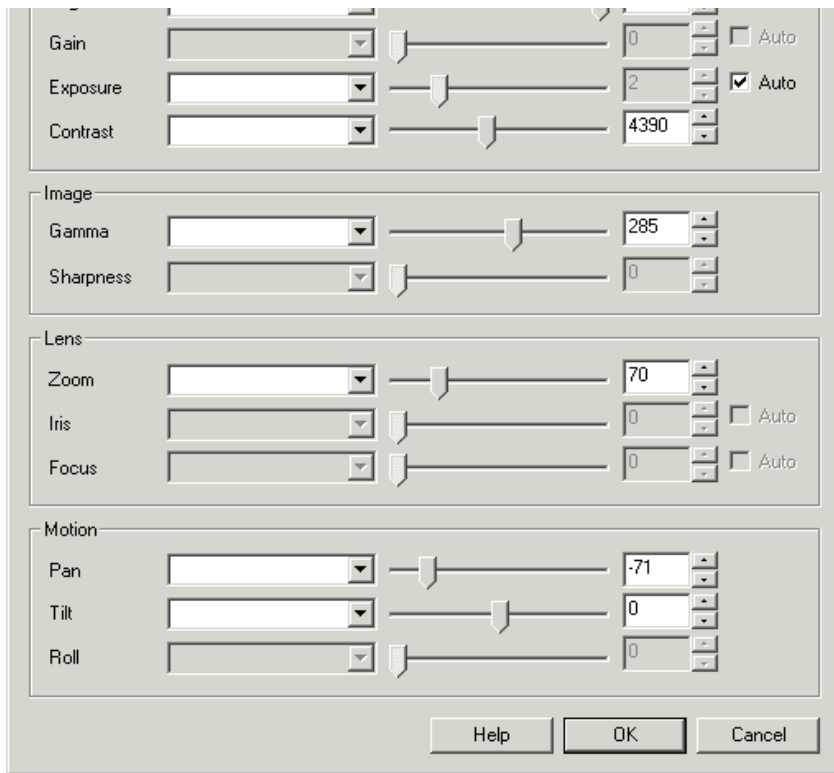
Some cameras also provide the Pan/Tilt/Zoom directX commands. For example the Creative Live Motion exposes the pan and tilt commands in DirectX to correlate with the motorized pan/tilt built into that camera. However, the Logitech Orbit translates the DirectX pan and tilt commands to a digital pan and tilt and NOT the motorized pan/tilt built into the camera. Thus you will have to experiment with your own camera to see what DirectX commands are supported and what they actually do.

Grayed out dialog interfaces mean they are not supported by your camera/driver. Sometimes updating to the latest camera driver will help to

introduce more controls for you to use. Often commands that do appear in the Options->Video->Capture button are not exposed to DirectX and are therefore not programmatically accessible. Contact your camera manufacturer and request better DirectX support if you are unable to use the desired command within this interface.

Interface





Instructions

1. Camera Name - Select the camera whose properties you wish to change. Note that only DirectX camera drivers are shown.
2. Video Format - Select which video format to use. RGB is always preferred as the most amount of color is transmitted using that mode.
3. Video Size - Select the appropriate video size to use. 320 x 240 should be sufficient for most applications.
4. Force Settings - As most webcams do not like receiving adjustments in realtime the module will ONLY send changes to the webcam when changes are detected after the initial configuration is set to the webcam. This ensure that the webcam settles on its current settings rather than trying to keep up with the new information on each pipeline cycle. This 'caching' of information can cause issue if some other module or device changes the default settings of the webcam unknown to this module. In this case you can select the 'force' checkbox which will send the camera configuration to the webcam on each pipeline cycle.
5. Wait for new Image - In most cases the next image is not desired due to time constraints. If using the current (or slightly in the past) image will suffice then unselect the "wait for new image" checkbox. This can often decrease the overall pipeline speed if the pipeline speed is slower than the image acquisition speed.
6. Edit the attributes by either selecting the appropriate value using the slider, typing in the number in the available edit area, increasing/decreasing the number using the spin/up,down arrows, or select a variable that contains the desired number to be used as configuration in this module. If you select "auto" the internal camera statistics will determine an appropriate value to use and disable the scroll and variable selection. If you instead chose a variable that contains the value to use the manual interface is then disabled.
7. Hue - refers to the overall adjustment of image colors along the color spectrum. You can use it to warm-up (add red) or cool-down (add blue) to an image.
8. Saturation - refers to the intensity of the colors, i.e. how red a red color is. By decreasing the saturation of an image you remove color and produce a monochrome grayscale picture that represents only darkness and brightness, or luminance. Increasing saturation in an image produces artificially intense colors.
9. Whitebalance - refers to the adjustment of the relative amounts of red, and blue primary colors in an image such that neutral colors are reproduced correctly.
10. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.
11. Gain - refers to the amplitude or magnification of the incoming video. Higher gain levels result in greater levels of brightness and contrast. Lower levels of gain will darken the image, and reduce the contrast. Essentially, gain modification affects the sensitivity to light of the CCD sensors. This concept is analogous to the ISO or ASA ratings of silver halide films in digital cameras.
12. Exposure - refers to how long your camera takes to record an image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.

13. Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.
14. Gamma - refers to the adjustment of red, green, or blue values of a pixel to affect how bright the image appears.
15. Sharpness - refers to the amount of edge definition or crispness in an image. The sharpness control can be used to smooth out rough edges.
16. Zoom - refers to the cameras ability to adjust focal length (Optical Zoom) or digitally increase the image size (Digital Zoom). Some cameras have built in optical or mechanical zoom which physically changes the focal length of a camera. This is unlike most digital cameras that just increase the image size using mathematical interpolation and return a clipped portion of that image.
17. Iris - refers to the control the aperture, or iris, of the camera lens.
18. Focus - refers to the amount of edge definition or crispness in an image.
19. Pan - refers to the ability for a camera to move the image viewed in the horizontal direction or X axis. This can be done mechanically or digitally. Mechanical movement is when the camera has the ability to physically move using a mechanical pan mechanism. Digital panning refers to the ability for a camera to shift the image focus to a different part of the imaged CCD. This is possible when your image is smaller than the full resolution capable of the camera or when a digital zoom is active. If you are viewing the highest camera resolution then digital panning will not be available.
20. Tilt - similar to panning but meant for movement of the image in the vertical direction or Y axis.
21. Roll - similar to panning but meant for movement of the image in the forward direction or Z axis.

Example

The following robofile shows a configuration that allows you to press the "Sample" button in the popup button interface to sample the absolute mean intensity value of an image. This is accomplished by disabling the auto-exposure that is normally used in most cameras to sample an image at a particular exposure setting (in this case 50). The script then samples the mean green value of the image and then sets the auto-exposure back on. This technique will allow for an intensity value to determine the overall lighting level in the current image without the auto-exposure. If you sample the image with the auto-exposure setting ON you will notice that the average intensity value will normally be around 128 regardless of the lighting level of the actual room/scene. This is what the auto-exposure setting is meant to do but will obviously cause absolute lighting measurements to be incorrect. You can use the script below to determine when a good time to switch on the room lights would be!

In order to run the following script you will need to edit BOTH camera properties modules and select your appropriate camera as it will most likely differ from what we have selected. To do this first disable (using the disable button) the IF statement in the pipeline as this will allow the contained modules to be executed. Then double click on the first Camera Properties module and edit the brightness, contrast, etc. until you get a good image. Repeat these settings for the next Camera Properties module and then enable the IF statement once again. On pressing the Green Sample button and clicking on the last module (Display Variables) you will see the value overlaid on the image. Now switch off/on some lights. You will notice the image change but overall remain similar but the value displayed will decrease/increase based on if you switched lights on or off.

[Click here](#) to download the lighting sample script.

Variables

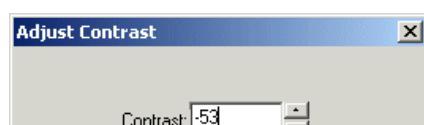
See Also

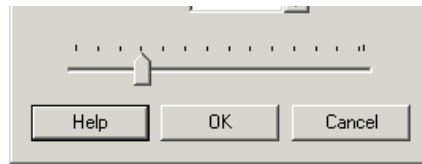
[Intensity](#)
[Contrast](#)
[Color Balance](#)

Contrast

The Contrast module changes an image's contrast. Increasing the contrast value reduces the contrast of the image. The image approaches gray as the value increases. Decreasing the contrast value causes the image to become biased to black and white pixels which tends to sharpen an image.

Interface





Example

Source Image



Contrast value of -46



See Also

[Normalize](#)

[Equalize](#)

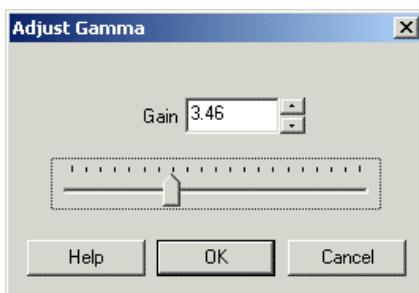
[Hue](#)

Gamma

The Gamma module helps to map pixel intensities into a more correct viewable image depending on the viewing device you are using. As most display devices like CRT tubes alter the image to < 1.0 gamma curve you might need to compensate for this display adjustment by increasing the gamma > 1.0 .

Adjusting the gamma for image will increase the midtones in intensity while leave the lowest and highest pixel intensities as they are not unlike contrast adjustment.

Interface

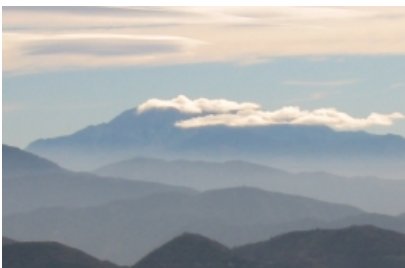


Instructions

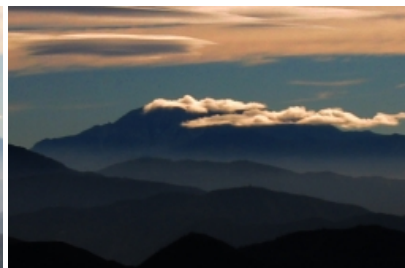
1. Gain - specify the gamma gain to be applied to the image. Note that a gain < 1.0 will increase the overall intensity of the image whilst a gain > 1.0 increases the image's contrast.

Example

Source



Gamma Adjusted



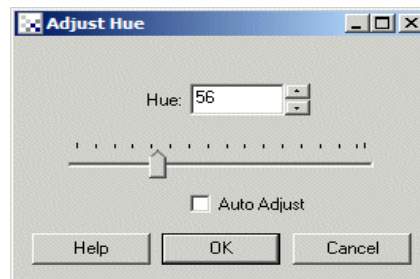
See Also

[Intensity](#)
[Contrast](#)

Hue

The Hue module changes an image's color amount. Changing the value from zero increases or decreases the amount of color in the image. The image approaches gray as the value decreases.

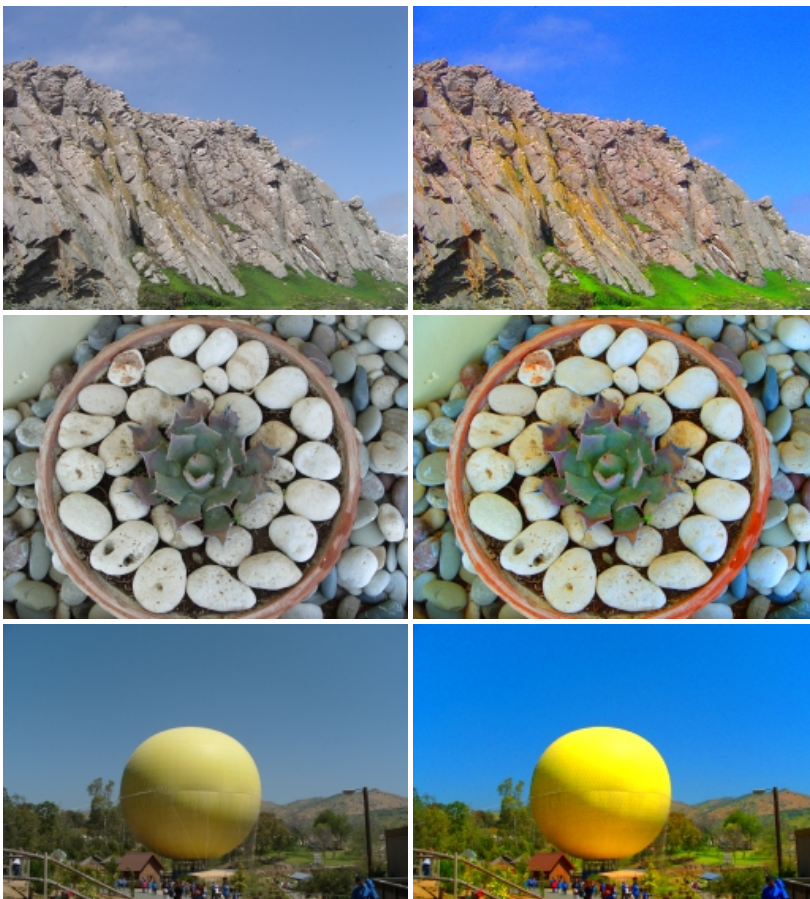
Interface



Example

Source Image

Auto Hue Adjustment





See Also

[Contrast](#)
[Normalize](#)
[Equalize](#)

Intensity

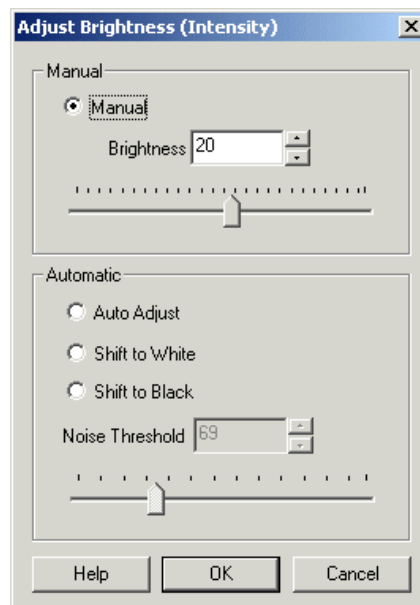
The Intensity module provides a way to brighten an image by adding a specified amount to each RGB pixel value. Note that you can specify a negative number to reduce the brightness of each pixel too. A pixels value ranges from 0 to 255. Adding 255 to each pixel turns the image white. Adding -255 will turn each pixel black.

$R=R+value$

$G=G+value$

$B=B+value$

Interface



Instructions

1. Manual - Use the count value to select how much to add to the current image. You can either type in a number or use the scroll bar. To

automate the adjustment of intensity you can use the [variable] [expression](#) in the text box in order to have the module read the value from a variable instead of using a set value.

2. Automatic - Select the appropriate method to use in determining the appropriate threshold.

- Auto Adjust - automatically adjusts the intensity level such that the mean intensity is at 128 (middle of 0 to 255).
- Scale to White - adjust all pixel values such that at least one pixel at value 255 exists. All pixels are affected by the same scale factor.
- Shift to White - adjust all pixel values such that at least one pixel at value 255 exists. This ensures that the image has more upper values of the 0 - 255 range.
- Shift to Black - adjust all pixel values such that at least one pixel at value 0 exists. This ensures that the image has more lower values of the 0 - 255 range.

3. Noise Threshold - the automatic intensity adjustment routines are very sensitive to pixels that are high in intensity. If even a spurious single pixel is pure white it will prevent most of the automatic routines from working well. Setting the noise threshold to higher than 0 will soften that requirement and instead require that a group of pixels are at a certain high intensity before adjusting the rest. This helps to ensure that the auto-adjustment occurs regardless of white noise pixels being present in the image.

Example

Source Image

Intensity value of 87



See Also

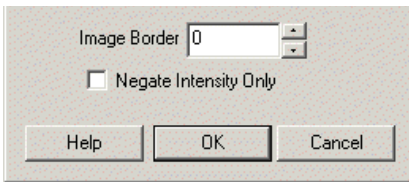
[Normalize](#)

[Equalize](#)

Negative

Interface





The Negative (or Solarize) module inverts all pixel values. For example; if a pixel is white it is changed to black, if it is black it is changed to white.

$R=255-R$

$G=255-G$

$B=255-B$

Instructions

1. Image Border - Some modules will zero out the image border to allow for quicker processing. When you negate these images those border areas will appear inverted. Use the border size to ignore those border boundaries and keep them black.
2. Negate Intensity Only - The default mode of the negative module is to invert all RGB values which can change the color of objects to their inverted color. Select the Negate Intensity Only to preserve colors and only invert a pixels intensity.

Example

Source Image

Negative



See Also

[Contrast](#)

Calculate Angle

The calculate angle module provides a way to easily calculate the angle between two lines defined by three points. The three points are assumed to already be accessible as RoboRealm variables. The result will be placed into the result variable.

Be wary of the resulting range as the angle is calculated relative to the ordering of the points that define the lines. Due to this the resulting angle might "take the long way around". For example, an angle at 120 degrees counterclockwise is also -240 degrees clockwise. See the examples below that illustrate more of this point.

Interface



Instructions

1. 3 Points Triangle - specify the points that make up a corner. The angle measured will be the smaller angle created by the three points.
2. 2 Points Line - specify two points that make up a line. The angle measure will be the angle of this line with horizontal right being 0 deg and up being 90 deg, etc. Note that due to line symmetry 180 and 0 are the same, likewise 270 and 90, etc.
3. 4 Points 2 Lines - specify four points that describe two lines that intersect. The angle will then be the smaller angle formed by the intersection of these two lines. The larger angle would then be $(360 - (2 * \text{angle})) / 2$.
4. Result - specify the resulting variable that should hold the calculated angle, whether that measurement should be provided in radians or degrees and what range that measurement should have. Note that if you specify 0 ->180 then only acute angles are provided by disregarding the ordering of the points. If you wish to measure the angle taking in consideration the ordering of the specified points select a range that extends the result such as -360 to 360.

Example

[Click here](#) to load a robofile that you can use to alter the resulting angle by clicking in the main RoboRealm interface.

The following shows the output of the example file using a right triangle. Note that only the result range is changed in each image to show the different interpretations that occur when measuring an angle.



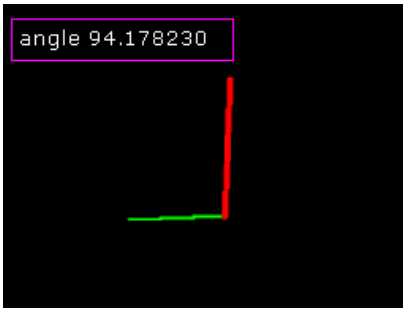
Range is 0 -> 180



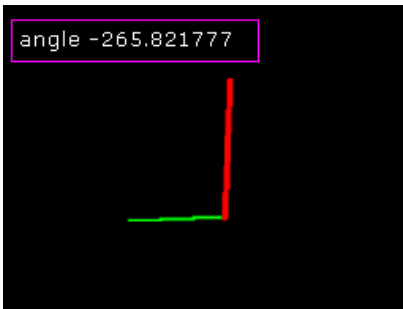
Range is 0 -> 360



Range is -180 -> 180



Range is 0 -> 180



Range is -360 -> 360

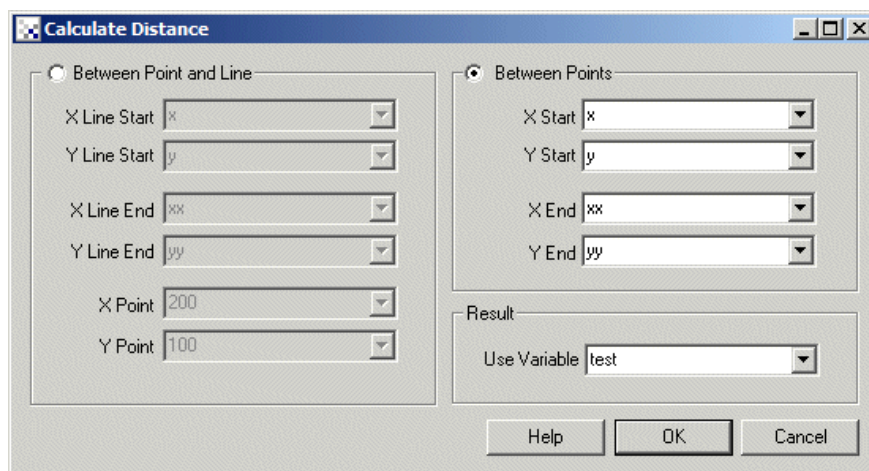
See Also

[Calculate Distance](#)

Calculate Distance

The calculate distance module provides a way to easily calculate the distance from two points. The two points are assumed to already be accessible as RoboRealm variables. The result will be placed into the result variable.

Interface



Instructions

1. Decided on which calculate type you'd like to perform. Between points means the distance from two points, Between Point and Line means the distance from a point to a line (perpendicular distance to that line).

2. Specify the variables that contain the start and stop points of the line and/or point to measure.
3. Enter the variable that will contain the calculation result. The "Calculate_Distance" will be used if nothing is specified.

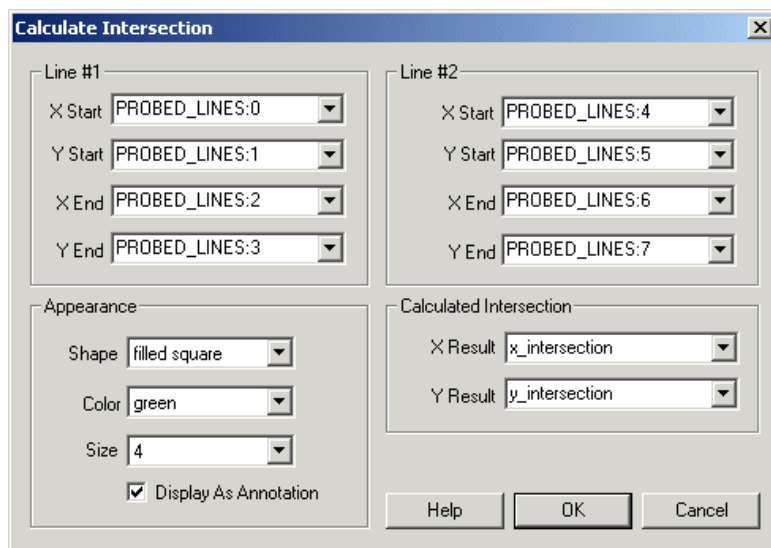
See Also

[Calculate Angle](#)

Calculate Intersection

The Calculate Intersection module provides a way to calculate the point at which two lines intersect. Note that if the lines are perfectly parallel the variables are removed as the lines will not intersect.

Interface



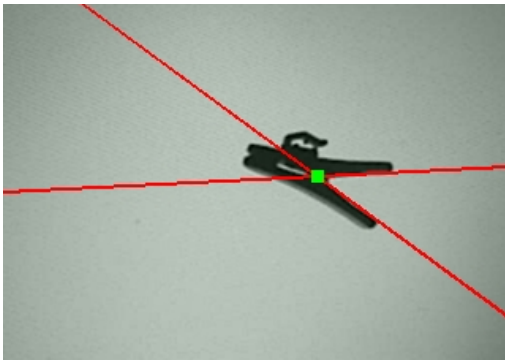
Instructions

1. Line#1, Line #2 - Specify the variables that contain the start and end points (4 points define a line) for the two lines. Note that the notation variable_array:X can be used to access values within variable arrays.
2. Calculated Intersection - Specify the variables that will hold the resulting calculated intersection.
3. Shape, Color, Size - Specify the shape, color and size of the graphic that is used to indicate the point location.
4. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

Example

Calculate_Intersection





 [Click Here](#) to load a robofile that was used to produce the above intersection calculation using the [Line Probe](#) module to determine the lines.

See Also

[Calculate Angle](#)

[Line Probe](#)

Center of Gravity

The Center of Gravity or Center of Mass statistic calculates where the COG of the image lies.

The COG is calculated by:

$$\text{COG_X} = \text{COG_X} + (I * x)$$

$$\text{COG_Y} = \text{COG_Y} + (I * y)$$

$$\text{Total} = \text{Total} + I$$

for each pixel where $I = (R+G+B)/3$ and x,y is the current pixel location. The resulting COG is then divided by the Total value:

$$\text{COG_X} = \text{COG_X} / \text{Total}$$

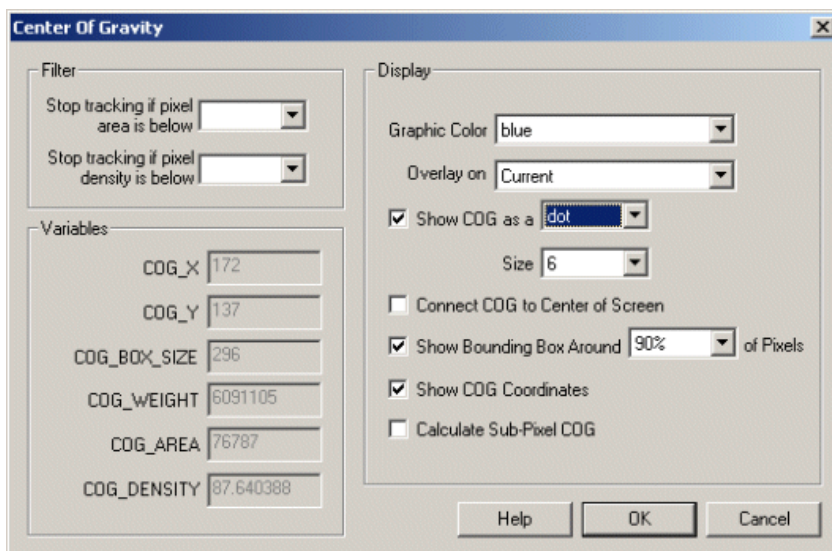
$$\text{COG_Y} = \text{COG_Y} / \text{Total}$$

to result in the final x,y location of the COG.

Note that based on the way this COG is calculated brighter pixels will exert more pull on the final COG location than darker pixels.

The COG module interface also provides for various graphical overlays.

Interface



Instructions

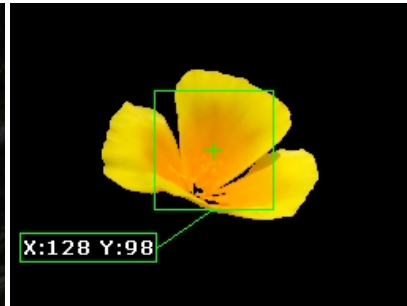
1. Graphic Color - Specify the color that overlay graphics should be displayed in.
2. Overlay On - Specify which image the graphics should be show on. This option allows you to revert the current image to the source image to display graphics.
3. Show COG as a - Specify how the COG should be displayed. Various options are provided as to how to indicate the location of the COG.
3. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Connect COG to Center - Specify if you want to draw a line from the COG to the center of screen. This is helpful to indicate how much off center the COG is. The difference between the COG and the center of screen can be used to drive differential motors accordingly.
5. Show bounding Box - Select if you would like to draw a box around the pixels that contribute to the COG. Since the image may have outlier pixels that are not part of the main object within the image you can select the percentage contribution instead of the entire image. Lowering the bounding box percentage will shrink the boxed area towards the COG.
6. Show Coordinates - Specify if you would like to show the actual COG values drawn in the image. This can give you a better idea of the actual COG values being used within your VBScript or other programming extensions.
7. Sub Pixel COG - If you need the COG calculation to be subpixel based select the "Calculate Sub-Pixel COG" which will change the COG_X, and COG_Y to decimal based.
8. Filter Area - Specify how solid the COG object must be for tracking to be enabled. You can specify the number of non-zero pixels (pixel area) that must exist in the image in order for tracking to continue. If the image contains less than the specified number of pixels then the module will terminate all processing of the pipeline. The pipeline processing would be then restarted after the next image capture.
9. Filter Density - Likewise the "pixel density" will determine how many non-zero pixels are within the current bounding box to determine how 'dense' the collection of pixels are. If just noise (small non-zero pixels spread across the image) is present the density of the bounding box will be very low. If, however, the bounding box is focused around a solid object then the density will be very high. Note that the density will be close to 100% for square objects and less than that for other shaped objects.

Example

Source



COG after thresholding



Variables

COG_X - center of gravity X coordinate

COG_Y - center of gravity Y coordinate

COG_AREA - non-zero pixels within the current image

COG_BOX_SIZE - width or height of the COG box. Note that the bounding box
is ALWAYS square

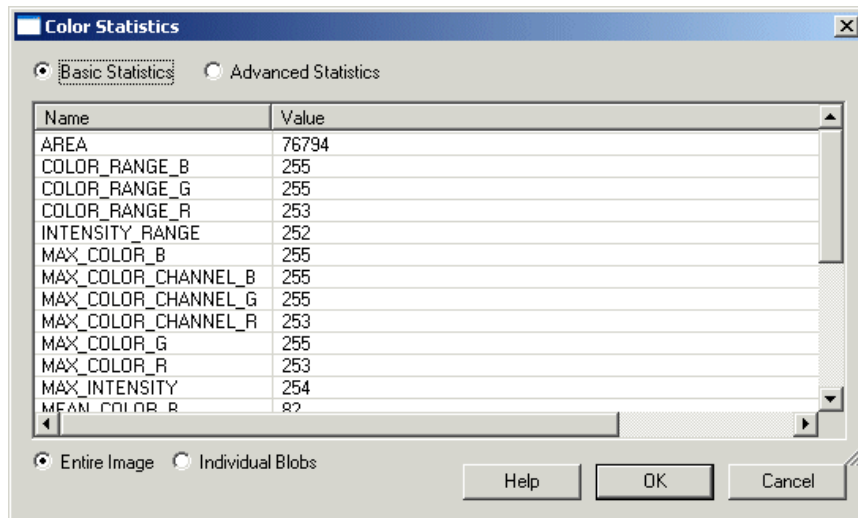
COG_WEIGHT - summation of pixels contributing to the COG

COG_DENSITY - $\text{COG_AREA} / (\text{COG_BOX_SIZE} * \text{COG_BOX_SIZE})$ provides a sense of how
dense the bounding box is

Color Statistics

The Color Statistics module adds a couple of image statistic variables centered around image color. These variables can then be used in RoboRealm [conditional statement](#), [VBScript module](#) or exported to external programs using [RoboRealm Plugins](#).

Interface



1. Scope - Click on the appropriate radio button to select if you want the statistics to be generated from the entire image or from an individual blob. If you select individual blobs non-black pixels in the current image will be used to form blobs. To see the statistics for an individual blob click on the blob within the main RoboRealm GUI window. That will switch the variable(x) denotation to that particular blob.

2. Pixel Source - If you need to group pixels of different colors into a blobs threshold or otherwise segment the current image BEFORE using this module and then select the "source" image as the image that contains the actual pixel values. This allows the module to segment blobs (of the same thresholded color) but then use the actual pixel values across that blob to determine the statistics.

Without this functionality the module would not be able to segment the current image correctly into individual blobs and end up generating hundreds of individual pixel blobs as a blob is defined as a collection of pixels with the SAME color/intensity.

Variables

UNIQUE_COLORS - Number of unique colors in the current image

MODE_COLOR_R

MODE_COLOR_G

MODE_COLOR_B - The dominant color or the color which has the most number of pixels.

MODE_COLOR_COUNT - The number of pixels that are of the dominant color

MIN_COLOR_R

MIN_COLOR_G

MIN_COLOR_B - the minimum color or rgb of darkest pixel

MAX_COLOR_R

MAX_COLOR_G

MAX_COLOR_B - the maximum color or rgb of brightest pixel

MIN_COLOR_CHANNEL_R

MIN_COLOR_CHANNEL_G

MIN_COLOR_CHANNEL_B - the minimum value of the specific color channel

MAX_COLOR_CHANNEL_R

MAX_COLOR_CHANNEL_G

MAX_COLOR_CHANNEL_B - the maximum value of the specific color channel

COLOR_RANGE_R

COLOR_RANGE_G

COLOR_RANGE_B - minimum color minus minimum color (MAX_COLOR_CHANNEL - MIN_COLOR_CHANNEL)

COLOR_RANGE_B - maximum color minus minimum color (MAX_COLOR_CHANNEL - MIN_COLOR_CHANNEL)

STD_COLOR_R

STD_COLOR_G

STD_COLOR_B - standard deviation of colors

SUM_R

SUM_G

SUM_B

SUM_I - the pixel value summation of the specific color channel

NORMALIZED_SUM_R

NORMALIZED_SUM_G

NORMALIZED_SUM_B

NORMALIZED_SUM_I - the normalized pixel value summation of the specific color channel (pixel value / image size)

VARIANCE_COLOR_R

VARIANCE_COLOR_G

VARIANCE_COLOR_B - variance of colors (std²)

MEAN_COLOR_R

MEAN_COLOR_G

MEAN_COLOR_B - the mean color

MEAN_INTENSITY - mean intensity/gray level

MAX_INTENSITY - maximum intensity/gray level

MIN_INTENSITY - minimum intensity/gray level

INTENSITY_RANGE - difference between lowest and highest intensity value

ENTROPY - representation of image entropy

Note that you can highlight the appropriate statistics and press CTRL-C to copy the information to your clipboard. This information can then be pasted into Excel or other statistics programs.

See Also

[Center of Gravity](#)

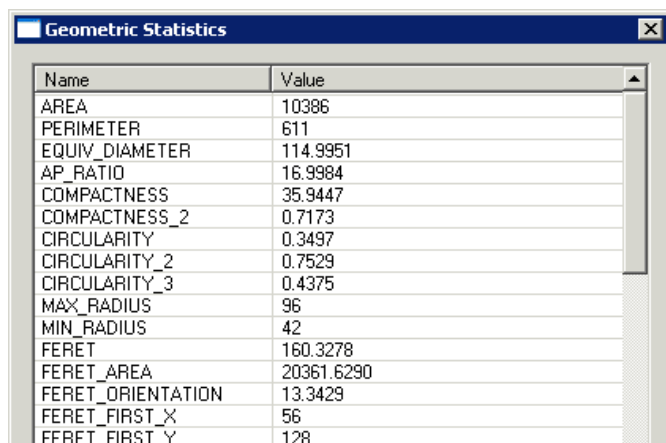
[Geometric Statistics](#)

[Moments of Inertia](#)

Geometric Statistics

The Geometric Statistics module adds a couple of image statistic variables centered around image geometry. These variables can then be used in RoboRealm [conditional statement](#), [VBScript module](#) or exported to external programs using [RoboRealm Plugins](#).

Interface



Name	Value
AREA	10386
PERIMETER	611
EQUIV_DIAMETER	114.9951
AP_RATIO	16.9984
COMPACTNESS	35.9447
COMPACTNESS_2	0.7173
CIRCULARITY	0.3497
CIRCULARITY_2	0.7529
CIRCULARITY_3	0.4375
MAX_RADIUS	96
MIN_RADIUS	42
FERET	160.3278
FERET_AREA	20361.6290
FERET_ORIENTATION	13.3429
FERET_FIRST_X	56
FERET_FIRST_Y	128

FERET_SECOND_X	212
FERET_SECOND_Y	165
BREATH	127.0001
ROUNDNESS	0.5145

Help OK Cancel

1. Scope - Click on the appropriate radio button to select if you want the statistics to be generated from the entire image or from an individual blob. If you select individual blobs non-black pixels in the current image will be used to form blobs. To see the statistics for an individual blob click on the blob within the main RoboRealm GUI window. That will switch the variable(x) denotation to that particular blob.

Variables

AREA - the number of non-black (0,0,0) pixels in the current image

ANGLE - the orientation angle of the blob. This is calculated by determining the angle from the center of gravity of the blob to the furthest point from that center along the blob's perimeter.

ANGLE_ALT - alternative orientation angle of the blob. This is calculated by determining the vector formed by the center of gravity to the perimeter's center of gravity. If your blob has holes in it this may be a more stable way to determine blob orientation than ANGLE.

ANGLE_ALT_2 - alternative orientation angle of the blob. This is calculated by determining the average vector formed from a current perimeter point to the center of gravity.

COG_X, COG_Y - the center of gravity of the blob

PERIMETER - the number of pixels that surround non black blobs

AP_RATIO - AREA / PERIMETER, determines how round an image is; also known as "Perimeter Equivalent Diameter".

EQUIV_DIAMETER - the diameter of a circle with the same area as the region. Calculated as $\sqrt{4 * \text{Area} / \pi}$

COMPACTNESS - (PERIMETER*PERIMETER) / AREA : ratio of the square of the perimeter to the area; also known as "shape".

COMPACTNESS_2 - $\sqrt{4 * \text{AREA} / \pi} / \text{FERET}$: alternate compactness measure

CIRCULARITY - $4 * \pi * \text{AREA} / (\text{PERIMETER} * \text{PERIMETER})$, measure of circularity.

CIRCULARITY_2 - alternate measure of circularity.

CIRCULARITY_3 - MIN_RADIUS/MAX_RADIUS : another alternate measure of circularity.

MAX_RADIUS - radius of enclosing circle around center of gravity

MIN_RADIUS - radius of enclosed circle around center of gravity

FERET - Feret's diameter also known as the caliper length or largest axis length: the greatest distance between any two non-zero pixels

FERET_AREA - FERET * BREATH; also know as Feret's bounding box

FERET_ORIENTATION - orientation of the Feret's diameter in degrees

FERET_FIRST_X

FERET_FIRST_Y

FERET_SECOND_X

FERET_SECOND_Y - points that makeup the Feret diameter

BREATH - the largest axis perpendicular to the Feret diameter

ROUNDNESS - $4 * \text{AREA} / (\pi * (\text{FERET} * \text{FERET}))$, measure of roundness

MIN_X - minimum bounding box x coordinate

MIN_Y - minimum bounding box y coordinate

MAX_X - maximum bounding box x coordinate

MAX_Y - maximum bounding box y coordinate

EXTENT - proportion of the pixels in the bounding box that are also in the region. Calculated as (blob area / bounding box area).

TOP_LEFT_X
TOP_LEFT_Y
TOP_RIGHT_X
TOP_RIGHT_Y
RIGHT_TOP_X
RIGHT_TOP_Y
RIGHT_BOTTOM_X
RIGHT_BOTTOM_Y
BOTTOM_LEFT_X
BOTTOM_LEFT_Y
BOTTOM_RIGHT_X
BOTTOM_RIGHT_Y
LEFT_TOP_X
LEFT_TOP_Y
LEFT_BOTTOM_X
LEFT_BOTTOM_Y - the extrema points of the image

FERET_ASPECT_RATIO - FERET/BREATH

BOX_ASPECT_RATIO - (MAX_X-MIN_X)/(MAX_Y-MIN_Y)

WIDTH - the width of the object in pixels

HEIGHT - the height of the object in pixels

Note that you can highlight the appropriate statistics and press CTRL-C to copy the information to your clipboard. This information can then be pasted into Excel or other statistics programs.

See Also

[Center of Gravity](#)

[Moments of Inertia](#)

[Color Statistics](#)

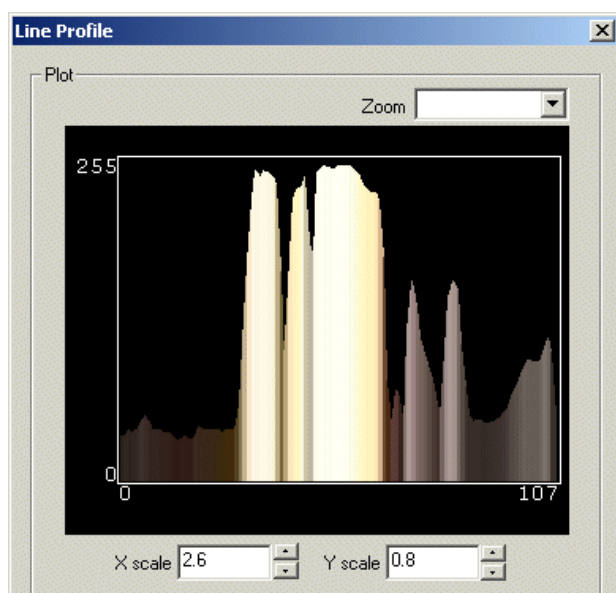
Line Profile

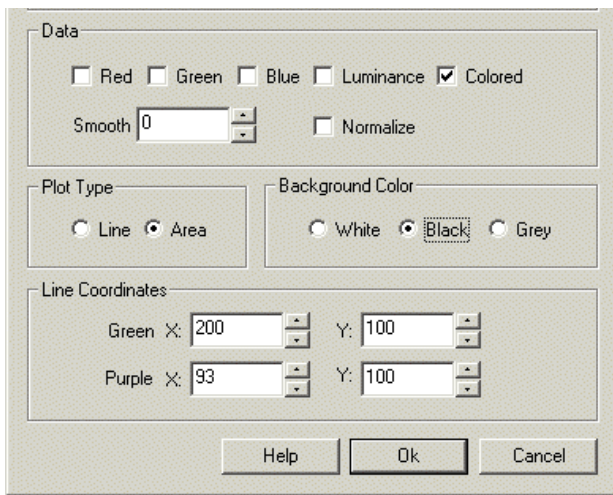


The line profile module will allow you to better investigate the pixels values along a specified line. The values of each pixel is charted such that the intensity of the pixel is the Y axis (height) and the pixel number along the specified line is the X axis (width). This arrangement better illustrates the neighborhood of a particular pixel from a different perspective.

The Line Profile module will also place the generated chart and the sampled pixel values into variables that can be accessed by other modules or saved for later usage.

Interface





Instructions

1. Line Coordinates - Specify the appropriate coordinates for the line. You will see a green and purple square connected by a line in the main RoboRealm image. This illustrates where the line is currently sampling the image. You can either specify a number directly in the provided text fields, or use the up and down spin buttons to change the value of these coordinates. Alternatively you can use the [\[expression\] syntax](#) to programmatically move the line location using other variables or formulas.
2. Data - Select which data channel you wish to plot. The default Colored channel uses the pixels intensity to determine height but then uses the original pixel values color when drawing the area chart.
3. Smooth - If the plotted data appears too noisy try increasing the smoothing factor to reduce the noise. This will cause the plot to become smoother and more rounded. Peaks and valleys become more evident with smoothing.
4. Normalize - If selected the data will be normalized which will map the pixel values into a full intensity range to better illustrate the peaks and valleys of the sampled data.
5. Plot Type - Line will display the data as a single line plot whereas Area will fill the chart to better show the height of the sampled pixels.
6. Background Color - Depending on the data being plotted the chart background can be changed to better show the graph values. Select any of the radio buttons to change the background color.
7. Display Line - if you don't wish to see where the line is being sampled uncheck this checkbox. Note that sometimes when switching to other images the line annotation will still be active, thus switching off the display will remove this confusion.
8. Draw Chart Graphics - This specifies that the axis lines and values are included in the line profile image. While convenient to view, often the profile graphic is used in subsequent processing (exposed in the main RoboRealm GUI using the [Marker](#) module) which requires no axis or annotations.

Variables

LINE_PROFILE_VALUES - the array of pixel values that generated the chart

LINE_PROFILE - the chart image saved as an image that can be used in other modules that use an image dropdown selection.

See Also

[Sample Line](#)
[Sample Color](#)
[Line Probe](#)

Moments of Inertia

Moments of Inertia are statistical values that quantify the rotational inertia of a rigid body (in this case the image intensities). Basically they quantify how much effort would be required to start spinning the image if each image pixel intensity was associated with a certain mass. The higher the pixel

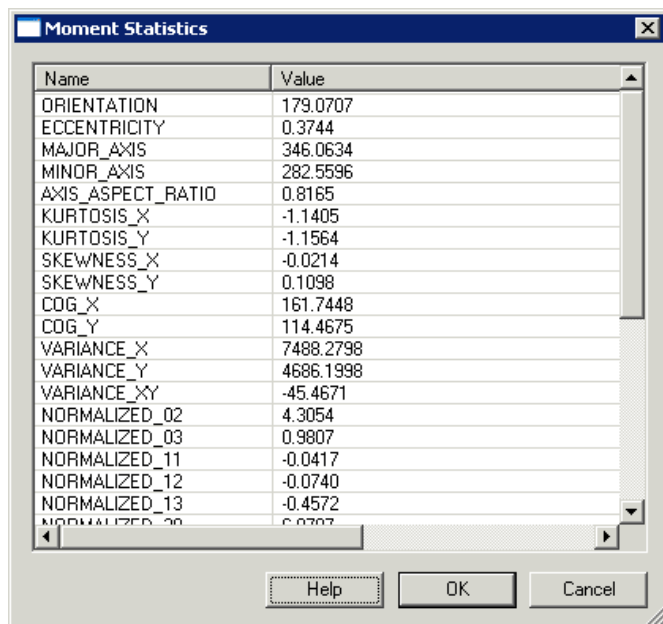
how much effort would be required to start spinning the image if each image pixel intensity was associated with a certain mass. The higher the pixel intensity the more mass it represents which means the harder it is to induce a rotational movement. (See below for more information).

You can use Moments as an effective compact image descriptor that can be used to differentiate it from other images.

Note that Moments are typically used on images that have already been segmented and operate only on image intensity. Color images are converted to grayscale prior to calculating the statistics.

The Moments module calculates moment based statistics and adds them to the RoboRealm variable collection. These variables can then be used in RoboRealm [conditional statement](#), [VBScript module](#) or exported to external programs using [RoboRealm Plugins](#).

Interface



1. Scope - Click on the appropriate radio button to select if you want the statistics to be generated from the entire image or from an individual blob. If you select individual blobs non-black pixels in the current image will be used to form blobs. To see the statistics for an individual blob click on the blob within the main RoboRealm GUI window. That will switch the variable(x) denotation to that particular blob.

2. Pixel Source - If you need to group pixels of different colors into a blobs threshold or otherwise segment the current image BEFORE using this module and then select the "source" image as the image that contains the actual pixel values. This allows the module to segment blobs (of the same thresholded color) but then use the actual pixel values across that blob to determine the statistics.

Without this functionality the module would not be able to segment the current image correctly into individual blobs and end up generating hundreds of individual pixel blobs as a blob is defined as a collection of pixels with the SAME color/intensity.

Variables

ORIENTATION - angle (in degrees) between the x-axis and the major axis of the ellipse

ECCENTRICITY - ratio of the distance between the foci of the ellipse representation of the image and its major axis length

MAJOR_AXIS - if you were to represent the image as an ellipse this would be the length in pixels of the major axis of that ellipse

MINOR_AXIS - if you were to represent the image as an ellipse this would be the length in pixels of the minor axis of that ellipse

AXIS_ASPECT_RATIO - MINOR_AXIS/MAJOR_AXIS - aspect ratio of representational ellipse

KURTOSIS_X

KURTOSIS_Y - a measure of the "peakedness" of the image intensities

SKEWNESS_X

SKEWNESS_Y - a measure of the asymmetry of the image intensities

COG_X

COG_Y - the X,Y coordinate of the center of mass/gravity of the region

VARIANCE_X

VARIANCE_Y

VARIANCE_XY - variance around the mean given the appropriate axis

NORMALIZED_ij - the scale invariant centralized moments (ij means 00 or 01 or 20 etc.)

INVARIANT_ij - the Hu invariant centralized moments (ij means 00 or 01 or 20 etc.)

Note that you can highlight the appropriate statistics and press CTRL-C to copy the information to your clipboard. This information can then be pasted into Excel or other statistics programs.

See Also

[Geometric Statistics](#)

[Color Statistics](#)

Reception Quality

The Reception Quality module attempts to detect when wireless video reception has either been lost or has not been received correctly. Often when using wireless cameras that transmit NTSC type signals over the airwaves a lot of disturbance and image distortions can occur. This is disastrous to a machine vision system that is watching for certain shapes and colors. Often when the wireless signal goes bad a lot of primary colors are seen which can trigger most color detectors to falsely report a color. These bad signals are typically temporary and usually recovered in a couple of seconds.

The module creates a new variable called RECEPTION_QUALITY which will contain a zero when no signal is detected, 1 when the signal appears to be stable, and >1 when the signal is considered distorted. Note that in very large changes of the camera this module may determine that the image signal has gone bad, thus this module will not work in environments where there is significant image changes.

Instructions

1. Add the reception quality module to your processing pipeline
2. Using conditional logic check to see if the RECEPTION_QUALITY equals 1. Proceed normally if so.

Example

Source

Bad Reception Quality = 3



Variables

RECEPTION_QUALITY - variable that holds the reception quality typically from 0 to 5

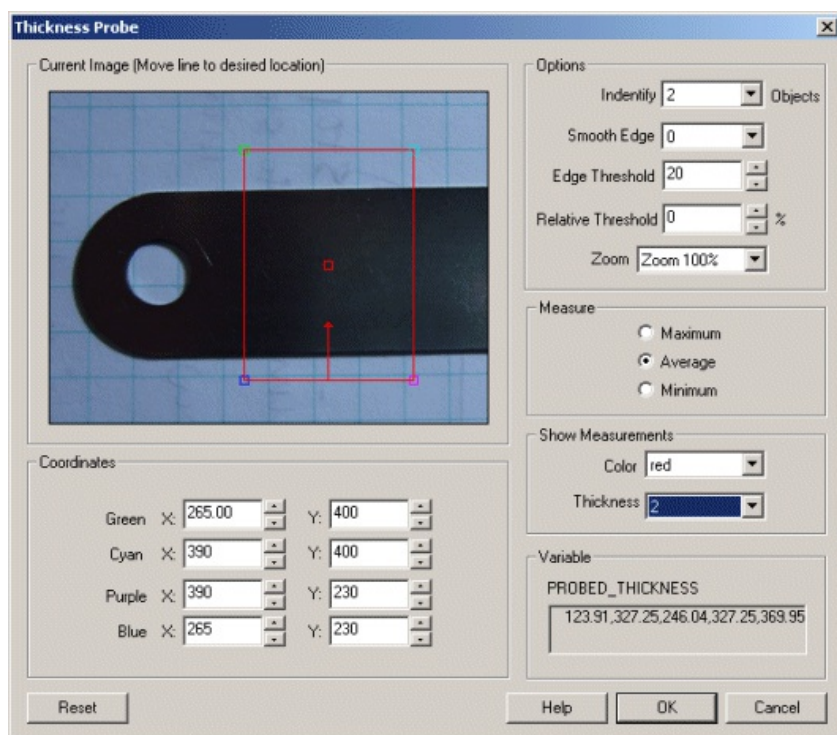
See Also

[Focus](#)

Thickness Probe

The Thickness Probe module provides a way to sample two edges and determine the maximum, average or minimum distance between the detected edges. This is essentially relates to the thickness of an object.

Interface



Instructions

1. Current Image - move the probe graphic to the location which you want to probe for lines. You drag the red square to move the entire probe or move each of the corners to change the shape and size of the probe. Note that the arrow dictates the search direction and should always be perpendicular to the edge being detected. You can also move each endpoint individually by changing the coordinates using the text boxes below in the Coordinates area.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the probe of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

3. Options Identify - select how many objects you want to measure. Note that if you select many objects to be identified you are not guaranteed to get that many objects depending on the thresholding information below. The "Identify" number specifies a maximal number of objects to detect.

4. Smooth Edge - to reduce noisy edges select the amount of smoothing that should be applied to the edge prior to edge detection.

5. Edge Threshold - to eliminate very weak edges from being detected as part of a object boundary select an appropriate edge threshold (0-255) that will remove edges whose edge intensity is below the threshold. Note that smoothing the edge will also reduce the edge intensity and thus the Edge Threshold will need to be adjusted after the smoothness is specified.

6. Relative Threshold - to only select those edges that are significant you can specify the relative threshold (0-100) that will remove successive edges that are the relative threshold percent less than the previous edge. For example, consider the highest 3 edge values as 130, 120 and 40. If the relative threshold is 50% then the last edge 40 would be eliminated since 40 is less than 50% of 120. As 120 is 92% of 130 it is not eliminated (unless the threshold where set at 95%).

7. Zoom - you can zoom the current image by selecting an appropriate zoom. This will allow you to better select the appropriate values with finer resolution.

8. Measure - select the appropriate measurement that you want to perform on the object.

Maximum will measure the maximum span between the start and end edge

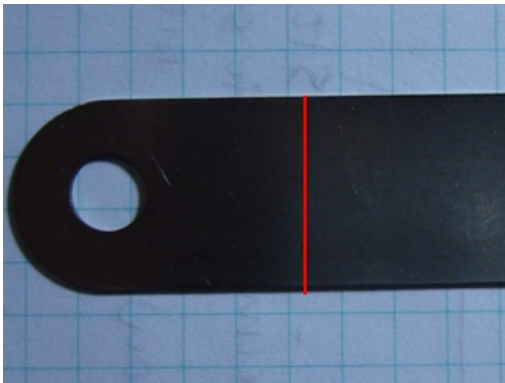
Average will measure the average span

Minimum will measure the minimum span between the start and end edge

9. Show Measurements - to visually see which span is detected (in the main RoboRealm interface) select the appropriate Color and Thickness of the circle that will indicate where in the image the maximum, average, minimum span have been detected.

Example

Result of above configuration



Variables

PROBED_THICKNESS_COUNT - the number of detected objects.

PROBED_THICKNESS - an array containing the detected objects. The format is

1. object span length (either maximum, average or minimum)
2. x start coordinate
3. y start coordinate
4. x end coordinate
5. y end coordinate

See Also

[Origin Probe](#)

[Edge Probe](#)

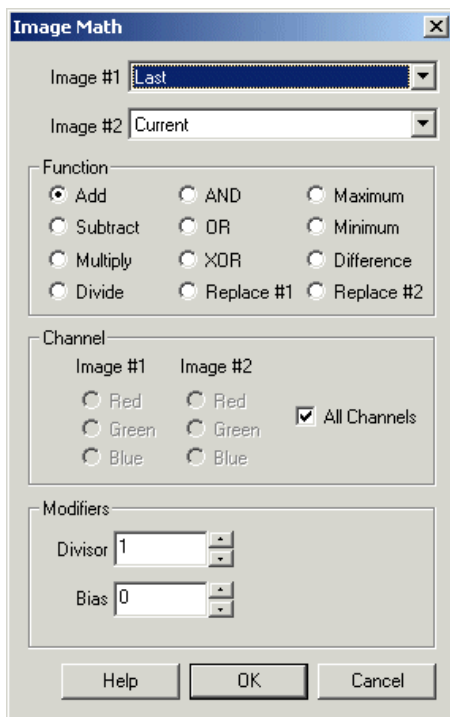
[Line Probe](#)

[Circle Probe](#)

Image Math

The Image Math module provides you with a way to perform arithmetic operations on a combination of two images.

Interface



Instructions

1. Select the appropriate Image #1 and Image #2 on which to perform the function.

Source - the original image that was initially loaded or captured into RoboRealm

Current - the currently processed image within RoboRealm

Last - the last image processed by RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created.

If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

2. Select the appropriate function used to combine the two images.

Add Image #1 + Image #2

Subtract Image #1 - Image #2

Multiply Image #1 * Image #2

And Image #1 & Image #2 (binary AND)

Or Image #1 | Image #2 (binary OR)

XOR Image #1 ^ Image #2 (binary XOR)

Maximum if Image #1 > Image #2 then result = Image #1 else result = Image #2

Minimum if Image #1 < Image #2 then result = Image #1 else result = Image #2

Minimum If Image #1 > Image #2 then Result = Image #1 else Result = Image #2

Difference | Image #1 - Image #2 | (Absolute value of subtraction)

Divide Image #1 / Image #2

Replace #1 If Image #2 then Image #1 else 0

Replace #2 If Image #2 then Image #1 else 255

3. Select the appropriate channel to perform the function on.

All channels - Performs the operation on each RGB channel

Red, Green, Blue - Performs the operation only on the selected channel. The result is placed in the selected Image #2 channel.

4. Select the appropriate divisor and bias.

Divisor - divides the function's result by the specified amount.

Bias - added to the function's result.

Example

Source Image #1



Source Image #2



Image #1 added to Image #2



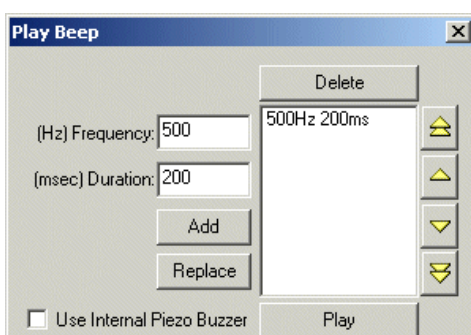
See Also

[Marker](#)

Beep

Creates an audible beep when run. Note that you probably want to use this as a signaling technique and not play a beep every time a frame is processed!

Interface



Help

OK

Cancel

Instructions

1. Manual Frequency - Specify beep frequency
2. Manual Duration - Specify beep duration
3. Manual Add - Press "Add" to add the beep into the play list
4. Manual List - Move the beep list items up, down, or edit the list as needed
5. Variable Frequency - Specify a variable or type one in that will contain the frequency of the tone to produce. If this is zero there will be no audible tone.
6. Variable Duration - Specify how long the note should play for by specifying a variable that contains that duration in milliseconds. If this is 0 the note will play continuously until either the frequency is set to zero or the application is terminated.
7. Specify if the beep should use the Internal Piezo Buzzer or play through the speakers.
8. Click OK to add the module to the image pipeline.

Note that when the variable selection is used the frequency can be changed even while the note is playing if the duration of the note is zero (continuous play).

See Also

[PlayWav](#)

[Play Rtttl](#)

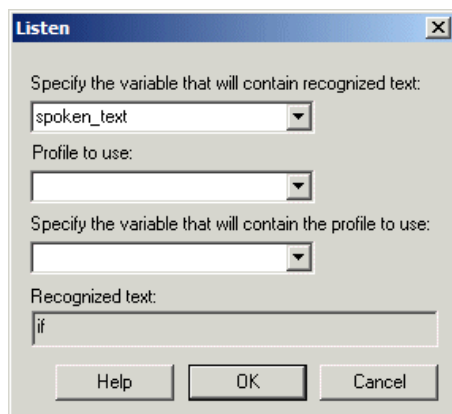
Listen

The Listen module uses the MicroSoft Speech 5.0 to listen for spoken text. This module is useful for changing states during the execution of the program when no keyboard is available on the robot.

Any text heard will appear in the text box and assigned to the variable specified in the interface. We suggest you complete at least one vocal training otherwise the speech recognition will be unusable.

Often the context of speech is important. If you find that you are issuing single word commands without much recognition luck, try to incorporate other words that help localize the context of the word. For example, instead of just saying "yellow" say "the color yellow" and look for the word yellow somewhere in the recognized text. This search can be done using a VBScript module using the InStr function.

Interface



Instructions

1. Specify the variable that will contain the recognized text. You can either select a variable from the dropdown or type in a new variable. Note that any recognized text will appear in the "Recognized text" box

any recognized text will appear in the recognized text box.

2. Profile to use - select which voice profile you'd like to use in recognizing text.
3. If you wish to change the profile automatically based on some other event you can instead specify a variable that will contain the numerical index or the actual name of the profile as seen in the dropdown list. When this variable changes the profile will automatically switch to the one specified.

Downloads

The Listen module requires you to download the Microsoft Speech API 5.0 You can find the download at <http://www.microsoft.com/speech/download/sdk51/>

See Also

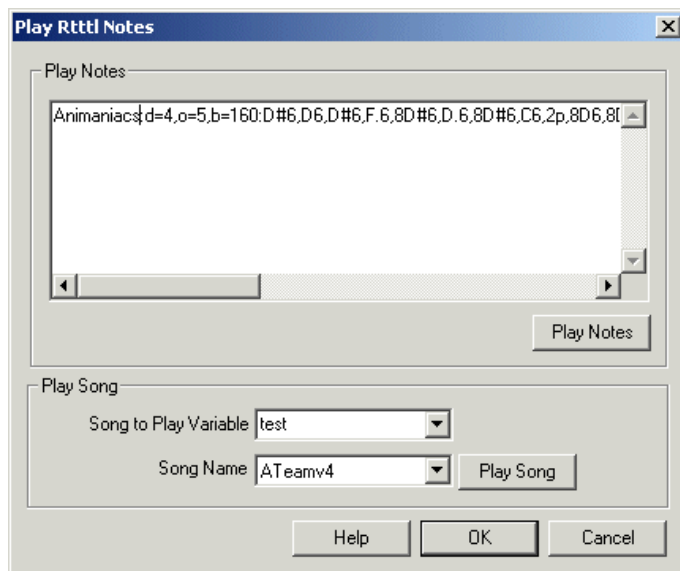
[Speak](#)

Play RTTTL



The Play RTTTL notes module will play the specified RTTTL (Ringtone) notes on a PC's speaker. This is similar to the [Beep](#) and [Play Wave](#) modules but offers a different type of tone generation.

Interface



Instructions

1. Play Notes - you can paste RTTTL notes right into the Play Notes text area to play them on your PC speakers. Note that each time this module is run it will play the notes so you may want to surround this module with the [If Statement](#) module to selectively play those notes.
2. Play Variable - you can also choose to just play one of the default songs provided with RoboRealm in the music.rttl file installed alongside RoboRealm. To do this select a variable that contains or will contain the name of the song to play. To select which song to play you can select it from the Song Name dropdown, play the song using the Play Song button and then incorporate its name in the play variable. Note that after the song is played the variable will be cleared to avoid repetition until it is reset.

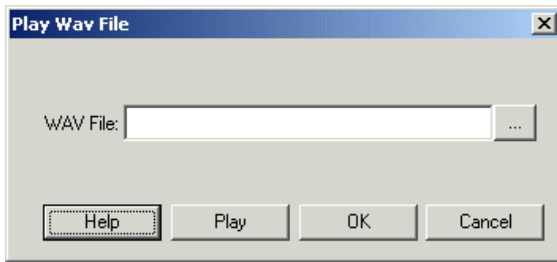
See Also

[Beep](#)
[Play Wave](#)

Play Wav File

Plays the specified WAV file.

Interface



Instructions

1. Specify the wav file to play by clicking on the browse "... " button.

See Also

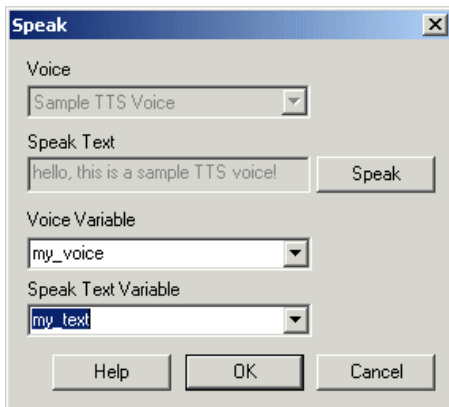
[Beep](#)
[Play Rtttl](#)

Speak

The Speak module uses the MicroSoft Speech 5.0 download to speak written text. This module is useful for indicating certain states within the execution of the program when no screen is available on the robot.

There are two ways to use the speech module. You can either specify a variable that will contain text that will be spoken or you can type in the text directly. Using a variable is useful if you wish to change the spoken text based on a [VBScript](#) module or other [plugins](#).

Interface



Instructions

1. Voice - select the voice to use when speaking from the dropdown.
2. Speak Text - type in the words that should be spoken in the specified voice.
3. Voice Variable - Specify the variable that holds the voice to use. This can either be a number indicating the appropriate voice as seen in the dropdown list or can be the name, i.e. "Microsoft Sam". Using the voice variable will disable the voice dropdown to indicate that the voice used to speak is now being specified from a variable.
4. Speak Text Variable - Specify the variable that will contain the text to speak. You can either select a variable from the dropdown or type in a new variable. If this is used the manual textbox is disabled to indicate that text is now coming from the contents of a variable.

Example Audio

[Speak](#): Robo Realm is a neat program. It makes my computer speak to me!

Note that when specifying text to speak you can play around with punctuation to change the emphasis. For noun words you may try separating the word to improve enunciation. Try it with the text above but merge "Robo realm" into one word. We found it sounds better separated.

Downloads

The Speak module requires you to download the Microsoft Speech API 5.0 You can find the download at [Microsoft Speech](#)

See Also

[Play Wavefile](#)
[Beep](#)

Blob Border

The Blob Border module is used to identify just the outline of blobs of different colors such that they can be processed as edges by other modules. For instance, if an image is segmented into colored blobs simply by reducing the number of colors used there will be many blobs that touch each other. Applying a circle detection module will assume that only edges are present in the image and will not function correctly unless edges are present. Using the Blob Border module prior to the circle detection module will ensure that the circle module treats each differently colored blobs as an edge.

Instructions

1. Insert the Blob Border module. All blobs will be converted to just their outlines.

Example

Source



Blob Border after Flood Fill



See Also

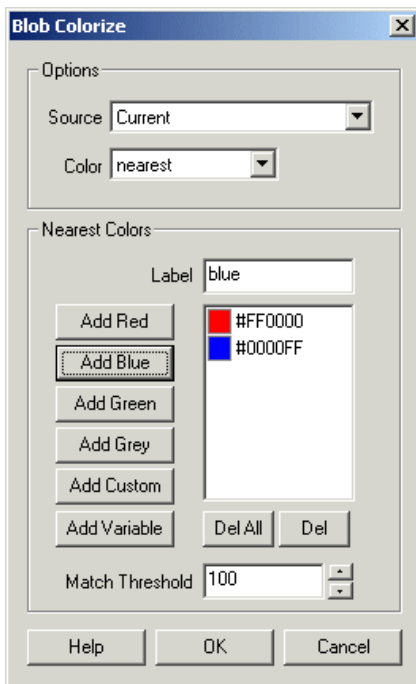
[Border](#)
[Blob Separate](#)
[Flood Fill](#)

Blob Colorize

The Blob Colorize module provides a way to recolorize blobs (segmented objects) based on another image (normally the source pixel data). This is required when you have processed an image into separated blobs but need to recombine the blobs with their original color information that was lost during processing.

The Blob Colorize will use a source pixel image and replace each non-black object with the various colors drawn from the source image. This is functionally similar to using the blobs as masks into the source pixel image and summing the resulting pixels to form the new color.

Interface



Instructions

1. Source - Select which image represents the source pre-processed RGB pixel image of the current image.
2. Color - Select which color statistic to gather based on the source image that will be the new color of the respective blob. When 'nearest' is selected the bottom part of the GUI interface will be enabled.

mean - Determines the mean color of the Source image masked with the current image and replaces the current blob with that color. This colors the current blob with its average color in the Source image.

min/max intensity - Determines the minimum or maximum pixel intensity of the Source image masked with the current image and replaces the current blob with that color. This colors the current blob with its min/max intensity as seen in the Source image.

min/max color - Determines the minimum or maximum pixel color of the Source image masked with the current image and replaces the current blob with that color. This colors the current blob with its min/max color as seen in the Source image.

nearest - Determines the mean color of the Source image masked with the current image and replaces the current blob with the color closest to that mean. This colors the current blob with one of the colors as specified in the Nearest Colors list.

summation - Replaces the current blob with the added intensity values based on the Source image. This enables the Weight field that is used to determine how the final intensity value is adjusted. When the weight is set to 0.25 the pixels that belong to a blob are added as specified in the Source image multiplied by the weight. This means a blob with 4 pixels of 255 intensity will result in a 255 white blob. This allows smaller dimmer blobs to be eliminated (via a threshold) but smaller bright blobs to remain.

3. Label - specify the name of the color (as will be used in the variable array) such that you can easily identify a color based on a name as apposed to an RGB triplet.

4. Add X - add in colors as needed in order to create a list of colors that will be matched against. For each detected blob in the image the color those most closely matches that blobs color will become that blobs color. This allows an almost white or almost red blob to become pure white or pure red assuming that you have both white and red as "Nearest Colors".

5. Match Threshold - specify how closely a color should match one of the "Nearest Colors" to be replaced with that color.

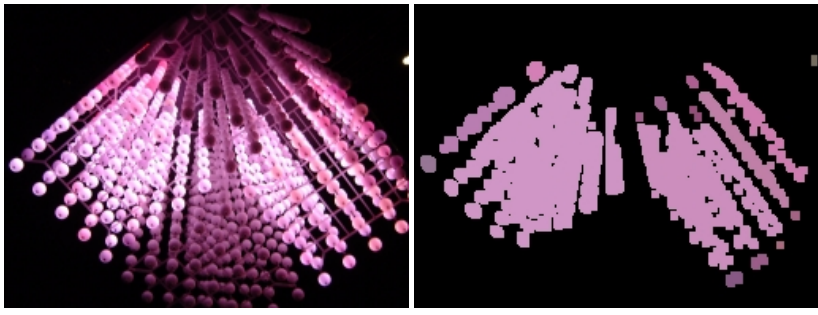
6. Summary Weight - enabled when Summation is selected. A weighting factor that is multiplied on each pixel prior to the final intensity being

of binary image. These measurements are then used to compute the distance from the center of gravity to the boundary, being capped.

Example

Source

Blob Colorize



The above demonstrates the blob colorizing routine after thresholding the image for high intensity, dilating by 2 and then using the blob colorizing module to recolor the resulting white blobs.

This technique is very handy for laser light detection. The issue with laser lights is that while we see them as red (or green) but the camera sees them as white light surrounded by a red halo due to the color range limitation of CCDs. In order to detect "red" spots we need to first detect likely laser spots, expand the blob and then test for color. The following shows two laser lights from the [SRV-1 robot](#) and the final step of the detected two laser lights with the X coordinate of the blobs (this can be used for distance sensing). [Download](#) the robofile that accomplishes this.

Source

Laser Detected



Variables

BLOB_COLORIZE_LABELS - contains an array of matched color information for each blob.

The array contains blocks of 6 values per colorized blob. The elements are as follows:

Offset	Contents
0	x coordinate of the center of gravity of the blob
1	y coordinate of the center of gravity of the blob
2	hex formatted color information or label (when nearest color used)
3	Red color value
4	Blue color value

For example, to look for red colored blobs you could now use

```
labels = GetArrayVariable("BLOB_COLORIZE_LABELS")
for i = 0 to ubound(labels)-1 step 6
  if labels(i+3) = "255" and labels(i+4) = "0" and labels(i+5) = "0" then
    write "Found a red blob" & vbCrLf
  end if
next
```

See Also

[Colorize](#)

Blob Count

This module simply counts the number of non-black pixel groups (blobs) within the current image. This value is then set into a variable called BLOB_COUNT.

Example

BLOB_COUNT = 11



Variables

BLOB_COUNT - number of identified blobs

See Also

[Blob Label](#)
[Blob Filter](#)

Blob Filter

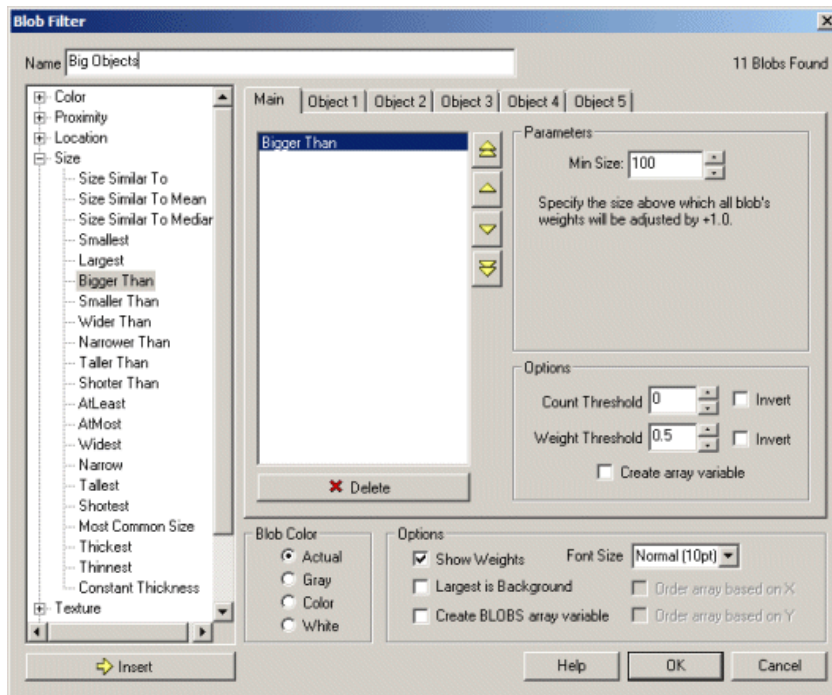
The blob filter module (also known as Particle Filter or Analysis) provides a way to identify a particular blob based on features that describe aspects of the blob itself and in relation to other blobs. The purpose of the blob filter is to provide enough feature descriptions of the desired blob to ensure that the identification of a particular blob is reliable and dependable despite a noisy background.

The blob filter must be used after a blob segmentation module like the [RGB Filter](#), [Segment Colors](#), [Flood Fill](#), [Threshold](#), etc. modules that will group pixels in some meaningful way into blobs of a single color with black pixels being the background. The module you will use to

perform this segmentation will depend on your particular project task. Once the image has been grouped into blobs this Blob Filter module is then used to remove or filter those blobs remaining in the image that are not wanted. For example, if you have an image that was detected for the red color using the [RGB Filter](#) module and the image included a red or orange cone the blob filter can be used to remove all blobs that are too small and not triangular shaped in the image. Thus any red dirt or tree bark while present after the red color detection would be removed by using the blob filter as they would fail a triangular shape test (assuming this is one of the attributes filtered on).

Once you have your image segmented into various blobs you then add in each blob attribute seen below and specify a weight threshold or count threshold to remove those unwanted blobs. Keep in mind that you can add multiple attributes one after the other that will remove blobs along the way in order to finish with the final desired blob. Look for attributes that create a wide distinction between your desired blob and other unwanted blobs (see the Show Weights checkbox to see all weights given the current attribute). Using the checkbox "Create Blob Array" will create the final BLOBS array variable that will contain the COG (center of gravity) of the blobs left after the filtering process. This variable can then be used to react to the presence of a particular object.

Interface



Instructions

1. Name - Name this blob filter for easy identification
2. Select the appropriate feature(s) and insert into the feature list
3. Select Count Threshold - Specify how many blobs should result after the filter (i.e. threshold blob count)
4. Select Weight Threshold - Specify the minimum weight of resulting blobs (i.e. threshold blob weight)
5. Create array variable - If you would like to get an array that shows how relevant the blob is to the current filter selection select this checkbox. A variable array will be created for each blob that indicates what weight that blob has towards the current filter criteria.
6. Invert Threshold / Invert Count - If you want to invert the selection (i.e. reverse the current selections) click on the invert selections. Thus if you had 1 big blob selected the invert will change that list to be all the blobs minus the one big blob.
7. Blob Color - Adjust how the resulting blobs should be colored after the module. "Actual" means no change in blob coloring, "Gray" will color each blob based on its final weight, i.e. higher weighted blobs will appear brighter than others. "Color" refers to labeling each blob with a different color so that they can easily be distinguished. "White" refers to creating a mask regardless of the current blob colors.
8. Show Weights - Click *Show Weights* to indicate each blob's final weight. If the image appears crowded you can either change the font size or just click on one of the blobs in the main RoboRealm image window to isolate the value printing for just that blob.
9. Largest is background - In the case of images that contain blobs that are not white there may be cases where the background is not perfectly black. When this happens you can select this checkbox to ensure that the largest blob is treated as the background and thus not reported as part of the active blobs.
10. Create BLOBS array - If you would like to create an array that contains the resulting center coordinates of all detected blobs (for use in other modules like the scripting modules) select the "Create BLOBS variable array". This creates a floating point array that contains the X and Y coordinates of the center of gravity of each blob that makes it through the filters. A second variable called BLOB_COUNT is also created that indicates how many blobs have passed through the filter. This value will be 1/2 the size of the BLOBS array (since the array has

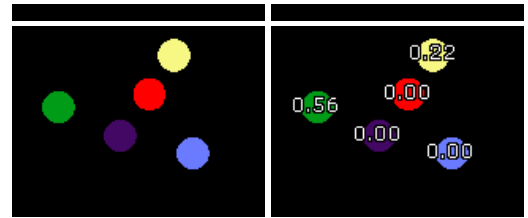
created that indicates how many blobs have passed through the filters. This value will be 1/2 the size of the BLOBS array (since the array has 2 entries per blob).

Blob Features

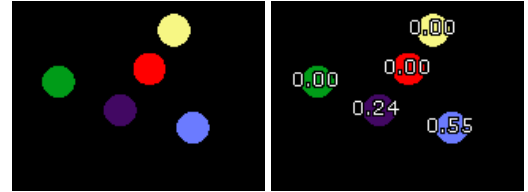
Note that for many comparisons the objects COG defines the point at which comparisons are made.

	Source Image	Filtered Image
Color		
RGB Closest to #FF0000 - Blob with color closest to specified color gets highest weight		
Brightest - Blob with highest intensity gets highest weight		
Brightness - Blob with intensity above specified weight is preserved (intensity is 0 - 255 corresponding to 0.0 - 1.0)		
Darkest - Blob with lowest intensity gets highest weight		
Darkness - Blob with intensity below specified weight is preserved (intensity is 0 - 255 corresponding to 0.0 - 1.0)		
Brighter Than (150) - Blobs with lower intensity than that specified are removed		
Darker Than (150) - Blobs with higher intensity than that specified are removed		
Color Amount - Blobs with more color/hue regardless of which color are given higher weights		
Red Amount - Blobs with more red are given higher weights		

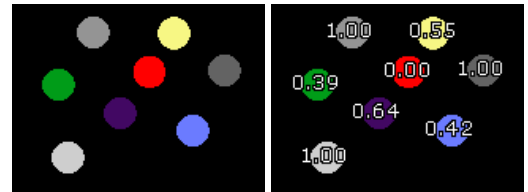
Green Amount - Blobs with more green are given higher weights



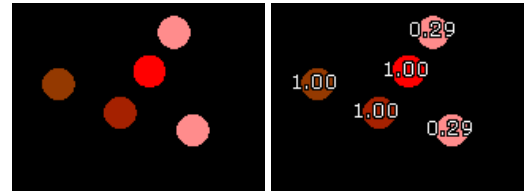
Blue Amount - Blobs with more blue are given higher weights



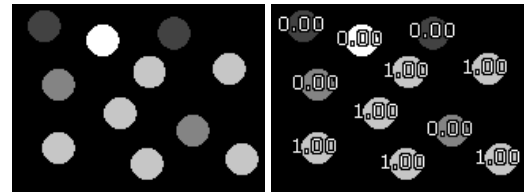
Gray Amount - Blobs with more less color are given higher weights



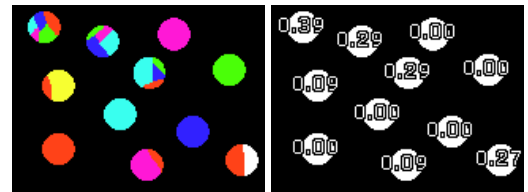
Saturation - Blobs with richer color are given higher weights



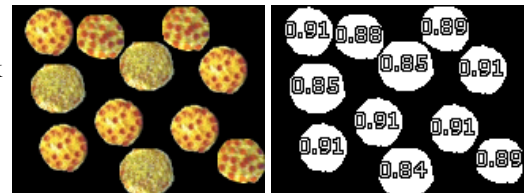
Most Common Intensity - Blobs with intensity most similar to the intensity mode are given higher weights



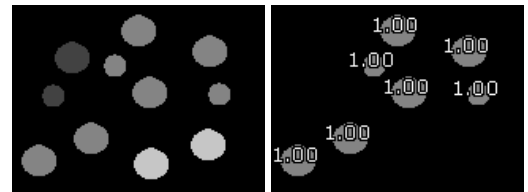
Color Variance - Blobs with most changes in color within the blob are given higher weights. Note this requires a mask to be used in the Blob Filter module otherwise different colors will be counted as isolated blobs. Use a mask to isolate blobs and refer back to the colored image.



Average Histogram - Blobs with a histogram (intensity distribution) most similar to the average histogram of all blobs are given higher weights. Note this requires a mask to be used in the Blob Filter module otherwise different intensities within a blob will be counted as isolated blobs. Use a mask to isolate blobs and refer back to the colored image.

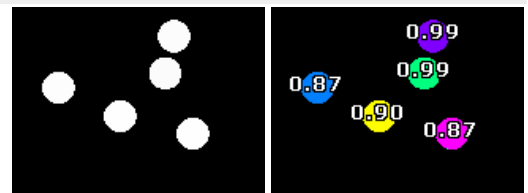


Intensity Similar To Mean - Blobs with intensity most similar to mean intensity of all blobs gets highest weight



Proximity

Nearest - Blobs nearest to each other get high weights. You can also enter in (x,y) coordinates and variables using [] notation.

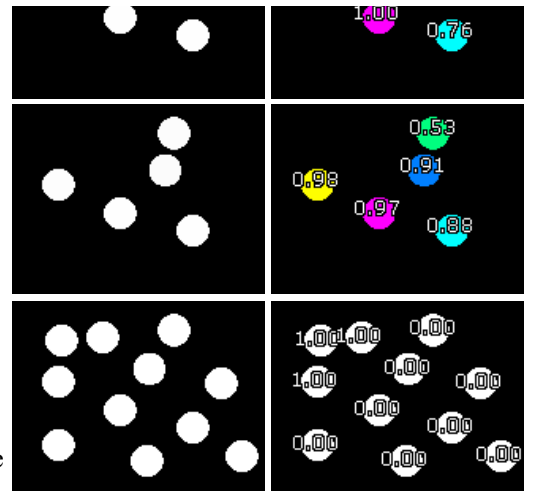


Nearest X - Blobs nearest to the specified X coordinate get higher weights. (70 in



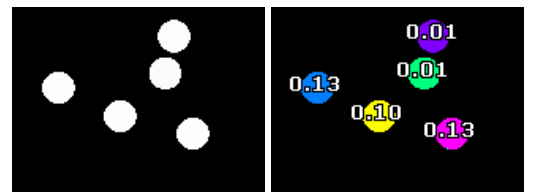
this example)

Nearest Y - Blobs nearest to the specified Y coordinate get higher weights. (60 in this example).

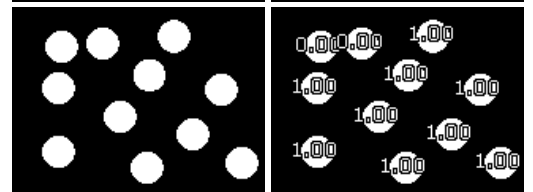


Nearer Than - Blobs nearer to other blobs by a specified amount are kept while the others are removed.

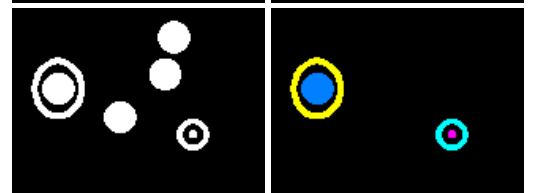
Furthest - Blobs furthest from each other i.e. most isolated will get higher weights. You can also enter in (x,y) coordinates and variables using [] notation.



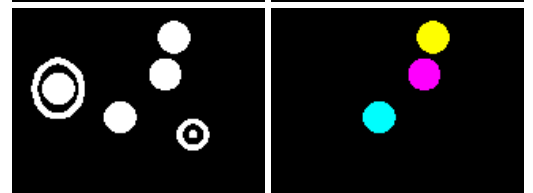
Further Than - Blobs further than a specified amount from other blobs are kept, blobs too close are removed.



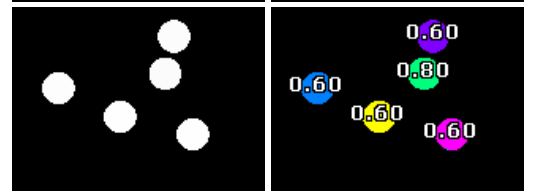
COG Nearer Than - Blobs with COG's nearer to other blob COG's by a specified amount are kept, others are removed.



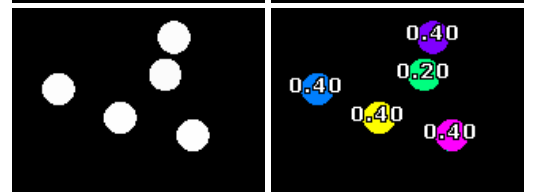
COG Further Than - Blobs with COG's further than a specified amount from another blob's COG are kept, blobs with close COGs are removed.



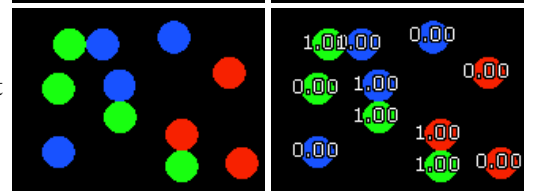
Most Neighbors - Blobs with many close neighboring blobs will get higher weights



Least Neighbors - Blobs with no close neighbors will get higher weights

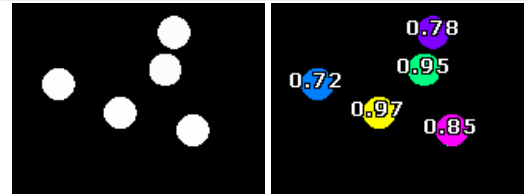


Touches Another - Blobs that touch other blobs (of different color/intensity) will get a weight of 1.0

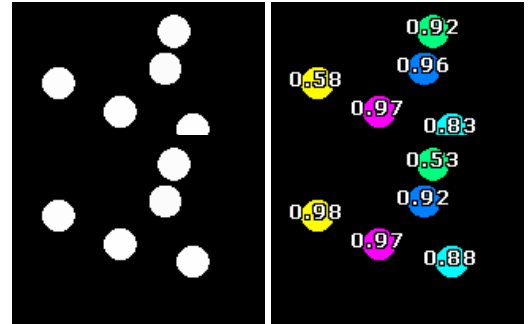


Location

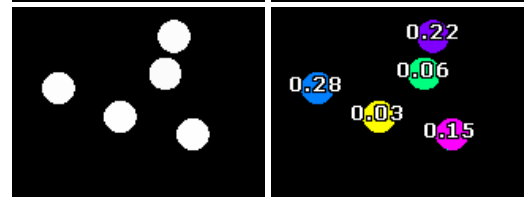
Center - Blobs closest to the center of screen get higher weights



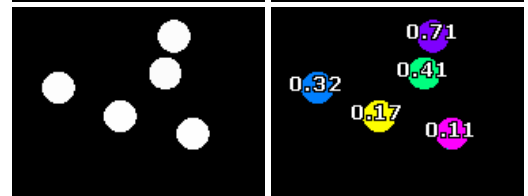
Horizontal Center - Blobs closest to the horizontal center only of screen get higher weights



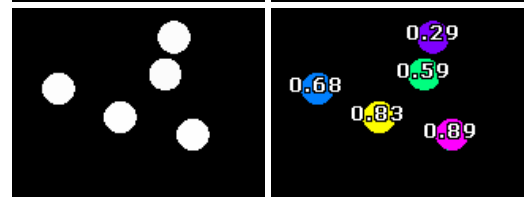
Vertical Center - Blobs closest to the vertical center only of screen get higher weights



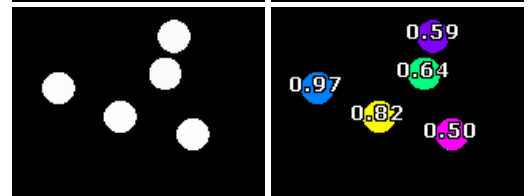
Outward - Blobs furthest from center (closest to screen border) get higher weights



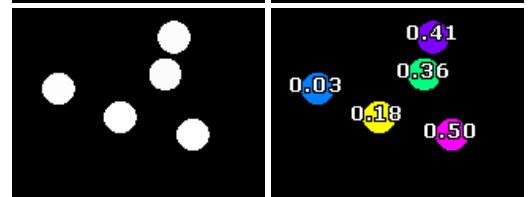
Top - Blobs closest to the top of screen get higher weights



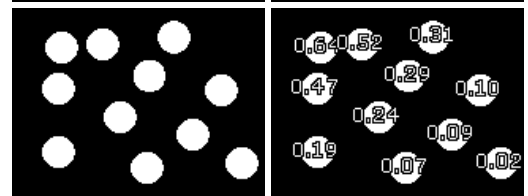
Bottom - Blobs closest to the bottom of the screen get higher weights



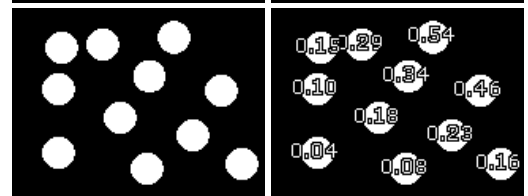
Left - Blobs closest to the left of the screen get higher weights



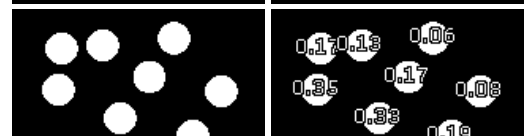
Right - Blobs closest to the right of the screen get higher weights



Top Left - Blobs closest to the top left of screen get higher weights



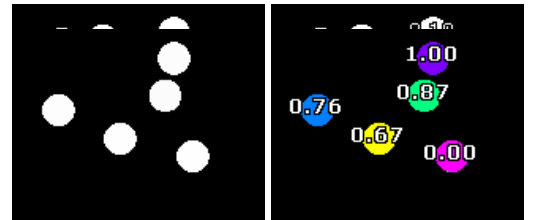
Top Right - Blobs closest to the top right of screen get higher weights



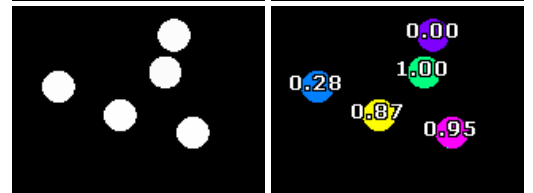
Bottom Left - Blobs closest to the bottom left of screen get higher weights



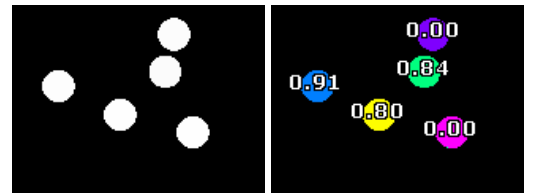
Bottom Right - Blobs closest to the bottom right of screen get higher weights



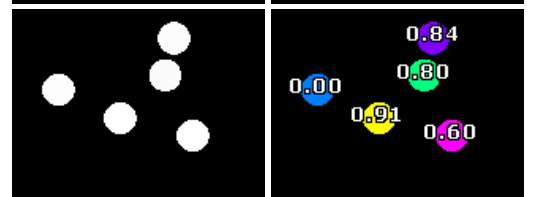
Above - Blobs above their closest neighbor get higher weights



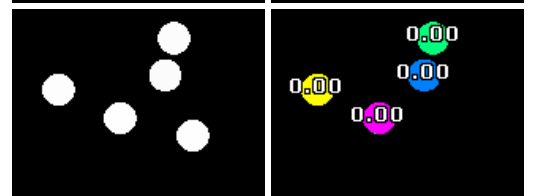
Below - Blobs below their closest neighbor get higher weights



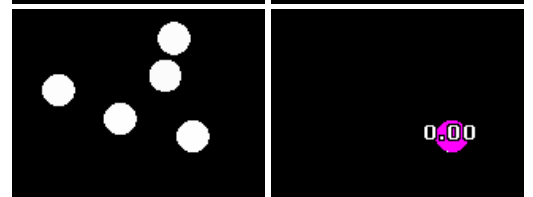
LeftOf - Blobs to the left of their closest neighbor get higher weights



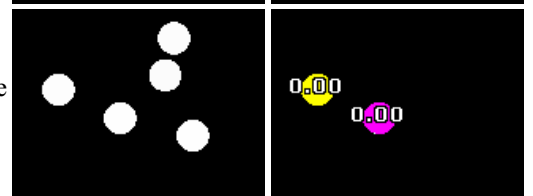
RightOf - Blobs to the right of their closest neighbor get higher weights



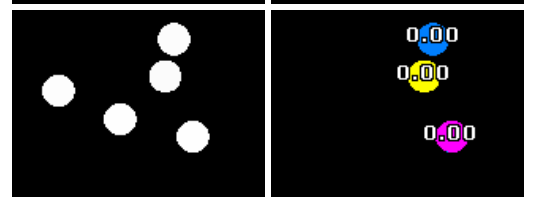
AboveY (50) - Blobs above the specified Y coordinate are kept, those below are removed



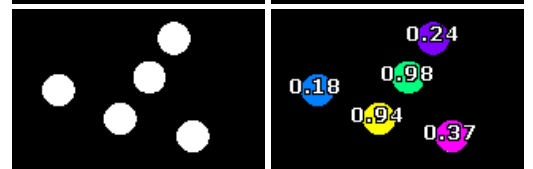
BelowY (50) - Blobs below the specified Y coordinate are kept, those above are removed



LeftOfX (80) - Blobs left of the specified X coordinate are kept, those right of X are removed



RightOfX (80) - Blobs right of the specified X coordinate are kept, those to the left are removed

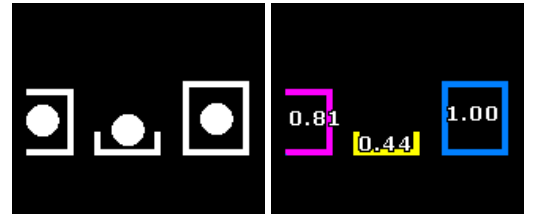


Between - Blobs between close neighboring blobs get higher weights

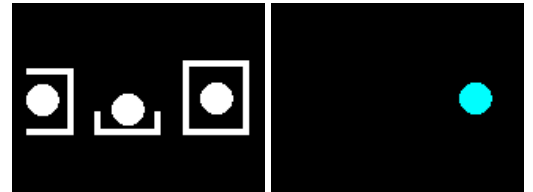
Within (top 3) - Blobs located within other blobs get higher weights



Around (top 3) - Blobs surrounding other blob get higher weights

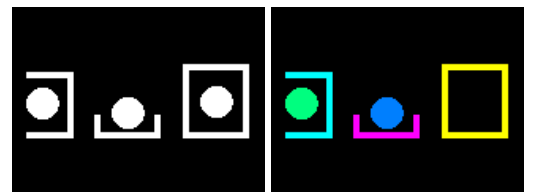


Overlaps - Blobs overlapping blobs in other Object tabs are preserved



Inner - Only blobs located within other blobs are kept

Outer - Only blobs surrounding other blobs are kept



Touches Border - Only blobs that touch the image border are kept



Touches Bottom Border - Only blobs that touch the bottom image border are kept



Touches Top Border - Only blobs that touch the top image border are kept



Touches Left Border - Only blobs that touch the left image border are kept



Touches Right Border - Only blobs that touch the right image border are kept



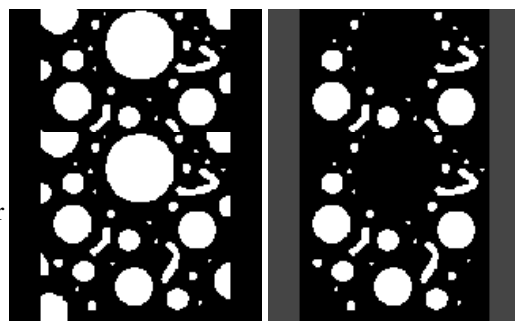
Avoids Border - Only blobs that do not touch (but avoid) the image border are



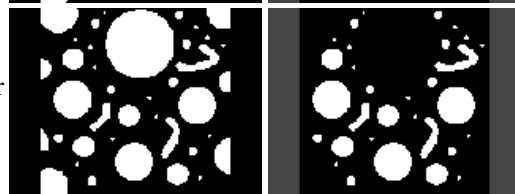
kept. I.e. objects completely contained within the current image.



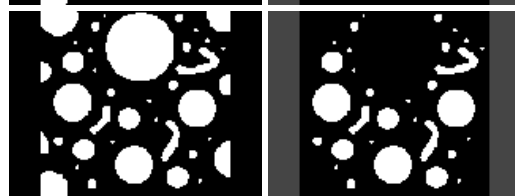
Avoids Bottom Border - Only blobs that do not touch (but avoid) the bottom image border are kept.



Avoids Top Border - Only blobs that do not touch (but avoid) the top image border are kept.



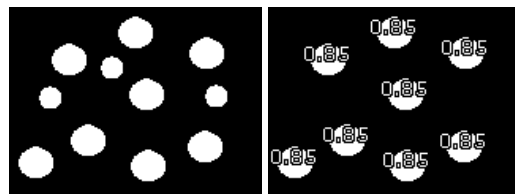
Avoids Left Border - Only blobs that do not touch (but avoid) the left image border are kept.



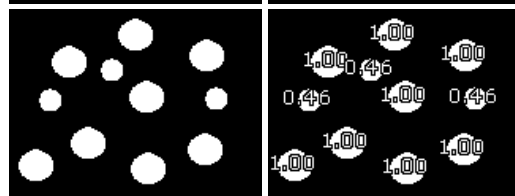
Avoids Right Border - Only blobs that do not touch (but avoid) the right image border are kept.

Size

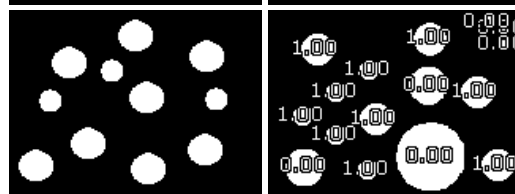
Size Similar To - Blobs closest to specified size have higher weights



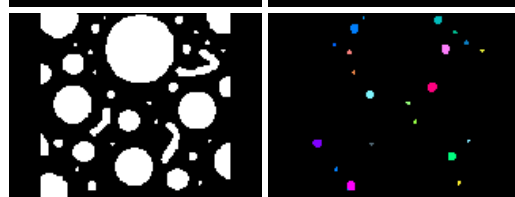
Size Similar To Mean - Blobs closest to the mean blob size have higher weights



Size Similar To Median - Blobs closest to the median (middle) blob size have higher weights



Most Common Size - Blobs with similar sizes to other blobs get higher weights. This allows for multiple common sizes to exist.



Smallest (above 0.98) - Smaller blobs get higher weights



Largest (top 1) - Larger blobs get higher weights

Bigger Than (200) - Blobs bigger than the specified size are kept, those below are removed

Smaller Than (10) - Blobs smaller than the specified size are kept, those above are removed

AtLeast 100 (top 10) - Blobs above the specified amount get decreasing weights according to their size

AtMost 100 (top 10) - Blobs smaller than the specified amount get decreasing weights according to their size

Widest (0.10) - Wider/thicker blobs (horizontal size) get higher weights

Narrow (top 10) - Narrow/thinner blobs (horizontal size) get higher weights

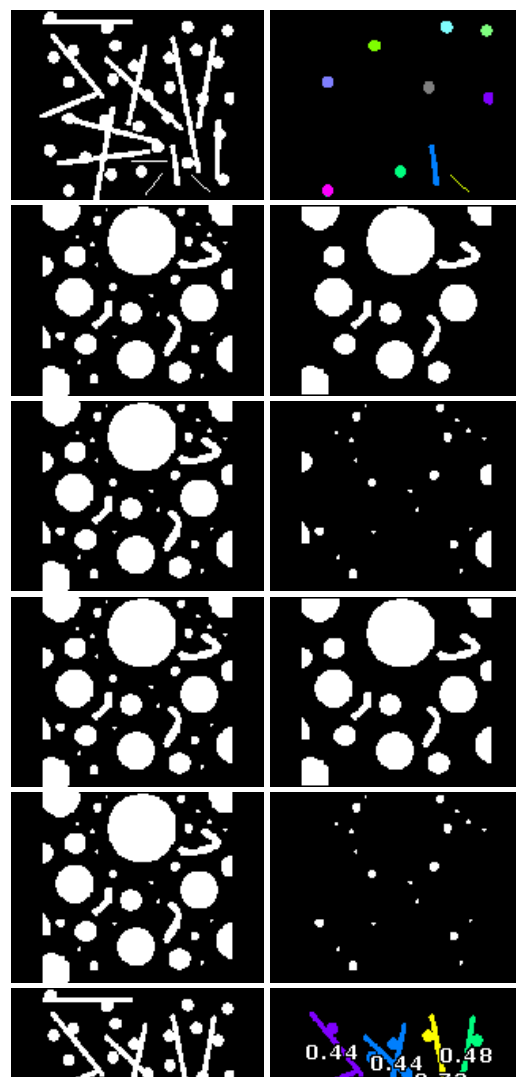
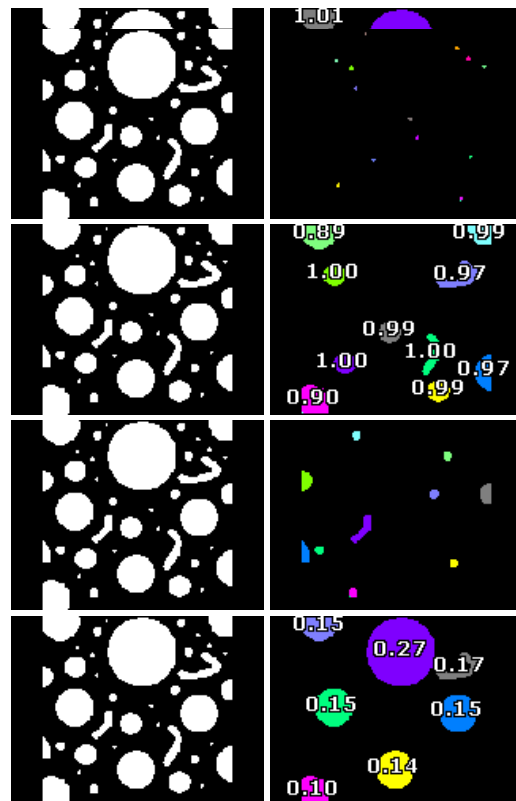
Wider Than (10) - Blobs thinner than the specified amount are removed

Narrower Than (10) - Blobs wider than the specified amount are removed

Taller Than (10) - Blobs shorter than the specified amount are removed

Shorter Than (10) - Blobs taller than the specified amount are removed

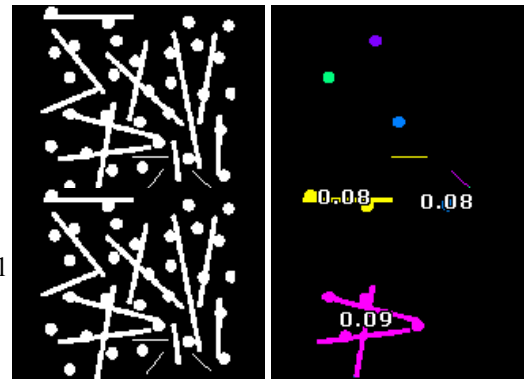
Tallest (top 10) - Taller blobs (vertical size)



Tallest (top 10) - Taller blobs (vertical size)



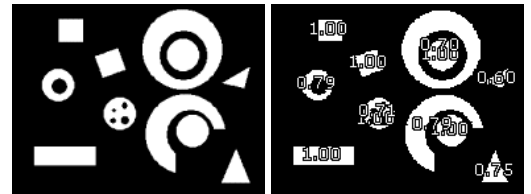
Shortest (top 5) - Shorter blobs (vertical size) get higher weights



Thickest (top 3) - Thicker blobs (minimum distance from object perimeter to medial axis) get higher weights



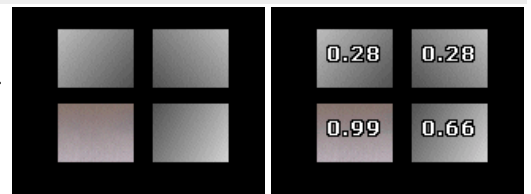
Thinnest (top 3) - Thinner blobs (minimum distance from object perimeter to medial axis) get higher weights



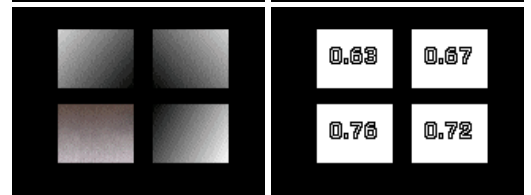
Constant Thickness - Blobs with more constant thickness get higher weights.

Texture

Gradient (270 degrees) - Blobs with specified surface gradient (shading) get higher weights



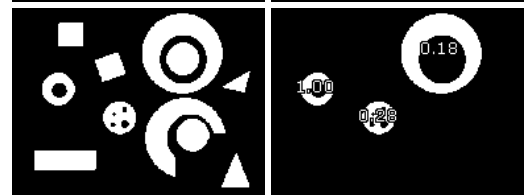
Gradient Strength - Blobs with weight by their surface gradient (shading) strength



Number of holes (4) - Blobs with specified number of holes are kept, the others are removed



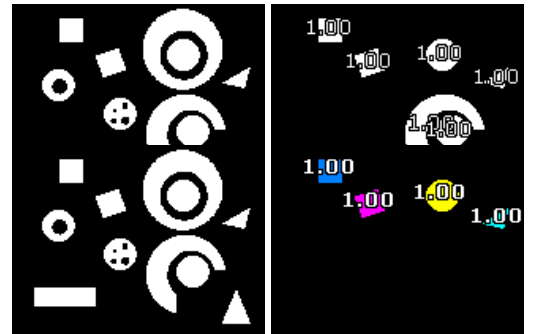
Hole Size (100) - Blobs with specified hole size get higher weights



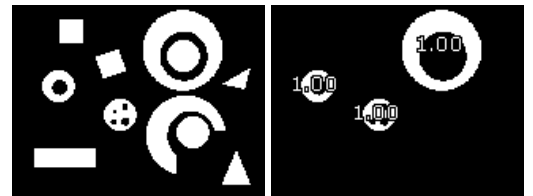
Most Holes (top 4) - Blobs with many holes (regardless of hole size) get higher weights



At Most Holes (0) - Blobs with at most the specified number of holes get higher weights



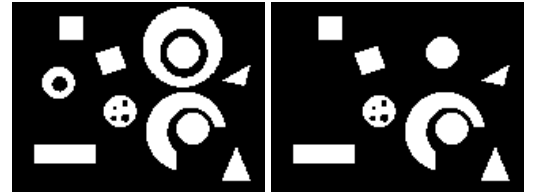
Least Holes (top 4) - Blobs with fewest holes (regardless of hole size) get higher weights



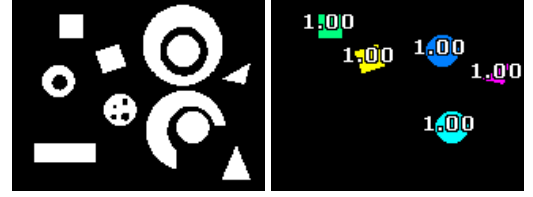
At Least Holes (1) - Blobs with at least the specified number of holes get higher weights



Hole Bigger Than (50) - Blobs with a hole bigger than the specified amount get higher weights



Hole Smaller Than (50) - Blobs with a hole smaller than the specified amount get higher weights

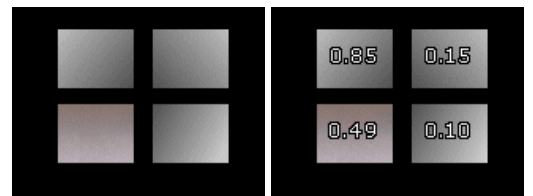


Densest - Blobs that are more solid (least or smallest holes) get higher weights



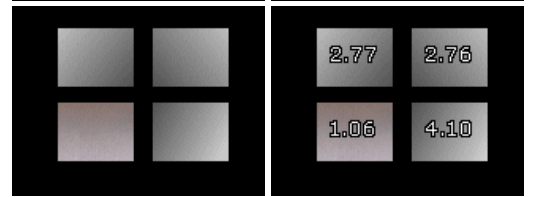
Least Densest - Blobs that are least solid (many or few large holes) get higher weights.

Density Ratio - Blobs that are the specified density ratio (area of blob versus hole) get higher weights.

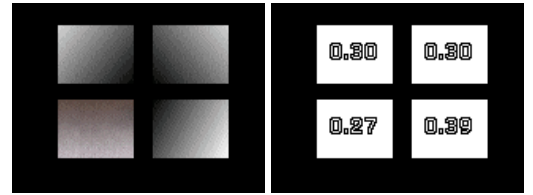


Orientation - Blobs that are oriented in the specified degree will get higher weights.

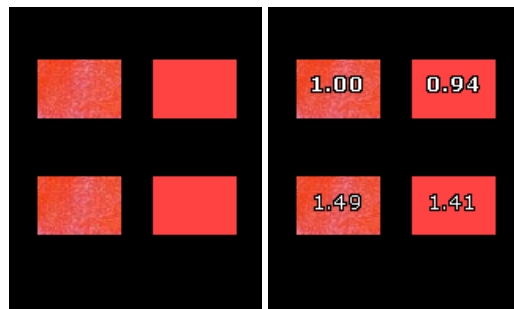
Orientation Strength - Blobs that have a strong orientation will get higher weights.



Contrast - Blobs that have a stronger contrast will get higher weights.

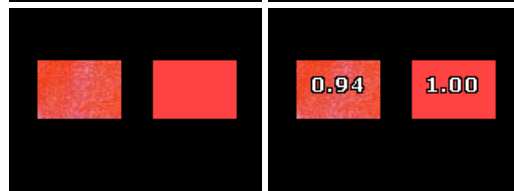


Roughest - Blobs that have more texture (based on the specified image) than other blobs get higher weights



Roughness - Blobs are weighted based on roughness as an absolute value

Smoothest - Blobs that have less texture (based on the specified image) than other blobs get higher weights



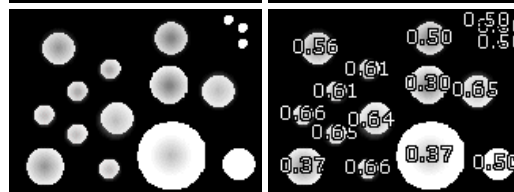
Smoothness - Blobs are weighted based on smoothness as an absolute value



Variance - Blobs are weighted based on variance of pixel intensities relative to mean intensity of the blob



Center Mean Ratio - Blobs are weighted based on the difference between the center pixel and the average intensity of the blob

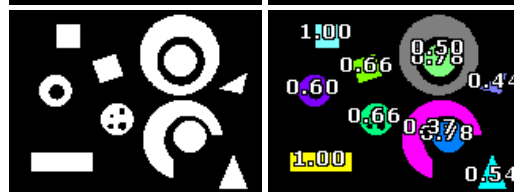


Shape

Aspect (1.0) - Blobs with the specified aspect ratio (width/height) get higher weights



Boxed - Blobs that best fit their square bounding box get higher weights



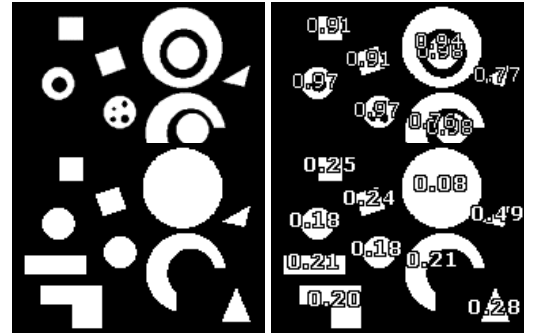
Circular Area - Circular blobs get higher weights



Circular Deviation - Blobs with circular perimeters (ignoring holes) get higher weights

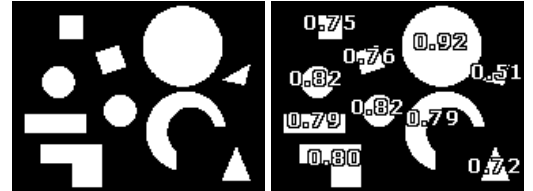


Circular perimeter - Blobs with circular perimeters (ignoring holes) get higher weights

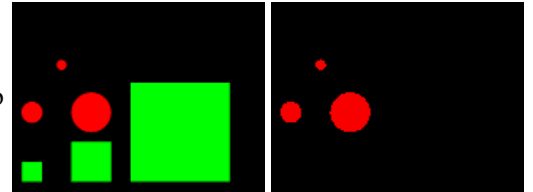


Perimeter / Area - Calculates perimeter / area ratio for each blob

Area / Perimeter - Calculates area / perimeter ratio for each blob

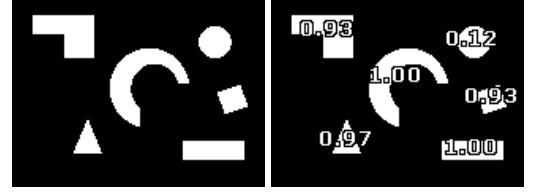
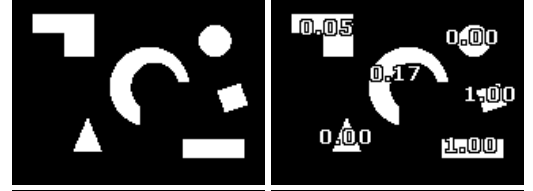


Area / (Perimeter ^ 2) - Calculates area / (perimeter*perimeter) ratio for each blob



Quadrilateral Sides - Estimates a perfect rectangle from blob outline and then determines squareness of a blob by comparing how well two sides of the ideal extracted rectangle compare in length. The more rectangular the blob the more the opposing sides will match in length

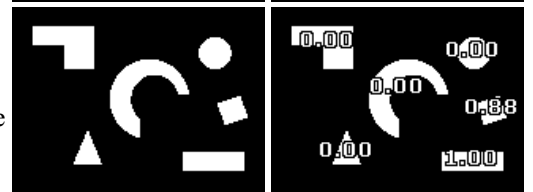
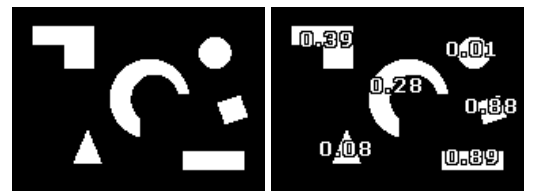
Quadrilateral perimeter - Estimates a perfect rectangle from blob outline and then compares how the blob's perimeter matches the perimeter determined by the ideal four sides



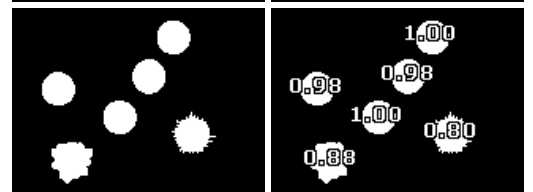
Quadrilateral Area - Estimates a perfect rectangle from blob outline and then compares how well does the blob's area matches the area determined by the ideal four sides

Quadrilateral Deviation - Estimates a perfect rectangle from blob outline and then compares how well does the blob's outline deviates from the ideal extracted rectangle

Squareish - Blobs appearing like a square (perimeter is 4*side) get higher weights



Smoothness - Blobs with smooth perimeters get higher weights



Straightest (top 5) - Blob shape is mainly in a single direction

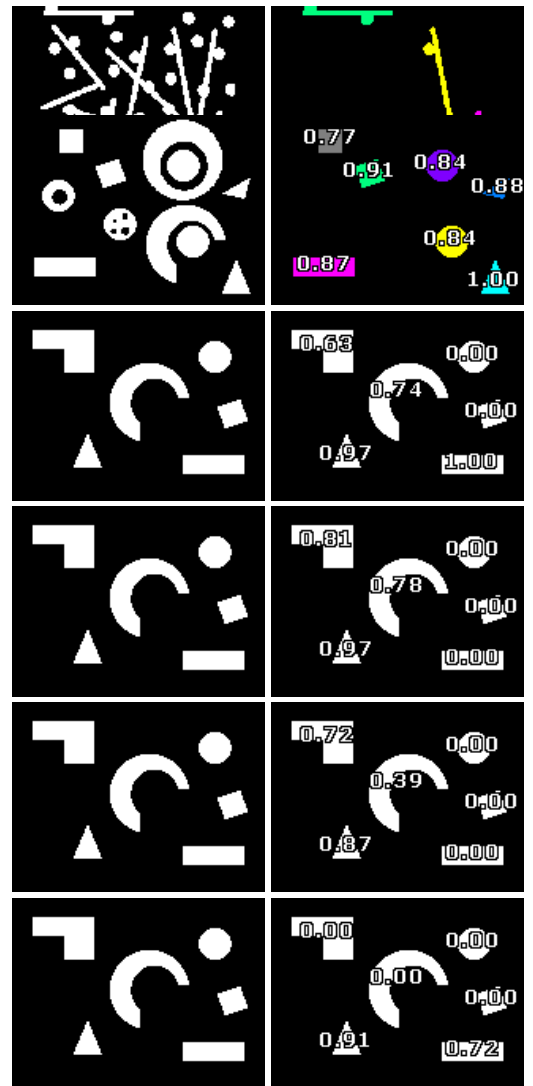
Triangular - Triangular shaped blobs get higher weights based on how well their area and perimeter agree based on a triangle


Triangle Sides - Estimates a perfect triangle from blob outline and then determines triangularness of a blob by comparing how well sides of the ideal extracted triangle compare in length

Triangle perimeter - Estimates a perfect triangle from blob outline and then determines how well does the blob's perimeter matches the perimeter determined by the ideal three sides

Triangle Area - Estimates a perfect triangle from blob outline and then determines how well does the blob's area matches the area determined by the ideal three sides

Triangle Deviation - Estimates a perfect triangle from blob outline and then determines how well does the blob's outline deviates from the ideal extracted triangle



Bitmap using  - Blobs most similar to shape bitmap get higher weights

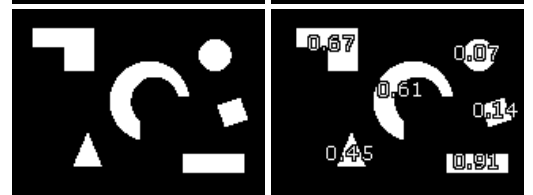


Moments of Inertia

Axis Aspect (1.0) - Blobs with the specified aspect ratio (minor axis/major axis) get higher weights



Eccentricity - ratio of the distance between the foci of the ellipse representation of the image and its major axis length



Invariant_1 - the first Hu invariant centralized moment

Invariant_2 - the second Hu invariant centralized moment

Normalized_02 - the 02 scale invariant centralized moment

Normalized_20 - the 20 scale invariant centralized moment

Normalized_03 - the 03 scale invariant centralized moment

Normalized_30 - the 30 scale invariant centralized moment

Normalized_11 - the 11 scale invariant centralized moment

Normalized_12 - the 12 scale invariant centralized moment

Normalized_21 - the 21 scale invariant centralized moment

Kurtosis_X - a measure of the "peakedness" of the blob shape along the X Axis

Kurtosis_Y - a measure of the "peakedness" of the blob shape along the Y Axis

Skewness_X - a measure of the asymmetry of the blob shape along the X Axis

Skewness_Y - a measure of the asymmetry of the blob shape along the Y Axis

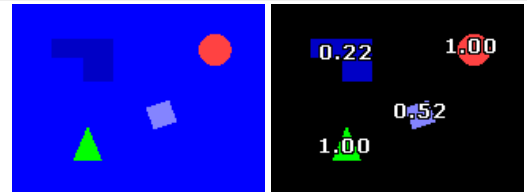
Variance_X - variance around the object center given the appropriate axis

Variance_Y - variance around the object center given the appropriate axis

Variance_XY - variance around the object center given the appropriate axis

Edge

Strength - Calculates how strong the blobs edges are using the specified source image. This is a measure of how stable the shape of the blob is



Spiked - Calculates how unsmooth the edge of the blob is. This is done by determining how many pixels belong to a straight edge (vertical, horizontal or diagonal).

Grittiness - Similar to Spiked but eliminated overlapping count of edge shape.

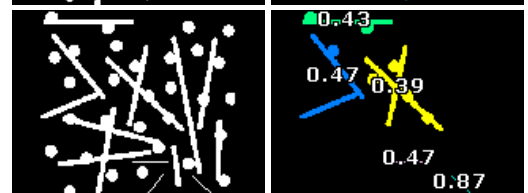
Variance - Calculates a measure of how different the source pixels of the blob are to their average along the blob edge.

Orientation

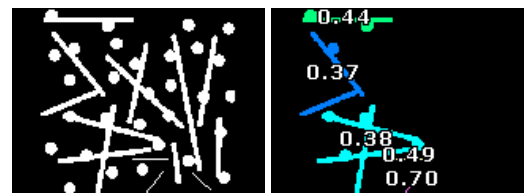
Diagonal - Diagonal oriented blobs get higher weights



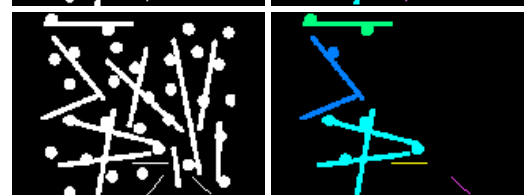
Diagonal Left - Left slanted blobs get higher weights



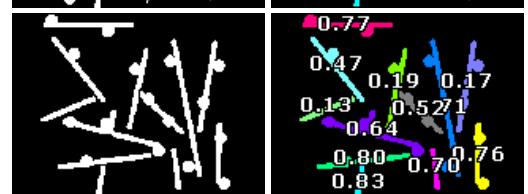
Diagonal Right - Right slanted blobs get higher weights



Horizontal - Horizontal oriented blobs get higher weights



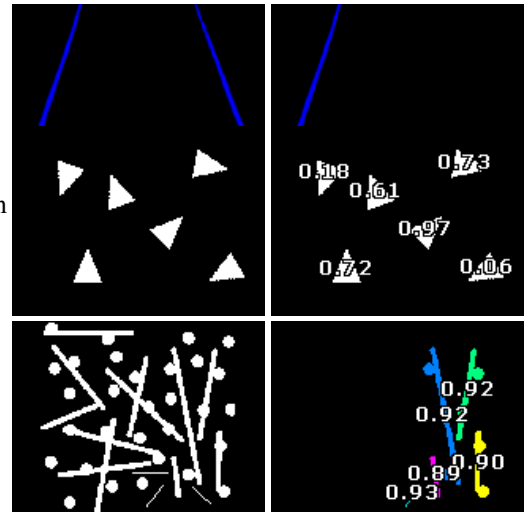
Max Radius Degree - Determines furthest point in blob from COG of blob and then ranks blobs based on the angle of the line from the COG to that furthest point (45 degrees)



Degree Range - Filters out blobs that do not fall within the specified orientation range.

Triangle Degree - Estimates a perfect triangle from blob outline and then ranks them based on the triangle's orientation (45 degrees)

Vertical - Vertical oriented blobs get higher weights



Other

Join - Used to ensure that an objects detected in one tab filter are also detected in another. Can be used to join blobs across tab filters.

Variables

BLOBS - contains the COGX and COGY of all detected blobs. You can use this array within a VBScript program to further process the blobs. For example, the below script will calculate the distance to the first two detected blobs. Note that the blob filter should allow at least two blobs to exist for this script to work correctly.

BLOB_FILTER_START_COUNT - the number of blobs entering into the blob filter module.

BLOB_FILTER_END_COUNT - the number of blobs remaining after all filters are applied

```
blobs = GetArrayVariable("blobs")

if isArray(blobs) then
  if ubound(blobs) >=2 then

    xdiff = blobs(0)-blobs(2)
    ydiff = blobs(1)-blobs(3)

    dist = sqrt((xdiff*xdiff) + (ydiff*ydiff))

    SetVariable "distance", dist

  end if
end if
```

See Also

[Blob Size](#)
[Blob Inspection](#)
[Shape Match](#)

Blob Label

The Blob Label module provides a way to identify each blob by coloring it with a pseudo color. This module can help you understand how many and what blobs are being identified by the system for further processing. This module is typically used after some initial processing such as thresholding or other binarizing module.

Instructions

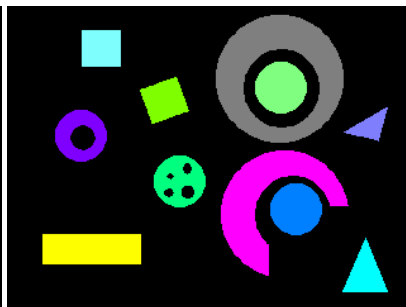
1. Insert the blob label model 2. You should see each identified blob as a separate color. This can help you

Example

Binarized Image



Blobs Identified



Variables

BLOB_LABEL_COUNT - number of identified blobs
BLOB_LABEL_X - x coordinate of each labeled blob
BLOB_LABEL_Y - y coordinate of each labeled blob
BLOB_LABEL_MIN_X - minimum x coordinate of each labeled blob
BLOB_LABEL_MIN_Y - minimum y coordinate of each labeled blob
BLOB_LABEL_MAX_X - maximum x coordinate of each labeled blob
BLOB_LABEL_MAX_Y - maximum y coordinate of each labeled blob
BLOB_LABEL_R - red color of each labeled blob
BLOB_LABEL_G - green color of each labeled blob
BLOB_LABEL_B - blue color of each labeled blob

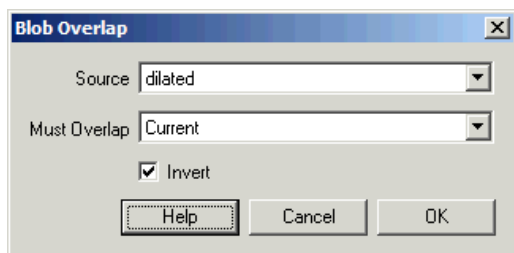
See Also

[Blob Filter](#)

Blob Overlap

The blob overlap module is used to filter existing blobs by their overlapping of other blobs. This module allows separately processed images to be merged together and combine their filtering results.

Interface



Instructions

1. Source - The image which to filter

2. Overlap - The image used to determine which blobs overlap both images
3. Invert - Remove blobs that overlap instead of those that do not.

Example

The following [robofile](#) is used to isolate the star in the following image. The identification process uses the brightness of the star and the fact that it is next to a red and black object (i.e. the color to either side of the star). The red and black areas are detected and dialted (grown) in order to force them to overlap into the space the star occupies. By using the overlapping of the three areas (white area, red area and dark area) we can isolate the star since it has all these properties.

Source

Blob Overlap



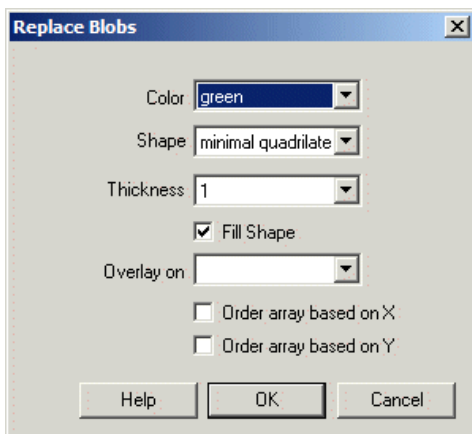
See Also

[Blob Filter](#)

Replace Blob

The replace blob module will replace white blobs present in the current image with the specified color object shape as selected in the interface. The shapes' sizes will be based on the area of the blob being replaced. This module is useful if you want to better define the shape of the blob with a known shape. For example, replacing detected blobs with a better defined circle can aid in subsequent detection or analysis.

Interface



Instructions

1. Color - Select the color the replaced shape will appear in
2. Shape - Select the shape that is used to replace each blob present in the image.

bounding box replaces each blob with it bounding box extents (x_{max} , y_{max} , x_{min} , y_{min})
minimal enclosure replaces each blob with its minimal enclosure parallelogram (rectangle)
circular enclosure replaces each blob with a circle that encompasses the entire blob (aka Circumscribed Circle)
inscribed circle replaces each blob with the largest inner circle that fits into the blob
crosshair marks each blob's center of gravity with a cross hair

- square area replaces each blob with a square of the same area centered at the center of gravity of the blob
- circle area replaces each blob with a circle of the same area centered at the center of gravity of the blob
- triangle area replaces each blob with a triangle of the same area centered at the center of gravity of the blob
- star area replaces each blob with a star of the same area centered at the center of gravity of the blob
- diamond area replaces each blob with a diamond of the same area centered at the center of gravity of the blob
- inner square replaces each blob with the largest Cartesian (not rotated) square that fits in the blob
- inner rectangle replaces each blob with the largest Cartesian (not rotated) rectangle that fits in the blob
- minimal quadrilateral replaces each blob with the largest 4 sides polygon that fits around the blob
- minimal parallelogram replaces each blob with the largest parallelogram that fits around the blob
- ellipse replaces each blob with an ellipse
- best fit rectangle uses a best fit algorithm to create 4 lines that best represent all the edge points of the blob

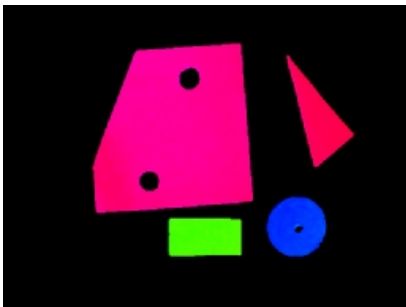
3. Thickness - if drawn as lines this specifies how thick the drawn line should be.
4. Fill Shape - if you want to ignore holes in the object select that the object should be filled in.
5. Overlay - Select what the resulting graphic will be drawn on.

Source - the original captured image
 Current - the current image seen at this point in the processing pipeline
 XXX - other images created by the [Marker](#) module.

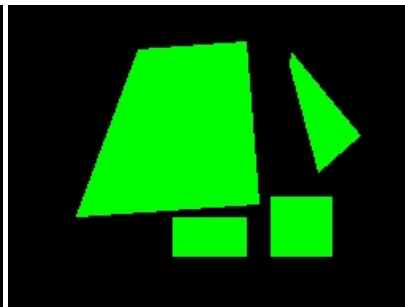
6. Order - Each replace creates variables (see below) that contain the values that represent the replaced blob. The order of these blobs and thus their numerical results can be changed by selecting the order checkboxes. For example, if you want to order the array in increasing X then select the "Order array based on X" which will sort the blobs in increasing X order. This can be handy if you are relying on a specific order of the blobs in order to continue processing in subsequent modules or scripts.

Example

Source



Minimal Quadrilateral



The example above shows an image RGBFiltered, Dilated and then blob replaced with red circles. Notice how the red traffic light is now better pronounced than in the source image.

Variables

BOUNDING_COORDINATES - p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y coordinates of the Bounding Box of the replaced blob when this is selected. Note that this is an ARRAY with 8 numbers per blob.

CROSSHAIR_COORDINATES - x, y coordinates of the CrossHair that replaces a blob when this replace method is selected. Note that this is an ARRAY with 2 numbers per replaced blob.

SQUARE_COORDINATES - p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y coordinates of the Square that replaces the blob when this is selected. Note that this is an ARRAY with 8 numbers per blob.

CIRCLE_COORDINATES - x , y , radius of the Circle that replaces the blob when this is selected. Note that this is an ARRAY with 3 numbers per blob.

TRIANGLE_COORDINATES - x , y , base of the Triangle that replaces the blob when this is selected. Note that this is an ARRAY with 3 numbers per blob.

STAR_COORDINATES - x , y , radius of the Star that replaces the blob when this is selected. Note that this is an ARRAY with 3 numbers per blob.

DIAMOND_COORDINATES - x , y , radius of the Diamond that replaces the blob when this is selected. Note that this is an ARRAY with 3 numbers per blob.

BFR_COORDINATES - $p1x$, $p1y$, $p2x$, $p2y$, $p3x$, $p3y$, $p4x$, $p4y$ coordinates of the Best Fit Rectangle of the replaced blob when this is selected. Note that this is an ARRAY with 8 numbers per blob.

MEQ_COORDINATES - $p1x$, $p1y$, $p2x$, $p2y$, $p3x$, $p3y$, $p4x$, $p4y$ coordinates of the Minimal Quadrilateral of the replaced blob when this is selected. Note that this is an ARRAY with 8 numbers per blob.

MER_COORDINATES - $p1x$, $p1y$, $p2x$, $p2y$, $p3x$, $p3y$, $p4x$, $p4y$ coordinates of the Minimal Enclosed Rectangle for each detected blob when the . Note that this is an ARRAY with 8 numbers per blob.

MEP_COORDINATES - $p1x$, $p1y$, $p2x$, $p2y$, $p3x$, $p3y$, $p4x$, $p4y$ coordinates of the Minimal Enclosed Polygon for each detected blob. Note that this is an ARRAY with 8 numbers per blob.

See Also

[Convex Hull](#)
[Smooth Hull](#)

Blob Separate

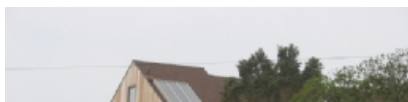
The Blob Separate module is used to separate blobs of different colors into distinct objects such that they can be processed as individual blobs by the morphological routines which assume a binary colored image. For instance, if an image is segmented into colored blobs simply by reducing the number of colors used there will be many blobs that touch each other. Applying a morphological module like "erode" will assume that all non-black pixels constitute a single blob of connected pixels. Using the blob separate module prior to the erode module will ensure that the erode module treats each differently colored blobs as a separate entity.

Instructions

1. Insert the separate blob module. Blobs of similar colors will be separated from other colored blobs by a black line

Example

Source



Blob Separated after Flood Fill





See Also

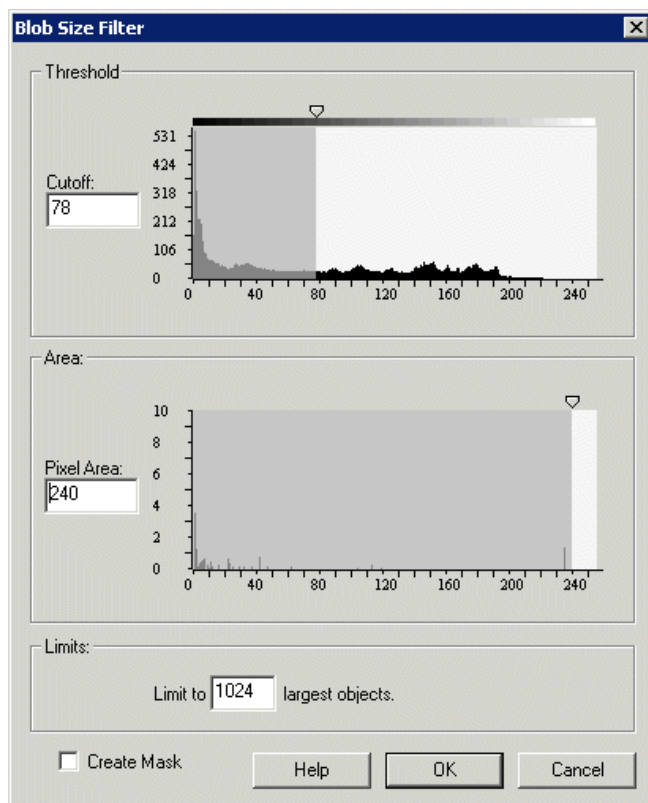
[Flood Fill](#)
[Blob Border](#)
[Border](#)

Blob Size Filter

The Blob Size filter can be used to remove objects below a certain size. The Threshold histogram seen in the Blob Size interface is used to threshold the image into object/no object values. Once objects have been identified you can use the 'size histogram' slider to remove objects below a certain pixel size. Note that the pixel size is the objects pixel area.

Note that this module includes an intensity threshold function. If you want to use color to create a thresholded image see the [Threshold](#) function.

Interface



Instructions

1. Intensity Threshold - Select the appropriate threshold cutoff. All pixel intensities that fall below the cutoff will be set to black. This allows the module to determine separated blobs and their respective size. Note that if you already have a binary image specification of the cutoff may not be needed.
2. Object Area - Select the appropriate blob area to remove from the image. Enter the value into the "pixel area" textbox if above 256 pixels or drag the move over the small cursor above the histogram to move the level.
3. Limits - You can also chose to remove all but the X largest blobs. Enter that number into the "limit to" textbox. Note that if you specify 1 in the "limit to" textbox the pixel area value is essentially ignored.

4. Create MASK - If you want to create a binary black/white image from the results select the "Create MASK" check box.

5. Treat as Color Image - Indicates to the module to use the objects color to determine when a new object is encountered. If this is not checked any pixel that is non-zero is considered object whereas any black pixel is not. Selecting this checkbox will caused objects next to each other with different colors (such as the Color Mask after the [Color Filter](#) module) will cause the objects to be analyzed separately.

Example

Original Image



Blobs < 240 removed



Variables

BLOB_SIZE_COUNT - The number of remaining objects after the blob size module is done.

See Also

[Threshold](#)

Blob Split

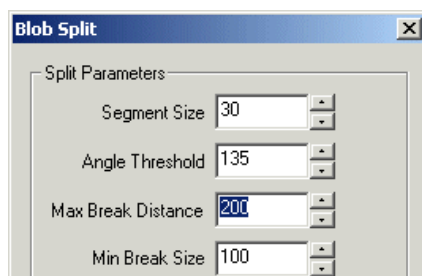


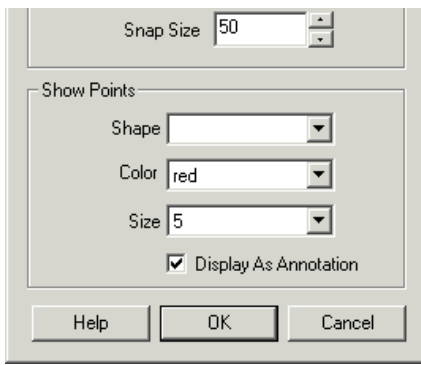
The Blob Split is functionally similar to the [Open](#) and [Watershed](#) modules which seek to split blobs into reasonable separate blobs in order to allow for additional processing on single objects. Blob splitting is very useful when objects cannot be segmented into separate objects and are touching each other. Running any statistics or shape matching on connected objects would cause incorrect results. The Blob Split module offers more control over the splitting process than previous modules.

The Blob Split module is different than the previous morphology based modules in that it will pay much more attention to the object curvature in order to deduce what a good split segment would be. There are many more parameters on the Blob Split that provide much better control over what objects are split.

For example, the Watershed module will separate even the smallest object from its parent whereas the Blob Split module can be configured to only separate the largest.

Interface





Instructions

1. **Segment Size** - The Blob Split module determines an objects curvature and uses this to determine where an effective split point can be. In performing this calculation X number of points before and after any point need to be taken into account. The larger the segment size is the smoother the object curvature will be and the fewer split points will be created. Thus increasing the segment size will reduce the sensitivity of the splits. The smaller the segment size the smaller the bumps in the object need to be in order to elicit a split point.
2. **Angle Threshold** - Once the object outline has been smoothed each point's angular intensity (the amount of curvature) will be calculated. Reducing the Angle Threshold will remove those points with less sharp angles, whilst increasing the angle threshold will allow more and more straighter points to be included as split points.
3. **Max Break Distance** - The maximum distance between two split points that will be allowed to create a split. Reducing this number will prevent very thick joints from splitting, increasing it will allow thicker joints to be split.
4. **Min Break Size** - The minimum size that break can create. For example, if you have a small little node on your object outline it will normally be split from the larger parent object. Increasing the Min Break Size will prevent this smaller object from splitting from the parent. Thus an split object will have to still be larger in size than the Min Break Size in order to be split from its parent. Any objects that are smaller will not be split.
5. **Snap Size** - The Blob Split module will determine many split points that indicate potential split points. Some (but not all) of these split points could match to another split point in order to create a good split. In order to ensure that split points do match you can specify a snap size which will cause split points to capture connections from opposing split points.
6. **Show Points** - Specify the desired parameters in order to show the split points. While the split points are not visually needed they are useful when adjusting the above parameters in order to get the desired split results.
7. **Split Line Thickness** - When the module splits blobs it will draw a line between the two detected endpoints. You can specify the thickness of this line to ensure that the blobs are correctly separated with enough distance to avoid connecting the blobs in successive modules.

Example

Source



Blob Split





[Click Here](#) to try the robofile that produced the above results.

See Also

[Open Watershed](#)

Blob Tracking

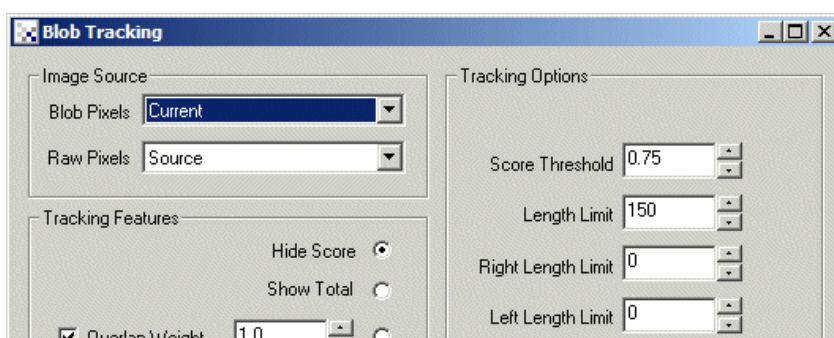


The Blob Tracking module provides a way to track blobs (collections of pixels) from one image to the next. Often as part of any object tracking solution it is necessary to identify not only that object moved but which object was which when comparing the current frame and the last frame.

The Blob Tracking module will label each blob with a specific id that will be attached to the same or similar blob in the next frame. What defines a blob as being similar in two images depends on how you have configured the Blob Tracking module.

Note that two images are needed for the blob tracking to work. The first is an image that defines the blobs. This is basically an image that is the result of a threshold, flood fill, segment colors, etc. and normally has many fewer unique colored pixels than the original image. The second image is the actual original image that was used to create the blob image. This image is needed in order to create better statistics than what the blob image would define. For example, a blob image would contain a single color for an entire blob, whereas the original image may have different colors that would better identify the blob in successive images. Thus one image is used to define what is a blob and the second is used to understand the features of that defined blob.

Interface



Instructions

1. Blob Pixels - Select the image that represents connected blobs. This is normally an image that has very few colors (segmented) such that same color pixels can be used to define a blob.
2. Raw Pixels - Select the image that contains the original raw pixel values that was used to define the blobs. This provides the raw pixel information needed to calculate the tracking features.
3. Tracking Features - Select the required features that are used to determine a blobs similarity from one image to the next. Try to use as few features as possible for your application. What features are required will depend on your application and what is being tracked. Selecting the appropriate checkboxes will add that feature into the similarity checking. Note that if the feature is of less importance but still valid you can decrease the weight to something <1.0 . Note that the weights can be above 1.0 if you want to instead strengthen a particular feature with regards to all the others. Again, what weights are needed for what features will depend on your project requirements.

Overlap - Similarity is defined as the overlapping of two blobs. I.e. some blob pixels in the current image are in the same location as in the previous image.

Location - Similarity is defined as the proximity or closeness of two blobs. I.e. a blob in the current image close to a blob in the previous image means they are similar.

Size - Similarity is defined as the area in pixels of two blobs. I.e. a blob in the current image with the same size as a blob in the previous image means they are similar.

Color - Similarity is defined as the pixel color of two blobs. I.e. a blob in the current image with the same color as a blob in the previous image means they are similar.

Aspect - Similarity is defined as the aspect ratio (width/height) of two blobs. I.e. a blob in the current image with the same aspect ratio of a blob in the previous image means they are similar.

Orientation - Similarity is defined as the orientation/angle of two blobs. I.e. a blob in the current image with the same orientation (second order moment) of a blob in the previous image means they are similar.

Shape - Similarity is defined as the shape comparison of two blobs. I.e. a blob in the current image with the same shape as a blob in the previous image means they are similar.

Histogram - Similarity is defined as the color histogram of difference of two blobs. I.e. a blob in the current image with the same color distribution as a blob in the previous image means they are similar.

4. Score Threshold - Specify the match similarity threshold which defines if a successful match is made between a current blob and a blob in the last image. If the similarity score between the blobs is above the threshold then the match is made, otherwise the blob is considered to be new and not connected to a previous blob.
5. Length Limit - Specify the length limit that connected blobs can move in any direction from one frame to the next. If you have slow moving objects you can lower the length limit to ensure that no spurious connections over that limit are made. You can enter zero to ignore this limit.
6. Left, Right, Down, Up - Specify the length limit that connected blobs can move from one frame to the next in the appropriate axis. Right is positive X, Up is positive Y. If you have slow horizontal moving objects you can lower the X length limit to ensure that no spurious connections over that limit are made. Both directions (positive and negative) are supported to allow right (positive X direction) but limited left (set negative to -1) movement. You can enter zero to ignore any limits and allow for as much change in distance as needed.
7. Precision - When tracking a blob an array of a blobs position is noted. The precision defines how many samples are added to the trail of blob positions. Increasing the precision will shorten the blob's trail list and remove smaller movements caused be noise. Decreasing the precision will increase the length of the blob's trail. See the variables list below ...

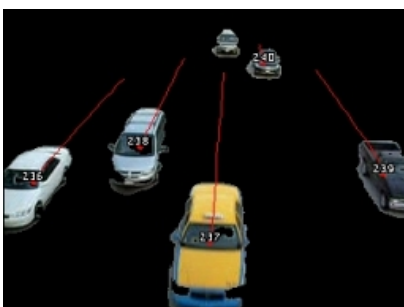
8. Present After Frames - To avoid having blobs that appear momentary to being treated as an active blob the Present After Frames number will require a blob to be present for X number of frames in order to be declared present and start being tracked.
9. Present After Pixel - To avoid stable blobs that do not move from being tracked the Present After Pixels number prevents blobs that remain within X number of pixels from their first location from being tracked. Thus if you use 10 then any blob needs to move more than 10 pixels (relative to their center) in order for the blob to start being tracked.
10. Absent After Frames - When tracking a blob it might suddenly disappear due to a momentary glitch or image disturbance. It is very desirable not to have the blob be reassigned a new id just due to a quick disappearance. The Absent frame number provides a way for a blob to be remembered for a couple of frames in order to allow the blob to be reacquired and retain its current id. By setting the number higher than zero you allow the module to remember blobs in previous frames in hope to preserve the id. Setting the number too high may cause blobs to stay in memory around for too long a time and then be reassigned to new blobs appearing close to the departure point of a previous blob.
11. Reset Tracking - Specify a variable whose value will be monitored to identify restarting of a video stream. For example, in the Media_Reader module the VIDEO_FRAME variable indicates which frame in the video is currently being viewed. When this counter resets below a previous value the Blob_Tracking module knows to also reset tracking information since the video looped.
12. Remove Un-Tracked - You can chose to remove those blobs that are present in the image but are not being tracked (perhaps they are just stationary objects).
12. Single Color/Labelled Color - The color of the overlaid id and trail (connecting blob arrow). If you select Labelled Color each track and blob will use a unique color (similar to [Blob Label](#)).
13. Display Trail - If you want to see how a blob has moved in previous frames select the Display Trail checkbox which will enable that graphic to show.
14. Samples - Select how many samples you want the trail length to be. The longer the trail the further in the past the blob's position can be seen.
15. Display Blob Id - Each tracked blob is provided a unique id that identifies that blob from image to image. This id can be displayed ontop of each blob for viewing purposes.
16. Font Size - Select the font size of the displayed blob id.
17. Display As Annotation - Select if you want the graphics to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

Notes

1. There is a maximum limit of 100 tracked positions. I.e. a trail cannot be more than 100 samples.
2. There is a maximum blob tracking count of 8192. If you have more than 100 blobs in the current image you will need to reduce that number down by either filtering the image or increasing the segmentation amount.

Example

Blob_Tracking after movement detection on Freeway



Variables

Each new image will create a bunch of variables that reflect the current tracking status. There are two types, blobs that are currently being tracked and those that have stopped tracking (most likely the blob is no longer present in the image). All the following are arrays that contain the current tracking state. Use the [Watch Variable](#) module to see these being created and modified per image.

BLOB_TRACKING - an array of $id1, x1, y1, id2, x2, y2, \dots$ of all objects that are currently being tracked

BLOB_TRACKING_IDS - an array of ids that are currently being tracked

BLOB_TRACKING_Z - for each id being tracked a variable holding an array of past COGs is created

BLOB_TRACKING_Z_X - the current X coordinate for the Z blob being tracked

BLOB_TRACKING_Z_Y - the current Y coordinate for the Z blob being tracked

BLOB_TRACKING_LOST_IDS - contains an array of ids that stopped tracking.

I.e. a blob being tracked has suddenly disappeared.

BLOB_TRACKING_LOST_X - for each lost id a variable hold an array of past COGs is created.

See Also

[Object Tracking](#)

[Flood Fill](#)

[Auto Threshold](#)

[Segment Colors](#)

[Reduce Colors](#)

[Blob Filter](#)

Convex Hull

The convex hull module will transform a binary blob into its convex hull. The convex hull of a binary blob is the shape formed if a rubber band were wrapped around the "outside" points of the blob. More specifically any angle formed by three perimeter points must be less than 180 degrees for a convex hull to be formed.

The convex hull is useful when you want to simplify the shape of a binary blob by simplifying the perimeter.

Example

Source

Convex Hull



See Also

Line Counter



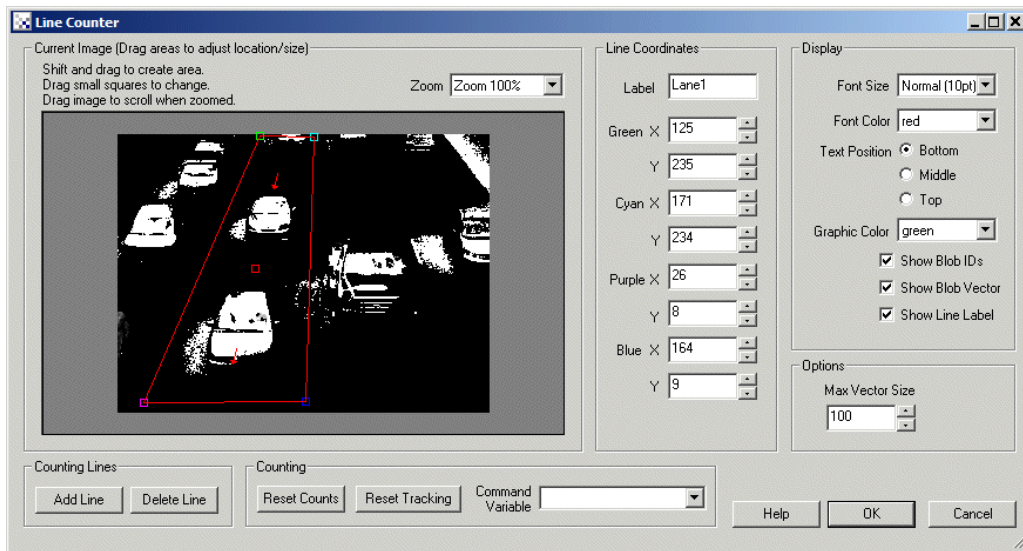
The Line Counter module will count objects/blobs within a specified area while keeping tracking of previous blobs to ensure objects are only counted once. This is a common issue within production lines that have multiple products within view of a camera that can proceed at different conveyor speeds or even be stopped momentarily. This module uses object spacing (the distance between object) to keep track of a single object from one image to the next in order to know when it disappears from the camera view (indicating a count).

The module will also accommodate some merging and splitting of objects as long as the most common object size is the size of a single object to be counted.

Functionally similar to the [Blob Tracking](#) module, the Line Counter module will track objects only based on relative position to previous blobs. Thus the module assumes that objects are moving at about the same speed and in the same direction. Because of these constraints the Line Counter module does not require as many characteristics as the Blob Tracking module does in order to track objects. This allows for simpler configuration and more reliable operation in production line operation.

The module will count objects as they disappear from view (i.e. leave the specified line coordinates).

Interface



Instructions

1. Current Image - Move the red rectangles to the place you want to track objects. You can move the entire line by dragging the smaller red square in the center or by adjusting the individual corner squares. To fine tune the positioning you can change the coordinates of the overall square using the textboxes below in the Line Coordinates area.

Use SHIFT-drag to create a new Line Area of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

Note the direction of the arrows indicates the flow of objects. This MUST be the only direction objects flow otherwise the Line Counter module will not work correctly.

2. Line Coordinates - To accurately place the line area you can modify its coordinates using the provided numerical coordinates.

3. Label - Use the label text box to specify the name of each line. The label will be used to set the count for each line when using multiple count lines per image.

4. Add / Delete Line Buttons - You can add or remove counting lines by using the appropriate button. Adding a new counting line will create a new area that can be moved to the appropriate location. Having an existing line selected such that its coordinates are visible in the Line Coordinates interface and pressing the Delete button will remove that line from the interface.

5. Reset Counts - Will reset the number of objects counted to zero for ALL counting lines.

6. Reset Tracking - Its possible that during the course of counting objects some may need to be removed (due to quality assurance checks). When this happens you can reset the tracking which will prevent all previous objects or if desired lock previous images. This will accommodate for object

this happens, you can reset the tracking which will recount all in view objects as if they suddenly appeared in view. This will compensate for object being removed or added as part of a natural production line without causing the count to be incorrect.

7. Command Variable - In order to automate the resetting you can specify a variable that includes **reset** (resets counter and restarts tracking), **reset_count** (resets just the count) or **reset_tracking** (recounts current objects).

8. Font Size, Font Color, Text Position - Specifies size, color and position of the count text. This is separate from the graphic colors which can be removed by setting the colors to blank.

9. Graphic Color - The color used to show the line counting area, object ids and object vectors.

10. Show Blob IDs - Indicates the object number graphically as the objects move by.

11. Show Blob Vector - Shows the vector that connects a blob from the previous image to the current one.

12. Show Line Label - Shows the label for each line counting area.

13. Max Vector Size - The maximum movement in pixels that an object can move from one image to the next.

Variables

Line_X - the count for the particular line assuming no label was used. Otherwise a variable with the label name will be created to contain the current line count.

See Also

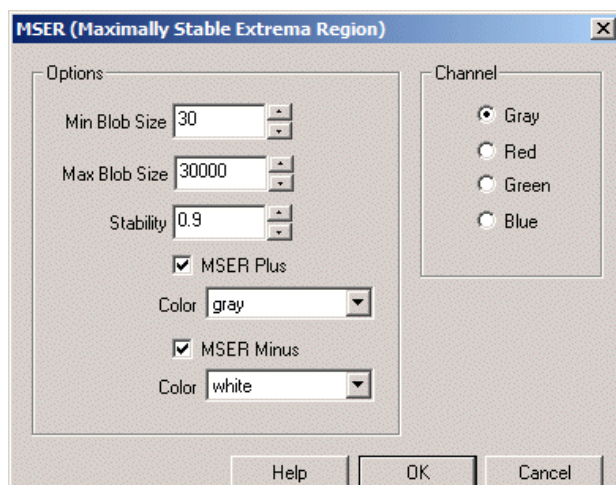
[Blob Tracking](#)
[Object Tracking](#)

MSER

The MSER (Maximally Stable Extremal Regions) module is used to segment an image into those parts that are maximally stable in the specified channel. I.e. the module finds those blobs that don't change much in size while growing the blob to include its surrounding neighbors. The technique starts with each pixel and slowly grows blobs by increasing (MSER Plus) or decreasing (MSER Minus) the intensity range that makes up a particular blob. While this growing of blobs happens the module monitors the growth of each blob and isolates those that don't grow much over a number of intensity increases. This attribute will tend to isolate blobs that are well bounded from their surroundings.

This form of segmentation results in similar results to the Adaptive Threshold module but can perform better on text, labels, etc. as a precursor to OCR (Optical Character Recognition).

Interface



Instructions

1. Min Blob Size - The Minimum size a blob needs to be before being considered an MSER.
2. Max Blob Size - The Maximum size a blob can be to be considered an MSER.
3. Stability - Defines how stable in intensity (i.e. how well defined) a blob needs to be in order to be identified.
4. MSER Plus - Specifies increasing intensity levels are using during blob growth. Isolates darker blobs.
5. MSER Minus - Specifies decreasing intensity levels are using during blob growth. Isolates lighter blobs.
6. Channel - Sometimes isolating this blob growth to a specific color channel can help segmentation. By clicking on the selected channel only that channel will be used during blob growth.

Example

Source



MSER



See Also

[Adaptive Threshold](#)

Smooth Hull

The smooth hull module will smooth a blobs shape. The blobs perimeter is averaged within the specified window. The final outline is then weighted against the new averaged outline and the original. A 100% weight replaces the original outline with the averaged one.

The result of the module is to smooth the blob perimeter without loss of edges.

Interface



Instructions

1. Specify the Window Size that the outline of the blob will be averaged across
2. Specify the weight that the final outline is biased towards. 0 % is unchanged, 100% means the outline is replaced by the averaged

Example

Source



Smoothed



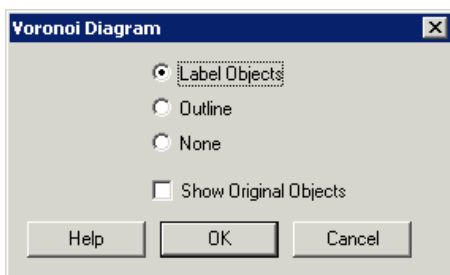
See Also

[Convex Hull](#)

Voronoi Diagram

The Voronoi Diagram creates a voronoi diagram based on the current binary image. The voronoi diagram represents the borders of an object that bound the space of the next neighboring object. The diagram indicates the distance between objects.

Interface



Instructions

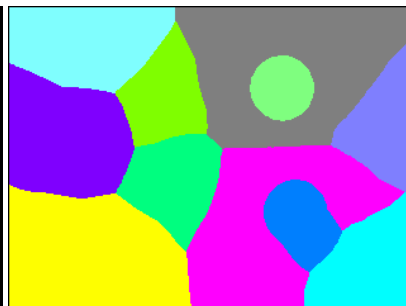
1. Select how the voronoi cells are depicted. Label Objects will colorize each cell with differing colors. Outline shows the border only.
2. If you'd like to see the original blobs select the "Show Original Objects" checkbox.

Example

Source



Voronoi Diagram

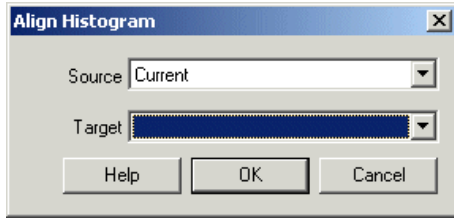


Align Histogram

The Align Histogram module is a useful precursor to any image comparison routines such as template matching, cross correlation, stereo depth or simple image subtraction. The module takes as input two images and will align the colors in the "source" image with those specified in the "target" image. Thus if you wish to compare images with respect to color, this module will provide a means to match and/or align different histograms and color maps.

image. Thus if you wish to compare images with respect to colors this module will provide a way to greatly reduce different lighting and color shift effects that can happen between two successive image captures or even in simultaneous capture by different cameras.

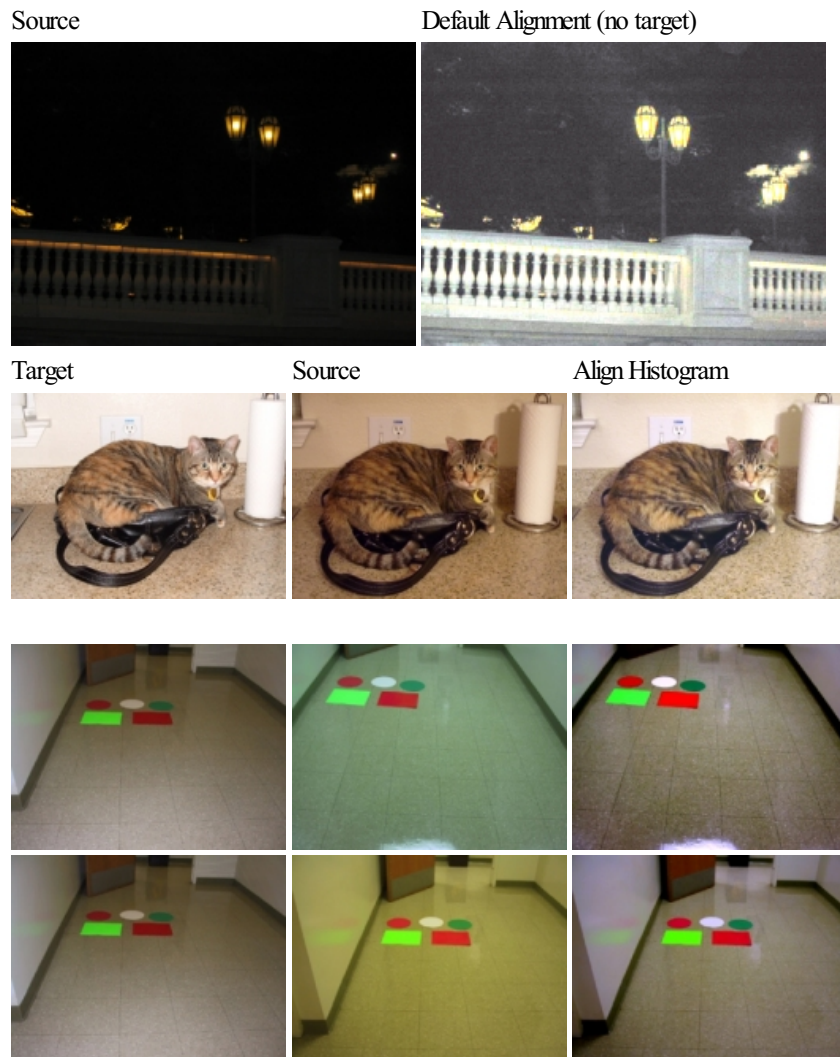
Interface



Instructions

1. Source - Select the source image or the image that should be transformed into the target's colors/intensity.
2. Target - Select the target image or the image that has the "right" colors/intensity. Note that if no target image is specified the source image is transformed into a standard histogram where the red, green and blue channels are normalized.
3. Overlay - If your source image has had a color remapping relative to the target in a non-linear way you may want to chose the "Align using overlap occurance count" which will compare pixels in the source and target at the exact same location and use the combination that is most frequent in the two images. Thus if a red pixel in the source image is always in the same location as a blue pixel then the source image red pixel will be mapped/converted to a blue pixel to best match the target image. This is useful in situations where you would like to compare the two images and ensure that the source is as close as possible to the target image without actually changing the pixels location.

Example



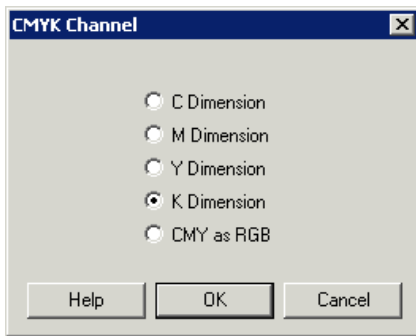
[Click Here](#) to load a robofile that was used to produce the above alignment using the source and target image. You can specify the image to align to in the Load Module within that robofile and then just drag/drop images that you want to transform into the main RoboRealm GUI interface.

See Also

CMYK Channel

The CMYK module provides color conversion into the CMYK color space. The CMYK color space is an abbreviation for cyan, magenta, yellow, and key. This color space is a subtractive color model frequently used in color printing.

Interface



Instructions

1. Select the appropriate color channel to convert the current image into. Note that for CMY as RGB the K dimension is lost as only 3 channels are available. It is therefore not possible to convert back from RGB into CMYK within RoboRealm and that option is removed.

Example

Source



CMYK Channel K



See Also

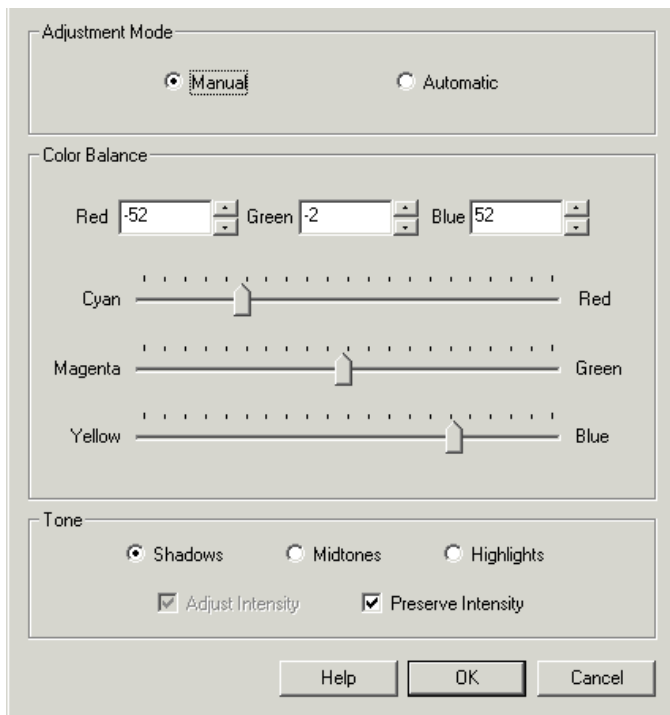
Color Balance

Processing images for color is one of the easiest ways to identify an object within an image. However, color is never constant and can change drastically based on the illumination (sunshine, incandescent bulb, etc.) and even on the type of camera used. In an effort to reduce these effects the Color Balance module was added to provide you both a manual and automatic way of adjusting the overall colors within an image. If you're trying to detect color be sure to try the automatic mode of the Color Balancer which may help your values to be more consistent from frame to frame.

The Color Balance module has similar results to the Equalize module but can provide a softer alignment of colors that is more appealing than some of the results of the Equalize module.

Interface





Instructions

1. Adjustment Mode - select if you want to manually adjust the color balance or let RoboRealm make a guess as to what a good color image should look like. If you select Automatic then all the manual controls will be disabled.
 2. Color Balance - if manual mode is selected you can drag each of the horizontal scroll bars to change the values in each of the RGB channels. You can also use the spin up and down buttons to fine tune the channel value or simply type it into the editable text box. For each channel you can increase or decrease the value. Increasing the value will add more of that color into the image while decreasing it will remove that color and allow the other colors to dominate.
- By adjusting the levels it is possible to remove dominant colors within the image to create a more heterogeneous color image.
3. Tone Balance - if you find that using the above adjustments do not sufficiently adjust the current image select a different intensity level to adjust instead. The three levels are Shadow, Midtone, or Highlight and focus the color changes in the lower, middle and upper intensity regions. Thus if you have a very bright image whose dominant color is red you would select the Highlight option and adjust the Red level negatively.
 4. Preserve Intensity - as you adjust the color channels the image may change in overall intensity. Select the "Preserve Intensity" checkbox to ensure that the image keeps its overall intensity level when adjusting the colors.
 5. Adjust Intensity - when in Automatic mode you can also select to adjust the image intensity in addition to the color. This will ensure very dark or very light images are adjusted to better reveal the overall image details.
 6. Max Contrast - when in Automatic mode each color channel's contrast is altered in order to cover a wide range of intensities. Sometimes this change can cause undesired effects when very little of a channel is present. If you notice that dark areas are appearing blue reduce the Max Contrast value in order to reduce the contrast expansion that each color channel goes through. This should cause the image to still be enhanced but not to large extremes.

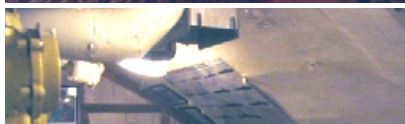
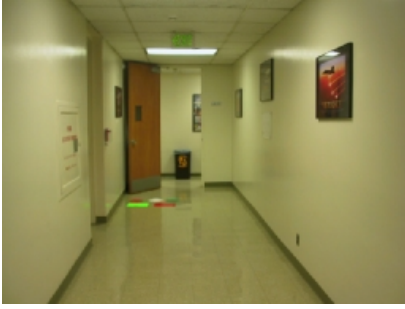
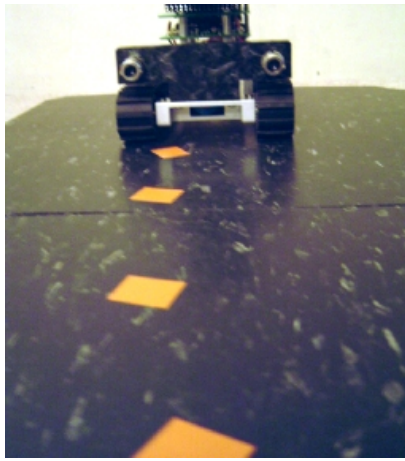
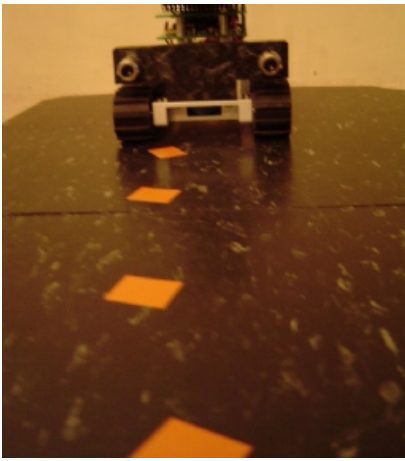
Example

Source



Color Balanced (automatic)







See Also

[Align Histogram](#)

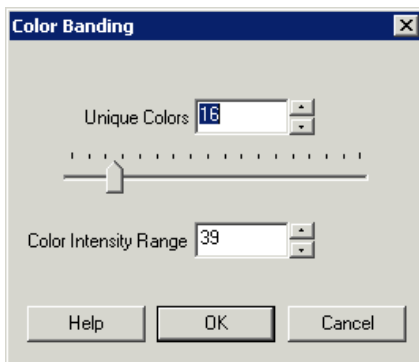
[Equalize](#)

[Normalize](#)

Color Banding

The Color Banding module is similar to the [Pseudo Color](#) module in that it creates an artificial color palette on which to apply the current image. In the case of the Color Banding module the colors are unique colors (colors that differ enough from each other to be visually different) and are looped around the full palette until the entire range is used. This palette configuration will intensify small intensity changes in a surface and reveal subtle textures that may not be visible within the current full color image. See the water surface below.

Interface



Instructions

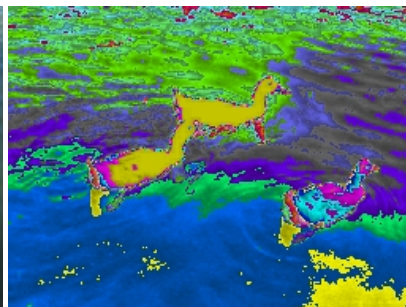
1. Select the number of unique colors used in the color banding process
2. Specify the intensity range used by each unique color. For example, if the intensity range is 5 and a the next unique color is red then the next 5 used colors will be from (152,0,0) (177,0,0) (202,0,0) (227,0,0) (252,0,0). This proceeds until the entire color spectrum has been filled. Once the unique number of colors are used the process restarts from the first unique color.

Example

Source



Color Banded



See Also

[Pseudo_Color](#)

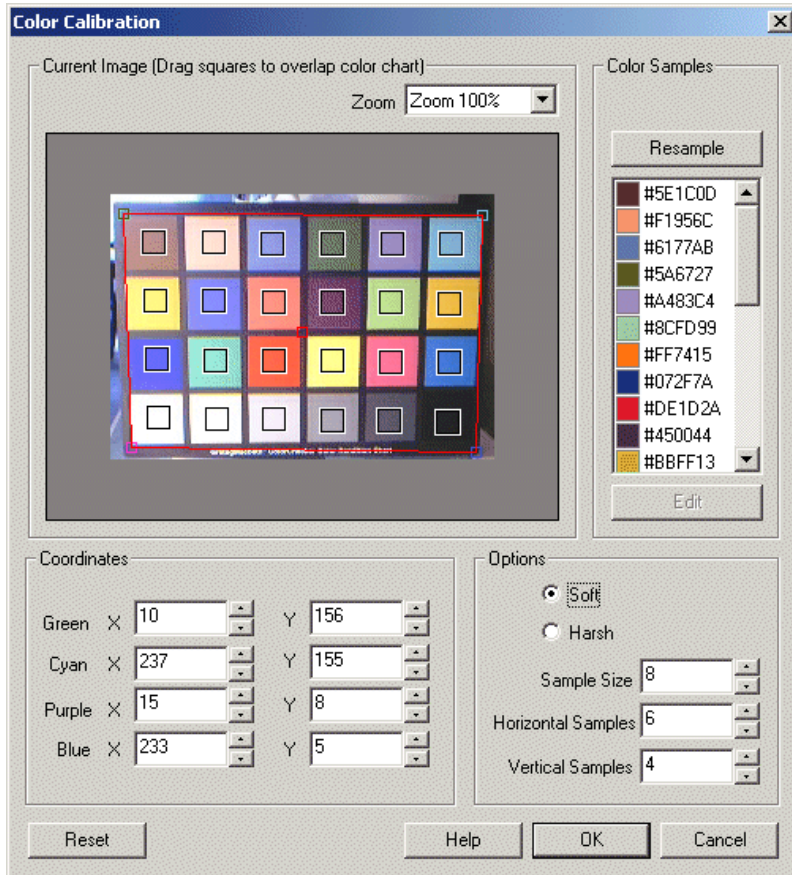
Color Calibration



The Color Calibration module provides a way to better align the colors within an image to that of reality. This is accomplished by including a standardized color chart within the image that can be used as a color reference. By aligning the image color chart with the known color chart the rest of the image's pixels can be moved towards the calibrated colors.

Note that images that are severely out of color alignment will cause pixelated artifacts as the module tries to move pixel colors to aligned colors. It is always better to configure your hardware imaging devices to produce the best colors possible and thereafter attempt software correction.

Interface



Instructions

1. Green, Cyan, Purple, Blue squares - drag the squares to align with the color chart that should be present in the image.
 2. Coordinates - Use the text boxes to refine the coordinates
 3. Color Samples - The Color Samples are the known colors that are used to calibrate against. The module is configured for the Macbeth color chart by default. If you are using a different calibration chart you can press the Resample button to sample the colors from the new chart (once aligned) and then use either the edit button or double click on the color to change its value.
 4. Soft vs Harsh Option - Select how strong the alignment should be. Selecting Soft will align the source image to the calibrated color with respect to intensity and contrast which will help align the colors but not radically change the image colors.
- Note that on harsh calibration flat areas will appear more pixelated
5. Sample Size - Specify the size of the sample area. The sample area should include only those pixels that specify a specific color and are average to produce the alignment color. If the sample size is too large then non-color chart colors may influence the calibration so be sure the squares fit in the chart area correctly.
 6. Horizontal & Vertical Samples - Specify the appropriate number of samples that your color chart contains by adjusting the horizontal and vertical sample number. Note that when adjusting these values the number of squares and sample colors will change to reflect the new dimensions.

Example

Source Soft Calibration





Harsh Calibration

Target



See Also

- [Color Balance](#)
- [Align Histogram](#)

Color Depth

To reduce the number of colors in the image select the appropriate bit count in the color depth interface. Doing so will reduce the number of bits used for the RGB triad for color representation and will effectively reduce the image to fewer colors.

Note the color selection is not optimized and simply performed as a bitwise right shift of the RGB colors.

On selections of uneven number of bits the green is provided more bits.

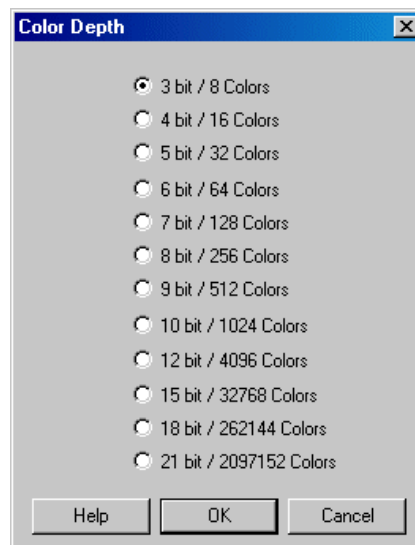
If 21 bits is selected

$R = R \gg 1$

$G = G \gg 1$

$B = B \gg 1$

Interface



Instructions

1. Chose the appropriate number of colors to reduce the current image to.

Example

Source Image



Reduced to 8 colors



See Also

[Flood Fill](#)

[Color Depth](#)

[RGB Channel](#)

[RGB Filter](#)

[Threshold](#)

[Auto Threshold](#)

Color Filter

The Color Filter module provides a way to isolate specific colors from the current image. This module functions similar to the [RGB Filter](#) but allows for custom colors and grays to be detected.

The module will convert the current image into the HLS (see [HLS Channel](#)) format and use the Hue and Lighting fields to segment the image into the specified colors.

Keep in mind that the amount of light plays an important part in detecting colors. Objects not under enough lighting will appear different than objects with appropriate lighting and may cause detection failure. Object with too much light will appear much more pale and actually not contain much 'color' information. The best detection can be performed under even illumination that lights all the parts evenly.

Care should also be given to try to eliminate any reflective surfaces. Reflective surfaces will cause that surface to appear different in color than it actually is. (See the extra blue detected around the spirals below which is actually the white table reflecting the color blue.) This is also apparent on glossy objects that will reflect the illumination directly. You can see this issue on the brown marker where the top middle of the marker is reflecting white light. This causes some failure in detection of that object as the colors will not match.

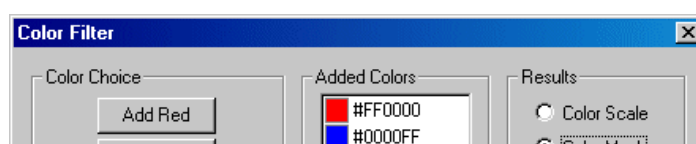
Even images of gray, white, black surfaces will typically have some small amount of color information but will be mainly due to noise. Depending on the settings of the parameters below you may see these non-colors be detected incorrectly as colors. If this happens check the minimum color setting as that will help to eliminate detection of non-colored surfaces as colored. You can also reduce this error by adding in grays that are close to the surface causing the error.

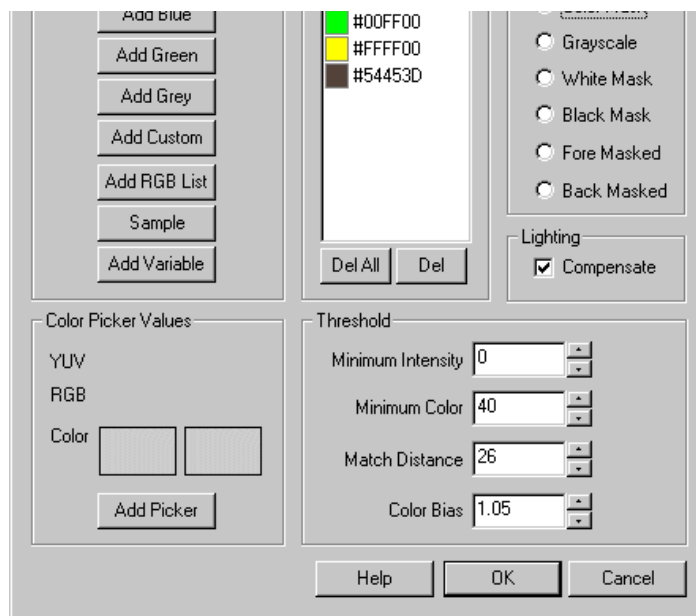
While the module will attempt to best connect a pixel with a selected color it can still make what appear to be mistakes to us. This illustrates the way that we perceive color is different from the way the module compares colors. If this happens try to add colors into the "Added Color" list that are as close as possible to the desired color.

Note that often objects may contain dithered (speckled) colors that differ from one pixel to the next but appears different when viewed from a distance. If this occurs add the [Mean](#) module just before this module in order to blur the colors into each other which will aid in detection.

The image will be black if no colors are specified. Once colors are entered in the image will change to show the detected color as specified in the "Added Colors" list. With very few colors most of the detection may be incorrect depending on the color settings. Thus, if you are detecting the color red, have added that to the list but the entire image now appears red try reducing the Match Distance as that will help to tighten up the color detection. Or, conversely, if you find that your red object is not being detected try to increase that value.

Interface





Instructions

1. Color Choice - Chose the desired way to add RGB values to the focus Added Colors.
 - Add Red - adds the RGB red value to the list.
 - Add Blue - adds the RGB blue value to the list.
 - Add Green - adds the RGB green value to the list.
 - Add Grey - adds the RGB grey value to the list.
 - Add Custom - you can select the RGB value from the color picker.
 - Add RGB List - provides an interface to add a list of RGB values specified as hexadecimal or decimal triplet.
 - Add Sample - adds the RGB values from the current image. You will be provided an interface to define an image in the current image. On pressing OK the most common color will be added. Thus you don't need to select your color area exactly as small amounts of the wrong color will be ignored.
 - Add Variable - adds the RGB value specified in the selected RoboRealm variable.
2. Color Picker - Shows the color values directly under the mouse as seen on your computer screen as an YUV, RGB value and as a color square. Clicking with the mouse will update the second color box with that clicked on color. Pressing the "Add Picker" button adds that clicked on color to the "Added Color" list.
3. Added List - Shows the colors to be detected within the image. If you had make a mistake and added a color incorrectly you can delete a color by using 'Del All' or selecting the color and pressing 'Del'.
3. Minimum Intensity - Dark colors, i.e. black, will still have some color information contained within them (although not visibly) which may cause some incorrect matches to occur. If you are not detecting black it is a good idea to remove those pixels from match consideration by increasing the Minimum Intensity to a higher number. 0 would just eliminate pure black whereas 255 would eliminate white. Most appropriate values will be around 50. Note that this value will not depend on the colors in the Added List.
4. Minimum Color - White, gray and black are not real colors in that they contain no color information but are instead based entirely on intensity. If you are not detecting those colors increase the minimum color threshold until non-colors are removed from the image. Note that this value will not depend on the colors in the Added List.
5. Match Distance - Specifies how close a match should be before it is considered a match. If you specify 0 then the colors in image will have to be exactly the same as those in the Added List in order for a match to occur. Normally this is highly unlikely given the noise level present in most real images. In order to relax this precision match incase the value to all less precise matches to occur. This value will be the most influential in attempting to remove unwanted objects.
6. Color Bias - If your Added List does not include grays you can bias the detection more towards colors in order to better match darker or lighter colors. Biases above 1.0 will match more based on color, whereas below 1.0 will favor matches based more on intensity.
7. Lighting Compensation - If you find that your colors are not matching well try selecting this checkbox. This will magnify colors and decrease intensity variations in the image in an attempt to get a better color match.
8. Results - Select how the results should be represented
 - RGB Scale - resulting values are scaled RGB values from 0 to 255 that depend on how close the original color is to one in the Added Colors list
 - RGB Mask - resulting value is the color that is most similar to one in the Added Colors list
 - Grayscale - resulting value is how close the color is to a color in the Added Colors list but represented in grayscale values
 - White Mask - resulting match value is white, non-match is black
 - Black Mask - resulting match value is black, non-match is white

Fore Masked - resulting image shows original color where a match is made

Back Masked - resulting image shows original colors where a match was NOT made. This is useful in determining how well your matched colors are working on the actual image.

The color picker works by showing the color of the pixel the mouse is currently hovering over. When you click the second square box is updated with that color. Selecting Add Picker will then add that value to the Added Colors list. This functionality allows you to select any color on your desktop (except colors within the RGB Filter interface) to be added to the Added Colors list.

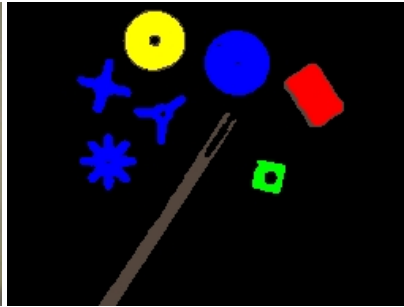
One a single instance of a color will be added to the Added List. If you attempt to add exactly the same color to the list it will not be added. The list order is irrelevant and does not affect color detection.

Example

Source Image



Color Filter on Red, Green, Blue, Yellow and Brown ([download](#))



Source Image



Color Filter on Purple



See Also

[RGB Filter](#)

[RGB Channel](#)

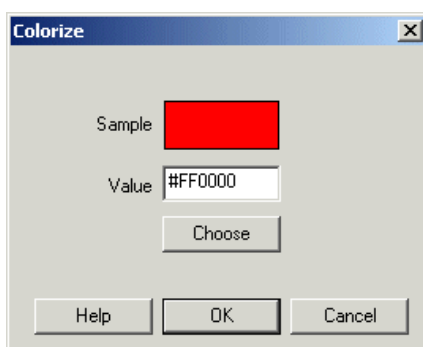
[Threshold](#)

[Auto Threshold](#)

Colorize

The Colorize module will recolorize and non-black pixels with the selected color. This module is useful for inspection or illustration of detected object that can then be colored into an obvious color to be overlayed with the original image.

Interface

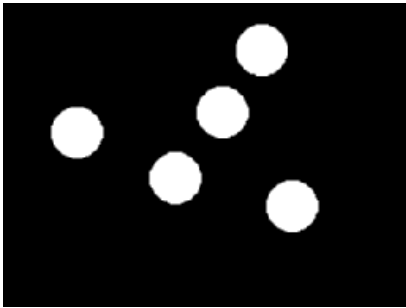


Instructions

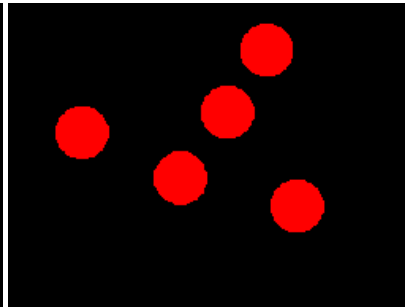
1. Select the appropriate color you would like to replace all non-black pixels with. Either type in the color using #FFFFFF or 255,255,255 format.

Example

Source Blobs



Colorized Red



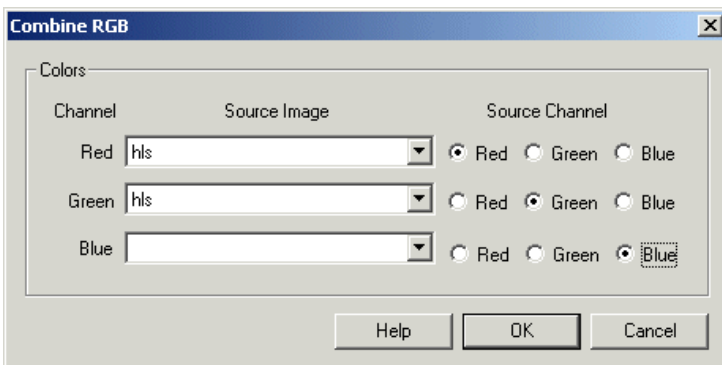
See Also

[Pseudo Color](#)
[Color Depth](#)

Combine RGB

The Combine RGB provides you with a way to create an RGB color image based on different image channels. This means you can separate an image into individual channels, filter the channels separately and then recombine those processed channels into a new image.

Interface



Instructions

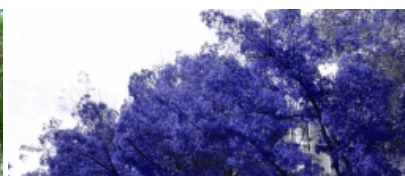
1. Select the source images to use for each channel
2. Select which channel in the specified source image the data will be grabbed from

Example

Source



Combined RGB without Saturation





[Click here](#) to download the robo-file that sets the saturation to zero in an image. This process essentially turns green and red objects into a nice blue!

See Also

[Math](#)

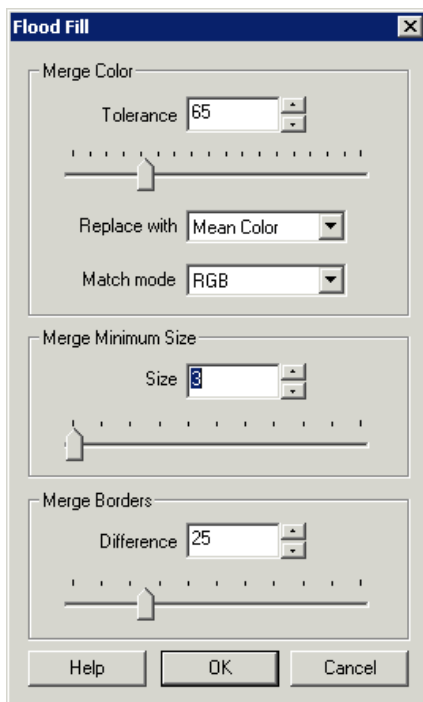
Flood Fill

The Flood Fill module functions similar to a paint program's "flood fill" in that it fills areas of common color (within a degree of error) with a single color. This is very useful in segmenting images into discrete color blobs that can then be processed for shape or other features. Note that as the error threshold increases you will notice bleeding of one color area into another until the entire image becomes one color.

Once this process is complete there are often many pixel irregularities remaining due to interpolation that occurs between objects. These pixels can be removed by merging them with their closest larger neighbor. The size of these pixel groups that can be merged can be specified in the "Merge Minimum Size".

To further consolidate blobs into meaningful objects you can select neighboring blobs without a significant border (i.e. color change) between them to be merged. This is a process that happens after the initial objects have been segmented. This helps to merge similar blobs without increasing the color tolerance which causes bleeding into other unwanted objects.

Interface



Instructions

1. Select what the color area should be replaced with
2. Select how the color areas should be compared. Different color bands and spaces will create different combination patterns. Selecting different matching modes will allow you to better adapt to your specific image segmentation task.

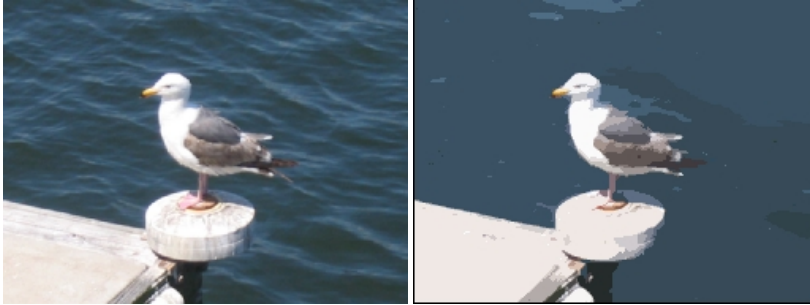
matching modes will allow you to better adapt to your specific image segmentation task.

3. Select the error tolerance level by typing in a number or by moving the horizontal scroll bar. You should notice the image change based on that setting.
4. Specify the object size of pixel groups that should be merged into their closest neighbor regardless of color. This helps to reduce noise within the image.
5. Specify the border difference below which neighboring objects will be merged.

Example

Source

Flood Fill Tolerance = 56



Border Difference = 22



See Also

[Threshold](#)

[Auto Threshold](#)

[Reduce Colors](#)

[Color Depth](#)

Grayscale

The Grayscale module converts a color RGB image to grayscale values using the following formula techniques. Note that R,G,B represent the current pixels colors values in RGB color space.

Method 1

$$\text{pixel} = (R+G+B)/3$$

Method 2

$$\text{pixel} = 0.299R + 0.587G + 0.114B$$

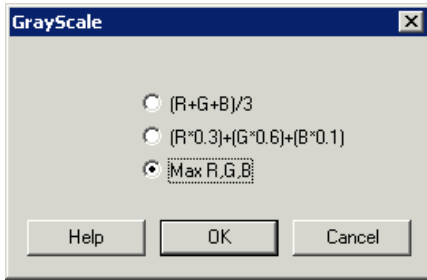
Method 3

$$\text{pixel} = \text{maximum}(R,G,B)$$

the 'pixel' value is then assigned to the Red, Green and Blue channels to create the final image.

Note that each technique will change the way colors are converted to grayscale which will effect the relative intensities of each color. For example, method #1 makes Red and Blue seem like the same grayscale color whereas method #2 does not.

Interface



Instructions

1. Specify which technique you would like to use in converting your image to grayscale by selecting the appropriate radio button

Example

Source Image



Method #1



Method #2



Method #3



See Also

[Equalize](#)

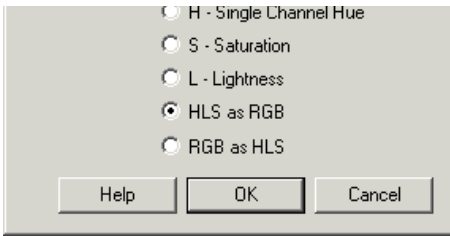
[Normalize Color](#)

HLS Channel

The HLS channel module converts the image into a hue, saturation, and lightness components instead of the RGB representation. The Hue is the color of the image, the Saturation is the pureness of the hue color, and the lightness is the strength of the hue. By selecting the appropriate radio button you can see the appropriate channel without the other two components similar to the RGB and YUV channel module.

Interface





Instructions

1. Select the appropriate channel

Multi Channel Hue - image hue approximated as maximum RGB channels

Single Channel Hue - image hue represented as grayscale

Saturation - image saturation represented as grayscale amount

Lightness - essentially the grayscale representation of the image

HLS as RGB - converts from RGB to HLS by placing the HLS components placed into each of the RGB channels

RGB as HLS - converts from HLS to RGB by assuming each of the RGB channels contains the HLS channels

Example

Source Image



Image Hue



Image Saturation



Image Lightness



See Also

[RGB Filter](#)

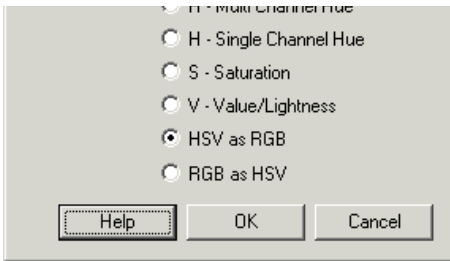
[YCbCr Channel](#)

HSV Channel

The HSV channel module converts the image into a hue, saturation, and value/lightness components instead of the RGB representation. The Hue is the color of the image, the Saturation is the pureness of the hue color, and the value is the strength of the hue. By selecting the appropriate radio button you can see the appropriate channel without the other two components similar to the RGB and YUV channel module. Note this module is very similar to the HLS module with the only difference being the luminance channel.

Interface





Instructions

1. Select the appropriate channel

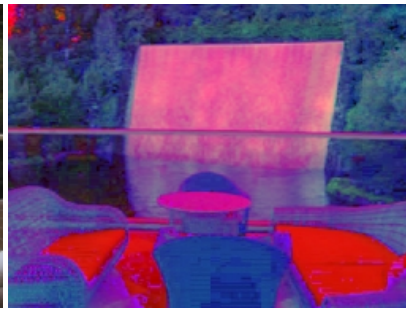
- Multi Channel Hue - image hue approximated as maximum RGB channels
- Single Channel Hue - image hue represented as grayscale
- Saturation - image saturation represented as grayscale amount
- Value - essentially the grayscale representation of the image
- HSV as RGB - converts from RGB to HSV by placing the HSV components placed into each of the RGB channels
- RGB as HSV - converts from HSV to RGB by assuming each of the RGB channels contains the HSV channels

Example

Source Image



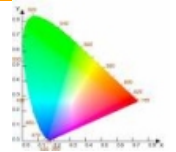
RGB as HSV



See Also

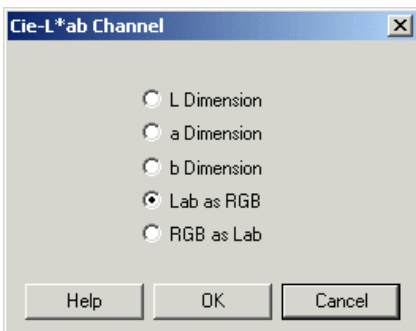
- [RGB Filter](#)
- [YCbCr Channel](#)

Lab Channel



The Lab Channel module provides a way to convert to and from the CIE-L*a*b color space. The Lab color space converts a pixels light into the L channel with a and b being color channels.

Interface



Instructions

1. Select the appropriate Channel to convert the source image to

1. Specify the transform to apply to the current image

L - The L dimension of the Lab color space

a - The a dimension of the Lab color space

b - The b dimension of the Lab color space

Lab as RGB - convert the current image into Lab color space and apply each of the Lab channels into the RGB channels

RGB as Lab - convert from Lab space into RGB space by assuming that the current RGB image includes Lab color space data.

Example

Source



Lab as RGB



See Also

[XYZ Channel](#)

[YCbCr Channel](#)

[HLS Channel](#)

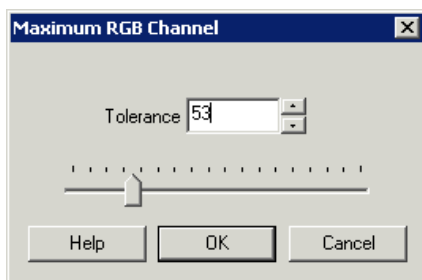
[RGB Channel](#)

Maximum RGB Channel

The Max RGB Channel module will replace every pixel with its strongest RGB channel component. For example, if a pixel's color value is 255,45,22 then this component will replace that pixel with 255,0,0 since red is the largest of the triplet RGB values.

This module is a good way to investigate the color basis of an image and to provide a form of color segmentation.

Interface



Instructions

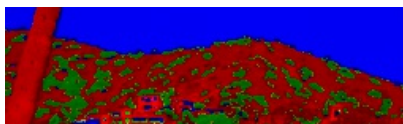
1. Select the appropriate tolerance level. The tolerance level is used to determine when two colors are close enough to be considered the same. For example, if the color pixel is 255, 249, 22 and the color tolerance level is 10 then the new pixel will become 255, 255, 0 since the green channel with within 10 values of the red channel.

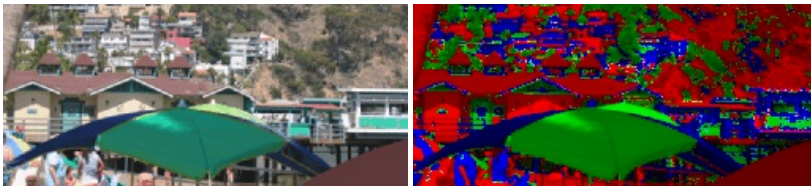
Example

Source



Max RGB Channel





See Also

[Normalize Color](#)

[RGB Channel](#)

[RGB Filter](#)

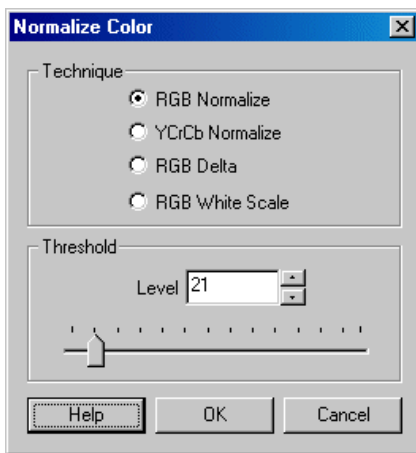
Normalize Color

The Normalize Color module removes all intensity values from the image while preserving color values.

This module has the effect of removing shadows or lighting changes on same color pixels.

The side effect of this technique is that pixels can become aberrantly colored and may not reflect the actual color value of the image. However, this function can be used in conjunction with thresholding to consistently segment a color image.

Interface



Instructions

1. Technique - Select which technique to use to create the normalized image.

RGB Max - The module processes each pixel by dividing the pixel by its intensity value.

$$R = R/(R+G+B)$$

$$G = G/(R+G+B)$$

$$B = B/(R+G+B)$$

YCbCr - Similar to the RGB Max technique but first converts the pixels from RGB color space to YCbCr color space and uses the Y channel as the intensity level.

RGB Spread - Spreads the RGB color channel to the full possible pixel range of 0-255 for each color channel.

RGB White Scale - Scales each RGB color such that at least one of the channels is the maximum at 255.

Middle Intensity - Sets intensity level of each pixel to 128 while preserving color. See [Flatten](#) module.

2. Threshold - In most images not all the pixels have color information with those pixels appearing as gray scale pixels (with intensity level 0 - 255). Using the threshold you can remove the less color saturated pixels from the scene.

Example

Source Image



Color Normalized Image



See Also

[Normalize](#)

[YCbCr](#)

[Equalize](#)

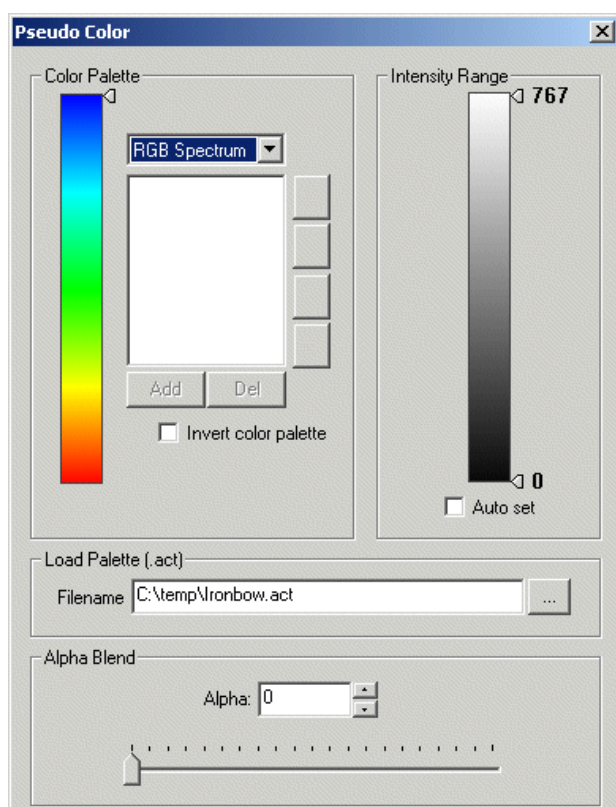
Pseudo Color

Artificially coloring an image can reveal textures and qualities within the image that may not have been apparent in the original coloring. You can use the pseudo coloring module to reveal an image's hidden texture.

The Pseudo Color module colonizes the image based on its grayscale value which maps to a full RGB color range. Pseudo Color images can help to reveal image qualities that would not be readily visible within the image's true color.

The false color of a pixel is created by determined by summing its RGB values and mapping them into a 768 row lookup table. This lookup table is created by oscillating through the RGB color table from Blue to Red to create 768 unique colors.

Interface



Help

OK

Cancel

Instructions

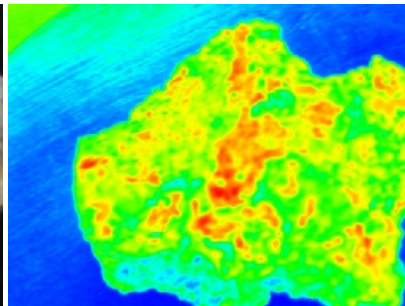
1. Select the color palette that you would like to use. The default is the full RGB Spectrum. If you would like to create your own palette chose the "custom" (second last selection) setting which will enable the palette buttons for you to add, delete and reorder your own palette. If you wish to load your own 768 byte RGB order .act palette file select the "load palette" option (last selection) which will load in the file specified below as the palette.
2. If you want to reverse the high/low values of the palette (i.e. red to black instead of black to red) you can select the 'Invert color palette' checkbox
3. If you would like to shift the color palette grab the little palette knob seen on the right side of the color palette and drag it to the desired location. Shifting the color palette can be helpful in correlating surfaces with different lighting. For example, an image on a sunny day may have values shown in green using the RGB Spectrum palette. Taking the same image on a cloudy day would result in those values shown as red pixels. Shifting the color palette as a form of 'color correction factor' can help you better manually align those comparisons.
4. To have the module automatically set the minimum and maximum pixel intensity values used in coloring click on the 'Auto set' checkbox. When this is enabled the module will scale the color palette from the minimum value to the maximum value to ensure that the full palette colors are used.
5. If the auto set checkbox is not enabled you will see the minimum and maximum values next to two knobs on the right side of the grayscale intensity palette. You can grab these knobs to change the respective values manually. Manually setting the minimum and maximum intensity values can be helpful in more closely analyzing specific intensity ranges within images.
6. The alpha blend allows you to view a combination of the original image and the pseudo colored image to gain a better understanding of the actual surface that is being colored. If you would like to blend the original color image with the specified palette select the appropriate alpha value. A zero alpha value replaces each image pixel with the palette value, a 100 value leaves the original image intact with the colored image being completely transparent, a 50 value combines 50% of the color palette with 50% of the original image.

Example

Source



Pseudo Colored



See Also

[Color Banding](#)

[Normalize](#)

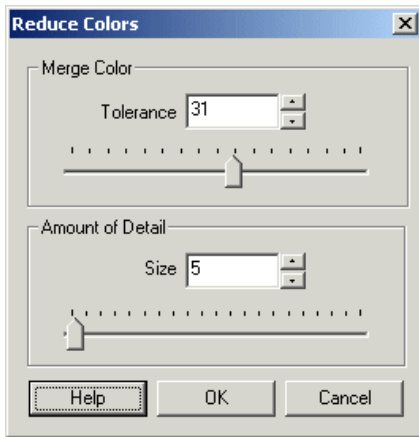
[Normalize Color](#)

Reduce Colors

The Reduce Colors module is functionally similar to the [Flood Fill](#) module which reduces the number of colors in an image in an effort to create meaningful blobs. Meaningful blobs are pixels grouped together that share a common color that hopefully make up a representative grouping of an object to be analyzed.

The Reduce Colors differs from the Flood Fill in that it uses a faster blob creation routine by using enough colors to represent the global color space used given the tolerance. Flood Fill will use the local pixel colors to determine blob boundaries whereas Reduce Colors takes into account all colors used in the image.

Interface



Instructions

1. Merge Color - Select the tolerance level which will reduce or increase the number of colors used to represent the image. Increasing the tolerance will force fewer colors, whereas decreasing the tolerance will increase the number of colors
2. Detail Size - Often when creating blobs a little blur to remove noise is desired such that created blobs are large enough to be representative of an area of the image. If you find that much of your image contains single pixels increase the Detail Size to remove those individual pixels.

Example

Source

Reduce_Colors



See Also

[Flood Fill](#)

[Segment Colors](#)

RGB Channel

Splits the RGB color image into a single color band. Use the interface to select which color band to view.

If Red is chosen:

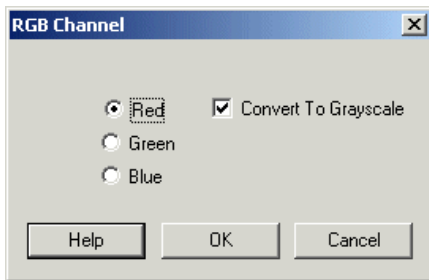
R = R

G = 0

B = 0

Note that white pixels will have the same intensity as pure red pixels. If you want to segment an image into a particular color try the RGB filter instead.

Interface



Instructions

1. Specify which color channel you would like to view.
2. Select "Convert to grayscale" if you would prefer to see the intensities as grayscale instead of in the selected color.

Example

Source Image



Red Channel



Green Channel



Blue Channel



See Also

[RGB Filter](#)
[YCbCr Channel](#)

RGB Filter

The RGB Filter uses RGB values to focus the attention towards the primary RGB colors. Depending on the color selected this filter will diminish all pixels that are not of the selected colors. This function is different than RGB Channel in that white pixels are also diminished even though they may contain the color selected.

For example, if Red is chosen:

$$R = ((R-B)+(R-G))$$

$$G = 0$$

$$B = 0$$

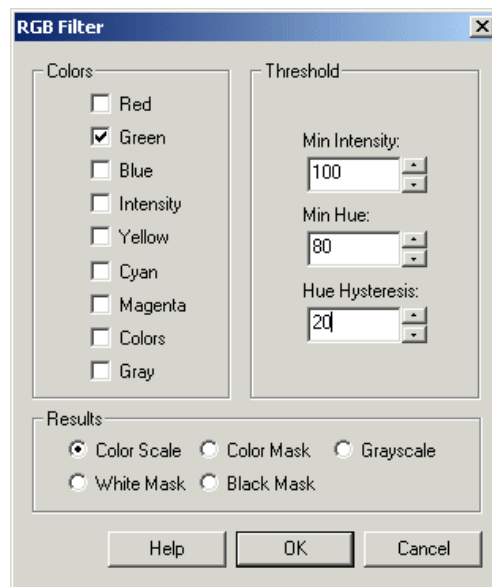
R is then normalized with respect to the maximum red value.

Based on the above formula it can be seen that white pixels result in a zero value whereas pure primary colors (R=255, G=0, B=0) R doubles its value. Thus function does a better job than RGB Channel in filtering for a particular color as white light is removed.

Due to normalization really dark pixels can be elevated in intensity and generate too much noise in the resulting image. The Min Pixel Value allows you to specify a minimum value below which pixels are considered to be black and will be ignored when calculating the image results. Default value is 40 (0-255).

You can use this filter to focus the image towards certain colors even with diminished lighting conditions.

Interface



Instructions

1. Colors - Chose the desired color Red, Green, Blue, etc. by selecting the appropriate checkbox. Note that "Colors" refers to those pixels that have strong colors in the image regardless of what color the pixel is (color saturation) while "Gray" refers to how close to a gray color pixels are. These two selections can be used to diminish colors and pronounce gray areas ("Gray") or diminish white areas and pronounce colors ("Colors").
2. Threshold - Select the min intensity pixel value that specified which values should be removed that are below a certain threshold. This helps remove pixels that are dark and do not contain enough color information. This is typically seen when detecting blue in dark areas.
3. Threshold - Select the hue threshold. This removes colors that are not 'blue' enough or not 'red' enough, etc.
4. Threshold - If needed select a hysteresis level. The hysteresis level will allow blobs that have at least one pixel above the hue threshold to grow into the surrounding area as long as the hue level is above the threshold minus the hysteresis level. Thus if the hue threshold is set to 80 and the hysteresis is set to 20 then only blobs that have a pixel above 80 will be preserved and any that do will include all pixels with hue above 60. This is useful to grow/connect a blob below the hue threshold while still preserving the filtering effects of the hue level.
5. Results - Select how the results should be represented
 - RGB Scale - resulting values are scaled RGB values from 0 to 255 that depend on how close the original color is to one in the Colors list
 - RGB Mask - resulting value is the color that is most similar to one in the Color list
 - Grayscale - resulting value is how close the color is to a color in the Colors list but represented in grayscale values

White Mask - resulting match value is white, non-match is black

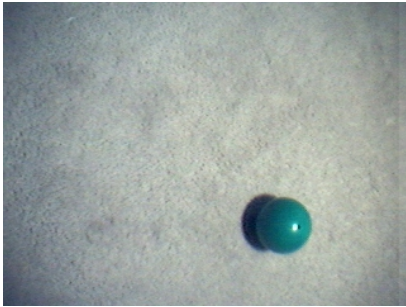
Black Mask - resulting match value is black, non-match is white

Fore Masked - resulting image shows original color where a match is made

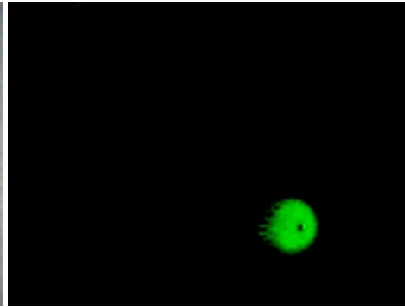
Back Masked - resulting image shows original colors where a match was NOT made. This is useful in determining how well your matched colors are working on the actual image.

Example

Source Image



Green Filter with 10, 30 threshold



Variables

RGB_FILTER_COLORS_COUNT

RGB_FILTER_GRAY_COUNT

RGB_FILTER_RED_COUNT

RGB_FILTER_GREEN_COUNT

RGB_FILTER_BLUE_COUNT

RGB_FILTER_INTENSITY_COUNT

RGB_FILTER_CYAN_COUNT

RGB_FILTER_MAGENTA_COUNT

RGB_FILTER_YELLOW_COUNT

- specifies the number of pixels detected belonging to that particular color/intensity category. Note that even low intensity pixels may be included.

See Also

[Color Filter](#)

[RGB Channel](#)

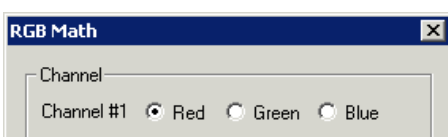
[Threshold](#)

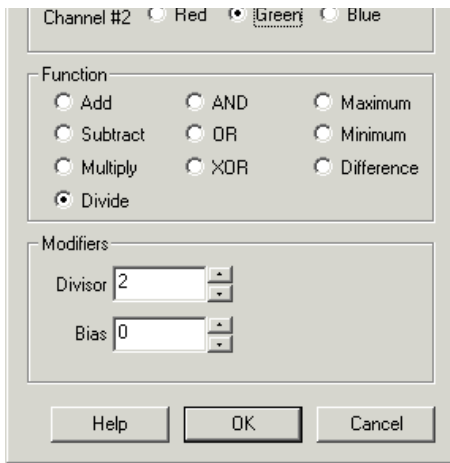
[Auto Threshold](#)

RGB Math

The RGB Math module provides you with a way to perform arithmetic operations between the color channels within the current image. This module is similar to the [Color Normalization](#) module in that it can highlight certain colors with an image. By using color channels the resulting image can be made much less sensitive to light changes.

Interface





Instructions

1. Select the appropriate Color Channel #1 and Channel #2 on which to perform the function. Note that the functions below are performed on two channels within the same image.
2. Select the appropriate function used to combine the two images.

Add Channel #1 + Channel #2

Subtract Channel #1 - Channel #2

Multiply Channel #1 * Channel #2

And Channel #1 & Channel #2 (binary AND)

Or Channel #1 | Channel #2 (binary OR)

XOr Channel #1 ^ Channel #2 (binary XOR)

Maximum if Channel #1 > Channel #2 then result = Channel #1 else result = Channel #2

Minimum if Channel #1 < Channel #2 then result = Channel #1 else result = Channel #2

Difference | Channel #1 - Channel #2 | (Absolute value of subtraction)

Divide Channel #1 / Channel #2

3. Select the appropriate divisor and bias.

Divisor - divides the function's result by the specified amount.

Bias - added to the function's result.

Example

Source Image

Red / Green



Notice the orange cone above that is now highlighted in the Red / Green image?

See Also

[Normalize Color](#)

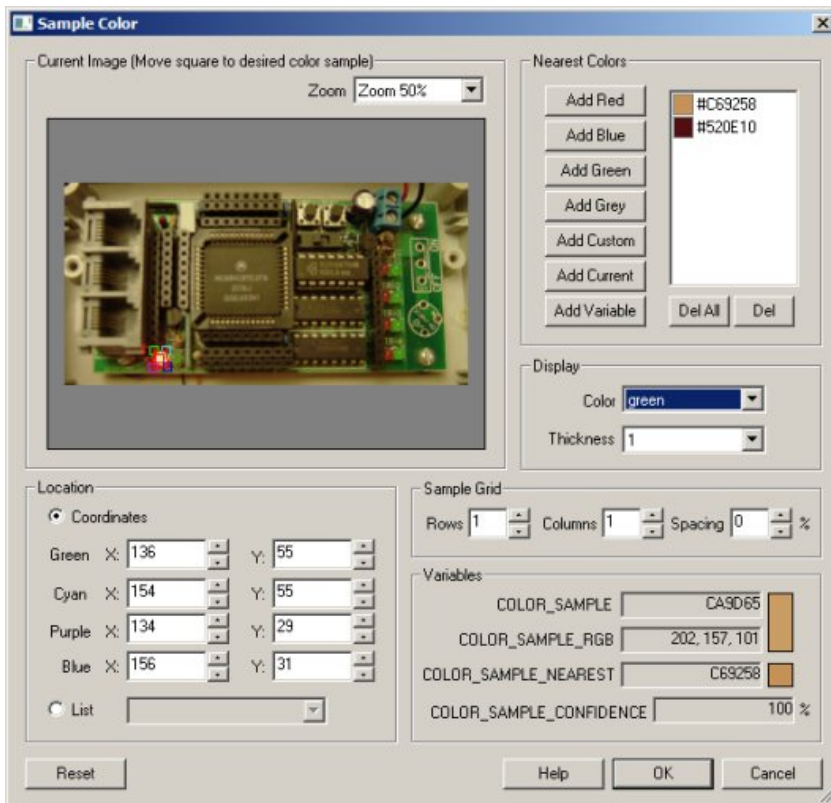
[Image Math](#)

Sample Color

The Sample Color module provides a way to compare a specific image area with known colors to determine which color is most representative of

the area. The module performs an averaging over the specified image area and compares this resulting value with color values entered into the Nearest Color list. Successive modules can then access the results of this module via variables to trigger actions based on the results of this module.

Interface



Instructions

1. Current Image - Move the red square to the place you want to sample the colors. You can move the entire square by dragging the smaller red square in the center or by adjusting the individual corner squares. To fine tune the positioning you can change the coordinates of the overall square using the textboxes below in the Coordinates area.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the square sample area of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Add Red/Blue/etc - Add the colors to compare to the sample square by clicking on one of the appropriate buttons.

3. Add Custom - Add a custom color to the list

4. Add Current - Add the current intensity to the list

5. Add Variable - Add a variable that contains a color to compare to

6. Del/Del All - Delete a or all the selected colors in the list.

7. Display - Select the color and thickness of the line that will be drawn on the actual image to reflect the sample area.

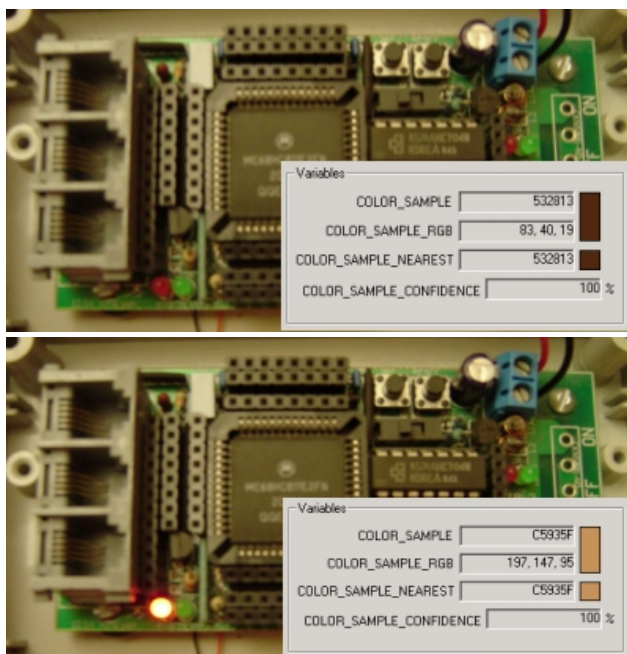
8. Sample Grid - Specify how the grid sample area is broken into rows and columns. Creating more sample squares will convert the results into an array of values.

9. Coordinates - As apposed to moving the editing squares in the Current Image area you can fine tune the positions by updating the coordinates of each of the editing squares. Note that you can also use the [variable] [expressions](#) in each of the coordinate boxes to automatically move the sample area to different locations. You can also specify an array to be used as the input to sample locations. This array should contain at least 8 points per area (4x X,Y locations).

10. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

Example

Source



Variables

COLOR_SAMPLE - holds the current average intensity in hex format

COLOR_SAMPLE_RGB - holds the current average intensity in rgb integer format

COLOR_SAMPLE_NEAREST - holds the nearest color hex value to the sampled area

COLOR_SAMPLE_NEAREST_RGB - holds the nearest color rgb value to the sampled area

COLOR_SAMPLE_CONFIDENCE - specifies the confidence amount (0-100%) that

the nearest color is the current color.

See Also

[Line_Profile](#)

[Sample_Line](#)

[RGB_Filter](#)

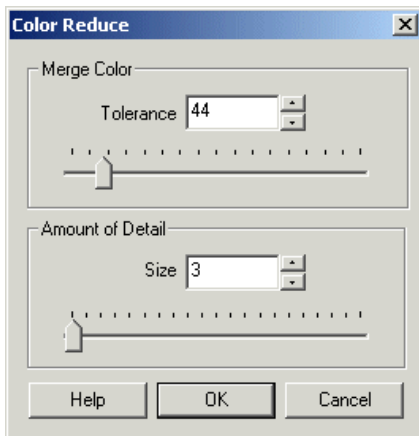
[Color_Probe](#)

Segment Colors

The Segment Colors module is functionally similar to the [Flood Fill](#) module which reduces the number of colors in an image in an effort to create meaningful blobs. Meaningful blobs are pixels grouped together that share a common color that hopefully make up a representative grouping of an object to be analyzed.

The Segment Colors differs from the Flood Fill in that it uses a faster and more stable blob creation routine that should create blobs that are more similar to past blobs in preceding frames. The Flood Fill module has a tendency to flicker the blob's color when viewed at a very high level of grouping that can cause blobs not to be matched correctly in successive frames. In addition to the other modules the Segment Colors offers a new way to create blobs.

Interface



Instructions

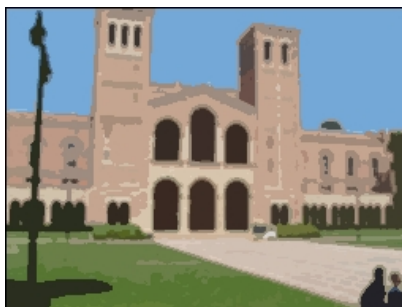
1. Tolerance - Select the Tolerance level for merging of colored pixels. Lower values create larger blobs (and fewer colors)
2. Detail Size - Select how much detail should remain in the blob image. Larger values will create bigger connected blobs. Smaller values (including 0) will retain single pixel blobs and best represent the image pixel colors.

Example

Source



Segment Colors



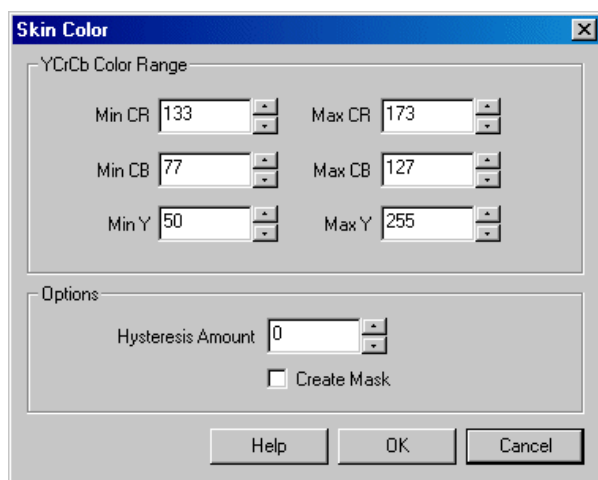
See Also

[Flood Fill](#)
[Reduce Colors](#)
[Color Depth](#)

Skin Color

The Skin Color module will eliminate any non skin color pixels from the image. Note that different cameras will have slight color distortions that may cause this module to not pick up all skin color types.

Interface

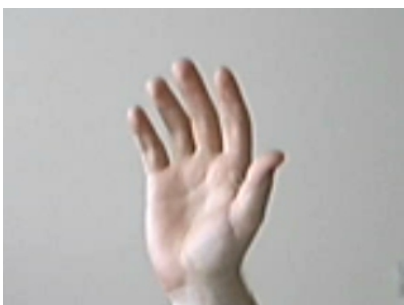


Instructions

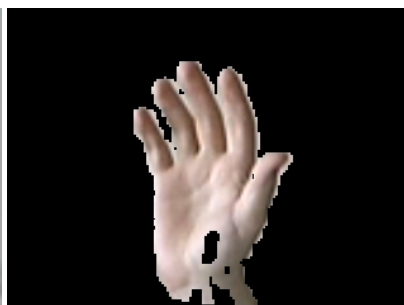
1. YCrCb Colors - The RGB image is converted into the YCrCb color space and thresholded against the appropriate numbers that have been found to be the best values to accommodate a large amount of skin colors. If you need you can change these values to better fit your skin samples. Keep in mind that when adjusting these parameters you may cause some skin types not to be detected.
2. Options - You can adjust the Hysteresis value for the detected skin colors. Increasing the value will gain back colors adjacent to detected skin colors. This allows for colors that are very close to skin colors to be detected assuming that they are touching valid skin colors.
3. Create Mask - Replaces all the detected skin colors with white pixels that can be used as a mask in other modules.

Example

Source Image



Skin Color Only



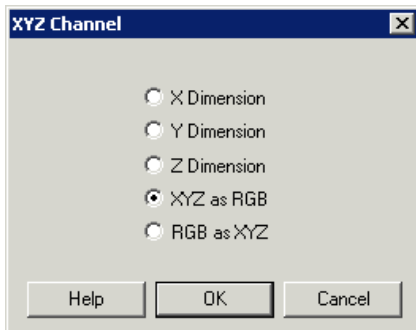
See Also

[RGB Filter](#)

XYZ Channel

The XYZ Channel module provides a way to convert to and from the XYZ color space. The XYZ color space was created in 1931 by the CIE to produce every color with positive tristimulus values.

Interface



Instructions

1. Specify the transform to apply to the current image

X - The X dimension of the XYZ color space

Y - The Y dimension of the XYZ color space

Z - The Z dimension of the XYZ color space

XYZ as RGB - convert the current image into XYZ color space and apply each of the XYZ channels into the RGB channels

RGB as XYZ - convert from XYZ space into RGB space by assuming that the current RGB image includes XYZ color space data.

Example

Source



XYZ as RGB (note the space appears very dark in RGB)



See Also

[YCbCr Channel](#)

[HLS Channel](#)

[RGB Channel](#)

YCbCr Channel

The YCbCr channel module converts the image into a luminance, chroma blue, and chroma red components instead of the RGB representation.

Similar to the YUV model the YCbCr color space attempts to better model the human color perception (although not as accurately as the HLS color space).

Using the YCbCr color space you can determine how the image you are currently viewing appears from a purely intensity or color hue point of view. Doing so may allow more precise color detectors to be created since color intensity is removed when viewing the Cr or Cb vectors.

Interface



Instructions

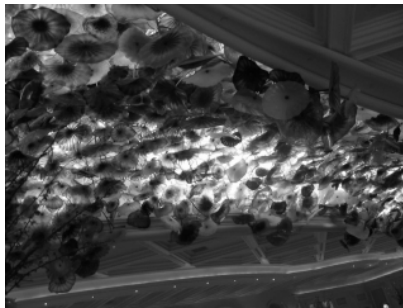
1. Specify the appropriate color vector to view by clicking on one of the radio buttons. The image display will update to show that color vector. Note that the selected vector is displayed in the RGB color space for purposes of viewing.

Example

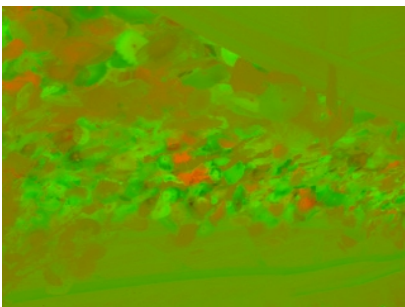
Source Image



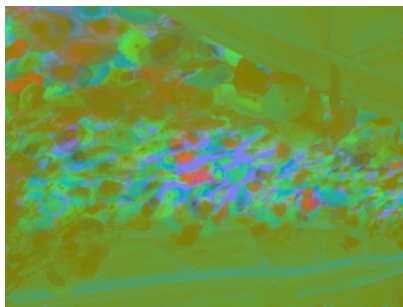
Y Channel



CbCr Channel



As RGB



See Also

[YUV Channel](#)

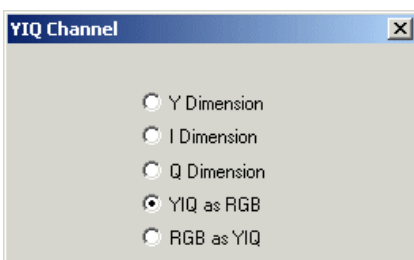
[HLS Channel](#)

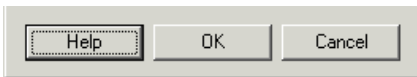
[RGB Channel](#)

YIQ Channel

The YIQ Channel module provides a way to convert to and from the YIQ color space. The YIQ system is used by National Television System Committee (NTSC) for color TV broadcasting.

Interface





Instructions

1. Specify the transform to apply to the current image

Y - The Y dimension (lightness) of the YIQ color space

I - The I dimension of the YIQ color space

Q - The Q dimension of the YIQ color space

YIQ as RGB - convert the current image into YIQ color space and apply each of the YIQ channels into the RGB channels

RGB as YIQ - convert from YIQ space into RGB space by assuming that the current RGB image includes YIQ color space data.

Example

Source

YIQ as RGB



See Also

[YCbCr Channel](#)

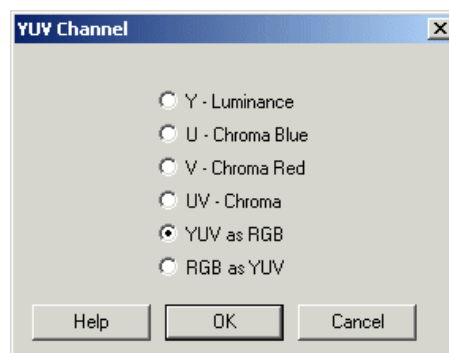
[HLS Channel](#)

[RGB Channel](#)

YUV Channel

The YUV channel module converts the image into a luminance, chroma blue, and chroma red components instead of the RGB representation. Similar to the YCbCr space the YUV space attempts to better model the human color perception (although not as accurately as the HLS color space).

Interface



Instructions

1. Specify the appropriate color vector to view by clicking on one of the radio buttons. The image display will update to show that color vector. Note that the selected vector is displayed in the RGB color space for purposes of viewing.

Y - Luminance or light

U - Chroma Blue or Cyan Hue Dimension

U - Chroma Blue or first color dimension

V - Chroma Red or second color dimension

UV - Chroma or color dimensions

YUV as RGB - convert from RGB to YUV by representing each YUV channel as RGB channels

RGB as YUV - convert from YUV to RGB by assuming each RGB channel as YUV channel

Example

Source Image

Y Channel



V Channel

As RGB



See Also

[YCbCr Channel](#)

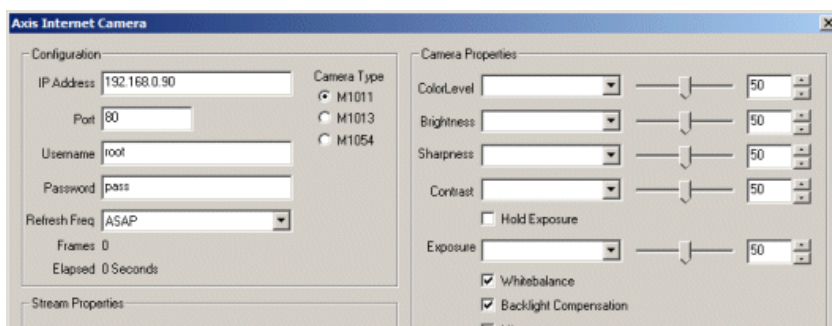
[HLS Channel](#)

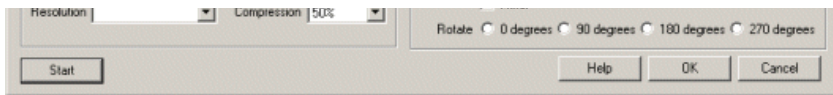
[RGB Channel](#)

Axis Internet Camera

The Axis Internet Camera module provides an interface to the Axis camera series. This module is similar to the [HTTP Read](#) in that it reads images over the Internet but also provides an easier interface to the camera adjustment capabilities of the Axis cameras. Note that when accessing cameras over the Internet there can be significant delays between the image capture time and viewing time.

Interface





Instructions

1. IP Address - specify the IP address of the camera that you would like to connect to. (Default is 192.168.0.90)
2. Port - If you are using a different port number than port 80 specify that port number here.
3. Username - the username needed to access the camera. (Default is root)
4. Password - the password needed to access the camera. (Default is pass)
5. Refresh Freq - how quickly the system should request new images. This will allow you to reduce the Internet traffic of the streaming video if you don't need rapid updates. Default is "As Fast As Possible"
6. Resolution - The image size dimension to request from the camera.
7. Compression - The amount of compression to apply to the image prior to transmission from the camera.
8. ColorLevel - refers to how much color/saturation to apply to the image.
9. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.
10. Sharpness - refers to the amount of sharpening to apply to the image.
- 11 Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.
12. Hold Exposure - Holds the current exposure level
13. Exposure - Specifies the relative amount of exposure time to apply to the image. This will also cause the camera to adjust the exposure to keep the same level.
14. White balance - Specifies that automatic white balance be applied to the image
15. Back light Compensation - Specifies that back light compensation be applied to the image.
16. Mirror - Mirrors (flips horizontal) the image coming from the camera.
17. Rotate - Rotates the image by the specified degrees.

Edit the camera properties by either selecting the appropriate value using the slider, typing in the number in the available edit area, increasing/decreasing the number using the spin/up,down arrows, or select a variable that contains the desired number to be used as configuration in this module. Note that once a variable is selected the manual controls will be disabled to indicate that the property is under variable control.

Tips

1. Finding the IP address of the camera can be tricky. If you have not used the camera in a while it may not be setup to automatically request a DHCP IP address and thus may still be using an IP address that is no longer accessible on your network. In this case Axis recommends to reset the device (instructions below) and use a cross-over Ethernet cable to connect the camera directly to your computer. You will then have to change your computers local LAN connection to a 192.168.0.X number (where X is any number but 90, for example 10). This requires you to access the Network setup in your control panel.

Note that all modern computers are able to self correct even if a non-cross-over Ethernet cable is used, so if you do not have a cross-over cable just plug in any that you have around (we've verified this works with the Axis camera).

2. If you had set the camera to automatically acquire an IP address (just like when your laptop joins a wifi network) and you don't know what IP address is provided, you will have to check your router interface which would supply you a network list of connected devices. If you don't have access to this you can try using a network scan application like [netscan.exe](#) to scan your network for active devices. The camera should then appear, if it does not, you will have to reset it. Again, when you reset it and your network is not 192.168.0.X then you will still not see the IP address even using the scanner. If you and the camera are on different network numbers you will just not see each other.

3. If you do NOT enter a username and password you will not be able to change the camera parameters. You will also NOT be able to view the camera image UNLESS you have the "Basic Setup->Users->Enable anonymous viewer login" checkbox set within the Axis Administration configuration. To avoid these complications, be sure to enter in the supplied username and password.

4. The default username and password is normally root:pass but this may have been changed on first login into the interface. If you have forgotten the password you will have to reset the device. You can do so by switching off the camera, hold the white button on the back of the camera while

the password you will have to reset the device. You can do so by switching off the camera, hold the white button on the back of the camera while turning it back on. Hold that button until the light around the camera lens turns amber (as apposed to the normal green) which indicates that you have reset the camera. Now when you access the administration interface via a browser on IP address 192.168.0.90 it will ask you for the username/password combination to use.

See Also

[HTTP Read](#)

[DLink Internet Camera](#)

[Linksys Internet Camera](#)

[TRENDnet Internet Camera](#)

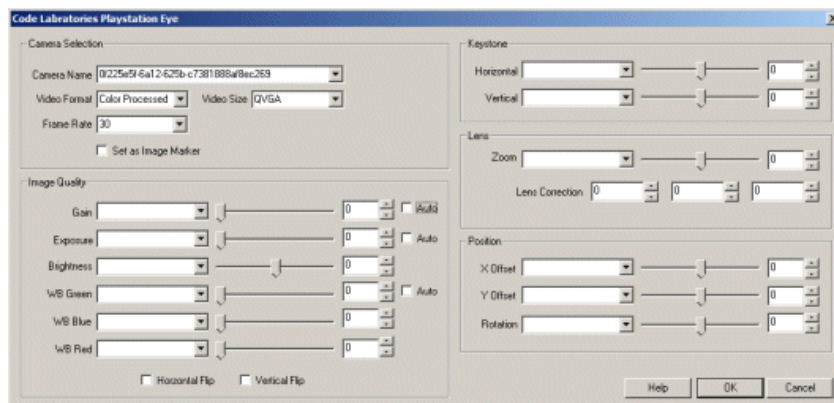
Code Laboratories Playstation Eye



The Code Laboratories Playstation Eye module provides an interface to the Playstation Eye Camera using the Code Laboratories Eye driver. The CL Drivers provide numerous interface components to the Playstation Cameras including the ability to access more than one camera at a time. This provides the ability to utilize these cameras for stereo purposes.

While not typical webcams the Playstation Eye cameras do provide for very high speed fps (frames per second) and good low light sensitivities which can make the cameras very useful in certain situations. Additional calibration and keystoneing capabilities provide for numerous configuration options which make the camera appropriate for many applications.

Interface



Instructions

1. Camera Name - select the appropriate camera. This is the camera GUID as defined by the camera hardware.

2. Video Format

- CLEYE_MONO_PROCESSED - black & white processed image (calibration & keystoneing)
- CLEYE_COLOR_PROCESSED - color processed image (calibration & keystoneing)
- CLEYE_MONO_RAW - black & white not processed
- CLEYE_COLOR_RAW - color image
- CLEYE_BAYER_RAW - unprocessed color image

3. Video Size

- QVGA - 320x240 image size
- VGA - 640x480 image size

4. Frame Rate - The requested frames per second from the camera

5. Set As Image Marker - Save the current image as a marker (memory image).

6. Gain - refers to the amplitude or magnification of the incoming video. Higher gain levels result in greater levels of brightness and contrast. Lower

levels of gain will darken the image, and reduce the contrast. Essentially, gain modification affects the sensitivity to light of the CCD sensors. This concept is analogous to the ISO or ASA ratings of silver halide films in digital cameras.

7. Exposure - refers to how long your camera takes to record an image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.

8. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.

9. Whitebalance Red/Blue/Green - refers to the adjustment of the relative amounts of red, and blue primary colors in an image such that neutral colors are reproduced correctly.

10. Keystone - Adjusts the horizontal and vertical perspective to adjust for non-perpendicular image planes.

11. Zoom - Adjusts the amount of image zoom.

12. Lens Correction - Amount of radial correction to perform on the image to correct for lens distortion.

13. X Offset - Horizontal offset from center of screen to center of imaging CCD.

14. Y Offset - Vertical offset from center of screen to center of imaging CCD.

15. Rotation - Amount of image rotation to correct for rotation of imaging CCD.

Notes

You will need to download and install the [Code Laboratories Platform SDK](#) as the module needs the CLEyeMulticam.dll in order to work. You should also NOT be currently accessing the camera from the built in webcam accessible camera. As webcams are NOT shared resources you can only access the camera from one application at a time. Be sure the CAMERA button is not currently connected to the same camera.

Variables

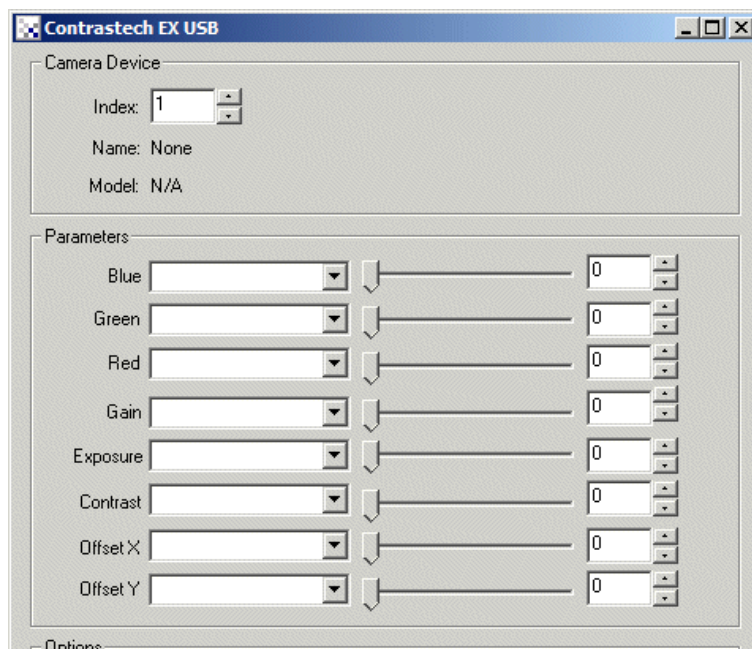
CL_EYE_IMAGE - image name when saved as a marker

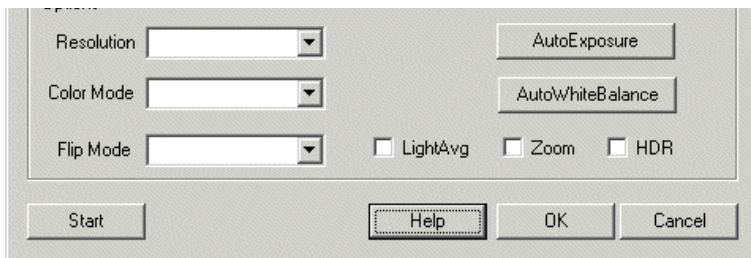
Contrastech EX



The Contrastech module provides an interface from RoboRealm to their USB lines of machine vision cameras. You can find Contrastech's product list [here](#) or for a more English version [here](#).

Interface





Instructions

1. Index - The camera device index to connect to if more than one camera is connected.
2. Red/Green/Blue - The gain for each color channel.
3. Gain - refers to the amplitude or magnification of the incoming video. Higher gain levels result in greater levels of brightness and contrast. Lower levels of gain will darken the image, and reduce the contrast. Essentially, gain modification affects the sensitivity to light of the CCD/CMOS sensors. This concept is analogous to the ISO or ASA ratings of silver halide films in digital cameras.
4. Exposure - refers to how long your camera takes to record an image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.
- 5 Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.
6. OffsetX / OffsetY - You can use the OffsetX and OffsetY parameters to change what portion of the sensor pixels you want to capture. This allows one to specify a global AOI (Area of Interest) which helps to reduce processing requirements and network consumption.
7. Resolution - Select what resolution the desired image should be. Note that larger images are not always better!
8. Color Mode - Specify the desired color mode. If no color is required, you can chose GRAY which will help to
9. Flip Mode - If the image is in the wrong orientation select how to rotate the image to correct this.
10. Light Average - If your illumination flickers, select the Light Averaging feature to help stabilize the lighting.
11. Zoom - select if you want to zoom into the current image to enlarge distant features.
12. High Dynamic Range - If your image has an uneven lighting, try using the HDR setting to even out the lighting.
13. AutoExposure / AutoWhiteBalance - Press the appropriate button to set the specified attribute by calculating a good setting from the current image.

Variables

CONTRASTECH_CAMERA_NAME - name of the connected camera

CONTRASTECH_CAMERA_SERIAL - serial number of the connected camera

CONTRASTECH_CAMERA_MODEL - model of connected camera

See Also

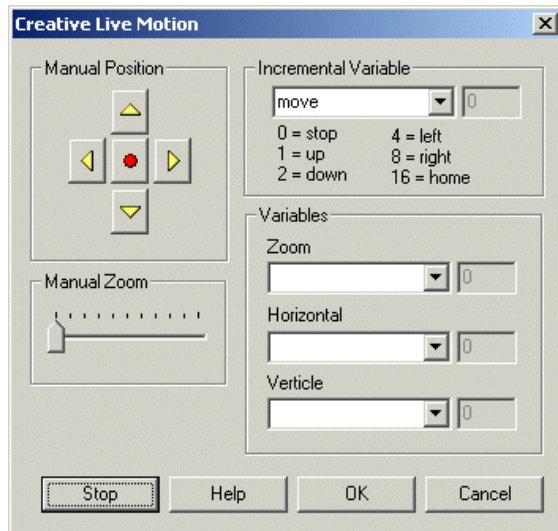
[GenICam](#)

Creative Live Motion Pan/Tilt

The Creative Live Motion module provides control over the Creative Live Motion's motorized pan/tilt capabilities. Similar to servo control systems the Creative Live Motion provides left and right panning and up and down tilting. Using the Creative Live Motion module you can control the camera based on what it sees.

Note for compatibility with other systems this module features redundant variables. The first variable is used similar to the [Logitech Orbit](#) camera in that it specifies incremental changes to the current position. The second set of variables is unique to the Creative Live Motion system in that they offer absolute positioning of the camera, In this way it is possible to memorize certain configurations and reproduce the exact pan/tilt/zoom coordinates to go back to a known position.

Interface



Instructions

1. To test out the pan/tilt select and hold the appropriate buttons. You should see the camera start moving. The red center button will move the camera to coordinates 0,0,0 (pan, tilt, zoom)
2. To test the zoom drag the scroll bar and the image will zoom/widen depending to the slider amount.
3. To automatically control the camera select a variable that will contain relative increments based on the code specified in the interface, i.e. 1 for up, 2 for down, etc. This command variable should contain the following values as appropriate:

0 = stop

1 = up

2 = down

4 = left


8 = right

16 = home

4. Or specify variables that contain the absolute positioning for zoom, pan, and tilt.

Note that even though the Creative camera has absolute positioning the pan range is from -70 to 70, tilt from -30 to 30 and zoom from 0 to 50. The values you provide are scaled to those numbers from -255, 255 for pan and tilt only. Thus you may notice that small value changes will NOT produce any movement within the camera since the camera only moves on increments of 10 in either pan or tilt.

Example

 [Click here](#) to load a configuration to move the camera using your cursor keys.

Variables

CREATIVE_PAN - the current pan (horizontal) angle of the camera

CREATIVE_TILT - the current tilt (vertical) angle of the camera

See Also

[Logitech Orbit](#)

[DLink Internet Camera](#)

[Dream Cheeky Missile Launcher](#)

Creative Senz3D



The Creative Senz3D module provides access to the Creative Senz3D RGB-D camera. Similar to the Kinect and the Xtion Pro, the Senz3D provides information about the distance to objects as well as RGB images. The main differences between the Senz3D is the method which is used to determine distance. The Kinect and Xtion both used a projection system to project a dot pattern which is then processed accordingly. The Senz3D uses 'time-of-flight' in that it determines the amount of time it takes for light to travel and reflect from objects within its field of view. Because of this technique, edges are sharper than seen in the Kinect but at a lower resolution (320x240 is the maximum depth image size in the Senz3D).

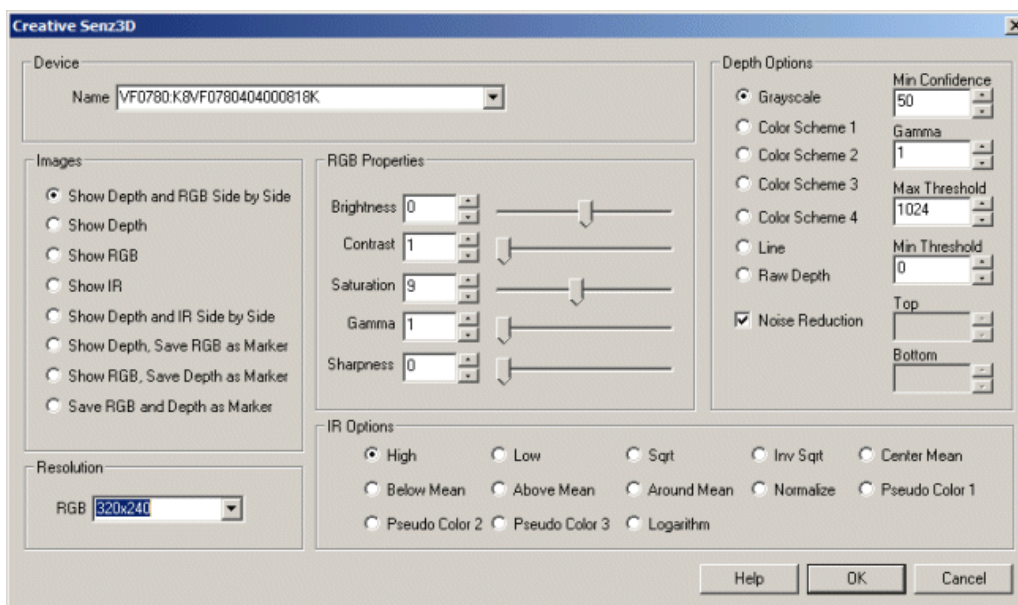
Other differences include only requiring a single USB cable (no power cable like the Kinect) and much smaller form factor. The Senz3D is rated for 0.5ft to 3.25ft which is a much shorter range than the Kinect but allows for closer objects to register (the Kinect has about a 2ft blind spot for objects too close). The IR image in the Senz3D is higher bit depth which allows for more of the scene to be viewed (i.e. it is a very useful night camera).

Currently, the camera does not appear to be as popular as the Kinect largely due to lack of apps/games/etc. Due to this it is currently available from the Creative website at \$99.00 USD. This is the cheapest depth camera that we are currently aware of (as of Jan 12th, 2015).

A note on installation: While the Senz3D is part of the Intel Real-Sense initiative, we found that installing the [32bit DepthSenseSDK](#) from SoftKinetic rather than the Creative or Intel programs that came along with the device to be much quicker and provide just the basics to use the camera. The applications provided along with the camera appear to be quite large (several gigs) and install various webcam annoyances that caused ALL webcams to suddenly be under face tracking control!

Once we understood what to install, the installation process is much easier than any other RGBD camera that we have tried. The Kinect continues to be very difficult to install and v2 now requires USB3.0 and Win8 to work. We found the Senz3D to work well on Win7 with USB 2.0.

Interface



Instructions

1. Device - Select while Senz3D device you wish to connect to.
2. Images - Select what data you'd like to view and which you'd like to save as a Marker. [Markers](#) (images saved in memory) can be brought up

later and used in other modules.

3. RGB Brightness - refers to an intensity or luminosity scale of the RGB image that ranges from totally black to totally white and has no effect on color values.
4. RGB Contrast - refers to how far pixel values can deviate from gray in the RGB image. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.
5. RGB Saturation - refers to the intensity of the RGB colors, i.e. how red a red color is. By decreasing the saturation of an image you remove color and produce a monochrome grayscale picture that represents only darkness and brightness, or luminance. Increasing saturation in an image produces artificially intense colors.
6. RGB Gamma - specify the gamma gain to be applied to the image. Note that a gain < 1.0 will increase the overall intensity of the image whilst a gain > 1.0 increases the image's contrast.
7. RGB Sharpness - refers to strong the edges in the image are.
8. Depth Options - The depth image that the devices produce is a 10 bit (0 - 1024 values) number that can be displayed in several ways. The most common depth map display uses intensity (black to white) to indicate the distance to any particular pixel within the image. This can be useful for successive processing (for example [threshold](#)) but may not best indicate the different depth levels. Thus you can use the different color schemes to provide more insight into what depth levels are present in the image.

Grayscale - reduces the depth information from 10 bits to 8 for each RGB pixel

Color Scheme X - uses the depth value as an index into a color palette. Each palette uses a different color transition.

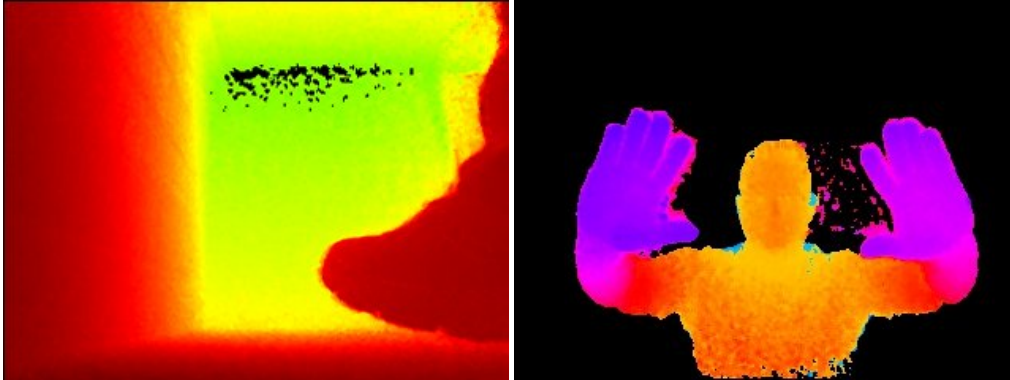
Line - produces a line that indicates the closest depth to an object. This is similar to what a SICK or LIDAR laser would produce. Note this option enables the Top and Bottom settings which allow you to crop the depth map to a specific height. Reducing the height allows objects that are above or below the needed detection level to be ignored when creating the distance line.

Raw Depth - the 10 bit number is encoded with 8 bits in the red channel and the remaining 2 in the green channel. This allows you to access the full depth resolution in external applications by decoding the R and G channel. (Note the B channel is set to 0).

9. Noise Reduction - Due to the depth determination method the device can produce a lot of noisy pixels. By selecting the Noise Reduction checkbox a [Median](#) filter is automatically applied to help reduce the noise. This is very noticeable when using the Line mode which looks for maximum valued pixels and is very noise prone.
10. Min Confidence - Each depth value is also correlated with a confidence map. To eliminate very noisy pixels increase the confidence until the pixels stabilize. Note that pixels representing objects further away are often more noisy than closer pixels. The confidence measure is essentially how bright a particular pixel is within the IR space. The brighter (and closer) a pixel is the higher the confidence is on its depth.
11. Gamma - While the devices can produce up to 1024 depth levels not all the values will be displayed or used given a particular scene. You may want to focus more on closer depths and lose some resolution at more distant depths (since gray images are 0-255 you lose 2 bits). Using the gamma correction can help to bunch up values to better highlight the different depth values.
12. Min/Max Threshold - You may not want objects that are too far or too close from the devices to appear in the depth map. Using the threshold value you can remove far away objects and also compress the color space of the resulting image to better show depth differences. Using the threshold you can eliminate all but nearby objects that can then be further processed by other modules.
13. Top/Bottom Line - Specifies the row to start and stop when displaying the Line view. This allows you to eliminate floor planes or objects that are too high or low to be of concern.
14. Resolution - The resolution for the RGB image. Note that the Depth and IR image are set at 320x240 and cannot be changed.
15. IR Options - The IR image pixel is 13 bits which is 5 more than what can be represented in an image. To focus what part of that range you want to preserve, select the appropriate technique. There are many ways this can be accomplished:

- High - Uses the upper 8 bits of the image
- Low - Uses the lower 8 bits of the image
- Sqrt - Square root's the image pixel to the 8 bit range
- Inv Sqrt - The Sqrt function will favor darker pixels, the Inv Sqrt favors lighter
- Center Mean - Forces the high bit range to be centered at the image mean
- Below Mean - Shows only pixel below the image mean compressed into 24 bits
- Above Mean - Shows only pixel above the image mean compressed into 24 bits
- Around Mean - Similar to Center Mean but thresholds values on either side to improve contrast
- Normalize - Determines image low and high values and scales to 24 bit
- Pseudo X - Translates image intensity into a higher color range for improved visibility
- Logarithm - Takes the logarithm of the image pixel and then stretches to 8 bit range

Example



Variables

SENZ3D_DEPTH - when saved as a marker this variable contains the depth image

SENZ3D_RGB - when saved as a marker this variable contains the RGB image

SENZ3D_LINE - array containing the Line data (only when line mode enabled)

See Also

[LeapMotion Controller](#)

[Microsoft Kinect](#)

[OpenNI 1.0 & 2.0](#)

[Hokuyo Laser](#)

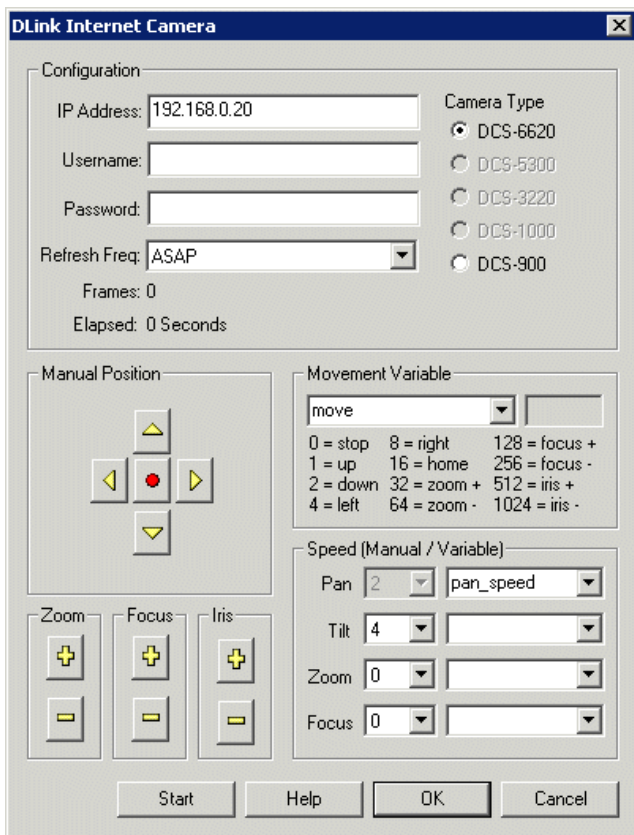
[Stereo Depth](#)

[Visible Laser Line](#)

DLink Internet Camera

The DLink Internet Camera module provides an interface to the DLink camera series. This module is similar to the [HTTP Read](#) in that it reads images over the internet but also provides an easier interface to the DCS900 and pan/tilt capabilities for the other DLink cameras. Note that when accessing cameras over the Internet there can be significant delays when panning or tilting the camera. Keep in mind the delay when using those buttons otherwise you will overreact the movement.

Interface



Instructions

1. IP Address - specify the IP address of the camera that you would like to connect to.
2. Port - If you are using a different port number than port 80 specify that port number here.
3. Username - the username for HTTP authentication access.
4. Password - the password for HTTP authentication access.
5. Refresh Freq - how quickly the system should request new images. This will allow you to reduce the Internet traffic of the streaming video if you don't need rapid updates. Default is "As Fast As Possible".
6. Camera Type - select which camera type you are connecting to. For disabled cameras (DCS 5300, DCS 3220 and DCS 1000) please [contact us](#) with a Internet accessible IP address to such a camera. Having access will allow us to QA the interface and enable that functionality.
7. Manual Position - use the buttons to move the camera as appropriate. The red center button will move the camera home.
8. Zoom - manual zooming
9. Focus - manual focusing.
10. Iris - manual iris change.
11. Movement Variable - the variable that contains the commands that will move or change the camera attributes. The following values within the variable will create the following movements:

0 = stop
 1 = up
 2 = down
 4 = left
 8 = right

- 16 = home
- 32 = tele-zoom
- 64 = wide-zoom
- 128 = near focus
- 256 = far focus
- 512 = open iris
- 1024 = close iris

12. Speed - manual or variable setting of pan, tilt, zoom and focus speeds. or variable zooming. Either select a manual speed from the dropdown or specify a variable that will contain the required speed. Note that the valid values are from -5 to 5. Values higher or lower will be truncated appropriately. You can use the variables to specify speeds in order to quickly make coarse or fine adjustments to the position of the camera instead of successive sequential small movements.

See Also

- [HTTP Read](#)
- [Foscam Internet Camera](#)
- [Linksys Internet Camera](#)
- [TRENDnet Internet Camera](#)

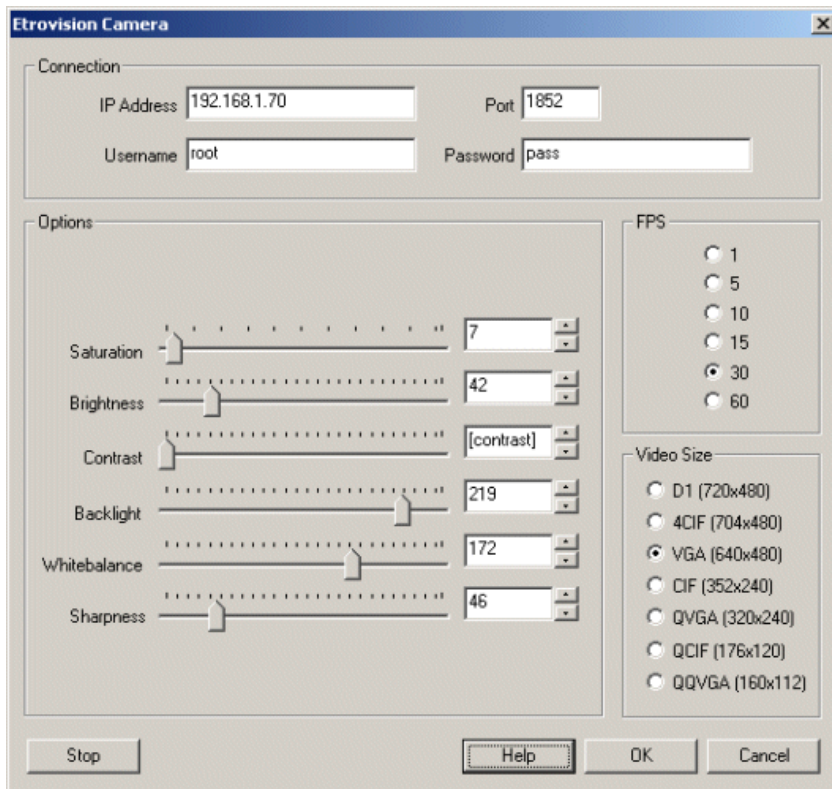
Etrovision Camera



The Etrovision Camera module provides an interface to the [Etrovision EV3151](#) video server to allow you to use images captured by this device within RoboRealm.

The EV3151A is a compact video server that provides H.264, MPEG-4, and M-JPEG dual streams simultaneously.

Interface



Instructions

1. IP Address - enter the IP address of the camera. If you do not know the address use the EtroScan application that was installed along with your camera to find its IP address. If this application does not provide any IP address ensure that the camera is powered and is connected to a network via a wired Ethernet link.
2. Port - enter the Port number that the camera listens to. This should be 1852 unless you have changed it.

3. Username/Password - enter in the username and password for the camera. These will be root/pass unless you have changed those. Note that it is recommended to change the username and password if you plan to make the camera available over the Internet.
4. Connect - Press the connect button to connect to the camera and start streaming the video into RoboRealm. If you have not entered the connection information correctly you will receive error messages. Correct these and try pressing the Connect button again. Once connected, the button will change function to a Stop button.
5. Saturation - If the image appears somewhat colorless try increasing the saturation which should increase the color within the image. Note that if you are using a black and white camera this setting will not change the image.
6. Brightness - If the image appears too dark try increasing the brightness to improve the image details.
7. Contrast - If the image appears too faded increase the contrast which will increase the black to white ratio which creates sharper edges and a more pronounced image.
8. Backlight - If you are taking pictures of an object in front of a bright light source try increasing the backlight value to compensate for this hard light source from overshadowing the object.
9. Sharpness - If the image appears a little blurred try increasing the sharpness to improve the details of the image.
10. FPS - Chose how quickly images are to be served from the camera. Higher FPS numbers provide smoother movements but increase the bandwidth required for viewing.
11. Video Size - The video size allows you to change the video size of the currently streaming image. Smaller sizes will require less bandwidth but provide less detail than higher resolutions. Currently the video size setting only works by clicking on a size several times and starting/stopping the connection. It appears that the camera does not respect the setting unless the active connection is severed but even this appears to be unreliable.

See Also

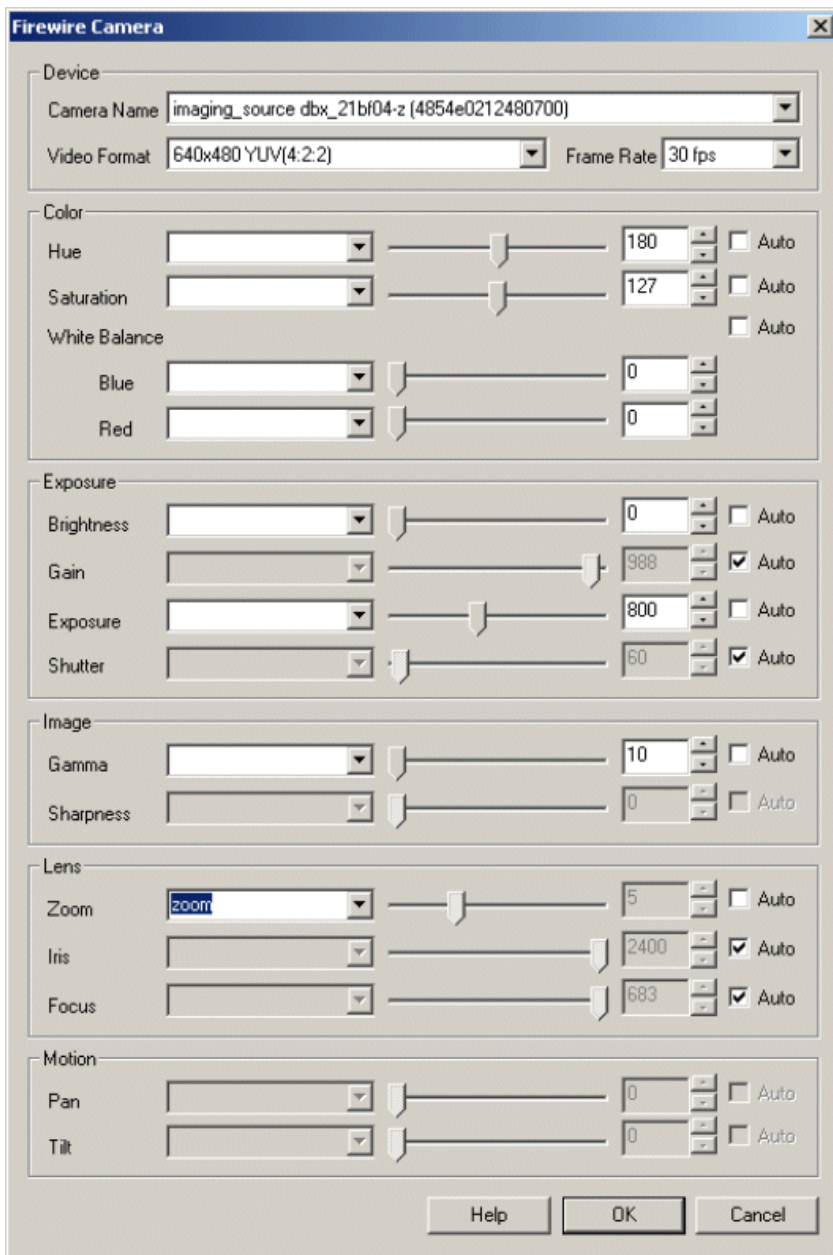
[DLink Internet Camera](#)
[TRENDnet Internet Camera](#)
[Linksys Internet Camera](#)
[HTTP Read](#)

Firewire Camera

The Firewire Camera module provides a way for RoboRealm to access firewire based cameras from your desktop PC. In order to gain access you will need to download and install the CMU 1394 firewire driver from [CMU 1394 Digital Camera Driver](#). This driver will provide access to most firewire cameras including the [DBX 21BF04-Z](#) from [The Imaging Source](#).

Note that some modes will require the use of the [Bayer Filter](#) module in order for the color to display correctly. If you see a black and white image that appears to have many light/dark gray dots in the image try adding the Bayer filter to introduce color into the image.

Interface



Instructions

1. Camera Name - select the appropriate Camera Name of the device you wish to connect to. The dropdown will show all the firewire devices accessible by the firewire module.
2. Video Format - select the video format that you wish to view. Note that the Monochrome formats often require the use for the [Bayer filter](#) module in order to display color in the image preview.
3. Frame Rate - select the maximum frame rate you desire. Note that the highest frame rate will be selected by default.
4. Edit the needed image attributes by either selecting the appropriate value using the slider, typing in the number in the available edit area, increasing/decreasing the number using the spin/up,down arrows, or select a variable that contains the desired number to be used as configuration in this module. If you select "auto" the internal camera statistics will determine an appropriate value to use and disable the scroll and variable selection. If you instead chose a variable that contains the value to use the manual interface is then disabled. The above screenshot shows the "zoom" variable being selected with a value of 5 that is created using either the [VBScript module](#) or the [Set Statement](#) module.
5. Hue - refers to the overall adjustment of image colors along the color spectrum. You can use it to warm-up (add red) or cool-down (add blue) to an image.
6. Saturation - refers to the intensity of the colors, i.e. how red a red color is. By decreasing the saturation of an image you remove color and

produce a monochrome grayscale picture that represents only darkness and brightness, or luminance. Increasing saturation in an image produces artificially intense colors.

7. Whitebalance - refers to the adjustment of the relative amounts of red, and blue primary colors in an image such that neutral colors are reproduced correctly.
8. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.
9. Gain - refers to the amplitude or magnification of the incoming video. Higher gain levels result in greater levels of brightness and contrast. Lower levels of gain will darken the image, and reduce the contrast. Essentially, gain modification affects the sensitivity to light of the CCD sensors. This concept is analogous to the ISO or ASA ratings of silver halide films in digital cameras.
10. Exposure - refers to how long your camera takes to record an image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.
11. Shutter - Same as Exposure
12. Gamma - refers to the adjustment of red, green, or blue values of a pixel to affect how bright the image appears.
13. Sharpness - refers to the amount of edge definition or crispness in an image. The sharpness control can be used to smooth out rough edges.
14. Zoom - refers to the cameras ability to adjust focal length (Optical Zoom) or digitally increase the image size (Digital Zoom). Some cameras have built in optical or mechanical zoom which physically changes the focal length of a camera. This is unlike most digital cameras that just increase the image size using mathematical interpolation and return a clipped portion of that image.
15. Iris - refers to the control the aperture, or iris, of the camera lens.
16. Focus - refers to the amount of edge definition or crispness in an image.
17. Pan - refers to the ability for a camera to move the image viewed in the horizontal direction or X axis. This can be done mechanically or digitally. Mechanical movement is when the camera has the ability to physically move using a mechanical pan mechanism. Digital panning refers to the ability for a camera to shift the image focus to a different part of the imaged CCD. This is possible when your image is smaller than the full resolution capable of the camera or when a digital zoom is active. If you are viewing the highest camera resolution then digital panning will not be available.
18. Tilt - similar to panning but meant for movement of the image in the vertical direction or Y axis.

See Also

[GenICam](#)

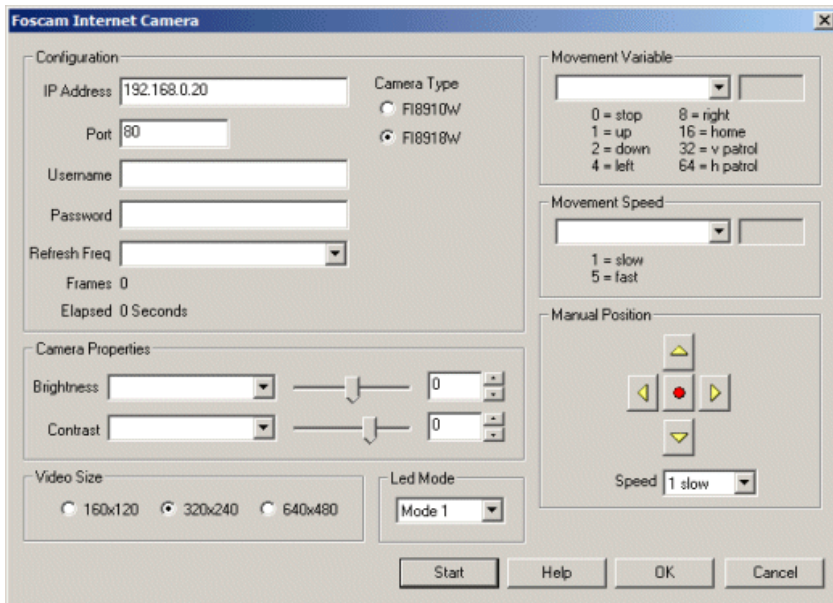
[Camera Properties](#)

Foscam Internet Camera

The Foscam Internet Camera module provides an interface to the Foscam camera series. This module is similar to the [HTTP Read](#) in that it reads images over the Internet but also provides an easier interface to the pan/tilt and camera adjustment capabilities for some of the Foscam cameras. Note that when accessing cameras over the Internet there can be significant delays when panning or tilting the camera. Keep in mind the delay when using those buttons otherwise you will overreact the movement.

To allow for RoboRealm to set the pan/tilt speed you will need to set the PTZ speed to 0 within the Foscam web browser setup. If not, the speed will not increase or decrease depending on the variable setting within the module.


Interface



Instructions

1. IP Address - specify the IP address of the camera that you would like to connect to.
2. Port - If you are using a different port number than port 80 specify that port number here.
3. Username - the username needed to access the camera.
4. Password - the password needed to access the camera.
5. Refresh Freq - how quickly the system should request new images. This will allow you to reduce the Internet traffic of the streaming video if you don't need rapid updates. Default is "As Fast As Possible"
6. Manual Position - use the buttons to move the camera as appropriate. The red center button will move the camera home/center.
7. Movement Variable - the variable that contains the commands that will move or change the camera attributes. The following values within the variable will create the following movements:
 - 0 = stop
 - 1 = up
 - 2 = down
 - 4 = left
 - 8 = right
 - 16 = home
 - 32 = vertical patrol
 - 64 = horizontal patrol
8. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values. Edit the camera properties by either selecting the appropriate value using the slider, typing in the number in the available edit area, increasing/decreasing the number using the spin/up,down arrows, or select a variable that contains the desired number to be used as configuration in this module. Note that once a variable is selected the manual controls will be disabled to indicate that the property is under variable control.
- 9 Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.

Example

 [Click Here](#) to download a robofile that interfaces the Foscam pan/tilt with a Joystick. The speed of the pan/tilt is determined by the joystick offset. Using this robofile you can move the camera quickly with large joystick movements and slowly with small joystick movements. Don't forget to configure your Foscam's IP address and username/password in the Foscam module. Also be sure to select the appropriate joystick device in the Joystick module.

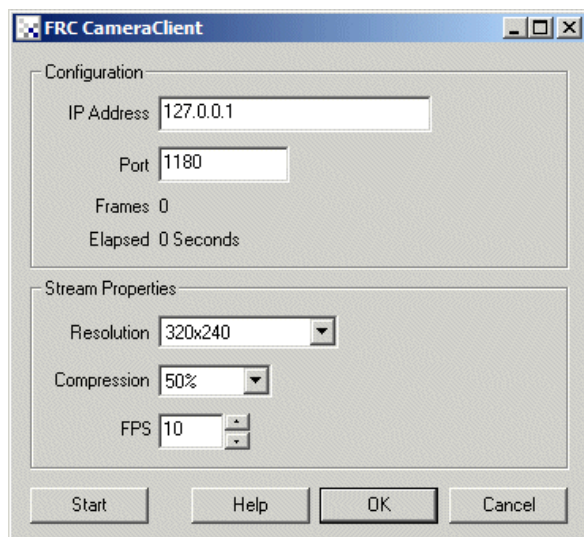
See Also

[HTTP Read](#)
[DLink Internet Camera](#)
[Linksys Internet Camera](#)
[TRENDnet Internet Camera](#)

FRC CameraClient

The FRC camera client module is a creation specific to the FIRST 2015 season. This module will communicate directly with the RoboRio to help expose an attached USB camera to RoboRealm running on a remote machine. Please note, the current specifications in the RoboRio are very sparse and subject to change without notice.

Interface



Instructions

1. Address - Specify your robot's roborio IP address
2. Port - CameraServer port, should normally be 1180 unless otherwise stated
3. Resolution - Specify how large an image you want the CameraServer to send. Note, there is a bandwidth limit enforced on all teams during the competition. This is NOT the responsibility of this module to replicate that.
4. Compression - Currently inactive, always set at 50 in the RoboRio. Included in case that feature gets implemented in the RoboRio.
5. FPS - Frames Per Second the camera server should send images to this module. Again, watch for that bandwidth limit.

See Also

[Axis Internet Camera](#)

GenICam

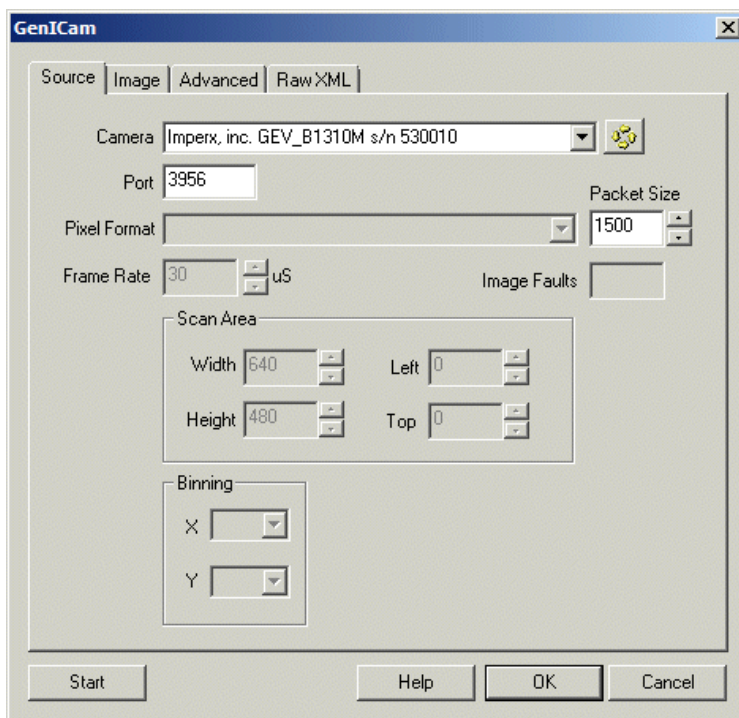
GenICam is a standard of the [European Machine Vision Association \(EMVA\)](#). The GenICam module provides access to GigE compliant cameras. The module will connect directly to the camera over an ethernet connection. The module assumes that the [genicam.org \(7 MB download\)](#) 32x runtime is installed which provides functionality to interpret the camera's configurations. The camera's native drivers need not be installed as communication functions over ethernet.

As this module communicates directly with the camera additional DirectShow drivers are not needed as with other cameras such as webcams. Using the RoboRealm Virtual Webcam driver you can expose a GigE camera as a webcam by utilizing this module and then activating the [virtual webcam](#) functionality within RoboRealm.

The GenICam standard includes the communication structure to GigE cameras but it does NOT enforce naming of custom features (such as setting exposure levels) which can differ in each camera implementation. The module attempts to make some assumptions about how to set exposure, black levels and gain but this may not function for your specific camera. If this is the case, see the Advanced Tab which shows all the possible settings for your camera and look for exposure related features. Typically you need to enable the manual setting and then adjust a 'raw' value to accomplish the change.

The GenICam standard utilizes UDP packets as its ethernet communication mechanism. While this is more efficient than TCP/IP (more common protocol) some routers and switches may not relay those messages over a wider area network. If your camera is not discovered your network may not be relaying UDP packets. You should connect the camera directly to a machine to verify that the network is not an issue when attempting to connect.

Interface



Instructions

1. Camera - Press the circular yellow arrow button to send a discovery request to all cameras on your network. Any connected cameras will appear in the Camera dropdown list after a couple seconds. Selecting one of these cameras and pressing the Start button will initiate a connection to that camera and start capturing images. Please note that the module assumes an exclusive use of the camera such that if any other applications are connected to the camera this will cause an error.
2. Port - The standard GenICam port used to communicate to cameras. Unless your cameras have been configured to work on different ports this should remain the default 3956.
3. Pixel Format - Once connected, the current Pixel Format is displayed as part of a dropdown that lists all formats supported by your camera. You can change the format by selecting a different dropdown from this list. Note that RoboRealm works with 24 bit images, one byte for red, green and blue. Formats that are a higher bit resolution (10, 12, 16) will be downsampled to 8 bits. If possible, use a maximum of 8 bits per channel to avoid unnecessary network overhead.

While the module supports many formats natively, there may be some that are unknown to the module. When this happens you will be presented with an interface to send us information about this format such that it can be incorporated into those supported formats.

4. Packet Size - The packet size option specifies how large each piece of information sent from the camera is. The larger the packet the less

overhead is incurred but the more information might be lost at any given time (based on the UDP protocol). 1500 is the largest recommended packet size as anything larger will cause many routers and switches to ignore the packet.

5. Frame Rate - This specifies in micro seconds the time per image frame. This can be changed to decrease or increase the frames per second (fps) that the camera attempts to send images at. This can help to reduce the network consumption from cameras if only a few images per second are needed.

6. Scan Area - You can use the Width, Height, Left, and Top parameters to change what portion of the sensor pixels you want to capture. This allows one to specify a global AOI (Area of Interest) which helps to reduce processing requirements and network consumption. Note that changing the Width and Height does NOT scale the image content accordingly as in many other consumer cameras.

7. Binning - Should you not need the maximum resolution that the camera can provide but still need the full field of view you can specify a binning amount. This will combine pixels in the appropriate direction (horizontal or vertical) in order to effectively reduce the image dimensions, i.e. scaling. Binning will reduce the image width and height by the binning factor. The default of 1 causes no change, the value 2 will effectively reduce the image width or height by a factor of 2, the value 3 by a factor of 3, etc.

8. Black Level, Exposure, Gain - To set specific camera settings, check the checkboxes to switch the camera into manual mode for those settings and change the values accordingly. It is not uncommon for the image to immediately become all black or white when these settings are activated. If this is the case, the manual settings may not be appropriate for the current lighting settings and can be changed to improve the image quality.

9. Advanced Tree - All camera settings are displayed in the Advanced tab. You can explore the tree by expanding the categories as appropriate to view the underlying features. When you click on one of the features you will see the description along with an interface that has the ability to change the value if allowed. In this way you can set camera features directly. Note that some features cannot be changed while the camera is operational. To change these features you will need to stop streaming before they will be accepted.

10. Higher Bit Images - RoboRealm operates on 24 bits RGB per pixel for performance reasons. When loading in images with higher bit depths RoboRealm needs to know how to process the image into a 24 bit image. There are many ways this can be accomplished:

- High - Uses the upper 8 bits of the image
- Low - Uses the lower 8 bits of the image
- Sqrt - Square root's the image pixel to the 8 bit range
- Inv Sqrt - The Sqrt function will favor darker pixels, the Inv Sqrt favors lighter
- Center Mean - Forces the high bit range to be centered at the image mean
- Below Mean - Shows only pixel below the image mean compressed into 24 bits
- Above Mean - Shows only pixel above the image mean compressed into 24 bits
- Around Mean - Similar to Center Mean but thresholds values on either side to improve contrast
- Normalize - Determines image low and high values and scales to 24 bit
- Pseudo X - Translates image intensity into a higher color range for improved visibility
- Reinhard/Drago - High Dynamic Range reduction techniques as specified in the FreeImage.dll
- Logarithm - Takes the logarithm of the image pixel and then stretches to 8 bit range

11. Process Color Channels Together - Specifies that color channels will be considered a single channel such that the relative color amounts will not change. Unselected, each color channel will be processed independently which can improve or worsen an images overall appearance.

Variables

`GENICAM_ALERT` - a variable set with the current (seconds) timestamp when the last communication failure was detected. This can be used to send alerts to staff to investigate image acquisition failures.

See Also

[Firewire Camera](#)

LeapMotion Controller

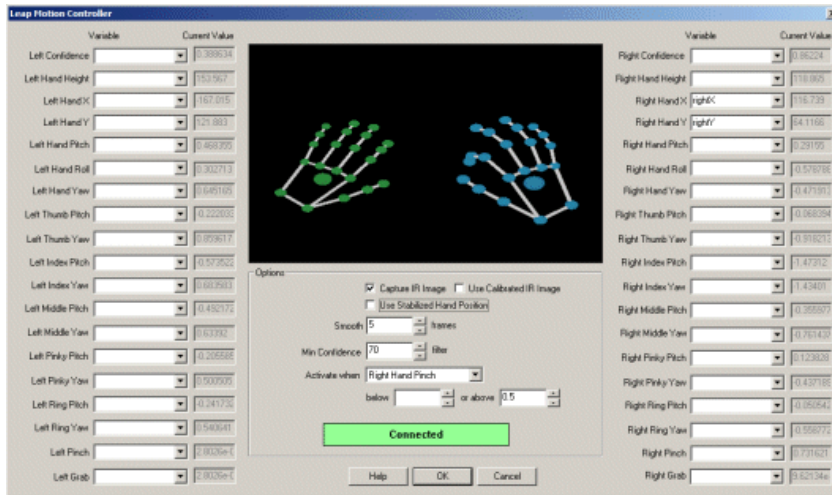


The LeapMotion Controller module provides an interface into the LeapMotion Controller device such that values generated by the device can be utilized within RoboRealm. The Controller produces a significant amount of information about hands and fingers including orientation and position. These values can be assigned to RoboRealm variables and reacted upon in the same manner as a Joystick, Mouse, etc.

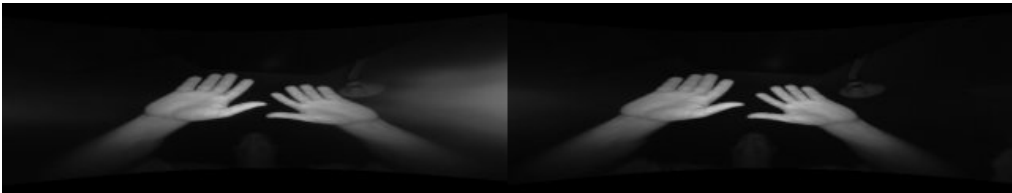
The Controller uses a stereo IR camera along with IR emitters to create an ideal hand segmentation environment. Note that in high ambient IR areas the controller will enter into a robust mode that attempts to compensate for ambient IR.

Because your hands are occupied with performing actions, the usage of position or orientation data can become unreliable as your hands enter or leave the sensor area. This can create problems when controlling sensitive devices that may receive irregular signals when your hands lose tracking. Because of this the module has several trigger actions that can start data capture.

Interface



Instructions

1. Skeleton View - Once activated the module will connect to the Controller and show the skeleton view of your hands in the module interface.
 2. Variables - Select or specify a variable that will contain the associated value. Once your hand is visible to the sensor the value associated with each value will be displayed in realtime next to the variable. Once specified the variable will contain the value and can be used to trigger actions in other modules.
 3. Capture IR Image - The device utilizes two IR sensitive cameras. The video images can be received from these two cameras and displayed in the main RoboRealm interface to be further processed. As the cameras have a significant field of view (approx 150 degrees) the images are significantly warped. Checking the "Use Calibrated IR Image" will correct for this warping such that the two images can be compared for depth.
- 
4. Stabilized Hand Position - The device has the ability to stabilize the hand position to help in finer control for certain applications. Note this is only for a hand's position and does not stabilize other numbers.
 5. Smooth - To provide a more stabilized number for all variables you can specify how many frames will be averaged together to produce a sensor value. The higher the smoothing number the more stable the result will be but will slow down the reaction time to change.
 6. Min Confidence - Specify the minimum confidence needed for the values to be recognized. This helps to eliminate values when the hands are only partially recognized due to entry and exit in the field of view or when the hands are too far back or forward.
 7. Activation - To ensure that values are not used unless desired the module provides an activation trigger that when valid will enable all values otherwise the values will not be updated. You first select what gesture will activate the values and then specify what that value should be (above or below a threshold) will trigger activation. Note, once you chose an activation gesture that gesture cannot be used in any way that will cause the gesture to become invalid as updating will stop. Thus if you use Pinching as an activation gesture and also plan to use that to specify the distance between a gripper, once the pinch falls below a certain value it will stop any updates. Therefore, be sure to use a gesture for activation that you do NOT intend to use for control.

[Click Here](#) to download an example that will cause your mouse to follow your right hand ONLY when you pinch your finger and thumb.

[Click Here](#) to download an example that will cause your speaker to beep based on the height of your right hand ONLY when you pinch your finger and thumb.

[Click Here](#) to download an example that will allow you to draw in the main RoboRealm image in green based on the position of your right hand

and red based on the position of your left hand but ONLY when you pinch your finger and thumb. If you grasp your right hand it will clear the image.

Variables

LEAP_LEFT_HAND_ACTIVE, LEAP_RIGHT_HAND_ACTIVE - indicates that a particular hand is active

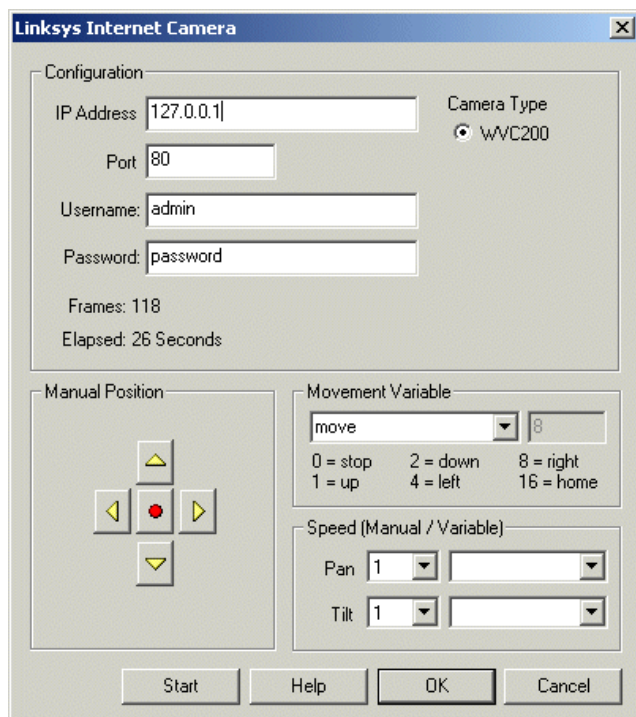
See Also

[Creative Senz3D](#)
[Microsoft Kinect](#)
[OpenNI Kinect](#)

Linksys Internet Camera

The Linksys Internet Camera module provides an interface to the Linksys camera series. This module is similar to the [HTTP Read](#) in that it reads images over the Internet but also provides an easier interface to the Linksys Camera and its pan/tilt capabilities. Note that when accessing cameras over the Internet there can be significant delays when panning or tilting the camera. Keep in mind the delay when using those buttons otherwise you will overreact the movement.

Interface



Instructions

1. IP Address - specify the IP address of the camera that you would like to connect to.
2. Port - If you are using a different port number than port 80 simply type that number into the port field.
2. Username - the username for HTTP authentication access.
3. Password - the password for HTTP authentication access.
4. Camera Type - select which camera type you are connecting to. For unlisted Linksys cameras please [contact us](#) with a Internet accessible IP address to such camera. Having access will allow us to QA the interface and enable that functionality.
5. Manual Position - use the buttons to move the camera as appropriate. The red center button will move the camera home.

6. Movement Variable - the variable that contains the commands that will move or change the camera attributes. The following values within the variable will create the following movements:

0 = stop

1 = up

2 = down

4 = left

8 = right

16 = home

7. Speed - manual or variable setting of pan and tilt speeds. or variable zooming. Either select a manual speed from the dropdown or specify a variable that will contain the required speed. Note that the valid values are from 1 to 10. Values higher or lower will be truncated appropriately. You can use the variables to specify speeds in order to quickly make coarse or fine adjustments to the position of the camera instead of successive sequential small movements.

See Also

[HTTP Read](#)

[DLink Internet Camera](#)

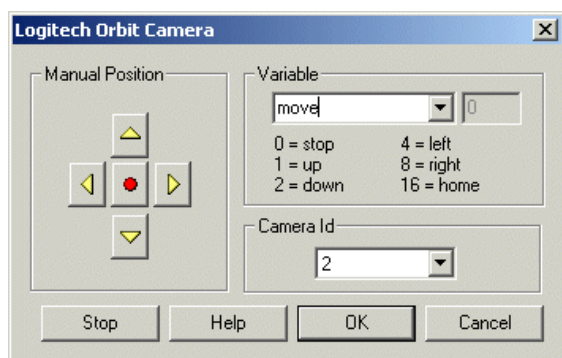
[TRENDnet Internet Camera](#)

[Foscam Internet Camera](#)

Logitech Orbit/Sphere, MP, AF

The Logitech Orbit module provides control over the Logitech Orbit Webcam's motorized pan/tilt capabilities. Similar to servo control systems the Orbit provides left and right 189 degree panning and 102 degree tilt. Using the Logitech Orbit module you can control the Orbit camera based on what it sees.

Interface



Instructions

1. Manual Position - If you have the Orbit Camera installed you should be able to use the buttons within the Logitech Orbit dialog to manually move the camera in the direction of the pressed button.

2. Variable - To automate the movements based on programmatic control select which variable holds the appropriate bitwise command. This command variable should contain the following values as appropriate:

0 = stop

1 = up

2 = down

4 = left

8 = right


16 = home

3. Camera Id - If you have more than one Logitech camera running on your system you can change the camera id which will cause the control to switch to the next Logitech camera. This also works across the different versions of the Orbit/Sphere/etc. product line.

4. Stop - To stop executing the value within the variable for a period of time press the STOP button. This will terminate camera movement until the same button (now called START) is pressed again or the program is restarted.

Example

 [Click here](#) to load a configuration to move the Orbit camera using your cursor keys.

 [Click here](#) to load a configuration that will cause the Orbit camera to track the color red.

See Also

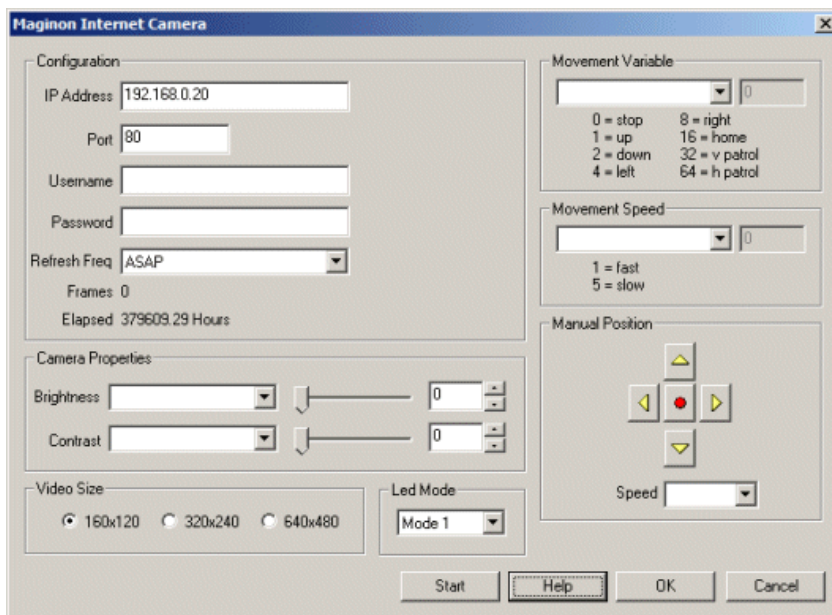
- [Creative Live Motion](#)
- [DLink Internet Camera](#)
- [Dream Cheeky Missile Launcher](#)

Maginon Internet Camera

The Maginon Internet Camera module provides an interface to the Maginon camera series. This module is similar to the [HTTP Read](#) in that it reads images over the Internet but also provides an easier interface to the pan/tilt and camera adjustment capabilities for some of the Maginon cameras. Note that when accessing cameras over the Internet there can be significant delays when panning or tilting the camera. Keep in mind the delay when using those buttons otherwise you will overreact the movement.

To allow for RoboRealm to set the pan/tilt speed you will need to set the PTZ speed to 0 within the Maginon web browser setup. If not, the speed will not increase or decrease depending on the variable setting within the module.

Interface



Instructions

1. IP Address - specify the IP address of the camera that you would like to connect to.
2. Port - If you are using a different port number than port 80 specify that port number here.

3. Username - the username needed to access the camera.
4. Password - the password needed to access the camera.
5. Refresh Freq - how quickly the system should request new images. This will allow you to reduce the Internet traffic of the streaming video if you don't need rapid updates. Default is "As Fast As Possible"
6. Manual Position - use the buttons to move the camera as appropriate. The red center button will move the camera home/center.
7. Movement Variable - the variable that contains the commands that will move or change the camera attributes. The following values within the variable will create the following movements:

0 = stop

1 = up

2 = down

4 = left

8 = right

16 = home

32 = vertical patrol

64 = horizontal patrol

8. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.

Edit the camera properties by either selecting the appropriate value using the slider, typing in the number in the available edit area, increasing/decreasing the number using the spin/up.down arrows, or select a variable that contains the desired number to be used as configuration in this module. Note that once a variable is selected the manual controls will be disabled to indicate that the property is under variable control.

- 9 Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.

See Also

[HTTP Read](#)

[DLink Internet Camera](#)

[Linksys Internet Camera](#)

[TRENDnet Internet Camera](#)

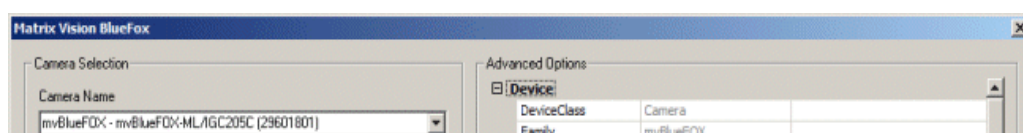
Matrix-Vision BlueFox

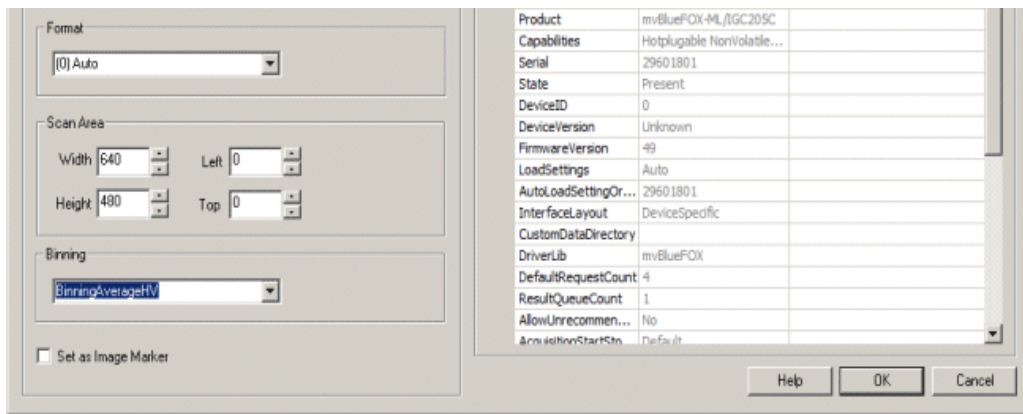


The Matrix-Vision BlueFox module provides a direct interface to the Matrix-Vision BlueFox series of USB cameras. For more information about the camera please see the [Matrix Vision](#) website.

While all Matrix Vision cameras are also accessible using DirectShow this module specifically exposes all the options of the BlueFox camera to allow for finer control via RoboRealm variables. As the DirectShow interface is a driver level connection RoboRealm cannot directly modify settings if dynamic control is needed (i.e. a variable within RoboRealm controls the exposure setting). This module exposes all of the camera settings in an interface that allows for variables to be used to control aspects of the camera AND will restore those settings on the camera when RoboRealm is relaunched. The DirectShow interface is a generic one, this module is much more specific.

Interface





Instructions

1. Camera Name - Displays a dropdown of available cameras that the PC recognizes. If your camera is not on this list, check your connection, close this Module's interface and double click on it again to reedit. The list is only repopulated when you re-edit the module.

Selecting a camera from this list should immediately start displaying that camera's image in the main RoboRealm window.

2. Format - Specifies how the camera should transmit images to the PC. In most cases '[0] Auto' should suffice but in those situations where USB bandwidth needs to be reduced once can select more compact modes.

3. AOI Area - Specifies which part of the current image is of interest. This can be used to reduce the image size which increases system performance.

4. Binning - Specifies how the current image should be reduced by binning the pixels which helps decrease sensor noise while reducing image size.

5. Advanced Options - Provides access in a tree like structure to all camera parameters. Expanding the tree can reveal addition parameters that the camera provides. Information that is grayed is read only. For information that can be edited you can click on the information to directly change that information. Once accepted this is immediately passed to the camera to change its properties. Various types of information will have different interfaces due to the type of information being edited. Once updated, the value will be saved with the current robofile (RoboRealm configuration) such that on reloading the same robofile will once again send those configuration attributes to the camera REGARDLESS of the current camera settings. This ensures that each robofile can set its own camera configuration.

It is often desirable to automate the changing of various parameters. When you click on the third column within the Advanced Options tree you can type in or select a variable that will contain the desired value to send to the camera. This variable can then be changed in other modules and sent to the camera automatically. Once you press OK and save the RoboRealm configuration (robofile) those variable relationships will also be preserved and sent to the camera again on execution. Note that for read only parameters the variable will be initialized to the associated value on connection. This allows you to display values from the camera in a more visible interface.

6. Higher Bit Images - RoboRealm operates on 24 bits RGB per pixel for performance reasons. When loading in images with higher bit depths RoboRealm needs to know how to process the image into a 24 bit image. There are many ways this can be accomplished:

- High - Uses the upper 8 bits of the image
- Low - Uses the lower 8 bits of the image
- Sqrt - Square root's the image pixel to the 8 bit range
- Inv Sqrt - The Sqrt function will favor darker pixels, the Inv Sqrt favors lighter
- Center Mean - Forces the high bit range to be centered at the image mean
- Below Mean - Shows only pixel below the image mean compressed into 24 bits
- Above Mean - Shows only pixel above the image mean compressed into 24 bits
- Around Mean - Similar to Center Mean but thresholds values on either side to improve contrast
- Normalize - Determines image low and high values and scales to 24 bit
- Pseudo X - Translates image intensity into a higher color range for improved visibility
- Logarithm - Takes the logarithm of the image pixel and then stretches to 8 bit range

11. Process Color Channels Together - Specifies that color channels will be considered a single channel such that the relative color amounts will not change. Unselected, each color channel will be processed independently which can improve or worsen an images overall appearance.

Example

[Click Here](#) to download an example that will cause the AOI to change in order to keep a 320x240 window on a bright object. This shows how to change the Advanced properties to send property changes to the camera and simulates a pan/tilt system. Once you run this robofile within RoboRealm, move the camera until you see a single bright object (like an overhead light) and then slowly move the camera to one side to see the image jumps due to the change in the X and Y of the AOI properties.

Variables

MATRIX_VISION_BLUEFOX_ERROR - Set when an error occurs within the module.

See Also

[GenICam](#)

Microsoft Kinect



The Microsoft Kinect module provides an interface to the Microsoft Kinect XBOX 360 sensor. The sensor is remarkably useful for robotics and is a great depth sensor at a reasonable price. While the Kinect was created as an addition to the XBOX it can be used just in conjunction with a PC to gain access to the depth information that it provides.

The main feature of the Kinect is that it provides a 640x480 depth map in realtime (30 fps) that indicates which objects are near versus far in the given scene. This is invaluable for use in obstacle avoidance and navigation. In addition to the depth sensor the Kinect also provides a traditional 640x480 RGB image, a 3 axis accelerometer, the ability to tilt its head and a nifty LED that you can change the colors on!

The Kinect RoboRealm module provides access to the depth map, RGB image, accelerometer information and provides the ability to control the head tilt angle all from within RoboRealm. Note that once the depth image is within RoboRealm all the other traditional filters can now be applied to process the depthmap in new ways. If you would prefer to just have the image as a webcam you can use the [RoboRealm virtual camera](#) to pipe the image into other webcam compatible application. Or if you just want to work on the image yourself you can use the various ways to [interface](#) with RoboRealm including our popular and flexible [API](#).

While the Kinect is a great addition to robotics there are some limitations. First, the depth map is only valid for objects more than 2.5 feet away from the sensing device. If an object is closer than 2 feet to the Kinect its depth will NOT be detected correctly and be presented as a zero depth. If you have a large robot you can place the sensor at the back such that the depth is generated at the border of the robot. For smaller bots you'll just have to avoid getting too close to objects otherwise they will fall too close to the sensor and come up as unknown (black) areas.

Second, the Kinect performs its magic by using an IR projector with an IR camera (a form of stereo vision) which means that our favorite bright yellow friend, the sun, will cause some issues with its functionality on a sunny day outside. If you use the Kinect outside it will NOT work in direct sunlight but will in shady areas. However, a nice aspect of having IR being projected is that it provides high resolution images in total darkness. In this case your robot can navigate around without needing any daylight!

Third, as the device is essentially a projector it draws about 2Amps of power in order to run. This may not be an issue on a large robot that has large batteries but may be an issue on mid sized robots that are power conservative.

Regardless of its shortcomings the Kinect offers a much richer data set than most advanced laser systems and at a significantly lower cost! It produces one of the best stereo depth maps that we've ever seen ... and in realtime!

Win XP Kinect PC Installation

The installation of the Kinect on a PC is not as straightforward as connecting other plug and play USB devices. Because official drivers for the Kinect on a PC have not yet been created/released we have to use an alternative driver that provides sufficient flexibility to get things working. For that reason you will need to perform the following steps to get your Kinect functioning with a Windows PC. Follow these steps BEFORE plugging it in.

1. Download - [Download](#) the open source libusb-win32 application from sourceforge and unzip it on your computer. You will need the bin version as apposed to the src version unless you plan to compile the library yourself. This system provides a generic windows driver that can be used to communicate with the Kinect if you know the right commands. The main file you need within this download is called libusb0.sys.
2. Download - [Download](#) the Kinect .inf and .cat files. These files help Windows understand what driver to use with the Kinect. You will have to also unzip this into a location that you know as you will need to specify this path later. These files were generated by using the inf-wizard.exe program within the libusb package but are included here as downloads for speed and ease of use.
3. Kinect - Plug in the Kinect into the PC. The PC should respond with a driver install dialog. Select the radio button that allows you to tell Windows where the drivers are "Install from a list or specific location (Advanced)". On the next interface using the Browse button you can specify the path where you saved the .inf and .cat files. That should make Windows happy!
4. libusb - Once the .inf files are used Windows may complain about not knowing where the libusb0.sys file is. You will have to find that file in the c:\???\libusb-win32-bin-1.2.2.0\bin\x86 folder where you downloaded and installed the libusb files. Note that the ??? might be "Program Files" but

it will depend on where you chose to install the libusb application.

5. Repeat the above steps of specifying where the .inf files are and the usblib0.sys file is for the Motor, Camera and Audio devices of the Kinect (3x). Note that the Motor driver includes the accelerometer and led drivers.
6. RoboRealm - Once this is complete you can now fire up RoboRealm and insert the Kinect module (easiest way is to select the Search tab and type in Kinect). Once this module has been inserted the default configuration should show you the depth map and the RGB color image as a single large image within the main RoboRealm GUI.

Win 7 Kinect PC Installation

1. Uninstall - You **MUST** uninstall KinectSDK for the following to work with Windows 7. Windows 7 will automatically download the KinectSDK but due to commercial licensing restrictions we **CANNOT** support those drivers. You can uninstall the drivers by going to your Device Manager, right clicking on the Kinect drivers and select uninstall. If you fail to uninstall these drivers the following drivers will not be able to receive images or depth information from the Kinect and appear as a black image.
2. Download - [Download](#) the open source libusb-win32 application from sourceforge and unzip it on your computer. You will need the bin version as apposed to the src version unless you plan to compile the library yourself. This system provides a generic windows driver that can be used to communicate with the Kinect if you know the right commands.
3. Download - [Download](#) the Kinect .inf and .cat files. These files help Windows understand what driver to use with the Kinect. You **MUST** unzip this into the c:\???\libusb-win32-bin-1.2.2.0\bin\ location of the previously download and unzipped file. The reason this location is required is that Win7 will **NOT** permit specifying where the .sys file is if it is in a different location than the .inf file. Placing those files in this folder will allow Win7 to find all the appropriate files in one location.
4. Kinect - Plug in the Kinect into the PC. The PC should respond with a FAILED driver install. Ignore this but let things settle (i.e. no more popups) before continuing.
5. hdwwiz - Click on Windows button and in the Run dialog box type in hdwwiz and press OK. This will run the hardware wizard. Click Next to get started.
6. hdwwiz - Select "Install the hardware that I manually select from a list (Advanced)". Click Next.
7. hdwwiz - Keep the "Show All Devices" selection and press Next.
8. hdwwiz - Click on the Have Disk button and select that libusb bin folder where you also unzipped the .inf (Driver) files into. Click Next.
9. hdwwiz - Select one of the 3 items (Camera, Audio, Motor) to install. Click next and allow driver to be installed when security message pops up.
10. hdwwiz - Repeat running hdwwiz for the next two items such that all 3 devices have been installed.
11. RoboRealm - Once this is complete you can now fire up RoboRealm and insert the Kinect module (easiest way is to select the Search tab and type in Kinect). Once this module has been inserted the default configuration should show you the depth map and the RGB color image as a single large image within the main RoboRealm GUI.
12. Problems - If this does not work try first rebooting and re-run RoboRealm.

Vista Kinect PC Installation

1. Download - [Download](#) the open source libusb-win32 application from sourceforge and unzip it on your computer. You will need the bin version as apposed to the src version unless you plan to compile the library yourself. This system provides a generic windows driver that can be used to communicate with the Kinect if you know the right commands.
2. Download - [Download](#) the Kinect .inf and .cat files. These files help Windows understand what driver to use with the Kinect. You **MUST** unzip this into the c:\???\libusb-win32-bin-1.2.2.0\bin\ location of the previously download and unzipped file. The reason this location is required is that Vista will **NOT** permit specifying where the .sys file is if it is in a different location than the .inf file. Placing those files in this folder will allow Vista to find all the appropriate files in one location.
4. Kinect - Plug in the Kinect and wait for the cannot locate driver software popup.
5. Select "Locate and install driver software". Click next.
6. Press 'Continue/Accept' when security confirmation request appears.
7. Select 'I don't have the disk, show other options' when Vista complains about not finding driver.
8. Click on 'Browse my computer for driver software (Advanced)'
9. Specify the libusb bin folder where you also unzipped the .inf (Driver) files into. Click Next.

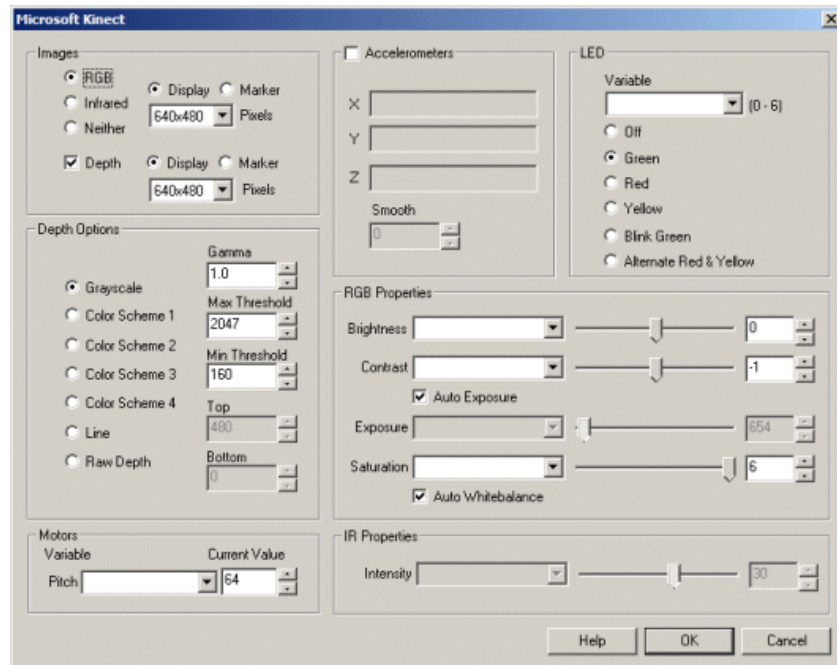
10. Select "Install this driver software anyway." when Vista complains about the driver not being signed.

11. Repeat the process for the next two devices.

12. RoboRealm - Once this is complete you can now fire up RoboRealm and insert the Kinect module (easiest way is to select the Search tab and type in Kinect). Once this module has been inserted the default configuration should show you the depth map and the RGB color image as a single large image within the main RoboRealm GUI.

13. Problems - If this does not work try first rebooting and re-run RoboRealm

Interface



Instructions

1. Images - select which image you'd like to focus on. This will display that image within the main RoboRealm GUI window. By selecting the different radio buttons you can choose between the depth image (the image that shows the distance to an object based on pixel intensity/color) and/or the RGB image (the typical colored image you see from webcams) and/or the IR image. Because you may be using the images in different ways you can choose to have images saved in memory as [marker](#) images or be shown in the main RoboRealm interface. Note that if you are not using both images you can choose the Show Depth or Show RGB only as they help to reduce USB traffic in case you are working with a slower USB bus. Also note that the IR and RGB image CANNOT be used at the same time.

2. Motor - The Kinect has the ability to tilt or bow its head. You can select the current value from 0 to 128 (with 64 being about level) to reposition its head. To automate this movement you can instead select a variable that contains or will contain a numeric value from 0 to 128 which will set the tilt angle. Note that once you select or type in a variable the manual number will be disabled to indicate that the tilt angle is now under programmatic control.

3. Depth Options - The depth image that the Kinect produces is a 11 bit (0 - 2047 value) number that can be displayed in several ways. The most common depth map display uses intensity (black to white) to indicate the distance to any particular pixel within the image. This can be useful for successive processing (for example [threshold](#)) but may not best indicate the different depth levels. Thus you can use the different color schemes to provide more insight into what depth levels are present in the image.

Grayscale - reduces the depth information from 12 bits to 8 for each RGB pixel

Color Scheme X - uses the depth value as an index into a color palette. Each palette uses a different color transition.

Line - produces a line that indicates the closest depth to an object. This is similar to what a SICK or LIDAR laser would produce. Note this option enables the Top and Bottom settings which allow you to crop the depth map to a specific height. Reducing the height allows objects that are above or below the needed detection level to be ignored when creating the distance line.

Raw Depth - the 12 bit number is encoded with 8 bits in the red channel and the remaining 4 in the green channel. This allows you to access the full depth resolution in external applications by decoding the R and G channel. (Note the B channel is set to 0).

4. Gamma - While the Kinect can produce up to 2048 depth levels we've not found it to go much beyond 1600. Because of this not all the colors will be displayed or used. You may also want to focus more on closer depths and lose some resolution at more distant depths (since gray images are 0-255 you lose 3 bits). Using the gamma correction can help to bunch up values to better highlight the different depth values.

5. **Threshold** - You may not want objects that are far away from the Kinect to appear in the depth map. Using the threshold value you can remove far away objects and also compress the color space of the resulting image to better show depth differences. Using the threshold you can eliminate all but nearby objects that can then be further processed by other modules.
6. **Accelerometers** - The 3 axis accelerometer provides position information for the Kinect. The X and Y values indicate roll and tilt with the Z axis being a check to determine if the device is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (Z axis).
7. **Smooth** - Increasing the smooth value will smooth the unwanted jitter due to noise that the accelerometer numbers will typically display. By decreasing the smooth value you make the device much more responsive to movement, increasing the value will delay the sharpness of the movement but also make it much more stable and change gradually.
8. **LED** - Use the radio buttons to select what state you want the led to be in. To programmatically control that state select a variable that contains or will contain a value from 0 to 6 that will cause the led to change state.
9. **Brightness** - refers to an intensity or luminosity scale of the RGB image that ranges from totally black to totally white and has no effect on color values.
10. **Contrast** - refers to how far pixel values can deviate from gray in the RGB image. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.
11. **Auto Exposure** - Allow the Kinect to adjust exposure time for the RGB image.
12. **Exposure** - refers to how long your camera takes to record an RGB image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.
13. **Saturation** - refers to the intensity of the RGB colors, i.e. how red a red color is. By decreasing the saturation of an image you remove color and produce a monochrome grayscale picture that represents only darkness and brightness, or luminance. Increasing saturation in an image produces artificially intense colors.
14. **Auto WhiteBalance** - Allows auto adjustment of the RGB image colors such that white pixel are adjusted to a white color.
15. **IR Properties** - Specifies the intensity the projector that is transmitting the IR dots. Note that the 1280x1024 mode has the highest IR projector value whilst the smaller modes allow for some adjustment.

Example

Microsoft Kinect Depth Map with RGB image



Variables

KINECT_DEPTH - when saved as a marker this variable contains the depth image

KINECT_RGB - when saved as a marker this variable contains the RGB image

KINECT_X - the X axis accelerometer value

KINECT_Y - the Y axis accelerometer value

KINECT_Z - the Z axis accelerometer value

See Also



OpenNI

The OpenNI module provides an interface to the Microsoft Kinect and Asus Xtion sensors. These sensors are remarkably useful for robotics and are great depth sensors at a reasonable price.

The main feature of both sensors are that they provides a 640x480 depth map in realtime (30 fps) that indicates which objects are near versus far in the given scene. While the Asus does not provide an RGB image it does permit for a smaller 320x240 depth map that helps not to overload your USB bus. It is also much smaller than the Kinect and does not require an external power supply. In addition to the depth sensor the Kinect also provides a traditional 640x480 RGB image, a 3 axis accelerometer, the ability to tilt its head and a nifty LED that you can change the colors on!

The OpenNI RoboRealm module provides access to the depth map on either device. Note that once the depth image is within RoboRealm all the other traditional filters can now be applied to process the depthmap in new ways. If you would prefer to just have the image as a webcam you can use the [RoboRealm virtual camera](#) to pipe the image into other webcam compatible application. Or if you just want to work on the image yourself you can use the various ways to [interface](#) with RoboRealm including our popular and flexible [API](#).

While these devices are a great addition to robotics there are some limitations. First, the depth map is only valid for objects more than 2.5 feet away from the sensing device. If an object is closer than 2 feet to the sensor its depth will NOT be detected correctly and be presented as a zero depth. If you have a large robot you can place the sensor at the back such that the depth is generated at the border of the robot. For smaller bots you'll just have to avoid getting too close to objects otherwise they will fall too close to the sensor and come up as unknown (black) areas.

Second, these devices perform their magic by using an IR projector with an IR camera (a form of stereo vision) which means that our favorite bright yellow friend, the sun, will cause issues with its functionality on a sunny day outside. If you use the sensors outside it will NOT work in direct sunlight but will in shady areas. However, a nice aspect of having IR being projected is that it provides high resolution images in total darkness. In this case your robot can navigate around without needing any daylight!

Third, the Kinect device draws about 2 Amps of power in order to run. This may not be an issue on a large robot that has large batteries but may be an issue on mid sized robots that are power conservative. Note that the Xtion does has much lower power needs.

Kinect or Xtion Installation

The installation of the devices on a PC is not as straightforward as connecting other plug and play USB devices. Follow these steps BEFORE plugging it in.

1. Uninstall - If you do not have the OpenNI drivers already installed please remove any Microsoft, OpenKinect or Libusb drivers before proceeding. Failure to do so will cause your device not to function correctly.
2. Download and Install - [OpenNI Package](#) and run the installer. Please note that even if you have a 64bit machine you MUST install the 32bit drivers.

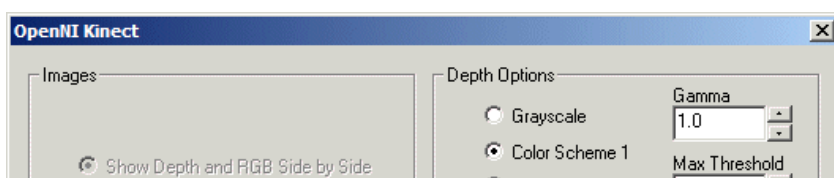
Kinect Cont Installation

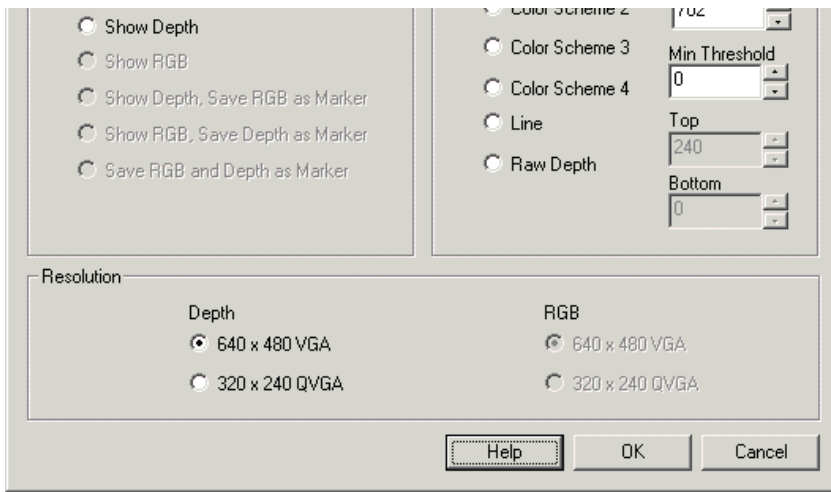
3. Download and Install - [Download](#) the SensorKinect file and install. These files provide the connection from the Kinect to the OpenNI system.

Kinect or Xtion Cont Installation

4. Plug the Kinect or Xtion into the PC. The PC should respond by installing the needed drivers.
5. RoboRealm - Once this is complete you can now fire up RoboRealm and insert the OpenNI module (easiest way is to select the Search tab and type in OpenNI). Once this module has been inserted the default configuration should show you the depth map and the RGB color image as a single large image within the main RoboRealm GUI.

Interface





Instructions

1. Images - select which image you'd like to focus on. This will display that image within the main RoboRealm GUI window. By selecting the different radio buttons you can choose between the depth image (the image that shows the distance to an object based on pixel intensity/color) and/or the RGB image (the typical colored image you see from webcams). Because you may be using the images in different ways you can choose to have images saved in memory as [marker](#) images or be shown in the main RoboRealm interface. Note that if you are not using both images you can choose the Show Depth or Show RGB only as they help to reduce USB traffic in case you are working with a slower USB bus.

2. Depth Options - The depth image that the devices produce is a 11 bit (0 - 2047 value) number that can be displayed in several ways. The most common depth map display uses intensity (black to white) to indicate the distance to any particular pixel within the image. This can be useful for successive processing (for example [threshold](#)) but may not best indicate the different depth levels. Thus you can use the different color schemes to provide more insight into what depth levels are present in the image.

Grayscale - reduces the depth information from 12 bits to 8 for each RGB pixel

Color Scheme X - uses the depth value as an index into a color palette. Each palette uses a different color transition.

Line - produces a line that indicates the closest depth to an object. This is similar to what a SICK or LIDAR laser would produce. Note this option enables the Top and Bottom settings which allow you to crop the depth map to a specific height. Reducing the height allows objects that are above or below the needed detection level to be ignored when creating the distance line.

Raw Depth - the 12 bit number is encoded with 8 bits in the red channel and the remaining 4 in the green channel. This allows you to access the full depth resolution in external applications by decoding the R and G channel. (Note the B channel is set to 0).

4. Gamma - While the devices can produce up to 2048 depth levels we've not found it to go much beyond 1600. Because of this not all the colors will be displayed or used. You may also want to focus more on closer depths and lose some resolution at more distant depths (since gray images are 0-255 you lose 3 bits). Using the gamma correction can help to bunch up values to better highlight the different depth values.

5. Threshold - You may not want objects that are far away from the devices to appear in the depth map. Using the threshold value you can remove far away objects and also compress the color space of the resulting image to better show depth differences. Using the threshold you can eliminate all but nearby objects that can then be further processed by other modules.

Example

OpenNI Depth Map from Xtion



Variables

OPENNI_DEPTH - when saved as a marker this variable contains the depth image

OPENNI_RGB - when saved as a marker this variable contains the RGB image

See Also

[Stereo Depth](#)

[Creative Senz3D](#)

[Microsoft Kinect](#)

[LeapMotion Controller](#)

[Hokuyo Laser](#)

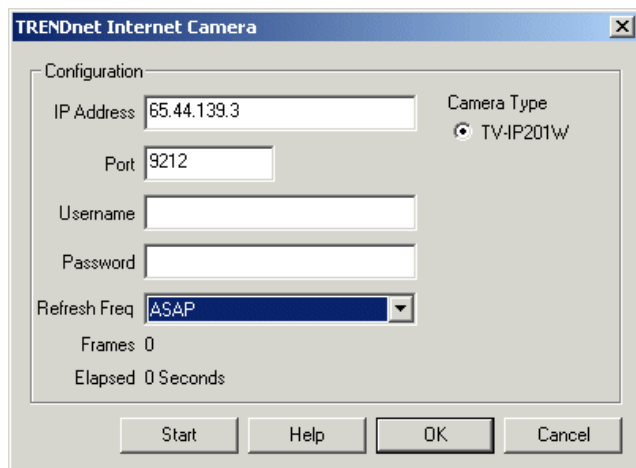
[Visible Laser Line](#)

TRENDnet Internet Camera



The TRENDnet Internet Camera module provides access to TRENDnet TV-IP201 Internet cameras such that you can capture the image from the Internet into RoboRealm for further processing.

Interface



Instructions

1. IP Address - Enter the IP address of the camera you wish to process
2. Port - Enter the port number used to access the camera
3. Username - If you have specified authentication for your camera enter the username here
4. Password - If you have specified authentication for your camera enter the password here
5. Refresh Freq. - If you just need a frame every once in a while specify that frequency here. This will allow you to reduce the Internet traffic of the streaming video if you don't need rapid updates. Default is "As Fast As Possible"
6. Camera Type - Select the camera model you are connecting to. Note that the IP422 is the only camera that supports pan/tilt capabilities.
7. Manual Position - use the buttons to move the camera as appropriate. The red center button will move the camera home.
8. Movement Variable - the variable that contains the commands that will move or change the camera attributes. The following values within the variable will create the following movements:

0 = stop

1 = pan

1 = up

2 = down

4 = left

8 = right

16 = home

9. Preset - select the variable that contains the preset number to go to or just click on one of the buttons manually to change the camera position. Note that you can use the preset locations to move the camera to known locations quicker than using the movement variable above which performs a relative incremental move.

10. Start Button - To actually start streaming the video press the START button

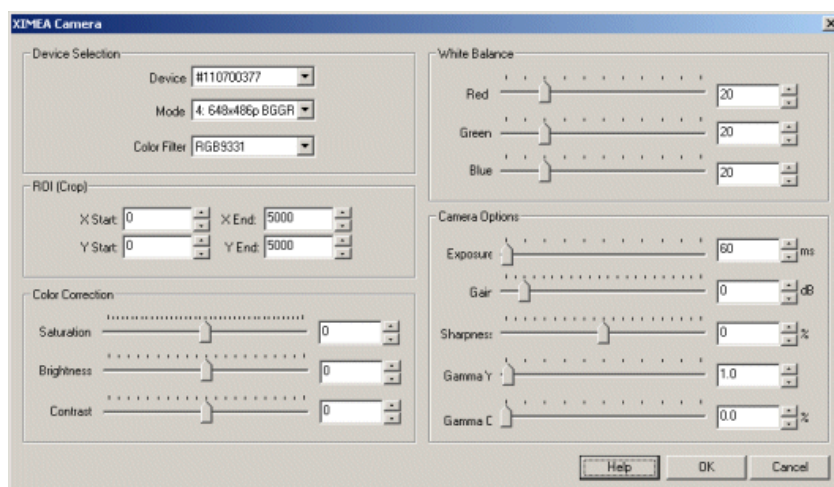
XIMEA Camera



The XIMEA Camera module provides access to the XIMEA line of cameras that use the [M3API](#) protocol. XIMEA produces a range of high quality cameras for use in industrial machine vision applications. One of their latest cameras, the CURRERA RL13, is innovating the way smart cameras are created by combining high end optics with an embedded PC capable of running Microsoft Windows all in a small compact package. This environment makes a great host for the RoboRealm application when industrial solutions are desired.

The module will connect RoboRealm via the M3API to the connected camera. You must have downloaded the m3Api.dll and associated .TM files and placed them in the RoboRealm folder (this is typically c:\program files\RoboRealm). If the DLL is missing RoboRealm will alert you to that situation. If the .TM files are missing, upon executing the XIMEA module the RoboRealm application will abruptly exit without warning.

Interface



Instructions

1. Device Selection - If you have connected your XIMEA camera to your computer you should see one or more devices listed in the Device dropdown. You can select the appropriate camera to connect to. Once selected the Mode list will update and provide you with various image sizes from which to select from. Finally, you can choose the color format that you wish to use.
2. Mode -
3. Color Filters - This allows you to select between gray and color modes.
4. ROI - Should you not need the entire image you can crop the image to a specified size. Note that the coordinates are specified as origin x,y and destination x,y.
5. LED settings - Specifies how the corresponding LED should be triggered.

LED_LINK_HB - set led to blink if link is ok (led 0), heartbeat (led 1)

LED_TRIGGER - set led to blink if trigger detected

LED_EXT_EVENT - set led to blink if external signal detected

LED_STREAMING - set led to blink if data streaming

LED_INTEGRATION - set led to blink if sensor integration time

LED_INTEGRATION - set led to blink if sensor integration time

LED_ISACQ - set led to blink if device busy/not busy

LED_NIC - set led to blink if link is ok

LED_ZERO - set led to zero

LED_ONE - set led to one

6. GPIO - Specifies how the input and output lines should be handled.

SET_TRIGGER - set gpi to trigger input SET_EXT_EVENT - set gpi to input external event SET_INPUT_OFF - set gpi off

For GPO pins you can select a variable that contains or will contain a value to send to the GPO pins. Note that you can either select an existing variable or type in a new variable. The variable should contain one of the following string values or the ordered numerical equivalent (0-5)

SET_ZERO - set gpo to zero SET_ONE - set gpo to one SET_STROBE_OUT - set gpo to output "strobe out" signal (device busy)

SET_STROBE_OUT_INV - set gpo to output inverted "strobe out" (device busy) SET_STROBE_OUT_INT - set gpo to output "strobe out" signal (integration) SET_STROBE_OUT_INT_INV - set gpo to output inverted "strobe out" (integration)

7. White Balance - If your acquired image's white areas appear to be tinted in a certain color you can use the White Balance values to move the image's white areas away from a colored tint.

8. Camera Options - Use the selections to change the Camera parameters on how the image is acquired. These parameters will change the behavior of the cameras capture parameters and thus are more powerful in attaining a better image than performing the same task in software. Note that GammaY and GammaC are another name for Brightness and Saturation values but are instead sent to the camera hardware to operate on a pixel acquisition rather than performing the task in software.

Note that changing certain parameters like the Exposure will increase the length of time an image requires to be created and thus will slow down the frame rate (fps). Should you require a fast exposure attempt to use the Gain or Brightness controls to provide a satisfactory image while maintaining a faster fps.

9. Exposure - refers to how long your camera takes to record an image. In a well-lit scene, exposure times can be very short because plenty of light is available stimulate the CCD pixels with enough energy to record an image. At nighttime, exposure time will increase dramatically due to the near absence of light. A quick exposure time will also reduce motion blur, whereas a slow will introduce more blur if the object is moving.

10. Gain - refers to the amplitude or magnification of the incoming video. Higher gain levels result in greater levels of brightness and contrast. Lower levels of gain will darken the image, and reduce the contrast. Essentially, gain modification affects the sensitivity to light of the CCD sensors. This concept is analogous to the ISO or ASA ratings of silver halide films in digital cameras.

11. Sharpness - refers to the amount of edge definition or crispness in an image. The sharpness control can be used to smooth out rough edges.

12. Gamma Y - refers to the adjustment of intensity values of a pixel to affect how bright the image appears.

13. Gamma C - refers to the adjustment of red, green, or blue values of a pixel to affect how bright the image appears.

14. Color Correction - Once the image is acquired you can alter the image in software to produce a more pleasing result. Use the sliders and/or spin boxes to change the saturation (color), brightness (intensity) or contrast (spread) of the image. Note that these alter pixel values AFTER being acquired from the camera.

15. Saturation - refers to the intensity of the colors, i.e. how red a red color is. By decreasing the saturation of an image you remove color and produce a monochrome grayscale picture that represents only darkness and brightness, or luminance. Increasing saturation in an image produces artificially intense colors.

16. Brightness - refers to an intensity or luminosity scale that ranges from totally black to totally white and has no effect on color values.

17. Contrast - refers to how far pixel values can deviate from gray. The higher the contrast the more black and white and image appears. The lower the contrast the more gray an image appears.

BasicMicro_RoboClaw



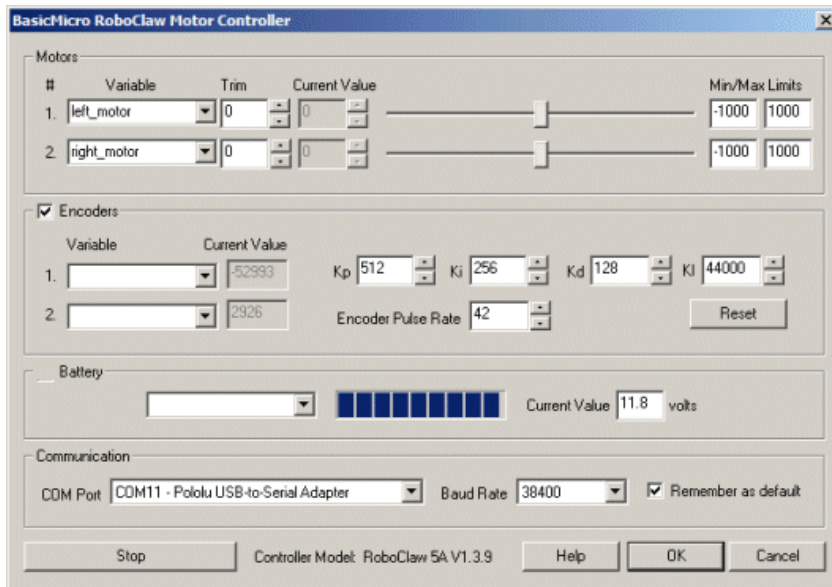
The BasicMicro_RoboClaw provides an interface from RoboRealm to the [RoboClaw](#) distributed by BasicMicro. The RoboClaw accepts multiple types of inputs (including PWM) but the module is created to interface via a serial interface in order to take advantage of the quadrature encoder inputs that the board supports. Having PID loops calculated and reacted to onboard the board allows for quick reaction time and offloads this time consuming function from other computing devices.

Note that the serial communication is TTL levels so you will need an additional device such as the [Pololu USB-to-Serial Adapter](#) to connect to SW1 & SW2 on the board to a USB cable.

This module expects to communicate with the RoboClaw using Packet Serial Mode. See the documentation for your specific device as to how to

set it to receive that form of communication. The assumed address/ID is 128.

Interface



Instructions

1. COM Port - Select the COM port that has registered your USB to serial adaptor. You will only see COM ports that are recognized by your computer. Note that direct serial connections normally use com ports lower than 4 whilst virtual COM ports created by USB connections are much higher. You will know when you have the right COM port by the population of the Controller Model text seen next to the stop button.

2. Baud Rate - The connection speed. No reason to change from 38400 unless your serial adaptor is slow or over a wireless link.

3. Motors - After specifying the COM Port you should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the "Current Value" text box in the Motors area. If the motors do not move (or whine) check your COM Port setting and/or board connections.

4. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the RoboClaw. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Extension based program or other modules. See [Variable Control](#) for more information on how to programatically move the robot.

6. Current Value - To manually set the motor position type in the appropriate number (-1000 to 1000, 0 is the default neutral/stop) into the text area or use the slider to adjust the value. The motor speed will be updated as appropriate.

7. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the robot does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case you are testing your robot in a precarious position.

8. Encoders - To enable the encoder area click on the group checkbox. This will enable the encoder readings and display the number of encoder ticks that the robot has traveled. This also switches the motor commands to now respect the encoder feedback. You may notice a speed change by enabling this checkbox.

If you enable this area and then move the robot using the Motor sliders you will see the encoder values change depending on how fast the robot is moving. Specifying variables in the dropdown will cause those variables to be set to the current values read from the robot. Also note the "Reset" button that can be used to reset the encoder values on the robot to zero.

9. Kp, Ki, Kd - Specifies the Proportional, Integral, Derivative values that the RoboClaw uses to maintain speed. The board uses these values in a formula to adjust the actual motor values to be different than that of the desired motor values to ensure that the motors are actually moving in such a way that best represents the desired speed. For example, if you are traveling along a flat surface with motor speeds set at 30 (slowly forward) and suddenly encounter a hill this slow speed will not suffice to get the robot over the hill. With the PID loop enabled the lack of actual motion will be sensed by the reduced encoder count and cause the actual motor values to increase such that the encoders continue to report the same count. Thus the robot will not slow down when encountering a hill. See [Wikipedia](#) for more information about PID loops.

10. Encoder Pulse Rate - The RoboClaw will drive the motors based on the pulse rate received from the encoders. As you may use different encoders this rate may differ. If you notice that the wheel decrease in speed when the encoders are enabled increase this number till its about the same rate as the non-encoder speed when the robot is raised on blocks (i.e. does not experience any weight). When the speed is about the same with or without encoders you can be sure that the full speed range is being used and your encoder pulse rate is set correctly. The default of 40 is set to the US Digital encoders.

11. Battery - once the RoboClaw is communicating with RoboRealm you can turn on the battery monitor to test the connection. Do this by simply selecting the Battery area checkbox in the interface. The power level will be shown in blue bars with the number of volts in the "Current Value" textbox. If you want to use the volts within other areas of RoboRealm select a variable that the module will write the number of volts to. This variable will then be updated each time the RoboClaw module is run with the current voltage of the robot. Note that this value is multiplied by 10 to keep things in integer format. (You can divide by 10 to get the actual voltage).

12. Stop - press the stop button to quickly stop the motors. The motors controls will then be disabled (in case variables are selected) and will only restart when you click the same button again (now renamed to "Start").

See Also

[Phidgets Motor Controller](#)
[Dimension Engineering Sabertooth](#)

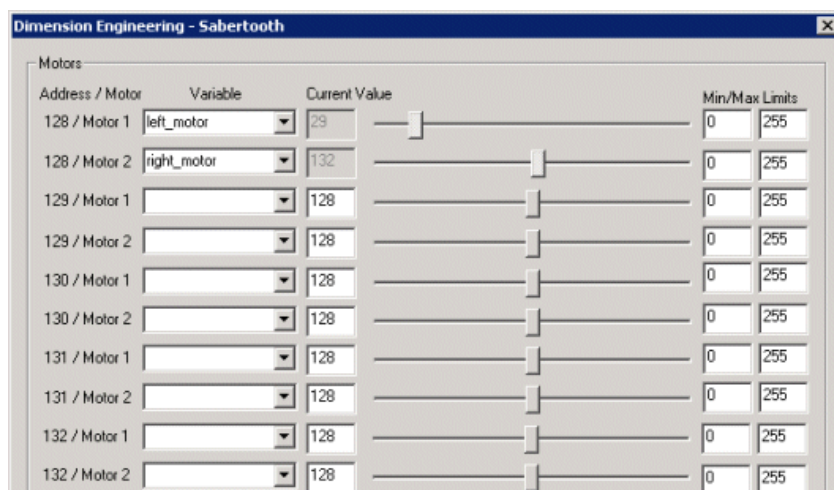
Dimension Engineering - Sabertooth

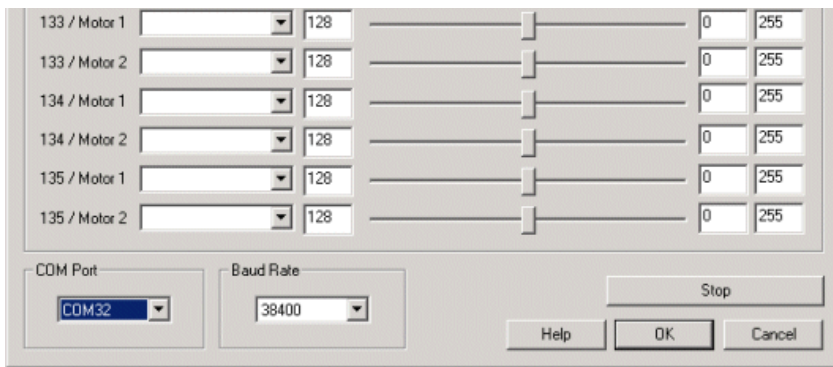
The Sabertooth motor driver allows you to interface RoboRealm to DC brushed motors. The Sabertooth is available from [Dimension Engineering](#) and can support up to two DC motors at 5A per board. To ensure reliable communication RoboRealm utilizes the packetized serial mode communication with the Sabertooth.

You can control up to 8 Sabertooth boards from a PC totaling 16 DC motors by specifying the appropriate address on the Sabertooth board using the provided DIP switches. See the Sabertooth documentation for more information as to how to set those DIP switches.



Interface





Instructions

1. Select the appropriate COM port that the USB or Serial cable is attached to.
2. Select the highest possible baud rate (typically 38400)
3. Move the sliders next to the correct motor address (128 to 135) and number (left/right side motor) to test the movement for that motor.
4. Select or type in a variable that will contain the power value that should be sent to the motor controller.

Installation

To connect the Sabertooth to a PC you will need a level converter circuit to reduce the serial COM/USB power levels that ranges from +12V to -12V to the 5V TTL levels that the Sabertooth requires. You can construct your own circuit using a MAX232 chip or use prebuilt solutions such as [USB to TTL Cable](#).

To setup the USB connection you need to connect the black wire (ground or Blk-) from the USB cable to the 0V connector on the Sabertooth and the green wire (ttl in or TX) to the S1 connector. Note: do **NOT** connect the red wire to the 5V connector as that is used to power external circuits as apposed to receiving power from the USB cable. Also note that the serial cable may use different colors so check the cable documentation before attaching.

For more instructions please see [Controlling a Sabertooth with Roborealm](#)

See Also

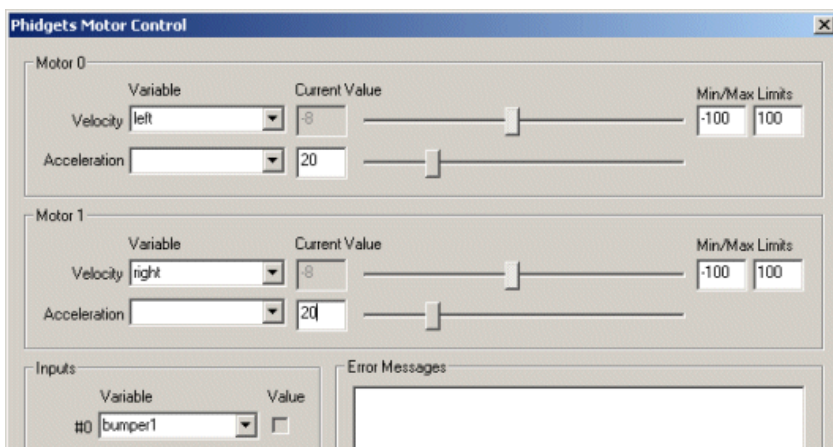
[Phidgets Motor Controller](#)
[BasicMicro RoboClaw](#)

Phidget Motor Controller

The Phidget Motor Controller module allows you to interface RoboRealm to the Phidgets Motor Controllers (LV and HC) made by [Phidgets](#). Both motor controllers are USB based and can control two brushed DC motors independently for direction, velocity and acceleration. The LV (low voltage) version also has 4 digital inputs.



Interface





Instructions

1. Motor 0,1 - On inserting the module you will be able to control the velocity of the motor by either entering in a number from -100 to 100 in the text area or by dragging the scroll bar to the left or right. The motor velocity will be updated as appropriate.
2. Acceleration 0,1 - You can also set the acceleration by sliding the slider or entering a number in the provided text area.
3. Variable - To control the velocity or acceleration automatically select an appropriate variable that contains or will contain the value that will be sent to the controller board. This is used to automatically change the values based on your VBScript (using the SetVariable function) or Plugin based program. You can also use the min/max limits to ensure that even if the variables specify too large or low values (due to programming errors) that the board does not actually attempt to set the values above or below the specified limits. This can be used as an additional precaution in case your motors cannot exceed certain thresholds.
4. Press STOP if you need to quickly zero the velocity and return to the neutral position.

See Also

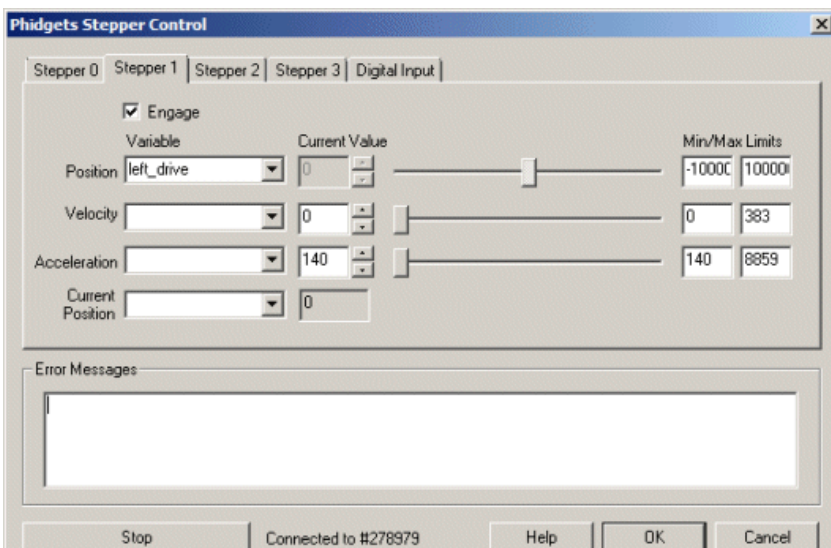
- [BasicMicro RoboClaw](#)
- [Dimension Engineering Sabertooth](#)
- [Phidgets Encoder](#)
- [Phidgets Servo](#)

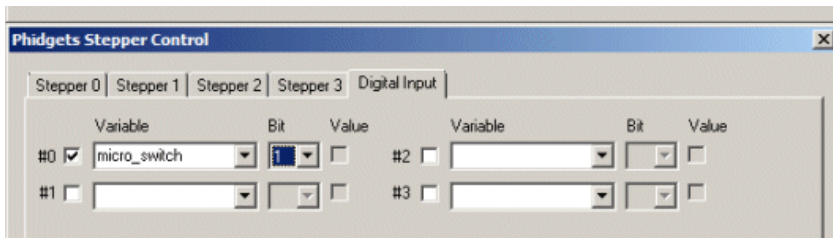
Phidgets Stepper

The Phidgets Stepper module allows you to interface RoboRealm to the Phidgets Stepper Controller ([1062](#) and [1063](#)) made by [Phidgets](#). Both Stepper controllers are USB based and can control stepper motors independently for position, velocity and acceleration. The 1063 version also has 4 digital inputs.



Interface





Instructions

1. Stepper 0,1,2,3 - Click on the tab that represents the stepper you wish to move.
2. Engage - Click on the Engage checkbox to activate that stepper. This will cause the stepper board to provide power to the stepper.
3. Position - You will be able to control the position of the stepper by either entering in a number in the text area or by dragging the scroll bar to the left or right. The stepper position will be updated as appropriate. Note that once the stepper position is reached the stepper motor will stop.
To automatically control the position select an appropriate variable that contains or will contain the value that will be sent to the stepper board. This is used to automatically change the values based on your VBScript (using the SetVariable function) or Plugin based program. You can also use the min/max limits to ensure that even if the variables specify too large or low values (due to programming errors) that the board does not actually attempt to set the values above or below the specified limits. This can be used as an additional precaution in case your motors cannot exceed certain thresholds.
4. Velocity - Specify how quickly the stepper will move to reach its goal. Again, you can do this by either entering a value into the Current Value, using the slider or by specifying a variable that contains the velocity value.
5. Acceleration - Specify how quickly the stepper will accelerate to the specified velocity. Again, you can do this by either entering a value into the Current Value, using the slider or by specifying a variable that contains the velocity value.
6. Current Position - Indicates the current position of the stepper. This value will increase or decrease towards the specified position at the specified velocity. If you specify or select a variable that variable will contain the current stepper position value.
7. Digital Input - The 1063 supports digital inputs. Enable the input by selecting the checkbox. Type in a variable that will contain a value when the input pin is grounded. If you want to OR the value into the variable specify which bit should indicate the input value. When the pin is grounded, the Value checkbox will indicate this condition.
8. Stop - Press STOP if you need to stop the stepper from moving.

RoboteQ Motor Controller



The RoboteQ Motor Controller module provides an interface from RoboRealm to the [RoboteQ Motor Controllers](#). The RoboteQ line of Motor Controllers feature a high-performance microcomputer and quadrature encoder inputs to perform advanced motion control algorithms. Additionally, digital out, digital in and analog in are provided on most boards.

This module provides an interface to configure the appropriate communication port and speed to the controller and allows you to view and modify settings by using the GUI based sliders and text editing areas. To automatically specify the values you should select variables from the dropdown menus that will contain the values that will be sent to the controller. Note that once variables have been selected manual controls are disabled.

The module is not meant to provide as comprehensive configuration that the Roborun+ application distributed with the RoboteQ controllers does, but, instead, provides an interface to those parameters that change within normal execution. To change a controller's default configuration, exit RoboRealm, use Roborun+ to make the needed changes, exit Roborun and then start RoboRealm.

Please note that depending on the type of controller you are using you may have more or less functionality that is represented in this module. Pins are often shared amongst different capabilities so take care not to overlap pins. For example, Analog 1 is the same pin as Pulse 5 and Digital Input 5. Thus if you use Analog 1 you will not be able to use Pulse 5 or Digital Input 5.

Also note that prior to using this module your pin needs to be configured correctly in RoboRun. For example, as Pin 5 can be used as a digital input and analog input, you need to use RoboRun to configure the pin to the appropriate mode. If you configure the Pin as a digital input and attempt to use it as an analog input, the value will always be zero.

Interface



#	Variable	Trim	Current Value	Min/Max Limits
1.	<input type="text"/>	0	-53	-1000 1000
2.	<input type="text"/>	0	0	-1000 1000

Battery Current Value volts

Communication
COM Port Baud Rate Remember as default

Stop Controller Model: N/A Help OK Cancel

RoboteQ Motor Controller

Motors | Analog Input | Digital Output | Digital Input | Pulse Input

Channel	Variable	Value
1.	<input type="text"/>	0
2.	<input type="text"/>	0
3.	<input type="text"/>	0
4.	<input type="text"/>	0

RoboteQ Motor Controller

Motors | Analog Input | Digital Output | Digital Input | Pulse Input

Channel	Variable	Bit	Channel	Variable	Bit
1.	<input type="text"/>	<input type="text"/>	6.	<input type="text"/>	<input type="text"/>
2.	<input type="text"/>	<input type="text"/>	7.	<input type="text"/>	<input type="text"/>
3.	<input type="text"/>	<input type="text"/>	8.	<input type="text"/>	<input type="text"/>
4.	<input type="text"/>	<input type="text"/>			
5.	<input type="text"/>	<input type="text"/>			

RoboteQ Motor Controller

Motors | Analog Input | Digital Output | Digital Input | Pulse Input

Channel	Variable	Bit	Channel	Variable	Bit
1.	<input type="text"/>	<input type="text"/>	6.	<input type="text"/>	<input type="text"/>
2.	<input type="text"/>	<input type="text"/>	7.	<input type="text"/>	<input type="text"/>
3.	<input type="text"/>	<input type="text"/>	8.	<input type="text"/>	<input type="text"/>
4.	<input type="text"/>	<input type="text"/>	9.	<input type="text"/>	<input type="text"/>
5.	<input type="text"/>	<input type="text"/>	10.	<input type="text"/>	<input type="text"/>

RoboteQ Motor Controller

Motors | Analog Input | Digital Output | Digital Input | Pulse Input

Channel	Variable	Value
1.	<input type="text"/>	0
2.	<input type="text"/>	0
3.	<input type="text"/>	0
4.	<input type="text"/>	0
5.	<input type="text"/>	0
6.	<input type="text"/>	0

Instructions

1. COM Port - Select the appropriate COM Port that the Controller is connected to. You will only see COM ports that are recognized by your computer.
2. Baud Rate - Select the appropriate baud rate. 115200 is the default speed used.

3. Battery - once the Controller is communicating with RoboRealm you can turn on the battery monitor to test the connection. Do this by simply selecting the Battery area checkbox in the interface. The power level will be shown in blue bars with the number of volts in the "Current Value" textbox. If you want to use the volts within other areas of RoboRealm select a variable that the module will write the number of volts to. This variable will then be updated each time the Controller module is run with the current voltage of the robot.
4. Motors - After specifying the COM Port you should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the "Current Value" text box in the Motors area. If the motors do not move (or whine) check your COM Port setting and/or board connections. See [Variable Control](#) for more information on how to programmatically move the robot.
5. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the motors. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Extension based program.
6. Current Value - To manually set the motor position type in the appropriate number (-1000 to 1000, 0 is the default neutral/stop) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate.
7. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the controller does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case you are testing your robot in a precarious position.
8. Analog Input - The module provides for 4 user analog inputs. If you enable the Analog pin you will notice the values in the corresponding read only text boxes change based on the voltage across those pins. If you have a distance sensor connected to an analog pin move your hand in front of the sensor to see these values change. By specifying a variable in the corresponding dropdown you can access that value elsewhere in RoboRealm in order to make decisions about the sensors value.
9. Digital Input - In addition to the 4 analog inputs the module provides 10 digital input pins.
The checkbox (in a disabled state) will reflect the high or low states of the pin.
To automatically send or receive a bit, select or type in a variable that will be set based on the pins value. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown. If no bit is specified the entire value is considered during a set/output.
10. Digital Output - The module provides 8 digital output pins.
As an experiment, you can connect an LED to pin 1 in the Controller. Then by checking and un-checking the checkbox you can make the LED blink under manual control. If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively a higher bit will slow the blinking by a factor of 2 for every bit.
11. Pulse Input - Similar to the Analog Pins you have access to up to 6 pulse pins that are typically driven from an RC type controller.
12. Stop - press the stop button to quickly stop the motors. The motors controls will then be disabled (in case variables are selected) and will only restart when you click the same button again (now renamed to "Start").

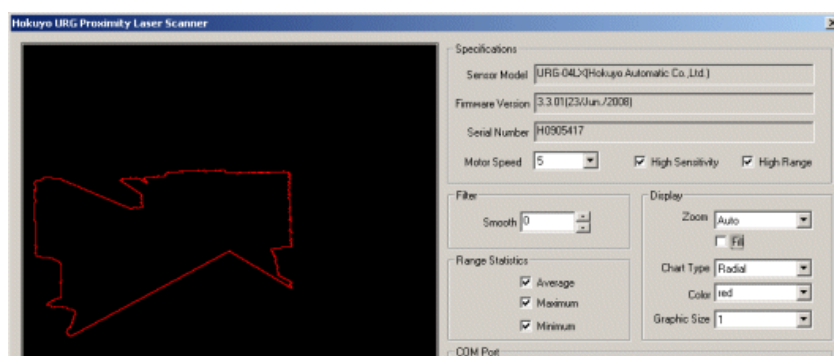
Hokuyo URG Laser Scanner



The Hokuyo URG module provides an interface to the URG laser scanner. This small, compact and speedy scanner is ideal for robots that need obstacle avoidance or navigation using a laser scanner. The scanner is capable of 100 ms scans (10 fps) and can detect objects up to distances of several meters. With a USB connection to a PC this is an ideal scanner to use on your Robot with RoboRealm. You can purchase one of these great scanners from stores like [RoadNarrows](#).

Note that the graphic seen in the interface is also saved as an image in memory with the name HOKUYO_URG_GRAPHIC and can be accessed using the [Marker](#), [Display Image](#), etc. modules.

Interface





Instructions

1. COM Port - select the appropriate COM port that the laser uses. Note that 115K baud will be used for the communication speed.
2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the URG module is loaded the com port will be auto-selected. This ability allows you to not have to constantly change the COM port when loading in successive RoboRealm configurations.
3. Motor Speed - if you are using more than one laser at a time you may wish to select a different motor speed for each of the devices. This helps to isolate any interference between the devices.
4. High Sensitivity - Select if you want to turn on high sensitivity. High sensitivity will increase the lasers detection ability by about 20%. However there may be chances of measurement errors due to strong reflective objects near 22m.
5. High Range - By default the laser produces 12 bit values. If you need greater range select the checkbox and the laser will return 18 bit range values.
6. Filter Smooth - If you notice a bit of errors within the range information try smoothing out the values by increasing the Smooth value. This will average out each signal with its neighbors to provide smoother values.
7. Display Zoom - Select how much zoom to apply to the current view. Auto will detect the furthest object and scale the display accordingly.
8. Display Fill - Check if you want the graphic to be filled with the specified color.
9. Display Chart Type - Select how you want the values to be displayed.



Radial - Displays the range points connected as you would expect them to be seen in reality. Note that areas perpendicular to the sensor are connected despite no actual values being present. (i.e. changes in depth are connected between values).



Cartesian - Displays the range values in a traditional XY chart.



Intensity - Displays the range values as a single row with intensity values based on actual range. Brighter objects are closer.



Points - Displays the raw points as received from the laser scanner.

10. Display Color - Select the color you want the graphic display to use.
11. Graphic Size - Select the line, or point size that should be used when displaying the graphic.

12. Average Statistic - Select if you want a variable to be created that indicates the average of all the range data.

13. Maximum Statistics - Select if you want a variable to be created that indicates the maximum distance within the range data.

14. Minimum Statistics - Select if you want a variable to be created that indicates the minimum distance within the range data.

Variables

HOKUYO_URG_AVERAGE_VALUE - average value of all range data

HOKUYO_URG_AVERAGE_ANGLE - average angle of all range data

HOKUYO_URG_MINIMUM_VALUE - the minimum value detected in all the range data

HOKUYO_URG_MINIMUM_ANGLE - the angle towards the minimum detected value

HOKUYO_URG_MAXIMUM_VALUE - the maximum value detected in all the range data

HOKUYO_URG_MAXIMUM_ANGLE - the angle towards the maximum detected value

Markers

Hokuyo_URG_Graphic - graphic shown in GUI

See Also

[RoboPeak RPLidar](#)

[Laser Line](#)

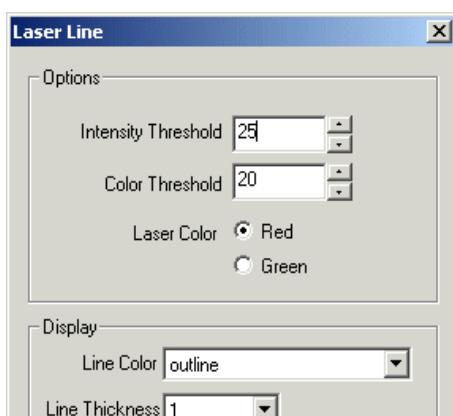
Laser Line

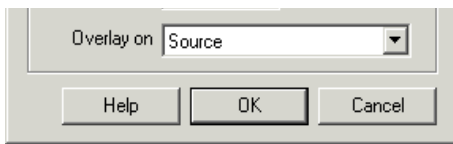
The Laser Line module detects a single red laser line within the current image assumed to be spanning the image in a horizontal direction from the right side to the left side of the image. The primary purpose of the Laser Line module is to detect a horizontal laser line such that one can use the detected values for collision or obstacle avoidance.

This module has been tested with the inexpensive laser line diode from [Instapark](#). We have found this laser to be quite good quality for the price in comparison to other handheld line lasers.

Most lasers will oversaturate any camera. In order to best detect laser spots or lines you should increase the camera shutter speed or decrease the amount of light that is used to create an image. When making this change the overall image will reduce in light but allow the laser line to be better distinguished from the rest of the image and appear more reddish in color. Oversaturating the camera will cause color to become white as the red sensors will bleed into surrounding green or blue sensors.

Interface





Instructions

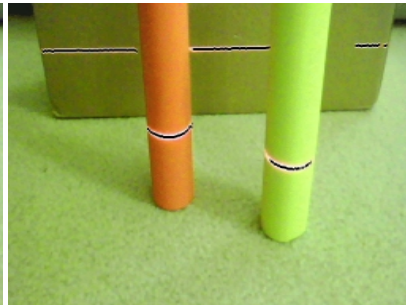
1. Intensity Threshold - the value at which the laser light must be higher/brighter than in order to be detected. You can increase this value to remove darker areas from being incorrectly detected.
2. Color Threshold - the value at which the laser light must contain more red color in order to be detected. Depending on the intensity of your laser and the quality of your camera the laser line may appear more white than red. If this is the case then decrease this value which will allow more non-colored points to comprise part of the line.
3. Laser Color - the color of the laser being used.
4. Line Color - the color of the graphic line drawn where the laser line is detected.
5. Line Thickness - the thickness of the graphic line drawn where the laser line is detected.
6. Overlay On - the image on which the graphic line is drawn.

Example

Source



Laser_Line



Variables

LOWEST_LASER_X, LOWEST_LASER_Y - The X,Y coordinate of the lowest detected part of the line.

HIGHEST_LASER_X, HIGHEST_LASER_Y - The X,Y coordinate of the highest detected part of the line.

AVERAGE_LASER_X, AVERAGE_LASER_Y - The average X,Y coordinate of the line

LASER_SLOPE - The average slope of the detected line

LASER_POINTS - An array containing the Y position of the detected line. Note that this array is the same size as the current width of the processed image. This array is NOT an X,Y list but in fact a single line bitmap that contains a 0 if no red laser point was detected in the column or a non-zero number that indicates the Y position of that part of the detected line. Note that if no line is detected then this array will contain all 0 values. If a straight laser line is detected in the middle of the image against a planar object then all values will be IMAGE_HEIGHT/2.

See Also

[Laser Spot](#)

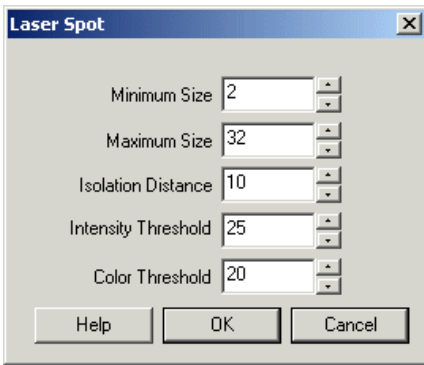
Laser Spot

The Laser Spot module provides an easy way to identify a red laser spot within the current image. The detection routine looks for characteristics that indicate the presence of a red spot that include intensity, color and shape. The module will graphically indicate the location of the laser dots and also create a LASER_POINTS variable array that holds the X,Y locations of the detected spots.

This module can be used with handheld lasers such as those sold by [OrbTronic](#)

This module can be used with hardware lasers such as those sold by [SLO Home](#).

Interface



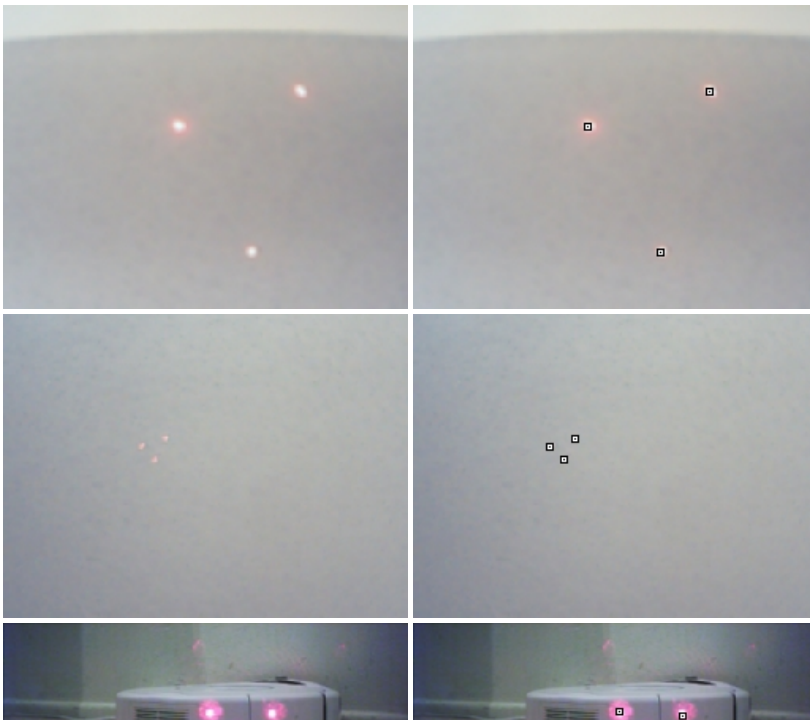
Instructions

1. Minimum Size - Specify the smallest laser dot that you would like to detect. If you notice that your image has many incorrectly detected laser dots try increasing the minimum size which will make the detection much less sensitive to noise. Often noisy pixels can be detected as very small laser spots.
2. Maximum Size - Similar to minimum size the maximum size of the laser spot can be set.
3. Isolation Distance - Specify how close each laser dot can be to other laser dots. Because laser points include noise the detector may decide that one laser dot is in fact more than one. Using the isolation distance you can prevent laser dots from "clumping" together due to noisy detection.
4. Intensity threshold - Laser dots are typically higher in intensity than most (but not all) other parts of the image. Use the intensity threshold if you are unable to detect the laser dot (due to a weak laser) and need to decrease the recognized intensity. Likewise, if too many dots are identified try increasing the value to eliminate bad matches.
5. Color threshold - This module detects red laser dots only. Reduce this threshold if your red dot is not getting recognized. Increase it if too many dots are being identified.
6. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
7. Shape, Color, Size - Specify the shape, color and size of the graphic that is used to indicate the point location. Note the default is the outline shape which has a set size and color.

Example

Source

Laser Spots





Variables

LASER_POINTS - an X,Y array (even numbered array) that contains the X,Y locations of each identified laser dot.

See Also

[Laser Line](#)

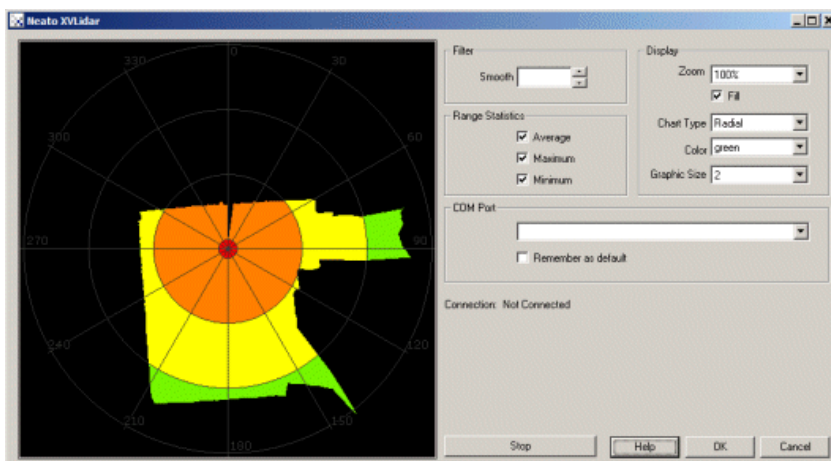
Neato XVLidar



The Neato XVLidar module provides an interface to the XVLidar embedded within a Neato vacuum robot. The XVLidar is a low cost lidar sensor suitable for indoor robotic SLAM application. It provides 360 degree scan field, 5.5hz rotating frequency with a 10 - 15 foot distance. XVLidar is the ideal sensor in cost sensitive areas like consumer robotics and hardware hobbyists.

This module assumes that the [Get Surreal XV Lidar Controller](#) is used to connect a PC via serial to an XV Lidar.

Interface

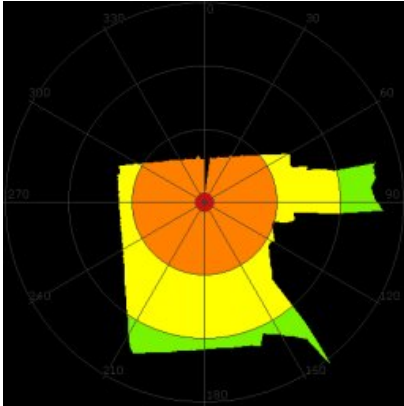


Instructions

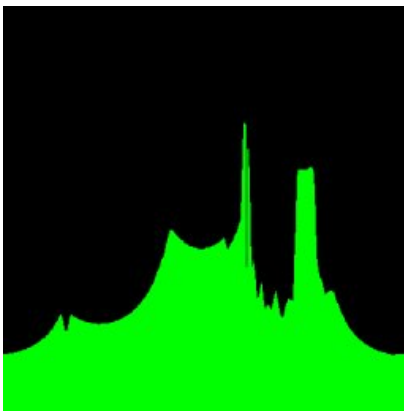
1. COM Port - Specify which COM port the XVLidar is attached to. Once connected the Green connection bar will show.
2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the lidar is loaded the COM port will be auto-selected. This ability allows you to not have to constantly change the COM port when loading in successive RoboRealm configurations.
3. Filter - To smooth out the signal and avoid gaps you can smooth the distance values coming in from the lidar within their local neighborhood. This noisy spikes are reduced as their previous and next angular measurements are averaged into each value.

THIS NOISY SPIRES ARE REDUCED AS THEIR PREVIOUS AND NEXT ANGULAR MEASUREMENTS ARE AVERAGED INTO EACH VALUE.

- 4. Display Zoom - Select how much zoom to apply to the current view. Auto will detect the furthest object and scale the display accordingly. This may cause the map to jump in scale as distant parts are typically noisy.
- 5. Display Fill - Check if you want the graphic to be filled with the specified color as apposed to an outline.
- 6. Display Chart Type - Select how you want the values to be displayed.



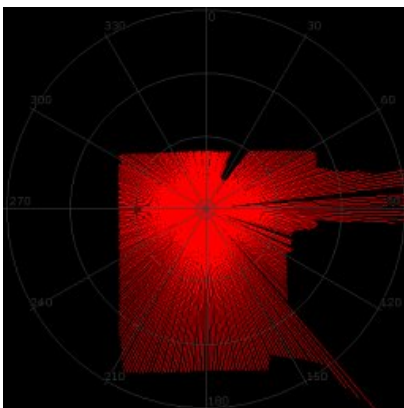
Radial - Displays the range points connected as you would expect them to be seen in reality. Note that areas perpendicular to the sensor are connected despite no actual values being present. (i.e. changes in depth are connected between values).



Cartesian - Displays the range values in a traditional XY chart.



Intensity - Displays the range values as a single row with intensity values based on actual range. Brighter objects are closer.



Points - Displays the raw points as received from the laser scanner.

7. Display Color - Select the color you want the graphic display to use.
8. Graphic Size - Select the line, or point size that should be used when displaying the graphic.
9. Average Statistic - Select if you want a variable to be created that indicates the average of all the range data.
10. Maximum Statistics - Select if you want a variable to be created that indicates the maximum distance within the range data.
11. Minimum Statistics - Select if you want a variable to be created that indicates the minimum distance within the range data.

Variables

NEATO_XVLIDAR_POINTS - the distance points as returned by the XVLidar device

NEATO_XVLIDAR_AVERAGE_VALUE - average value of all the range data

NEATO_XVLIDAR_AVERAGE_ANGLE - average angle of all the range data

NEATO_XVLIDAR_MINIMUM_VALUE - the minimum value detected in all the range data

NEATO_XVLIDAR_MINIMUM_ANGLE - the angle towards the minimum detected value

NEATO_XVLIDAR_MAXIMUM_VALUE - the maximum value detected in all the range data

NEATO_XVLIDAR_MAXIMUM_ANGLE - the angle towards the maximum detected value

Markers

Neato_XVLidar_Graphic - graphic shown in GUI

See Also

[Hokuyo URG
Laser Line](#)

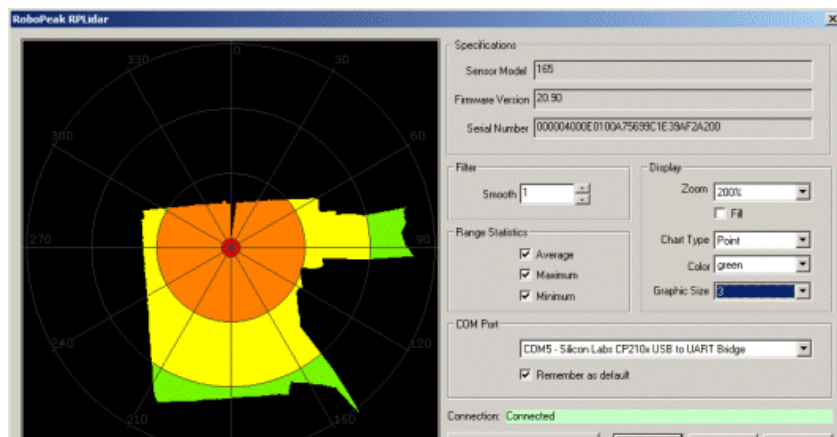
RoboPeak RPLidar



The RoboPeak RPLidar module provides an interface to the RPLidar device. The RPLidar is a low cost lidar sensor suitable for indoor robotic SLAM application. It provides 360 degree scan field, 5.5hz rotating frequency with guaranteed 6 meter ranger distance. RPLidar is the ideal sensor in cost sensitive areas like consumer robotics and hardware hobbyists.

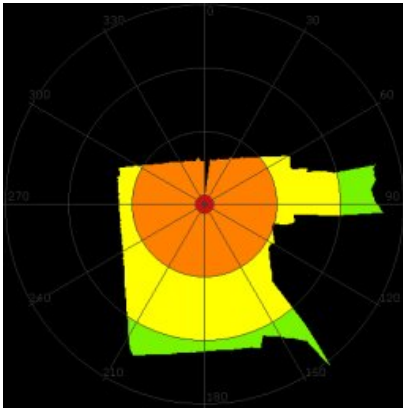
Special thanks to Dave Everett for the loan of the RPLidar. Be sure to visit this [Robot Workshop](#) in Sydney.

Interface

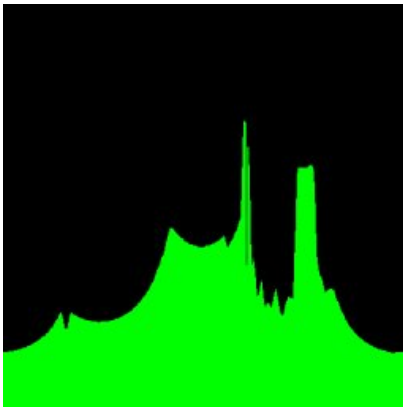


Instructions

1. COM Port - Specify which COM port the RPLidar is attached to. Once connected the Green connection bar will show and the device Model, Firmware and Serial number will appear in the interface.
2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the lidar is loaded the COM port will be auto-selected. This ability allows you to not have to constantly change the COM port when loading in successive RoboRealm configurations.
3. Filter - To smooth out the signal and avoid gaps you can smooth the distance values coming in from the lidar within their local neighborhood. Thus noisy spikes are reduced as their previous and next angular measurements are averaged into each value.
4. Display Zoom - Select how much zoom to apply to the current view. Auto will detect the furthest object and scale the display accordingly. This may cause the map to jump in scale as distant parts are typically noisy.
5. Display Fill - Check if you want the graphic to be filled with the specified color as apposed to an outline.
6. Display Chart Type - Select how you want the values to be displayed.



Radial - Displays the range points connected as you would expect them to be seen in reality. Note that areas perpendicular to the sensor are connected despite no actual values being present. (i.e. changes in depth are connected between values).



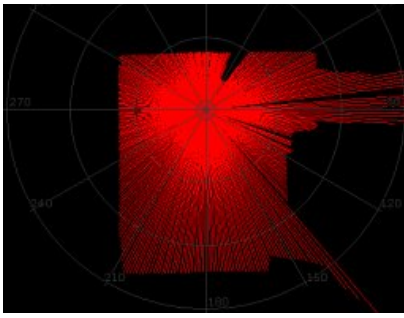
Cartesian - Displays the range values in a traditional XY chart.



Intensity - Displays the range values as a single row with intensity values based on actual range. Brighter objects are closer.



Points - Displays the raw points as received from the laser scanner.



7. Display Color - Select the color you want the graphic display to use.
8. Graphic Size - Select the line, or point size that should be used when displaying the graphic.
9. Average Statistic - Select if you want a variable to be created that indicates the average of all the range data.
10. Maximum Statistics - Select if you want a variable to be created that indicates the maximum distance within the range data.
11. Minimum Statistics - Select if you want a variable to be created that indicates the minimum distance within the range data.

Variables

ROBOPEAK_RPLIDAR_POINTS - the distance points as returned by the RPLidar device

ROBOPEAK_RPLIDAR_AVERAGE_VALUE - average value of all the range data

ROBOPEAK_RPLIDAR_AVERAGE_ANGLE - average angle of all the range data

ROBOPEAK_RPLIDAR_MINIMUM_VALUE - the minimum value detected in all the range data

ROBOPEAK_RPLIDAR_MINIMUM_ANGLE - the angle towards the minimum detected value

ROBOPEAK_RPLIDAR_MAXIMUM_VALUE - the maximum value detected in all the range data

ROBOPEAK_RPLIDAR_MAXIMUM_ANGLE - the angle towards the maximum detected value

Markers

RoboPeak_RPLidar_Graphic - graphic shown in GUI

See Also

[Hokuyo URG](#)
[Laser Line](#)

A-WIT BOL-BOT



The A-WIT BOL-BOT module provides access to the BOL-BOT robot made by [A-WIT Technologies](#) that runs the c-stamp microprocessor.

The module expects to interact with a preloaded cstamp application that can be downloaded [here](#) or you can download the entire source for the project [here](#).

The cstamp application expects to interact with a PC connected via bluetooth which needs to be assembled on the provided BOL-BOT breadboard. See the BOL-BOT website documentation for further instructions about connecting the BOL-BOT to a PC via bluetooth. Note that pins 36 (TXD), 34 (CTS), 35 (RXD), 33 (RTS) and 32 (RST) are assumed by the cstamp program as those in use by the ESD200 bluetooth chip.

Interface

A-WIT BOL-BOT

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Variable	Trim	Current Value	Min/Max Limits	
Pin 16	-3	550	300	800
Pin 17	-15	550	300	800
Pin 18	0	550	300	800
Pin 19	0	550	300	800

Communication

COM Port: CDM77 - Standard Serial over Bluetooth link

Baud Rate: 9600

Stop

Help OK Cancel

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Enable

Variable	Pin	Value	Variable	Pin	Value
#1			#5		
#2			#6		
#3			#7		
#4			#8		

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Enable

Variable	Pin	Value
#1		
#2		
#3		
#4		

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Enable

Variable	Pin	Value
#1		
#2		
#3		
#4		

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Enable

Variable	Tx Pin	Rx Pin	Value
#1			
#2			
#3			
#4			

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Frequency: [] []

Duration: [] []

Pin: []

Play

Servos | Bumpers | Photo Resistors | Sonars | IR Sensors | Sound | Contrast Sensors | IO Pins

Enable

	Variable	Pin	Value
#1	<input type="text"/>	<input type="text"/>	<input type="text"/>
#2	<input type="text"/>	<input type="text"/>	<input type="text"/>
#3	<input type="text"/>	<input type="text"/>	<input type="text"/>
#4	<input type="text"/>	<input type="text"/>	<input type="text"/>

Servos Bumpers Photo Resistors Sonars IR Sensors Sound Contrast Sensors IO Pins									
<input type="checkbox"/> Enable									
	Variable	Type	Pin	Value		Variable	Type	Pin	Value
#1	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	#5	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
#2	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	#6	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
#3	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	#7	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>
#4	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	#8	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>

Instructions

1. COM Port - First select the appropriate COM port where the bluetooth has been attached to. Note that only the COM ports that are recognized by the system are displayed. If you have not yet attached your bluetooth USB device or activated the internal bluetooth wireless capabilities you should do so before running RoboRealm otherwise the appropriate COM port will not show up.

Once RoboRealm has connected to the BOL-BOT successfully the LED4 will flash for as long as the connection is held. If either the PC or the BOL-BOT stops responding a 3 second failsafe will be activated and terminate the current BOL-BOT actions. At that time all servos will stop but pins will retain their last setting as specified by this module. If you want to use LED4 for your own purposes select the "Deactivate LED4 connection flashing".

2. Baud Rate - Select the appropriate baud rate. Currently the ESD200 supports 9600 baud so the default should be left as set.

3. Servos - To test the module ensure that two servos are connected to the servo pins 16-19 and slowly move the appropriate servo sliders. If everything is working the servos should slowly start to move. If this does not happen check the wiring on the bluetooth chip, check your COM port settings, or that your bluetooth device is on. Note that when first connecting the devices can take up to 30 seconds to secure a connection so be patient. Be on the lookout for the pairing password prompt that may pop up during the initial communication between your PC and the BOL-BOT. If you have not set a pairing password the default 1234 should suffice.

If the servos move you can then select a variable (or type one in) that will be populated from another module that will contain the values to send to the BOL-BOT. Note that 550 is considered neutral whilst 300 and 800 are the opposite extremes. If your servos move even when using 550 as a setting adjust the trim values until they stop. By customizing the trim values you could change servos without needing to adjust all the servo speeds used to calculate movements.

4. Bumpers - assuming you have constructed the bumpers on the BOL-BOT breadboard pressing one or both of the bumpers will change the appropriate checkbox by turning it on or off. Be sure to have enabled the Bumpers by clicking on the checkbox in the Bumpers tab which tells RoboRealm to start querying the bumper states on the BOL-BOT. Be sure to specify which pin the bumpers are connected to. If you are following the instructions in the BOL-BOT assembly you will have used pins 14 and 37.

If you want to access the bumper state from other modules select or type in the variable name that will contain a "1" when the bumper is pressed and "0" when the path is clear.

5. Photo Resistors - select the appropriate pin that a photo resistor is connected to and you should start seeing the bar graph adjust based on the detected lighting. Note that the value will range from 0 to 5 to reflect the amount of voltage (5V max) allowed to pass through the photo resistor. To utilize this number in other modules select or type a variable that will be set to the resistance amount. Keep in mind that any ambient light will be detected by the resistor including sunlight or spotlights.

6. Sonars - select the appropriate pin that the sonar is connected to. On selection the bar graph will represent the distance amount that the sonar is detecting. The distance will range from about 2 to 30. Distances past 30 are detected but will not be very reliable. Keep in mind that sonar uses sound to detect distances. You should see the green light on top of the Sonar sensor brightness reflect the distance being detected. Sound also reflects and thus can be confused by flat surfaces that reflect the return sound away from the sensor. In this case the sensor will return a distance much further than in reality. This can be seen by using a flat book and angling the front at a 45 deg with respect to the sensor.

7. IR Sensors - assuming that you have constructed the IR sensors on the BOL-BOT the appropriate checkbox will turn on when something is in front of the IR sensor. Be sure to have turned on the IR Sensors by selecting the checkbox in the IR Sensor tab which will tell RoboRealm to start querying the state of the IR sensors on the BOL-BOT. If you move your hand in front of the IR sensor you should see the check in the checkbox disappear. Note that the IR sensor used is a digital sensor and detects an object at a distance based on the resistor used.

If you want to access the IR states from other modules select or type in the variable name that will contain a "1" when the IR sensor detects an object and "0" when the path is clear. Note that the IR sensor is a binary sensor and does not return range information as some other IR sensors do.

8. Sound - to test that the piezo buzzer is working select a frequency from the second dropdown menu and a appropriate during in the next dropdown. Click on Play and you should hear the note coming from the BOL-BOT.

To automatically play a sound type in or select a variable for both frequency and duration that will contain the appropriate frequency and duration number. When these variables become set RoboRealm will command the BOL-BOT to play the note and then will remove the variables' values. The variables are cleared to prevent the note from being played again and again. To once again play a note, simply set the variables again with appropriate values.

9. Contrast Sensors - the contrast sensor is best used for line following. Once the appropriate pin that the contrast sensor is connected to is specified the BOL-BOT will start sending back the amount of light detected by the sensor in the associated bar graph. If you move a piece of paper in front of the sensor you should notice that value change.

10. IO Pins - The BOL-BOT comes with many IO pins. Most of these pins are for user usage and can be used as input or output pins in both analog and digital formats. RoboRealm provides a way to either receive or send signals to these pins via the IO Pins tab. Note that several of the pins may already be in use by the bumpers, photo resistors, sonars, etc. Using the IO Pins tab you can use the interface to interact with other digital devices other than those provided in the base BOL-BOT kit.

Each pin can either be set as DIN (digital input), DOUT (digital output), AIN (analog input), and AOUT (analog output). Do this by selecting the appropriate Type dropdown next to each pin. Use the corresponding text field to enter in a manual number for DOUT and AOUT, otherwise that value will come from the selected variable. For DIN and AIN the specified variable will contain the value received from that pin.

For DIN and DOUT the text field will reflect a 0 or 1 to signal the value.

Arcbotics Sparki



The Arcbotics Sparki module provides an interface to the [Arcbotics Sparki robot](#). Sparki is a very capable robot that is great for beginners to learn about robotics and great fun for advanced folks to test their programming capabilities. Sparki is loaded with sensors and motors that provide a very rich environment to work in. In addition, a bluetooth module that allows Sparki to communicate to a PC wirelessly makes a great addition to the suite of abilities.

The RoboRealm module provides access to all these abilities in a graphical way such that you can test and learn more about what the sensors do and when they work best.

Sparki's 'brain' is based on the Arudino microcontroller. In order for RoboRealm to successfully communicate with Sparki you need to download the sketch program file that will need to be uploaded to the robot before the RoboRealm module can communicate with Sparki. You can also use this file as another example of how to program the onboard Arudino to access the various components of Sparki.

Download the [RoboRealm Sparki Sketch](#) that needs to be compiled and uploaded into your Sparki before this module will become active. This sketch provides a communication protocol over serial to Sparki from the PC. Note that it is highly recommended to start with a connected USB cable to Sparki as the serial method. Once you have verified correct operation you can then switch to bluetooth by selecting the different serial port AND baud rate (note, that Bluetooth communication is only 9600 baud).

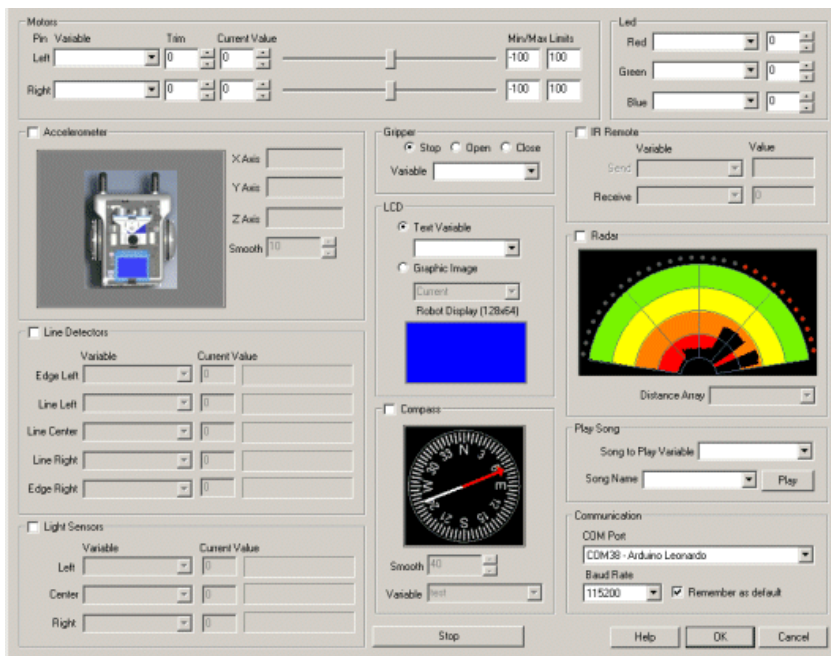
With the USB cable connected to Sparki and previously mentioned sketch program being uploaded, you should then select the right COM port with the baud rate set to 115200. Once this is done, select the checkbox to enable to accelerometer interface. You should see numbers changing on the right of that image with the image moving to illustrate what the numbers are showing. If you now pick up or tilt/twist Sparki you should see the image change accordingly.

The checkboxes located in many of the interface areas allow you to selectively decide what information you want to receive. This is done in order to reduce the overall communication overhead with Sparki to ensure quick response. As many configurations do not require all sensor information from Sparki these checkboxes can be used to effectively reduce the communication and processing load on Sparki.

Installation

1. Download and install the [Arcbotics SparkiDuino Application](#).
2. Download our [RoboRealm Sparki Sketch](#) files.
3. Unzip this into a folder that you can remember.
4. Load in the ArcBotics_Sparki.ino file into Sparkduino
5. Upload that to Sparki over a USB connected cable (press the right arrow button).
6. Run RoboRealm
7. Click on the Search tab (upper left corner) and type in Sparki. Double click the results to insert that module into the pipeline.
8. Select the appropriate COM port, enable the accelerometer checkbox and see if you get movement when touching Sparki.

Interface



Instructions

1. COM Port - Specify the COM port that Sparki is connected to. Note that Sparki comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.
2. Baud Rate - Specify the baud rate of 115200 to communicate with Sparki. Note that this is much higher than the standard 9600 normally used. The higher rate provides much better responsiveness than the lower rates. If you plan to use bluetooth the default rate is 9600 which will be much slower than a USB connection.
3. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever a Sparki module is inserted into the RoboRealm pipeline the COM port and baud rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations as long as the connection properties remain the same.
4. Motors - Once communications is specified you should be able to move the appropriate slider in order to see the wheels move. If the motor does not move, check your USB connection and battery switch. To learn more about Sparki's wheels see the [Sparki Wheel Parts](#).
5. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the motors. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Plugin based program or Joystick or any other module that is used to update variables. See [Variable Control](#) for more information on how to programatically move the robot.
6. Current Value - To manually set the motor position, type in the appropriate number (-100 to 100, 0 is the default neutral) into the text area or use the slider to adjust the value. The motor speed will be updated as appropriate.
7. Sliders - You should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the current value text box. If the motors do not move check your USB cable and/or board power connections.
9. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case your motors cannot physically move beyond certain limits.
10. LED - The led is a full color LED capable of generating a lot of colors. You can use the spin buttons or type in a number in the provided text boxes to see an immediate change in the LED. The 3 channels (red, green and blue) can be used in combination to specify many colors. Note that when a variable is selected the value field will be disabled to indicate that the specific LED channel value is now being received from a variable. To learn more about Sparki's LED see the [Sparki LED Part](#).
- 11 Accelerometer - The 3 axis accelerometer provides orientation information for Sparki. To enable this functionality click on the checkbox next to "Accelerometer" which will enable receiving of orientation information. The X and Y values indicate roll and tilt with the Z axis being a check to determine if the robot is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (that's what the compass is used for). The values are stored in 3 variables that can be accessed by other modules (SPARKI_X_AXIS, SPARKI_Y_AXIS, SPARKI_Z_AXIS) for you to use.

Note that the simulation graphic uses the X Axis1 and Y Axis for tilt sensing and checks the Z Axis to determine if the robot is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (Z axis) and thus the simulation graphic will assume you are holding the robot with the LCD screen towards you and the sonar pointing away.

To learn more about Sparki's Accelerometer see the [Sparki Accelerometer Part](#).

To learn more about Sparki's Accelerometer see the [Sparki Accelerometer Part](#).

12. Smooth - If the accelerometer values are not as responsive as you would like try reducing the smooth value. This will allow the numbers to be much more reactive but may also cause some unwanted jitter due to noise in the determined values. By decreasing the smooth value you make the sensors much more responsive to movement, increasing the value will delay the sharpness of the movement but also make it much more smooth. Note this is reflected in the simulation graphic.

13. Gripper - To open/close/stop the gripper manually click on the appropriate radio button. In order to allow for this feature to be programmed, you can type in a variable that holds the value 'open' or 'close' or 'stop' which will then command the gripper to that state. This allows you to trigger the gripper movement based on some other module. The gripper motor will activate for a short period of time and then stop to avoid over burdening the motor. The timing should be sufficient for fully opening and closing the gripper. To learn more about Sparki's Gripper see the [Sparki Gripper Part](#).

14. IR Remote - You can receive button press information from the IR remote control that is included in your Sparki purchase. To enable this functionality click on the checkbox next to "IR Remote" which will enable receiving of IR remote information. When enabled, you can press a button on the IR remote and that corresponding button name will appear in the interface. You can then select or type in a variable that will also be set to that button to use this information in other modules. To learn more about Sparki's IR Remote Sensor see the [Sparki Infrared Remote Part](#).

15. LCD - Sparki comes equipped with a 128 x 64 pixel blue and white LCD screen. This can be controlled individually as pixels or as text. If you select the 'Text Variable' radio button and type in a variable that contains some text (for example the IMAGE_COUNT variable) you will see that text represented in the simulated LCD screen but also transmitted to Sparki too. You can also select a 'Graphic Image' to be used instead. When a 'Graphic Image' is selected this will cause the current image loaded in RoboRealm (including any webcam image currently active) to be reduced in size, dithered and sent over to Sparki such that the blue and white LCD will then display to image. Keep in mind that while the connection to Sparki is fast and the image is small, it cannot keep up with most webcams so you will notice some lag.

If you prefer to create your own custom screens you can use the [Resize Canvas](#) module to change the image within RoboRealm to a 128 by 64 pixel image and then use the display modules to draw your own graphics on that image. Sending this to Sparki will then reflect that screen on Sparki's LCD panel. Keep in mind that Sparki's screen is two color (white or blue) which means images may not appear as they do on your monitor.

To learn more about Sparki's LCD see the [Sparki LCD Part](#).

16. Radar - The most striking feature of Sparki is the sonar sensor mounted on top of a servo. This gives Sparki a personality by emulating a robot's head. The combination of a sonar sensor mounted onto of a servo allows for the sonar to move and scan in the environment.

To enable sonar scanning click on the checkbox next to "Radar" which will enable the movement of the servo and receiving of distance information generated by the sonar. This information will be graphically displayed as the sonar sweeps through its range. Note that because the sonar is a point sensor, only part of the radar map will be updated at any particular time. The red dots shown just beyond the distance information indicates where the sonar is currently aiming.

The dark areas near the center of the radar are assumed to be obstacle free. The values that show color are objects that blocked the sonar's scan and thus are considered to be obstacles.

The distance information is represented as "cm" and provided in an array of values when you type in or select a variable next to the "Distance Array" text. This variable will then contain all the information that is used to create the radar graphic and can be further analyzed to determine an obstacle free path.

To learn more about Sparki's Ranger see the [Sparki Ultrasonic Range Finder Part](#).

16. Line Detectors - Sparki comes equipped with 5 line sensors located just under the gripper. These sensors are IR based and produce a value that indicates how much IR light they are receiving based on how much IR is reflected from the surface that Sparki is currently on. These values will differ based on the intensity of the surface, lower values for dark surfaces, higher values for lighter surfaces. Using the differences in values you can determine how to move Sparki to keep on the line.

To enable this functionality click on the checkbox next to "Line Detectors" which will enable receiving of line intensity information. The blue indicators will show the sensed values with the 'Current Value' box showing the actual number. To perform calculations on these numbers select or type in a variable that will be set to that value. These variables can then be used to perform calculations in other modules.


To learn more about Sparki's Infrared Reflectance Sensors see the [Sparki Infrared Reflectance Sensor Part](#).

17. Light Sensors - Similar to the line sensors, Sparki also has light sensors mounted in the front to detect light intensity. These controls have similar functionality to the Line Detectors. Once you activate these sensors you can use a flashlight to shine across the front of Sparki to see these values change. To learn more about Sparki's Light Sensors see the [Sparki Light Sensor Part](#).

18. Play Song - You can quickly select a song and press play to hear Sparki play a quick melody. To automate the playback of songs select a variable that will contain the song name as seen in the dropdown. Note that after the song begins playing the variable is cleared to avoid repeating the song. To add your own songs you can edit the "Music.rtttl" file in the RoboRealm folder. This file contains RTTTL formatted melodies which are converted and sent to Sparki for playback. Note that the RTTTL format is the Nokia Cell Phone ringtone format and can be found for free in many sites.

Be aware that while Sparki is playing the tone the robot is unable to complete any other tasks. Playing a song requires the full attention of Sparki and thus will become unresponsive to other commands while the song is playing.

To learn more about Sparki's Buzzer see the [Sparki Buzzer Part](#).

 [Click Here](#) to download an example that shows how you can move Sparki using a Joystick. You will have to edit (double click) on the Joystick module to ensure your joystick is connected and the Sparki module to ensure that the correct COM port is selected. You can also press button 1 on your joystick to close the gripper and button 2 to open it. If you'd like to use different buttons you can reconfigure this by editing the Joystick module and moving the variable from button 1 and 2 to some other button.

Variables

SPARKI_X_AXIS - X axis accelerometer value

SPARKI_Y_AXIS - Y axis accelerometer value

SPARKI_Z_AXIS - Z axis accelerometer value

See Also

[BirdBrain Technologies Finch Robot](#)

[GCtronic Epuck](#)

CoroWare CoroBot

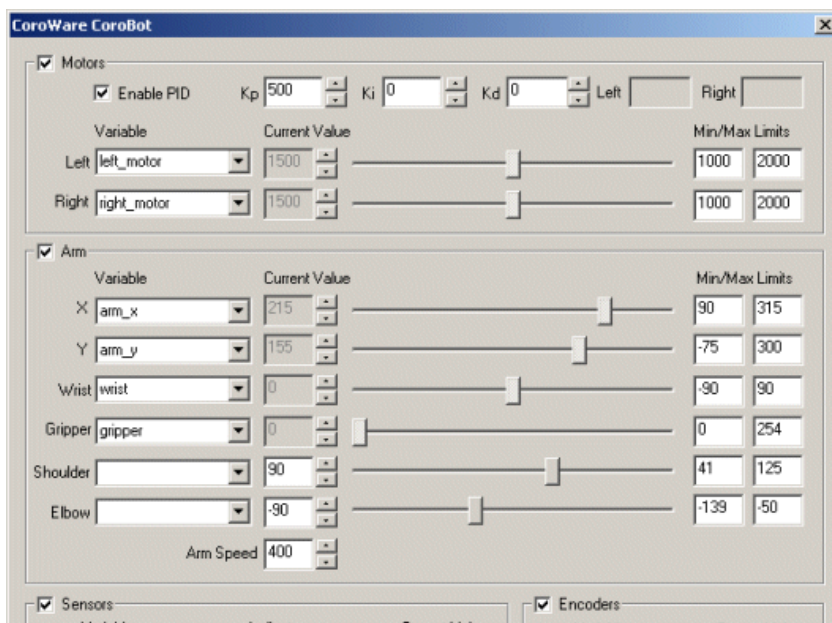


The CoroWare CoroBot module provides an interface to the CoroBot robot created by CoroWare and distributed by [RoadNarrows](#). The CoroBot is a PC based robot that is capable of running RoboRealm onboard (WinXP version). The robot is constructed using several standard interface cards that allow the PC to communicate to the underlying hardware using both serial and USB connections. The CoroBot uses the [Lynxmotion SSC](#) for motor and arm control, the [Phidgets Encoder](#) board for motor encoders and the [Phidgets 888 Interface Kit](#) to read the battery voltage, gripper switch and front/back IR distance sensors.

While the CoroBot module provides connection to all the required boards to command the robot you can also disable the appropriate segments and instead add in the full board modules supported by RoboRealm if you need lower level interfacing or require more aspects of that particular board. For example, if you need to add additional IR sensors you can disable the Sensor group in this interface and add in the [Phidgets 888 Interface Kit](#) board to provide you with the current 4 sensors (Battery, Front/Back IR, Gripper Touch) and any additional channels you may need.

The CoroBot contains a full PC, therefore all other modules within RoboRealm such as the [Play Wave File](#) and [Speech Recognition](#) will work with the appropriate attachment of speakers, mic, etc.

Interface





Instructions

1. Motors - Select the checkbox next to "Motors" to enable the motor channels. To test the motors slowly move the appropriate sliders. If everything is working the motors should slowly start to move. If this does not happen check that your SSC is connected and configured on COM2.

If the motors move you can then select a variable (or type one in) that will be populated from another module that will contain the values to send to the CoroBot. 1500 is considered neutral whilst 1000 and 2000 are the opposite backward/forward extremes.

When a variable is selected the slider and value controls will be disabled to indicate that control is now performed through the use of a variable.

You can change the Min and Max values for the motors. This will ensure that regardless of the variable's value that the motors will never be sent a command beyond those limits. You can tighten the limits to 1250 and 1750 to ensure that the robot moves slowly regardless of variable errors.

Keep in mind that the CoroBot runs both wheels on either side at the same speed. So to pivot set one side to a value higher than 1500 and the other side to less than 1500. Speeds while moving forward versus turning will differ due to the stress placed on the wheels when turning. Also keep in mind that the CoroBot runs better over floors than carpet largely due to the wheel friction during turning on carpet.

The Motor commands are passed to the [Lynxmotion SSC](#) to perform the actual servo movements.

See [Variable Control](#) for more information on how to programatically move the robot.

2. Enable PID - Enables Proportional, Integral, Derivative control over motor movements. This formula will adjust the actual motor values to be different than that of the desired motor values to ensure that the motors are actually moving in such a way that best represents the desired speed. For example, if you are traveling along a flat surface with motor speeds set at 1300 (slowly forward) and suddenly encounter a hill this slow speed will not suffice to get the robot over the hill. With the PID loop enabled the lack of actual motion will be sensed by the reduced encoder count and cause the actual motor values to increase such that the encoders continue to report the same count. Thus the robot will not slow down when encountering a hill. See [Wikipedia](#) for more information about PID loops.

3. Arm - Select the checkbox next to "Arm" to enable the optional Lynxmotion Arm controls. Please note that as the arm moves beyond its actual abilities increasing X or Y may cause the arm to move in unexpected ways. Also note that it is possible to position the arm in a destructive position such as against the robot or below the floor. The following controls are very unrestricted and limits should be reduced depending on your application to prevent the arm from damage.

The Arm commands are passed to the [Lynxmotion SSC](#) to perform the actual servo movements.

4. Arm X - Select or type in a variable that will contain the desired X location of the arm. X is always positive and moves the arm horizontally parallel to the floor plane. The numbers specified are in millimeters. Modifying the X location will change the Shoulder and Elbow angles accordingly (inverse kinematics).

5. Arm Y - Select or type in a variable that will contain the desired Y location of the arm. Y can be negative (touching the floor) and moves the arm vertically. The numbers specified are in millimeters. Modifying the X location will change the Shoulder and Elbow angles accordingly (inverse kinematics).

6. Arm Wrist - Select or type in a variable that will contain the desired wrist degree that ranges from -90 to 90 with 0 being horizontal.

7. Arm Gripper - Select or type in a variable that will contain the desired width of the gripper that ranges from 0 (Open) to 254 (Closed).

8. Arm Shoulder - Select or type in a variable that will contain the desired shoulder degree of the arm. Changing this value will change the X and Y location of the arm accordingly (forward kinematics).

9. Arm Elbow - Select or type in a variable that will contain the desired elbow degree of the arm. Changing this value will change the X and Y location of the arm accordingly (forward kinematics).

10. Arm Speed - Select how quickly that the arm will move. Higher millisecond times will case the arm to move slower. Smaller times will cause the arm to move faster but become much more jerky when moving due to servo torques.

11. Sensors - Check the checkbox to enable sensor readings. The Sensors use the [Phidgets 888 Interface Kit](#) to read the appropriate values.

12. Front Sensor - Select or type in a variable that will contain the IR sensor reading in the front of the robot. This sensor will typically range from around 0 to 500 with 500 being very close. Once objects pass nearer than the 500 mark the values may actually decrease slightly before the object

around 0 to 500 with 500 being very close. Once objects pass nearer than the 500 mark the values may actually decrease slightly before the object contacts with the robot.

13. Back Sensor - Select or type in a variable that will contain the IR sensor reading in the back of the robot. This sensor will typically range from around 0 to 500 with 500 being very close. Once objects pass nearer than the 500 mark the values may actually decrease slightly before the object contacts with the robot.

14. Battery Sensor - Select or type in a variable that will contain the battery voltage. The voltage will range from a peak of 12 volts down to zero (note that the robot will stop functioning well before 0 is reached!).

15. Gripper Sensor - Select or type in a variable that will contain the pressure value of the touch sensor within the gripper. A value of 1000 means the gripper is very securely closed with a value of 0 meaning nothing is touching.

16. Encoders Left/Right - Select or type in a variable that will contain the left and right encoder values read from the motor encoders. These values are also used internally by the PID loop to ensure constant motor movement.

17. Rotate Image - Often the CoroBot camera is mounted upside down. Select this checkbox to flip the image over for easier viewing.

Examples

It is HIGHLY recommended to place the CoroBot on a stand before accessing any of the below examples. Some tests will require the joystick #1 button to be pressed in order to start but others will not. Thus accessing any of the following examples and running them on the CoroBot may suddenly cause the robot to start moving in a possibly hazardous direction.

Due to the high friction of the CoroBot wheels most of these demos will NOT work correctly on thick carpet as the robot is unable (even with PID enabled) to pivot in place. Try it with the Joystick demo to see if your carpet passes the test!

Joystick


 [Click Here](#) to load a robofile that will allow you to drive your CoroBot using a joystick. When you load this configuration you will need to edit the Joystick module to select which joystick to use to control the robot.

This robofile is configured for a playstation type of joystick with the first knob providing motion control. Moving the knob forward/backward moves the robot forward/backward. Moving the knob left/right pivots the robot in the appropriate direction (if not on carpet!). Moving the second knob forward/backwards moves the arm up/down (Y axis) and moving left/right moves the arm forward/backward (X axis). Clicking buttons 6 and 8 opens and closes the gripper with buttons 5 and 7 twisting the arm.

If you do not like the configuration we have chosen simply edit the Joystick module and change the variables around to different buttons.

Once that is done have a safe and happy drive!


Laser Steering

 [Click Here](#) to load a robofile that will allow you to drive your CoroBot using a red laser light spot placed in front of the robot visible from the camera. You should adjust the camera to a higher location than installed by default and angled down towards the floor in order for this example to work. The higher location of the camera will allow more coverage in front of the robot. The default camera has a narrow field of view and cannot see much floor space in front of the robot as installed.


By moving the laser forward the robot will move forward, placing the laser spot close to the robot will cause it to back up. Moving it to the side will cause the robot to pivot. Be sure not to move the laser too quickly and keep it within view of the camera.

Be warned that any spot lights maybe detected as laser lights and cause the robot to move at random. Be sure to test on blocks before releasing the robot.

IR Avoidance

 [Click Here](#) to load a robofile that will cause the CoroBot to avoid objects using the forward IR sensor. The strategy is to run forward until a close object is detected, back up for a few encoder counts, rotate and then proceed forward. Be sure to test this while the robot is elevated before committing the robot to moving towards a wall!

Laser Line Avoidance

 [Click Here](#) to load a robofile that will cause the CoroBot to avoid objects using a red laser line as seen in the [Laser line](#) module. The CoroBot does NOT come with a laser line and thus you will have to purchase one in order to use this example.


Using the red laser line (ideally in a low light situation) the robot can detect objects and turn to avoid them with more reliability than using the IR detector. The additional reliability is due to the larger field of view that is possible with the camera + laser line than what the IR detector will provide.

You should adjust the camera to a higher location than installed by default and angled down towards the floor

You should adjust the camera to a higher location than installed by default and angled down towards the floor.

Keep pressing the Joystick button #1 in order for this demo to run. Releasing the button will cause the CoroBot to stop.

Line Following

 [Click Here](#) to load a robofile that will cause the CoroBot to follow a line using the build in camera. You should adjust the camera to a higher location than installed by default and angled down towards the floor in order for this example to work. The line is ideally black electrical tape against a white floor. If you instead have a white line on a black floor disable or delete the Negative module in this example.

Keep pressing the Joystick button #1 in order for this demo to run. Releasing the button will cause the CoroBot to stop.

See Also

- [Lynxmotion SSC](#)
- [Phidgets 888 Interface Kit](#)
- [Phidgets Encoder](#)
- [Rotate](#)

BrookStone Rover



The BrookStone Rover module provides an interface to the BrookStone Rover. The Rover is a inexpensive mobile camera that is targeted towards telepresence functions but also makes a decent robot when combined with a vision system. As the robot transmits images much like an IP Camera does (in fact the internals are partially based on an IP Camera) the processing of images needs to happen off the robot. As the communication uses 802.11 the robot is capable of transmitting 320x240 or 640x480 sized images at 30 frames per second (fps) assuming

sufficient lighting. This image can then be processed within RoboRealm and movement commands can be send back to the robot.

As with most IP Cameras and webcams in general, if insufficient lighting is available you will get a reduced frame rate. In addition, more motion blur will be visible as the camera image exposure will increase. To get best results keep the robot within sufficient lighting or turn on the IR light.

Note that this Rover (version 1.0 is white) is no longer available from Brookstone but you can find them available in [Amazon](#) or [Ebay](#).

Interface

Motors	
Variable	Current Value
Left: left_motor	0
Right: right_motor	0

Connection	Video Size	Other
IP Address: 192.168.1.100	<input type="radio"/> 320x240	<input type="checkbox"/> IR Activated
Port Number: 80	<input checked="" type="radio"/> 640x480	
Username: AC13		
Password: AC13		
<input checked="" type="checkbox"/> Remember as default		


Instructions

1. Connection - Switch on the Rover and connection to the AC13_XXXXX ssid that should be available a couple seconds after switching on the Rover. Assuming you have not changed any of its configuration all the default information should be correct. After a couple seconds the module should connect to the Rover and start streaming video.

It is possible to connect the Rover to your own network and drive the Rover via the Internet by using a different IP address and a copy of RoboRealm on the remote side. While possible, the setup of such a configuration is atypical and should not be attempted by those unfamiliar with port forwarding, firewalls, etc.

2. IP Address - The IP address of the Rover. Do not change unless you have changed this on the Rover. Default is 192.168.1.100.
3. Port - The connection port of the Rover. Do not change unless you have changed this on the Rover. Default is 80.
4. Username/Password - The authorization information used to gain access to the Rover. Default is AC13/AC13
5. Remember as Default - If you have changed the default configuration be sure to leave the Remember checkbox checked. This will ensure that each time you add the Rover module it will use your last configuration information as the default.
6. Motors - The motor control allow you to change the direction and speed of the Rover's motors (i.e. to drive it).
7. Motors Slider - By dragging the sliders you can move the Rover wheels appropriately. Note that this is not the easiest way to drive the Rover as you can only change one wheel at a time by dragging with the mouse. The sliders are meant for testing purposes (more on easier driving later).
8. Motors Current Value - The motor values will range from 10 to -10 depending on the desired speed. Again this interface is for testing purposes as you will not normally driver the Rover using an actual number.
9. Motors Variable - The best way to driver the Rover is to use another module as the interface which modifies variables that contain the value sent to the Rover motors. You should select those variables (or type them in if not already created) into the dropdown box. Note that this will disable the manual slider and value boxes to indicate that the motors are now under control from the variable. These variables will then be set appropriately by other interface modules and provide a common communication method to the Rover. See [Variable Control](#) for more information on how to programatically move the robot.
10. Video Size - You can select how large an image you want to receive from the Rover. Most of the smartphone apps will use 320x240 but the Rover is capable of 640x480 which can easily be handled by a PC desktop/laptop/netbook.
11. IR Activated - The Rover comes with an IR light that can be turned on when in really dark locations (like your closet!). The checkbox will activate the IR light which will increase the image intensity. Note that this is an Infrared Light which means you will NOT see this light so watch the image to see this become activated.

Example

 [Click Here](#) to load a configuration to control the Rover robot using a joystick.

See Also

[WowWee Rovio](#)
[Erector Spykee](#)
[Xaxxon Oculus](#)

IRobot Create



The IRobot Create module provides an interface to the IRobot Create robotic platform. The module provides an interface to most of the Create's capabilities and allows you to control the robot and respond to sensor values on the robotics platform.

Note that for a truly mobile platform you will need a bluetooth or other serial wireless connectivity to the Create from a PC platform.

Interface

Variable	Current Value	Min/Max Limits
0 (pin 23)	0	0 128
1 (pin 22)	0	0 128
2 (pin 24)	0	0 128

Variable	Current Value
Play LED	<input type="checkbox"/>
Advance LED	<input type="checkbox"/>
Power LED Color	0
Power LED Intensity	0

Open Interface Mode	Full
COM Port	COM1

Stop Help OK Cancel

Instructions

1. COM Port - Select the appropriate communications serial port.
2. Motors - You can move the motors individually by moving the scroll bars or by changing the values in the editable text box. To automate the movement of the robot you can select or type in variables that contain values that will be sent to the robot as motor intensity values. Keep in mind the STOP button at the bottom of the interface to quickly stop an out of control Create! See [Variable Control](#) for more information on how to programatically move the robot.
3. Digital Outputs - If you connect devices to the 29 pin output port in the back of the Create you can turn the digital output pins off and on by selecting the checkboxes OR by specifying variables that when non-zero are meant to turn on that pin.
4. PWM Drivers - similar to the motor interfaces you can control the PWM pins in the cargo bay connector used to extend your Create.
5. Sensors - The create has many sensors (around 47 in all) that indicate various detected external and internal conditions of the create. To react to the sensors and use those values within the RoboRealm pipeline select the sensor name and type or select in a variable that will contain the value. You can see the current value in the "Current Value" text area. This data will be placed into the selected variable for use in other modules such as the VBScript module.
Check with your Create Open Interface for all the sensor meanings and descriptions.
6. Play Song - you can quickly select a song and press play to hear the Create play a quick melody. To automate the playback of songs select a variable that will contain the song name as seen in the dropdown. Note that after the song begins playing the variable is cleared to avoid repeating the song. To add your own songs you can edit the "Music.rtttl" file in the RoboRealm folder. This file contains RTTTL formatted melodies which are converted and sent to the Create for playback. Note that the RTTTL format is the Nokia Cell Phone ringtone format and can be found for free in many sites.
7. LEDs - to quickly indicate status you can use the provided LED's that are already on the Create. You can click on the checkbox or use the scroll bars to switch off/on the Create's power, advance and play LEDs.
8. OI Mode - to ensure that the robot provides some rudimentary safety reactions leave the OI Mode to Safe. If you need to move the robot in such a way that triggers a stop you can switch the OI mode to full.
9. OI Variable - should you wish to change the Create mode with a variable enter the name of that variable here. Then whenever that variable contains one of the values Off, Passive, Safe or Full the Create will be placed into that mode.

Example

This [robo-file](#) provides a keyboard interface configuration to steer your Create robot using the cursor keys. Press the spacebar to stop! Note that the robot motor limits are set to prevent any sudden jolt and keeps the robot speed very slow. See the [keyboard module](#) to see how to add other key commands.

Wireless Connectivity

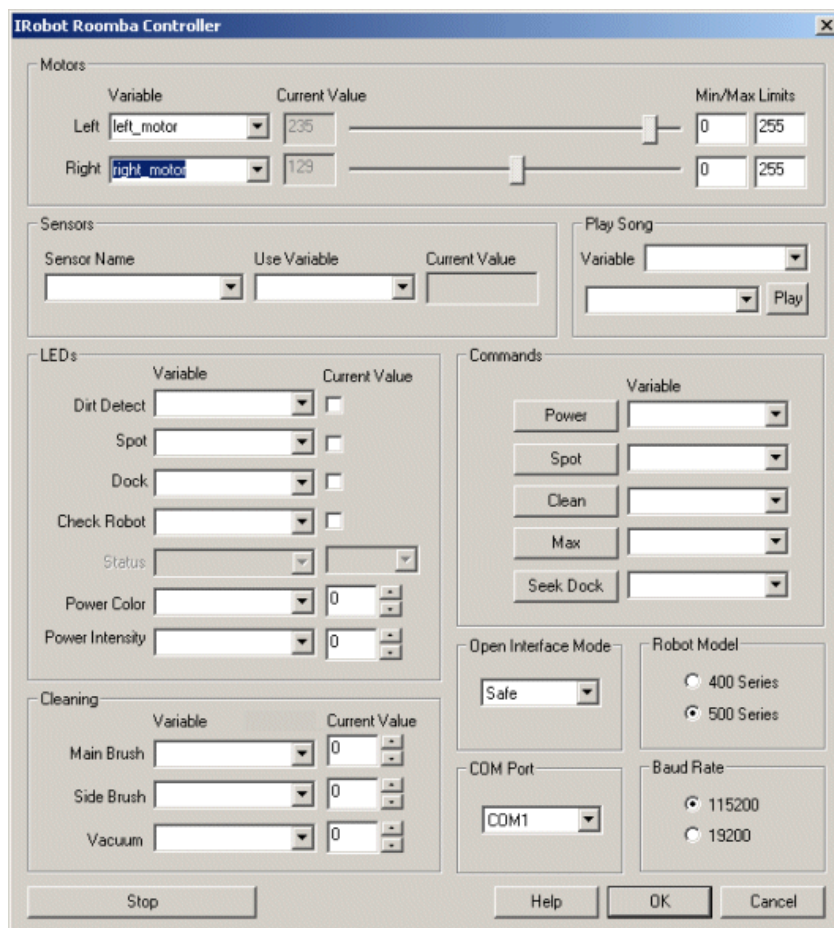
For those of you looking for a wireless solution to control the Create from a PC be sure to have a look at [Element Direct](#) and their [BAM Create Bluetooth](#). We've tested this out and it works very nicely with RoboRealm! Setup only takes about 10 minutes.

IRobot Roomba

The IRobot Roomba module provides an interface to the IRobot Roomba robotic vacuum platform. The module provides an interface to most of the Roomba's capabilities and allows you to control the robot and respond to sensor values on the platform.

Note that for a truly mobile platform you will need a bluetooth or other serial wireless connectivity to the Roomba from a PC platform. The serial connector from the PC to the Roomba requires a special cord and will not work with a regular serial cable. You will either need to create a cord yourself or purchase the [IRobot Create Serial cable](#) which has been tested and works with the Roomba 530. Other version have not been tested so check with IRobot for cable compatibility.

Interface



Instructions

1. Robot Model - Select the appropriate robot model. This is required as the communication speed and protocols are slightly different for the 400 versus 500 Roomba series.
2. COM Port - Select the appropriate communications serial port. This will normally be COM1-4 if you are using the Create serial cable. If you are using a USB to serial converter this will normally be above COM4.
3. Baud Rate - By default the 500 series is 115K and the 400 series is 57K.

4. Open Interface Mode - Select which mode you want to control the robot using. The OI mode allows you to place the robot in a safer mode that will not respond to commands that could potentially cause the robot to move uncontrollably. To ensure that the robot provides some rudimentary safety reactions leave the OI Mode to Safe. Once set, your Roomba Clean (green light) should disappear, i.e. when in program mode no LED's are on. This is a good chance to test the communication with the Roomba by switching on one or more LED's to check that communication has been established.

5. LEDs - to quickly indicate status you can use the provided LED's that are already on the Roomba. You can click on the checkbox or use the scroll bars to switch off/on the Roomba's power, advance and play LEDs.

6. Motors - You can move the motors individually by moving the scroll bars or by changing the values in the editable text box. To automate the movement of the robot you can select or type in variables that contain values that will be sent to the robot as motor intensity values. Keep in mind the STOP button at the bottom of the interface to quickly stop an out of control Roomba! See [Variable Control](#) for more information on how to programmatically move the robot.

7. Cleaning - similar to the motor interfaces you can control the cleaning motors of the Roomba by specifying non-zero values in the provided text boxes. Using the dropdown next to the values you can automate the control of the cleaning motors by setting the appropriate variables using other modules in RoboRealm.

8. Commands - to command Roomba to perform certain cleaning tasks or to power off you can manually click on the Command Buttons to switch in to docking mode, max cleaning, etc. By using the variable dropdown the command will execute when the provided variable has a non-zero value.

9. Sensors - The Roomba has many sensors (around 58 for 500 series) that indicate various detected external and internal conditions of the robot. To react to the sensors and use those values within the RoboRealm pipeline select the sensor name and type or select in a variable that will contain the value. You can see the current value in the "Current Value" text area. This data will be placed into the selected variable for use in other modules such as the VBScript module. To test this select the Bump Left or Right sensor from the long Sensor Name dropdown. Manually press the left or right bump sensor on the robot. The Current Value should change to 1 to indicate that it detected your press.

Check with your Roomba Open Interface documentation for all the sensor meanings, descriptions and range values.

10. Play Song - you can quickly select a song and press play to hear the Roomba play a quick melody. To automate the playback of songs select a variable that will contain the song name as seen in the dropdown. Note that after the song begins playing the variable is cleared to avoid repeating the song. To add your own songs you can edit the "Music.rtttl" file in the RoboRealm folder. This file contains RTTTL formatted melodies which are converted and sent to the Roomba for playback. Note that the RTTTL format is the Nokia Cell Phone ringtone format and can be found for free in many sites.

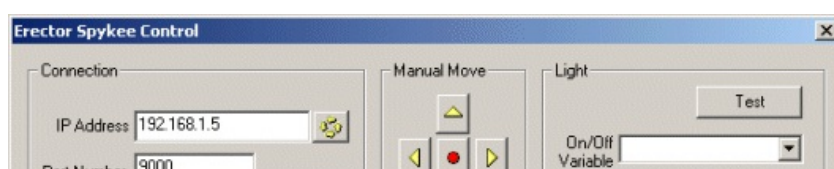
Trouble Tips

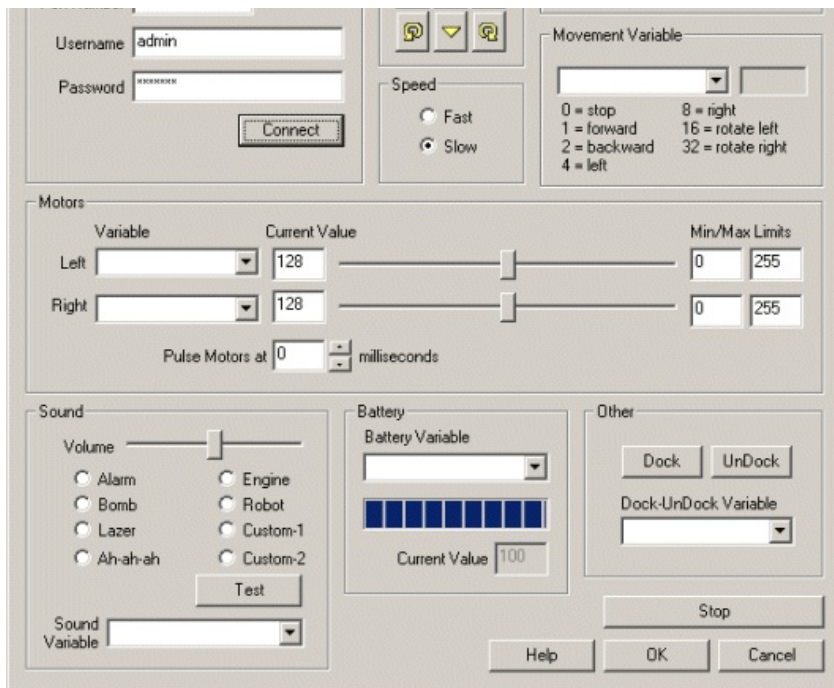
- The serial cable for the Roomba is normally located below the top face plate. You will have to pry this faceplate off (no screws need be loosened) in order to plug your serial cable into the Roomba.
- If the robot stops responding or does not respond try re-selecting or changing the OI mode. This can help to reset communication with the robot and regain control.
- The LEDs will be off when in program mode. Be sure to switch off the Roomba correctly by pressing the Power button after you are done experimenting otherwise the robot will remain on and run out of battery!
- Be sure that all hardware parts are functioning correctly before programming. If the dust collector is not correctly placed in the robot the cleaning motors may not activate.
- If all else fails try resetting the robot by simultaneously holding down the Spot and Dock buttons for 10 seconds. Then try re-selecting the OI mode.

Erector Spykee

The Spykee module provides an interface from RoboRealm to the Erector Spykee robot. Using the module you can command the robot to respond to images processed from the Spykee robot. The robot sends images over 802.11 (WiFi) to a PC. The PC is running RoboRealm that accesses that video stream, processes the video as per your needs and then sends the resulting motor commands back to the robot also using 802.11. Using this technique you can control your Spykee robot from your PC and use the full power and flexibility of your PC to control the Spykee robot. This allows you to extend the functionality of the Spykee robot beyond its original limitations.

Interface





Instructions

1. IP Address - specify the ip address of the Spykee. This can be determined by using the default address or by clicking on the yellow circular button to the right of the text box which will broadcast out to the network to see if a Spykee robot is connected. If this fails you can also check your wireless router to see what IP address has been assigned to the Spykee robot.
2. Username / Password - specify the username and password for the Spykee.
3. Connect - press the connect button to connect to the Spykee and start streaming video.
4. Manual Move - to test the motors on the Spykee you can use the Manual move buttons to move the robot in the specified direction. Note that the robot will continue to move for as long as the buttons are pressed. The middle button stops the robot.
5. Speed - to make the robot move quicker when using the manual move buttons you can select a faster speed. Note this speed adjustment are just used for the manual move buttons.
6. Motors - use the variable dropdown to select a variable that has a value that would be sent to the motors of the Spykee OR you can change the values manually by moving the provided sliders or by changing the value directly in the edit box. See [Variable Control](#) for more information on how to programmatically move the robot.
7. Pulse Speed - the Spykee robot is a very fast robot. The motors even at value 129 (+1 from off) will move the robot quicker than most would want when fine control is needed. You can use the pulse time to pulse the motor value to the robot in order to slow the motors down while still retaining enough torque to move the robot. This will cause the motor value being sent to the robot to oscillate causing a jerky but slow movement. The recommended value is about 50 but you will have to experiment on your Spykee to see what value works best. 0 means no pulsation is used.
8. Light - you can test the main Spykee light by pressing the Test button. If you want to control the light via a variable (zero or non-zero value) select that variable in the provided dropdown.
9. Movement Variable - instead of using the motor separate motor variables you can select a variable that contains one of the command numbers that will cause the robot to move in the specified manner. Note that this is used in lieu of the separate motor variables.
10. Sound - To test the default sound effects click on one of the effect radio buttons and press the Test button. This should cause the Spykee to produce that sound. You can adjust the volume of the sound by adjusting the slider bar left to lower the volume and right to increase it.
11. Sound variable - If you'd like programmatic control to produce a sound specify a variable that will contain a number from 1 to 8 or contain the name of the effect (Alarm, Bomb, etc). The effect will then be played on the Spykee robot. Note that the variable is cleared once played to ensure that the sound does not repeat.
12. Battery - The battery level is indicated by the progress bar and the actual current value. To programmatically react to the battery value select a variable that will contain the battery value. 0 - 100 means the robot is in operation and feeding video. -1 to -99 indicates the battery level while on the charger. Thus you can use this value to determine when charging is required and when it is complete.
13. Other - To dock or undock the robot press on the appropriate button. To programmatically dock or undock the robot select a variable that would contain either "dock" or "undock" (without the quotes) that would indicate what action the robot should perform.

Example

[Click Here](#) to load a configuration to control the Spykee robot using a joystick.

[Click Here](#) to load a configuration to connect to the Spykee robot using the default parameters to chase a red ball. The optimal configuration of the Spykee camera is to move it such that it sees the floor immediately in front of the robot.

Note that for the examples to work you will need to configure the IP address and the username/password within the Spykee module.

See Also

[BrookStone Rover 1.0](#)

[WowWee Rovio](#)

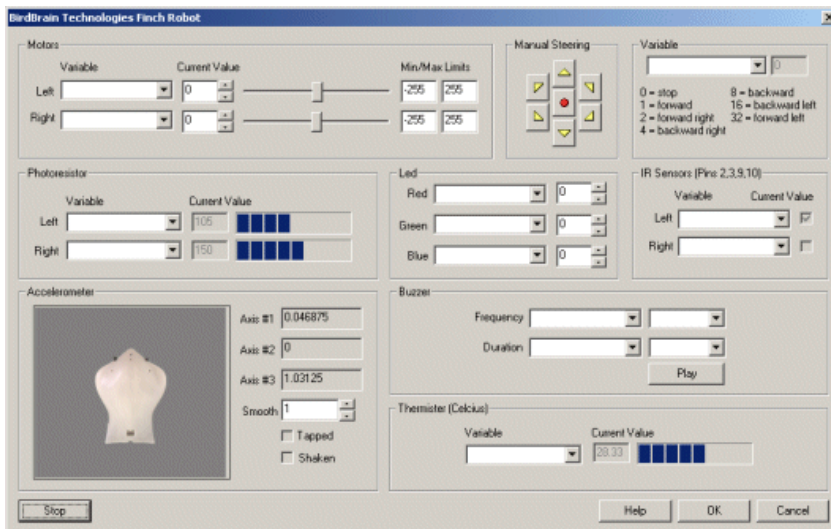
[Xaxxon Oculus](#)

Finch Robot



The Finch Robot module provides an interface to the BirdBrain Technologies Finch Robot from RoboRealm. This \$99 dollar robot has 2 servo motors, 2 photoresistors (light sensors), 2 IR distance sensors, a thermister, buzzer, 3 axis accelerometer and a multi-colored LED. This module provides access to all the robot's features from a GUI and allows you to combine them with the additional capabilities of a camera.

Interface



Instructions

1. You should have already plugged in your Finch robot otherwise the module will indicate that it cannot find the robot. If you see this popup warning ensure that your Finch is plugged in and cycling the LED nose colors prior to activating the module. Once activated the Finch's LED nose should be unlit. This indicates that the Finch has power and is waiting on commands.
2. Manual Steering - There are two ways to move the finch. The manual steering buttons will move the robot as long as the button is pressed. Releasing the button will stop the robot moving in a particular direction.
3. Variable - If you want to automate the movement of the robot you can select a variable in the dropdown menu that contains or will contain which direction to move the robot. Note that this variable will be populated by some other module. It offers a simple way to direct the robot without needing to worry about what wheel needs to do what in order to generate a particular movement. See [Variable Control](#) for more information on how to programatically move the robot.
4. Motors - To independently control the wheels you can move the scroll bars or change the values in the editable text box. To automate the movement of the robot you can select or type in variables that contain values that will be sent to the robot as motor intensity values. Keep in mind the STOP button at the bottom of the interface to quickly stop an out of control robot!

Once you select a variable you will not be able to manually adjust the motor bar as it will now respond to the value within the variable. Also be sure to select the appropriate minimum and maximum values to ensure that the variables do not exceed the motor limits. Note that the values for the motors range from 0 to 255 with 128 being neutral.

5. Photo Resistors - you should start see the bar graph adjust based on the amount of detect light. To utilize this number in other modules select or type a variable that will be set to the resistance amount. Keep in mind that any ambient light will be detected by the resistor including sunlight or

spotlights.

You can notice the change in the value by covering both your hands over the finch's nose. You should see the values decrease and increase when you release your hands.

6. LED - The led is a full color LED capable of generating a lot of colors. You can use the spin buttons or type in a number in the provided text boxes to see an immediate change in the nose LED. The 3 channels (red, green and blue) can be used in combination to specify many colors. Note that when a variable is selected the value field will be disabled to indicate that the specific LED channel value is now being received from a variable.

7. IR Sensors - The appropriate checkbox will turn on when something is in front of the IR sensor. If you move your hand in front of the robot you should see one or both checks in the checkboxes. Note that the IR sensor used is a digital sensor and detects an object at a set distance. The dropdown combo box next to the checkboxes can be used to specify a variable that will contain the value of 1 or 0 depending on if something is detected by the specific IR sensor.

8. Buzzer - to test that the piezo buzzer is working correctly select a frequency from the second dropdown menu and a appropriate during in the next dropdown directly below. Click on Play and you should hear the note coming from the robot.

To automatically play a sound type in or select a variable for both frequency and duration that will contain the appropriate frequency and duration number. When these variables become set RoboRealm will command the robot to play the note and will then remove the variables' values. The variables are cleared to prevent the note from being played again and again. To once again play a note, simply set the variables again with appropriate values.

9. Thermister - The temperature value is shown in the bar chart with the actual value in Celsius in the disabled text box. You can specify a variable in the combo box that will contain this value as sent back from the Finch.


10 Accelerometer - The 3 axis accelerometer provides position information for the Finch. The X and Y values indicate roll and tilt with the Z axis being a check to determine if the device is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (Z axis). The values are stored in 3 variables that can be accessed by other modules (FINCH_AXIS_X, FINCH_AXIS_Y, FINCH_AXIS_Z).

Note that the simulation graphic uses Axis1 and Axis2 for tilt sensing and checks Axis 3 to determine if the device is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (Z axis) and thus the simulation graphic will assume you are holding the robot with the USB cable connection oriented towards you and the nose pointing away.

11. Smooth - if the accelerometer values are not as responsive as you would like try reducing the smooth value. This will allow the numbers to be much more reactive but may also cause some unwanted jitter due to noise in the determined values. By decreasing the smooth value you make the sensors much more responsive to movement, increasing the value will delay the sharpness of the movement but also make it much more smooth. Note this is reflected in the simulation graphic.

Examples

Joystick


 [Click Here](#) to load a robofile that will allow you to drive your Finch using a joystick. When you load this configuration you will need to edit the Joystick module to select which joystick to use to control the robot.

This robofile is configured for a playstation type of joystick with the first knob providing motion control. Moving the knob forward/backward moves the robot forward/backward. Moving the knob left/right pivots the robot in the appropriate direction.

If you do not like the configuration we have chosen simply edit the Joystick module and change the variables around to different buttons.

Once that is done have a safe and happy drive!

Color Sample

 [Click Here](#) to load a robofile that will change the nose LED color based on the sample within the current image in RoboRealm. By moving the camera around you can change the nose color to the average color seen in the camera.

Variables

FINCH_AXIS_X - the X value of the accelerometer

FINCH_AXIS_Y - the Y value of the accelerometer

FINCH_AXIS_Z - the Z value of the accelerometer

FINCH_TAPPED - true (non-zero) when the finch was tapped

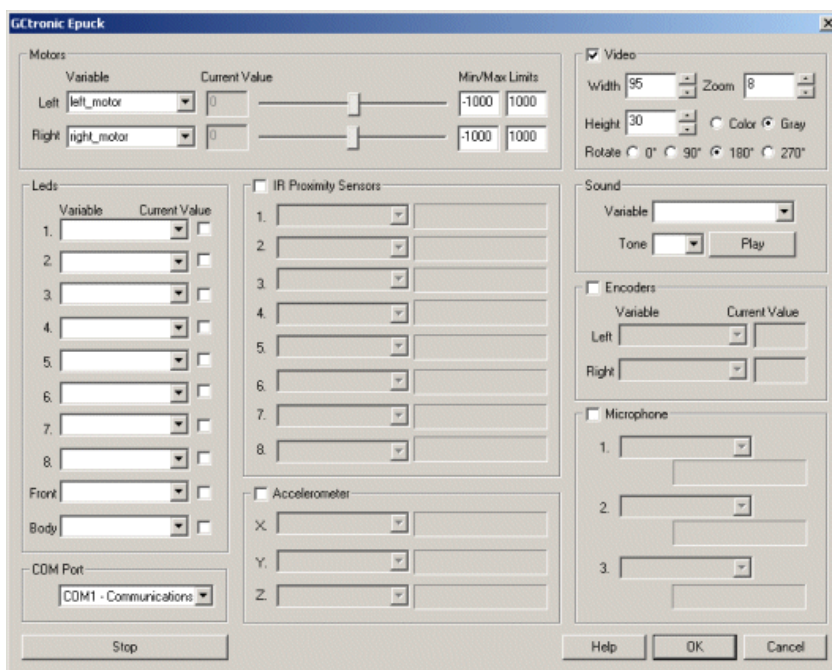
GCtronic e-puck



The GCtronic e-puck module provides an easy way to interface RoboRealm to the GCtronic e-puck robot sold in the USA by [RoadNarrows Robotics](#). The E-puck has several sensors and is an ideal robot for small desktop based research and experimentation. More about this robot can be read from the [RoadNarrows website](#).

The e-puck provides for on-board robot programming. For RoboRealm to interface correctly to the e-puck you will have to have installed the [GCtronic demo](#) installed on the robot with the selector (the white turnable knob on the top of the robot) turned to 3 (or the dot inbetween 2 and 4). Turning the selector to 3 runs the Advanced Serial Communication program. This application on the e-puck is commonly used with the [e-puck monitor program](#). The RoboRealm module uses the same serial interface as the e-puck monitor to bring control of the e-puck within access of other RoboRealm modules.

Interface



Instructions

1. COM Port - First select the appropriate COM port where the bluetooth has been attached to. Note that only the COM ports that are recognized by the system are displayed. If you have not yet attached your bluetooth USB device or activated the internal bluetooth wireless capabilities you should do so before running RoboRealm otherwise the appropriate COM port may not show up.

If you do not know the COM port associated with your bluetooth device check your task bar for the blue icon that represents the bluetooth connection and investigate the properties of the epuck_XXXX connection to see which serial port is attached to the e-puck.

Once RoboRealm has connected to the e-puck successfully the yellow/orange LED will be on for as long as the connection is held.

2. Motors - To test the module slowly move the appropriate motor sliders. If everything is working the motors should slowly start to move. If this does not happen check that a bluetooth connection is made and check your COM port settings, or that your bluetooth device is on. Note that when first connecting the devices can take up to 30 seconds to secure a connection so be patient. Be on the lookout for the pairing password prompt that may pop up during the initial communication between your PC and the e-puck. Note that the pairing password format is indicated on the provided in the [mini-doc](#).

If the motors move you can then select a variable (or type one in) that will be populated from another module that will contain the values to send to the e-puck. Note that 0 is considered neutral whilst -1000 and 1000 are the opposite extremes.

See [Variable Control](#) for more information on how to programatically move the robot.

3. Video - To enable the video stream from the e-puck select the checkbox next to the "Video" text. This will tell RoboRealm to start requesting the image from the e-puck's camera. Note that there are several controls available that allow you to customize the type of image being returned by the

e-puck. This is to allow you to configure the image to your project requirements. As the e-puck uses bluetooth to transmit the image there is a severe limitation on the size and number of frames a second you can receive from the e-puck.

Width/Height - Use the text area to specify the width and height for the image to be sent. The maximum number of pixels sent per image is 3200 bytes. Thus you can change the width and height as desired but need to keep the size limited below that amount. If this amount is exceeded the image will default back to a 40x40 color mode. Note that increasing the width and height using the provided spin buttons will ensure that the total size is below this requirement by adjusting the parameters as possible.

Zoom - If you want a closer look at the object within view of the robot you can select a lower zoom value (1-8) that will cause the e-puck camera to digitally zoom into the captured image. Note that as the e-puck's camera is actually capturing images at a 640x480 resolution the zoom is not an interpolated view but in fact the unsampled pixels of the actual image captured. Thus the zoom preserves the quality of the original detailed image.

Color/Gray - Selecting Gray reduces the pixel buffer requirements by 2x (color takes 2 bytes) which will transmit the images quicker over bluetooth and achieve a faster rate. You can either remain with the faster rate or increase the image size to achieve the same rate but with a larger resulting image. If your project requires just intensity levels then using the gray option will increase the frames per second and provide you with a higher intensity resolution as the color option only provides up to 64 gray levels while the gray mode provides 256.

Rotate - Depending on the configuration of your e-puck you may need to rotate the image. Select the appropriate radio button to perform that action.

4. **LEDs** - You can switch on or off any of the robot LEDs by clicking on the checkbox next to the LED you wish to turn on or off. This manual method is useful to check which LED is which on the robot. To automatically switch LEDs on/off chose a variable in the dropdown (or type one in) that will contain a non-zero value when the LED should be switched on. Variables can then be changed by other modules such as the [Set_Variable](#) or [VBScript](#) or can be changed by numerous other methods including the [API](#).

5. **IR Proximity Sensors** - Enable this sensor readout by clicking on the checkbox next to the "IR Proximity Sensors" text. This will tell RoboRealm to start querying those sensor values from the e-puck. If you move your hand in front of the IR sensor you should see the progress bars indicate the proximity of your hand. Higher values indicate closer objects. Note that the IR sensors are triggered by very close objects. In order to trigger the highest value the object will most likely be touching the robot in that case.

If you want to access the IR value from other modules select or type in the variable name that will contain a value ranging from 0 to approximately 3850 to indicate the proximity of an object.

6. **Accelerometer** - Similar to the IR proximity interface you can enable the accelerometer sensors which will start feeding those values from the e-puck into RoboRealm. To capture those values into a variable for use elsewhere in the RoboRealm application you can specify a variable using the dropdown (or type one in) next to the sensor readout. Note that these are the raw values from the accelerometers and have not been converted to angular measurements.

7. **Sound** - to test that the speaker on the e-puck is working select a Tone from the second dropdown menu and click on Play. You should hear a sound coming from the e-puck.

To automatically play a sound type in or select a variable that will contain the number of the tone to play. When this variable becomes set RoboRealm will command the e-puck to play the tone and will then remove the variable's value. The variable is cleared to prevent the note from being played again and again. To once again play a note, simply set the variable again with appropriate value.


8. **Encoders** - To read the values of the wheel encoders switch on the interface by selecting the checkbox next to the "Encoders" text. You can then specify variables that will contain the encoder count for each wheel as read from the e-puck. Encoder values will increase when the motors are driver forward and decrease (below zero in some cases) when moving backwards.


9. **Microphone** - There are 3 microphones on the e-puck that indicate that can be used to check the volume of sound nearby the robot. To access these volume levels you can switch on this interface by checking the checkbox next to the "Microphone" text. This will cause RoboRealm to show the volume levels of each microphone in the progress bars. The specific values of the microphones can then be placed into variables by specifying the variable in the provided dropdown.

10. **Stop** - The stop button is provided to stop the e-puck's motors and terminate communication to the e-puck in an attempt to stop the robot as fast as possible. Pressing the button again will resume communication with the robot.

Example

For these examples to run you must have the selector (the white turnable knob on the top of the robot) turned to 3 (advanced serial interface).

 [Click Here](#) to load a robofile that will allow you to drive your e-puck using a joystick. When you load this configuration you will need to edit the e-puck module (first module) by double clicking on it to change the COM port to the correct port. This will be the port your bluetooth device has chosen for communication with your e-puck. Second, you will need to edit the Joystick module to select which joystick to use to control the robot. Once that is done have a safe and happy drive!

 [Click Here](#) to load a robofile that will have the e-puck chase after a bullseye target using vision. Note that you will have to edit the e-puck module (first module) by double clicking on it to change the COM port to the correct port. Then set the e-puck within view of a target like



(you can print this to about a 2x2 inch size) and see how the robot reacts. Careful, as the image update speed is very slow AND you only have a 40x40 image size to work with you will need to move the target VERY slowly otherwise the target will seem to just disappear from the robots view. To see a larger image in RoboRealm select a 400% zoom from the main GUI interface.

The robot is programmed to align with the target and then move forwards or backwards depending on how far or close the robot is to the target. Slowly moving the target should incur the appropriate movement from the robot.

Lego

The Lego module provides an interface to the Lego Mindstorms Robot Kit ver 1.1 and 2.0. The module allows you to control the motors, read the sensors and interact with onboard programs using the variable set/get interface. RoboRealm uses the Phantom.dll to facilitate communication between the PC and Lego Mindstorm IR Tower. Note that the RCX brick needs to be within transmission proximity of the IR tower for communication to be successful. While this does impose a range limitation on the usage of RoboRealm and the RCX this limitation will soon be reduced due to use of Bluetooth and other wireless technology.

Note that in order to use the Lego control module RoboRealm will need access to the Phantom.dll file that is included in the RoboRealm download. If this module complains about not finding the Phantom.dll please copy that file from your RoboRealm installation folder (normally c:\program files\RoboRealm) into your system folder (normally c:\windows\system32).

Interface

A screenshot of the 'Lego Mindstorm' control window. The window has a title bar with the text 'Lego Mindstorm' and a close button. The main area is divided into several sections: 1. 'Motors (Output) 1-3': A checkbox is checked. Below it are three rows for Motor # 1, 2, and 3. Each row has a dropdown menu for the motor name (IMAGE_COUNT, IMAGE_HEIGHT, and an empty one), a 'Current Value' input box, a slider, and 'Min/Max Limits' input boxes (set to -7 and 7). 2. 'Sensors (Input) 1-3': Three rows for Sensor # 1, 2, and 3, each with a dropdown menu and a 'Current Value' input box. 3. 'Sound (Output)': A dropdown menu and six radio buttons labeled 'Sweep Up', 'Beep', 'Sweep Down', 'Click', 'Fast Sweep', and 'Error'. 4. 'Connection Strength (Input)': A dropdown menu. 5. 'Variables': Seven rows, each with a dropdown menu and radio buttons for 'get' and 'set'. 6. 'Options': 'COM Port' dropdown (set to USB) and 'Connection Type' dropdown (set to IR). 7. 'Battery (Input)': A dropdown menu. At the bottom are buttons for 'Stop', 'Help', 'OK', and 'Cancel'.

Instructions

Motors

1. Ensure that the checkbox is selected in the upper left corner of the interface. If this checkbox is not selected the motors will not be set. Unchecking this box will allow an onboard program to modify the motor values without disturbance from RoboRealm.
2. Use the moveable bar or the current value edit box to change the value of the motor.
3. To allow RoboRealm to control the motor values you can select an existing RoboRealm variable that contains the value to be send to the RCX motors. Note that the motor value range is from -7 to 7. Note that you can also type in a new RoboRealm variable that you intend to use but have not yet created.

Sensors

1. To read the sensor values from the RCX select the appropriate variable that should receive the sensor value. These variable can then be used in other RoboRealm modules. You can also type in a new RoboRealm variable that will be used in other modules yet to be added to the processing pipeline.

Sound

1. You can test the sound values on the RCX by clicking on one of the sound radio buttons. The sound should play on the RCX and then unselect your selection. To allow RoboRealm to control the sound played select a variable that will contain a number from 0 to 5 which will play the appropriate sound.

Connection Strength

1. If the RCX is moved out of range of the IR tower communication will cease to happen between RoboRealm and the RCX. The connection strength input can be used to determine when this has happened. By selecting a variable to receive the connection strength number (0-99) you can detect when the RCX box goes out of range. 0 indicates no connection, 100 indicates good connection.

Variables

1. To communicate with internal RCX programs you can use map RCX variables to RoboRealm variables. To reduce communication overhead you need to specify if the variable should be written to or read from the RCX. To send a variable to RCX use the 'set' radio button. To get a variable from the RCX brick select the 'get' radio button. Note that even though 32 variables are possible in the RCX brick the interface allows access to only the first 8. There is also a scaling by a factor of 10 when moving variables in and out of the RCX. When variables are moved out of the RCX they are multiplied by 10 and when moving back in they are divided by 10. We believe that this is due to how the RIS handles decimal variables.

Options

1. Select the appropriate communication port. Typically for Mindstorms 1.1 you will select one of the COM ports, for Mindstorms 2.0 you will select USB. This relates to the type of connection your PC has to the IR tower that will communicate with the RCX brick.
2. Select the tower connect type.

Battery

1. Specify the variable that should accept the battery value from the RCX. You can use this value to detect when the RCX battery is getting low.

Other

1. If you want to quickly stop the motors press the 'stop' button. This will deactivate the RCX motors until the 'start' button is pressed.

Downloads

You will need to download the Phantom DLL (no longer available!) in order for RoboRealm to be able to communicate with the Lego RCX control.

See Also

[SSC](#)
[Parallax](#)

Lego NXT

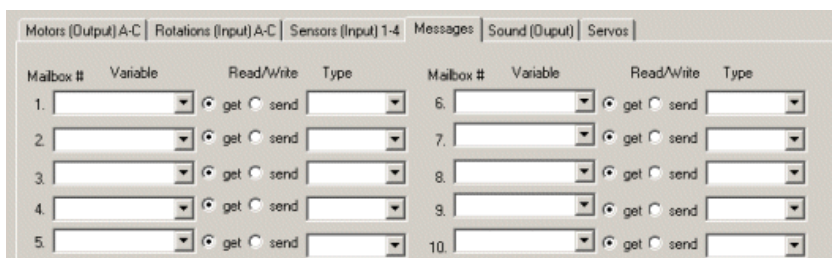
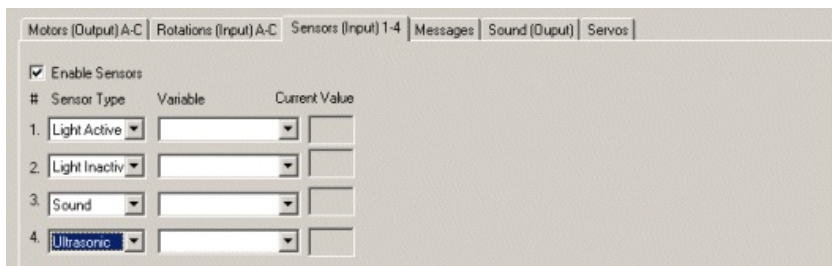
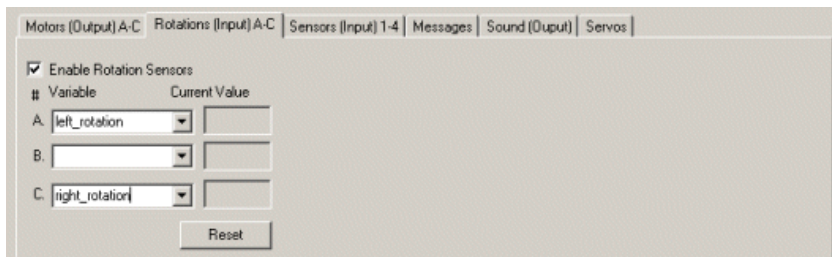
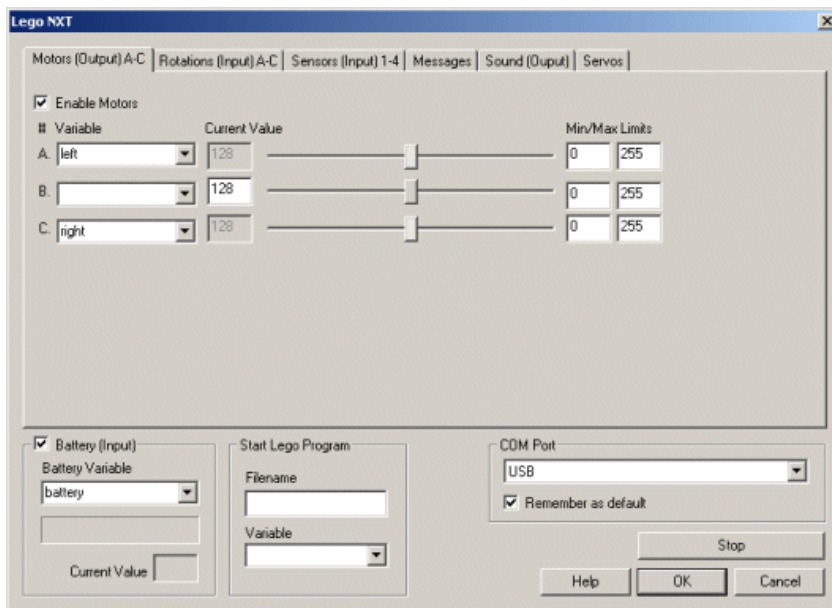
The Lego NXT module allows you to control your [Lego NXT Mindstorms Robot Kit](#) using RoboRealm over a wireless Bluetooth link. Controlling the NXT kit from RoboRealm provides you a way to incorporate vision into your Lego robot. The module allows you to control the motors and read the sensors on board the NXT. RoboRealm uses the Bluetooth serial connection to communicate directly with the NXT platform. Note that the NXT brick needs to be within transmission proximity of you PC for communication to be successful (~10 meters).

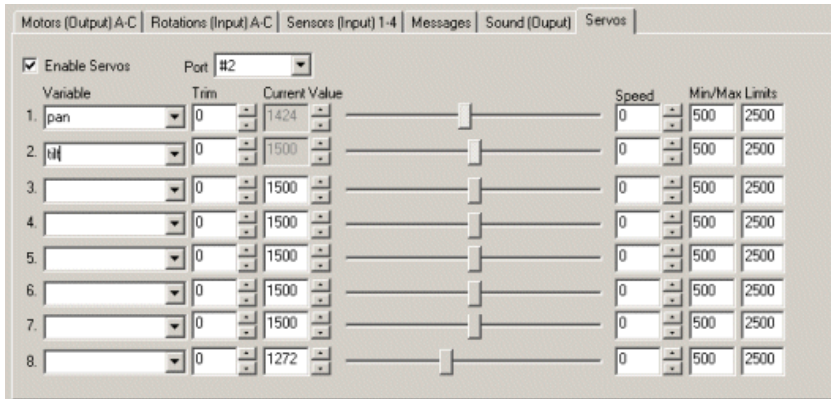


The main difficulty in working with the Lego NXT system is to get the Bluetooth serial connection up and running. In our experience we had no luck using the provided drivers that came with our USB based Bluetooth device. Using only WindowsXP native drivers (after deleting the provided drivers) did the Bluetooth device work in such a way that the NXT brick could detect our connection. We highly recommend working with the provided Lego Mindstorms NXT program to first create a connection to your NXT brick. After the connection has been established you can quit the program and use RoboRealm to directly communicate with the robot.

Note you must NOT be running the NXT-G application when trying to connect to the Lego NXT using RoboRealm. Both applications cannot connect simultaneously to the robot. They can do so one at a time but not together.

Interface





Instructions

1. Click on the appropriate checkbox to enable the functionality that you wish to use in the Lego NXT. Enabling a particular section will start the correspondence between your PC and the NXT brick. This will increase the Bluetooth traffic and can slow down the reaction time that your Lego NXT will take so be sure to only enable those parts that you need.
2. Motors - The motor control variables can be specified by either using one of the existing variables provided in the dropdown list or by typing in a new variable. Once you select a variable you will not be able to manually adjust the motor bar as it will now respond to the value within the variable. Be sure to select the appropriate minimum and maximum values to ensure that the variables do not exceed the motor limits. Note that the values for the motors range from 0 to 255 with 128 being neutral. See [Variable Control](#) for more information on how to programatically move the robot.
3. Rotations - If you would like to receive information about the rotation of the motor (aka encoder values) then select the rotation checkbox and specify the variables that should contain the values sent by the encoders. If you enable this area and then move the motor by hand you will notice the value field change in accordance to the rotation of the motor.
4. Sensors - The NXT has several different kinds of sensors. You need to let RoboRealm know which kind is attached to which port. To do so simply select the appropriate dropdown to indicate the type of sensor attached to the port. Then select the variable or type in a new one that is meant to receive the value of that sensor. Be sure to click on the checkbox above the Sensor group to enable reading of sensors. Note that the current sensor value will be shown in the right hand box.
5. Sound - To control the playback on the NXT of a sound click the checkbox in the sound group. You can either specify a frequency and duration OR a sound file to play on the NXT depending on the values of the provided variables. Note that the sound files must exist on the NXT. If you wish to play other sound files be sure to first download them to the NXT and they can then be played. Note that the variables contain the values to be played as apposed to the sensors where the variables are used to store values FROM the NXT.
6. Battery - You can keep a check on the battery life of the NXT by enabling the battery checkbox. Adding in a variable will place the current battery power rating into that variable for use by other modules such as the VBScript module.
7. Messages - To communicate to a running program on the Lego NXT you use mailbox communication. To send a message to a particular mailbox on the NXT specify the variable that contains the message in the appropriate mailbox list seen in the right hand side of the above interface. Be sure to select "send" if you want to send a message to a mail box on the NXT. If you want to receive a message from a mailbox on the NXT select the "get" radio button. The value will then be placed into that RoboRealm variable and the message will be removed from the NXT. In this way you can use the Send and Receive Messages from the NXT-G Block interface to create a program that runs on the NXT and communicates back to RoboRealm running on a PC.
8. Servos - To extend the Lego NXT capabilities you can add extra servos using the MindSensor's [NXTServo](#) board. This board allows you to

add 8 additional traditional servos to the Lego NXT for use in providing additional control over your robot. For example, the board + servos are a great way to add in a pan/tilt unit for a camera.

To enable the board click on the "Enable Servos" and select which Sensor port you will be connecting the NXTServo board to. Note that you must NOT connect the board to a motor port otherwise this may damage the board.

Note that you CANNOT run the NXTServo board while connected via USB to the PC. You must run any servo based application over bluetooth in order for the PC <-> NXT connection to work correctly using the NXTServo I2C interface.

Once the board is connected you can select a variable that will contain a value for the servo (500 to 2500 with 1500 being neutral or middle). This provides an automatic way to move the servos. Use the Trim field to center the servo if your servo when neutral is not quite in the middle. Use the Value field to manually specify a value for the servo or use the slider to accomplish this same task. This is a good way to test the servo prior to using a variable to specify the desired position. Use the Speed field to control how quickly the servo will move to the desired position. 0 means ASAP, whereas 1 means really slow with 255 being just as fast as 0. Finally use the Min and Max fields to limit how far the servo can move. This can be used as an insurance against the servo moving too far out of limits if your project requires a limited servo movement.

9. Start Lego Program - If you would like RoboRealm to automatically run a program on the Lego NXT specify the program name in the Filename text field. If you prefer to use a variable to contain the value of the program to run select or specify that variable. The module will then look at the value of that variable as the filename to run. Note that changing the filename or setting it to blank will terminate any previously running program.

10. COM - Finally specify the port number that the Bluetooth serial communication port is setup in. This port number will be reported in your Bluetooth configuration screen that depends on what Bluetooth hardware device you are using. An alternative to a Bluetooth connection you can select the USB option (first option in the dropdown) which requires that the robot be tethered via a USB cable to the PC. This mode provides a quick and reliable way to keep you robot connected to the PC if you feel that Bluetooth is not functioning correctly or you are operating in a noisy Bluetooth environment. Please note that messages will NOT be sent when connected via USB; messages are only sent when connected via Bluetooth.

11. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the Lego NXT module is loaded the com port will be auto-selected. This ability allows you to not have to constantly change the COM port when loading in successive RoboRealm NXT configurations.

12. Stop - If at any time you would like to STOP the robot press the stop button (this can save many crashes). Pressing start will enable the motors and reactivate communication with the NXT robot.

Communicating with NXT G-Programs

You can use RoboRealm to communicate specific values from the PC to a running program within the Lego NXT brick. The reason one would do this would be to communicate a specific condition that RoboRealm has picked up on (for example seeing a large red cup) so that the G-Block program can change its behavior based on this information. In this manner RoboRealm is acting like an additional sensor that can be tested for specific conditions.

Receiving


The way this communication occurs is by using the Lego NXT Mailboxes. These are akin to the "Variables" known in the Lego Mindstorms RCX system (the former Lego system). You will need to add in the G-Block "Receive Message" (yellow square menu) to receive messages sent from RoboRealm. In this Block you will have to specify the type of data to be received and which mailbox the message will be received in. The mailbox number MUST correlate with the same mailbox number seen in the mailbox list on the right side of the RoboRealm Lego NXT interface. If these numbers do not match then no values will be received in the correct mailbox. In the RoboRealm interface, specify a variable name (NOT TEXT) in the dropdown/text editable box. RoboRealm will only transmit the contents of variables so be sure to specify a variable that has some content in it. If you are just testing the communication you can select the IMAGE_COUNT variable which will increment by one for each image being processed. Also be sure to select "send" radio button in the RoboRealm interface so that RoboRealm knows to send the contents of the specified variable to the Lego NXT.

Sending

To send a value from the Lego NXT back to RoboRealm you can add in the "Send Message" G-Block within the Lego programming interface. You MUST specify the 0 Connection in that G-Block's options. 0 means that when the NXT is acting in slave mode it should buffer the sent messages in outgoing mailboxes that the PC can request the contents of. You can then type in the message and the mailbox that should be used to send the message. Again, the mailbox MUST match one of the mailboxes in RoboRealm and a variable MUST be specified in the RoboRealm interface that will receive this value from the NXT. Note that your NXT must be in slave mode with respect to the PC for this to work. In other words, allow your PC to first connect to the NXT rather than use the NXT to connect to the PC. Since the PC is not another NXT the connection option needs to be set to 0 which allows for master/slave communication.


Examples

Joystick Example

 [Click Here](#) to download a robofile that you can use a joystick connected to a PC to control the NXT.

- Check that the joystick you use moves the joystick graphic above where the joy_x and joy_y variables are specified in the Joystick interface. Double click on that in the pipeline to edit the module after loading this robofile into RoboRealm.
- Check in the NXT module that the left and right motors are in the right channels. This robofile assumes Motor A is the left motor and Motor B is the right motor.
- Finally, if you find that the robot moves right when left is expected or back when forward is expected, select the appropriate checkboxes in the Differential Drive module. Depending on the values of your joystick inverting the signal to its opposite meaning sometimes provides a more natural feel.

Keyboard Example

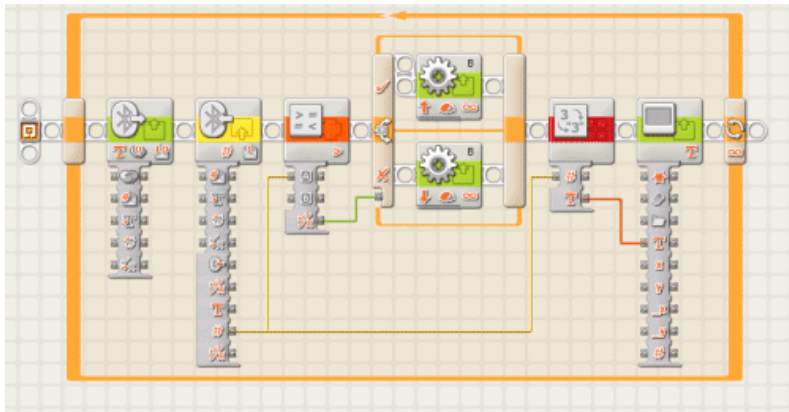
 [Click Here](#) to download a robofile that you can use a joystick connected to a PC to control the NXT.

This assumes that Motor B is the right motor and Motor C is the left motor. If not, edit the NXT module and swap those variables.

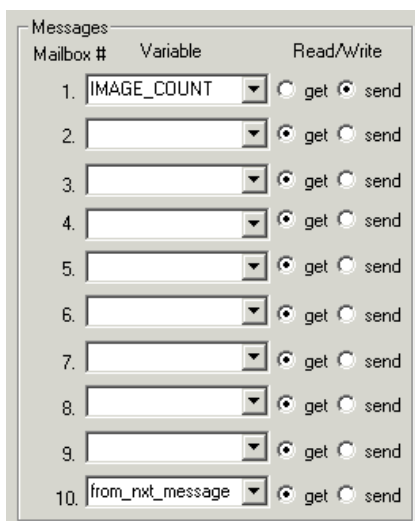
Use the keyboard right and left cursor keys to steer the robot. Use forward and backward to move forward and backwards. Use the space key to stop.

Mailbox Example

Following is the G-Block screen shot of an example of this bidirectional communication. It shows how you can use a message from the PC (COG_X) to control the movement of motors. (click image to enlarge):



And here is the corresponding RoboRealm interface:



You can [download the G-Block example](#) and the [RoboRealm example](#). Note that our NXT BT interface is configured on COM17. Yours will most likely be different.

Mailbox #2

Here is a [simpler G-Block example](#) that just displays the number from the [RoboRealm robofile](#) that the set_variable module creates. You can use this to test if your NXT is correctly receiving numbers from RoboRealm. Note that this robofile uses the USB connection which should be the same if you connect your NXT via USB. This also eliminates any Bluetooth connection issues. Also note that mailbox #1 is used. You should be able to double click on the Set_Variable module, change the value to some other number and see that value on your NXT screen.

See Also

[Lego RIS2.0](#)

Example

[Ball Picker Tutorial](#)

Parallax Boe-Bot

The Boe-Bot is one of [Parallax's](#) most popular products. With many add-ons and applications available it also proves to be a very versatile product and is used widely in academic, research and hobbyist communities.

The Parallax Boe-Bot module provides control of the Parallax Boe-Bot from RoboRealm. The module provides you with a way to send commands to the Boe-Bot based on variables created within RoboRealm.

This module makes three assumptions:

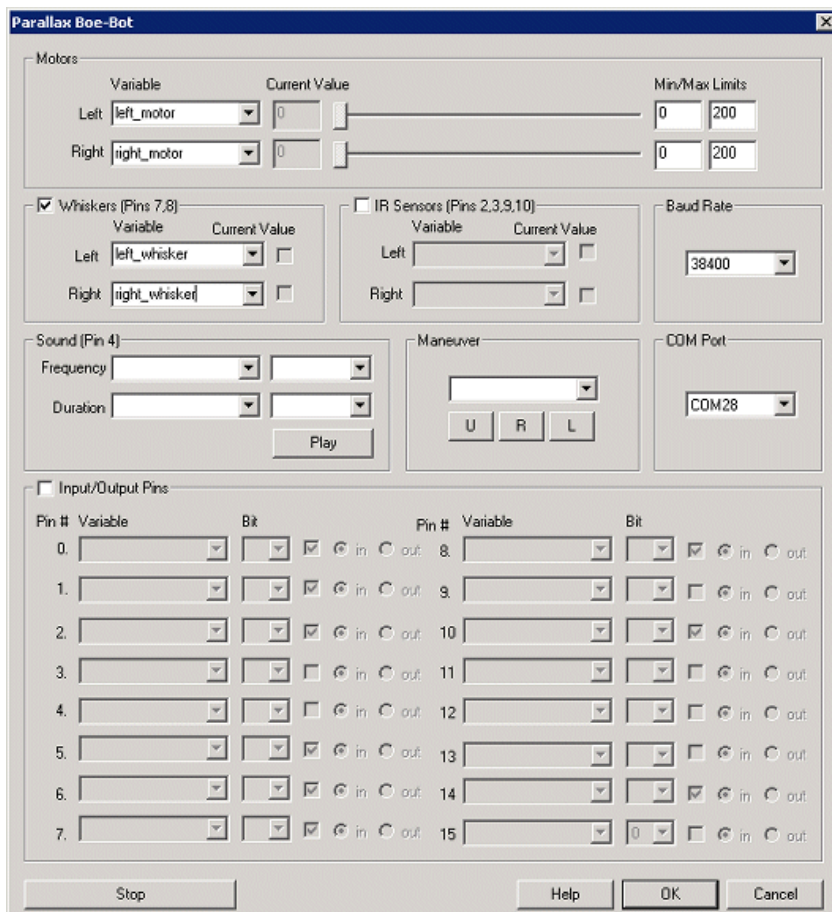
1. You have loaded the BS2 [BoeBotControl.bs2](#) into the Boe-Bot using the Parallax Basic Stamp Editor.
2. You are using the easyBT (or older eb500) wireless bluetooth connection to the Boe-Bot. If you are using the older eb500 interface card you will need to switch the pin configuration at the top of the bs2 program referenced in assumption #1.
3. You have finished the Whisker, IR, Speaker (Piezo Buzzer), constructions on the Boe-Bot breadboard.

While RoboRealm can still communicate with the BoeBot connected directly to the Boe-Bot is it assumed that for most of the interesting applications of vision and the Boe-Bot one expects the robot to be wireless and free from constraints.

If you prefer to create your own Boe-Bot communication protocol you can use the [Serial Module](#) to send custom commands to the Boe-Bot of your own choice. See the [Boe-Bot forum discussion](#) on how that can be accomplished.

The provided Boe-Bot module will allow you to quickly get RoboRealm and the Boe-Bot communicating. Note that as the communication between the PC and the Boe-Bot increases you will notice a decrease in processing frame rate. This reduction in frame rate is due to bluetooth wireless delays that are normal in most bluetooth devices. To keep the frame rate usable be sure to turn off any non-essential communications between the Boe-Bot and the PC. This is accomplished by un-checking the checkboxes next to the appropriate groups.

Interface



Instructions

1. COM Port - First select the appropriate COM port where the easyBT has attached to. Note that only the COM ports that are recognized by the system are displayed. If you have not yet attached your bluetooth USB device or activated the internal bluetooth wireless capabilities you should do so before running RoboRealm otherwise the appropriate COM port will not show up.

2. Baud Rate - Select the appropriate baud rate. It is best to use 38400 if your bluetooth device can support that baud rate as this will ensure minimal communication slowdown on the processing framerate.

3. Motors - To test the connection slowly move the Left and Right motor sliders. If everything is working the motors should slowly start to move. If this does not happen check that the little led in the upper right corner of the easyBT is on. If not check your COM port settings, or that your bluetooth device is on. Note that when first connecting the devices can take up to 30 seconds to secure a connection so be patient.

If the motors work you can then select a variable (or type one in) that will be populated from another module that will contain the motor values to send to the Boe-Bot. Note that 100 is considered neutral whilst 0 and 200 are the opposite extremes.

If you motors move even when the value is at 100 (neutral) you may need to use the trim values to offset the amount until they stop moving. This amount is automatically added to values sent to the robot in order to keep both servos at the same speed.

See [Variable Control](#) for more information on how to programatically move the robot.

4. Whiskers - assuming you have constructed the whiskers on the Boe-Bot breadboard pressing one or both of the whiskers will change the appropriate checkbox by turning it on or off. Be sure to have enabled the Whiskers by clicking on the checkbox next to "Whiskers (Pins 7,8)" which tells RoboRealm to start querying the Whisker states on the Boe-Bot. Note that this will slow down the frame rate.

If you want to access the Whisker states from other modules select or type in the variable name that will contain a "1" when the whisker is pressed and "0" when the path is clear.

5. IR Sensors - assuming that you have constructed the IR sensors on the Boe-Bot the appropriate checkbox will turn on when something is in front of the IR sensor. Be sure to have turned on the IR Sensors by selecting the checkbox next to "IR Sensor" which will tell RoboRealm to start querying the state of the IR sensors on the BoeBot. Note that this will also slow down the frame rate.

If you want to access the IR states from other modules select or type in the variable name that will contain a "1" when the IR sensor detects an object and "0" when the path is clear. Note that the IR sensor is a binary sensor and does not return range information as some other IR sensors do.

6. Sound - to test that the piezo buzzer is working select a frequency from the second dropdown menu and a appropriate during in the next dropdown. Click on Play and you should hear the note coming from the Boe-Bot.

To automatically play a sound type in or select a variable for both frequency and duration that will contain the appropriate frequency and duration number. When these variables become set RoboRealm will command the Boe-Bot to play the note and then will remove the variables' values. The variables are cleared to prevent the note from being played again and again. To once again play a note, simply set the variables again with appropriate values.

7. Maneuver - To perform a set sequence of movement click on the appropriate button to execute that movement on the BoeBot. The following are the movements for the provided buttons:

U - back up and then turn around

R - back up and then turn right

L - back up and then turn left

If you wish to automatically execute these movements without pressing the button select or type in a variable that will contain a value (presumably U, R or L) that will be sent to the BoeBot to execute the appropriate movement. Note that you can modify the bs2 program to accept more movement commands by changing the bs2 code appropriately. Specifying that new command in the Maneuver variable will then send that command to the BoeBot as RoboRealm does not restrict the command letters that are passed to the Boe-Bot.

8. Input/Output Pins - The Boe-Bot comes with 16 IO pins on the Board of Education. Most of these pins are for user usage and can be configured as input or output pins. RoboRealm provides a way to either receive or send signals to these pins via the bs2 program. Note that several of the pins are already in active use by the Whiskers, IR Sensor, Buzzer, etc. The IO Pins interface in RoboRealm provides you with a way to interact with other digital devices other than those provided in the base Boe-Bot kit.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low.

For input pins the checkbox will reflect the high or low states of the pin but remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

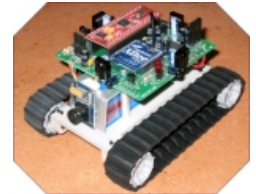
As an experiment, you can connect an LED to pin 15 via a 220 resistor as specified in the Boe-Bot manual. Then select the pin as an "out" and by checking and un-checking the checkbox you can make the LED blink. If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

See Also

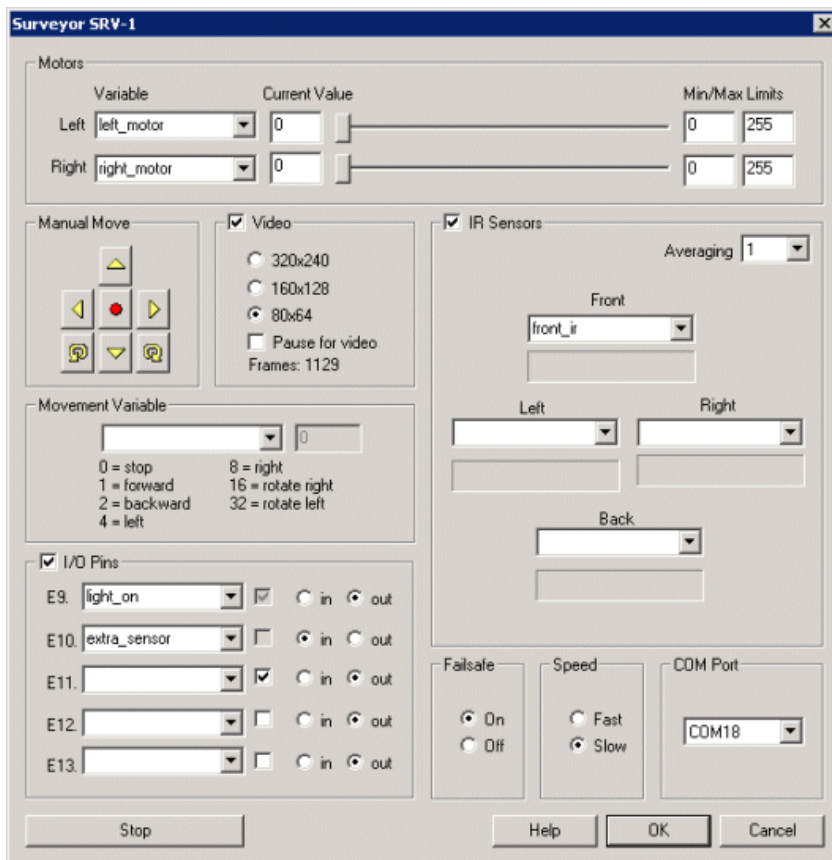
[Parallax Servo Controller](#)
[Sparkfun Arduino](#)

Surveyor SRV-1

The Surveyor SRV-1 module provides an interface to the Surveyor SRV-1 robot. The module will request images from the Robot via the specified COM port and allow you to view and process the images using RoboRealm. Motor commands based on the captured images can then be sent back to the SRV-1 robot to change its direction/speed.



Interface



Instructions

1. COM Port - select the appropriate COM port that the SRV-1 is currently communicating on. If you have connected the robot directly to your computer the lower port numbers (1-4) are most likely the correct ports. If you are using the Zigbee wireless connection the port numbers will most likely be higher than 4. You should only see COM ports that are active in your system in the dropdown.
2. Manual Move - there are three ways to move the robot. The manual move allows you to press the appropriate buttons and move the robot in a direction for a period of 500 ms.
3. Motors - the second way to move the robot is to move the Motor sliders that control each of the individual motors. Note that this interface is provided to map RoboRealm variables to each of the motors to enable differential steering control from your RoboRealm control program. You can use the Min and Max editable areas to limit the range of movement that the robot will be commanded to execute. Often variables may contain

irregular values due to scripting error so ensuring a limit on the motor values will ensure that your robot does not suddenly dart off in an unknown direction. A slow suggested range is 90 - 160.

The motor values range from 0 to 255 with 128 being neutral or off. Higher numbers move the robot forward while lower numbers will move the robot backwards.

See [Variable Control](#) for more information on how to programmatically move the robot.

4. Movement Variable - the final and third way to move the robot is to select a movement variable. This variable should contain the following values which mimic the manual move buttons. This style of automated movement control is not as refined as the individual motor command but can be useful for interfacing to interface devices like the keyboard or joystick.

0 = stop

1 = forward

2 = backward

4 = left

8 = right

16 = rotate right

32 = rotate left

5. Video - to view the video stream ensure that the checkbox next to the Video group is checked. This will start video streaming from the SRV-1. Select the appropriate size while keeping in mind that smaller sizes allow for more rapid updates. At an image size of 80x64 you should get around 4 fps. Once the video is enabled you should start to see the image within the main RoboRealm interface. You can now proceed with machine vision based filters to process what the robot is seeing.

The "Pause for video" checkbox will ensure that while an image is being downloaded from the SRV-1 that its motors are off. This ensures that the robot does not move while it is blind. If you find your SRV-1 does not appear to be responding to visual stimuli or overshooting the target, select this button to reduce the motor speed and gain better control over the image processing and motor reaction loop.

6. IR Sensors - to use the IR sensors make sure the checkbox next to the IR sensor group is checked. This will enable IR readings from the robot to be sent back to RoboRealm. To utilize these values type in a variable above the IR reading bar that will contain the value of the IR sensor. This variable can then be used to alter the robot's movement. See below for a quick script that displays this functionality.

As IR values can be very noisy an averaging filter can be required prior to the value being placed into the selected variables. Selecting a number in the Average box will average the past X IR samples and then place the value in the variable. This allows a more stable sampling of the IR values.

IR range values range from 0 to 160.

7. I/O Pins - to extend your Surveyor Robot you can use the IO Pins located on the robot to interface with other sensors or signal additional lights/LEDs/etc. To enable pin setting check the checkbox next to the I/O Pin group. The interface will then be enabled.


The GUI interface allows you to manually turn the pins off and on using the checkbox. Note that the checkbox is only enabled when the pins are set to output. The checkbox is invalid when the pin is set to input values. To automate the setting or getting of pin values assign a variable to the appropriate pin. If the pin is configured as an input then this variable will be a 1 or 0 if the input is high or low. If the pin is configured as an output then variable values of "0", "", "off", or "low" will set the pin low otherwise any other value will set it high.


8. Failsafe - to protect the robot when a loss of communication occurs the failsafe mode will ensure that the robot turns off motor commands if the wireless communication is no longer present. This can occur for a number of reasons but is normally related to the distance between the robot and PC Zigbee receiver.

9. Speed - a manual selection for the speed of the robot when using the Manual Move buttons.

10. Stop - sometime the most important button in the interface is the stop button. This button will terminate the robot's movement and allow you to regain control of the robot regardless of the variables specified in the motor interfaces.

Example

 [Click here](#) to load a configuration to move the SRV-1 using your cursor keys. Note that you will need to change the COM port before this will work. We used COM18. Also note that the video and IR has been disabled to increase the response time over wireless communication.

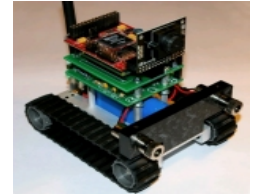
 [Click here](#) to load a configuration to move the SRV-1 forward until it comes into proximity of an obstacle in front of the robot. This is a good test of the IR capabilities. Note that really dark/black objects will not be detected as readily as light/white objects due to reflectance properties of the

IR beam

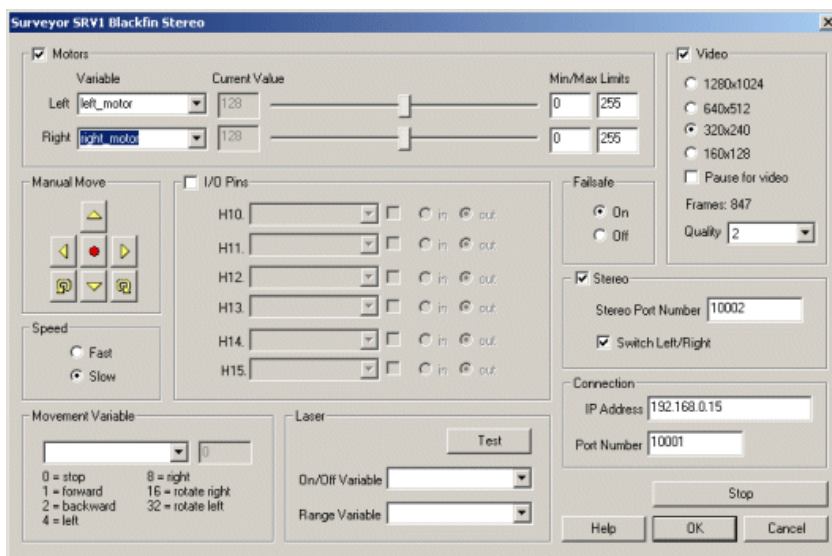
[Click here](#) to check out our brief tutorial on moving the SRV-1 within a constrained maze while avoiding walls.

Surveyor SRV-1 Blackfin WiFi

The Surveyor SRV-1b module provides an interface to the Surveyor SRV-1 blackfin robot. The module will request images from the Robot via 802.11 WiFi and allow you to view and process the images using RoboRealm. Motor commands based on the captured images can then be sent back to the SRV-1b robot to change its direction/speed.



Interface



Instructions

1. IP Address / Port - specify the appropriate IP address (192.168.0.15) to what you have configured your SRV-1b robot to respond to. Be sure to change the default port number 100001 if you have also changed that default configuration.
2. Manual Move - there are three ways to move the robot. The manual move allows you to press the appropriate buttons and move the robot in a direction for a period of 500 ms.
3. Motors - the second way to move the robot is to move the Motor sliders that control each of the individual motors. Note that this interface is provided to map RoboRealm variables to each of the motors to enable differential steering control from your RoboRealm control program. You can use the Min and Max editable areas to limit the range of movement that the robot will be commanded to execute. Often variables may contain irregular values due to scripting error so ensuring a limit on the motor values will ensure that your robot does not suddenly dart off in an unknown direction. A slow suggested range is 90 - 160.

The motor values range from 0 to 255 with 128 being neutral or off. Higher numbers move the robot forward while lower numbers will move the robot backwards.

If you find that your robot is not moving ensure that you at least hear a 'whine'. If not increase the values being sent to a higher range and ensure that the min and max are set to allow for that higher value.

You can also try lifting the robot off from the ground and test the motor values. Once the motors do not have as much momentum to overcome they may start to move. This also indicates that the motor values are too low to cause the robot to move.

See [Variable Control](#) for more information on how to programatically move the robot.

4. Movement Variable - the final and third way to move the robot is to select a movement variable. This variable should contain the following values which mimic the manual move buttons. This style of automated movement control is not as refined as the individual motor command but can be useful for interfacing to interface devices like the keyboard or joystick.

0 = stop

- 1 = forward
- 2 = backward
- 4 = left
- 8 = right
- 16 = rotate right
- 32 = rotate left

5. Video - to view the video stream ensure that the checkbox next to the Video group is checked. This will start video streaming from the SRV-1. Select the appropriate size while keeping in mind that smaller sizes allow for more rapid updates. At an image size of 160x128 you should get more than 5 fps. Once the video is enabled you should start to see the image within the main RoboRealm interface. You can now proceed with machine vision based filters to process what the robot is seeing.

To further increase the video rate you can select a lower quality level for the video image. This will cause more compression of the image which results in fewer bytes being transferred to the PC system but also causes degradation in the image.

The "Pause for video" checkbox will ensure that while an image is being downloaded from the SRV-1 that its motors are off. This ensures that the robot does not move while it is blind. If you find your SRV-1 does not appear to be responding to visual stimuli or overshooting the target, select this button to reduce the motor speed and gain better control over the image processing and motor reaction loop.

6. Stereo - If you have the stereo version of the SRV-1 you can enable the capture of the second image by selecting the checkbox next to the Stereo group. If your SRV is configured to use a different port number specify that port number in "Stereo port number". The second image will then be captured and stored as a "surveyor_stereo" and can be accessed using the [Marker](#) module or by other modules that require additional images than the currently viewed image such as the [3D Viewer](#) module. If you would like to quickly check the second image select the "Switch Left/Right" which will cause the second image captured to become the current image seen in the main RoboRealm interface.

7. Laser - you can test the laser lights by clicking on the "Test" button which will turn on the lasers for 5 seconds. Be sure to have the lasers pointed to a safe direction before turning testing.

To turn the lasers on and off using automatic control select a variable that when non null (i.e. it contains some text or number) the laser will turn on. Likewise when the variable is "" or is deleted the lasers will turn off.

When the lasers are turned on the Range variable will be set with the estimated distance from the robot to whatever the lasers are shining on. Note that this range variable is only valid if the lasers are turned on.

8. I/O Pins - to extend your Surveyor Robot you can use the IO Pins located on the robot to interface with other sensors or signal additional lights/LEDs/etc. To enable pin setting check the checkbox next to the I/O Pin group. The interface will then be enabled.

The GUI interface allows you to manually turn the pins off and on using the checkbox. Note that the checkbox is only enabled when the pins are set to output. The checkbox is invalid when the pin is set to input values. To automate the setting or getting of pin values assign a variable to the appropriate pin. If the pin is configured as an input then this variable will be a 1 or 0 if the input is high or low. If the pin is configured as an output then variable values of "0", "", "off", or "low" will set the pin low otherwise any other value will set it high.

9. Failsafe - to protect the robot when a loss of communication occurs the failsafe mode will ensure that the robot turns off motor commands if the wireless communication is no longer present. This can occur for a number of reasons but is normally related to the distance between the robot and the 802.11 receiver.

10. Speed - a manual selection for the speed of the robot when using the Manual Move buttons.

11. Stop - sometime the most important button in the interface is the stop button. This button will terminate the robot's movement and allow you to regain control of the robot regardless of the variables specified in the motor interfaces.

12. Servos - The base SRV robot has the ability to produce PWM signals on pins TMR2, TMR3, TRM6 and TRM7. Pins TRM2 and TRM3 are connected to the onboard motors and thus are not available for use with servos. Pins TRM6 and TRM7 are available and can drive regular PWM based servos. The Servos tab provides access to those pins and can be configured to move servos based on entering in the appropriate variable that will contain the value for those servos. Note that if you are using the SVS system then you will have 4 additional PWM signals on the second SRV board. As this board is not connected to any motors the TMR2 and TMR3 pins are available.

13. RCM Servos - For even more servos you can connect the RCM expansion board to the I2C interface of the SRV and have access up to 20 additional servo control signals. The RCM tabs provide an interface to those servos to allow them to be programmed via a variable setting.

Working with the SRV and RoboRealm

1. Ensure that your computer is connected to the SRV1b using a Peer to peer network or that the SRV1b is connected to a wireless network router. Note that you need to be sure in either case that your network address is compatible with what is being used. The SRV1b by default is configured for 192.168.0.15 which means that your computer needs to be using something like 192.168.0.X where X is anything but 15, 0 or 1. (The 0 and 1 are normally reserved for gateways and routers). If this is not the case then you can either change your computers IP address or the SRV1b's address.

Remember As Default - Select the "Remember as Default" checkbox if you would like the current IP setting to be remembered by RoboRealm

such that whenever the SRV1b is loaded the IP address will be auto-populated to the current setting. This ability allows you to not have to constantly change the IP address when loading in successive RoboRealm SRV configurations.

2. Switch On The Surveyor. If you are using a peer to peer connection your wireless link indicator should go green on your computer.

3. Run RoboRealm.exe

4. Insert in the SRV1b module. The quickest way to do this is to click on the Search tab in the upper left corner and type in surveyor. The module will appear in the list below your typing. Double click on that module to insert it into the pipeline. Note that there are TWO Surveyor_SRV modules. One without the 'b' which is the old system and one with a 'b'. Chose the one with a 'b'.

5. Configure the module to use the correct IP address if you are not on a 192.168.0.X network. If you are not and you change the IP address you will have to wait a bit after you are done changing the IP address to let the current network connection timeout. Once this happens the new IP address should be accepted and a connection will be attempted. You should see a single image from the SRV1b in the RoboRealm main GUI at this point.

SVS version (Stereo SRV1b) only from this point on.

6. Click on the Stereo checkbox. If you are using a non-standard Stereo port you will have to change that setting. Again, once you do this you will have to wait for the current network connection attempt to timeout before the setting will be accepted by the application.

7. Using the Switch Left/Right checkbox you will now be able to flip between the two images coming from the SVS.

8. Close this GUI by clicking on OK and insert the 3D Viewer module (again using the search tab mentioned above). This module is used to create 3D viewing images from the two image streams coming from the SVS.

9. Click on the "Two Separate Images" radio button to tell the module to grab the left and right images from two different sources. Then select one of them using the dropdown as "surveyor_stereo" which should be the last entry in the dropdown list. You can leave the other one as "Current" as that is the other image side.

10. You will now see a color Anaglyph. You can change the viewing options using the "Type" dropdown in the Options group. For example you can view the two images side by side by selecting "Side by Side".

11. Note that the two images will most likely not be very aligned and cause difficulty in creating a stereo image. Click on the "Sample" button which will cause the module to sample 10 images and then move one of the images such that it is better aligned with the other. You can also adjust these values manually by increasing or decreasing the values as needed.

12. Finally, you might not get the correct left/right image combination so try using the "Switch Left/Right image" checkbox to switch the left and right images quickly. It does matter which way around these are so try both until you have an image that appears more 3D. Note that once you click on the Switch checkbox you will need to realign the images as switching the images throws off the alignment.

13. Note that due to the disparity of the SVS you may first want to view objects far away from the SVS which will feel more relaxed on your eyes. As objects move closer the increasing disparity will start to strain your eyes in order to keep stereo viewing.

Example

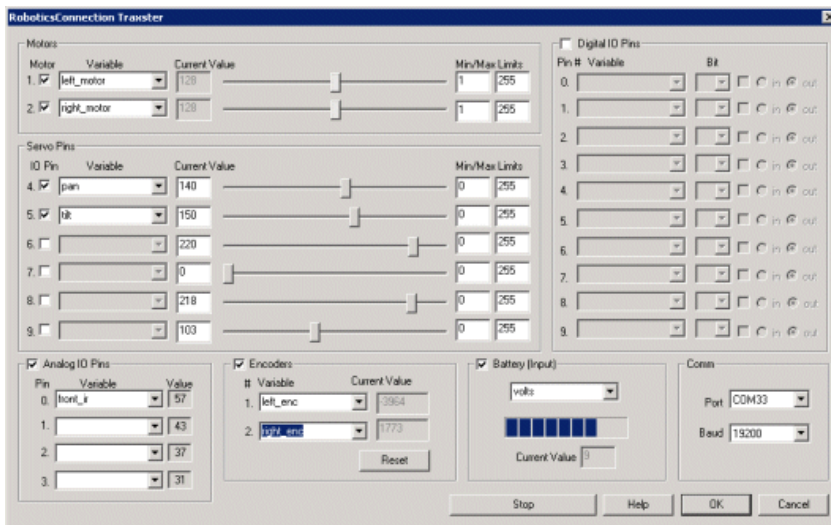
[Click here](#) to check out our tutorial on the SRV-1 blackfin following a trail of orange squares.

RoboticsConnection Traxster Robot



The RobConn_Traxster module provides an interface from RoboRealm to the [RoboticsConnection Traxster](#) robot. The module provides an interface to configure the appropriate communication port and speed to the Traxster and allows you to view and modify settings by using the GUI based sliders and text editing areas. To automatically specify the values you should select variables from the dropdown menus that will contain the values that will be sent to the robot. Note that once variables have been selected manual control is disabled.

Interface



Instructions

1. **Comm Port** - Select the appropriate COM Port that the Traxster is connected to. You will only see COM ports that are recognized by your computer. Note that direct serial connections normally use com ports lower than 4 whilst virtual COM ports created by bluetooth connections are much higher. Check your bluetooth configuration manager to see which port your robot communications is installed on. Note that this will be an Outgoing port.
 2. **Baud Rate** - Select the appropriate baud rate. 19200 is the default speed used. Select a higher baud rate if you want to and can increase the communications speed. Be aware that wireless devices may not support the full baud range of the Serializer board (Traxster's cpu board). If you accidentally set the baud speed to something faster than your wireless device supports you will have to connect the Traxster directly to your PC in order to change the baud rate to a lower setting.
 3. **Battery** - once the Traxster is communicating with RoboRealm you can turn on the battery monitor to test the connection. Do this by simply selecting the Battery area checkbox in the interface. The power level will be shown in blue bars with the number of volts in the "Current Value" textbox. If you want to use the volts within other areas of RoboRealm select a variable that the module will write the number of volts to. This variable will then be updated each time the Traxster module is run with the current voltage of the robot.
 4. **Motor Sliders** - After specifying the COM Port you should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the "Current Value" text box in the Motors area. If the motors do not move (or whine) check your COM Port setting and/or board connections.
 5. **Variables** - Select the appropriate variables that contain or will contain the position value that will be sent to the robot. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Extension based program.
 6. **Current Value** - To manually set the motor position type in the appropriate number (0-255, 128 is the default neutral/stop) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate.
 7. **Min/Max Limits** - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the robot does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case you are testing your robot in a precarious position.
 8. **Servo Pins** - Using a similar interface as the motors you can specify the position for up to 6 servos. Click on the first checkbox next to each servo to enable it. Once you do so you will notice one of the Digital IO pins become disabled. This is due to the servo pin and IO pin being the same pin. While there are 10 IO pins only 6 can be used to control servos. Note that additional power supply and heat sinks may be needed if you are using 6 servos.
- You can use the slider bar to move the servo or type in a number directly in the "Current Value" textbox to move the servo to that position. To move the servo automatically select a variable that contains or will contain the value to move the servo to.
9. **Analog Pins** - The Traxster provides for 4 user analog inputs. (There is actually one more but that is tied to the battery). If you enable the Analog area you will notice the values in the corresponding read only text boxes change based on the voltage across those pins. If you have a distance sensor connected to an analog pin move your hand in front of the sensor to see these values change. By specifying a variable in the corresponding dropdown you can access that value elsewhere in RoboRealm in order to make decisions about the sensors value.

10. Encoders - To enable the encoder area click on the group checkbox. This will enable the encoder readings and display the number of encoder ticks that the robot has traveled. If you enable this area and then move the robot using the Motor sliders you will see the encoder values change depending on how fast the robot is moving. Specifying variables in the dropdown will cause those variables to be set to the current values read from the robot. Also note the "Reset" button that can be used to reset the encoder values on the robot to zero.

11. Digital IO - In addition to the 4 analog inputs the Traxster provides 10 digital input/output pins. Note that if you are using servos some of those pins will already be used. The remaining pins can be used to drive outputs or receive inputs.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output you can manually control the output by selecting the checkbox which will then turn that pin high or low.

For input pins the checkbox (in a disabled state) will reflect the high or low states of the pin.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown. If no bit is specified the entire value is considered during a set/output.

As an experiment, you can connect an LED to pin 0 in the Traxster. Then select the pin as an "out" and by checking and un-checking the checkbox you can make the LED blink under manual control. If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.


12. Direct Serial Command Variable - As the serializer has many additional capabilities not reflected in the GUI the direct serial command variable will send the text contained in the specified variable to the serializer and place the returned results back into that same variable. Thus you can specify a variable (or type a new one) into the provided textbox and set the value to any of the serial commands that the serializer understands. For example, you can use

```
SetVariable "my_command", "i2c r 4 1"
```

in a VBScript module to query the line following sensor.

13. Stop - press the stop button to quickly stop the motors. The motors controls will then be disabled (in case variables are selected) and will only restart when you click the same button again (now renamed to "Start").

Example

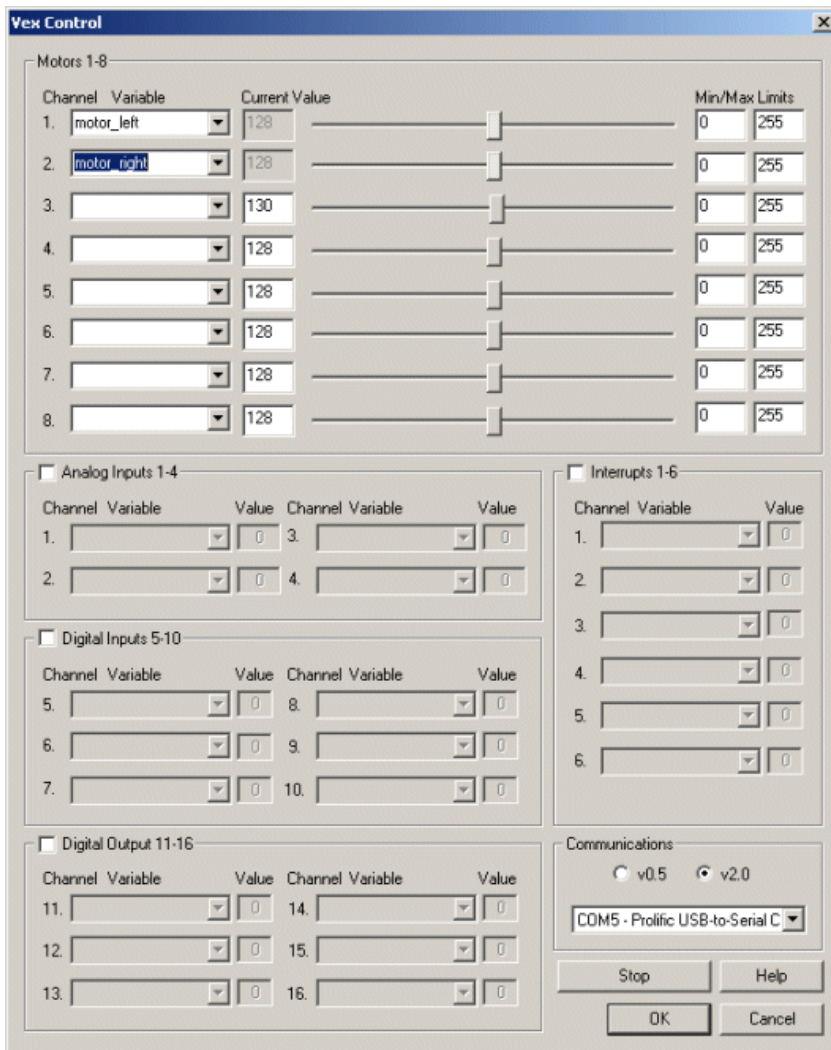
 [Click here](#) to load a configuration to move the Traxster robot using your joystick. Be sure to have your joystick attached before running RoboRealm. You will also need to select the correct COM port on the Traxster module interface. Do this by double-clicking on the RobConn_Traxster module and selecting the correct Port in the dropdown menu. Also note that the pan is configured for servo #4 and tilt for servo #6. Selecting the appropriate buttons on your joystick will pan and tilt the robot turret in increments.

Vex Controller

The Vex Controller module provides control of the Innovation First Intelitek Vex Controller from your PC. You *must* have downloaded the online mode using the intelitek easyC for Vex controller or the iLoader.exe. Ensure that the online mode is working correctly by using the intelitek On-Line Window found under the Build & Download -> On-Line Window menu option.

Note that more than one version of the online mode is currently available. If you recently purchased the Vex Starter kit then you most likely have ver2.0, otherwise use ver0.5. There is no difference between the two versions from the PC control perspective.

Interface



Instructions

1. Communications - Select the appropriate Easy C version and the correct COM Port. The Prolific USB-to-Serial Comm port is selected by default which assumes you have the USB to Serial connection. If this is not installed chose the correct COM port.
2. Motors - After specifying the version and COM port you should be able to move your servos/motors by dragging the sliders to the right or left or by specifying a number within the current value text box in the Motor configuration area. If the servos do not move check your version and COM port settings. You should see the yellow light in the Vex controller flash when any serial communication is present. If you do not see this light flashing when moving the scroll bar around check your connection with the intelitek application. If you do see a flashing light with no movement check which channel you plugged the servo/motor into.
5. Variable - Select an appropriate variable that contains or will contain the position value that will be sent to the servo. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program. See [Variable Control](#) for more information on how to programatically move the robot.
6. Min/Max Limits - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
7. Analog Inputs - There are four analog inputs that can be connected to the Vex controller. Be sure to enable the analog inputs before testing by checking the checkbox next to the "Analog Inputs" group text. Once enabled you should see the current values next to the variable dropdowns. Select or type in a variable that should contain the current analog input value. This variable is then accessible from other RoboRealm modules or API commands.
8. Digital Inputs - There are 6 digital inputs that can be connected to the Vex controller. Unlike analog inputs whose values have a range, digital inputs are either on or off. These are often used for switches to detect a certain state such as touching or not touching an object.

Be sure to enable the digital inputs before testing by checking the checkbox next to the "Digital Inputs" group text. Once enabled you should see the

current values next to the variable dropdowns. Select or type in a variable that should contain the current analog input value. This variable is then accessible from other RoboRealm modules or API commands.

9. Digital Outputs - There are 6 digital outputs that can be connected to the Vex controller. These can be used to switch devices on or off, such as an led. Be sure to enable the digital outputs before testing by checking the checkbox next to the "Digital Outputs" group text. Once enabled you should see the current value of the selected variable next to the variable dropdowns. The selected variable should contain the value that will be sent to the Vex Controller to switch the output on or off.

10. Interrupts - Similar to the digital inputs, interrupts detect an on / off situation that can be used to interrupt normal processing in order to deal with a priority situation. Once enabled, the current interrupt values will appear next to the variable dropdowns. Specify variables that will contain the current interrupt value to be used in other RoboRealm modules or accessed via the API.

10. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.

Notes

The newer VexPro controllers permit you to add additional software to stream video from the controller to RoboRealm. Paul Bouchier from the Dallas Personal Robotics Group has created exactly that software. this can be checked out of SVN repository of the Dallas Personal Robotics Group at: `rrTest` and `rrClient` (no longer available online!). `rrTest` runs the RoboRealm API test from the VexPro and `rrClient` sends images from a VexPro-connected webcam to RoboRealm and pulls back variables. Paul has used this solution in combination with a Roomba robot. More information about the competition that system was built for can be found at [RoboRama2012a](#).

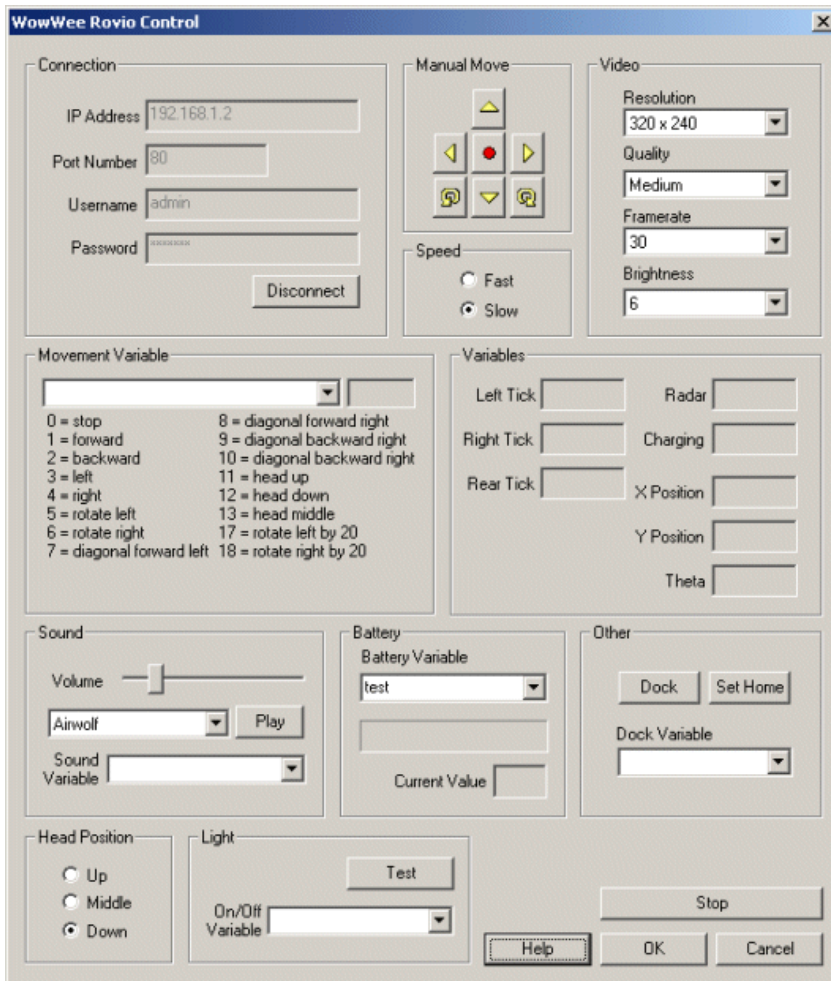
WowWee Rovio

The Rovio module provides an interface from RoboRealm to the WowWee Rovio robot. Using the module you can command the robot to respond to images seen from the Rovio robot. The robot sends images over 802.11 (WiFi) to a PC. The PC is running RoboRealm that accesses that video stream, processes the video as per your needs and then sends the resulting motor commands back to the robot also using 802.11. Using this technique you can control your Rovio robot from your PC and use the full power and flexibility of your PC to control the Rovio robot. This allows you to extend the functionality of the Rovio robot beyond its original limitations.



Be sure to have updated your Rovio firmware and are running at or above firmware 5.03 (stable) if you find any issues between the RoboRealm module and the Rovio. For example, if your video has more than 1 second lag between the actual time and the time the image is displayed or if you lose connection with the Rovio after 60 seconds be sure to update the firmware. You can download the latest version which includes these fixes from RoboCommunity which used to be online at <http://www.robocommunity.com/download/15224/Rovio-firmware-UI-v5.00b10-Beta/>

Interface



Instructions

1. IP Address - specify the ip address of the Rovio. This can be determined by using the default address or by checking your wireless router to see what IP address has been assigned to the Rovio robot.
2. Username / Password - specify the username and password for the Rovio.
3. Connect - press the connect button to connect to the Rovio and start streaming video.
4. Manual Move - to test the motors on the Rovio you can use the Manual move buttons to move the robot in the specified direction. Note that the robot will continue to move for as long as the buttons are pressed. The middle button stops the robot.
5. Speed - to make the robot move quicker when using the manual move buttons you can select a faster speed. Note this speed adjustment are just used for the manual move buttons.
6. Video - to change the video parameters select the appropriate dropdown box. There will be a slight pause as the robot updates its internal parameters before video resumes. Changing Resolution will change the image size streamed from the robot. Quality will change the compression rate of the video being sent. Low will produce high compression resulting in a small video size, High will induce very light compression and therefore high bandwidth requirements but with high quality images. Framerate determines the maximum number of images that are send each second. Brightness effects the lighting level of the camera. Note that this parameter is set to the maximum setting. The image will appear dark in an evening type of setting. We recommend you use the robot in a high lighting environment.
7. Enhance - To help reduce the lighting issues you can ensure that the Enhance checkbox is selected which will attempt to realign the image colors and intensity in order to create a more pleasing image. This will not affect images with good lighting but should substantially improve those with bad.
8. Movement Variable - you can select a variable that contains one of the command numbers that will cause the robot to move in the specified manner. Note that this also includes the value to move the camera head up and down.
9. Variables - shows the different numbers being read from the robot. The Tick variables are from the wheel encoders and indicate how much a wheel has turned. Radar is the value of the IR obstacle detection (0 = no object, 1 = object). Charging indicates if the robot is being charged (i.e. it is docked). X, Y, and theta reflect the robots location within a room where the overhead beacon can be seen. Note that the values will fluctuate even when standing still.

10. Sound - To test the audio playback on the Rovio select on the RTTTL songs and press the play button. The song will then play on the Rovio.

To programmatically play a song select a variable that will contain the name of the song to play. Be sure to only specify this once otherwise the songs will overlap on each other (not a great audio experience!). You can adjust the volume of the sound by adjusting the slider bar left to lower the volume and right to increase it. If you want to add songs you can edit the music.rtttl file in the RoboRealm installation folder and add in your own ringtone songs or sounds.

11. Battery - The battery level is indicated by the progress bar below and the actual current value. To programmatically react to the battery value select a variable that will contain the battery value.

12. Other - To dock or set the new home position press on the appropriate button. To programmatically dock or set the home position select a variable that would contain either "dock" or "set home" (without the quotes) that would indicate what action the robot should perform

13. Head Position - To move the camera head into a different position select one of the appropriate radio buttons.

14. Light - you can test the main Rovio light by pressing the Test button. If you want to control the light via a variable (zero or non-zero value) select that variable in the provided dropdown.

Examples

Please see our [tutorials](#) for examples on how to use the Rovio with RoboRealm

Variables

ROVIO_REAR_TICK - rear wheel encoder value

ROVIO_LEFT_RICK - left wheel encoder value

ROVIO_RIGHT_TICK - right wheel encoder value

ROVIO_RADAR - indicates an obstacle in front of the
robot (1) or not (0)

ROVIO_X_POSITION - the X position in the room based
on the overhead IR pattern

ROVIO_Y_POSITION - the Y position in the room based
on the overhead IR pattern

ROVIO_THETA - the orientation of the robot based
on the overhead IR pattern

See Also

[BrookStone Rover 1.0](#)

[Erector Spykee](#)

[Xaxxon Oculus](#)

Xaxxon Oculus

The Oculus module provides a way to interface RoboRealm with the Xaxxon Oculus robot. Specifically this module provides access to the Arduino based motor controller that is embedded within the robot. The commands are sent over USB from RoboRealm to the Arduino which then regulates the motor power.



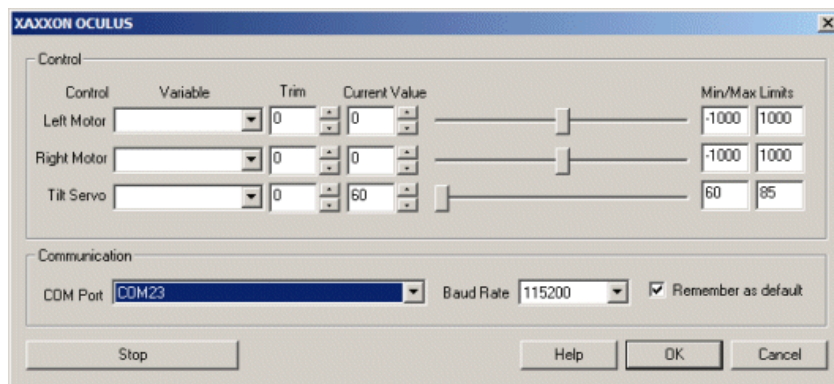
The Oculus platform instantly converts a small laptop into a telerobotic, remotely operated vehicle. The platform allows the robot to also be converted into an Autonomous System by running applications on the netbook. Using the onboard camera provided in most netbooks today, you can use the Oculus robot as an inexpensive vision based robot that provides a familiar keyboard, touchpad, and monitor (i.e. the netbook) to work with.

The robot is powered by the netbook through USB. While this low power causes the motors not to be sufficient for very thick carpet they do allow very controlled movement of the robot over a hard surface.

Note that you cannot run more than one system at a time as the serial port and camera will NOT be shared amongst different applications. If you want to modify/enhance the camera output before sending it via the web you can install the RoboRealm [Virtual Camera](#) that will allow both RoboRealm and other systems to share the same camera.

You can order the Oculus robot by browsing to the [Oculus Store](#).

Interface



Instructions

1. COM Port - Specify the COM port that the Oculus is connected to. Note that the Oculus comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.
2. Baud Rate - Specify the baud rate of 115200 to communicate with the Oculus. Note that this is much higher than the standard 9600 normally used. The higher rate provides much better responsiveness than the lower rates.
3. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the Oculus module is loaded the COM port and baud rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.
4. Motors/Servo - Once communications is specified you should be able to move the appropriate slider in order to see the motor move. If the motor does not move, check your USB connection and motor wiring.
5. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the motors. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Plugin based program.
6. Current Value - To manually set the motor position type in the appropriate number (-1000 to 1000, 0 is the default neutral) into the text area or use the slider to adjust the value. The motor speed will be updated as appropriate.
7. Sliders - You should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the current value text box. If the motors do not move check your USB cable and/or board power connections.
9. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case your motors cannot physically move beyond certain limits.

Examples

Joystick

[Click Here](#) to load a robofile that will allow you to drive your Oculus using a joystick. When you load this configuration you will need to edit the Joystick module to select which joystick to use to control the robot.

This robofile is configured for a playstation type of joystick with the first knob providing motion control. Moving the one knob will move one wheel forward/backward, while the other will do the same for the other wheel. This is a tank like drive.

If you do not like the configuration we have chosen simply edit the Joystick module and change the variables around to different buttons.

Once that is done have a safe and happy drive!

Green Ball Tracking

[Click Here](#) to load a robofile that will move the Oculus based on a green ball. When you load this configuration you might need to tweak the

RGB filter to correctly detect the green ball. Also check the COM port in the Oculus module to be sure the correct port number is set. When ready press the Run button to start. When you hold a green ball in the center of the camera image it will not move. Moving the ball slowly to the right will cause the robot to turn to keep the ball in the middle. When you move the ball down the robot will move backwards and vice versa.

Chasing Movement

[Click Here](#) to load a robofile that will move the robot towards movement. Detecting movement is also a useful behavior for a robot to have. Click the run button to start. The robot will move in short segments towards the last movement done. When it stops moving it will again look for movement and start in that direction again. Note that while the robot is moving it ignores any movement since most of the image is changing due to camera movement and not an object moving in the scene. See if you can steer the robot by wiggling just your hand in the direction you want it to move.

Docking Station Recognition

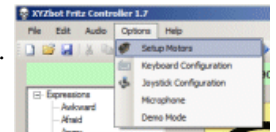
[Click Here](#) to load a robofile that will detect the docking station graphic. The pipeline includes two blob filter modules to eliminate false positives. The idea is to first threshold the image in a way to have the white triangular shape isolated from the rest of the scene. This involves removing smaller unwanted blobs, removing those blobs that do not share the same center and those that are not triangular in shape. Once this triangular shape is isolated we negate the image and isolate the inner circle. If the circle is not detected then we conclude that we isolated the wrong triangular shape. Given this dual requirement the detection all of the docking station shape should be quite stable.

XYZbot Fritz: A robotic puppet

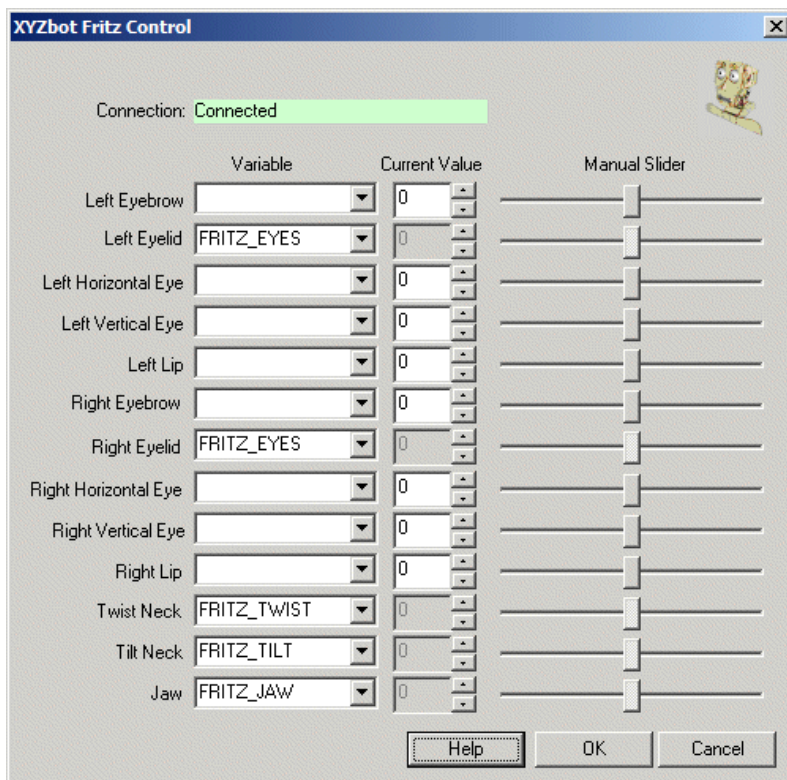


Currently a successfully funded [Kickstarter campaign](#), the XYZbot Fritz module provides an interface to the [Fritz robotic puppet](#) created by XYZbot. There are 13 servos in the advanced version that are driven by an Arduino microcontroller. This module will communicate to the Arduino to set each of those servos based on variables within RoboRealm. Using this interface you can move Fritz based on visual stimuli such as colored objects, fiducials, movements, etc.

The simple interface allows you to either move the parts manually using the sliders or automatically by selecting a variable whose value will be sent as the servo position. Note, the values range from -1000 to 1000 with 0 being neutral. The values represent the entire range as configured using the Setup Motors menu option in the software provided by XYZbot. This module assumes that you have already configured your Fritz with respect to the minimum and maximum values for each servo. The values sent by this module (-1000 to 1000) are then scaled into this minimum and maximum values before moving the servo. Thus you can use this module without modification on different Fritzes as long as they have already been setup.



Interface




Instructions

1. Connection - The module will automatically seek out the appropriate COM port that Fritz is configured on. If this connection fails to connect, check that you have the most recent firmware loaded on your Arduino and that you have appropriate power connected to Fritz.
2. Left Eyebrow, Left Eyelid, etc. - Each row represents a connection to a particular face part of Fritz. The Left and Right are specified as if you are looking at Fritz. Thus Left is your left and Right is your right as apposed to Fritz's left and right. I.e. the labels are NOT mirrored.
3. Variable - The variable dropdown allows you to select a variable from within RoboRealm that should be sent to Fritz in order to change the servo position. Note that this value should range from -1000 to 1000. Once you type in or select a variable, the manual slider and value will be disabled to indicate that the servo is now under automatic control.
4. Value - The value field represents the current numerical position of the servo as scaled by the minimum and maximum values configured within Fritz. You can change this by typing in a new number directly into the provided text box or use the up and down arrow buttons to change the value.
5. Manual Slider - You can also change the position of a servo by moving the slider using your mouse. This is an easy way to check that the connection is working and that you have control of Fritz from within this module. Note that once a variable is selected both the slider and the value controls are disabled.


Examples

Joystick

 [Click Here](#) to load a robofile that will allow you to drive your Finch using a joystick. When you load this configuration you will need to edit the Joystick module to select which joystick to use to control the robot. This is similar to the joystick capabilities of the software provided with Fritz.

Use the twist and rudder to move Fritz's neck left/right and up/down. Press button 1 to open the Jaw, press button 2 to close the eyes. Use the joystick to move the eyes. Note that the horizontal movement is the same for both eyes but the vertical is reversed (try doing that with your own eyes)! Can you fix that?

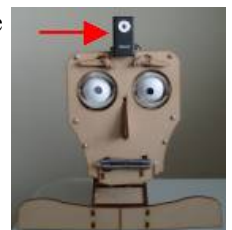
Fiducial Tracking

 [Click Here](#) to load a robofile that will cause Fritz to track a fiducial located in the Fiducials folder where you installed RoboRealm. You can print out any of those fiducials and show them to Fritz (avoid your fingers getting in the way) and he will track the object (slowly!) by moving his neck from right to left and up to down.



You will probably need to configure the Fiducials folder to the one where you installed RoboRealm. Once that is done, you should also press the Train button to ensure that RoboRealm learned those fiducials.

This example assumes that a camera is mounted ontop of Fritz's face. The way this works is that the camera takes the image while attached to Fritz, the image is fed back into RoboRealm running on a PC which processes the image and sends commands to Fritz via the Arduino serial connection.



See Also

[Sparkfun Arduino](#)

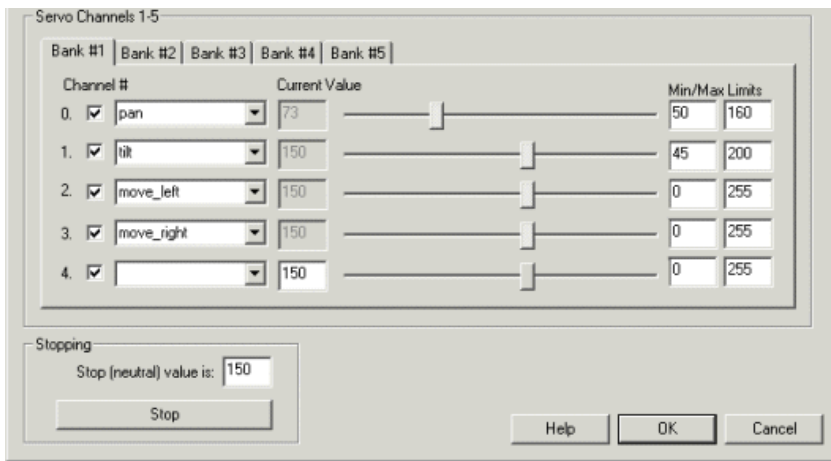
Endurance Servo Controller

The Endurance Servo module allows you to interface RoboRealm to servos using a controller made by [Endurance R/C](#). The USB based servo controller supports up to 25 servos and provides for an external servo power supply in order to run 5v, 6v, or 9v servos.



Interface





Instructions

1. **Checkbox** - To enable/disable a particular servo channel select the checkbox to change its state. When disabled the servo position will not change.
2. **Scroll Bars** - Move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Current Value textfield. If the servos do not move check your USB and/or board external power connections. You should see the red led light up on the controller if the USB connection is functioning correctly and providing power to the board. Note that in order for the servos to move you need to have the external power supply connected.
3. **Variable Dropdown** - Select an appropriate variable that contains or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program.
4. **Min/Max Limits** - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
5. **Press STOP** if you need to quickly disable all the servos and return them to the middle or neutral (150) position.

Note that if you set a servo channel to be positioned using a variable's value you will lose manual control over that servo. Changing the variable value, however, will change that servo's position.

See Also

- [Endurance PCTx](#)
- [Endurance WiFi Servo](#)

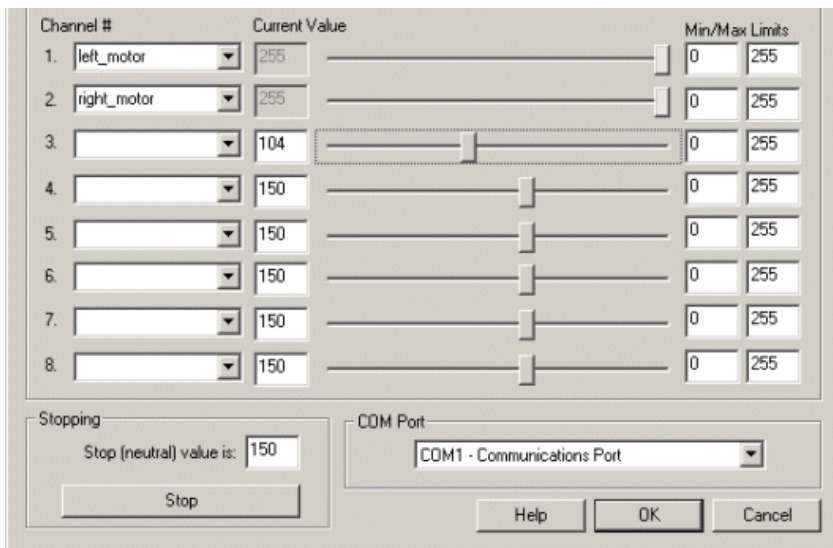
Endurance WiFi Servo Controller

The Endurance WiFi Servo Controller module allows you to interface RoboRealm to servos using a controller made by [Endurance R/C](#). The WiFi based servo controller supports up to 8 servos and can be controlled over a WiFi 802.11 wireless signal. The connection to the remote board appears as a COM port to the host PC.



Interface





Instructions

1. COM Port - Select the appropriate COM Port. Note that you will only see COM ports that are recognized by your computer.
2. Scroll Bars - After specifying the COM port you can try moving your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Current Value textfield. If the servos do not move check your wireless connection and/or board power connections.
3. Variable Dropdown - Select an appropriate variable that contains or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function), Extension based program or [Set Variable](#) module.
4. Min/Max Limits - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
5. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (150) position.

Note that if you set a servo channel to be positioned using a variable's value you will lose manual control over that servo. Changing the variable value, however, will change that servo's position.

See Also

[Endurance PCTx](#)
[Endurance Servo](#)

Lynxmotion SSC-32



The Lynxmotion SSC-32 module allows you to interface RoboRealm to servos using a controller made by [Lynxmotion](#) called the [SSC-32 Servo Controller](#). The servo controller supports up to 32 channels of 1uS resolution. In addition the board provides synchronized movement so that all servos will update at the same time. The board also supports 4 digital or analog inputs for adding additional sensors to your robotic projects.

Interface



Channel	Variable	Current Value	Min/Max Limits	Rate
0	<input checked="" type="checkbox"/> left_arm	500	500 2500	5000
1	<input checked="" type="checkbox"/> right_arm	500	500 2500	5000
2	<input type="checkbox"/>	1500	500 2500	5000
3	<input type="checkbox"/>	1500	500 2500	5000
4	<input type="checkbox"/>	1500	500 2500	5000
5	<input type="checkbox"/>	1500	500 2500	5000
6	<input type="checkbox"/>	1500	500 2500	5000
7	<input type="checkbox"/>	1500	500 2500	5000

Inputs			COM Port	Baud Rate
Channel	Variable	Value	COM1	115200
A.	touch_switch	1		
B.		1		
C.		1		
D.		1		

Digital Analog

Stop

Help OK Cancel

Instructions

1. COM Port - Select the COM port that the controller is connected to on your computer. Note that you will only see COM ports that are recognized by your computer.
2. Baud Rate - Select the appropriate baud rate. Note that 115200 is the default and should not be changed unless required. For instance, if you are using the controller board through a wireless connection you may need to slow down the connection speed due to the wireless hardware limitations. Be sure to also change the jumpers on the Lynxmotion SSC to reflect the desired rate.
3. Checkbox - To enable/disable a particular servo channel select the checkbox to change its state. When disabled the servo will not be sent any further position commands.
4. Scroll Bars - After specifying the COM port and speed you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Config area. If the servos do not move check your COM port setting and/or board connections. You should see a small green light flash on the Lynxmotion SSC board when any serial communication is present. If you do not see this light flashing when moving the scroll bar around check your connections. If you do see a flashing light with no movement check the channel you plugged the servo into.
5. Variable - Select an appropriate variable that contains or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.
6. Min/Max Limits - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
7. Rate - specify the speed at which your servos should move at. Note that once a new value is sent to the board the old position and speed will be overwritten.
8. Bank # - Select the appropriate bank of servos that you would like to configure. Note the channel number which corresponds to the channel number of the Lynxmotion SSC board. Only eight channels are visible at any time.
9. Inputs - select which type of input you wish to receive. Digital will set a one or zero. Analog will set a number between 0 and 255 depending on the voltage across the pins. By selecting or typing in a variable next to the appropriate input channel you can access that variable's value in other modules such as the VBScript or Write_Variables modules.
10. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (1500) position.

Note that if you set a servo channel to be positioned using a variable's value you will lose manual control over that servo. Changing the variable value, however, will change that servo's position.

The Lynxmotion SSC has servo position feedback information that is used to update the RoboRealm servo variable (if specified) based on where the servo currently is. As the servo rate can be changed the servo may not immediately adjust its position to the final destination but slowly move towards it. During this movement the servo variable within RoboRealm will be updated to reflect the current position of the servo. This can be used in successive VBScript or If_statement modules to change the behavior of the servo prior to completing the full move. Thus be aware that after the Lynxmotion_SSC module the servo variables may not have exact the values specified before the module is executed.

Example

[Click here](#) to load a configuration to move a servo in response to movement in front of the camera. Click on the previous link. This should run RoboRealm and start processing images. Be sure that the camera button is pressed and you have a webcam attached to your computer. If you see a black screen check the Options button and select an active camera in the dropdown list.

Ensure that the Lynxmotion servo controller board is connected to the computer. If you are using a different COM port than COM1 double click on the Lynxmotion_SSC module in the main RoboRealm dialog and change the COM setting in the interface that appears. Press ok to close the dialog.

Try waving your hand in front of the camera. Notice the green COG box tracking your hand. The servo should also move in one direction and in the other when you move your hand back. Note that when you stop moving your hand the picture disappears and the servo remains stable.

Click on each of the modules to see how we accomplished this task.

See Also

[SSC](#)

Nubotics Wheel Commander

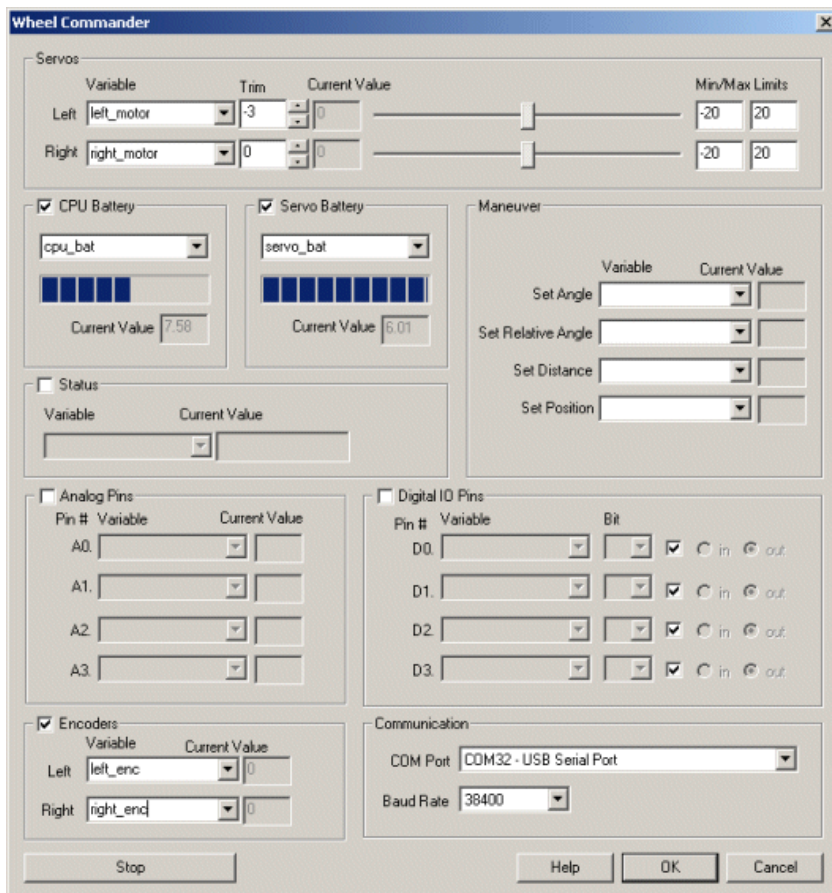


The [Nubotics WheelCommander™](#) is a servo/motor controller that incorporates closed loop functionality to differentially driven robots when combined with odometry encoders.

Motor control is derived from geometrical robot dimensions to create a highly abstracted motion control module.

The WheelCommander module allows you to interface this device within RoboRealm.

Interface



Instructions

1. Communication - Select the appropriate COM port that the controller is connected to on your computer. Note that you will only see COM ports that are recognized by your computer.

2. Baud Rate - Select the appropriate baud rate. Note that 38400 is the default.

3. Servos - After specifying the COM port and baud you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servos area. If the servos do not move check your COM port setting and/or board connections. You should see a small red light flash on the board when motor commands are being sent. If you do not see this light when moving the scroll bar around check your connections. If you do see a flashing light with no movement check the servo and left/right channel you plugged the servo into.

5. Variable - Select an appropriate variable that contains or will contain the position value that will be sent to the board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.

6. Min/Max Limits - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

7. CPU Battery - The CPU/Digital battery level is indicated by the progress bar and the actual current volts. To programmatically react to the battery value select a variable that will contain the battery value. The value indicates how many volts are being fed into the board to support the digital operations of the board. Note that you must select the checkbox next to the CPU Battery text in order to activate this functionality.

8. Servo Battery - The servo battery level is also indicated by the progress bar and the actual current volts. To programmatically react to the battery value select a variable that will contain the battery value. The value indicates how many volts are available to drive servos. Note that you must select the checkbox next to the Servo Battery text in order to activate this functionality.

9. Status - If you want to know the status of the board specify the variable that is used to store the status value in. Please refer to the WC user manual to determine what each of the status bits mean. Note that you must select the checkbox next to the Status text in order to activate this functionality.

10. Maneuver - If you want to specify a high level command to the board (as apposed to using the Servo Left/Right interface) you can issue Angle, Relative Angle, Distance and Position commands directly to the board using the provided variables. For example, to move the robot 90 degrees you would specify a variable like "robot_angle" and set its value to 90. During the robots rotation the variable robot_angle would then contain the current value the robot is repositioning itself to. This can be used to monitor the robots progress.

11. Analog Pins - If you want to sample the analog values specify a variable that will contain the value from the appropriate pin. Note that you must select the checkbox next to the Analog Pins text in order to activate this functionality.

12. Digital IO Pins - There are 4 Digital IO pins that can be configured as either input or output pins. Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low.

For input pins the checkbox will reflect the high or low states of the pin but remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

As an experiment, you can connect an LED to pin D0. Then select the pin as an "out" and by checking and un-checking the checkbox you can make the LED blink. If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

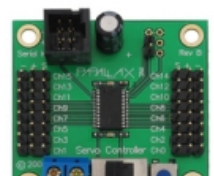
Note that you must select the checkbox next to the "Digital IO Pins" text in order to activate this functionality.

13. Encoders - To read the values of the encoders (remember to purchase the Wheel Watcher encoders along with this board) you can specify variables that will contain the encoder count for each wheel. Note that without encoders these values will not change from 0. You must select the checkbox next to the "Encoders" text in order to activate this functionality.

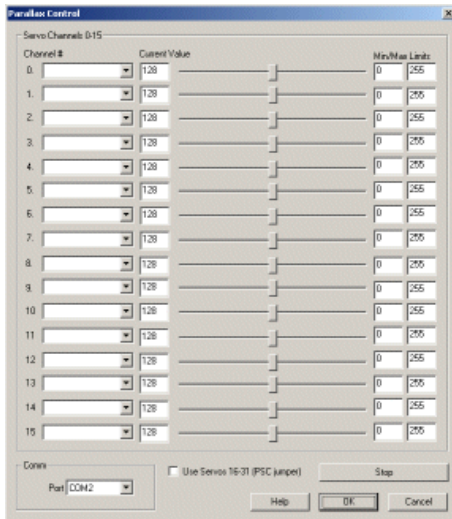
14. Stop button - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (0) position. This will also cause the servos to release power and enter into a lower power mode (coast).

Parallax Servo Controller

The Parallax module allows you to interface RoboRealm to servos via a servo controller made by [Parallax](#). The [Parallax Servo Controller \(Serial\)](#) or [Parallax Servo Controller \(USB\)](#) boards support 32 servo control channels.



Interface



Instructions

1. Comm - Select the appropriate port in the Comm group.
2. Servo Channels -After specifying this information you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Channels area. If the servos do not move check your Comm settings and/or board connections. Note that RoboRealm communicates using 2400 baud to the Parallax controllers.
3. Channel # - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program. You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

The ramp field is used to change the ramp speed of the servo to increase or decrease the rate at which the servo travels to its position. The valid range is from 0 to 63. If you wish to programmatically control the ramp field create a second variable with the same name as the specified variable but append a '_ramp' postfix to it. For example if your main variable to control the position is called "pan" then the second variable name to control the ramp value would be "pan_ramp".

4. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral position.

See Also

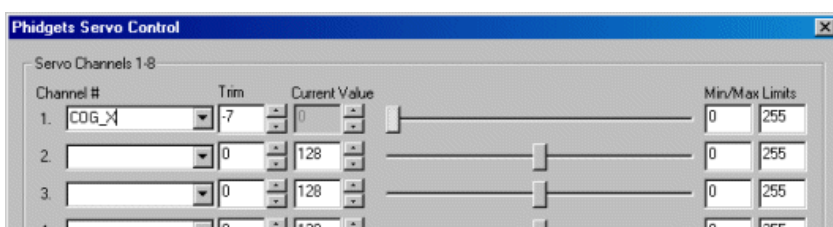
[Parallax Boe-Bot](#)

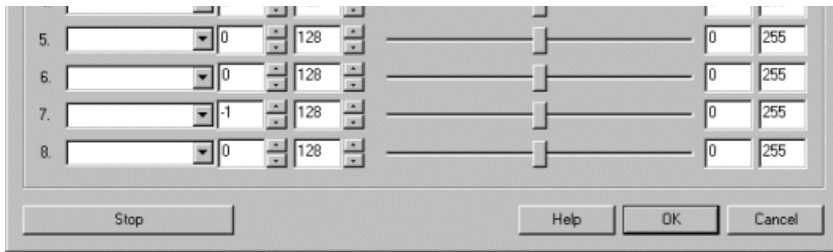
Phidget Servo Controller

The Phidget module allows you to interface RoboRealm to the Phidgets Servo Controllers made by [Phidgets](#). The Phidget controller has either 1, 4 or 8 control channels (depending on which controller you purchased) which are represented by the 8 dropdown lists.



Interface





Instructions

1. Sliders - On inserting the module you will be able to control each of the servo channels by either entering in a number from 0 to 255 in the text area or by dragging the scroll bar to the left or right. The servo position will be updated as appropriate.
2. Value - To fine tune the actual value you can change the number in the Value text area or use the up and down buttons to increase or decrease the value.
3. Trim - If the servo moves a little even when it is not supposed to be moving increase or decrease the trim value. This provides a bias to the servo value to ensure that it is correctly centered and behaves as expected.
4. Variables - To control the servo automatically select an appropriate variable that contains or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your [VBScript](#) (using the SetVariable function) or [Plugin](#) based program
5. Min/Max - You can also use the min/max limits to ensure that even if the variables or you manually specify too large or low values (due to accidental programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
6. Stop Button - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position. Note that when stopped the interface adapts a red color to indicate the interface is stopped. Press the same button (now called Start) to re-enable the interface.

See Also

- [Phidgets Motor Control](#)
- [Lynxmotion SSC](#)
- [Parallax SSC](#)

Pololu Maestro

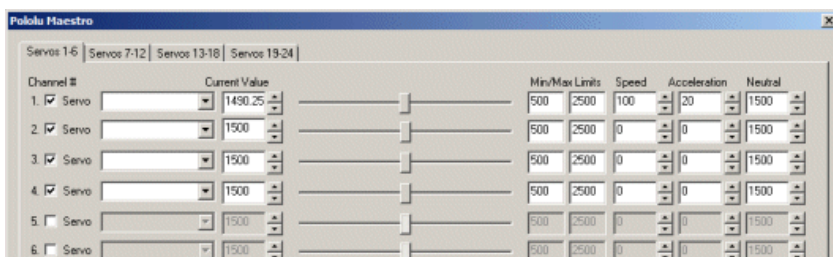


The [Pololu Maestro](#) is a very compact solution for controlling up to six radio control (RC) servos from a computer or microcontroller. The Pololu Maestro RoboRealm module provides a way to interface the visual processing of RoboRealm into servo movements using the Pololu Maestro board.

While the Pololu Maestro is compatible with the MiniSSC II protocol the RoboRealm module provides access to the more advanced features available in the Pololu specific protocol such as servo speed, resolution, neutral point, increased range and access to multiple servo banks. The Maestro module within RoboRealm uses the USB protocol to avoid the confusion of which COM port to select as well as to expose the more advanced capabilities of the board.

Note that the interface supports up to 24 servos to accommodate both the Micro and Mini Maestros line of products. Thus, if using the Micro Maestro servos 7-24 will simply not work as the board only supports up to 6 servos.

Interface





Instructions

1. Enable - To enable/disable using a particular servo you can click on the checkbox next to each channel. A disabled servo will not be sent any signals.
2. Servo/Input/Output text - indicates the mode that the particular channel is set to. Input mode is read only and will disable all controls except for variable selection which will then hold the input value. Output and Servo will produce those values on the appropriate pins. To change the mode you will have to use the Pololu Maestro Control Center application. Be sure to press Apply All in order to send the changes to the Maestro. Note that you will have to exit RR in order to run that application and vice versa.
3. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.
4. Current Value - To manually set the servo position type in the appropriate number (500-2500, 1500 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 500 and 2500 will most likely be beyond the capabilities of your servos so you may want to tighten the range.
5. Sliders - You should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box. If the servos do not move check your USB cable and/or board power connections.
6. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
7. Speed - You can set the servo speed using the provided text box. Using 0 will move the servo as quickly as possible. Setting it to 1 will move the servo very slowly to the destination position. Using 127 will move it quite quickly to the appropriate position.
8. Acceleration - You can set the servo acceleration using the provided text box. Using 0 will accelerate the servo as quickly as possible. Setting it to 1 will accelerate the servo very slowly to the top speed. Using 127 will accelerate it quite quickly to the appropriate speed.
9. Neutral - If your base servo position is different from 1500 you can change the neutral position of the servo by adjusting the neutral number.
10. STOP - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.
11. Connection - In most cases when connected via a USB cable the Maestro will be functioning over USB. In those cases where you need to control the Maestro over a long distance a serial cable will fair better than a USB one. In that case a serial connection from the PC to fed into a serial to TTL converter that is then connected to the Maestro. Because the protocol is now different, you can switch the module to use serial communication instead of USB communication. When doing so, be wary of the baud rate. It will have to be set to a level that is compatible with your USB to serial (from the PC) and the serial to TTL cable (into the Maestro). 9600 baud will probably work but that will cause the servo to have a choppy movement. Faster baud rates are recommended.

Variables

POLOLU_MAESTRO_MOVING - equals 1 when any servo is moving, 0 when all have stopped (or failed to reach their target)

X_MOVING - equals 1 when the servo associated with the variable X is moving. For example the presence of the variable pan_MOVING would mean that the servo associated with the variable 'pan' is still moving towards whatever the value within the variable 'pan' is.

See Also

[Pololu SSC](#)

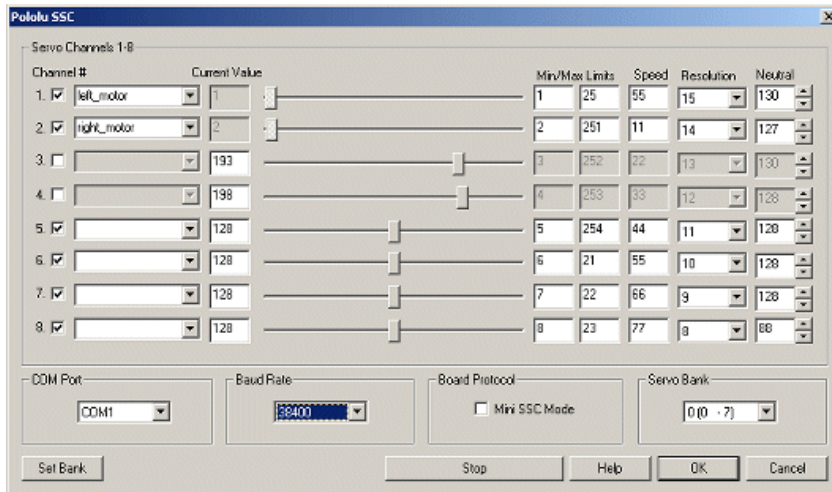


The [Pololu micro serial servo controller \(SSC\)](#) is a very compact solution for controlling up to eight radio control (RC) servos from a computer or microcontroller. The Pololu RoboRealm module provides a way to interface the visual processing of RoboRealm into servo movements using the Pololu SSC board.



While the Pololu SSC is compatible with the Mini SSC protocol the RoboRealm module provides access to the more advanced features available in the Pololu specific protocol such as servo speed, resolution, neutral point, increased range and access to multiple servo banks.

Interface



Instructions

1. COM Port - Select the appropriate COM Port. You will only see COM ports that are recognized by your computer.
2. Baud Rate - Select the appropriate baud rate. 38400 is the default speed used to provide fastest response. Select a lower baud rate if you have problems communicating. Note that when in Mini SSC mode 9600 baud is the fastest rate you can use.
3. Mini SSC Mode - Select if you want to use the Mini SSC mode (checked) or the Pololu native mode (unchecked). This will need to correspond to the state of the protocol selection jumper (off for native Pololu mode, on for Mini SSC mode.) If the jumper and the selection in this user interface do not match the SSC will not work correctly.
4. Servo Bank - Select which servo bank should be accessed. The Servo bank allows more than one SSC board to be controlled from the serial port. To add a second board on bank 1 (8-15) remove all but that board from the serial connection. Select the bank 1 in the dropdown interface. By pressing the "Set Bank" button RoboRealm tell the servo board to respond to the second bank servo commands. You will have to reset the SSC board by powering off and then on again.
You can then reattach any other SSC boards. By keeping the same bank selected the module will only talk with the appropriate servo. By adding more than one SSC module into the RoboRealm pipeline and selecting the correct servo bank you can control up to 128 separate servos.
5. Sliders - After specifying the COM Port you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Channels area. If the servos do not move check your COM Port setting and/or board connections.
6. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.
7. Current Value - To manually set the servo position type in the appropriate number (0-255, 128 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Be sure to select the correct COM port that you are using to control the SSC.
8. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
9. Speed - You can also set the servo speed using the last text box. Using 0 will move the servo as quickly as possible. Setting it to 1 will move the servo very slowly to the destination position. Using 127 will move it quite quickly to the appropriate position.
10. Resolution - If you need finer control over a smaller turning range of the servo reduce the resolution number towards 1. At 15 you will still have about 180 degree control, as you approach 1 the range will allow for finer adjustments to the position by reducing the total range of the servo.
11. Neutral - As you increase the resolution the range reduces. Using the neutral setting will adjust the absolute middle position of the servo such that the total range will now focus around that point. This allows you to move the servo to a set position anywhere along the 180 degree turn and

that the total range will now focus around that point. This allows you to move the servo to a set position anywhere along the 100 degree turn and then control the movement within a very precise but limited range.

12. STOP - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.

13. Enable - To enable/disable using a particular servo you can click on the checkbox next to each channel. A disabled servo will not be sent any signals and will be turned off. This allows you to manually move the servo as needed.

See Also

[SSC](#)

Robotis Dynamixel



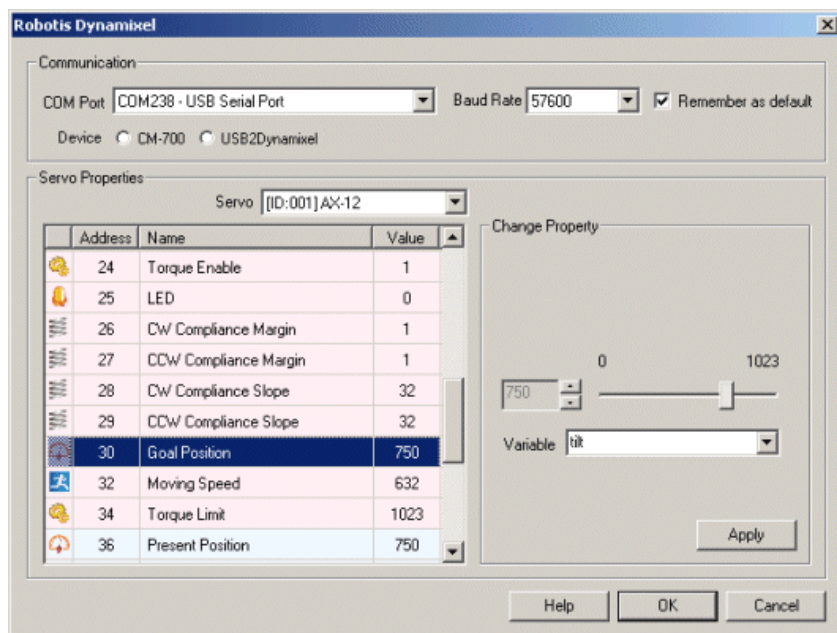
The Robotis Dynamixel module provides an interface to the [Dynamixel servos](#) made by [Robotis](#). The Dynamixel series robot actuator is a smart, modular actuator that incorporates a gear reducer, a precision DC motor and a control circuitry with networking functionality, all in a single package. The RoboRealm Dynamixel module provides a similar interface to the servo as the RoboPlus application provides but also incorporates RoboRealm variables into the module for automatic control.

The module will save all configuration specified to a particular servo and reproduce those changes once activated. This in combination with exposing variables for various servo attributes provides a very accessible control interface to the many capabilities of the Dynamixel servos.

Note that the module expects communication with a CM-700 CPU board set in Monitor mode or the USB2Dynamixel device. No application uploads to the CM-700 board is needed. The module uses the built in Actuator control program.

The module provides an easy to use GUI interface to the [RoadNarrows Graboid Series D](#) gripper. The Graboid is an ideal solution for those looking for a simple, low-cost gripper with force feedback for their robotic application. Built around 2 Robotis Dynamixel AX-12A Actuators, the gripper offers two degrees of freedom, wrist up/down, and gripper open/close.

Interface



Instructions

1. Communication - Select the appropriate COM port that the CM-700 or USB2Dynamixel device is connected to.

2. Device - Select which device you have to control the dynamixel servos.

3. Servo - Once the appropriate COM port has been selected you will see all the connected servos listed in the Servo dropdown. Selecting a servo will update the interface to show that servo's information.

4. Servo Properties - You can then select properties of the servo which will produce an interface to the right of the property list that allows you to change the property configuration. Note that once change you MUST press the APPLY button otherwise the change will not be transmitted to the

The following table (reproduced from [Robotis Support Website](#)) outlines the different properties and their meaning with respect to the Dynamixel.

Address	Name	Description
0	Model Number(L)	Lowest byte of model number
1	Model Number(H)	Highest byte of model number
2	Version of Firmware	Information on the version of firmware
3	ID	ID of Dynamixel
4	Baud Rate	Baud Rate of Dynamixel
5	Return Delay Time	Return Delay Time
6	CW Angle Limit(L)	Lowest byte of clockwise Angle Limit
7	CW Angle Limit(H)	Highest byte of clockwise Angle Limit
8	CCW Angle Limit(L)	Lowest byte of counterclockwise Angle Limit
9	CCW Angle Limit(H)	Highest byte of counterclockwise Angle Limit
11	the Highest Limit Temperature	Internal Limit Temperature
12	the Lowest Limit Voltage	Lowest Limit Voltage
13	the Highest Limit Voltage	Highest Limit Voltage
14	Max Torque(L)	Lowest byte of Max. Torque
15	Max Torque(H)	Highest byte of Max. Torque
16	Status Return Level	Status Return Level
17	Alarm LED	LED for Alarm
18	Alarm Shutdown	Shutdown for Alarm
24	Torque Enable	Torque On/Off
25	LED	LED On/Off
26	CW Compliance Margin	CW Compliance margin
27	CCW Compliance Margin	CCW Compliance margin
28	CW Compliance Slope	CW Compliance slope
29	CCW Compliance Slope	CCW Compliance slope
30	Goal Position(L)	Lowest byte of Goal Position
31	Goal Position(H)	Highest byte of Goal Position
32	Moving Speed(L)	Lowest byte of Moving Speed
33	Moving Speed(H)	Highest byte of Moving Speed
34	Torque Limit(L)	Lowest byte of Torque Limit
35	Torque Limit(H)	Highest byte of Torque Limit
36	Present Position(L)	Lowest byte of Current Position
37	Present Position(H)	Highest byte of Current Position
38	Present Speed(L)	Lowest byte of Current Speed
39	Present Speed(H)	Highest byte of Current Speed
40	Present Load(L)	Lowest byte of Current Load
41	Present Load(H)	Highest byte of Current Load
42	Present Voltage	Current Voltage
43	Present Temperature	Current Temperature
44	Registered	Means if Instruction is registered
46	Moving	Means if there is any movement
47	Lock	Locking EEPROM
48	Punch(L)	Lowest byte of Punch
49	Punch(H)	Highest byte of Punch

Example



[Click Here](#) to download a robofile that interfaces with the RoadNarrows Graboid device. It provides a simple button interface to open and close the gripper, and tilt the wrist of the device. Note that IDs 1 (wrist) and 6 (gripper) are used.



The SOR_Axon module provides a GUI interface to the Axon Microcontroller in conjunction with the Axon .hex program that can also be downloaded [here](#) or for the Axon II [here](#). This application provides a quick way to use the Axon MCU to control up to 29 servos without having to write any code for the Axon.



Once you have added power and connected the USB cord to the Axon you can download the above file that includes the source .c file and the entire AVRStudio project file for the Axon servo controller program. Note that RoboRealm uses the specified protocol in that file to communicate in a reliable manner with the Axon MCU to control all 29 servos.

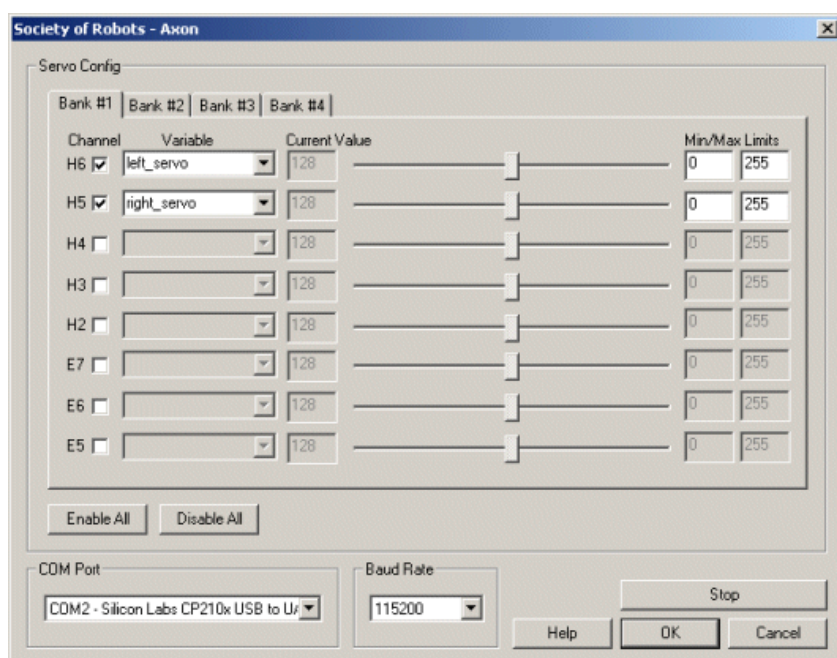
Once you have downloaded and unzipped the above file you will have to locate the Default folder which contains the file Servo_Controller.hex which is the file to be uploaded to the Axon MCU. You will also notice the upload.bat file in that folder. You can edit that file and change the port number in the file to make it easy for you to update the Axon. The parameter -c2 is for the COM port. You will have to update that to 1,2,3,4 based on your system. If your assigned COM port for the Axon is assigned higher than 4 (this happens frequently) you will have to use force the port number to one of those as the boot loader program does not work with any COM ports higher than 4.

To force your COM port:

1. Right click My Computer and select Manage.
2. Click on Device Manager in the interface that opens
3. Open the Ports (COM & LPT) tree
4. Right click on Axon USB which normally reads "Silicon Labs CP210x USB to ..." and select Properties.
5. Click on Port Settings tab.
6. Select Advanced button
7. Select an appropriate COM Port Number using the provided dropdown menu.
8. Click ok until all dialogs are gone.

Once you are ready you should just be able to open up a command console (Start Button->Run->type in "cmd") and cd to the Default folder that you unzipped. Switch off you Axon, then run upload.bat, then switch on your Axon. That should hopefully upload the servo controller program to the Axon. Once this is complete you can use this module to communicate to the Axon and control one of 29 servos. Please see [Society of Robots](#) if you have issues with how to upload programs to your Axon.

Interface



Instructions

1. COM Port - Select the COM port that the Axon is connected to on your computer. Note that you will only see COM ports that are recognized by your computer.
2. Baud Rate - Select the appropriate baud rate. Note that 115200 is the default and should not be changed unless required.
3. Checkbox - To enable/disable a particular servo channel select the checkbox to change its state. When disabled the servo will not be sent any further position commands and can be manually moved.
4. Scroll Bars - After specifying the COM port and speed you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Config area. If the servos do not move check your COM port setting and/or board connections. You should see a small green light flash on the Axon board when any serial communication is present. If you do not see this light

flashing when moving the scroll bar around check your connections. If you do see a flashing light with no movement check the channel you plugged the servo into.

5. Variable - Select an appropriate variable that contains or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program.

6. Min/Max Limits - You can use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

8. Bank # - Select the appropriate bank of servos that you would like to configure. Note the servo port number which corresponds to the port number of the Axon board. Only eight port are visible at any time.

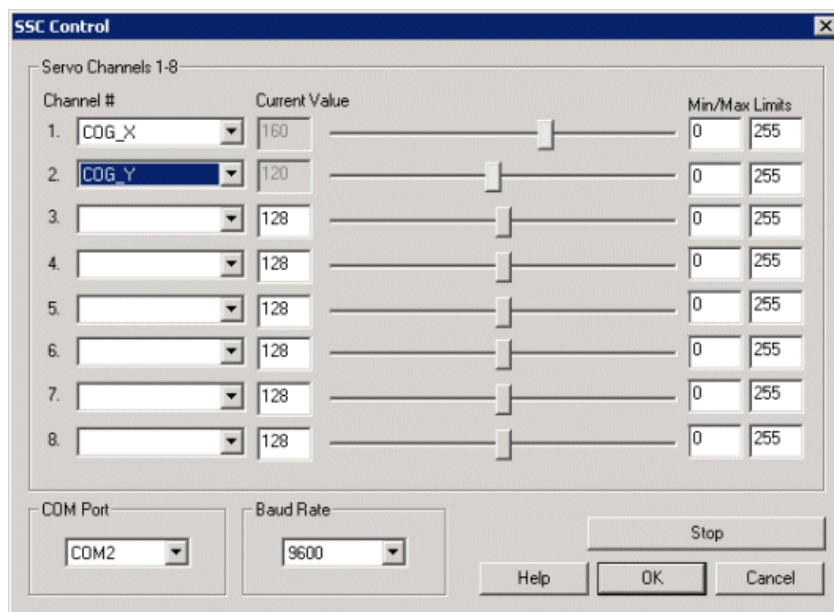
9. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.

Note that if you set a servo channel to be positioned using a variable's value you will lose manual control over that servo. Changing the variable value, however, will change that servo's position.

SSC

The SSC control interface allows you to interface RoboRealm to the Serial Servo Controllers made by [Scott Edwards Electronics](#). The SSC supports 8 control channels which are represented by the 8 dropdown lists.

Interface



Instructions

1. COM Port - Select the appropriate COM Port. Note that you will only see COM ports that are recognized by your computer.

2. Baud Rate - Select the appropriate baud rate. Note that 9600 is the default. Select 2400 if your SSC does not support the 9600 baud rate.

3. After specifying the Comm Port you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Channels area. If the servos do not move check your Comm Port setting and/or board connections.

4. Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program. You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

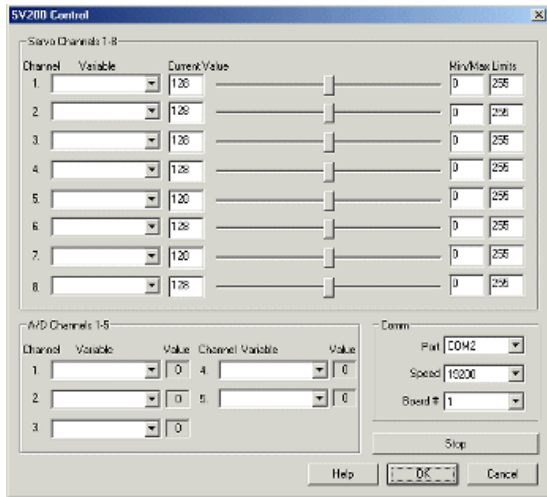
5. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.

To manually set the servo position type in the appropriate number (0-255, 128 is the default) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Be sure to select the correct COM port that you are using to control the SSC.

SV200

The [SV200 or SV203](#) control interface allows you to interface RoboRealm to servos via a motor controller made by [PONTECH](#). The SV203 supports multiple boards each with 8 servo control channels and 5 Analog to Digital inputs.

Interface



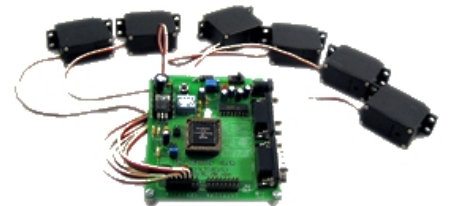
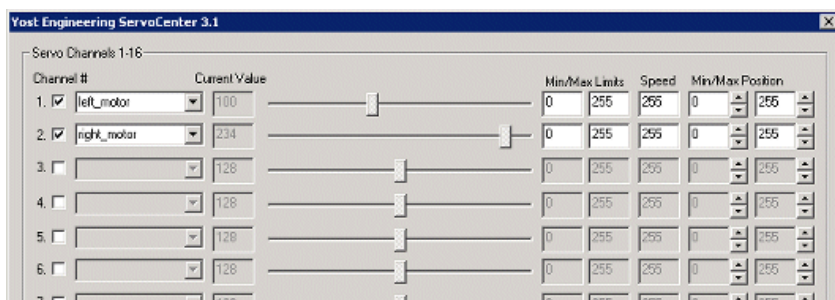
Instructions

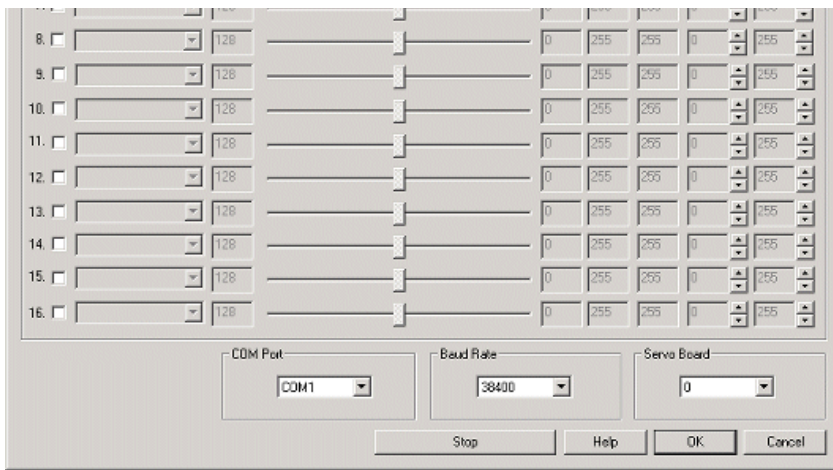
1. Select the appropriate Port, Speed and Board number in the Comm group. SV203 supports up to 255 boards but only 32 are specified in the RoboRealm dropdown.
2. After specifying this information you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Channels area. If the servos do not move check your Comm settings and/or board connections.
3. Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program. You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
4. Select the appropriate variables that will contain the inputs from the AD converter on the board. The variables can then be accessed using the VBScript or other extension programs to perform appropriate actions.
5. Press STOP if you need to quickly disable all the servos and return them to the middle or neutral position.

Yost Engineering ServoCenter™ 3.0

The ServoCenter™ board (no longer available!) provides for the control of up to 16 servos per board. Each board can be separately identified and serially chained to control up to a total of 256 servos. The ServoCenter offers seek speed and scaled resolution for each individual servo. This allows you to move one servo to a position slowly, while another is moving to a different position at a faster speed.

Interface





Instructions

1. **COM Port** - Select the appropriate COM Port that the ServoCenter is connected to. You will only see COM ports that are recognized by your computer.
2. **Baud Rate** - Select the appropriate baud rate. 9600 is the default speed used to provide fastest response. Select a higher baud rate if you want to increase the communications speed. Be aware of the Jumper settings for JP2 as they specify the baud rate. Be sure to select a baud rate in the interface that corresponds with the jumper settings. Note that the board comes default to 9600 with no jumpers on.
3. **Servo Board** - Select the id of the ServoCenter board that you want to communicate to. Note that the board comes default with id 0.
4. **Sliders** - After specifying the COM Port you should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box in the Servo Channels area. If the servos do not move check your COM Port setting and/or board connections.
5. **Variables** - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program.
6. **Current Value** - To manually set the servo position type in the appropriate number (0-255, 128 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Be sure to select the correct COM port that you are using to control the ServoCenter.
7. **Min/Max Limits** - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
8. **Speed** - You can also set the servo speed using the next text box. Using 255 will move the servo as quickly as possible. Setting it to 1 will move the servo very slowly to the destination position. Using 127 will move it quite quickly to the appropriate position.
9. **Min/Max Position** - If you need finer control over a smaller turning range of the servo increase the minimum position or decrease the maximum position to confine the range in which the servo will move. At 0 and 255 you will have about 180 degree control. If you use 0 and 128 or 128 and 255 you will reduce the range to 90 degrees but with twice the resolution. Note that using 0 and 128 will restrict the movement in the first half of the total range whilst using 128 and 255 will restrict it to the second half of the total range. In this way you use the min/max positioning to reduce the range of the servo AND also specify the range position.
10. **STOP** - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.
11. **Enable** - To enable/disable using a particular servo you can click on the checkbox next to each channel. A disabled servo will not be sent any signals and will be turned off. This allows you to manually move the servo as needed. Once you enable it again the servo will return to its last position before being disabled.

cmRobot Element

The cmRobot_Element module provides an interface from RoboRealm to the [cmRobot Element](#) robot controller. The module provides an interface to configure the appropriate communication port and speed to the controller and allows you to view and modify settings by using the GUI based sliders and text editing areas. To automatically specify the values you should select variables from the dropdown menus that will contain the values that will be sent to the robot. Note that once variables have been selected manual controls are disabled.



Interface

ENCODERS

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

GPIO	Variable	Trim	Current Value	Min/Max Limits	
8		0	0	-99	100
9		0	0	-99	100
6		0	0	-99	100
7		0	0	-99	100
4		0	0	-99	100
5		0	0	-99	100

Communication

COM Port: COM10 - Standard Modem over Bluetooth link | Baud Rate: 19200 | Remember as default

Stop | Help | OK | Cancel

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

GPIO	Variable	Bit	Input	Output	GPIO	Variable	Bit	Input	Output
0			<input type="checkbox"/>	<input type="checkbox"/>	5			<input type="checkbox"/>	<input type="checkbox"/>
1			<input type="checkbox"/>	<input type="checkbox"/>	6			<input type="checkbox"/>	<input type="checkbox"/>
2			<input type="checkbox"/>	<input type="checkbox"/>	7			<input type="checkbox"/>	<input type="checkbox"/>
3			<input type="checkbox"/>	<input type="checkbox"/>	8			<input type="checkbox"/>	<input type="checkbox"/>
4			<input type="checkbox"/>	<input type="checkbox"/>	9			<input type="checkbox"/>	<input type="checkbox"/>

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

Pin	Variable	Value
0		0
1		0
2		0
3		0
4		0

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

#	Variable	Trim	Current Value	Min/Max Limits	
1		0	0	-99	100
2		0	0	-99	100

Use PIDL | Kp: 0 | Ki: 0 | Kd: 0 | Kf: 0

Stinger
 Traxster
 Custom

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

Enable

#	Variable	Current Value
1		0
2		0

Single Encoder Type
 Quadrature Encoder Type

Reset

cmRobot Element

Servos | DIO | Analog | Motors | Encoders | Other

Battery (Input)

Leds

Led	Variable	Rate
Led1		0
Led2		0

Current Value:

Instructions

1. COM Port - Select the appropriate COM Port that the Element is connected to. You will only see COM ports that are recognized by your computer. Note that direct serial connections normally use com ports lower than 4 whilst virtual COM ports created by bluetooth connections are much higher. Check your bluetooth configuration manager to see which port your robot communications is installed on. Note that this will be an Outgoing port.
2. Baud Rate - Select the appropriate baud rate. 19200 is the default speed used. Select a higher baud rate if you want to and can increase the communications speed. Be aware that wireless devices may not support the full baud range of the Element board. If you accidentally set the baud speed to something faster than your wireless device supports you will have to connect the Element directly to your PC in order to change the baud rate to a lower setting.
3. Battery - once the Element is communicating with RoboRealm you can turn on the battery monitor in the Other tab to test the connection. Do this by simply selecting the Battery area checkbox in the interface. The power level will be shown in blue bars with the number of volts in the "Current Value" textbox. If you want to use the volts within other areas of RoboRealm select a variable that the module will write the number of volts to. This variable will then be updated each time the Element module is run with the current voltage of the robot.
4. Motors - After specifying the COM Port you should be able to move your motors by dragging the sliders to the right or left or by specifying a number within the "Current Value" text box in the Motors area. If the motors do not move (or whine) check your COM Port setting and/or board connections.
5. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the robot. This is used to automatically change the motor values based on your VBScript (using the SetVariable function) or Extension based program. See [Variable Control](#) for more information on how to programatically move the robot.
6. Current Value - To manually set the motor position type in the appropriate number (-100 to 100, 0 is the default neutral/stop) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate.
7. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the robot does not actually attempt to move the motors above or below the specified limits. This can be used as an additional precaution in case you are testing your robot in a precarious position.
8. Servos - Using a similar interface as the motors you can specify the position for up to 6 servos. Once you do so you will notice one of the Digital IO pins become disabled. This is due to the servo pin and IO pin being the same pin. While there are 10 IO pins only 6 can be used to control servos. Note that additional power supply may be needed if you are using 6 servos.

You can use the slider bar to move the servo or type in a number directly in the "Current Value" textbox to move the servo to that position. To move the servo automatically select a variable that contains or will contain the value to move the servo to.
9. Analog Pins - The Element provides for 5 user analog inputs. (There is actually one more but that is tied to the battery). If you enable the Analog pin you will notice the values in the corresponding read only text boxes change based on the voltage across those pins. If you have a distance sensor connected to an analog pin move your hand in front of the sensor to see these values change. By specifying a variable in the corresponding dropdown you can access that value elsewhere in RoboRealm in order to make decisions about the sensors value.
10. Encoders - To enable the encoder area click on the group checkbox. This will enable the encoder readings and display the number of encoder ticks that the robot has traveled. If you enable this area and then move the robot using the Motor sliders you will see the encoder values change depending on how fast the robot is moving. Specifying variables in the dropdown will cause those variables to be set to the current values read from the robot. Also note the "Reset" button that can be used to reset the encoder values on the robot to zero.
11. Digital IO - In addition to the 4 analog inputs the Element provides 10 digital input/output pins. Note that if you are using servos some of those pins will already be used. The remaining pins can be used to drive outputs or receive inputs.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output you can manually control the output by selecting the checkbox which will then turn that pin high or low.

For input pins the checkbox (in a disabled state) will reflect the high or low states of the pin.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown. If no bit is specified the entire value is considered during a set/output.

As an experiment, you can connect an LED to pin 0 in the Element. Then select the pin as an "out" and by checking and un-checking the checkbox you can make the LED blink under manual control. If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.
12. Direct Serial Command Variable - As the Element has many additional capabilities not reflected in the GUI the direct serial command variable will send the text contained in the specified variable to the Element and place the returned results back into that same variable. Thus you can specify

will send the text contained in the specified variable to the Element and place the returned results back into that same variable. Thus you can specify a variable (or type a new one) into the provided textbox and set the value to any of the serial commands that the Element understands. For example, you can use

```
SetVariable "my_command", "i2c r 4 1"
```

in a VBScript module to query the line following sensor.

13. Stop - press the stop button to quickly stop the motors. The motors controls will then be disabled (in case variables are selected) and will only restart when you click the same button again (now renamed to "Start").

Conrad C-Control PRO Mini-Station

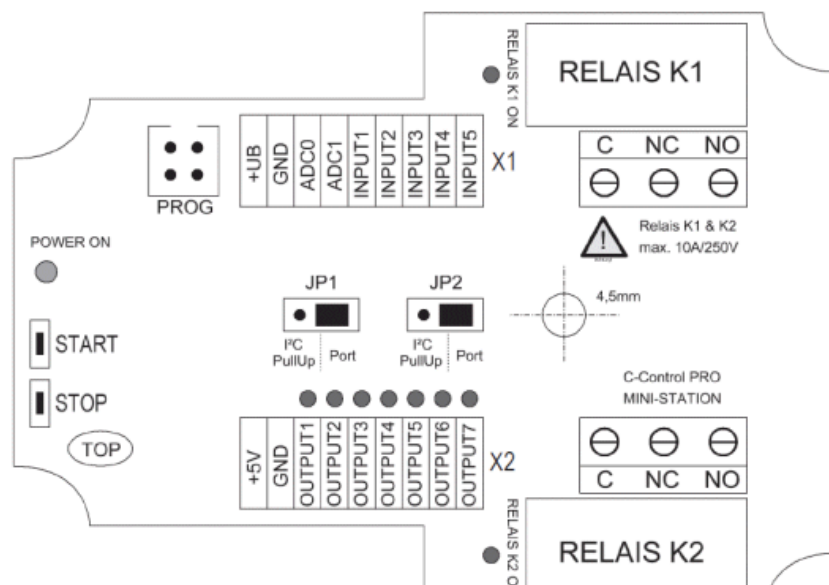


The Conrad Mini-Station module provides an interface from RoboRealm to the Mini-Station over a serial connection. The Mini-Station (distributed by [Conrad](#)) provides high current relays, digital in and out and analog to digital capabilities. The housing is extremely compact and is also dust and splash-proof till IP66, as such, it can be used in rough conditions. The Mini-Station can be programmed in Basic, Compact-C and Assembler.

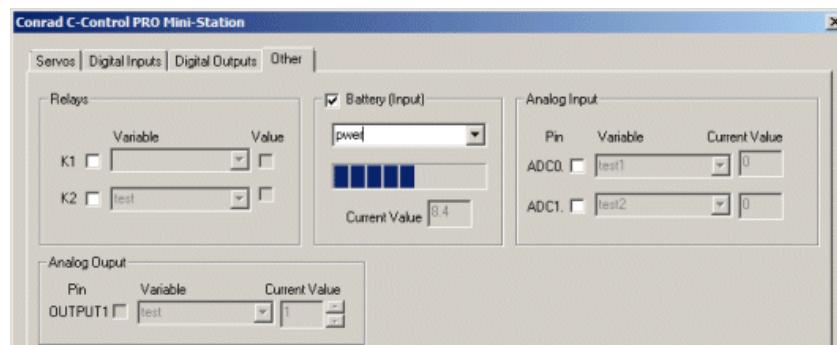
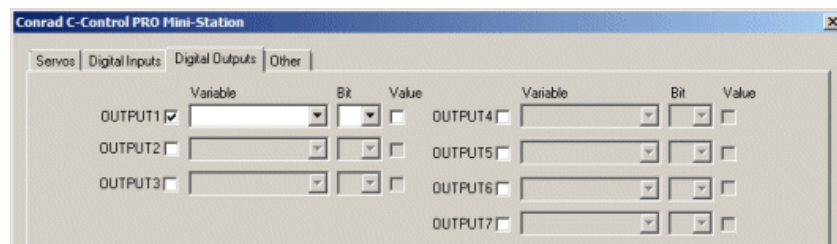
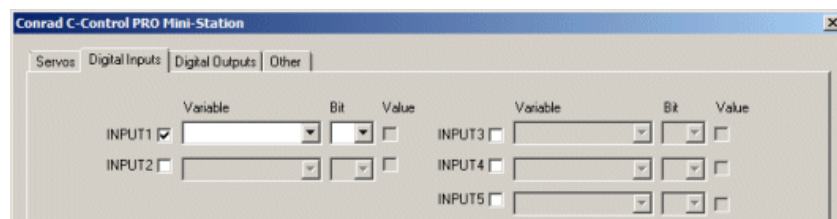
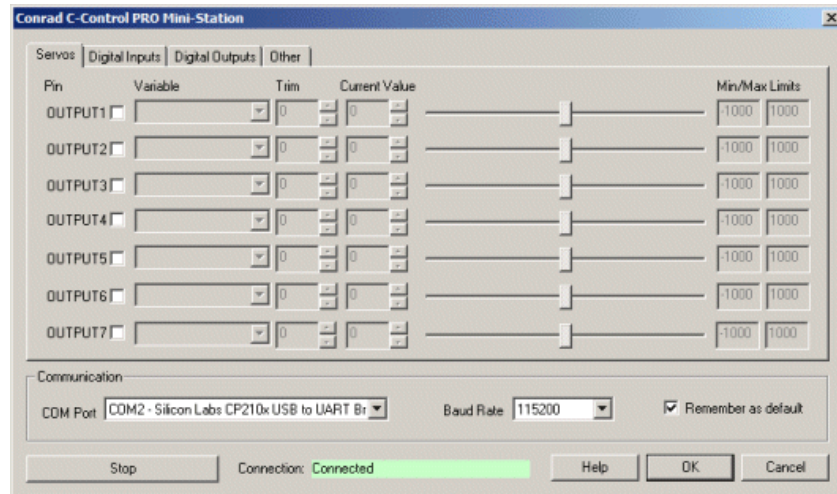
The module assumes that the following [code](#) has been uploaded via the C-Control IDE into the Mini-Station. This code is needed as it provides the interface to the RoboRealm module via a serial to USB interface. If this code is not uploaded to the Mini-Station this module will not function correctly.

Connection Diagram

This module makes assumptions about what pins are called what. When reviewing the interface below, the following diagram represents the location of the specified pins. Note that as the Mini-Station provides many kinds of functionalities, some pins (esp. the OUTPUT1 pin) is used across different functions. Should you notice that you cannot enable a particular pin that is most likely because it is being used elsewhere. As a single pin cannot be used more than once, the interface will reflect that.



Interface



Instructions

1. COM Port - Select the appropriate COM Port that the Mini-Station is connected to. You will only see COM ports that are recognized by your computer. Note that direct serial connections normally use com ports lower than 4 whilst virtual COM ports created by USB to Serial connections are much higher.
2. Baud Rate - Select the appropriate baud rate. 115200 which is the default speed used by the onboard code linked to above. If you modify the firmware code to change the baud rate you can select a different rate

miniature code to change the baud rate, you can select a different rate.

3. Digital Inputs - The Mini-Station provides 5 digital input pins. The checkbox (in a disabled state) will reflect the high or low state of the pin.

To automatically receive a bit select or type in a variable that will be set. Note that you can tell RoboRealm which bit of the variable you want to set by using the provided bit dropdown. If no bit is specified the entire value is considered during a set/output.

4. Digital Outputs - The Mini-Station provides 7 digital output pins. Note that if you are using servos those pins will already be used. The remaining pins can be used to drive outputs.

To automatically send a bit select or type in a variable that will contain the bit to set. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown. If no bit is specified the entire value is considered during a set/output.

As an experiment, you can select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 6 the LED associated with that output port will blink for every six frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

5. Battery - once the Mini-Station is communicating with RoboRealm you can turn on the battery monitor in the Other tab to test the connection. Do this by simply selecting the Battery area checkbox in the interface. The power level will be shown in blue bars with the number of volts in the "Current Value" textbox. If you want to use the volts within other areas of RoboRealm select a variable that the module will write the number of volts to. This variable will then be updated each time the Element module is run with the current voltage of the robot. Note that the volts are multiplied by 10.

6. Analog Input - The Mini-Station provides 2 user analog inputs. (There is actually one more but that is tied to the battery). If you enable the Analog pin you will notice the values in the corresponding read only text boxes change based on the voltage across those pins. If you have a distance sensor connected to an analog pin move your hand in front of the sensor to see these values change. By specifying a variable in the corresponding dropdown you can access that value elsewhere in RoboRealm in order to make decisions about the sensors value. The values will range from 0 to 1023 corresponding to 0 to 5Volts.

7. Relays - The Mini-Station provides 2 relays that can be activated by selecting the checkbox or by selecting a variable that will contain the value to switch off and on the relay. Note that there is an audible click and a red led that comes on/off corresponding to the state of the relay.

8. Analog Output - The Mini-Station can produce a PWM signal on OUTPUT1 which can be used to create an analog voltage on that pin using the Timer0 within the Mini-Station. Note that if this is enabled, the pin is no longer available for servo or output functionality.

DMP RoBoard



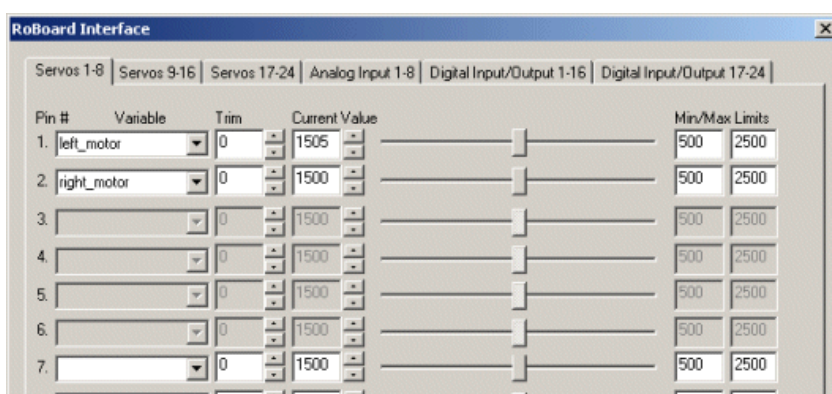
The RoBoard module provides an interface to the intrinsic control capabilities of the RoBoard nano sized PC based computer board. This miniature computer can run RoboRealm (assuming WinXP is installed) and provides a shared 24 PWM servo or digital IO pins and 8 analog in pins (not to mention the ability to attach a USB webcam!). This PC was built for embedded robotics and offers a great solution for onboard processing with RoboRealm.

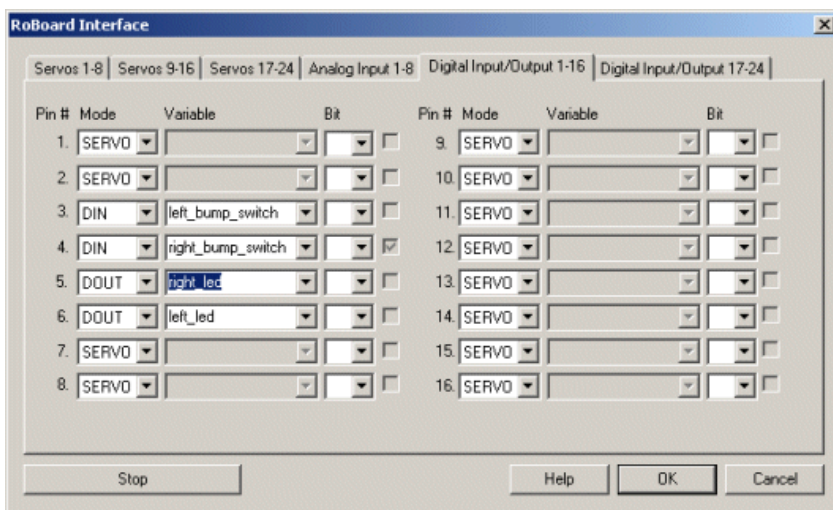
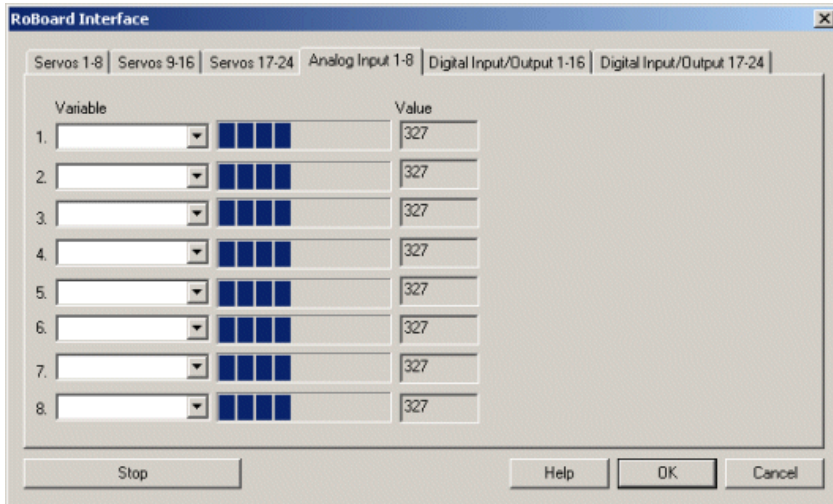
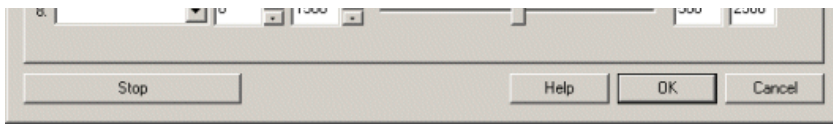
The RoboRealm module provides easier access to the boards capabilities by exposing the pin configuration in an easy to use GUI format that is run right on the board itself. The module allows for vision processing to directly affect the robots motors, servos, etc.

Note that the module will show on other computers than the RoBoard but will of course not reflect the correct values nor move any servos!

Depending on your installation you will need the Microsoft VC8.0 redistribution files in order for the RoBoIO.dll to run (this DLL provided by RoBoard). You can download these zipped files [here](#). Download and unzip directly into the RoboRealm folder if you find that the RoBoard module does not move any servos when it should.

Interface





Instructions

1. Servos - Select the appropriate tab that indicates the range of servos that you want to work with. Then test the servo out by using the slider to change the servo position or use the text box/spin buttons to change the values with more precision. Moving the slider or changing the current value should move the servo. If it does not then check the pin number and the servo condition.

To automate the movement of the servo you can select or type in the name of a variable that contains the position value to be sent to the servo. This variable can be created by other modules or set by external applications via the [API](#). Note that the slider and value fields are disabled when a variable is selected to indicate that the servo is now under automatic control and cannot be manually set. Removing the variable will again allow manual changes.

If you find that the servo drifts even at 1500 (neutral) then change the trim to stabilize the servo. In this way your application can still consider 1500 to be neutral without needing to change the number.

To protect your servos from damage you can change the Min and Max limits which will prevent values beyond the specified range from being sent to the servo incorrectly. This can also be used to limit the range of the servo if it physically cannot move within its full range.

Note that a Servo control will be disabled if the Pin is instead used as a General Purpose Digital Input/Output pin instead (GPDI/O).

2. Analog Input - The analog input tab shows the current values as registered by the Analog input pins. The value will range from 0 to 1024 (10 bit value). To react to these values type in or select a variable that will be assigned the current value present at the pin. Note that 5V produces a maximum value of 1024.

3. Digital Input/Output - Select the mode that you want to assign to the particular pin. All pins are by default set as Servo pins. By changing the mode you can instead use the pin as a digital input or digital output.

When you select the DOOUT (Digital Output) mode the variable and checkbox become enabled with the mirrored Servo control being disabled (i.e. the servo control at the same pin). By clicking on the checkbox you can set/unset the pin value. By attaching an LED to the appropriate pin you should be able to manually switch on and off the LED. Note that DOOUT produces an approximate 3V out so most LEDs should handle that easily.

SHOULD BE ABLE TO MANUALLY SWITCH ON AND OFF THE LED. NOTE THAT DOUT PRODUCES AN APPROXIMATE 5V OUT SO MOST LED'S SHOULD HANDLE THAT EASILY.

Selecting a variable will then disable the checkbox to indicate that the DOUT pin is under variable control. The variable needs to contain either a non-zero value for the pin to be set high OR if the BIT field is selected then the specified bit in the variable will need to be set for the pin to be set high. For example, if the bit is set to 2 then the variable will have to contain 4 to 7 for the bit to be set high (i.e. bit number 2 or the 3rd binary bit needs to be set).

Selecting DIN (Digital Input) will allow you to type in a variable that will contain a 1 when the pin is drawn low. By default the pin will be 1 when the pin is not connected and a 0 when connected. If you wish to instead set a bit within the variable select the appropriate bit number in the dropdown box. The bit selection allows you to construct a single binary number from multiple DIN fields. To do this use the same variable name in multiple input fields with increasing bit numbers.

4. Stop - Press the stop button to center all servos and prevent any variable values from being sent to the servos. Note that this does NOT disable the DIN, DOUT or Analog in tabs.

Notes

1. It is worth mentioning a couple tricks when working with the RoBoard. First, you will need the VGA card for the initial installation of the operating system, but after this you can remove the card and just use Remote Desktop to connect to the board. This just requires an Ethernet card (or wireless communication) in order to work with the RoBoard. Note that applications like VNC require the card to remain inserted as they are screen scrapers of which Remote Desktop is NOT. To enable Remote Desktop right click on the My Computer icon, select the Remote tab and check the "Allow uses to connect remotely ...".

2. Be VERY careful when changing values for the min/max for the servos. It is possible to change the values and send a very low or high value to the servos. The RoBoard is capable of sending a position value of 1 (much less than the 500 boundary) which can ruin a servo (this based on experience!).

3. The black or negative PWM wire is of the outside side of the board with the signal (white or yellow) being towards the middle.

4. As the RoBoard is a regular PC all the other modules within RoboRealm such as the [Play Wave File](#) and [Speech Recognition](#) will work with the appropriate attachment of speakers, mic, etc.

5. As apposed to just turning the RoBoard off use the Windows Start button->Run and type in 'shutdown -t 1' which will cause the board to shutdown normally.

6. The RoBoard is currently (March 8th, 2010) the SMALLEST PC based board that we are aware of but along with that comes some performance issues. It is recommended that you use as small an image as possible (80x60 or 160x120) as the USB speed of the board is not like that of your laptop or desktop. Keeping the image small will allow a capture rate of about 15fps with the appropriate lighting. In our tests an image size of 320x240 slows this down to about 10fps.

MCU Communicator



The MCU Communicator module provides a way to transmit RoboRealm variables to Microprocessors that are connected via a serial (or usb to serial) interface to a PC. While RoboRealm supports many other hardware devices with dedicated interfaces the MCU Communicator provides an easier way to 'roll your own' in terms of sending information from the PC to the MCU and have the MCU react to that information. The module requires that the appropriate communications program be on the MCU that understands the module's requests. The onboard program is meant to be as small as possible without sacrificing functionality or reliability with the expectation that you integrate the communications portion within your own application.

The goal of the MCU gateway program was to provide a small but effective way of communicating between the PC and the MCU that allows for ease of use and flexibility in what you need to send or receive. Currently several MCU's have the supported application provided. These also provide examples as to how other MCU's may also be integrated. If you require these applications for MCU's that are not currently supported you can [Contact Us](#) to see if we can help out.

The interface provides for a total of 20 mailboxes. A mailbox is what is used to send or receive information based on the type of mailbox. You need to specify the name, type and direction of the mailbox. Keep in mind that the mailbox direction is with respect to RoboRealm. Thus a "get" means receive information from the MCU to the PC, whereas "send" means send information from the PC to the MCU.

As most MCU's are not capable of floating point numbers the term "Number" refers to an integer value. If you need to transmit a floating point number you should first multiply that number by some factor (like 1000) and then send the result as an number/integer.

Note that while 20 mailboxes are supported your MCU program can use fewer than 20 if memory space is a concern. You'd just have to be sure

not to use those mailboxes beyond the supported amount.

The protocol the MCU Communicator module provides some basic error checking in case you are using a wireless communication which can produce noise in the resulting data. The protocol also supports data streaming from the MCU so that unneeded data requests are avoided.

Gateway Applications

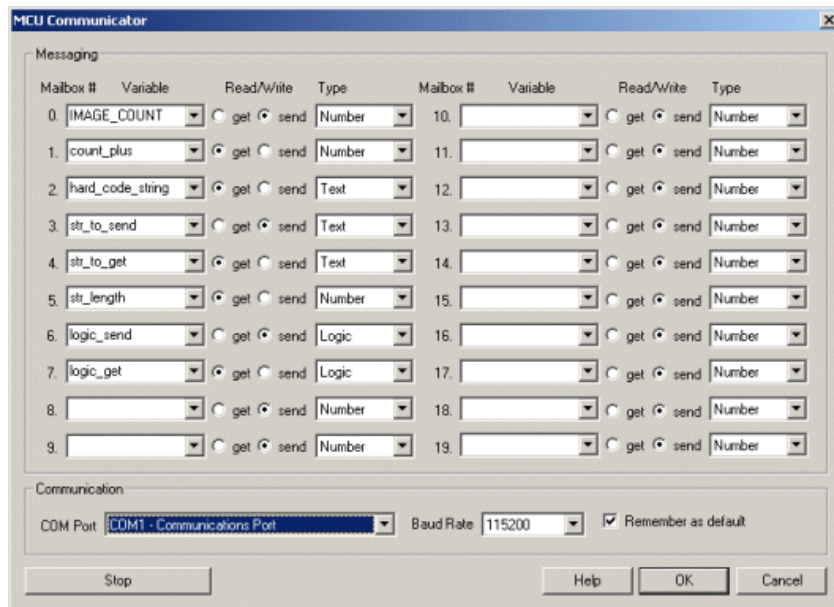
[Download Arduino Communicator](#)

[Download Orangutan SVP Communicator](#)

[Download SOR Axon Communicator](#)

[Download Parallax Propeller Communicator](#)

Interface



Instructions

1. Communication - Specify the COM port your device is connected to. Note that with usb to serial connectors they will most likely be higher than the physical COM1-4 ports.
2. Baud Rate - Specify the baud rate of 115200 to communicate with the MCU. Also be sure to support a baud rate that your MCU supports. If you are using a wireless connection you may need to slow down the baud rate depending on the supported rate of the device.
3. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the MCU module is loaded the COM port and BAUD rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.
4. Messaging - Specify the name of the variable you want to transmit. Note that for arrays you will only get the first element of the array. For elements beyond the first you will need to use the variable:index expression like blobs:2 to get the second element of the blobs array. Be sure to note the mailbox number as that number will be the array index within the MCU that will contain the value for that variable.

The MCU communicator supports 3 variable types. A number is a 32 bit integer number represented by 4 bytes. A logic type is a boolean 0 or 1 value represented by a single byte. A text value is a character string terminated by a zero with characters in the traditional ASCII range of 0-127. The maximum string length supported is 256 bytes.

Example

Within the provided gateway programs are examples on how to manipulate the variable contents and send them back to the PC. To test these examples run the following [robofile](#) in RoboRealm and specify the correct COM and baud settings. You can then use the [Watch Variable](#) module to see the changes to the variables.

IMAGE_COUNT - what image number you are on. Sent to the MCU
count_plus - should be IMAGE_COUNT+100. The 100 is added within the MCU
hard_code_string - a string sent back from the MCU that never changes
str to send - a string sent from the PC to the MCU what will be capitalized

str_to_get - the return string that should be capitalized
str_length - the length of the string received by the MCU to capitalize
logic_send - a logic value sent to the MCU
logic_get - the inverse of logic_send as performed by the MCU

See Also

[Serial](#)
[Sparkfun Arduino](#)
[Pololu Orangutan SVP](#)
[Society of Robots - Axon](#)

Pololu Orangutan SVP



The Pololu Orangutan SVP is an ideal controller for a mid to small sized robot. It provides many of the capabilities one would want on a robot including motors controllers, digital input and output, analog input, 2 line LCD screen and servo (PWM) control. All that functionality also comes at a very reasonable price! The SVP RoboRealm module provides an easy way to interface from your PC to this controller from RoboRealm.

The module is meant to allow RoboRealm to communicate with the SVP while connected via USB or bluetooth wireless link. It requires you to download the [AVR Studio C application](#) into the SVP board in order to facilitate communication with this module in RoboRealm.

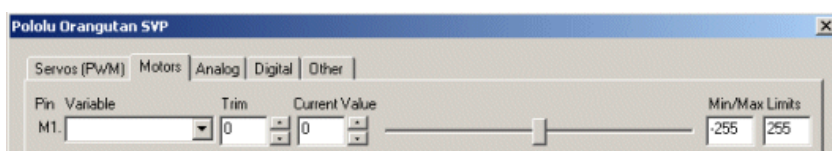
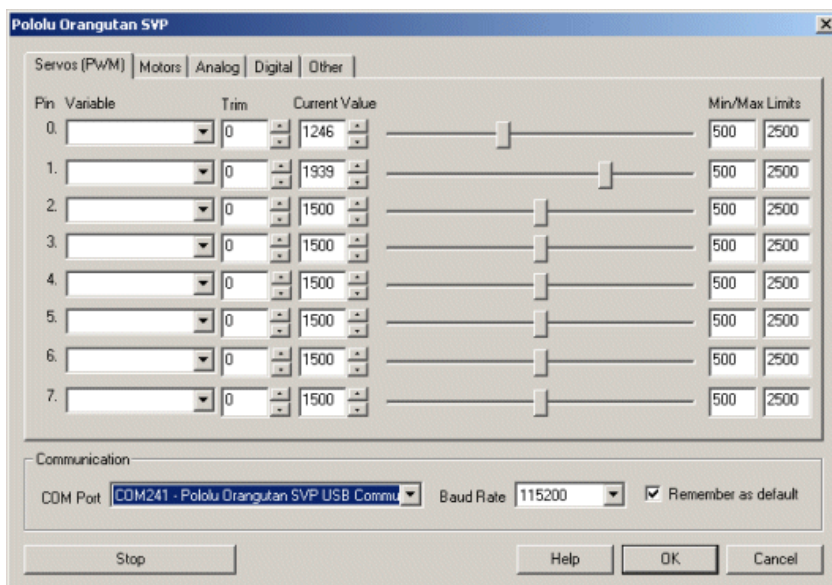
From there you can use the SVP as a servo controller, digital IO, input device, etc. from within RoboRealm by setting or reading the variables set by this module. If you have special requirements you can also modify the C program to do other processing (the link above is to the source code).

Naturally, you can roll your own RoboRealm SVP communication using the [Serial](#) module but having a stock module perform this means you can test things out quickly without needed to understand the internal programming of the SVP.

The communication protocol that the module uses to communicate with the SVP is a simplistic packet based protocol. The data transmitted is a 7 bit value with the 8th bit being a sync bit in case the SVP and PC become out of alignment. Only the header bit has the 8th bit set, all other bytes do not. This allows that when a new packet is to be processed that the first and only first byte has bit 8 set.

Values such as the analog signals are also passed back to the PC using the same structure also including a CRC to be sure that the bytes were received in order correctly.

Interface



M2

Pololu Orangutan SVP

Servos (PWM) Motors Analog Digital Other

Pin	Variable	Value	Pin	Variable	Value
A0	<input type="text"/>	0	A6	<input type="text"/>	0
A1	<input type="text"/>	0	A7	<input type="text"/>	0
A2	<input type="text"/>	0	A	<input type="text"/>	0
A3	<input type="text"/>	0	B	<input type="text"/>	0
A4	<input type="text"/>	0	C	<input type="text"/>	0
A5	<input type="text"/>	0	D	<input type="text"/>	0
			Tripot	<input type="text"/>	0

Pololu Orangutan SVP

Servos (PWM) Motors Analog Digital Other

Pin #	Variable	Bit	Input	Output
D0	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
(Red LED) D1	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
D2	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
D3	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>
(Green LED) D7	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	<input type="checkbox"/>

Pololu Orangutan SVP

Servos (PWM) Motors Analog Digital Other

Buttons

Button	Variable	Bit
PC2	<input type="text"/>	<input type="text"/>
PC3	<input type="text"/>	<input type="text"/>
PC5	<input type="text"/>	<input type="text"/>

Battery

Variable

Value 9990

Play Song

Song to Play Variable

Song Name

LCD

Display Text Variable

Display Static Text

Instructions

1. Setup - Open the AVR Studio development environment and download this [program](#) and upload it to your SVP board. The provided C program is how the communication between the PC and the SVP is accomplished. It is required otherwise the PC will not be able to tell the SVP what settings to make nor be able to read any analog or digital values.

The program can be modified as per your needs but you will need to be careful not to destroy any of the functionality otherwise the RoboRealm module may stop working as expected if it is not getting the right signals back from the SVP.

2. COM Port - Specify the COM port that your SVP is connected to. Note that the SVP comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports. It should be the serial port labeled as "SVP USB Communications".

3. Baud Rate - Specify the baud rate of 115200 to communicate with the SVP. Note that this is much higher than the standard 9600 normally used. The higher rate provides much better responsiveness than the lower rates. If you change the baud rate you will need to do so in the uploaded C application too.

4. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the SVP module is loaded the COM port and BAUD rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.

5. Servos - Once communications are specified and you have a servo hooked up to the board you should be able to move the appropriate slider according to which pin you used in order to see the servo move. Note that you MUST connect PB3 to SA, PB4 to SB, and PC0 to SC in order for the servos to work. Note that the PB3, PB4 and PC0 are across the board over the LCD from the SA, SB, and SC pins. Keep in mind that negative/black is on the outside of the board. If the servo does not move, check your wiring off the servo that the COM port/USB cable are all connected and properly specified.

6. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.
7. Current Value - To manually set the servo position type in the appropriate number (500-2500, 1500 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 500 and 2500 will most likely be beyond the capabilities of your servos so you may want to tighten the range.
8. Sliders - You should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box. If the servos do not move check your USB cable and/or board power connections.
9. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
10. Motors - Two motors can be connected to the M1 and M2 pins to drive DC motors. The motor controls are very similar to the servo controls and also include sliders and min/max limits. Note, however, that the neutral motor value is zero as apposed to 1500 for the servo. See [Variable Control](#) for more information on how to programatically move the robot.
11. Analog - If you want to sample the analog values specify a variable or type one in that will contain the value from the appropriate pin. Note that you must select the checkbox next to the Analog Pin in order to activate this functionality. As soon as the pin is activated the value read at that pin will be displayed. Note that even when unattached the pin will show a very low <200 value due to noise. Also note that when pin 0 goes high that other ungrounded pins will also go high due to signal bleed.

Once the value is placed within the variable you can then use this value in other modules, or accessed via the API, extension, etc. in order to move the value outside of the RoboRealm application.

The trimpot is also included in this interface. You can change its value by rotating the pot located right next to the USB connector.

12. Digital Pins - The SVP comes with many digital pins but when all other capabilities are in use 7 digital IO pins remain that can be configured as an input or an output. Unlike Analog pins, Digital pins can only receive a 1 or 0 (on or off) as apposed to a full numeric value. Likewise, a digital pin set to an output can only send an on or off to the pin similar to a switch.

The module provides a way to either receive or send signals to these pins via the C program. Keep in mind that the digital IO pins use inverted logic, they are Low/Off when the checkbox is checked. Also keep in mind that for output you need to connect to the two inner slots (positive and IO) as apposed to IO and ground.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low. Be sure the enable the pin (first checkbox) to enable the controls for that pin.

For input pins the checkbox will reflect the read high or low state of the pin but will remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

As an experiment, you can select the pin D1 as an "out", By checking and un-checking the checkbox you can make the Red LED blink.

If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

13. Buttons - The SVP has 3 physical micro buttons located on the board. If you enable this interface and then press one of those buttons you should see the checkbox change in the GUI interface. This allows you to react to those buttons by saving their state into a RoboRealm variable for later use.

14. Battery - You can keep a check on the battery life of the SVP by enabling the battery checkbox. Adding in a variable will place the current battery power rating into that variable for use by other modules such as the VBScript module. Note the value is in millivolts.

15. Play Song - you can quickly select a song and press play to hear the SVP play a quick melody. To automate the playback of songs select a variable that will contain the song name as seen in the dropdown. Note that after the song begins playing the variable is cleared to avoid repeating the song. To add your own songs you can edit the "Music.rtttl" file in the RoboRealm folder. This file contains RTTTL formatted melodies which are converted and sent to the SVP for playback. Note that the RTTTL format is the Nokia Cell Phone ringtone format and can be found for free in many sites.

16. LCD - The SVP comes equipped with a 2 line LCD display. The LCD controls in the SVP module provide a way to display text on that screen. The variable dropdown provides a way to send any text to that LCD screen while the Static Text allows you just to type some quick text to see on the screen. Note that if both are specified then the variable will take priority.

When sending a text message the first line will be used and wrap to the second UNLESS you specify a "\n" within the text string which will move to the second line. Remember, you only have 32 total characters so use them wisely!

See Also

[Sparkfun Arduino](#)
[Axon](#)
[Parallax BoeBot](#)
[A-WIT BOL-BOT](#)

Sparkfun Arduino Uno/Mega



The Sparkfun Arduino modules provides an interface to the Arduino [Duemilanove](#) or the [Mega](#) sold by [Sparkfun](#). These boards have a USB to serial connection with a PC and can operate at 115200 baud which makes it an ideal platform for communication with a PC. The Duemilanove has 6 PWM (Servo) capable digital signals, 6 10bit analog in and 14 digital IO lines (shared with the 6 used for PWM). The Mega has 14 PWM (Servo), 16 Analog and 32 Digital IO lines. Thus, you can use these inexpensive boards as a servo controller and an analog/digital IO board for your robotic projects.

These modules are meant to allow RoboRealm to communicate with an Arduino while connected via USB or bluetooth wireless link. It requires you to download the [Uno Sketch](#) or [Mega Sketch](#) (Arduino programs) into the Arduino board in order to facilitate communication with this module in RoboRealm.

From there you can use the Arduino as a servo controller, digital IO, input device, etc. from within RoboRealm by setting or reading the variables set by this module. If you have special requirements you can also modify the Sketch program to do other processing (the link above is to the source code).

Naturally, you can roll your own RoboRealm Arduino communication using the [Serial](#) module but having a stock module perform this means you can test things out quickly without needed to understand the internal programming of the Arduino.

The communication protocol that the module uses to communicate with the Arduino is a simplistic 1 or 4 byte packet. The data transmitted is a 7 bit value with the 8th bit being a sync bit in case the Arduino and PC become out of alignment. Only the header bit has the 8th bit set, all other bytes do not. This allows that when a new packet is to be processed that the first and only first byte has bit 8 set.

For the Uno/Due the 6 PWM lines are part of the 14 digital IO lines. If you need to control 6 servos then you will have 6 less digital IO lines. Pins 0 and 1 are tied to the serial communication and therefore when communicating over serial to a PC are not available for general use.

For the commands that do not require a value payload they only require 1 byte. This byte includes the 128 sync bit, the command 0-7, and the channel 0-6 or 2-14. The 4 packet size is used for those commands that have a value payload such as setting a servo position (typically 500-2500)

which requires 2 bytes for the data. The 3rd byte used is a CRC to ensure that they header and values were all received together and have not been mixed up.

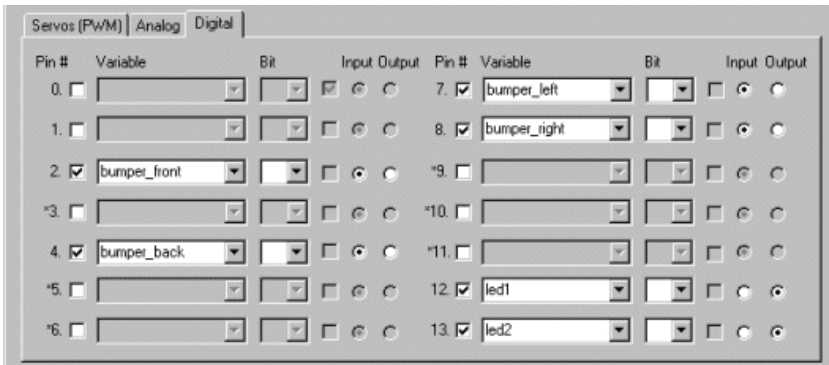
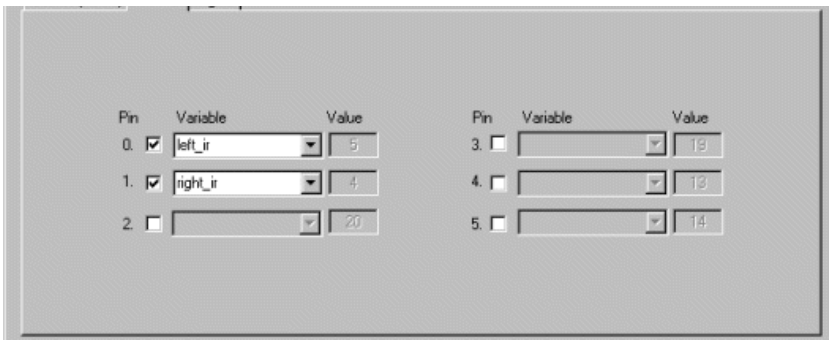
Values such as the analog signals are also passed back to the PC using the same 4 byte structure also including a CRC to be sure that the bytes were received in order correctly.

Interface

Pin	Variable	Trim	Current Value	Min/Max Limits
3	left_motor	0	1300	500 2500
5	right_motor	0	2000	500 2500
6		0	1500	500 2500
9		0	1500	500 2500
10		0	1500	500 2500
11		0	2026	500 2500

Communication
COM Port: COM233 - USB Serial Port Baud Rate: 115200 Remember as default

Buttons: Stop, Help, OK, Cancel



Instructions

1. Setup - Open the Arduino development environment and download this [Sketch](#) and upload it to your Arduino. The provided Sketch is how the communication between the PC and the Arduino is accomplished. Note that this is NOT a new bootloader program but just a regular user Sketch running on the Arduino. It is required otherwise the PC will not be able to tell the Arduino what settings to make nor be able to read any analog or digital values.

The Sketch can be modified as per your needs but you will need to be careful not to destroy any of the functionality otherwise the RoboRealm module may stop working as expected if it is not getting the right signals back from the Arduino.

2. Com Port - Specify the COM port that your Arduino is connected to. Note that the Duemilanove comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.

3. Baud Rate - Specify the baud rate of 115200 to communicate with the Arduino. Note that this is much higher than the standard 9600 normally used. The higher rate provides much better responsiveness than the lower rates.

4. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the Arduino module is loaded the COM port and baud rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.

5. Servos - Once communications are specified and you have a servo hooked up to the board you should be able to move the appropriate slider according to which pin you used in order to see the servo move. Note that the Duemilanove does NOT come with standard 3 pin servo connectors so you will need to connect the red/orange to power, black/brown to ground, and white/yellow to the pin you'd like to use. If the servo does not move, check your wiring off the servo, check that you have an external power supply connected (some servos are very power hungry) and check that the COM port/USB cable are all connected and properly specified. Note that you will be using the same port as the Arduino environment uses to upload applications so be sure that you are done uploading the Sketch before testing. Since RoboRealm holds the COM port open you will need to exit RoboRealm in order to upload a new Sketch.

6. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.

7. Current Value - To manually set the servo position type in the appropriate number (500-2500, 1500 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 500 and 2500 will most likely be beyond the capabilities of your servos so you may want to tighten the range.

8. Sliders - You should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box. If the servos do not move check your USB cable and/or board power connections.

9. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

10. Analog Input - If you want to sample the analog values specify a variable or type one in that will contain the value from the appropriate pin.

Note that you must select the checkbox next to the Analog Pin in order to activate this functionality. A message on the pin is notated above the pin number.

NOTE that you must select the checkbox next to the ANALOG PIN in order to activate this functionality. AS soon as the pin is activated the value read at that pin will be displayed. Note that even when unattached the pin will show a very low <200 value due to noise. Also note that when pin 0 goes high that other ungrounded pins will also go high due to signal bleed.

Once the value is placed within the variable you can then use this value in other modules, or accessed via the API, extension, etc. in order to move the value outside of the RoboRealm application.

11. Analog Output - If you want to send a specific voltage out over the analog pins specify a variable or type one in that will contain the value or type in a number from 0 to 255 in the provided text box. This will cause the Arduino to send a frequency wave over that pin which averages to a specific voltage, Note that while this is a form of PWM is it not the same format as that used to control servos.

Note that the analog out pins are NOT the same as the analog input pins but are in fact the same pins used in controlling servos and digital pins 3,5,6,9,10 and 11. These pins are functionally shared between servos, analog out and digital pins so use them sparingly!

12. Digital Pins - The Arduino comes with 14 digital IO pins that can be configured as an input or an output. Unlike Analog pins, Digital pins can only receive a 1 or 0 (on or off) as apposed to a full numeric value. Likewise, a digital pin set to an output can only send an on or off to the pin similar to a switch.

The module provides a way to either receive or send signals to these pins via the RoboRealm_Arduino Sketch program. Note that several of the pins are used by the Servos and thus enabling them as a digital pin will DISABLE the appropriate Servo to indicate this pins reassignment.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low. Be sure the enable the pin (first checkbox) to enable the controls for that pin. Again, if you enable pin 3,5,6,9,10,11 a Servo configuration in the first tab will be disabled. These pins are marked with a '*' in the Digital tab.

For input pins the checkbox will reflect the read high or low state of the pin but will remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

As an experiment, you can connect an LED to pin 2 and then select the pin as an "out", By checking and un-checking the checkbox you can make the LED blink. Alternatively, if you don't have an LED handy try this on pin 13 as that is connected to the LED next to the Tx and Rx leds on the Duemilanove.

If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

See Also

[Axon](#)

[Parallax BoeBot](#)

[A-WIT BOL-BOT](#)

Sparkfun Arduino Uno/Mega



The Sparkfun Arduino modules provides an interface to the Arduino [Duemilanove](#) or the [Mega](#) sold by [Sparkfun](#). These boards have a USB to serial connection with a PC and can operate at 115200 baud which makes it an ideal platform for communication with a PC. The Duemilanove has 6 PWM (Servo) capable digital signals, 6 10bit analog in and 14 digital IO lines (shared with the 6 used for PWM). The Mega has 14 PWM (Servo), 16 Analog and 32 Digital IO lines. Thus, you can use these inexpensive boards as a servo controller and an analog/digital IO board for your robotic projects.

These modules are meant to allow RoboRealm to communicate with an Arduino while connected via USB or bluetooth wireless link. It requires you to download the [Uno Sketch](#) or [Mega Sketch](#) (Arduino programs) into the Arduino board in order to facilitate communication with this module in RoboRealm.

From there you can use the Arduino as a servo controller, digital IO, input device, etc. from within RoboRealm by setting or reading the variables set by this module. If you have special requirements you can also modify the Sketch program to do other processing (the link above is to the source code).

Naturally, you can roll your own RoboRealm Arduino communication using the [Serial](#) module but having a stock module perform this means you can test things out quickly without needed to understand the internal programming of the Arduino.

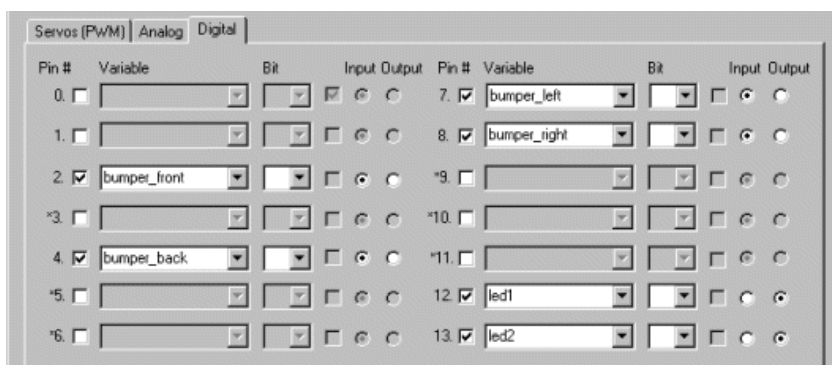
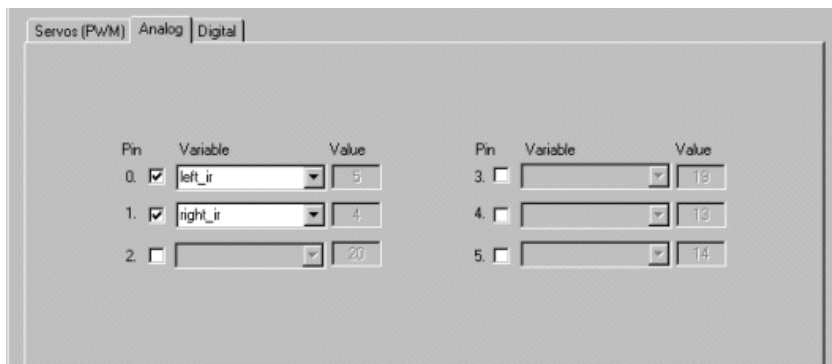
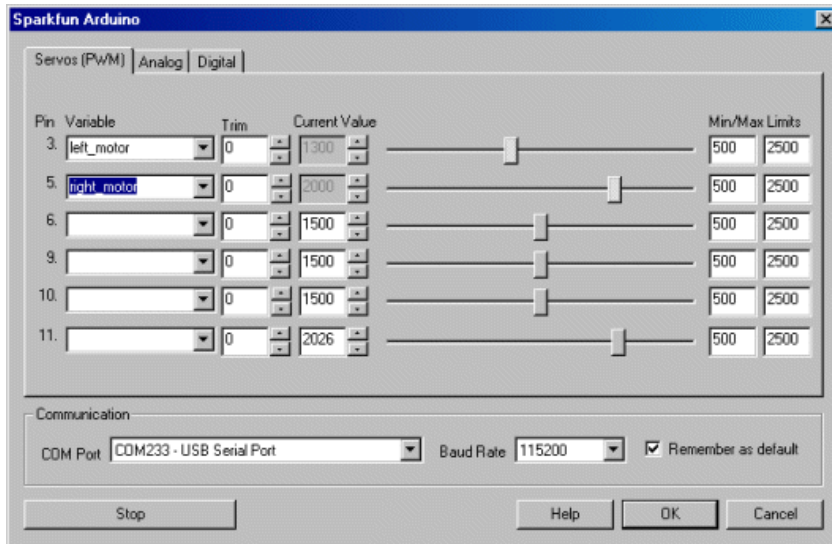
The communication protocol that the module uses to communicate with the Arduino is a simplistic 1 or 4 byte packet. The data transmitted is a 7 bit value with the 8th bit being a sync bit in case the Arduino and PC become out of alignment. Only the header bit has the 8th bit set, all other bytes do not. This allows that when a new packet is to be processed that the first and only first byte has bit 8 set.

For the Uno/Due the 6 PWM lines are part of the 14 digital IO lines. If you need to control 6 servos then you will have 6 less digital IO lines. Pins 0 and 1 are tied to the serial communication and therefore when communicating over serial to a PC are not available for general use.

For the commands that do not require a value payload they only require 1 byte. This byte includes the 128 sync bit, the command 0-7, and the channel 0-6 or 2-14. The 4 packet size is used for those commands that have a value payload such as setting a servo position (typically 500-2500) which requires 2 bytes for the data. The 3rd byte used is a CRC to ensure that they header and values were all received together and have not been mixed up.

Values such as the analog signals are also passed back to the PC using the same 4 byte structure also including a CRC to be sure that the bytes were received in order correctly.

Interface



Instructions

1. Setup - Open the Arduino development environment and download this [Sketch](#) and upload it to your Arduino. The provided Sketch is how the communication between the PC and the Arduino is accomplished. Note that this is NOT a new bootloader program but just a regular user Sketch running on the Arduino. It is required otherwise the PC will not be able to tell the Arduino what settings to make nor be able to read any analog or

digital values.

The Sketch can be modified as per your needs but you will need to be careful not to destroy any of the functionality otherwise the RoboRealm module may stop working as expected if it is not getting the right signals back from the Arduino.

2. Com Port - Specify the COM port that your Arduino is connected to. Note that the Duemilanove comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.

3. Baud Rate - Specify the baud rate of 115200 to communicate with the Arduino. Note that this is much higher than the standard 9600 normally used. The higher rate provides much better responsiveness than the lower rates.

4. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the Arduino module is loaded the COM port and baud rate will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.

5. Servos - Once communications are specified and you have a servo hooked up to the board you should be able to move the appropriate slider according to which pin you used in order to see the servo move. Note that the Duemilanove does NOT come with standard 3 pin servo connectors so you will need to connect the red/orange to power, black/brown to ground, and white/yellow to the pin you'd like to use. If the servo does not move, check your wiring off the servo, check that you have an external power supply connected (some servos are very power hungry) and check that the COM port/USB cable are all connected and properly specified. Note that you will be using the same port as the Arduino environment uses to upload applications so be sure that you are done uploading the Sketch before testing. Since RoboRealm holds the COM port open you will need to exit RoboRealm in order to upload a new Sketch.

6. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.

7. Current Value - To manually set the servo position type in the appropriate number (500-2500, 1500 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 500 and 2500 will most likely be beyond the capabilities of your servos so you may want to tighten the range.

8. Sliders - You should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box. If the servos do not move check your USB cable and/or board power connections.

9. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.

10. Analog Input - If you want to sample the analog values specify a variable or type one in that will contain the value from the appropriate pin. Note that you must select the checkbox next to the Analog Pin in order to activate this functionality. As soon as the pin is activated the value read at that pin will be displayed. Note that even when unattached the pin will show a very low <200 value due to noise. Also note that when pin 0 goes high that other ungrounded pins will also go high due to signal bleed.

Once the value is placed within the variable you can then use this value in other modules, or accessed via the API, extension, etc. in order to move the value outside of the RoboRealm application.

11. Analog Output - If you want to send a specific voltage out over the analog pins specify a variable or type one in that will contain the value or type in a number from 0 to 255 in the provided text box. This will cause the Arduino to send a frequency wave over that pin which averages to a specific voltage. Note that while this is a form of PWM is it not the same format as that used to control servos.

Note that the analog out pins are NOT the same as the analog input pins but are in fact the same pins used in controlling servos and digital pins 3,5,6,9,10 and 11. These pins are functionally shared between servos, analog out and digital pins so use them sparingly!

12. Digital Pins - The Arduino comes with 14 digital IO pins that can be configured as an input or an output. Unlike Analog pins, Digital pins can only receive a 1 or 0 (on or off) as apposed to a full numeric value. Likewise, a digital pin set to an output can only send an on or off to the pin similar to a switch.

The module provides a way to either receive or send signals to these pins via the RoboRealm_Arduino Sketch program. Note that several of the pins are used by the Servos and thus enabling them as a digital pin will DISABLE the appropriate Servo to indicate this pins reassignment.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low. Be sure the enable the pin (first checkbox) to enable the controls for that pin. Again, if you enable pin 3,5,6,9,10,11 a Servo configuration in the first tab will be disabled. These pins are marked with a '*' in the Digital tab.

For input pins the checkbox will reflect the read high or low state of the pin but will remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an output. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

As an experiment, you can connect an LED to pin 2 and then select the pin as an "out", By checking and un-checking the checkbox you can make the LED blink. Alternatively, if you don't have an LED handy try this on pin 13 as that is connected to the LED next to the Tx and Rx leds on the

the LED blink. Alternatively, if you don't have an LED ready try this on pin 15 as that is connected to the LED from the TX and RX LEDs on the Duemilanove.

If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

See Also

[Axon](#)

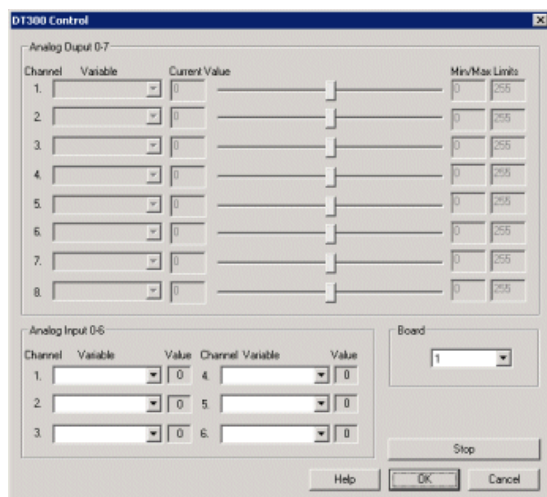
[Parallax BoeBot](#)

[A-WIT BOL-BOT](#)

DT300

The [DT304 or DT300 series](#) control interface allows you to interface RoboRealm to servos via a motor controller made by [Data Translation](#). The DT300 series support multiple Analog Input, Analog Output, and Digital IO channels.

Interface



Instructions

1. Board - Select the appropriate Board as shown in the dropdown.
2. Analog Output - After specifying this information you should be able to output different voltage levels by dragging the sliders to the right or left or by specifying a number within the current value text box in the Analog Output Channels area. If the voltage output does not change check your board specification and/or connections.
3. Variable - Select the appropriate variables that contain or will contain the value that will be sent to the DT300 board. This is used to automatically change the voltage values based on your VBScript (using the SetVariable function) or Extension based program. You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to send high voltage levels above or below the specified limits. This can be used as an additional precaution in case your motors cannot physically move beyond certain limits.
4. Analog Input - Select the appropriate variables that will contain the inputs from the Analog Inputs from the board. The variables can then be accessed using the VBScript or other extension programs to perform appropriate actions.

accessed using the VDScript or other extension programs to perform appropriate actions.

5. Stop - Press 'Stop' if you need to quickly reset all voltage levels and return them to the middle or neutral position.

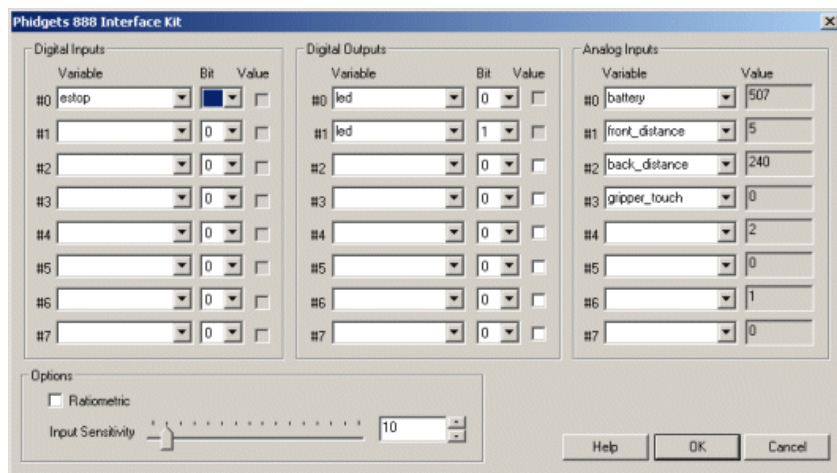
Phidgets_888_Interface_Kit



Provides an interface to the Phidgets 888 Interface Kit. The Interface kit has 8 digital output, 8 digital inputs and 8 analog inputs that provide you with enough inputs and outputs to interface with many sensors. The board is generic in that the analog inputs can be used with distance sensors such as IR or sonar but also with any analog sensor like compass, gyro's, etc. The digital outputs can be used to interface with LED's, relays, etc.

Note that the 888 Interface Kit module is also compatible with other Interface Kits like [0/0/4 Interface Relay Kit](#).

Interface



Instructions

1. Digital Inputs Variable - Select or type in a variable that will receive a 1 when a signal is detected in the appropriate channel.
2. Digital Inputs Bit - Select which bit within the variable should be set if a signal is detected in that digit port. Setting the bit selection to nothing will cause a 1 or 0 to be used. Setting the bit to 2 will cause the variables value to contain a 1 or 0 in the 3rd bit (4 value) of the variable. Note that setting a bit will NOT affect any other bits. Using this you can construct a binary number using a single bit per channel.
3. Digital Outputs Variable - Select or type in a variable that contains a non-zero number when a signal should be produced in the appropriate digital out channel/pin.
4. Digital Outputs Bit - Select which bit should be tested in the variable to determine if a signal is to be produced in the desired channel. Selecting a bit allows you to use output a binary number using several channels/pins.
5. Analog Input Variable - Select or type in a variable that will receive the analog channel's value. The actual value being read from the analog channel is displayed next to the variable dropdown in an inactive textbox.
6. Ratiometric - Select if your analog measurements are Ratiometric. Ratiometric refers to the output voltage of the sensor being a ratio of the input voltage. For example, if you double the input voltage then the output voltage will also double.
7. Input Sensitivity - select how sensitive analog voltage measurements should be. The default value of 10 should suffice for most applications.

See Also

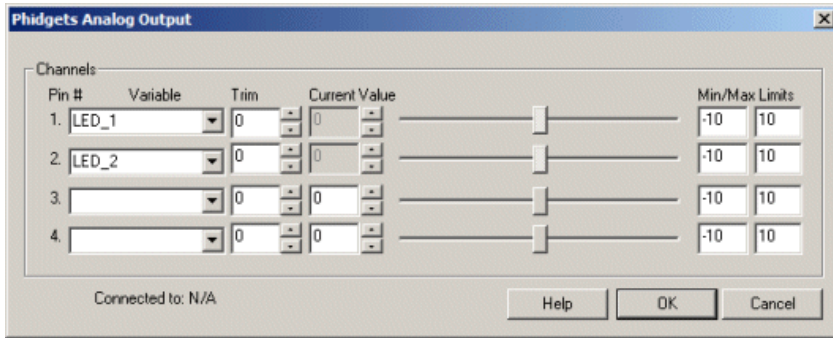
[Velleman K8055](#)

Phidgets Analog Output (4)



The Phidgets Analog Output module provides a way to interface to the Analog Output board made by Phidgets. The module provides access to four separate analog output channels. The voltages range from -10.0V to 10.0V.

Interface



Instructions

1. Variable - for each channel specify a variable that contains the value that should be sent to the channel output.
2. Trim - if the voltage is a little off from neutral use the trim to correct for bias values.
3. Values - if a variable is not specified you can type in the value of the voltage to send to the particular channel. You can either type a number directly into the textbox or use the up and down arrow keys to increment/decrement the value.
4. Slider - alternative to typing in a number you can use the slider bar to change the voltage level sent to a channel.
5. Min/Max - to ensure that you do not go under or over a voltage level in order to protect the connected device, change the min and max voltage settings that are allowed to be sent to a particular channel. This is a good way to ensure that despite programming mistakes you do not overpower your device.

See Also

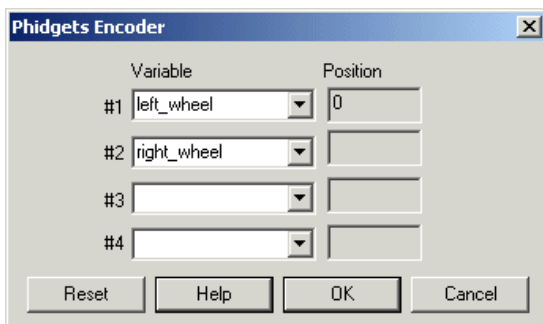
[Phidgets Interface Kit](#)

Phidgets Encoder



The Phidgets Encoder module provides a way to interface to the Encoder board made by Phidgets. The module provides access to four separate encoder boards. It is assumed that these boards are connected to wheel shafts to provide an indication on how quickly the wheel is rotating.

Interface



Instructions

1. Variable - for each encoder specify a variable that should contain the value of that encoder.

See Also

[Phidgets Servo Controller](#)



Sparkfun SerIO

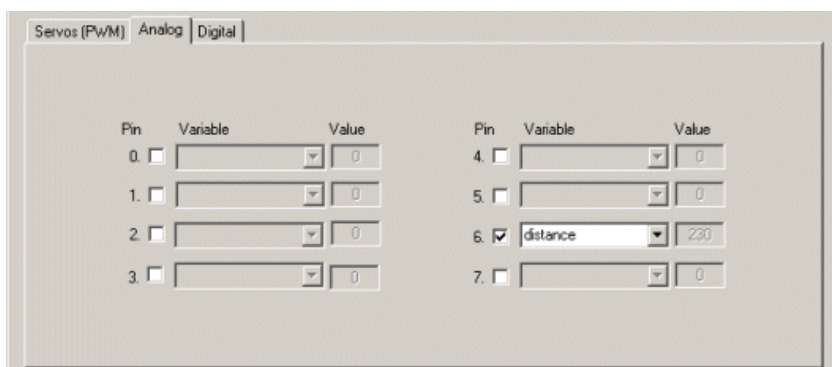
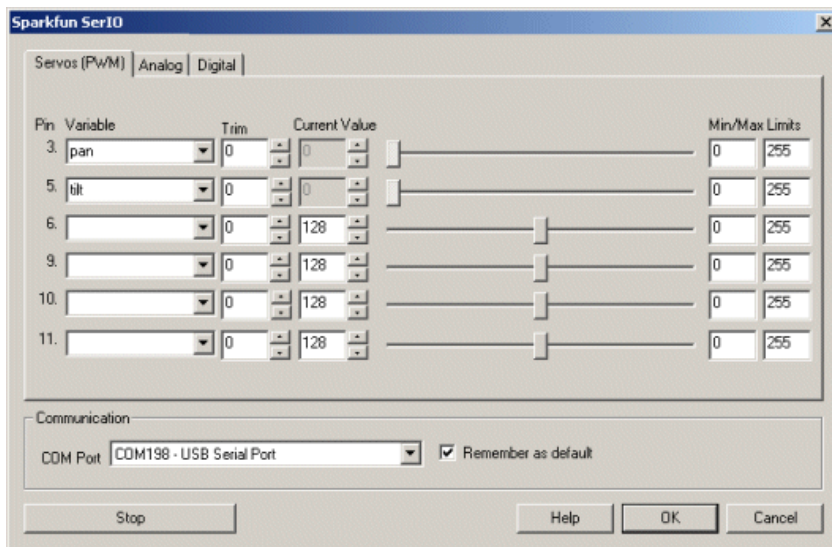


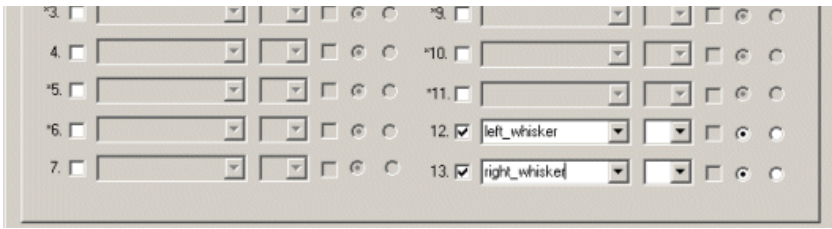
The Sparkfun SerIO module provides an interface to the SerIO (Serial Input/Output board) sold by [Sparkfun](#). This board has a USB to serial connection with a PC and can operate at 57600 baud which makes it an ideal platform for communication with a PC. The SerIO has 6 PWM/Analog Out (Servo) capable digital signals, 8 analog in and 14 digital IO lines (shared with the 6 used for PWM). Thus, you can use this inexpensive board as a servo controller and an analog/digital IO board for your robotic projects.

The SerIO is based on the Arduino platform (thus the pin similarity). The SerIO is beneficial over the Arduino in those cases where you prefer not to be responsible for programming an Arduino to provide servo and IO capabilities. If programming is your thing you'll probably be better off with the [Arduino](#). However, if you intend to move to custom control of the Arduino at a later date and need something to get up and running without much difficulty the SerIO offers a very nice solution in this regard.

The 6 PWM lines are part of the 14 digital IO lines. If you need to control 6 servos then you will have 6 less digital IO lines.

Interface





Instructions

1. COM Port - Specify the COM port that your SerIO is connected to. Note that the SerIO comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.
 2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the SerIO module is loaded the COM port will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.
 3. Servos - Once communications are specified and you have a servo hooked up to the board you should be able to move the appropriate slider according to which pin you used in order to see the servo move. If the servo does not move, check your servo connection and that the COM port/USB cable are all connected and properly specified.
 5. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Plugin based program.
 6. Current Value - To manually set the servo position type in the appropriate number (0-255, 128 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 0 and 255 will most likely be beyond the capabilities of your servos so you may want to tighten the range.
 7. Sliders - You should be able to move your servos by dragging the sliders to the right or left or by specifying a number within the current value text box. If the servos do not move check your USB cable and/or servo connections.
 8. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
 - 9 Analog - If you want to sample the analog values specify a variable or type one in that will contain the value from the appropriate pin. Note that you must select the checkbox next to the Analog Pin in order to activate this functionality. As soon as the pin is activated the value read at that pin will be displayed. Note that even when unattached the pin will show a very low value due to noise. Also note that when pin 0 goes high that other ungrounded pins will also go high due to signal bleed.
- Once the value is placed within the variable you can then use this value in other modules, or accessed via the API, extension, etc. in order to move the value outside of the RoboRealm application.
10. Digital Pins - The SerIO comes with 14 digital IO pins that can be configured as an input or an output. Unlike Analog pins, Digital pins can only receive a 1 or 0 (on or off) as apposed to a full numeric 0-255 value. Likewise, a digital pin set to an output can only send an on or off to the pin similar to a switch.

Each pin can either be set as an input or output. Do this by selecting the "in" or "out" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low. Be sure the enable the pin (first checkbox) to enable the controls for that pin. Again, if you enable pin 3,5,6,9,10,11 a Servo configuration in the first tab will be disabled. These pins are marked with a '*' in the Digital tab.

For input pins the checkbox will reflect the read high or low state of the pin but will remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.

As an experiment, you can connect an LED to pin 2 and then select the pin as an "out", By checking and un-checking the checkbox you can make the LED blink. Alternatively, if you don't have an LED handy try this on pin 13 as that is connected to the LED next to the Tx and Rx leds on the SerIO.

If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two frames captured. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.

See Also

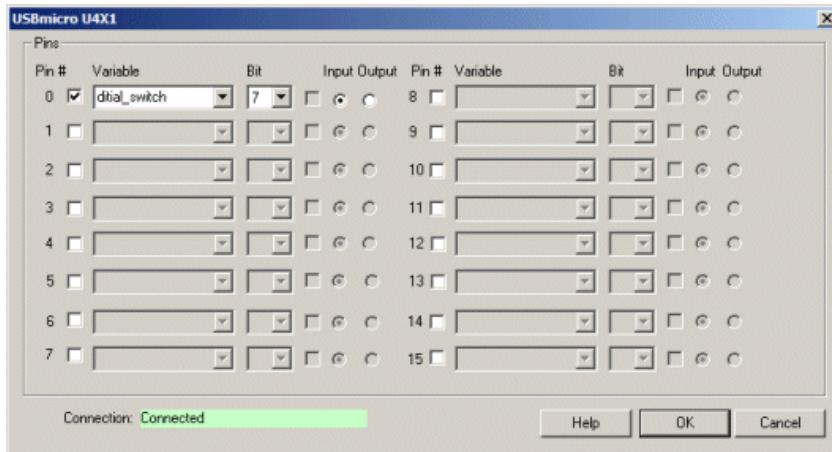
[Sparkfun Arduino](#)
[Axon](#)



USBmicro U4X1

The U421 or U401 is a USB solution that is pre-built, pre-programmed, and pre-tested and will get you interfacing your PC to various devices. These are great replacements for the now defunct Parallel Port. The boards offer up to 16 DIO lines that can be set as input or output at 5V.

Interface



Instructions

1. Connection - If the board has been detected and drivers installed the module will connect and display a green connected text. This indicates everything is working as expected on the board.
2. Pin - To enable a particular pin click the checkbox next to the appropriate pin. Note that on the U401 the first pin is the 15th pin starting from the left.
3. Input/Output - Select the appropriate radio button that specifies if you want to use the pin as an input or output.
4. Checkbox - For an Output line, you can then click on the checkbox next to the radio buttons to manually switch the line on and off.
5. Variable - To automatically receive the input state or set the output state, specify or type in a variable that will contain or currently contains the value to use.
6. Bit - If you want to only utilize one of the bits within that variable value select the bit number to test or set (depending on if the pin is set as an input or output).

See Also

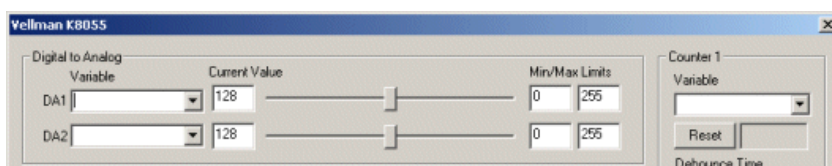
- [Sparkun SerIO](#)
- [Phidgets 888 Interface Kit](#)

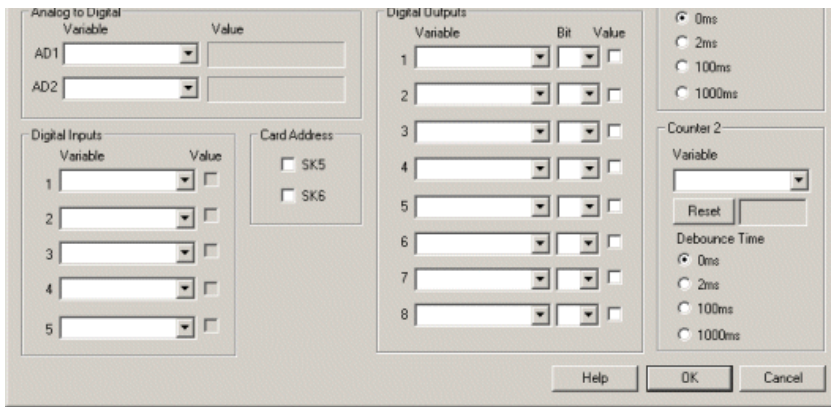
Velleman USB Interface Board



The Velleman K8055 module provides an interface to the Velleman board from RoboRealm. The K8055 interface board has 5 digital input channels and 8 digital output channels. In addition, there are two analogue inputs and two analogue outputs with 8 bit resolution.

Interface





Instructions

1. Card Address - Select the appropriate Card Address for the Velleman Board.
2. Digital to Analog Output - After specifying the Card Address you can changing the Digital to Analog ports by dragging the sliders to the right or left or by specifying a number within the current value text box in the Current Value textfield. If the ports do not activate check your USB connection and/or board power connections.
Selecting a variable in the provided dropdown will automatically change the servo values based on your VBScript (using the SetVariable function), Extension based program or [Set Variable](#) module.
3. Analog to Digital Input - Select an appropriate variable that will contain the value read from the analog port of the Interface board. The value is displayed as a progress bar and indicates the value being written into the specified variable.
4. Digital Inputs - The K8055 comes with 5 digital inputs. The specified variable selected or typed in the combo box will contain the value received from that digital input pin. The value of the pin will be reflected in the checkbox status, checked for 1 and unchecked for 0.
4. Digital Outputs - The K8055 comes with 8 digital outputs. To manually set a digital output click on the provided checkbox. When checked the pin will be set to high, unchecked the pin will be low. To automatically set the pin value select or type in the variable name that contains the appropriate value. You can then select the appropriate bit to check in that value in order to determine the pin status. This bit selection allows you to use a single variable in all digital output pins and use the bit selection to determine which pin should be on. For example, 255 would turn on all pins if each pin uses successive bits (1,2,4,8,16,etc).
4. Counter - Select or type in the variable names that will contain the value of the counter. You can press the reset button to clear the counter and start from zero. Use the radio buttons to adjust the debounce time.

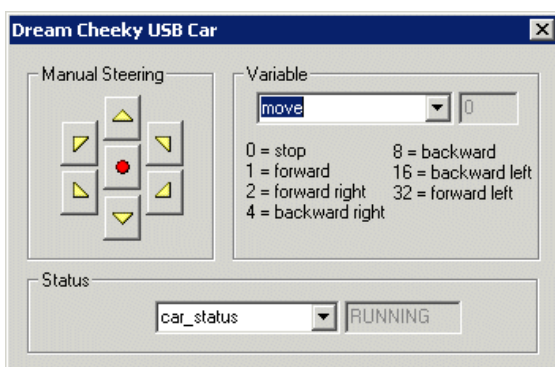
See Also

[Phidgets 888 Interface Kit](#)

Dream Cheeky USB Car

The DC_Car module provides an interface to the [Dream Cheeky USB Car](#). By specifying an appropriate variable the value will be sent to the device to control its movements. For testing purposes a manual interface (see yellow arrows below) has also been provided.

Interface





Instructions

1. Buttons - Move the DC Car by pressing the appropriate arrows. The middle red button is for stop. Note that once you release the button the car will assume a stop. This interface is used to test the connection to the USB docking station and the ability for RoboRealm to control the car. 2. Variable - The variable dropdown allows you to specify a variable that will contain the commands sent to the USB Car. This variable should contain one of the following numbers:

0 = stop
1 = forward
2 = forward right
4 = backward right
8 = backward
16 = backward left
32 = forward left

Thus if you wanted the car to move to the right you could specify a variable like "move" and set that variable's value (using either a [VBScript program](#) or the [Set_Variable](#) module) to 2. Note that the device keeps moving in the specified direction until that variable changes value. (like 0 for stop)

3. Status - select a variable that will contain the current status of the car. The car can be in one of 3 status:

RUNNING - car is operational and not in the docking station

CHARGED - car is fully charged and in docking station

CHARGING - car is being charged in docking station

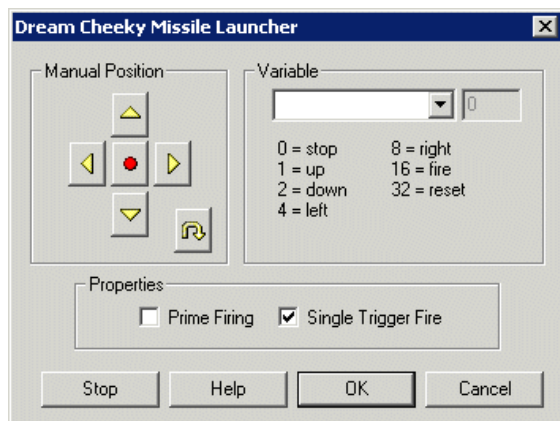
See Also

[Dream Cheeky Missile Launcher](#)

Dream Cheeky Missile Launcher

The DC_Missile module provides an interface to the [Dream Cheeky Missile Launcher](#). By specifying an appropriate variable the value will be sent to the device to control its movements. For testing purposes a manual interface (see yellow arrows below) has also been provided.

Interface



Instructions

1. Buttons - Move the DC Missile Launcher by pressing the appropriate arrows. The middle red button is for firing. To fire, hold down the button until the device has released the compressed air. This takes a couple seconds.


2. Reset - Next to the move buttons is the reset button. This will center the USB Missile Launcher and provide you with a known starting point. Note that the centering is based just on timing so if your machine is very taxed during the centering process the position may be a somewhat inconsistent.

3. Variable - Select the appropriate variable that contains the bits set based on what movement or action you want the device to perform. Note that multiple directions (up and left) can be combined into a single move command.

4. Prime Firing - Select the checkbox if you want to reduce the time between a fire command and the compressor reaching the launching point. If this button is selected the launcher will attempt to prime a launch by compressing enough air just before launch. Once a fire command is received the compressor will fire the missile and then prime the next tank.

5. Single Trigger Fire - Uncheck if you want to hold down the fire button for as long as it takes to reach launch compression. This provides you better control over the fire button but requires you to keep holding it down for the missile to fire.

Example

 [Try this robo-file](#) to control your missile launcher using your keyboard.

The movement numbers are

0 = stop
1 = up
2 = down
4 = left
8 = right
16 = fire
32 = reset

Thus if you wanted the device to move to the right you could specify a variable like "move" and set that variable's value (using either a [VBScript program](#) or the [Set_Variable](#) module) to 8. Note that the device keeps moving in the specified direction until that variable changes value. (like 0 for stop)

See Also

[Striker Missile Launcher](#)

EndurancePCTx

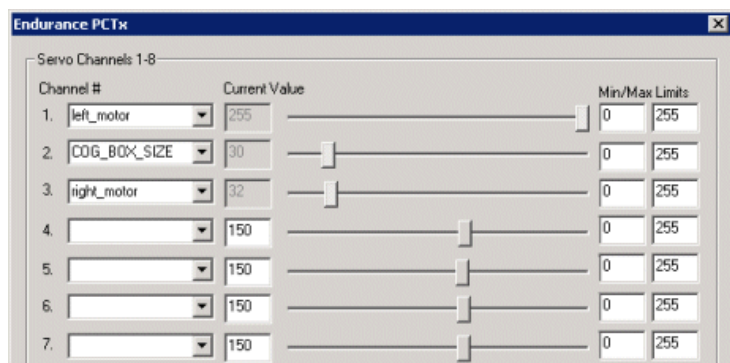
The PCTx made by [Endurance R/C](#) allows you to control a RC based vehicle using your PC. Endurance R/C provides the PCTx which is a USB connection between your PC and an RC transmitter's trainer port. Note that you need an existing transmitter with a trainer port to use the PCTx. Signals sent from your computer will be sent over USB to your RC transmitter's trainer port and then sent over the airwaves to your RC based car which will receive the commands destined for the car's servo motors. This control system requires minimal hardware changes in order to computerize your RC based systems.



The EndurancePCTx RoboRealm module will transmit the appropriate values from RoboRealm via the PCTx USB connection to your RC based servos. The module functions in much the same way as the [SSC](#) and [Parallax](#) modules function.

Note that it is assumed that you are using a camera capable of transmitting video to your PC. Thus all processing is performed off-robot. This system provides great flexibility in choosing a computing environment but has distance limitations imposed by your RC transmitter and the transmission range of the wireless camera.

Interface





Instructions

1. Servos - Select the appropriate variables that contain or will contain the position value that will be sent to the servo board. This is used to automatically change the servo values based on your VBScript (using the SetVariable function) or Extension based program. You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the device does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
2. Resolution - Select 255 for low resolution mode or 1024 for high resolution mode depending on which device you purchased.
3. Stopping - Press STOP if you need to quickly disable all the servos and return them to the middle or neutral (128) position.

To manually set the servo position type in the appropriate number (0-255, 150 is the default) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate.

See Also

[SSC](#)

[Parallax](#)

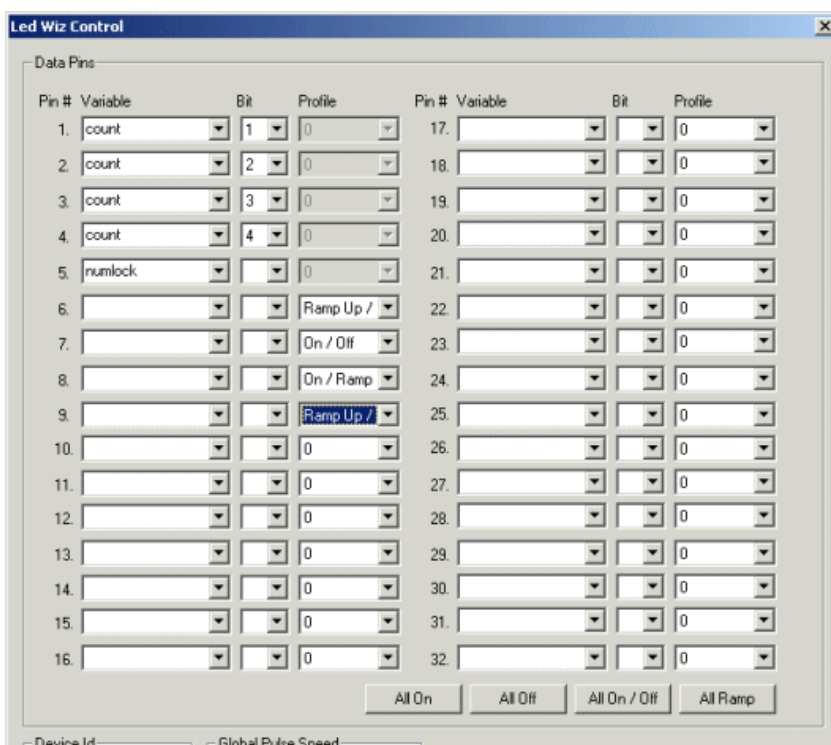
LED-Wiz

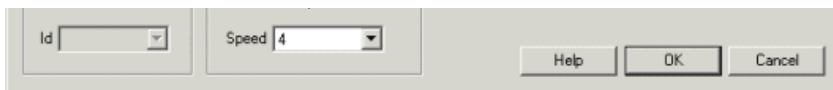


The LED-Wiz module provides an interface to the LED-Wiz board by IDVT Inc. / www.GroovyGameGear.com. The board provides for 32 Led switches that can be controlled by your PC via USB.

You need only connect the LED-Wiz to your computer to experiment with this module. The application server that comes with the LED-Wiz need not and should not be running with using the LED-Wiz with RoboRealm. Once connected you can change the profile dropdowns to switch the LEDs to different states. The LED should change as soon as the dropdown menu is selected.

Interface





Instructions

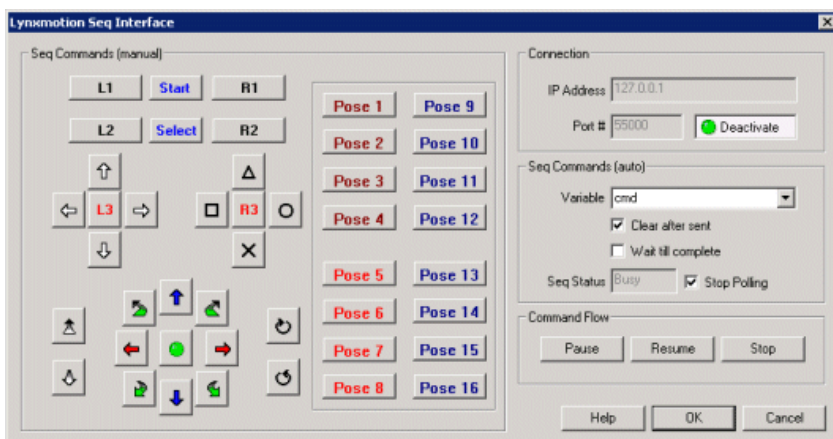
1. Data Pins - For each Pin # chose the variable that will contain a value that indicates that the corresponding LED is to be switched on.
2. Bit - Select which bit of the variable's data should be used to indicate on/off state. Note that this is useful if you wish to switch LEDs on or off to represent a binary value. This field is ignored by the Intensity Mode.
3. Profile - Select what kind of profile the LED should reflect.
 - 0 - switch the LED off
 - 12 : 48 - various intensity levels with 48 being the maximum LED intensity
 - Ramp Up / Dn - Slowly increase the LED intensity to a maximum, then switch it off and repeat.
 - On / Off - Switch the LED to full brightness and then switch it off, i.e. flash.
 - On / Ramp Dn - Start with the LED in full brightness and slowly decrease its intensity to zero. Then start again.
4. Buttons All On - All Off - All On / Off - All Ramp - sets all the LEDs to the specified profile.
5. Global Pulse Speed - Select the speed at which the LED should flash for the On / Off profile or increase / decrease intensity for the ramp profiles.

The Variable value can contain a number from 0 to 48 which would cause the LED intensity to change to that appropriate number. Note the dropdown profile only shows a subset of the numbers from 0 to 48 to conserve space. You can also set the variable's value to "Ramp Up / Dn" to set the LED to a different profile. The module will determine what to do based on the variable value (i.e. a number means intensity, otherwise a specific profile is enabled).

Lynxmotion Seq Interface

The Lynxmotion Seq Interface module provides a direct interface from RoboRealm to the [Lynxmotion Sequencer Program](#). The Sequencer program provides a GUI interface to creating servo motor sequences that are then sent to the [SSC-32 servo controller board](#). By interfacing RoboRealm to the Sequencer you can now trigger complex servo sequences based on something seen using RoboRealm. Note that you need to refer to Lynxmotion.com for purchasing and downloading of the Sequencer program.

Interface



Instructions

1. Start the Lynxmotion Sequencer and ensure that the Socket Server is operational.
2. IP Address - Enter the correct IP Address of that machine that is running the Sequencer (if it is the same machine enter 127.0.0.1).
3. Port # - Type in the appropriate port number, note that the default is 55000
4. Activate - Press the Activate button to connect to the Sequencer. All buttons will then become active as seen in the above interface.
5. You can now press the appropriate button for the command to be sent to the Sequencer. The buttons are exact copies of those you would see in the Lynxmotion Sequencer Advanced Play dialog. Any sequences that are associated with that button in the Sequencer will play. Note that you

need to setup and associate any servo sequences within the Sequencer program. RoboRealm simply provides an interface to that setup and is not used to create or modify button sequence associations.

6. To create automated triggering of a sequence select an appropriate variable (or type in a new one) that will be used to hold the command to be sent to the Sequencer. The variable's contents will be transmitted to the Sequencer. The following are valid commands that the Sequencer will recognize. Spaces between words need to be included as part of the command.

L1

L2

L3

R1

R2

R3

Start

Select

Up

Right

Down

Left

Triangle

Circle

Cross

Square

Move Forward

Move Backward

Turn Clockwise

Turn Anticlockwise

Blue Up

Red Right

Blue Down

Red Left

Up-Right

Down-Right

Down-Left

Up-Left

Green Button

Pose 1

Pose 2

Pose 3

Pose 4

Pose 5

Pose 6

Pose 7

Pose 8

Pose 9

Pose 10

- Pose 11
- Pose 12
- Pose 13
- Pose 14
- Pose 15
- Pose 16

For example, to simulate a button press of the Triangle button you would assign

SetVariable "command", "Triangle"

using the [VBScript module](#) or using the [Set_Variable](#) module.

This would cause the Lynxmotion_Seq module to send the triangle buttonpress over to the Sequencer for execution.

7. Clear after sent - If you want the variable holding the command to be cleared after being sent select the "Clear after sent" checkbox.
8. Wait till complete - As some sequences can take a while to execute you may want to pause RoboRealm while a sequence is in action to prevent another sequence from being generated and interrupting the current sequence. The "Wait till complete" will freeze RoboRealm until it receives acknowledgement from the Sequencer that the execution is over. Note that you NEED to have the "Advanced Play" interface open in the Sequencer for this action to correctly process.
9. Seq Status - shows the status of the sequencer.
10. Pause, Resume, Stop - You can pause, resume or stop the current sequence execution by pressing the appropriate buttons.

See Also

[Lynxmotion SSC-32](#)

Parallel Port

The Parallel Port module allows you to use RoboRealm to control the parallel port pins based on RoboRealm variables. The following is a brief discussion on the parallel port pins and its potential use.

The Parallel Port is an inexpensive way to communicate to external TTL 0-5V devices using your PC's parallel port. NOTE that if you are experimenting with new circuits you may want to purchase an external parallel port in case your wiring may not be correct. An external parallel port is much cheaper to replace than your motherboard!

The Parallel Port pins are divided into three groups of pins mean for difference purposes. The three sets are DATA (in/out), CONTROL (out) and STATUS(in). The DATA pins are meant for input and output of data (typically this was to send the printer the information to be printed). The CONTROL port was mean to output control information to the printer and the STATUS pins were inputs back from the printer to the PC meant to communicate things like 'out of paper', etc.

Each pin is used to literally communicate 1 bit of information. A pin is either a '0' or '1'. All pins can transmit information simultaneously in parallel as apposed to serially (one by one) as is done through a serial port. The standard parallel port is capable of sending 50 to 100 kilobytes of data per second.

The voltage level of an on pin is about 5volts and can be used to directly drive an LED. To test out your parallel port with RoboRealm you can configure you Parallel Port module as shown in the interface below, connect one end of an LED to pin3 (data2) and the other to pin25 (ground). As the image count increases the LED should blink on and off.

The parallel port can output/send 8 bits at a time and input 5 bits. On the more modern Standard Parallel Port (SPP) ports the 8 data pins can become input pins if the DIRECTION bit on the CONTROL port is set high. See below as to how to change this bit.

There are several versions of parallel ports beyond the original design. The Enhanced Parallel Port (EPP) was created by Intel, Xircom and Zenith in 1991. EPP allows for much more data, 500 kilobytes to 2 megabytes, to be transferred each second. The Standard Parallel Port (SPP) and has completely replaced the original design. Bidirectional communication allows each device to receive data as well as transmit it. Pins 18 through 25, originally just used as grounds, can be used as data pins also. This allows for full-duplex (both directions at the same time) communication. The Extended Capabilities Port (ECP) allows for bidirectional

Pin #	Name	Input/Output Bit	Inverted
1	Stroke	Out	Control 0
			Yes

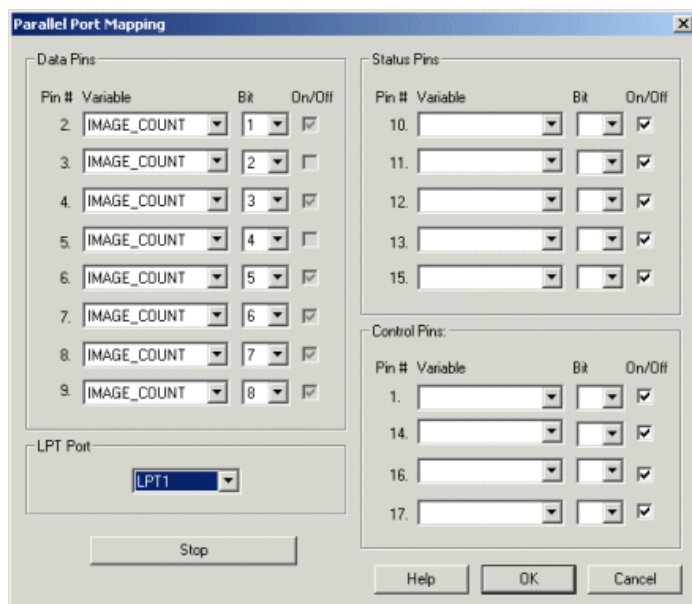
1	nStrobe	Out	Control-0	Yes
2	Data0	In/Out	Data-0	No
3	Data1	In/Out	Data-1	No
4	Data2	In/Out	Data-2	No
5	Data3	In/Out	Data-3	No
6	Data4	In/Out	Data-4	No
7	Data5	In/Out	Data-5	No
8	Data6	In/Out	Data-6	No
9	Data7	In/Out	Data-7	No
10	nAck	In	Status-6	No
11	Busy	In	Status-7	Yes
12	Paper-Out	In	Status-5	No
13	Select	In	Status-4	No
14	Linefeed	Out	Control-1	Yes
15	nError	In	Status-3	No
16	nInitialize	Out	Control-2	No
17	nSelect-Printer	Out	Control-3	Yes
18	Ground	Out	(Control-4)	No
19	Enable Bi-Directional Port	Out	(Control-5)	No
20-25	Ground	-	-	-

Keep in mind that the initial purpose of the parallel port was to interface the PC with a computer. So the pin name's will reflect that specific purpose. If you are using the parallel port to connect with your own devices the main quality of the pins that will be important to you will be whether they are input or output pins. I.e. if you using Pin 11 the "Busy" pin to input a bit of data from your circuit it does not have to reflect that your circuit is 'busy'.

Most of the bit values are logically on when the voltage level is high but 4 of the bits are swapped.

Note that interfacing directly with the Parallel Port using WinXP, Win2000, etc is not permitted by the OS. You need to have a system device driver to interface with the parallel port. RoboRealm uses the popular [inpout32.dll from LOGIX4U](#) to achieve this.

Interface



Instructions

1. Specify which variables should match to which bits on the parallel port. If no variable is selected you can toggle the checkbox for each pin manually. Changing the checkbox should change the appropriate pin on the parallel port. If a variable is selected you will need to select which bit within the variables value would represent the pin value. For example if your variable value is 2 and your selected bit is 1 then the appropriate pin would be 0. If the selected bit is 2 then the appropriate pin would be 1. Thus to send an 8 bit number across the parallel port you would select the bits as seen in the interface shot above.
2. Select the parallel port number. The default is LPT1

Note on some machines the parallel port address space is different. In RoboRealm the following address' are used in correlation to the LPTx

dropdown. If your machine configuration is different you may have to specify a different parallel port in the dropdown menu than what you believe to be true in order to specify the correct address space. Note that RoboRealm assumes a default configuration. Virtual parallel ports and such will change that base configuration.

lpt1Data = 0x378
lpt1Status = 0x379
lpt1Control = 0x37a

lpt2Data = 0x278
lpt2Status = 0x279
lpt2Control = 0x27a

lpt3Data = 0x3bc
lpt3Status = 0x3bd
lpt3Control = 0x3be

You can check the LPT1 address used by your computer by right clicking on "My Computer", select "Manage", click on "Device Manager" subtree, then expand "Ports (COM & LPT)", right click on the printer port (LPT1) and select "Properties". The I/O address is in the "Resources" tab. Note the first address only. Match that address with one above and select the appropriate LPT1-3 from the dropdown.

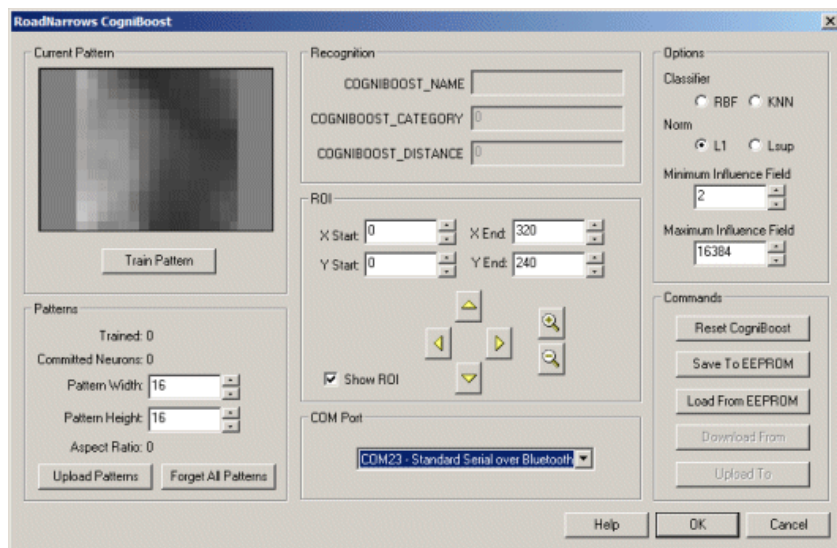
RoadNarrows CogniBoost

The RoadNarrows CogniBoost module provides an interface to the CogniBoost vision recognition board. Powered by the CogniMem CM-1K neural network chip by General-Vision, visual objects are readily detected and categorized from patterns previously trained. The recognized visual categories are then fed into RoboRealm to guide goal decisions, object localization, platform movement and navigation, and environment manipulation.

The CogniBoost RoboRealm module can be used to train patterns on the board. Recognized object categories are fed back into RoboRealm as variables that can be used by other modules or transported to external application/devices.

The module sends images to be recognized to the CogniBoost board. This allows images to be prefiltered, oriented, etc. using all of RoboRealm's other modules in order to improve recognition results or focus recognition on specific image aspects.

Interface



Instructions

1. COM Port - Select the appropriate COM port that is connected to the CogniBoost board. If the appropriate COM port does not appear ensure that all connections are made and that the CogniBoost board is turned on. Close the module and reopen to see if the COM port appears in the dropdown list.
2. Train Pattern - Pressing this button will cause CogniBoost to train on the currently viewed image. This image is stored in CogniBoost and will immediately start detecting the presence of that object. You should see the Recognition values being populated as soon as an object is trained.
3. Pattern Width & Height - The CogniBoost works with 256 'pixel' values. In order to create a two dimensional image a width and height of a pattern is required. If you find that your patterns are better detected in a more vertical or horizontal shape as apposed to the default square pattern

you can change the width and height appropriately. Note, however, when you change these parameters you WILL lose all currently trained patterns.

4. Upload Patterns To CogniBoost - uploads images from the PC to the CogniBoost board. Note that these images MUST have the same dimensions as what is currently configured in the CogniBoost board based on the ROI settings. See Pattern Width and Pattern Height for what this size needs to be. Note that images can be of any recognized image format but it is recommended to use a non-lossy image compression technique such as gif or png in order to store images. This ensures that the trained pattern is exactly the same as the stored image. Note that you may wish to "Forget All Patterns" before uploading to ensure that only those images uploaded are contained with CogniBoost.

5. Forget All Patterns - Causes CogniBoost to forget all trained images and set its pattern counter to zero. This should be used before training on valid images to clear out any experimental patterns that you may have trained on. This should also be used prior to uploading patterns to CogniBoost to clear out existing patterns. If you do NOT use this function prior to uploading patterns new patterns will be added to the current list within CogniBoost possibly duplicating existing patterns.

6. ROI - To focus on a specific area of an image you can change the ROI (seen as a red square in the live image) to change what part of the image is focused on. Often it is not possible to align the camera on the expected object position such that the object completely fills the field of view. When this happens adjusting the ROI to ignore non-object areas of the image can help improve object recognition. Note that the ROI area is scaled into the pattern width and height before sending it to the CogniBoost.

Note that changing these parameters may invalidate image patterns that you have trained on if the aspect ratio changes from what was trained. This is due to the scaling requirement to map the ROI into the pattern size.

X Start - the x (horizontal) start coordinate

Y Start - the y (vertical) start coordinate

X End - the x (horizontal) end coordinate

Y End - the y (vertical) end coordinate

Arrows - use the provided arrows to move the ROI as a whole around the image. Note that moving the entire ROI will NOT affect the CogniBoost image size and therefore will not affect the recognition of previously trained patterns.

Zoom - use the provided zoom and un-zoom buttons to change the field of view of the ROI. As the zoom will preserve the image aspect ratio your previous image patterns will work.

7. Options

Classifier - K-Nearest Neighbor or Radial Basis Function. The KNN classifier discards the relationship between the distance and influence field of a neuron. As a consequence, all the neurons fire and their distance and category can be read in sequence per increasing order of distance. The RBF classifier uses the Influence Field of the neurons at the time of the recognition. A neuron fires only if the distance calculated between the input vector and its vector in memory is less than its influence field.

Norm - Determines how to calculate the distance between the reference pattern or prototype stored in a neuron and an input pattern. (Manhattan distance versus Euclidian distance))

Minimum Influence Field - Defines the "Uncertainty" zones in the decision space.

Maximum Influence Field - The default value of a newly committed neuron when no other neuron recognizes the vector to learn

8. Reset CogniBoost - You can reset the CogniBoost board by pressing the "Reset CogniBoost" button. This will cause CogniBoost to forget all patterns in volatile memory and reload all its settings.

9. Save To EEPROM/Load From EEPROM - When training or uploading images or changing camera settings all these changes are stored in volatile memory on the CogniBoost board. This means that if you turn off the CogniBoost board all these settings are lost. To save these settings such that they are used again once the CogniBoost board is turned on you need to "Save to EEPROM". Likewise, if you wish to reload all saved settings (to erase all current settings stored in memory) you can press the "Load From EEPROM" button.

Variables

COGNIBOOST_NAME_X - The category name associated with the recognized category. Note that this is a RoboRealm specific association and is NOT present on the CogniBoost board. Up to four categories can be

recorded at one time. Note that they are ordered with regards to distance with the first being the closest best match.

COGNIBOOST_CATEGORY_X - The numerical category CogniBoost has assigned to the trained pattern. When the pattern is presented again to CogniBoost this category number will become available as a recognition identification code that the object is present.

COGNIBOOST_DISTANCE_X - A measure of recognition confidence or the "distance" of the current recognized pattern to its associated pattern stored in CogniBoost.

RoadNarrows RoboSight

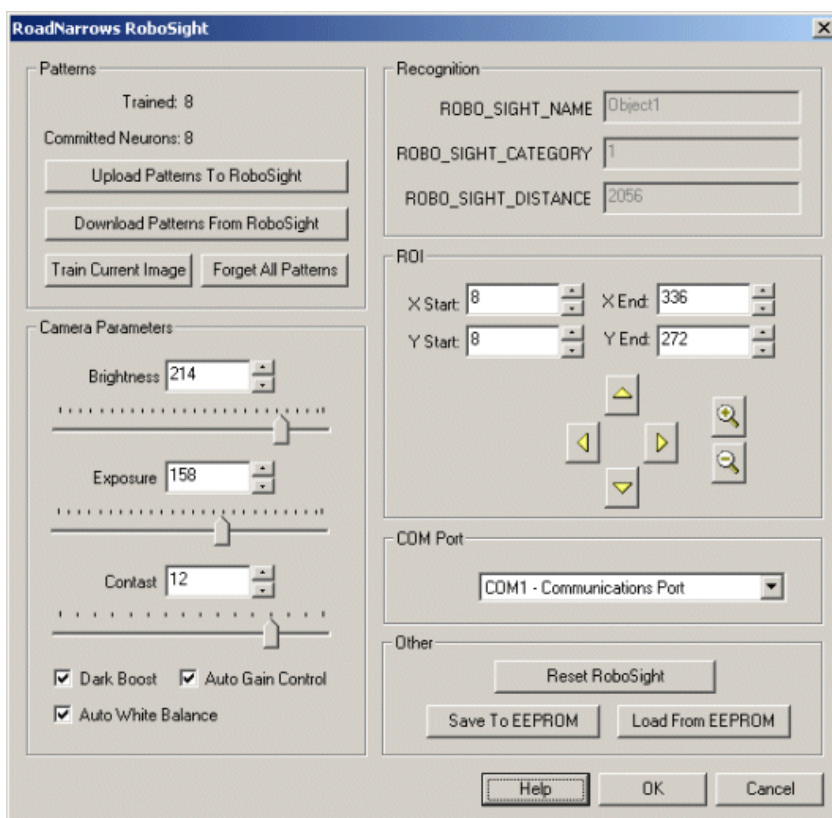


The RoadNarrows RoboSight module provides an interface to the RoboSight vision recognition board. Powered by the CogniMem CM-1K neural network chip by General-Vision, visual objects are readily detected and categorized from patterns previously trained. The recognized visual categories are then fed into RoboRealm to guide goal decisions, object localization, platform movement and navigation, and environment manipulation.

The RoboSight RoboRealm module can be used to configure the RoboSight camera properties as well as upload, download and train patterns on the board. Recognized object categories are fed back into RoboRealm as variables that can be used by other modules or transported to external application/devices.

Note that the RoboRealm RoboSight board assumes that the composite video connection on the RoboSight board is used to capture an image into the PC. You will need a NTSC digitizer such as a TV tuner that supports a 352x288 image size. Generic digitizers will work but may not support the 352x288 image size which will cause the viewed image on the PC to be somewhat distorted. This, however, does NOT affect RoboSight's ability to recognize objects as the PC view is just used for camera positioning and preview.

Interface



Instructions

1. COM Port - Select the appropriate COM port that is connected to the RoboSight board. If the appropriate COM port does not appear ensure that all connections are made and that the RoboSight board is turned on. Close the module and reopen to see if the COM port appears in the dropdown list.

2. Camera Parameters - Use the following controls to change the camera properties within RoboSight. Note that this assumes the NTSC out on the RoboSight board is being previewed via a digitizer on the PC. Changing camera properties will affect both the NTSC image and the image fed to the CogniMem vision chip.

Brightness - Adjusts the overall brightness of the image. If this value does not improve the overall brightness enough see the "Dark Boost" checkbox below.

Exposure - Adjusts the shutter speed of the RoboSight camera.

Contrast - Adjusts the contrast (amount of black and white) in the image.

Dark Boost - Improves the image light level when used in low light environments.

Auto Gain Control - Allows RoboSight to automatically adjust the image gain and compensate for irregular environmental lighting.

Auto White Balance - Allows RoboSight to automatically adjust the white balance of the image.

3. Patterns - Several buttons allow for the setup of the RoboSight board.

Train Current Image - Pressing this button will cause RoboSight to train on the currently viewed image. This image is stored in RoboSight and will immediately start detecting the presence of that object.

Forget All Patterns - Causes RoboSight to forget all trained images and set its pattern counter to zero. This should be used before training on valid images to clear out any experimental patterns that you may have trained on. This should also be used prior to uploading patterns to RoboSight to clear out existing patterns. If you do NOT use this function prior to uploading patterns new patterns will be added to the current list within RoboSight possibly duplicating existing patterns.

Upload Patterns To RoboSight - uploads images from the PC to the RoboSight board. Note that these images MUST have the same dimensions as what is currently configured in the RoboSight board based on the ROI settings. See ROBOSIGHT_WIDTH and ROBOSIGHT_HEIGHT variables for what this size needs to be. Note that images can any recognized image format but it is recommended to use a non-lossy image compression technique such as gif or png in order to store images. This ensures that the trained pattern is exactly the same as the stored image. Note that you may wish to "Forget All Patterns" before uploading to ensure that only those images uploaded are contained with RoboSight.

Download Patterns From RoboSight - downloads all trained patterns from RoboSight and stores them into a user specified folder. Note that the downloaded patterns will be stored as small (typically 16x16) gif images. These images can be modified using any paint application and saved for uploading back to RoboSight.

4. ROI - Changing the ROI or Region of Interest of RoboSight will cause the recognition to change what part of the image is focused on. Often it is not possible to align the camera on the expected object position such that the object completely fills the field of view. When this happens adjusting the ROI to ignore non-object areas of the image can help improve object recognition.

Note that the RoboSight image size WILL change when changing the width and height of the ROI area. Changing these parameters may invalidate image patterns that you have downloaded by not allowing you to upload the images back into RoboSight. Thus you should first set the ROI BEFORE training any valid patterns and downloading them to the PC.

X Start - the x (horizontal) start coordinate

Y Start - the y (vertical) start coordinate

X End - the x (horizontal) end coordinate

Y End - the y (vertical) end coordinate

Arrows - use the provided arrows to move the ROI as a whole around the image. Note that moving the entire ROI will NOT affect the RoboSight image size and therefore will not affect the ability to upload existing images.

Zoom - use the provided zoom and un-zoom buttons to change the field of view of the ROI. As the zoom will preserve the image aspect ratio your image pattern size will again not change.

5. (Other) Reset RoboSight - You can reset the RoboSight board by pressing the "Reset RoboSight" button. This will cause RoboSight to forget all patterns in memory and reload all its settings.

6.(Other) Save To EEPROM/Load From EEPROM - When training or uploading images or changing camera settings all these changes are stored in volatile memory on the RoboSight board. This means that if you turn off the RoboSight board all these settings are lost. To save these settings such that they are used again once the RoboSight board is turned on you need to "Save to EEPROM". Likewise, if you wish to reload all saved settings (to erase all current settings stored in memory) you can press the "Load From EEPROM" button.

Variables

ROBOSIGHT_NAME_X - The category name associated with the recognized category. Note that this is a RoboRealm specific association and is NOT present on the RoboSight board. Up to four categories can be recorded at one time. Note that they are ordered with regards to distance with the first being the closest best match.

ROBOSIGHT_CATEGORY_X - The numerical category RoboSight has assigned to the trained pattern. When the pattern is presented again to RoboSight this category number will become available as a recognition identification code that the object is present.

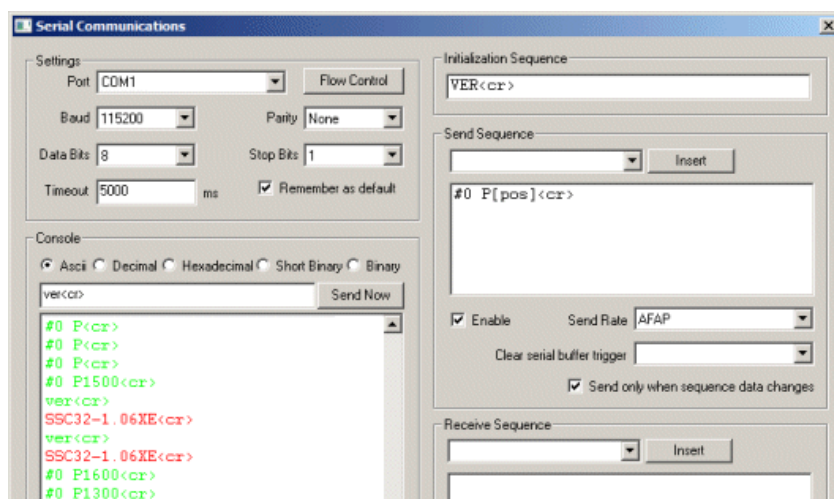
ROBOSIGHT_DISTANCE_X - A measure of recognition confidence or the "distance" of the current recognized pattern to its associated pattern stored in RoboSight.

Serial Communication

The serial module is used to communicate from RoboRealm to serial based controllers. The module can be used to communicate to SSC, Parallax, etc. type controllers. Before embarking on using the serial module check that the specific device is not already supported by another [RoboRealm module](#) as that will be easier than understanding and implementing a particular device's protocol.

The serial module allows you to configure the serial port for various communication parameters (baud rate, stop bits, etc.) and specify the communication protocol for initializing, sending and receiving data. The text boxes allow you to enter in the necessary command numbers for each respective sequence type (read or write).

Interface





Instructions

1. Port - Select the COM port to use. Note that COM1-4 are typically assigned directly to hardware ports. COM5+ are typically assigned to virtual ports created by USB devices.
2. Baud - Specifies how fast communication will occur between the PC and serial device. Typically rates are 9600 with the fastest being 115200. Note that if you see irregular characters being received from the serial device this may indicate a mismatch between baud rates.
3. Data Bits - Indicates how many bits are reserved for data communication between the serial device and the PC. Typical value is 8.
4. Parity - Indicates what kind of error checking that is performed during serial communication. Even means the last bit is set to ensure that an even number of bits are used, whilst odd means the last bit is set to ensure that an odd number of bits are used. When the receiving side sees a discrepancy (i.e. an odd number of bits being transmitted when even parity is required) the device knows a transmission error has occurred. Typical value is None.
5. Stop Bits - Refers to how many bits are used to indicate a delimiter between data bits.
6. Flow Control - Allows you to set the flow control options for the serial device. By default RoboRealm sets all flow control to off/disabled. By clicking on the Flow Control button you can set the following options
 - XON/XOFF for both transmission and reception
 - CTS/DRS (clear to send)/(data set ready)
 - DTR/RTS (data terminal ready)/(request to send)
7. Wait for reply - Forces the module to wait for a reply after characters have been sent. This ensures that the module pauses until a reply has been received from the remote device. Note, this can cause the pipeline to slow down due to the pause for a reply. Only select this checkbox if you require a reply before continuing the pipeline.
8. Session Timeout - when receiving information from a serial device this number represents how long the module will wait for a character until giving up and resetting the connection. Note that the value is in milliseconds, where 1 second = 1000 milliseconds. The default is 5000 or 5 seconds.
9. Message Timeout - When receiving information from a serial device this number represents how long the module will wait for the next character until giving up and assuming the message is complete. Note that the value is in milliseconds, where 1 second = 1000 milliseconds. The default is 100 or 0.1 seconds. This is very useful when no delimiter is specified and a complete message is determined by the lack of new characters for a small period.
10. Console - if the device supports reading you will start to see a large amount of numbers scrolling by in the console. This shows the current values being read from the device and provides a log of the ongoing communication activities. The green text is from RoboRealm, the red text are characters received and the green text are characters sent by RoboRealm to the device. To copy the log click on the Copy button, likewise to clear it click on the Clear button. Note that the console log only shows several lines at a time with older information being discarded. You can switch the output to various formats for better viewing:
 - ASCII - shows the data as ASCII characters. Any character outside of printable characters will be represented with a \ followed by the actual integer data that was read
 - Decimal - shows the data as sets of single byte integer numbers
 - Hexadecimal - shows the data as sets of hexadecimal numbers (base 16 numbers, e.g. FF)
 - Short Binary - shows the data as binary numbers with prefix 0's removed
 - Binary - shows the data as sets of single byte binary numbers
11. Send Now - often you need to quickly test the device by sending a certain sequence of characters. To do so just type in the character sequence (using for carriage return, [image_count] for variables, etc.) and click on Send Now. The text will be parsed and sent to the serial device and the response will then appear in the console log. Note that this is different from the send sequence which will be sent each time the serial module is encountered in the processing pipeline. The Send Now button is a manual testing mechanism meant for debugging purposes. Also note that the returned text will be parsed by the Receive Sequence so that you can test your parsing code and see if the variables have been created. Use the [Watch Variables](#) module to see those variables being created.
12. Refresh Rate - to slow the scrolling of the numbers select a different refresh rate for the console. This will just slow down how quickly

RoboRealm reads information from the device.

13. Initialization Sequence - The initialization data sequence sends the provided string to the serial device on initialization of communication. You may want to use this to command the receiving device into a specific mode ready for communication with RoboRealm. This initialization sequence is sent each time the serial communication is reset. This happens when you click on the "Stop" button in this interface, change one of the above parameters (Baud, Port, etc) or when the RoboRealm starts running for the first time. It is NOT sent when the Run button in the main RoboRealm interface is toggled. See the [Text Formats](#) page for additional information about the text string format.

14. Send sequence - Used to enter commands sent per pipeline loop (i.e. image processed) by RoboRealm. You can use this sequence to transmit variables created by other RoboRealm modules to the serial device. Each time an image is captured and processed the serial module will interpret the Send Sequence text and send the result to the serial device. See the [Text Formats](#) page for additional information about the text string format.

15. Enable - Allows you to temporarily disable sending text to the serial device while performing edits. Note that the Send Sequence textarea will turn red to indicate this setting.

16. Send Rate - Some serial devices cannot handle data streams even given a slower baud rate. Use this dropdown to select how quickly you want the data to be sent. At AFAP (As Fast As Possible) the data will be sent out about 30 times a second (this assumes a camera running at 30 fps).

17. Clear serial buffer trigger - there are cases when too much data may be buffered on the serial port that needs to be cleared due to some event. In this case specify a variable whose value when set to "clear" will remove all data scheduled to be sent over the serial port.

18. Send only on change - If your data does need to be sent out to your serial device every iteration through the processing pipeline loop this selection will prevent the same data from being sent to the serial device that was last sent. This is also an elegant way to reduce the data bandwidth to the device if your sequence does not change rapidly.

19. Receive sequence - used to receive and parse text send from the serial device. The text string is matched against the incoming serial bytes. When a match is found variables are added into RoboRealm for use in other modules. Reading into variables just requires adding in a variable at the appropriate spot within the receive string similar to the scanf routine in C/C++. The Receive sequence works similar to an expect string, i.e. you need to specify patterns that match the incoming text and substitute the areas that need to be fed into variables with the [variable_name] format. Note that even if you are missing one space or newline the patten will not match and the variable will be zero or blank. See the [Text Formats](#) page for additional information about the text string format.

20. Test Receive - instead of waiting for the correct data bytes to come from your serial device you can type in a test sequence that will be parsed by the receive sequence. This can be handy when first testing that your receive sequence is picking up the right bytes.

To learn more about what text can be used in the the Sequence boxes please refer to the [Text Formats](#) page.

Example

To read digital inputs from a [VEX Robotics System](#) robot over the serial port you can configure the settings to port COM5, baud 115200, 8 data bits, no parity, 1 stop bit with the sequences as:

Initialization Sequence

```
\x0f\x0f\x08\x40\xb8\x04
```

Send Sequence

```
\xaa\x55\x00\x00\x00\x0B\x00\x00
```

Receive Sequence

```
DIN [din1] [din2] [din3] [din4] [din5] [din6]<lf>
```

which would command the Vex in the online mode (be sure to download that code to your Vex robot) and read in the digital inputs into variables din1, din2, din3, etc. You can add the [Watch Variables](#) module to see these variables change when you press a bump switch on the Vex.

Note that the Vex online mode needs to be downloaded prior to using this serial sequence. Be sure your Vex is reacting to the online interface provided by Vex to ensure correct online operation.

See Also

[MCU Communicator](#)

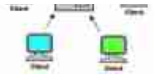
[SSC](#)

[Parallax](#)

Socket Client



The Socket Client module provides a generic interface to Socket based (network) servers. The Socket Client module should allow you to send and receive data to any network accessible server. Note that this is a generic module in that you will need to adhere to the protocol of the external server.

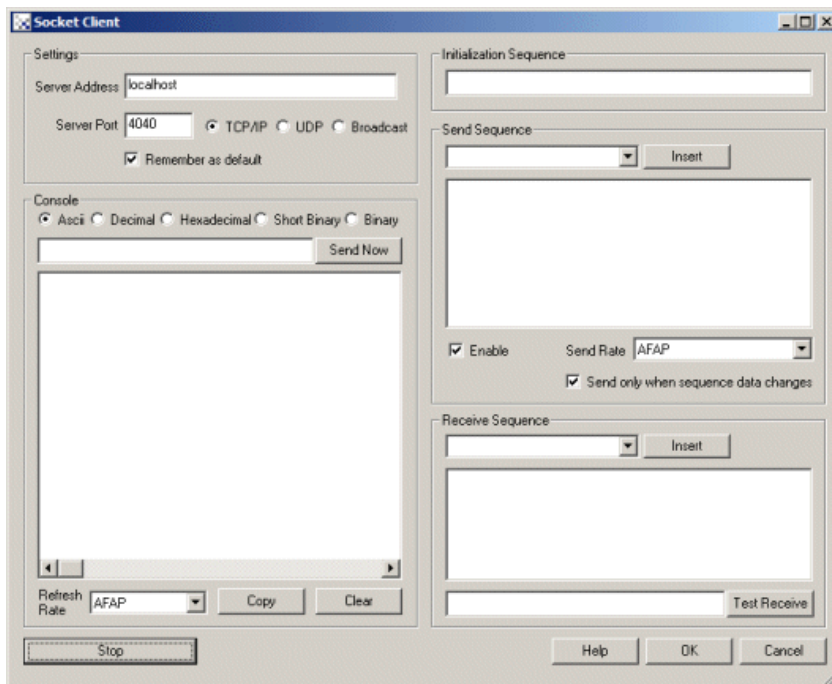


More specialized communications (like [HTTP](#)) have their own modules but cannot be used in servers that are not known to RoboRealm or created as custom products. This module provides the underlying basic networking connection to remote servers but requires that you specify what and how data is transmitted and received.

This module provides yet another way of integrating RoboRealm with external applications. For a more comprehensive overview please have a look at our [Integration](#) documentation.

Only **ONE** client connection is permitted at a time.

Interface



Instructions

1. Settings - Specify the name/IP address and port number of the server you would like to communicate to.
2. Method - Specify what protocol you would like to use when sending information. Note that TCP/IP should work in most cases across the Internet as it is a standard protocol with built in error checking. UPD and Broadcast modes are quicker than TCP but do not guarantee delivery of the information and should only be used on local networks as most routers will block UDP traffic.
3. Console - If the server responds with any data you will start to see a large amount of numbers scrolling by in the console. This shows the current values being read from the server and provides a log of the ongoing communication activities. The red text are characters received from the server while the green text are characters sent by RoboRealm to the server. To copy the log click on the Copy button, likewise to clear it click on the Clear button. Note that the console log only shows several lines at a time with older information being discarded. You can switch the output to various formats for better viewing:

ASCII - shows new data as ASCII characters. Any character outside of printable characters will be represented with a \ followed by the actual integer data that was read

Decimal - shows new data as sets of single byte integer numbers

Hexadecimal - shows new data as sets of hexadecimal numbers (base 16 numbers, e.g. FF)

Short Binary - shows new data as binary numbers with prefix 0's removed

Binary - shows new data as sets of single byte binary numbers

4. Send Now - often you need to quickly test the server by sending a certain sequence of characters. To do so just type in the character sequence (using for carriage return, [image_count] for variables, etc.) and click on Send Now. The text will be parsed and sent to the server whose response will then appear in the console log. Note that this is different from the send sequence which will be sent each time the module is encountered in the processing pipeline. The Send Now button is a manual testing mechanism meant for debugging purposes. Also note that the returned text will be parsed by the Receive Sequence so that you can test your parsing code and see if the variables have been created. Use the [Watch Variables](#)

module to see those variables being created.

5. Refresh Rate - to slow the scrolling of the numbers select a different refresh rate for the console. This will just slow down how quickly RoboRealm reads information from the server.

6. Initialization Sequence - The initialization data sequence sends the provided string to the server on initialization of communication. You may want to use this to command the receiving server into a specific mode ready for communication with RoboRealm. This initialization sequence is sent each time the communication is reset. This happens when you click on the "Stop" button in this interface or when the RoboRealm starts running for the first time. It is NOT sent when the Run button in the main RoboRealm interface is toggled. See the [Text Formats](#) page for additional information about the text string format.

7. Send sequence - Used to enter commands sent per pipeline loop (i.e. image processed) by RoboRealm. You can use this sequence to transmit variables created by other RoboRealm modules to the server. Each time an image is captured and processed the Socket Client module will interpret the Send Sequence text and send the result to the server. See the [Text Formats](#) page for additional information about the text string format.

8. Enable - Allows you to temporarily disable sending text to the server while performing edits. Note that the Send Sequence textarea will turn red to indicate this setting.

9. Send Rate - Some servers cannot handle rapid data streams. Use this dropdown to select how quickly you want the data to be sent. At AFAP (As Fast As Possible) the data will be sent out about 30 times a second (this assumes a camera running at 30 fps).

10. Send only on change - If your data does need to be sent out to your server every iteration through the processing pipeline loop this selection will prevent the same data from being sent to the server that was last sent. This is also an elegant way to reduce the data bandwidth to the server if your sequence does not change rapidly.

11. Receive sequence - used to receive and parse text send from the server. The text string is matched against the incoming bytes. When a match is found variables are added into RoboRealm for use in other modules. Reading into variables just requires adding in a variable at the appropriate spot within the receive string similar to the scanf routine in C/C++. The Receive sequence works similar to an expect string, i.e. you need to specify patterns that match the incoming text and substitute the areas that need to be fed into variables with the [variable_name] format. Note that even if you are missing one space or newline the patter will not match and the variable will be zero or blank. See the [Text Formats](#) page for additional information about the text string format.

Example

1. [Download](#) this Socket Server program that can be run by unzipping and executing the Release\Socket.exe application. Note the source is included for this small application which effectively shows how to listen on a socket port in C/C++. Once run, this application attaches to port 4040 and waits for any incoming messages.

2. Launch RoboRealm, click on the Search tab, and enter in Socket. Double Click on Socket_Client that will add this module to the pipeline. Immediately the module will connect to the running socket server.

3. Type in characters into the Send Now text area and press the send button. You should see the text reflected in the Socket server's window. If you type in \128 which means send the byte 128 the Socket Server program will respond with "Hello!".

4. This example shows how RoboRealm can communication with an external application listening on a socket port. As is seen from this example the protocol is not defined and you can make it as simple or as complex as needed. Note that for very complex interactions you may instead want to create a [Plugin](#) or use the [API](#) as they provide similar means of communication but in a more defined way.

See Also

[Socket Server](#)

[Serial](#)

[USB HID](#)

Socket Server

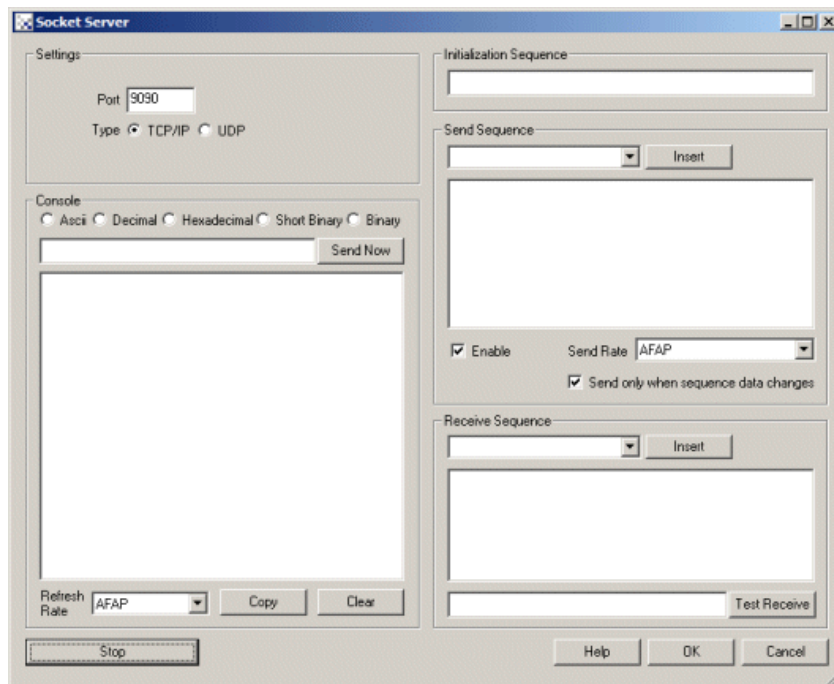
The Socket Server module provides a generic interface for Socket based (network) clients. The Socket Server module allows you to receive and send data from any network accessible client. Note that this is a generic module in that you will need to adhere to the data protocol of the external client.

More specialized communications (like [HTTP](#)) have their own modules but cannot be used in servers that are not known to RoboRealm or created as custom products. This module provides the underlying basic network server connection for remote clients but requires that you specify what and how data is transmitted and received.

Note that this module provides yet another way of integrating RoboRealm with external applications. For a more comprehensive overview please

have a look at our [Integration](#) documentation.

Interface



Instructions

1. Settings - Specify the port number the server should listen on for connections.
2. Method - Specify what protocol you would like to use when sending information. Note that TCP/IP should work in most cases across the Internet as it is a standard protocol with built in error checking. UPD is quicker (has less overhead) than TCP but does NOT guarantee delivery of the information and should only be used on local networks as most routers will block UDP traffic.
3. Console - If a client connects to the server you will start to see a large amount of numbers scrolling by in the console. This shows the current values being read from the client and provides a log of the ongoing communication activities. The red text are characters received from the client while the green text are characters sent by RoboRealm to the client. To copy the log click on the Copy button, likewise to clear it click on the Clear button. Note that the console log only shows several lines at a time with older information being discarded. You can switch the output to various formats for better viewing:

ASCII - shows new data as ASCII characters. Any character outside of printable characters will be represented with a \ followed by the actual integer data that was read

Decimal - shows new data as sets of single byte integer numbers

Hexadecimal - shows new data as sets of hexadecimal numbers (base 16 numbers, e.g. FF)

Short Binary - shows new data as binary numbers with prefix 0's removed

Binary - shows new data as sets of single byte binary numbers

4. Send Now - often you need to quickly test the client by sending a certain sequence of characters. To do so just type in the character sequence (using for carriage return, [image_count] for variables, etc.) and click on Send Now. The text will be parsed and sent to the server whose response will then appear in the console log. Note that this is different from the send sequence which will be sent each time the module is encountered in the processing pipeline. The Send Now button is a manual testing mechanism meant for debugging purposes. Also note that the returned text will be parsed by the Receive Sequence so that you can test your parsing code and see if the variables have been created. Use the [Watch Variables](#) module to see those variables being created.

5. Refresh Rate - to slow the scrolling of the numbers select a different refresh rate for the console. This will just slow down how quickly RoboRealm reads information from the server.
6. Initialization Sequence - The initialization data sequence sends the provided string to the client on initialization of communication. You may want to use this to command the receiving client into a specific mode ready for communication with RoboRealm. This initialization sequence is sent each time the communication is reset. This happens when you click on the "Stop" button in this interface or when the RoboRealm starts running for the first time. It is NOT sent when the Run button in the main RoboRealm interface is toggled.
7. Send sequence - Used to enter commands sent per pipeline loop (i.e. image processed) by RoboRealm. You can use this sequence to transmit variables created by other RoboRealm modules to the client. Each time an image is captured and processed the Socket Server module will interpret the Send Sequence text and send the result to the server. See the [Text Formats](#) page for additional information about the text string format.
8. Enable - Allows you to temporarily disable sending text to the server while performing edits. Note that the Send Sequence textarea will turn red to indicate this setting.
9. Send Rate - Some clients cannot handle rapid data streams. Use this dropdown to select how quickly you want the data to be sent. At AFAP (As Fast As Possible) the data will be sent out about 30 times a second (this assumes a camera running at 30 fps).
10. Send only on change - If your data does need to be sent out to your client every iteration through the processing pipeline loop this selection will prevent the same data from being sent to the client that was last sent. This is also an elegant way to reduce the data bandwidth to the client if your sequence does not change rapidly.
11. Receive sequence - used to receive and parse text send from the client. The text string is matched against the incoming bytes. When a match is found variables are added into RoboRealm for use in other modules. Reading into variables just requires adding in a variable at the appropriate spot within the receive string similar to the scanf routine in C/C++. The Receive sequence works similar to an expect string, i.e. you need to specify patterns that match the incoming text and substitute the areas that need to be fed into variables with the [variable_name] format. Note that even if you are missing one space or newline the patter will not match and the variable will be zero or blank. See the [Text Formats](#) page for additional information about the text string format.

Example

1. [Download](#) the Sensor Data Streamer for your iPhone/iPad and enter in your PC ip address and port 9090.
2. Launch RoboRealm, click on the Search tab, and enter in Socket Server. Double Click on Socket_Server that will add this module to the pipeline.
3. Select UDP as the method
4. Enter in **one** line

```
FS\w[ver]\f[acc_x]\f[acc_y]\f[acc_z]\f[gyro_x]\f[gyro_y]\f[gyro_z]\F[tes_x]\F
[tes_y]\F[tes_z]\F[mag_head]\F[true_head]\F[latitude]\F[longitide]\F[altitude]
```

into the Receive Sequence text box and press the START button to start the module.

5. This example shows how external applications can send data to RoboRealm. The Sensor Data Streamer will send UDP packets of accelerometer, gyro, etc. information collected on the iPhone/iPad to the PC over the network. This data stream is then interpreted by RoboRealm given the receive sequence. This shows how RoboRealm can be configured to receive information from external applications that have their own data protocol.

Note, if you add a [Watch](#) module you can see the specific variables change when you move your device.

As the Sensor Data Streamer app uses UDP you MAY not be able to receive packets from your iPhone as UDP packets are often blocked in WIFI networks. Also note that UPD is NOT a guaranteed protocol and without a CRC (checksum) there is no guarantee that the data from the iPhone is correct. Its very possible that only half the message be received which may cause momentary corruption of the data. Care should be taken when using this technique to steer or control large machinery!

See Also

[Socket Client](#)
[Serial](#)
[USB HID](#)

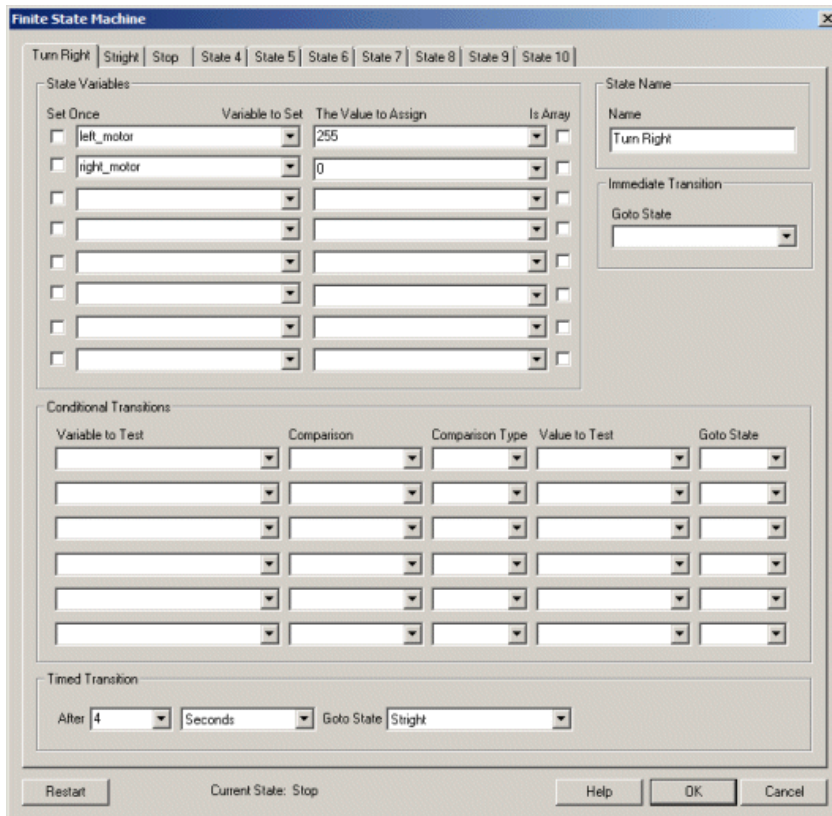


The State Machine (or Finite State Machine) module provides an easy way to define a set of states and transitions between those states that is often needed in robotic applications. The module defines a state as a set of variables that contain specific values. As RoboRealm communicates between modules using variables a state can define a specific action (like moving a robot forward). Transitions between those states are defined as conditional statements that check for specific values of other variables (such as the distance to an object from an IR sensor) in order to change the current state of the module. In this way you can define an action and its transition requirements in one interface.

For example, should you want a robot to move forward for 5 seconds, turn for 1 and then stop but also stop if at any time the IR sensor picks up an object close to the robot you can define multiple transitions. The primary transition would be that of time (i.e. transition to the next state after 5 seconds) and another transition would check the value of the IR sensor and skip over to the final step where the robot would stop.

By defining states and transitions you can create rudimentary behaviors for your robots based on its sensors and actuators.

Interface




Instructions


1. Name - A name that best represents the state. This is used for your purposes to help you identify the meaning of each state. For example, state names like "Driving Forward", or "Turning", etc. are useful.
2. State Variables - Define that variables should be set to what values when within this state. These variables would then be used in other modules to define speed or direction of movement or states of other actuators. Note that defining variables within this module will NOT cause any actions to occur unless those same variables are used in other modules that do communicate with specific hardware. Due to this, a state machine can be defined for one robot and then used on another by changing the hardware communication module (i.e. [IRobot_Roomba to Sparkfun_Arduino](#)).
3. Set Once - The module will set the state variables on each iteration of the pipeline. If the variable should be allowed to change (as in the case of speech generation) to avoid multiple reads select the "Set Once" checkbox. This will cause the checkbox to be set only once when the state is entered and not each time the processing pipeline iterates.
4. Is Array - Specifies that the value for the variable is an array. Each item is separated by a comma. On assignment this list is read, split into individual elements and assigned as an array.
5. Conditional Transitions - Define how the current state is transitioned into another. Typically this will be in reaction to a sensor value (again stored as a variable from a hardware module) or from a user input selection (as from the [Button](#) module).
6. Timed Transition - This defines a timed transition that is based on time instead of a particular variable. Thus it is possible to move from one state to another based on the passage of time.
7. Restart - Press the Restart button to clear the state machine and set the state back to state 1. Note that often a state machine never ends and can cycle through states without termination. In this case the Restart button may not be of value.

Examples

Robot Driving

 [Click Here](#) to download an example file that will change a left_motor and right_motor variable to turn right, straight and then stop. Note that NOTHING happens unless additional modules are added to send these variable values to a motor, servo, etc. If you add the [Watch Variable](#) module you can see these variables change as time progresses.

Speech

 [Click Here](#) to download an example file that will say hello and ask how you are. Based on a "fine" or "bad" response it will change the response. A speaker and a microphone are needed for this.

See Also

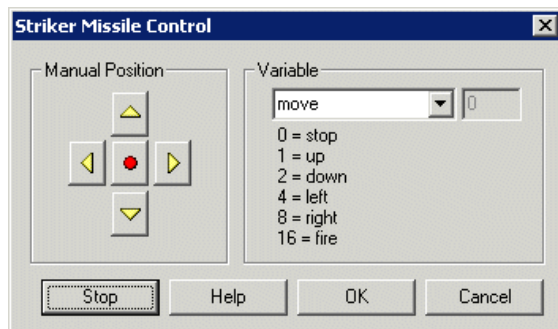
[If Statement](#)
[Set Variable](#)
[VBScript](#)
[CScript Program](#)
[Python Program](#)

Striker Missile Launcher



The Striker Missile module provides an interface to the Striker USB Missile Launcher. By specifying an appropriate variable the value will be sent to the device to control its movements. For testing purposes a manual interface (see yellow arrows below) has also been provided.


Interface



Instructions

1. Move the Striker Missile Launcher by pressing the appropriate arrows. The middle red button is for firing. Once the button is pressed the launcher will fire one of the missiles. Note that time between pressing the button and the actual firing takes a couple seconds.
2. Select the appropriate variable that contains the bits set based on what movement or action you want the device to perform. Note that multiple directions (up and left) can be combined into a single move command.

Example

 [Try this robo-file](#) to control your missile launcher using your keyboard.

The movement numbers are

0 = stop
1 = up
2 = down
4 = left
8 = right
16 = fire

Thus if you wanted the device to move to the right you could specify a variable like "move" and set that variable's value (using either a [VBScript](#)

[program](#) or the [Set_Variable](#) module) to 8. Note that the device keeps moving in the specified direction until that variable changes value. (like 0 for stop)

See Also

[Dream Cheeky Missile Launcher](#)

Trossen Robot Turret



The Trossen Robot Turret module provides an interface to the Trossen Robot Turret Pan/Tilt systems created and sold by [Trossen Robotics](#). The pan/tilt platform provides a very complete and easy way to integrate pan and tilt functionality into your robotics project. The starter Desktop version is an Arduino based Robot Turret board providing 2 servo PWM signals for the pan/tilt system and also has 2 motor controller ports, 3 analog in pins and 5 digital input/output pins. This provides enough control capability for whatever you happen to chose to attach to the top of the pan/tilt system.

The higher end versions starting with the PhantomX version offers more servo strength (can support more weight on the end of the arm). more digital IO, more analog IO and is based on the RobotiX line of controllers.

This module provides an interface to these platforms and provides you the ability to control or read in all inputs from the boards using RoboRealm variables. While it's a lot of fun to use the manual controls to move the system around the variables associated with each control is what provides the automation capabilities for the system.

Interface

Trossen Robot Turret Controller

Pan/Tilt Motors Digital IO Analog Input

Variable Current Value Min/Max Limits

X 438 0 1020

Y 518 0 1020

Invert Y

Variable Value

SSR

Robot Turret

Desktop (MosquitIO) PhantomX

COM Port

COM191 - USB Serial Port

Remember as default

Protocol

Version 1.1 Version 1.0

Stop Help OK Cancel

Pan/Tilt Motors Digital IO Analog Input

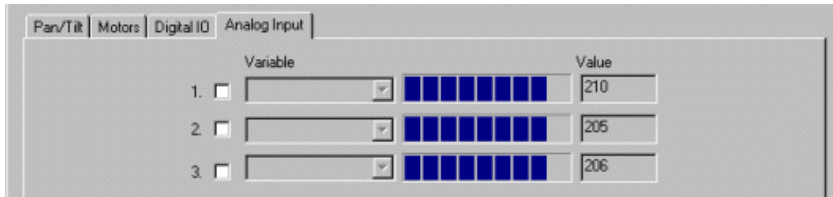
Variable Current Value Min/Max Limits

Motor 1 128 0 255

Motor 2 128 0 255

Pan/Tilt Motors Digital IO Analog Input

Pin #	Variable	Bit	Value	Input	Output
D1	test	3	<input type="checkbox"/>	<input checked="" type="radio"/>	<input type="radio"/>
D2			<input type="checkbox"/>	<input checked="" type="radio"/>	<input type="radio"/>
D3			<input type="checkbox"/>	<input checked="" type="radio"/>	<input type="radio"/>



Instructions

1. Com Port - Specify the COM port that your turret is connected to. Note that the turret comes with a USB cable that simulates a serial port on your PC. The port number for these kinds of virtual ports are normally above the standard 1-4 physical ports.
 2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current COM setting to be remembered by RoboRealm such that whenever the Robot_Turret module is loaded the COM port will be auto-populated to the current setting. This ability allows you to not have to constantly change the COM port setting when loading in successive RoboRealm robofile configurations.
 3. Turret Type - Select which Robot Turret you wish to control. Note that there are some differences between the turrets besides servo power. Some have more/less digital and analog IO.
 4. Protocol - The Desktop Robot Turret board is currently in version 1.1. If you have an order board you can try selecting the version 1.0 protocol which handles the Digital IO differently than version 1.0. Despite which version you may have the Pan/Tilt/M3 will work regardless as the protocol change does not affect these areas.
 5. Pan/Tilt - Once you have selected the appropriate COM port, click and drag anywhere in the white area to move the pan/tilt system to those coordinates. You can also drag the sliders on the side to move each axis separately.
 6. Variables - Select the appropriate variables that contain or will contain the position value that will be sent to the board. This is used to automatically change the servo values based on your VBScript/Python/etc. (using the SetVariable function) or Plugin based program. Note that once selected the manual controls are disabled to avoid interference.
 7. Current Value - To manually set the servo position type in the appropriate number (0-255, 128 is the default neutral) into the text area or use the slider to adjust the value. The servo position will be updated as appropriate. Note that the current limits of 0 and 255 will most likely be beyond the capabilities of your pan/tilt system so you may want to tighten the range.
 8. Min/Max Limits - You can also use the min/max limits to ensure that even if the variables specify large or low values (due to programming errors) that the board does not actually attempt to move the servos above or below the specified limits. This can be used as an additional precaution in case your servos cannot physically move beyond certain limits.
 9. SSR - Select a variable that will contain a non-zero value that will turn on the SSR led on the Robot_Turret board. To see where this LED is select the SSR checkbox which will cause the LED to turn on and off. This can be useful if you want to indicate some condition on the board without having a monitor nearby.
 10. Motors - Similar to the servo interface the motor interface provides the same controls that can be used to manually move the motors by changing the current value by typing in the number, using the up/down scroll arrows or by using the sliders. The Min and Max settings are also provided to limit the range of possible values that will be fed to the motors. Finally, the variable selection can be used to select a variable that will have the appropriate motor values to send to the motors automatically. Note selecting a variable will disable all the manual controls.
- See [Variable Control](#) for more information on how to programmatically move the robot.
11. Digital Pins - The Robot_Turret comes with 5 digital IO pins that can be configured as an input or an output. Unlike the Analog pins, Digital pins can only receive a 1 or 0 (on or off) as apposed to a full numeric value. Likewise, a digital pin set to an output can only send an on or off to the pin similar to a switch.

Each pin can either be set as an input or output. Do this by selecting the "input" or "output" radio button next to each pin. If you are specifying a pin as an output selecting the checkbox will then turn that pin high or low. Be sure the enable the pin (first checkbox) to enable the controls for that pin.

When specified as an input the pin's value checkbox will then reflect the read high or low state of the pin but will remain disabled.

To automatically send or receive a bit select or type in a variable that will be set if the pin is configured as an input or read into if the pin is configured as an input. Note that you can tell RoboRealm which bit of the variable you want to set/get by using the provided bit dropdown.


As an experiment, you can connect an LED to pin 1 and then select the pin as an "out". By checking and un-checking the checkbox you can make the LED blink.


If you then select the IMAGE_COUNT variable (which holds the current image counter) as the variable and select Bit 0 the LED will blink for every two times the processing pipeline is run. Selecting successively higher bits will slow the blinking by a factor of 2 for every bit.


12. Analog - If you want to sample the analog values specify a variable or type one in that will contain the value from the appropriate pin. Note that you must select the checkbox next to the Analog Pin in order to activate this functionality. As soon as the pin is activated the value read at that pin will be displayed in the progress bar and text area. Note that even when unattached the pin will show a value due to noise.


Once the value is placed within the variable you can then use this value in other modules, or accessed via the API, extension, etc. in order to move the value outside of the RoboRealm application.


Examples


 [Click Here](#) to load a robofile that will cause the Desktop Turret to track red objects. (Note this assumes 320x240 resolution!)


 [Click Here](#) to load a more sophisticated robofile that will cause the Desktop Turret to keep moving in the last known direction of red objects. This helps when objects are moving very quickly and move out of the frame. (Note this assumes 320x240 resolution!)

 [Click Here](#) to load a robofile that will cause the Desktop Turret to keep turn towards movement. Note that a lot of movement will confuse the turret and cause it to move erratically. If you want it to follow your hand try wiggling your fingers in front of the camera after it has stabilized and stopped moving. (Note this assumes 320x240 resolution!)

 [Click Here](#) to load a robofile that will cause the Interbotix line of Turrets (PhantomX, WidowX, etc) to track red objects. (Note this assumes 320x240 resolution!)

 [Click Here](#) to load a robofile that will cause the Interbotix line of Turrets (PhantomX, WidowX, etc) to track red objects with the laser on. The will cause the laser to point to the object that it is tracking! (Note this assumes 320x240 resolution!)

 [Click Here](#) to load a robofile that will cause the Interbotix (PhantomX, WidowX, etc) Turret to keep turn towards movement. Note that a lot of movement will confuse the turret and cause it to move erratically. If you want it to follow your hand try wiggling your fingers in front of the camera after it has stabilized and stopped moving. (Note this assumes 320x240 resolution!)

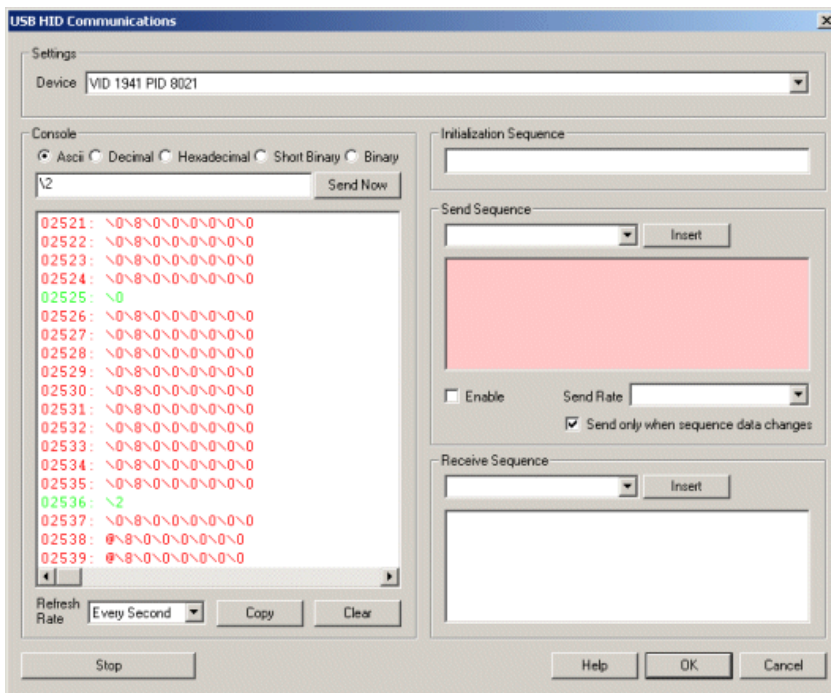
 [Click Here](#) to load a robofile that will cause the Interbotix (PhantomX, WidowX, etc) Turret to move towards movement. This is meant to be used with a stationary camera that will trigger the turret movement and laser on to target the movement. When it senses movement it will move the turret to point to the movement location for 3 seconds and then return the turret to the neutral position and wait again for movement. (Note this assumes 320x240 resolution!)

USB HID Interface

The USB HID module provides a generic interface to HID (Human Interface Devices) USB products. The USB HID module should allow you to control many devices that utilize this kind of USB driver that require simple exchanging of information otherwise not possible within RoboRealm.

Note that only HID devices will appear in the dropdown and that not all USB devices use the HID class.

Interface



Instructions

1. Settings - Select the device you wish to control from the device dropdown. The name of the device may have human readable words but often may only contain the VID (vendor ID) and PID (product ID) of the device. If you are not sure which entry is the appropriate device check with the manufacturer to find out the appropriate VID and PID to use.

2. Console - if the device supports reading you will start to see a large amount of numbers scrolling by in the console. This shows the current values being read from the device and provides a log of the ongoing communication activities. The red text are characters received from the device while the green text are characters sent by RoboRealm to the device. To copy the log click on the Copy button, likewise to clear it click on the Clear button. Note that the console log only shows several lines at a time with older information being discarded.

You can switch the output to various formats for better viewing:

ASCII - shows new data as ASCII characters. Any character outside of printable characters will be represented with a \ followed by the actual integer data that was read

Decimal - shows new data as sets of single byte integer numbers

Hexadecimal - shows new data as sets of hexadecimal numbers (base 16 numbers, e.g FF)

Short Binary - shows new data as binary numbers with prefix 0's removed

Binary - shows new data as sets of single byte binary numbers

3. Send Now - often you need to quickly test the device by sending a certain sequence of characters. To do so just type in the character sequence (using for carriage return, [image_count] for variables, etc.) and click on Send Now. The text will be parsed and sent to the device and the response will then appear in the console log. Note that this is different from the send sequence which will be sent each time the module is encountered in the processing pipeline. The Send Now button is a manual testing mechanism meant for debugging purposes. Also note that the returned text will be parsed by the Receive Sequence so that you can test your parsing code and see if the variables have been created. Use the [Watch Variables](#) module to see those variables being created.

4. Refresh Rate - to slow the scrolling of the numbers select a different refresh rate for the console. This will just slow down how quickly RoboRealm reads information from the device.

5. Initialization Sequence - The initialization data sequence sends the provided string to the device on initialization of communication. You may want to use this to command the receiving device into a specific mode ready for communication with RoboRealm. This initialization sequence is sent each time the communication is reset. This happens when you click on the "Stop" button in this interface, change one of the above parameters (Baud, Port, etc) or when the RoboRealm starts running for the first time. It is NOT sent when the Run button in the main RoboRealm interface is toggled. See the [Text Formats](#) page for additional information about the text string format.

6. Send sequence - Used to enter commands sent per pipeline loop (i.e. image processed) by RoboRealm. You can use this sequence to transmit variables created by other RoboRealm modules to the USB device. Each time an image is captured and processed the USB HID module will interpret the Send Sequence text and send the result to the device. See the [Text Formats](#) page for additional information about the text string format.
7. Enable - Allows you to temporarily disable sending text to the device while performing edits. Note that the Send Sequence textarea will turn red to indicate this setting.
8. Send Rate - Some devices cannot handle rapid data streams. Use this dropdown to select how quickly you want the data to be sent. At AFAP (As Fast As Possible) the data will be sent out about 30 times a second (this assumes a camera running at 30 fps).
9. Send only on change - If your data does need to be sent out to your device every iteration through the processing pipeline loop this selection will prevent the same data from being sent to the device that was last sent. This is also an elegant way to reduce the data bandwidth to the device if your sequence does not change rapidly.
10. Receive sequence - used to receive and parse text send from the device. The text string is matched against the incoming bytes. When a match is found variables are added into RoboRealm for use in other modules. Reading into variables just requires adding in a variable at the appropriate spot within the receive string similar to the scanf routine in C/C++. The Receive sequence works similar to an expect string, i.e. you need to specify patterns that match the incoming text and substitute the areas that need to be fed into variables with the [variable_name] format. Note that even if you are missing one space or newline the patter will not match and the variable will be zero or blank. See the [Text Formats](#) page for additional information about the text string format.

Example

The above interface screenshot shows controlling the Dream Cheeky USB Missile launcher. The launcher normally responds with 8 zeros that indicate that an extreme position has been met. Try connecting the device to your computer and send a \2 using the Send Now text box. Be sure to have the \0 typed in quickly afterwards to stop the motion. Likewise try \1 or \4 or \8 to move the launcher around. See how the commands you type in appear as green in the console?

[Here](#) is a robofile that shows how to use the USB HID module to prevent the launcher from moving past its limits and how to use the read sequence to read the information coming back from the launcher.

See Also

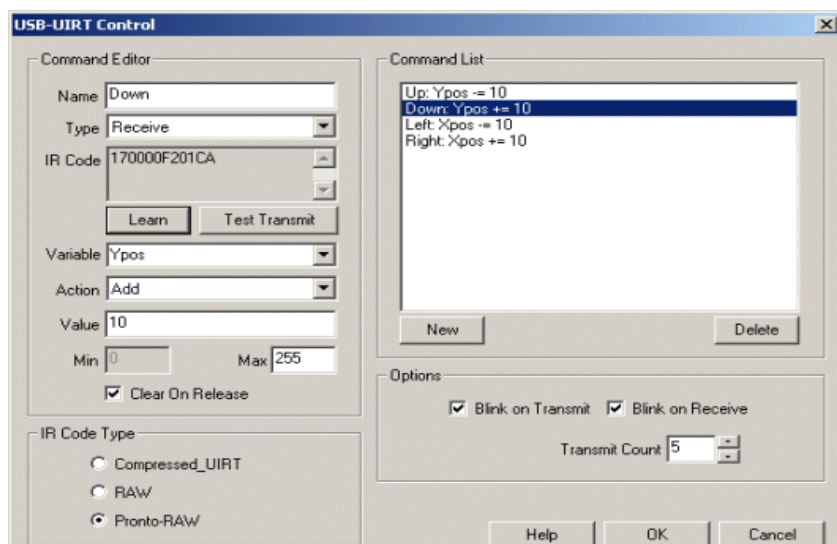
[Dream Cheeky Missile Launcher Serial](#)

USB-UIRT



The USB-UIRT module provides an interface to the [USB-UIRT](#) IR transmitter and receiver. This compact device can be used to record IR signals such as that generated by your TV remote control and then play them back based from your PC. The device opens up a wide range of control of such devices to your PC and ultimately under the control of RoboRealm. Besides TV remotes you can use the device to control most IR based robots such as the [Wowwee](#) line of robotic products.


Interface



Instructions

1. Command List - Click on the NEW button to create a new command configuration.
2. Name - Chose an appropriate name for you command that will help you to identify that command
3. Type - Select if the purpose is to transmit the code back out or if you are just watching for the presence of a code.
4. IR Code - Grab your remote, press the "Learn" button and keep it pressed until you see the IR Code populated. Note that it will just be a bunch of numbers that you will not understand. Press the "Test Transmit" button to retransmit the learned code to test if the correct code was learned.
5. Variable - The variable dropdown will either contain a signal as to when the code should be transmitted back out (Transmit Type) OR the name of the variable that will contain the value when a specific IR code is detected (Receive Type).
6. Action - On transmit the action will be used to compare the value of the variable to the specified value just below the action dropdown menu. In this way you can trigger the sending of the recorded code when the variable's content agrees with a simple formula. On receive the action will cause the variable's value to be changed in some way, i.e. adding, subtracting or just a straight setting.
7. Value - On transmit the value is used to test the specified variable to ensure that a transmission should occur. For example, when count = 10 send out a signal. On receive the value is used when changing the specified variable. For example, when the down button is pressed increase the YPos variable by 10.
8. Min, Max - specifies some hard limits that prevent the variable setting on receive from exceeding certain values.
9. Clear on Release - specifies that the variable that is either being used to transmit or receive will be cleared once the signal stops being sent. This will ensure that variables used to transmit codes will be cleared after they are sent, and ensures that variables receiving codes are also cleared once the receiving signal is stopped.
10. IR Code Type - select an appropriate IR code format. If your device does not work on RAW (default) try Pronto-RAW instead.

Example

 [Click here](#) to load a configuration to move your mouse using your remote control. Note that you will have to edit the USB_UIRT module and click on each of the command list items in order to learn your remote's code as it is most likely different from ours. You do this by clicking on a command list which will update the values to the left accordingly and then after clicking on the Learn button press the appropriate button on your remote control. The IR code will then be updated. Do this for each of the 4 buttons. After this you should be able to use the remote to control your mouse (assuming your physical mouse device is not moving).

See Also

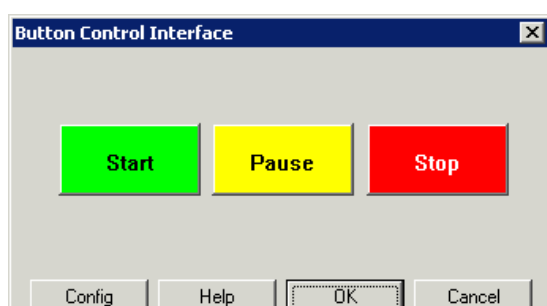
[Keyboard](#)

Button Interface

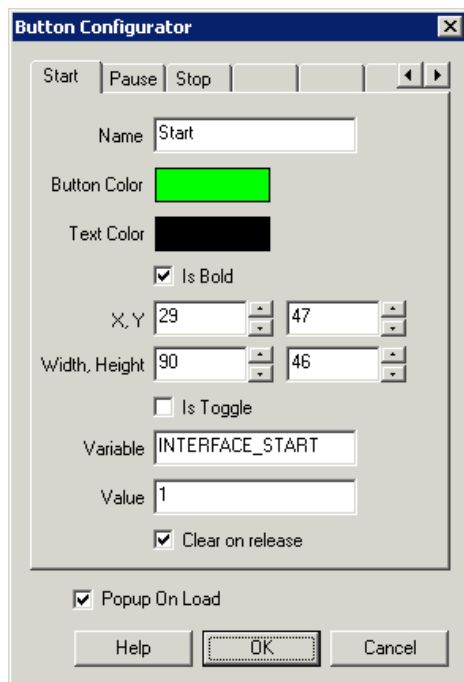
The Button Interface module provides you with a set of programmable buttons that can be made to create and reset variables when clicked. This allows you to create a single interface to set variables which can cause different actions elsewhere in the processing pipeline. This interface is very handy when you want to create a centralized stop or pause button that will stop your robot when needed.

Although the interface is currently restricted to just buttons a good deal of configuration is available by clicking on the config button.

Interface



Configuration Interface



Instructions

1. Buttons - Use the immediate interface to click on any of the buttons (maximum of 8 buttons) to set variables according to the button press.
2. Config - Click on the Config button to configure more buttons or change the behavior of the current default buttons.
3. Tabs - The configuration interface shows one tab per button. Click on the tab whose button you wish to configure.
4. Name - the display name of the button shown in the interface.
5. Button Color - the color of the background of the button
6. Text Color - the color of the text the Name will be displayed in
7. Is Bold - indicates if the button name is to be displayed in bold
8. X,Y - the x and y coordinates (position) of the button
9. Width, Height - the dimensions of the button
10. Is Toggle - determines if the button is a toggle button, i.e. it remains pressed until clicked again. When pressed the variable will retain its value, when released it will be removed.
11. Variable - the name of the variable to set when the button is pressed.
12. Value - the value to set the variable to when the button is pressed.
13. Clear on Release - indicates that the variable should be removed when the button is released or if the value is to remain after the button has been released. You will be responsible for removing the variable elsewhere within the pipeline in this case.
14. Popup on Load - causes the button interface to popup immediately once the program loads. This releases you from having to hunt and edit this module to view the interface.

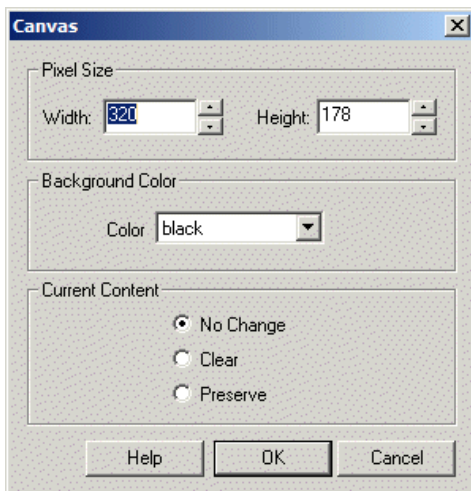
Example

[Click here](#) to load a configuration where you can view the created variables when a particular button is pressed.

Resize Canvas

The resize canvas module allows you to resize the image canvas. This allows you to create a blank image or realign the current image to a different width/height. This is different than scaling or resizing the pixels in that the current image is not scaled or changed. This module is also useful if you need to create a blank image of a specific size in order to place graphics/images onto the screen.

Interface



Instructions

1. Pixel Size - Select the width and height size that you want to change the image canvas to.
2. Background Color - The color used to Clear the image canvas to.
3. Current Content - No Change: Does not alter the current pixels. This will cause the current image to stretch or warp since the image canvas will change in dimension but the current image will not. This can be useful if you are trying to realign an image saved using the wrong width.
Clear - Clears the image area to the specified color
Preserve - Repaints the current image on the new canvas.
4. Position - Specify the relative location of the current image when the current content is to be preserved. This allows you to specify which side of the current canvas size will be extended.

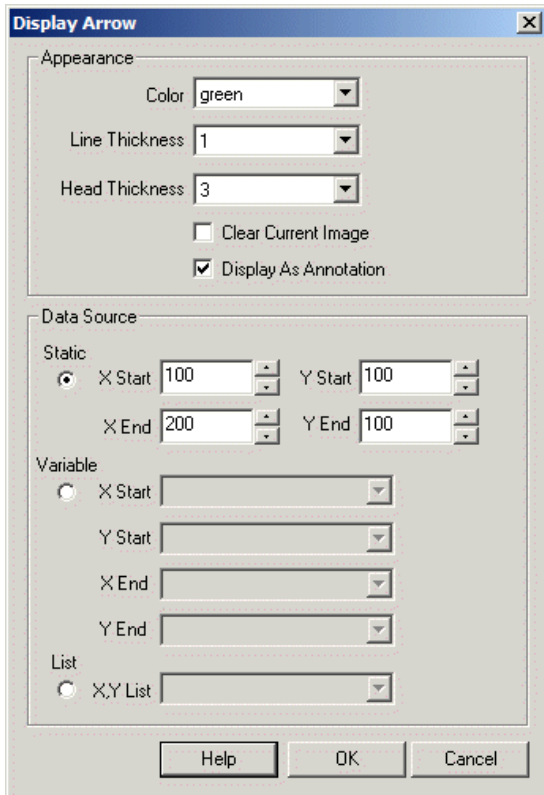
See Also

[Translate](#)
[Scale](#)

Display Arrow

The Display Arrow module provides a way to draw Arrows based on coordinates.

Interface



Instructions

1. Color - Select the appropriate Arrow color.
2. Line Thickness - Select how thick the Arrow line should be.
3. Head Thickness - Select how thick the Arrow point should be.
4. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeArrow will see the graphic as if it were part of the image and process it accordingly.
5. Data Source - Select where the Arrow coordinates should be taken from.
 - Static - specify the integer coordinates for the Arrow
 - Variable - specify the 4 variables that contain the start and stop point of the Arrow
 - List - specify the variable that contains a list of coordinates that specify Arrows. The format of the variable should be x,y,xx,yy for each Arrow to be drawn.
5. Clear current image - Select to clear the current image and draw the graphics on a black image.

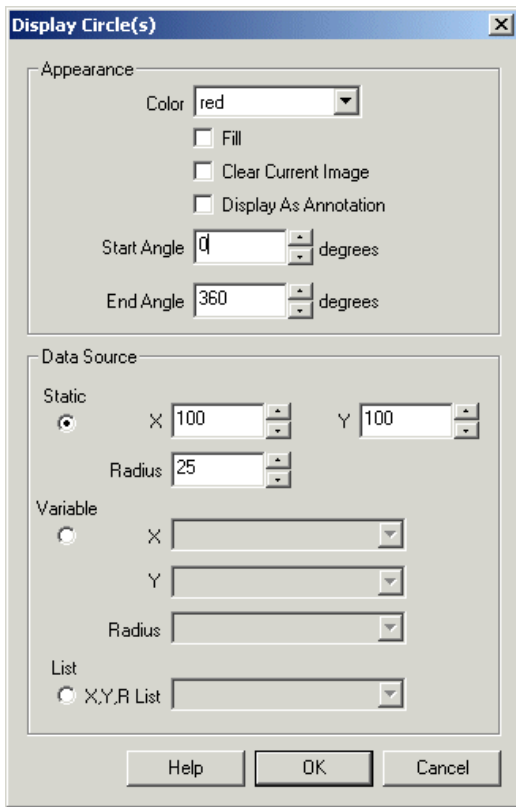
See Also

- [Display Line](#)
- [Display Point](#)
- [Display Image](#)

Display Circle

The Display Circle module provides a way to draw a Circle based on a single point plus radius. The Circle points can be specified as static numbers or accessed from variables in two different formats. Note that the Circle is drawn into the current image and will affect subsequent processing.

Interface

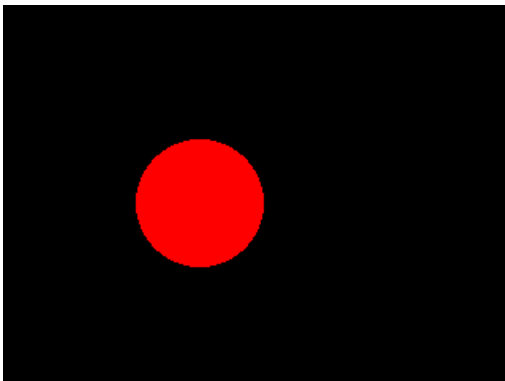


Instructions

1. Color - Select the appropriate circle color.
2. Fill - Select if the circle should be filled or empty.
3. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Angle - To draw a partial circle or arc you can specify the start and end angles of the circle to be drawn. These values are in degrees. Note that you can use variable expressions to automatically change this value. Angle are only respected on circle drawing and not when filling the circle.
5. Data Source - Select where the circle coordinates and radius should be taken from
 - Static - specify the integer coordinate and radius for the circle
 - Variable - specify the 3 variables that contain the coordinate and radius for the circle
 - List - specify the variable that contains a list of coordinates and radi that specify Circles. The format of the variable should be $x_1,y_1,r_1,x_2,y_2,r_2,x_3,y_3,r_3$ for each Circle to be drawn.
6. Clear current image - Select to clear the current image and draw the graphics on a black image.

Example

Display Circle

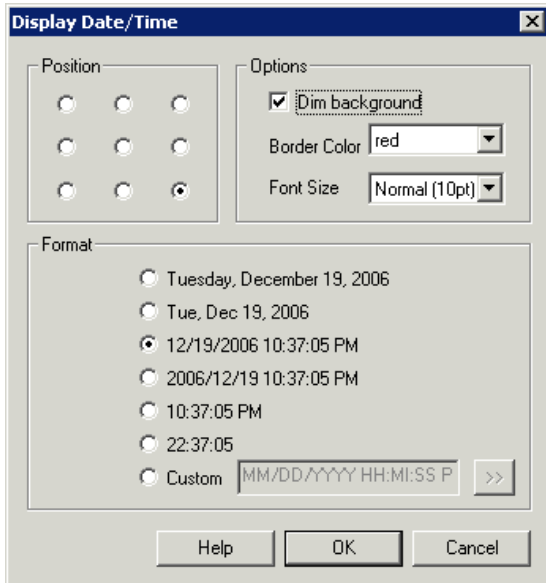


See Also

Display Date/Time

The "Display Date/Time" module provides a way to add a time/date stamp onto the video stream prior to saving or other transmission.

Interface



Instructions

1. Position - Select the position where the time/date stamp should appear. The circle radio buttons specify the 9 quadrants of the image. Clicking on a radio button will move the time/date stamp to that area.
2. Dim background - Specify if you would like the background around the time/date stamp to be slightly dimmer. Click on the "Dim background" button to enable this.
3. Border Color - Select the color of the border that should surround the time/date stamp. If you do not want a border around the time/date stamp select the first blank option.
4. Font Size - Select the appropriate font size to use.
5. Display as Annotation - Select if you want the graphic to be drawn after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
6. Format - Select the time/date stamp format that you would like to use. If you do not find an appropriate format select the Custom radio button and type in the appropriate format string. You can use the button to the right of the custom text field to select appropriate date/time strings. Note that any text not recognized as a time/date placeholder will be printed as text.

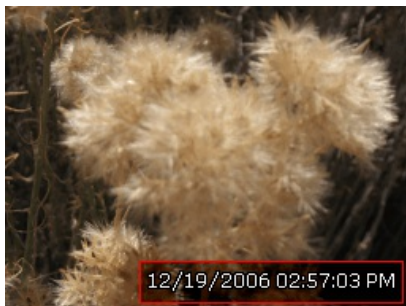
The following outlines the time/date stamp placeholders that can be used:

- SPACE
- DASH
- M - month number
- MM - month number - zero padded
- MON - 3 letter monthN
- Mon - 3 letter monthN2
- mon - 3 letter monthN3
- MONTH - full monthNTH
- Month - full monthNTH2
- month - full monthNTH3
- DDD - days in year
- DD - days in month - zero padded
- D - days in month
- LD - last day in month
- DW - day in week

DY - 3 letter weekday
Dy - 3 letter weekday
dy - 3 letter weekday
DAY - full weekday
Day - full weekday
day - full weekday
YYYY - 4 letter year
YY - 2 letter year
WW - week in year
W - week in month
HH - hour
HH24 - 24 hour
MI - minutes
SS - seconds
MS - milli-seconds
PM - AM / PM indicator
pm - am / pm indicator

Example

Display Date/Time



See Also

[Format Date Time](#)

[Display Variables](#)

Display Image

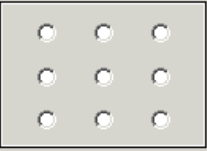
The Display Image module provides a way to display images within the current image similar to a picture in picture (PIP) functionality seen in many television sets today. The module is useful as many other modules will provide images as named images that can be displayed within the current image for reference. For example, the Image Matching module provides an "image_match" image which can be used in this module for display.

Interface



Mask: robot_mask.png

Position



X: [x] Y: [y]

Rotation: [rotation] degrees

Scale: 100 percent

Options

Border Color: [dropdown]

Border Size: [dropdown]

Display As Annotation

Help OK Cancel

Instructions

1. Image - Select which image to display.

Source - the source image that was initially loaded into RoboRealm

Current - the currently processed image within RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created. If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

2. Mask - If needed select a Mask that will indicate where the image pixels are valid and will overwrite the currently displayed image. The mask allows you to introduce an alpha or transparency to the image that you are displaying. Wherever the Mask is non-zero the pixels in the Image will be copied onto the currently displayed image.

3. Position - Select one of the nine possible positions to place the image. Or select the radio button next to the X,Y coordinates. Once this radio button is selected you can enter in the X,Y values for image placement or use the [variable] specification to use the value within a RoboRealm variable for placement.

4. Rotation - Specify the Z rotation for the image to be rotated prior to display. Note that this field is specified in degrees from 0 to 360. To use a variable in this field use the [variable] notation. This will cause the module to query the variable's value and use that for the rotation amount.

5. Scale - Select the PIP size. Note that this is relative to the original PIP image size and NOT the currently displayed image. Note that selecting 100% replaces the current image with the PIP selected image.

6. Border Color - Select the border color what will outline the PIP

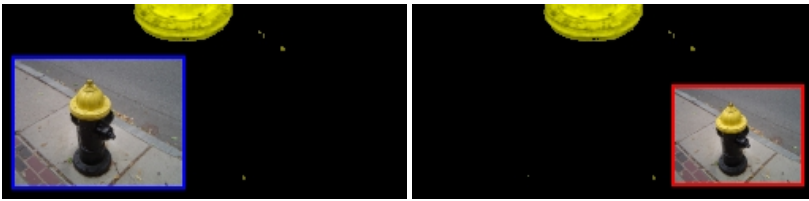
7. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

Example

Example#1 - RGBFilter Yellow

Example#2 - Smaller PIP on right side





See Also

[Display Variables](#)

Display Line

The Display Line module provides a way to draw lines based on line coordinates.

Interface

Display Line(s)

Appearance

Color: red

Thickness: 5

Data Source

X Start: 10 X End: 100
Y Start: 150 Y End: 150

X Start: x
Y Start: y
X End: xx
Y End: yy

X,Y List: list

Help OK Cancel

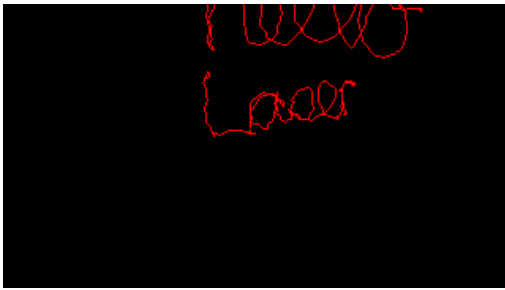
Instructions


1. Color - Select the appropriate line color.
2. Thickness - Select how thick the line should be.
3. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Data Source - Select where the line coordinates should be taken from.
 - Static - specify the integer coordinates for the line
 - Variable - specify the 4 variables that contain the start and stop point of the line
 - List - specify the variable that contains a list of coordinates that specify lines. The format of the variable should be x,y,xx,yy for each line to be drawn.
5. Clear current image - Select to clear the current image and draw the graphics on a black image.

Example

Display Line





 [Click here](#) to load the configuration used to draw the above graphic using laser light! Be sure that the room is somewhat dark otherwise you will get sudden unexpected lines drawn in the screen. To reset and start a new drawing double click on the last module (MATH), select the first image as Current (the image will clear) and then set it back to Last. You can then begin drawing again.

See Also

[Display Arrow](#)

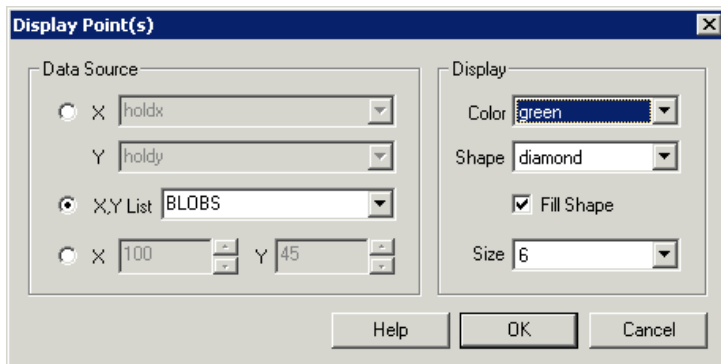
[Display Point](#)

[Display Image](#)

Display Point

The Display Point module provides a way to illustrate the location of certain points. Using a variety of ways the Display Point module can draw a symbol at a certain location.

Interface



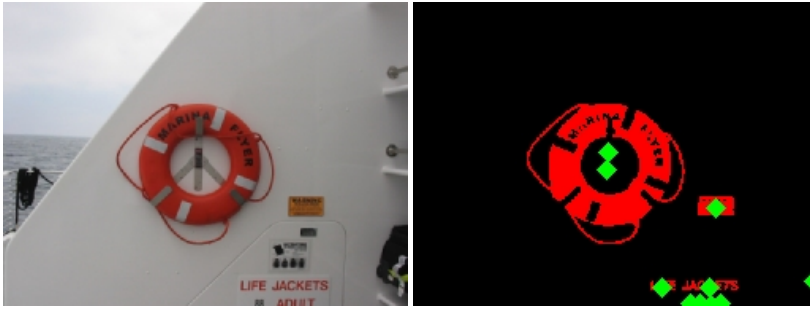
Instructions

1. Data Source - Select the appropriate variable that contains the coordinates of the point that you want to display. You can also specify the coordinates by typing them into the provided X and Y text boxes.
2. Appearance - Select the color, shape, and size of the symbol to draw at the specified location.
3. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Clear current image - Select to clear the current image and draw the graphics on a black image.
5. Data Source - Select where the point coordinates should be taken from.
 - Static - specify the integer coordinates for the point
 - Variable - specify the 2 variables that contain the x and y coordinates of the point
 - List - specify the variable that contains a list of coordinates that specify points. The format of the variable should be repeating x,y for each point to be drawn.
6. Shape, Color, Size - Specify the shape, color and size of the graphic that is used to indicate the point location.

Example

Source

Display Point (Blob COG labeling)



Display Rectangle

The Display Rectangle module provides a way to draw a four point convex rectangle/polygon. The rectangle points can be specified as static numbers or accessed from variables in two different formats. Note that the rectangle is drawn into the current image and will affect subsequent processing.

Interface

Display Rectangle(s)

Appearance

Color: orange

Fill

Clear Current Image

Data Source

Static

X1: 0 Y1: 100

X2: 50 Y2: 55

X3: 8 Y3: 15

X4: 10 Y4: 55

Variable

X1: [] Y1: []

X2: [] Y2: []

X3: [] Y3: []

X4: [] Y4: []

List

X,Y List []

Help OK Cancel

Instructions

1. Color - Select the appropriate rectangle color
2. Fill - Select if the rectangle should be filled or empty
3. Display as Annotation - Select if you want the graphic to be drawn after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Data Source - Select where the rectangle coordinates should be taken from.

Static - specify the integer coordinates for the rectangle

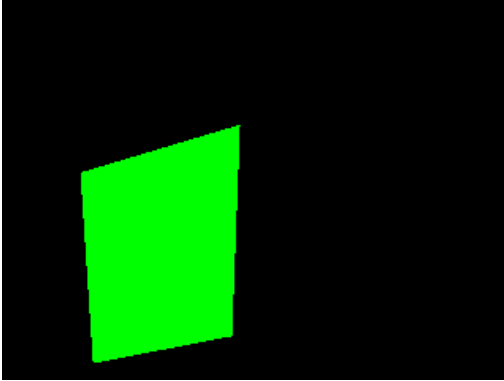
Variable - specify the 8 variables that contain the vertices the rectangle

List - specify the variable that contains a list of coordinates that specify rectangles. The format of the variable should be x1,y1,x2,y2,x3,y3,x4,y4 for each rectangle to be drawn.

5. Clear current image - Select to clear the current image and draw the graphics on a black image.

Example

Display Rectangle

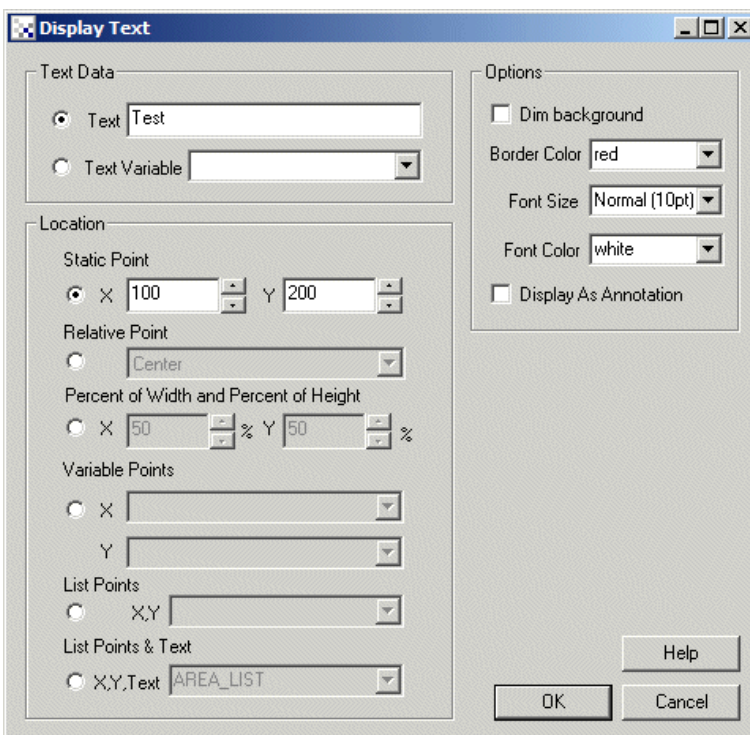


See Also

[Display Triangle](#)

Display Text

Interface



Instructions

1. Text - Specify the actual text to be displayed in the image.
2. Text Variable - Specify the text to display by selecting the variable that contains the text to display.
2. Data Source - Select the point location source

Static - specify the integer coordinates for the text

Relative Point - Select the relative location of the text. The text will then scale with the image to keep its position relative to the center or corner locations.

Percent of Width and Height - Similar to relative location but based on a percent of the current width and height. As the image size changes the location will change in accordance with the specified percentages.

Variable Points - specify the 2 variables that contain the x, y location for the text to be drawn

List Points - specify the variable that contains a list of coordinates where the text should be draw. Note that this list should be a multiple of 2.

List Points and Text - specify the variable that contains a list of coordinates and text string that will be used to display the text strings. Note that this list should be a multiple of 3.

3. Dim background - Select if you would like to "dim" the background of the text area. This reduces the background of the text intensity to make it easier to read the white text.

4. Border Color - Select the color of the border box that surrounds the text.

5. Font Size - Select the font size to use for displaying the variables.

6. Font Color - Select the font color to use for displaying the variables.

7. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

7. When done press ok to save the current configuration. Note that any changes are immediately visible in the image preview in the main dialog.

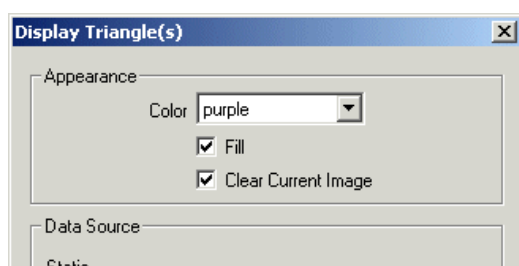
See Also

[Display Variables](#)

Display Triangle

The Display Triangle module provides a way to draw a triangle based on three coordinates. The triangle points can be specified as static numbers or accessed from variables in two different formats. Note that the triangle is drawn into the current image and will affect subsequent processing.

Interface

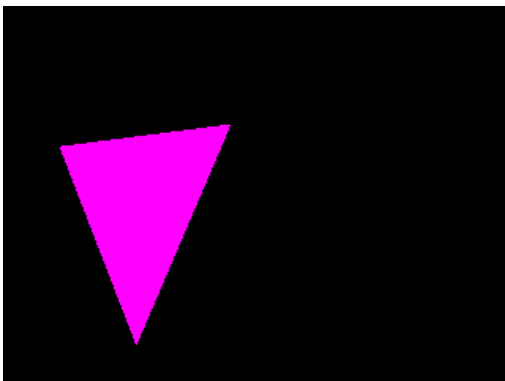


Instructions

1. Color - Select the appropriate triangle color
2. Fill - Select if the triangle should be filled or empty
3. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
4. Data Source - Select where the triangle coordinates should be taken from.
 - Static - specify the integer coordinates for the triangle
 - Variable - specify the 6 variables that contain the vertices the triangle
 - List - specify the variable that contains a list of coordinates that specify triangles. The format of the variable should be x_1,y_1,x_2,y_2,x_3,y_3 for each triangle to be drawn.
5. Clear current image - Select to clear the current image and draw the graphics on a black image.

Example

Display Triangle



See Also

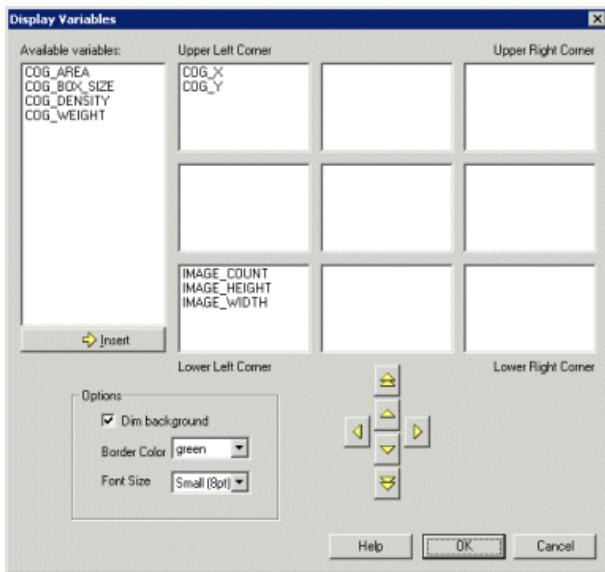
[Display Rectangle](#)

Display Variables

The Display Variables module will draw variables and their values into the current video stream for use in recording and display purposes. Note that the graphics are overlaid after all processing is completed to ensure that added graphics do not become part of the processed image.

When recording a filtered sequence it can be useful for end viewers to include any statistics within the image that you are trying to achieve.

Interface

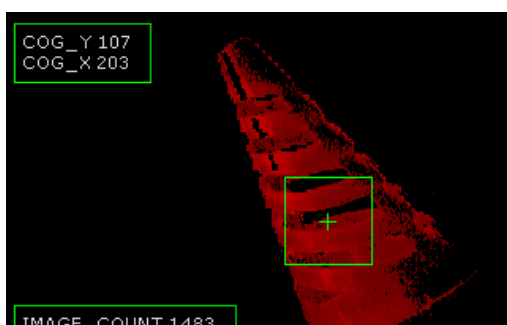


Instructions

1. Insert a variable by double clicking on an available variable or select a variable and click on the insert button.
2. Once a variable is in a slotted box it can be moved to alternate locations using the yellow arrow keys. Note that the double up and down arrows move the variable to the next up or below location box.
3. Dim background - Select if you would like to "dim" the background of the text area. This reduces the background of the text intensity to make it easier to read the white text.
4. Border Color - Select the color of the box that surrounds the text.
5. Font Size - Select the font size to use for displaying the variables.
6. Font Color - Select the font color to use for displaying the variables.
7. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
8. When done press ok to save the current configuration. Note that any changes are immediately visible in the image preview in the main dialog.

Example

Annotated image



See Also

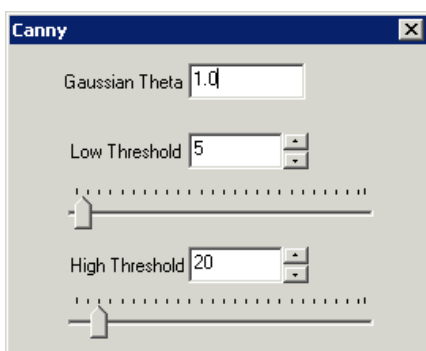
[Write Variables](#)
[Watch Variables](#)
[COG](#)

Canny

The Canny module implements the Canny Edge Detector first specified by John Canny in his paper "A Computational Approach to Edge Detection". The Canny edge detector is regarded as one of the best edge detectors currently in use. The basic algorithm can be outlined as follows:

1. Blur the image slightly. This removes the effects of random black or white pixels (i.e. noise pixels) that can adversely affect the following stages. Blurring is typically accomplished using a [Gaussian Blur](#). However, you should be able to use a box averaging filter like the [Mean](#) filter to accomplish the same effect. The mean filter can be executed faster than a Gaussian blur so for real-time applications the computational advantage can be worthwhile.
2. Compute the x and y derivatives of the image. This is basically an edge detection step. Similar to the [Sobel](#) edge detector the x and y derivatives are simply calculated by subtracting the values of the two surrounding neighboring pixels depending on which axis is being computed (left and right for x derivative and top and bottom for y derivative).
3. From the x and y derivatives you can compute the edge magnitude which basically signifies the strength of the edge detected at the current point.
4. If you were to look at the edge image at this point it would look very similar to the Sobel edge filter. Namely, the edges would be thick in many areas of the resulting edge image. To 'thin' these edges a nonmaximal filter needs to be applied that only preserves edges that are the top or ridge of a detected edge. The nonmaximal filter is applied by first analyzing what the slope of the edge is (you can determine this by the sign and magnitude of the x and y derivatives calculated in the previous steps) and use that slope to determine what direction the current edge is heading. You can then determine the two edge magnitudes perpendicular to the current heading to determine if the current pixel is on an edge ridge or not. If the current pixel is on a ridge its magnitude will be greater than either of the two magnitudes of the perpendicular pixels (think of walking on a mountain ridge with the ground decreasing on both sides of your heading). Using this ridge detection and subsequent elimination will result in very thin lined edges.
5. Once the edges have been thinned the last step is to threshold the detected edges. Edge magnitudes above the upper threshold are preserved. Edges below the upper threshold but above the lower threshold are preserved ONLY if they connect (i.e. 8 neighborhood touching) to edges that are above the upper threshold. This process is known as hysteresis and allows edges to grow larger than they would by using a single threshold without introducing more noise into the resulting edge image.

Interface



Help

OK

Cancel

Instructions

1. Theta - Select the Gaussian blur standard deviation amount. Also known as sigma. Typical values range from 0.60 to 2.40. The higher the value the greater the blurring of the image. Note that the window size of the filter is determined automatically.
2. Low Threshold - Select the low threshold above which possible edges can exist. Edges below the low threshold are removed from the image.
3. High Threshold - Select the high threshold above which edges are preserved in the image. Edges between the high and low threshold are ONLY preserved if they are connected to an edge that is above the high threshold. See hysteresis above.
4. Grayscale - To increase speed, you can select the grayscale option which will only examine the green channel to produce the line results.

Example

Source



Canny Edge Detection



See Also

[Sobel](#)

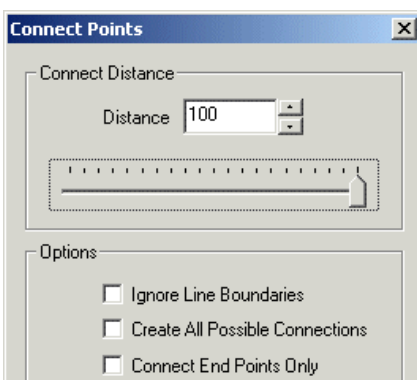
[Gaussian](#)

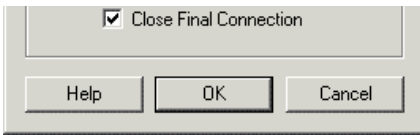
[Difference of Gaussian](#)

Connect Points

The Connect Points module will connect nearby points to create connected edges. This module is useful after thresholding where edge points can become disconnected. Unlike the hysteresis technique used in the [Canny module](#), points are connected based on the distance from each other.

Interface

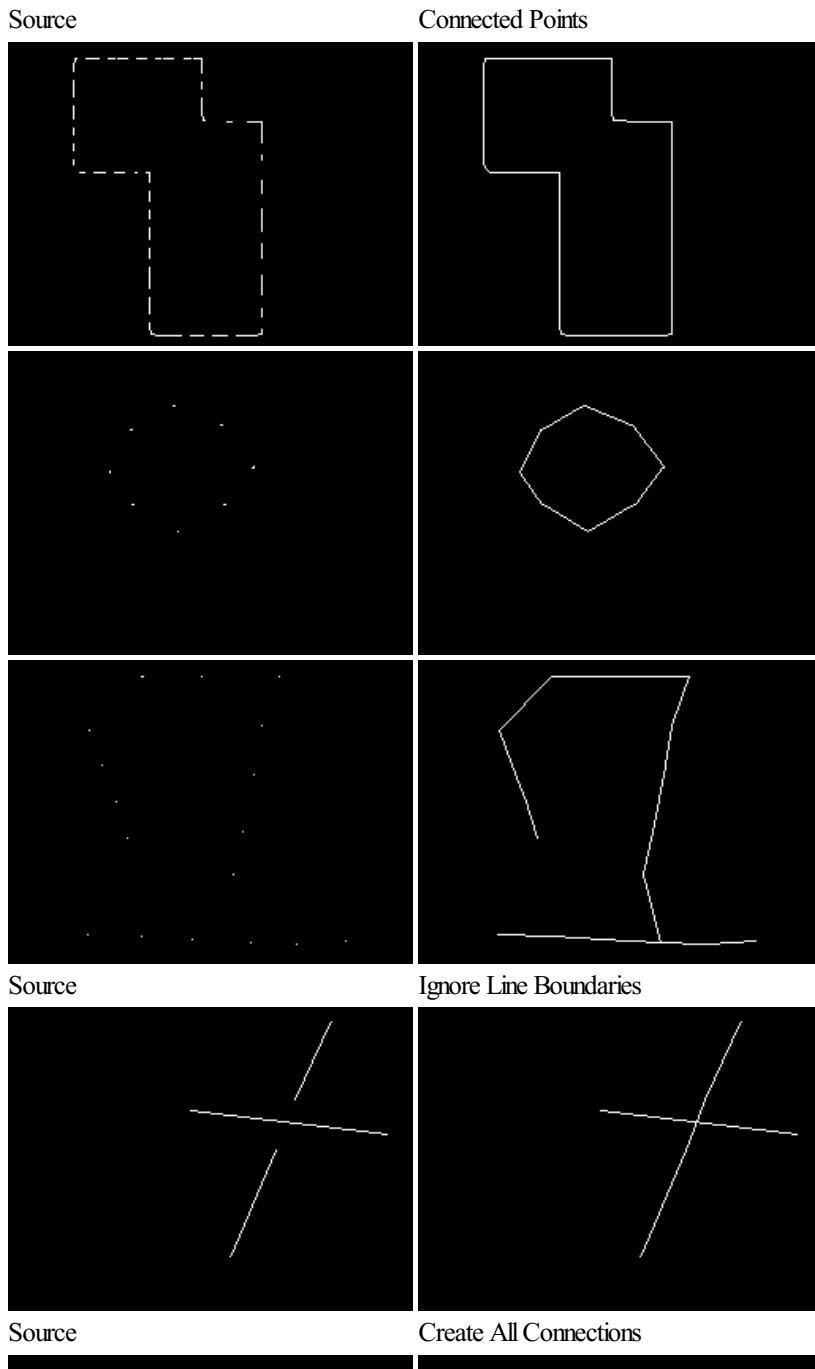


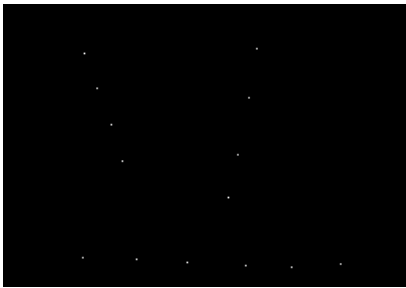


Instructions

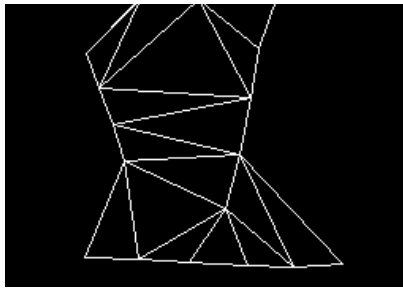
1. Distance - Select the maximum distance that points will be connected over.
2. Ignore Line Boundaries - When connecting points current lines are taken into account such that no new line will connect over an existing line. Select this checkbox to ignore this limitation and connect end points over existing lines.
3. Create All Possible Connections - When selected all points are connect to all other points. This is desired when a mesh like connection structure is desired.
4. Connect End Points Only - Only connect the line end points with other line end points. Thus if you have a single point next to a line it will not connect to the middle of the line by instead one of the line end points (assuming the distance specified above allows for it).
5. Close Final Connection - When serveral points are connection to form a new line the line will not close on itself as in the case of a circle unless this checkbox is selected.

Example

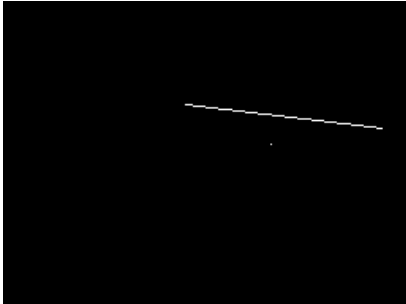




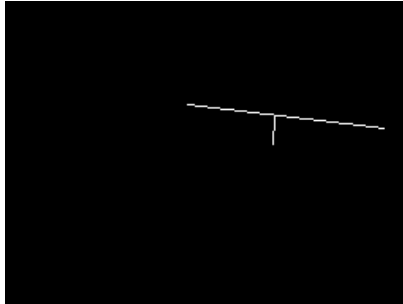
Source



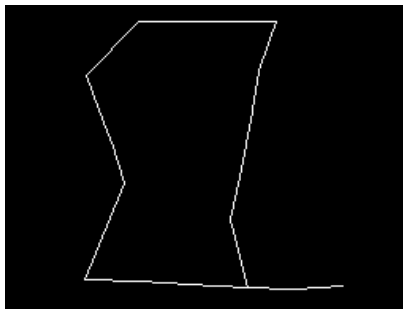
Connect Any Point (not just end points)



Source



Close Connection



See Also

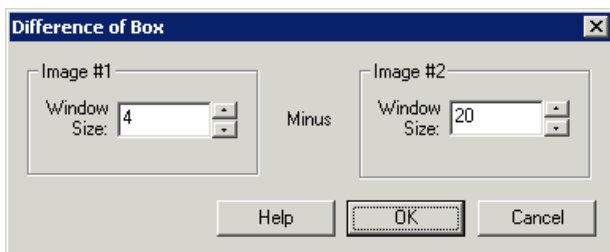
[Dilate](#)

Difference of Box (DOB)

The Difference of Box module is a filter that identifies edges. The DOB filter is similar to the [LOG](#) and [DOG](#) filters in that it is a two stage edge detection process.

The DOB performs edge detection by performing a mean blur on an image at a specified window size. The resulting image is a blurred version of the source image. The module then performs another blur with a smaller window size that blurs the image less than previously. The final image is then calculated by replacing each pixel with the difference between the two blurred images and detecting when the values cross zero, i.e. negative becomes positive and vice versa. The resulting zero crossings will be focused at edges or areas of pixels that have some variation in their surrounding neighborhood.

Interface



Instructions

1. Specify the window size of the first blur to be performed. The window size is how large a mean filter is applied to the image.
2. Specify the window size of the second blur to be performed. Note that the window size should typically be larger than the first window size in

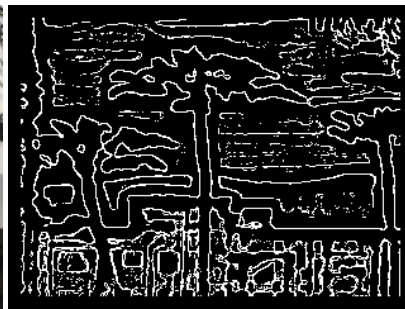
order to correctly detect edges.

Example

Source



DOB



See Also

[Difference of Gaussian](#)

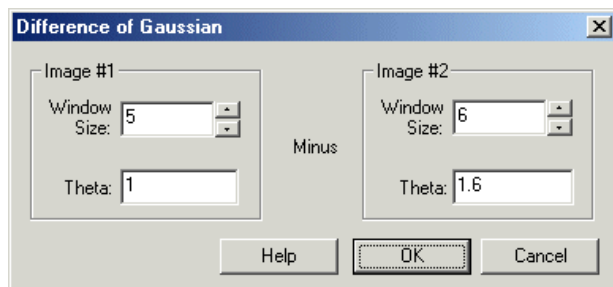
[LOG](#)

[Mean Filter](#)

Difference of Gaussian (DOG)

The Difference of Gaussian module is a filter that identifies edges. The DOG filter is similar to the [LOG](#) and [DOB](#) filters in that it is a two stage edge detection process.

The DOG performs edge detection by performing a Gaussian blur on an image at a specified theta (also known as sigma or standard deviation). The resulting image is a blurred version of the source image. The module then performs another blur with a sharper theta that blurs the image less than previously. The final image is then calculated by replacing each pixel with the difference between the two blurred images and detecting when the values cross zero, i.e. negative becomes positive and vice versa. The resulting zero crossings will be focused at edges or areas of pixels that have some variation in their surrounding neighborhood.



1. Specify the window size and theta of the first blur to be performed. The window size is how large a Gaussian filter is applied to the image. If the filter is too small the Gaussian filter starts to approximate a [box blur](#) filter. If the filter is too large then values at the ends become zero and extra work is performed which slows down processing.
2. Specify the window size and theta of the second blur to be performed. Note that the theta should typically be larger than the first theta in order to correctly detect edges.

See Also

[Gaussian Filter](#)

[Difference of Boxes](#)

[LOG](#)

For additional information on Difference of Gaussian (DOG) see

[Molecular Expressions Microscopy Primer - Difference of Gaussians Edge Enhancement Algorithm](#)

[Sussex Computer Vision: TEACH Vision3](#)

[Harvey Mudd College: Computer Image Processing and Analysis \(E161\)](#)

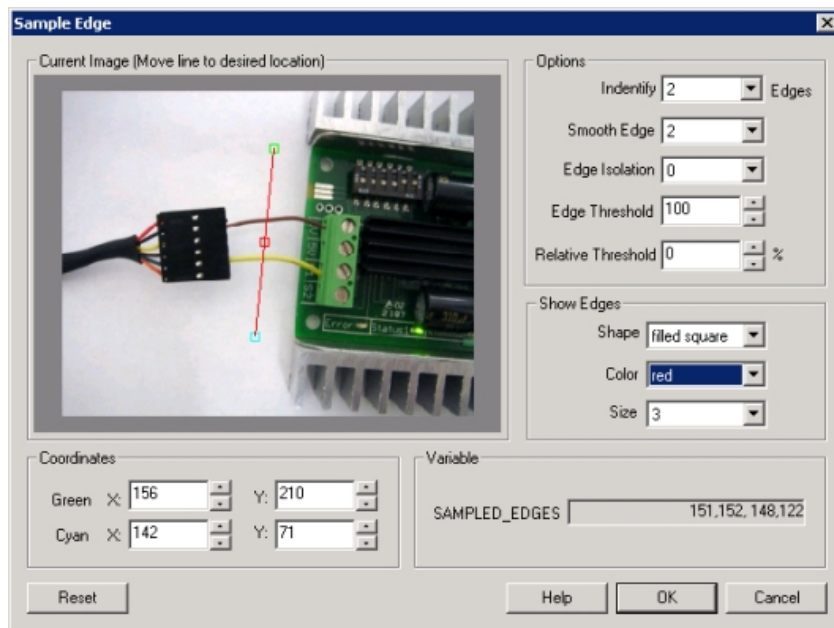
Edge Probe

The Edge Probe module provides a way to detect edges along a specified path. The path is a single line that you can manually place on the image or

control the position using variables. The edges detected are then saved into a variable for later usage. Note that the detected edges are with respect to the line angle.

This module is useful for detecting the placement of a particular object within an image or to detect that a certain number of edges need to be present in the form of a component quality assurance.

Interface



Instructions

1. Current Image - move the line to the location along which you want to probe for edges. You can use the red square to move the entire line, the endpoint squares to move each endpoint individually or change the coordinates using the text boxes below in the Coordinates area.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the probe of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

3. Identify - select how many edges you want to detect. Note that if you select many edges to be identified you are not guaranteed to get that many edges depending on the thresholding information below. The "Identify" number specifies a maximal number of edges to detect. Note that detected edges are ordered as they are encountered. The Arrow appearing in the modules GUI indicates which direction the edge probe proceeds. For multiple probes, the first probe line is the line connecting Cyan to Green with the Purple to Blue line being last.

4. Precision - the precision specifies how each pixel should be divided in order to determine the exact place that the edge occurs. If you have a blurred image you may need to reduce the precision to negative numbers. Negative numbers mean that an edge is comprised of several pixels (i.e. white to black transition occurs slowly over several pixels). If positive the precision will determine where the edge most likely exists between two successive pixels by interpolating values between the two pixel intensities.

5. Smooth Edge - to reduce noisy edges select the amount of smoothing that should be applied to the edge prior to edge detection. This prevents a single sharp noise pixel from being detected as an edge.

6. Edge Isolation - to avoid edges from bunching up together select how many pixels each edge should be isolated from the next detected edge.

7. Edge Threshold - to eliminate very weak edges select an appropriate edge threshold (0-255) that will remove edges whose edge intensity is below the threshold. Note that smoothing the edge will also reduce the edge intensity and thus the Edge Threshold will need to be adjusted after the smoothness is specified.

8. Relative Threshold - to only select those edges that are significant you can specify the relative threshold (0-100) that will remove successive edges that are the relative threshold percent less than the previous edge. For example, consider the highest 3 edge values as 130, 120 and 40. If the relative threshold is 50% then the last edge 40 would be eliminated since 40 is less than 50% of 120. As 120 is 92% of 130 it is not eliminated (unless the threshold were set at 95%).

9. Number of Probes - at times you may want to sample edges along a path using more than one edge probe. When you increase the number of edge probes beyond 1 the probe selector area will define a box where as you increase to more probes additional edge probes will be created within that rectangle. By increasing the probe count you can quickly probe a larger area to sample more potential failure points without having to

main and rectangle. By increasing the probe count you can query probe a larger area to sample more potential image points without having to add more modules. Note that if you need multiple probes in different seconds of the image you can add more probe modules and specify a query area for each.

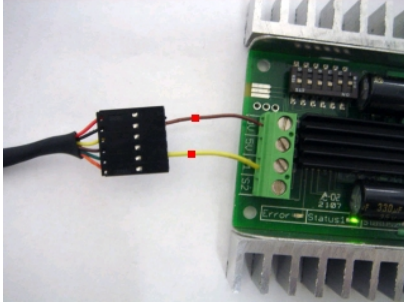
10. Show Edges - to visually see which edges are being detected select the appropriate Shape, Color and Size of the marker that will indicate where in the image the edges have been detected. Note that this appears in the main RoboRealm GUI window.

Note

To control the line location using variables specify the variable as [variable_name] within the coordinate textboxes.

Example

The detected edges marked with red squares using the configuration shown in the above screenshot.



Variables

PROBED_EDGES_COUNT - the number of edges found

PROBED_EDGES - an x,y array of the location of the detected edges.

PROBED_EDGES_DISTANCE - the distance between the first detected edge and the last. This defines the width in pixels between the outer detected edges.

PROBED_EDGES_MAX_DISTANCE - the max distance between the first detected edge and the last for each probe when multiple probes are used.

PROBED_EDGES_MIN_DISTANCE - the min distance between the first detected edge and the last for each probe when multiple probes are used.

PROBED_EDGES_MAX_INTENSITY - the max intensity of a pixel encountered along a probe.

PROBED_EDGES_MIN_INTENSITY - the min intensity of a pixel encountered along a probe.

See Also

[Origin Probe](#)
[Line Probe](#)
[Circle Probe](#)
[Sample Line](#)

Frei & Chen

The Frei and Chen module is used to detect edges with an image. The Frei and Chen edge detection technique is less sensitive to noise than other edge operators.

The final pixel value is the ratio between the multiplication of four kernels with the surrounding pixel neighborhood and the square sum of all pixels. The following are the kernels used to multiply the pixel neighborhood:

+2	+3	+4	+2	0	-2	+3	0	-3	+2	0	-2
0	0	0	0	-2	+3	+2	0	-2	-3	+2	0
-2	-3	-2	+3	-2	0	-2	0	+2	0	+2	-3

Example

Source Image

Frei & Chen



See Also

[Sobel Filter](#)

[Prewitt Filter](#)

Kirsch

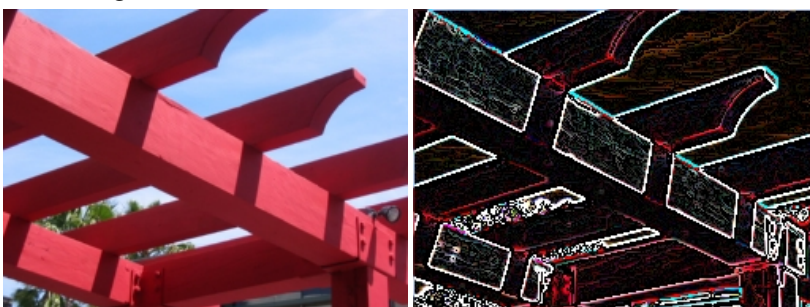
The Kirsch Edge module detects edges using eight compass filters. All eight filters are applied to the image with the maximum being retained for the final image. The eight filters are a rotation of a basic compass convolution filter. The filters are of the form:

+3	+3	+3	+3	+3	+3	-5	+3	+3	-5	-5	+3
+3	0	+3	-5	0	+3	-5	0	+3	-5	0	+3
-5	-5	-5	-5	-5	+3	-5	+3	+3	+3	+3	+3
-5	-5	-5	+3	-5	-5	+3	+3	-5	+3	+3	+3
+3	0	+3	+3	0	-5	+3	0	-5	+3	0	-5
+3	+3	+3	+3	+3	+3	+3	+3	-5	+3	-5	-5

Example

Source Image

Kirsch



See Also

[Sobel](#)
[Frei & Chen](#)
[Prewitt Filter](#)
[Convolution Filter](#)

Laplacian of Gaussian (LOG)

The LOG module performs a Laplacian of Gaussian filter. This filter first applies a [Gaussian blur](#), then applies the Laplacian filter (see [convolution](#)) and finally checks for zero crossings (i.e. when the resulting value goes from negative to positive or vice versa). The end result of this filter is to highlight edges.

The first stage of the filter uses a Gaussian blur to blur the image in order to make the Laplacian filter less sensitive to noise. If you run the Laplacian filter on a noisy image the result is an edge image with many many small edges that detract from the larger more meaningful edges. Other blur filters could also be used prior to the Laplacian filter but the Gaussian blur is more commonly used for this process.

The Laplacian filter is a convolution filter that is used to detect edges. The following are examples of 3 common Laplacian filters:

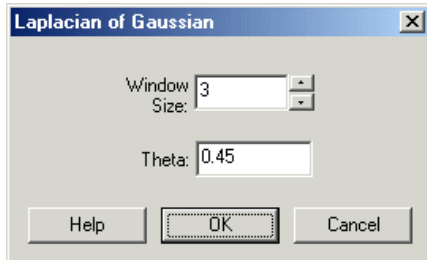
0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

-1	2	-1
2	-4	2
-1	2	-1

In the case of this module the Laplacian and Gaussian are convoluted together to create a single filter that is applied to the image.

Once the filter has been convoluted with the image the resulting values are negative and positive numbers. The final step detects where these numbers switch signs and marks that point in the image. The end result is an image that indicates edges.



1. Specify the window size of the filter to use. The larger the filter the slower the processing but the less resistant to noise in the images.
2. Specify the Gaussian Theta to use when creating the single LOG filter. Higher thetas result in more rounded or blurring of the image.

For additional information on Laplacian of Gaussian (LOG) see

[University of Edinburgh - HyperMedia Image Processing Reference2](#)

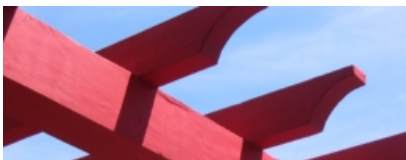
[Oxford University - Stephen M. Smith](#)

Max Edge

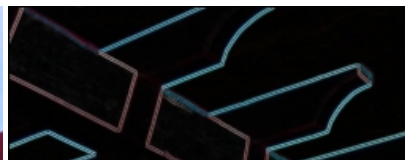
The Max Edge module will determine the maximum edge between each pixel in all color spaces. This is different from other edge detection techniques in that even if a pixel has a single strong edge with its neighbors it will still evoke a strong response as opposed to being softened by its other neighbors. This filter tends to magnify noise but is useful in detecting very soft edges that don't have much neighborhood support.

Example

Source



Max Edge





See Also

[Sobel](#)

[Canny](#)

Outline Filter

The Outline filter is used to outline edges present within the image. Similar to an outline convolution filter this filter reacts to changes in intensity or color across edges. Sharper edges will result in higher intensity values. The outline function uses a convolution filter size based on the user specified window size. Results are then normalized and displayed as grayscale values.

Prewitt Edge

The Prewitt Edge filter is used to detect edges based on applying a horizontal and vertical filter in sequence. Both filters are applied to the image and summed to form the final result. The two filters are basic convolution filters of the form:

Horizontal Filter Vertical Filter

1	1	1	-1	0	1
0	0	0	-1	0	1
-1	-1	-1	-1	0	1

For example, if a 3x3 window is used as such

p1	p2	p3
p4	p5	p6
p7	p8	p9

where the filter is centered on p5 with p4 being pixel[x-1][y] and p6 being pixel[x+1][y], etc. then the formula to calculate the resulting new p5 pixel is

$$\text{pixel} = (p1+p2+p3-p7-p8-p9)+(p3+p6+p9-p1-p4-p7)$$

which is then clamped to the 0-255 range.

Note that the actual formula uses the horizontal and vertical components into the final form

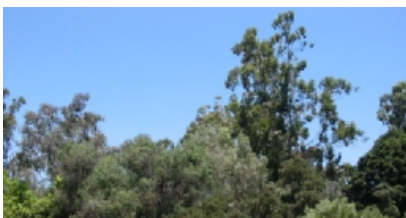
$$\text{pixel} = \text{SQRT}((X*X)+(Y*Y))$$

where $X = (p1+p2+p3-p7-p8-p9)$ and $Y = (p3+p6+p9-p1-p4-p7)$

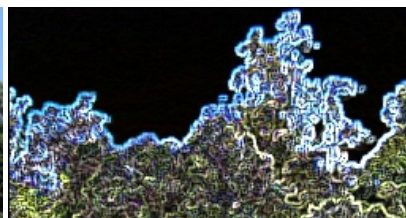
but for performance reasons we approximate the result and leave the final formula out.

Example

Source



Prewitt





See Also

[Sobel Filter](#)

[Convolution Filter](#)

Roberts Edge

The Roberts Edge filter is used to detect edges based on applying a horizontal and vertical filter in sequence. Both filters are applied to the image and summed to form the final result. The two filters are basic convolution filters of the form:

Horizontal Filter Vertical Filter

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

For example, if a 2x2 window is used as such

$$\begin{bmatrix} p1 & p2 \\ p3 & p4 \end{bmatrix}$$

where the filter is centered on $p1$ with $p2$ being $\text{pixel}[x+1][y]$ and $p3$ being $\text{pixel}[x][y+1]$, etc. then the formula to calculate the resulting new $p1$ pixel is

$$\text{pixel} = \text{abs}(p1-p4) + \text{abs}(p2-p3)$$

which is then clamped to the 0-255 range.

Note that the actual formula uses the horizontal and vertical components into the final form

$$\text{pixel} = \text{SQRT}((X*X) + (Y*Y))$$

where $X = \text{abs}(p1-p4)$ and $Y = \text{abs}(p2-p3)$

but for performance reasons we approximate the result and leave the final formula out.

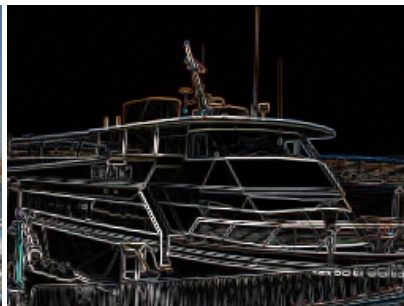
The Roberts Edge detector is fast since the filter is small but it is also subject to interference by noise. If edges are not very sharp the filter will tend not to detect the edge. See Prewitt or Sobel filters instead which are larger and less sensitive to noise.

Example

Source



Roberts Edge



See Also

[Prewitt Filter](#)

[Sobel Filter](#)

[Convolution Filter](#)

Robinson

The Robinson Edge filter is used to detect edges using a horizontal and vertical kernel filter. Both filters are applied to the image and summed to form the final result. The two filters are basic convolution filters of the form:

+2	+3	+2	+2	0	-2	0	-2	+3	+3	-2	0
0	0	0	+3	0	-3	+2	0	-2	-2	0	+2
-2	-3	-2	+2	0	-2	-3	+2	0	0	+2	-3

Example

Source Image



Robinson



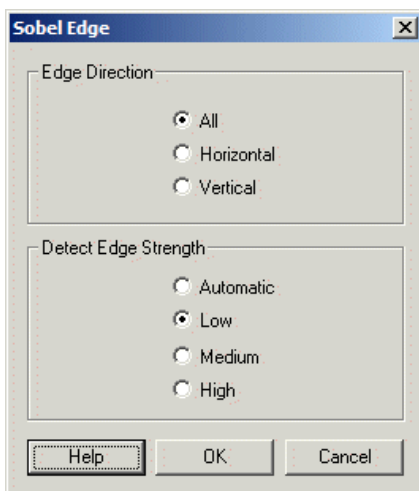
See Also

- [Frei & Chen](#)
- [Prewitt Filter](#)
- [Convolution Filter](#)

Sobel Edge

The Sobel Edge filter is used to detect edges based on applying a horizontal and vertical filter in sequence. Both filters are applied to the image and summed to form the final result. The two filters are basic convolution filters of the form:

Interface



Horizontal Filter Vertical Filter

1	2	1	-1	0	1
0	0	0	-2	0	2
-1	-2	-1	-1	0	1

For example, if a 3x3 window is used as such

p1	p2	p3
p4	p5	p6
p7	p8	p9

where the filter is centered on p5 with p4 being pixel[x-1][y] and p6 being pixel[x+1][y], etc. then the formula to calculate the resulting new p5 pixel is

$$\text{new_pixel_intensity} = (p1+(p2+p2)+p3-p7-(p8+p8)-p9)+(p3+(p6+p6)+p9-p1-(p4+p4)-p7)$$

which is then clamped to the 0-255 range.

Note that the actual formula uses the horizontal and vertical components into the final form

$$\text{new_pixel_intensity} = \text{SQRT}((X*X)+(Y*Y))$$

where $X = (p1+(p2+p2)+p3-p7-(p8+p8)-p9)$ and $Y = (p3+(p6+p6)+p9-p1-(p4+p4)-p7)$

but for performance reasons we approximate the result and leave the final SQRT formula out.

Note that the new pixel values need to be placed into a new image buffer so as not to corrupt the pixel values of the current image.

For color images the formula is applied to all three color channels separately.

The Edge Strength decides how much the detected edge is reduced before converting the number back to a 8 bit (256) RGB color. Low indicates that the number is not reduced which tends to enhance lower magnitude edges, high will reduce edges such that the intensity is approximate to the image pixels. Automatic will adjust the results such that all magnitudes are scaled to the 256 range.

Example

Source Image



Sobel



See Also

[Frei & Chen](#)

[Prewitt Filter](#)

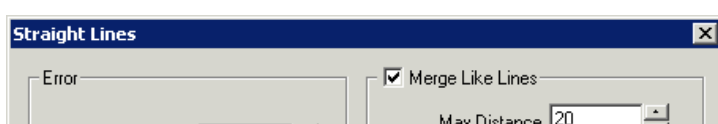
[Convolution Filter](#)

Straight Line

The Straight Line module provides a way to filter out straight lines from an image. This is functionally similar to the [Hough Transform](#) but will operate at a faster speed.

The Straight Line module expects an edge extracted image to work correctly. Thus you must use Canny or other edge finding modules before running the Line Corner module.

Interface



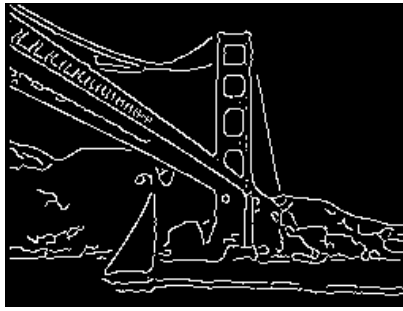
Instructions

1. Be sure to have an edge detected image (using something like [Canny](#)) before using the Line Corner detection module.
2. Allowed Line Error - as the module follows the edge contours the allowed line error will ensure that the creation of a straight line does not deviate from the allowed error. Decreasing this number will create straight lines that better match the contour, larger values will allow the straight line to cover more of the line at the expense of accuracy.
3. Min Max Length - removes lines that are shorter than min length and longer than max length.
4. Angle Filter - removes lines that are not within min and max angle. These are values in degrees. For example, setting min to 85 and max to 95 would remove all lines that are not vertical.
5. Merge Like Lines - concatenates lines that are similar in slope and whose endpoints are relatively close.
6. Max Distance - the maximum distance of line endpoints that will be merged. The smaller this number the closer the line endpoints need to be in order to be merged into a single line. The higher this number the more distant the lines can be.
7. Merge Error - when comparing the slope of the two lines the merge error dictates how unlike the two slopes can differ to still be merged. Lowering this number will cause lines to have very similar slopes in order to be merged. Increasing this number will cause lines that are less like in slope to be merged. Note that the final merged line will have a combination of both line's slopes.
8. Extend Lines to Closest Intersection - As lines can be broken short of their expected end the "Extend Lines" will extend the line to the closest intersection to help complete the lines path.
9. Intersect Distance - the maximum distance any line intersection that will be considered. Increasing this number will extend lines further to local intersections. Decreasing this number will require intersections be closer to the actual end of both lines in order for an intersection to cause line extending.
10. Intersect Angle - when two lines intersect and angle between them is formed. This number will ensure that extended lines are only extended if the nearest intersection forms an angle that is above the number specified here. Increasing this number will cause lines to have a more perpendicular intersection in order to be extended. Decreasing this number will allow more and more parallel lines to be extended to their nearest intersection.
11. Parallel Lines Only - used to eliminate any line that does not have a parallel line nearby. This is handy when a group of parallel lines is to be extracted from an image. Note that parallel lines will be identified in pairs.
12. Overlap - specifies how similar in length the two overlapping parallel lines need to be in order to be considered a parallel line. Increasing this number will require more of the two lines to overlap in space, decreasing the number will allow parallel lines that slightly overlap each other to be considered.
13. Max Distance - the maximum distance the two parallel lines need to be before being considered parallel. Increasing this number will allow parallel lines that are spatial further away from each other to be considered parallel. Decreasing this number will force them to be closer together.
14. Create STRAIGHT_LINE Array - creates a VBScript accessible array that holds the start and end coordinates of any resulting line. The first line start coordinate is STRAIGHT_LINE(0) as the X axis, STRAIGHT_LINE(1) as the Y axis, and STRAIGHT_LINE(2), STRAIGHT_LINE(3) being the endpoint of the line. The second line start point would then be STRAIGHT_LINE(4), STRAIGHT_LINE(5), and so on.

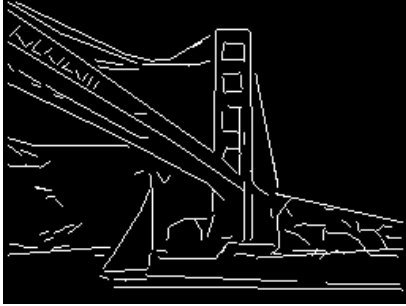
Example



Straight Lines Detected (>5 length)



Straight Lines Detected (>30 length)



Notice that some of the "straight" lines are in fact not perfectly straight in the original image. This is permitted due to the allowed error being high which can be made very lenient in what it considers to be a straight line.

Variables

STRAIGHT_LINE - the xstart, ystart, xend, yend points of all detected lines
in the current image.

See Also

[Hough](#)
[Line Corner](#)

Extending RoboRealm

Overview

RoboRealm Plugins allow you to create custom filters/algorithms that can be added into the RoboRealm image processing pipeline. Currently there are 4 types of ways to extend the main RoboRealm application. The techniques are ordered mainly in terms of increasing complexity but also in increasing capability. If you have questions or problems with these plugins please be sure to [send us a message](#)

RoboRealm Plugins are different from the [API](#) as the plugins are used to add new image processing modules into RoboRealm for use in any application/project. They also guarantee that each frame is processed by the plugin before execution is passed back to the image processing pipeline. The API is used for external applications to query the data within RoboRealm and/or manage the RoboRealm application remotely.

[Download](#) our example plugin programs to get your started on creating your own plugin. The examples include Java, CSharp, Visual Basic, C++ and Python versions.

Types of plugins

VBScript & Python programs offer a basic numerical comparison capability that reacts to variables created in RoboRealm. For example, the mapping of visual coordinates to servo intensities can be accomplished within the script. However, while direct pixel comparisons are possible the speed restrictions imposed based on the interpreted language can become unacceptable for use.

Pipes are a simple way to extend the processing capability of RoboRealm. Using windows Pipes the image data and variables are passed to an external program outside of RoboRealm. The pipe method is simple to use and provides the quickest way to extend RoboRealm.

Sockets are a similar communication technique that allows one to extend the RoboRealm program using external programs. The advantage of the socket communication is in providing a more standardized communication protocol that can be accepted by many different programming platforms.

DLL plugins are the most sophisticated mechanisms to extend RoboRealm but offer the greatest possibility in terms of user interface and processing performance.

Variables and pixels

There are two forms of data that are passed through the RoboRealm processing pipeline. The first and most relevant are the actual image pixels. Pixels are provided in BGR (Red and Blue are switched from the normal RGB) arrays that can be processed as needed by external programs. Pixel values range from 0 to 255 and are represented as 3 unsigned bytes.

Variables are more global image attributes that typically represent results from more detail image calculations. For example, variables can be used to store the average image intensity, image Center of Gravity, min and max pixel values, etc. Variables are passed and received from each plugin to allow modification and creation of variables.

When to use what?

[VBScript](#) / [Python](#) / [CScript](#) - You have a simple need to perform a simple calculation on one or more of the variables created within RoboRealm.

[Pipes](#) - You need to extend RoboRealm by a custom processing routine that alters pixels directly. You are comfortable with the Windows operating system but don't have a need to create GUI interfaces for your plugin.

[Sockets](#) - need to extend RoboRealm by a custom processing routine that alters pixels directly. You wish to utilize another machine to perform your processing to improve performance. Or you have an alternate programming platform (such as Java) that you would like to use to process image pixels and RoboRealm variables.

[Windows DLL](#) - you need to create the fastest possible processing of image pixels that will require some configuration that needs to be stored with the RoboRealm program. You are also very comfortable with the Windows GUI programming and compilers such as VC++ 6.0 and above.

Examples in Languages

C++ - [Pipe](#), [Socket](#), [Windows DLL](#)

CSharp - [Socket](#)

Java - [Socket](#) *(Thanks to Janahan Thevaseelan for the original version)*

Python - [Script](#), [Socket](#)

Visual Basic - [Script](#), [Pipe](#)

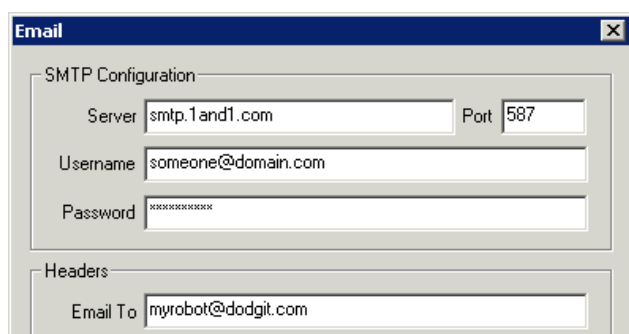
Visual Basic.Net - [Pipe](#)

Email

The Email module provides a way to send images and text to a specified email account. This is useful if you are monitoring a visual scene and you would like to get updated once in a while via email on what that scene looks like. This is an alternative to using the built in [WebServer](#) to monitor a remote scene.

Note that no emails will be sent out until you press the Start button.

Interface



Instructions

1. SMTP Configuration - The SMTP (Email) server is required if RoboRealm is operating within your local network. Basically check your current email program for these settings normally located under "Account Settings" or "SMTP Configuration", etc. The SMTP Server is the mail server that will accept emails from you. There are cases when this information is not needed as RoboRealm will try to deliver the email directly to the recipient if no SMTP Server is specified but this may not always work.

2. Use MAPI Client - if you have an email client configured you can use that client to send email. This is advantageous as RoboRealm will then use the configuration already specified in your email program to send email. This includes secure server connections and username, password information. Note that some email programs may require confirmation in order to send email. If RoboRealm stalls in sending email check your email client program to see if it requires confirmation to send email.

Most errors that occur in setting up a MAPI client are that the expected mail client isn't the default mail client. This can be verified by going into the Control Panel->Default Programs->Set Default Programs.

3. SMTP Port - Note that some SMTP servers require connection on a different port. The default is 25, but 587 is also often used instead.

4. Username/Password - Some SMTP servers require a username/password to login to the server before you are allowed to send email.

5. Email To - Specify who the email should be sent to

6. Subject - Specify the subject of the email

7. From Email - Specify which email address is sending the email (i.e. which address to use in a reply)

8. From Name - Specify the person's name whose email address is in "From Email".

9. Body - Type in any text that should be sent as part of the email. Note that you can use the dropdown list to select the attached image or any RoboRealm variable and then press INSERT in order to add in that text. Note that if you do not specify where to place the image in the email body it will be added as an attachment regardless.


10. Image Attachment - specify which image to send with the email.

11. Maximum Rate - used to limit the number of emails sent to a specified rate so that you do not overload the email server.

12. Start Button - no emails will be sent out until you press the Start button.

Notes

The Email module will keep sending out emails as per the "maximum rate". If you want to only send out emails based on other factors such as movement you will want to combine the module within an [If Statement](#) to only execute it when something moves.

For example,  [Click here](#) to load a configuration that will send an email when something large moves in front of the camera. Note that you will need to configure the email module before it will send an email to you. Double click on the Email text in the pipeline interface to bring up that configuration. Don't forget to press the Start button once you've configured the module.

If you do not know the SMTP configuration information try just specifying the email header and body and test to see if the email is received.

The email body is sent as a HTML text block so that you can include any HTML tags within the email body to be rendered in the email reader's client.

If you want to send the email to more than one address include multiple addresses (up to 16) using commas in the Email To field.

Watch the "Emails Sent" text in the lower right corner of the interface. Once you see it increment check your email account. Be sure to check your junk mail folder in case it ended up there!

See Also

[WebServer](#)

Image Distributor (Client & Server)

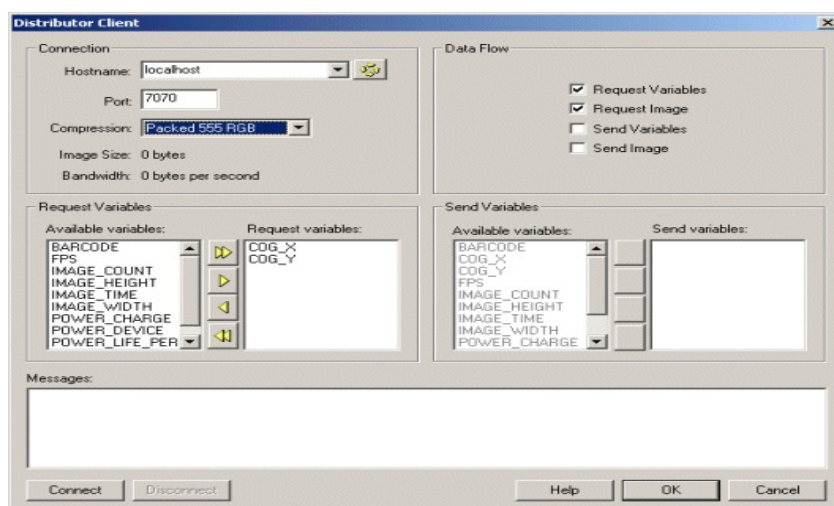
The distributor module provides a gateway into parallel distributed image processing where it is possible to use more than one computer to process images at a time. Using more than one computer enables more sophisticated processing that would otherwise over-tax a single computer. For example one can perform some basic processing on a mobile robot and then wirelessly transmit the image off robot to another networked system to perform more extensive analysis. We use this technique to simply display what the robot is seeing on another machine (our robots typically do not have monitors attached) and to also save that stream to a file for later playback in our tutorials.

The distributor system is broken into a client and server component. The server component distributes the 'source' image. Clients will connect to the server to grab images and variables as needed. Clients can also return the images (similar to our [plugins](#)) and/or variables to the source server computer. For example, a robot would be the server whereas a desktop connected to that robot would be the client.

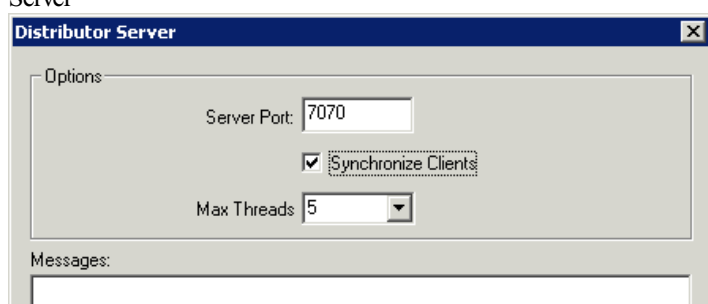
Adding the client distributor to the processing pipeline will insert two tags into the pipeline. Anything placed between these tags will be executed with the results being passed back to the server if the appropriate checkboxes are selected in the distributor client interface. In this way each client can perform a transformation that is then transferred back to the server as either an image or just resulting variables.

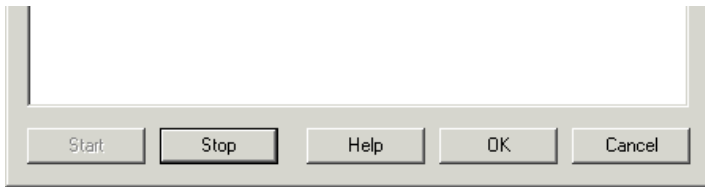
Interface

Client



Server





Server Instructions

1. Specify the port number on which to server images. Default is 7070
2. Specify if you want all the client requests to be synchronized. If the client requests are synchronized then all clients need to respond before the server will display the next image. This is useful when you require each client to process the current image before the server can continue. If you do not have this requirement leave the checkbox unchecked as this can slow the processing unnecessarily to the slowest connected client.
3. Specify the maximum number of threads that are created to serve images. If you have a large number of clients connecting to the server increase the value from the default 5. Note that you will need to stop and start the server to accept this change.
4. Press 'Start' to start serving images and variable data.

Client Instructions

1. Specify the server hostname that you will grab the image from.
2. Specify the port number on the hostname which will serve images. Default is 7070
3. Specify the compression performed when sending images to and from the client. See below for explanation of the compression technique.
4. Specify what types of data you are requesting and sending. Selecting the least amount of information flow will help to reduce network traffic. For example, if you are just viewing the video stream just select the 'Request Image' checkbox. You can also specify which variables specifically you want to send or request. This will help to reduce network traffic and prevent variables from the two communicating programs from overwriting each other as the variables are updated.
5. Press 'Connect' to connect to the server and start viewing the video.

Compression Techniques

In order to reduce network latencies several compression techniques are made available to compress image data prior to transmission to and from the client. Each technique has its advantages based on what your intended use of the image data is. The techniques are explained below relative to a 320x240 image.

- None - No compression. Image size is 320*240*3 bytes
- Packed 555 RGB - RGB 24 bit (3 bytes) image is packed into 15 (2 bytes) data. Image size is 320*240*2 bytes
- Packed 332 RGB - RGB 24 bit (3 bytes) image is packed into 8 bits data. Image size is 320*240 bytes

- Packed 323 RGB - same as 332 except that the green channel is given only 2 bits
- Packed 233 RGB - same as 233 except that the blue channel is given only 2 bits
- Greyscale - color image is reduced to greyscale image. Image size is 320*240
- LZW - image is compressed using LZW (GIF) compression with delta (previous frame is subtracted from current) compression. Image size varies but will be around 150000 bytes.
- Low Lossy 1 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is low. Image size varies but will be around 60000 bytes.
- Lossy 2
- Med Lossy 3 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is medium. Image size varies but will be around 15000 bytes.
- Lossy 4
- High Lossy 5 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is high. Image size varies but will be around 3000 bytes.
- Jpeg Superb - image is compressed using highest quality Jpeg compression.
- Jpeg Good
- Jpeg Normal
- Jpeg Average
- Jpeg Bad - image is compressed using lowest quality Jpeg compression.

Variables

DISTRIBUTOR_CLIENT_CONNECTED - when using the client module equals 1 when the client is connected to a server, removed when not.

DISTRIBUTOR_SERVER_CONNECTIONS - when using the server module, this equals the

number of currently connected distributor clients.

See Also

[Plugins](#)

Image Distributor (Client & Server)

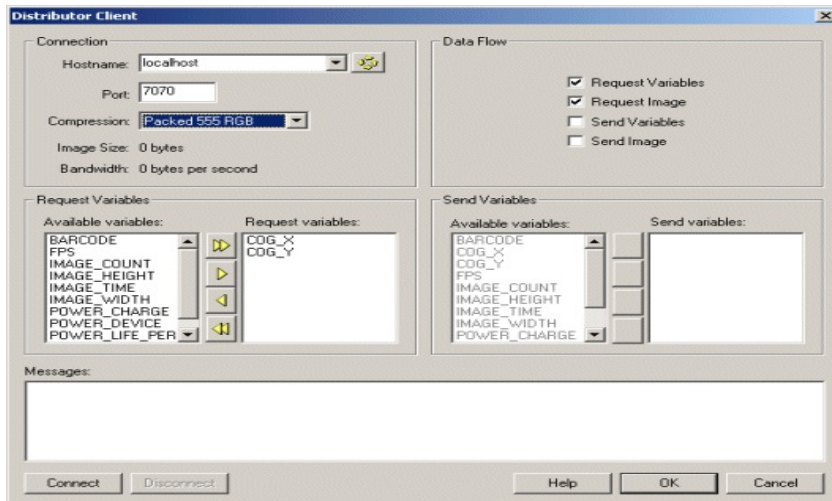
The distributor module provides a gateway into parallel distributed image processing where it is possible to use more than one computer to process images at a time. Using more than one computer enables more sophisticated processing that would otherwise over-tax a single computer. For example one can perform some basic processing on a mobile robot and then wirelessly transmit the image off robot to another networked system to perform more extensive analysis. We use this technique to simply display what the robot is seeing on another machine (our robots typically do not have monitors attached) and to also save that stream to a file for later playback in our tutorials.

The distributor system is broken into a client and server component. The server component distributes the 'source' image. Clients will connect to the server to grab images and variables as needed. Clients can also return the images (similar to our [plugins](#)) and/or variables to the source server computer. For example, a robot would be the server whereas a desktop connected to that robot would be the client.

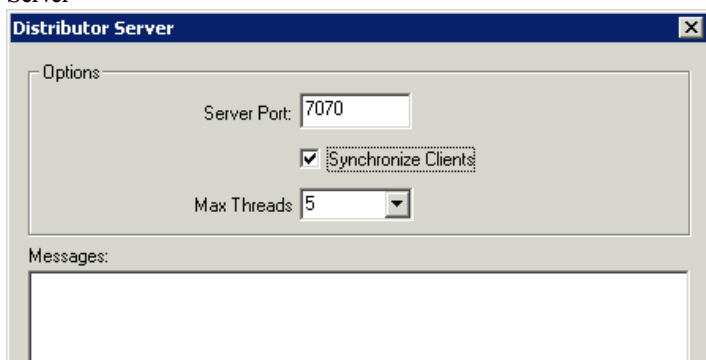
Adding the client distributor to the processing pipeline will insert two tags into the pipeline. Anything placed between these tags will be executed with the results being passed back to the server if the appropriate checkboxes are selected in the distributor client interface. In this way each client can perform a transformation that is then transferred back to the server as either an image or just resulting variables.

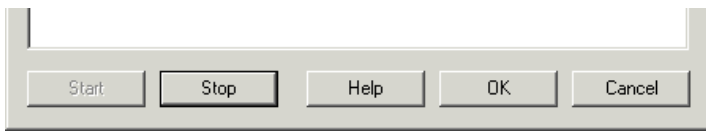
Interface

Client



Server





Server Instructions

1. Specify the port number on which to server images. Default is 7070
2. Specify if you want all the client requests to be synchronized. If the client requests are synchronized then all clients need to respond before the server will display the next image. This is useful when you require each client to process the current image before the server can continue. If you do not have this requirement leave the checkbox unchecked as this can slow the processing unnecessarily to the slowest connected client.
3. Specify the maximum number of threads that are created to serve images. If you have a large number of clients connecting to the server increase the value from the default 5. Note that you will need to stop and start the server to accept this change.
4. Press 'Start' to start serving images and variable data.

Client Instructions

1. Specify the server hostname that you will grab the image from
2. Specify the port number on the hostname which will serve images. Default is 7070
3. Specify the compression performed when sending images to and from the client. See below for explanation of the compression technique.
4. Specify what types of data you are requesting and sending. Selecting the least amount of information flow will help to reduce network traffic. For example, if you are just viewing the video stream just select the 'Request Image' checkbox. You can also specify which variables specifically you want to send or request. This will help to reduce network traffic and prevent variables from the two communicating programs from overwriting each other as the variables are updated.
5. Press 'Connect' to connect to the server and start viewing the video.

Compression Techniques

In order to reduce network latencies several compression techniques are made available to compress image data prior to transmission to and from the client. Each technique has its advantages based on what your intended use of the image data is. The techniques are explained below relative to a 320x240 image.

- None - No compression. Image size is 320*240*3 bytes
- Packed 555 RGB - RGB 24 bit (3 bytes) image is packed into 15 (2 bytes) data. Image size is 320*240*2 bytes
- Packed 332 RGB - RGB 24 bit (3 bytes) image is packed into 8 bits data. Image size is 320*240 bytes

- Packed 323 RGB - same as 332 except that the green channel is given only 2 bits
- Packed 233 RGB - same as 233 except that the blue channel is given only 2 bits
- Greyscale - color image is reduced to greyscale image. Image size is 320*240
- LZW - image is compressed using LZW (GIF) compression with delta (previous frame is subtracted from current) compression. Image size varies but will be around 150000 bytes.
- Low Lossy 1 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is low. Image size varies but will be around 60000 bytes.
- Lossy 2
- Med Lossy 3 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is medium. Image size varies but will be around 15000 bytes.
- Lossy 4
- High Lossy 5 - image is compressed with Harr Wavelet (color reduced), Delta, and finally LZW compression. Lossy factor is high. Image size varies but will be around 3000 bytes.
- Jpeg Superb - image is compressed using highest quality Jpeg compression.
- Jpeg Good
- Jpeg Normal
- Jpeg Average
- Jpeg Bad - image is compressed using lowest quality Jpeg compression.

Variables

DISTRIBUTOR_CLIENT_CONNECTED - when using the client module equals 1 when the client is connected to a server, removed when not.

DISTRIBUTOR_SERVER_CONNECTIONS - when using the server module, this equals the number of currently connected distributor clients.

See Also

[Plugins](#)

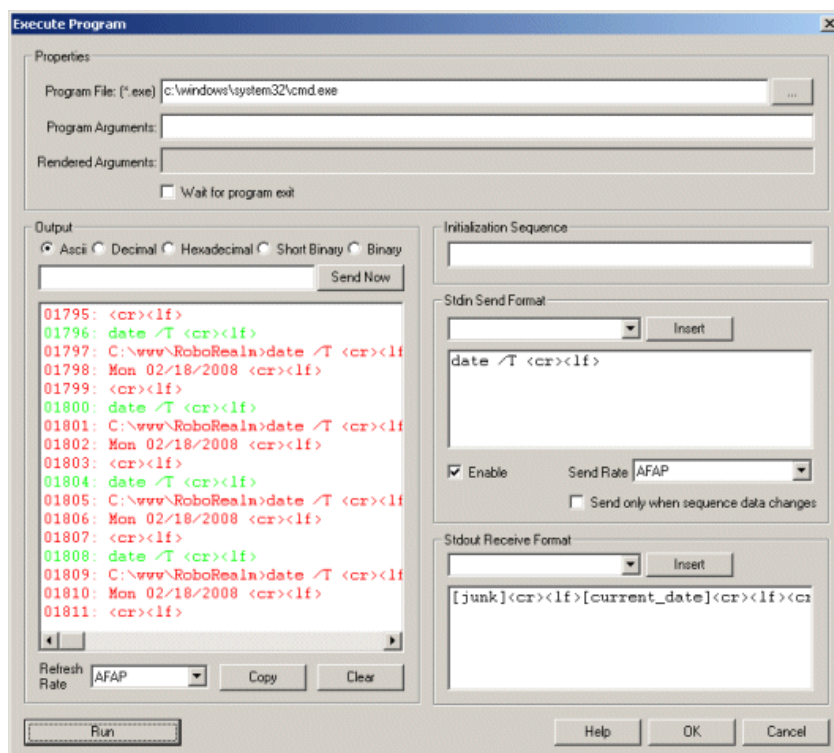
Execute Program

The Execute Program module will execute a windows based program and allow you to interact with it using the stdin and stdout pipes. This module allows you to command external programs whose source may not be accessible and does not provide for any API means of communication.

Most DOS based programs read input from the stdin pipe and write output to the stdout pipe. By specifying the text to send to the stdout and creating a parsing format to read the output text you can read and write information to the external program.

For example, below you can see the interface needed to execute the DOS cmd.exe and issue a "date" command and read in the return value.

Interface



Instructions

1. Program File - Specify the program file to run
2. Program Arguments - Specify any arguments that are passed on the command line to the program. Note that you can use RoboRealm variables by specifying [variable_name] surrounded by []'s

3. Wait for program exit - wait for the termination of the program before continuing to the next processing cycle.
4. Console - you will see the programs output in the console area as would be printed to the stdout (command console) when running that application. This shows the current values being read from the program and provides a log of the ongoing communication activities. The green text is from RoboRealm, the red text are characters received and the green text are characters sent by RoboRealm to the device. To copy the log click on the Copy button, likewise to clear it click on the Clear button. Note that the console log only shows several lines at a time with older information being discarded.
You can switch the output to various formats for better viewing:
 - ASCII - shows the data as ASCII characters. Any character outside of printable characters will be represented with a \ followed by the actual integer data that was read
 - Decimal - shows the data as sets of single byte integer numbers
 - Hexadecimal - shows the data as sets of hexadecimal numbers (base 16 numbers, e.g. FF)
 - Short Binary - shows the data as binary numbers with prefix 0's removed
 - Binary - shows the data as sets of single byte binary numbers
5. Send Now - often you need to quickly test the application by sending a certain sequence of characters. To do so just type in the character sequence (using for carriage return, [image_count] for variables, etc.) and click on Send Now. The text will be parsed and sent to the program and the response will then appear in the console log. Note that this is different from the send sequence which will be sent each time the module is encountered in the processing pipeline. The Send Now button is a manual testing mechanism meant for debugging purposes. Also note that the returned text will be parsed by the Receive Sequence so that you can test your parsing code and see if the variables have been created. Use the [Watch Variables](#) module to see those variables being created.
6. Refresh Rate - to slow the scrolling of output select a different refresh rate for the console. This will just slow down how quickly RoboRealm reads information from the application.
7. Initialization Sequence - The initialization data sequence sends the provided string to the program on initialization of communication. You may want to use this to command the custom application into a specific mode ready for communication with RoboRealm. This initialization sequence is sent each time the communication is reset. This happens when you click on the "Stop" button in this interface, change one of the above parameters (Baud, Port, etc) or when the RoboRealm starts running for the first time. It is NOT sent when the Run button in the main RoboRealm interface is toggled. See the [Text Formats](#) page for additional information about the text string format.
8. Send sequence - Used to enter commands sent per pipeline loop (i.e. image processed) by RoboRealm. You can use this sequence to transmit variables created by other RoboRealm modules to the program. Each time an image is captured and processed the Execute Program module will interpret the Send Sequence text and send the result to the program. See the [Text Formats](#) page for additional information about the text string format.
9. Enable - Allows you to temporarily disable sending text to the program while performing edits. Note that the Send Sequence textarea will turn red to indicate this setting.
10. Send Rate - Some applications cannot handle data rapid streams. Use this dropdown to select how quickly you want the data to be sent. At AFAP (As Fast As Possible) the data will be sent out about 30 times a second (this assumes a camera running at 30 fps).
11. Send only on change - If your data does need to be sent out to your program every iteration through the processing pipeline loop this selection will prevent the same data from being sent to the program that was last sent. This is also an elegant way to reduce the data bandwidth to the program if your sequence does not change rapidly.
12. Receive sequence - used to receive and parse text send from the program. The text string is matched against the incoming bytes. When a match is found variables are added into RoboRealm for use in other modules. Reading into variables just requires adding in a variable at the appropriate spot within the receive string similar to the scanf routine in C/C++. The Receive sequence works similar to an expect string, i.e. you need to specify patterns that match the incoming text and substitute the areas that need to be fed into variables with the [variable_name] format. Note that even if you are missing one space or newline the patter will not match and the variable will be zero or blank. See the [Text Formats](#) page for additional information about the text string format.

Notes

For an example of how to write a program using the Execute Program module you can download the [Plugins.zip](#) file. You will find a Console folder in there with an example program on how to write an Execute Program. The example program will receive input from RoboRealm and write whatever it receives (terminated by a newline) to the file "c:\temp\log.txt".

The "Stdin Send" text would then be

[COG_X]

[COG_Y]


to cause that program to write the COG variables to disk (this assumes you have inserted the COG module somewhere in the pipeline. Note that the "Program File" would need to be something like

C:\RoboRealm\Plugins\Console\Debug\Console.exe

Quick Example

Create a file called c:\temp\test.bat with the following one line:

```
@echo test text
```

which will print "test text" to the console. Then run this  [robofile](#) and edit the Watch_Variable module to see the variable "test" contain the contents being printed in that .bat file.

Do you see the flashing black box? That is the .bat file being run each time and then executing.

See Also

[Pipe Program](#)

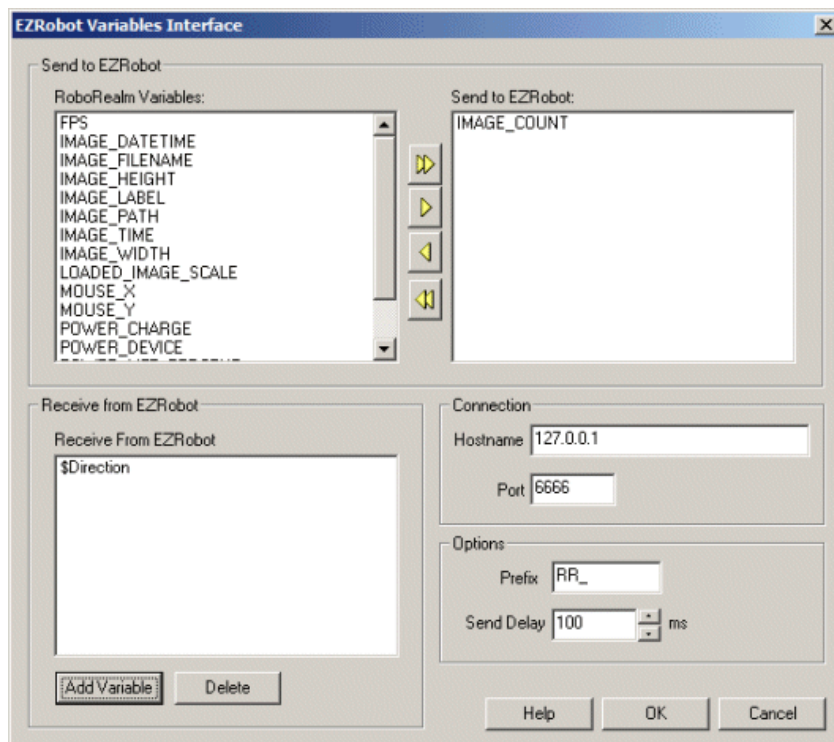
[VBScript Program](#)

EZRobot Variables



The EZRobot Variables module provides a communication mechanism between the [EZBuilder](#) application and RoboRealm. This module allows you to set and get variables from the two systems to allow for information to flow back and forth between the two applications. This allows both applications to combine abilities in order to accomplish your goal.

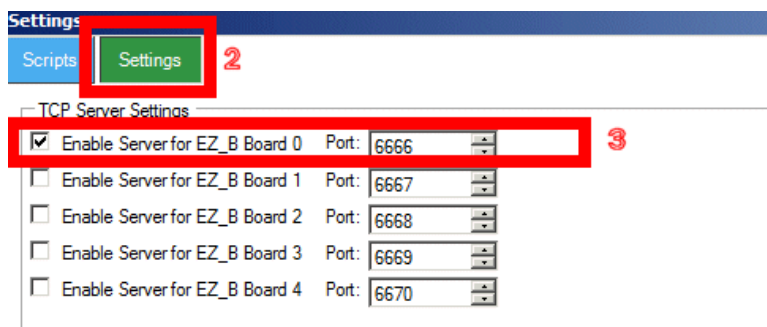
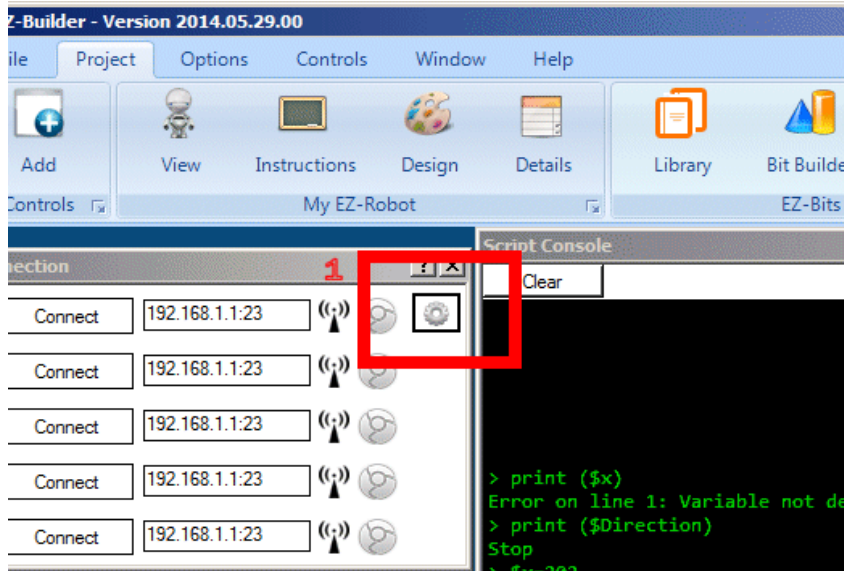
Interface



Instructions

1. RoboRealm Variables - The current variables that RoboRealm knows about within its application. These constitute the list of variables that can be sent to EZBuilder.

2. Send to EZRobot - Specifies those variables that will be sent to the EZBuilder application. Note that the type is auto detected and preserved across into EZBuilder.
3. Receive from EZRobot - Specifies the variables to query from EZRobot back into RoboRealm. Note that the '\$' is removed when moving the variable from EZRobot into RoboRealm. Instead an 'EZ_' is added to the start of each variable to avoid any naming collisions that might happen.
4. Hostname/Port - Specifies the connection hostname and port number that EZBuilder is running on. In order to enable the communication channel, you MUST enable the Connection->Setting icon->Settings tab->TCP Server Settings on Port 6666 within EZBuilder. This enables the TCP server that the module uses to communicate with EZBuilder. You can then use the Watch module within RR and the Variable Watcher within EZBuilder to see the variables changes in the two applications.



5. Prefix - When sending Variables to EZBuilder, the Prefix is prefixed to the variable to avoid any naming collision within EZBuilder and also indicate that the variable has been sent from RoboRealm. You can change this prefix to something else should you prefer a different naming convention.
6. Send Delay - Specifies the maximum rate at which RoboRealm will send variables to EZBuilder. To quickly and more data than needed may get sent, too slow and the data may not be updated within EZBuilder quick enough.

HTTP

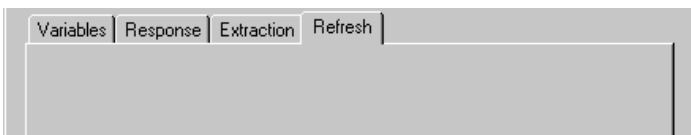
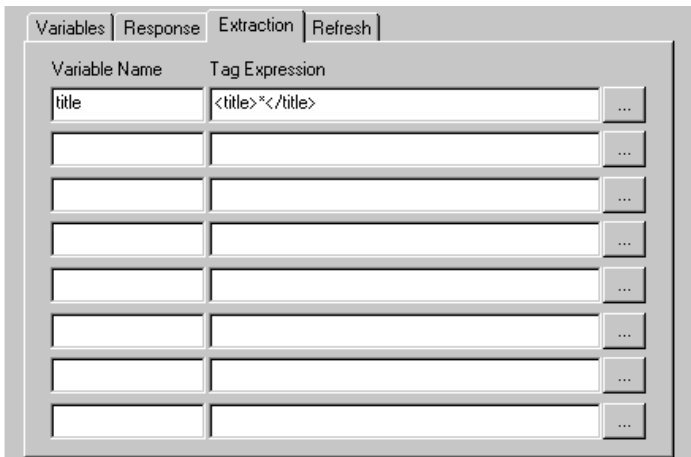
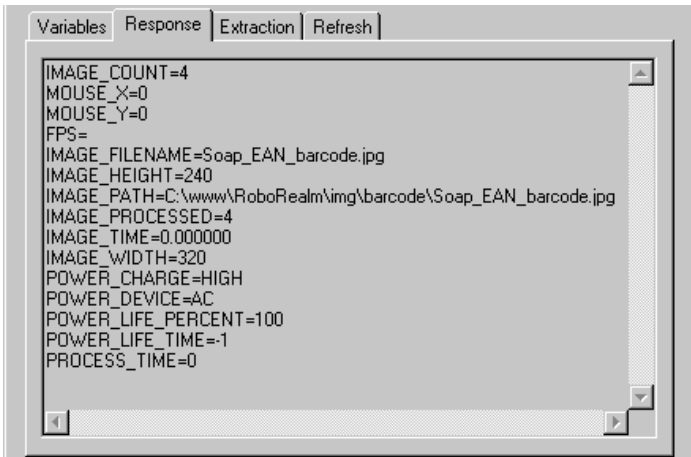
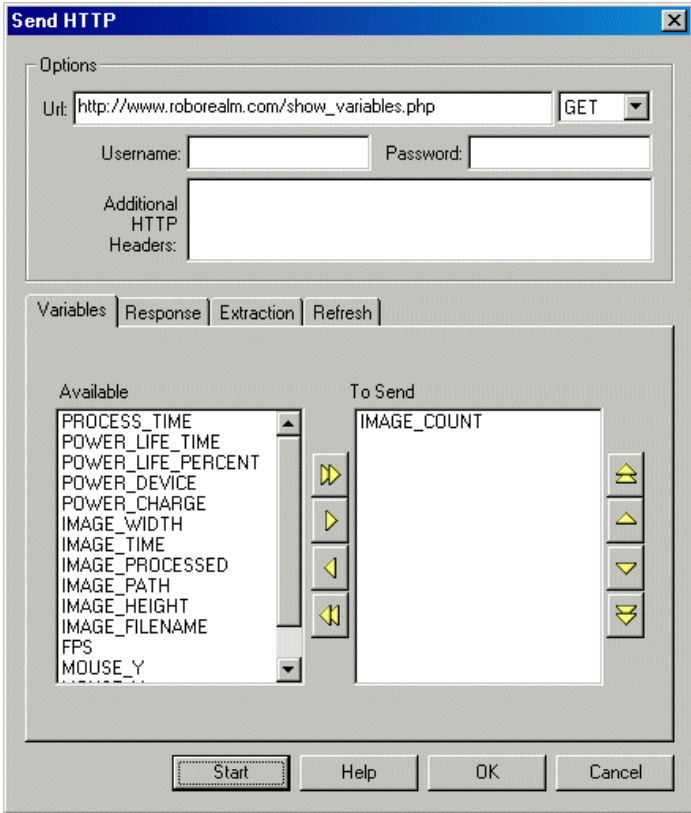


The HTTP module provides a way for RoboRealm to access web pages in order to post information within RoboRealm into remote websites and to read information contained in pages across the web. This is similar to the [HTTP_Read](#) module but allows for customized urls and processing of textual results instead of images as handled by the [HTTP_Read](#).

The main interface provides a selection mechanism to specify what variables are to be send given the provided url. This allows you to create a webpage on your website that will receive variables posted to it from RoboRealm.

Note that like other modules the url can be specified as text OR use the [variable] [expressions](#) notation to access the contents of a variable for the actual url to use.

Interface



Maximum Rate 1 every 5 seconds Requests: 2
 Only when variable arguments changes

Instructions

1. Url - Specify the url of the webpage that you would like to access. This is the url that you would normally paste into your browser location bar. You can have arguments at the end of the url in addition to specifying variables that would be sent too.
2. GET/POST - Specify what method of request you would like to use. If you are sending large amounts of information you may need to specify POST, otherwise GET should accomplish most tasks.
3. Username / Password - Specify the username/password if HTTP Basic auth is required to gain access to the webpage. Note that this is ONLY used when you see the browser popup its own window requesting for a username and password to gain access. If your website has a HTML login page in order to gain access to the website this username and password will NOT work for that form of login.
4. Additional HTTP Header - You can specify additional HTTP parameters (perhaps for login authorization) that will be sent with the query. This is provided for those advanced users that have a deep understanding of HTTP requests and what information can be provided along with those requests. Note that this module will issue requests as if they came from an IE browser (USER_AGENT).
5. Variables - Select those variables that should be sent to the webpage specified in the URL textbox. All specified variables will be added as CGI parameters and sent to the webpage. Those variables that are arrays will be converted to comma delimited strings and sent as such.
6. Response - After press the Start button the Response text area will show the text returned from the specified url. You can use this tag to investigate the returned text and/or errors that may be returned from the webserver as HTML. The returned text can then also be used as a guide to determine what information should be extracted back into variables for use within RoboRealm.
7. Extraction - As webpages come in various different formats the Extraction tab allows you to create HTML tag expressions that define where within a page is the piece of text to be extracted and placed into a variable. The Variable Name field should be used to specify a variable name that will become a variable within RoboRealm that contains the extracted text. The tag expression is a parsing expression that defines the start and end of the text to be extracted using HTML or XML tags to delineate the target text. For quick access to several examples click on the [...] button and select one of the displayed expressions. See below for more information on allowed formatting and specifications for tag expressions.
8. Options - Specify how frequently you want the request to be made. Note that when viewing a live webcam the rate of pipeline execution can be around 30 frames per second which would mean that 30 HTTP requests could be issued within one second. To lower the load on your webserver you should specify an appropriate request rate. The default is set to restrict requests to at most 1 per second. As an additional requirement you can also select the "Only when variable arguments change" checkbox which will only reissue the HTTP request when at least one of the variables that you specified in the first tab actually change. If nothing changes then no HTTP request is sent which assumes that no new information is present and thus no HTTP request is required. If no variables are being used this checkbox is ignored.
9. Options - Async - If you want to execute a query but not slow down the pipeline and only get the value when it is returned you can select the synchronous execution checkbox. This tells the module to issue the HTTP request but not wait for the answer and instead continue processing. When the request is returned by the remote webserver the information will be added as variables when the module is executed on the next pipeline iteration. This allows you to query slower sites without slowing down the pipeline execution in order to wait for the reply.
10. Start - Press the start button to execute a single request and then wait based on the refresh rate (and changed data) to execute the next. Note that regardless of the refresh rate one HTTP request will execute to allow you to see any errors and possible results to any extraction being performed.

Tag Expression

Tag expressions permit you to isolate a piece of text within the HTML page based on its context within the page. This allows you to perform text extraction on pages not necessarily built to make extraction easy (i.e. the returned text is meant for humans to look at and not as an XML, RSS feed, CSV file, etc). Most text within HTML pages is surrounded by HTML or XML tags that can be used to uniquely identify that text within the page. The tag expression is how you specify that context.

For example, suppose you wanted to extract out the title from a document such as

<html>

```
<head><title>This is a test</title></head>
```

```
<body>...</body>
```

```
</html>
```

you could use the tag expression `<title>*</title>` to extract out "This is a test". The expression basically means look for the title tag, then extract out text (i.e. `*`) until you see the end title tag. Thus you use the HTML tags to delineate what text to extract. A couple more examples:

Text: `<div class="header">This is a test</div>`

Expr: `<div class="header">*</div>`

Result: Extracts out "This is a test" within a div tag with the class equal to "header".

Text: `<meta name="author" content="Roy Lichenstein">`

Expr: `<meta name="author" content="*">`

Result: extracts out "Roy Lichenstein" or the value of content within an HTML meta tag

Text: `thi<e><r>ffe</r></e>s<g>test</g>`

Expr: `!<g>&</g>`

Result: Extracts out the word "test" since only text is extracted as specified by `!`

Text: `bolded`

Expr: `<i>*</i>|*`

Result: Extracts out "bolded" in either bold or italic tags as specified by the 'or' symbol `|`.

Text: `hellofromworld`

Expr: `*`

Result: Extracts out "from" since it is 2 font levels deep (i.e. nested tags).

Text: `hellofromworld`

Expr: `#`

Result: Extracts out "Hellofromworld" since the first font tag triggers the match and the `#` means extract out text and tags.

Text: `hellofromworld`

Expr: `*`

Result: Extracts out "Hellofromworld" since the first font tag triggers the match and the `*` means extract out only text and ignore the second level font tags.

Text: `From Andy Minors`

Expr: `From *`

Result: Extracts out "Andy Minors" since the bold tag and "From" text all match.

Text: `Cost $191.00`

Expr: `Cost \\$*`

Result: Extracts out "191.00" since the bold tags match and the word "Cost" starts the text. Note the use of `\\` to indicate that the '\$' is NOT a tag expression character but an actual literal character in the text. '\$' means skip text when no `\\` proceed it in the expression.

Text: `<div id="extract">text to be extracted</div>`

Expr: `<any id="extract">*</any>`

Result: Extracts out "text to be extracted" since the div tag matches the 'any' tag and the ids are the same in the text and the expression.

Text: `1,2,3,4,5,6,7,8`

Expr: `!,!,!,*,`

Result: Extracts out the single letter "4" since the first 3 numbers are ignored using `!` and all the commas match the text. Note that if the final comma were not present the extracted text would instead be "4,5,6,7,8".

Specifically the symbols understood by the tag expression parser are as follows

* means record text but skip tags.

means record text and tags.

! means skip text and tags.

% means skip text but record tags.

\$ means skip text only.

& means record text only.

+ means start recording matched tags.

- means stop recording.

Html Tag expressions can also use qualifiers within the tags to further refine the extraction process. The following are valid qualifiers:


depth = X - means the tag needs to be nested X deep in order to match.


index = X - means the X occurrence of the tag will only match.
start = X - means match the tag only after X of those tags has been encountered.
end = X - means stop match after the X matches have been made.
row = X - means match the Xth row within a table tag.
row_start = X - means match starting with the Xth row within a table tag.
row_end = X - means stop matching after the Xth row within a table tag.
column = X - means match only the Xth column within a table tag.
column_start = X - means match starting with the Xth column within a table tag.
column_end = X - means stop matching after the X columns have been matched within a table tag.


To reference any tag with a specific attribute use the tagname 'any'.


Examples


 [Test Submit](#) - Issues a request to the RoboRealm website at http://www.roborealm.com/show_variables.php?name=value which just returns back the variables and values that were submitted by the module. This is a useful test case just to see that information is flowing over to and back from the website correctly.


 [Stock quote](#) - Queries [Yahoo](#) for stock information. Shows the configuration and extraction information to determine the Open, Close, High, Low and Change of the [IRobot](#) stock. Note that the url includes the stock ticker symbol of IRBT. You can change that ticker to any other stock you wish to query. The information returned by the url is a comma delimited CSV file which is parsed by the extraction tag expressions.

 [Headline News](#) - Queries [Yahoo](#) for the first title of the headline news RSS feed. Note that while an RSS feed is actually XML the format is the same as HTML and thus the tag expressions work well to extract out just the required information.

 [Weather](#) - Queries [Yahoo](#) for the weather in Sunnyvale California. Have a look at [Yahoo Weather Access](#) to see what to change the 2502265 code to for your location. Again, while the information is being passed back as an RSS feed it is XML and thus the tag expressions work well to extract out just the required information.

 [Joke](#) - Queries [Joke of the Day](#) and extracts out the joke displayed on that homepage. In this case the information was not presented in an XML or CSV format but is extracted out from the page using the class = "quote content".

 [Horoscope](#) - Queries [www.astrology.com](#) website for the horoscope for Aries. Once again a div and a HTML class is used to isolate just the horoscope text in the page.

 [RoboRealm count](#) - Queries [Google](#) to see how many results there are for the query RoboRealm! Yea, ok, this one is just for us!

It is important to note that while very versatile the tag expression language does have a fatal flaw. If the website owners decide to change the format and tag structure of their website (i.e. a redesign) these text extraction expressions may cease to work. Thus it always makes sense to first look for an RSS, XML or other computer format feed and base the extraction on that as those formats are less likely to change when compared to regular HTML pages.

Variables

HTTP_RESULT - always contains the entire source text returned by
executing the HTTP url.

X - any variable specified in the Extraction tab.

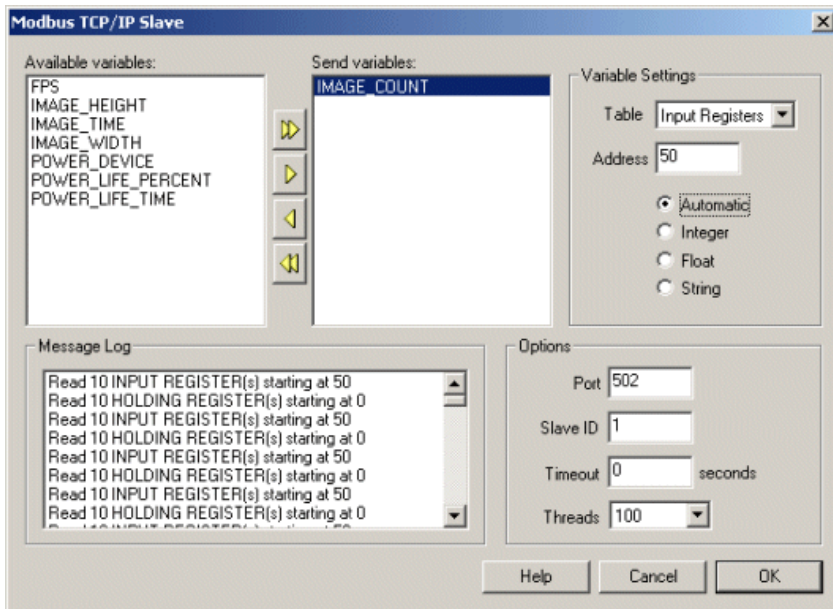
See Also

[HTTP Read](#)

Modbus Slave

The Modbus Slave module provides an interface between RoboRealm and a modbus master such as a PLC device. The modbus slave module requires an Ethernet based master and can handle multiple variable mappings between RoboRealm and the master.

Interface



Instructions

1. Available variables - Displays a list of available variables within RoboRealm that can be mapped to modbus memory locations.
2. Send variables - The list of variables that are configured with modbus settings such that a modbus master can access those variables.
3. Variable Settings - Used to configure how the RoboRealm variables are exposed as modbus registers or coils.
4. Table - The modbus table that the variable should be accessible via.
5. Address - The modbus address that is used to access the variable. This address is relative to the table used.
6. Automatic, Integer, Float, String - defines the type of the variable to transmit. Changing the type will alter the format of the transmitted data. The master modbus must understand the format in order to utilize the variable correctly.
7. Options - Specifies the network settings for modbus.
8. Port - The port that the modbus slave module is listening on in order to accept and process requests.
9. Slave ID - The modbus slave id that this slave module will accept.
10. Timeout - The timeout in seconds of any network interaction between the module and the modbus master.
11. Threads - The number of active threads that will respond to modbus requests. If you only expect to have one modbus master communicate with RoboRealm just use a single thread.

See Also

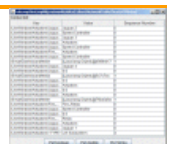
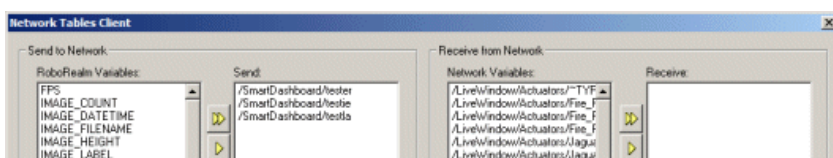
[Open Sound Control](#)

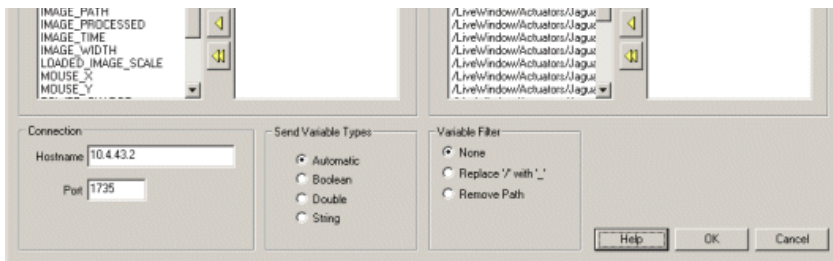
Network Tables

The Network Tables module implements the FIRST FRC Network tables protocol to provide sharing of information amongst machines including that of the CRIO. The module allows receiving and sending of variables to the network table.

Note that this module assumes Network Tables 2.0. It is NOT compatible with versions below 2.0.

Interface





Instructions

1. Connection - Specify the hostname and port for the network tables server. Note that the IP address is typically 10.X.Y.2 where X and Y are created via the team number.
2. Send to Network - Select which variables you want to send from RoboRealm into the network
3. Receive from Network - Select which variables that are currently in the network that you want to import into RoboRealm
4. Send Variables Type - Select how the RoboRealm variables should be interpreted when sent out onto the network. Automatic will chose the best type that represents the variable to send.
5. Variable Filter - Selects how the variable name from the network will be modified when entered into RoboRealm. As RoboRealm can use variables within expressions the '/' symbol can be interpreted as a division sign. To ensure that this does not confuse a RoboRealm expression you can select other options that will remove those '/' from the variable's name.
6. Prefix - Specify what prefix to use when sending and receiving variables. Each time a variable is sent out to the network tables system it will be prefixed with the prefix (default is SmartDashboard). This prevents you from having to rename variables to avoid naming collisions.
7. Send Delay - Delays the sending of information for the specified milliseconds to help group changes and reduce network bandwidth.

Example

To experiance the communication you can use the Set_Variable module within RoboRealm to create a variable, select that variable within the Network Tables module and then accessing it using the following code within the CRIO.

```
NetworkTable server = NetworkTable.getTable("SmartDashboard");

try
{
    System.out.println(server.getNumber("IMAGE_COUNT", 0.0));
}

catch (TableKeyNotDefinedException ex)
{
}
```

When working with arrays you can use

```
NetworkTable server = NetworkTable.getTable("SmartDashboard");

try
{
    final NumberArray targetNum = new NumberArray();
    server.retrieveValue("test1a", targetNum);
    if (targetNum.size()>0)
    {
        System.out.print(targetNum.get(0));

        System.out.print(' ');

        System.out.println(targetNum.get(1));
    }
}
```

```

}
}
catch (TableKeyNotDefinedException exp)
{
}

```

to view the first 2 elements of a double array.

Notes

For those using Network Tables and the roboRIO with 2015 code in the 2016 competition please note that the hostname changed from roboRIO-XXXX.local to roboRIO-XXXX-FRC where XXXX is the team number. Without the correct hostname communication will NOT appear to work as in previous years.

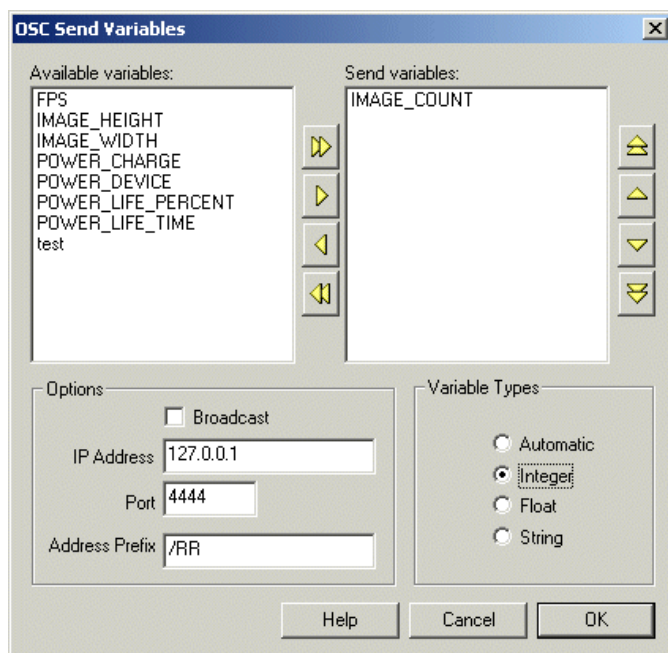
OSC Send Variables

The OSC Send Variables module provides a way for you to transmit variables using the Open Sound Control protocol. The OSC protocol is a simple UDP based way of communicating variable values to other applications like [VYVVV](#) or [Max/MSP](#).

The module allows you to select which variables should be transmitted. Note that as the OSC does NOT provide any means for transmitting large arrays so be weary of transmitting a large array.

Also note that most firewalls will block UPD traffic which may prevent the network packets to flow from one machine to another.

Interface



Instructions

1. Available Variables - Select the variables that you want to send. The left list shows all the current RoboRealm variables available to send given the current processing pipeline. Use the arrows (or doubleclick on a variable) to move it to the 'Send Variables' list. Use the up/down arrows to reorder that list if you want to change the order they are sent in.
2. Broadcast - To broadcast the variable packet on the local network select this checkbox. If not specify an address (IP or hostname) in the IP Address box to transmit the variable only to that machine.
3. Bundle - To bundle multiple variables together select this checkbox. Some applications require multiple variables to be bundled from the same broadcast message.
5. IP Address - If not broadcasting the packet enter the IP address of the receiving machine.

6. Port - Specify which port the data should be transmitted on. This needs to be the same port the receiving program is listening on.

7. Address Prefix - Specify the address prefix that will be used to name the transmitted variable. A transmitted variable will always be called with a suffix equal to the variable name. Using a prefix can better help you to sort the received data. For example, the interface shows the setup for /RR/test to be transmitted as the address.

8. Variable Types - Select how you would like the variable's data to be typed during transmission.

Automatic - Allow RoboRealm to determine the best type based on the values data. For example, "3" would be an Integer, "3.0" would be a float and "a3.0" would be a string.

Integer - Transmit the data using the OSC Integer type

Float - Transmit the data using the OSC Float type

String - Transmit the data using the OSC String type

Example

This [robofile](#) includes a setup to transmit the current IMAGE_COUNT to VVVV running [this VVVV program](#). You will have to have both applications running before you will see the variable changing. Note that the OSC implementation in VVVV is the basic one variable reader that is provided by VVVV as an OSC example.

Pipe Plugin

Please be sure to read the [Overview](#) page to get a general sense of if the Pipe RoboRealm Extension is a good fit for you.

Pipe programs communicate to/from RoboRealm using a windows named pipe. This communication mechanism is quicker than network based communication but also requires a unique pipe name that is shared system wide. If the pipe already exists RoboRealm will connect to that existing pipe otherwise it will run the specified program in expectation that it creates the pipe. This allows you to run your program from a debugged and have RoboRealm connect to that active process for debugging purposes.

To create your own pipe program to extend RoboRealm [download](#) our example pipe program. This example program (SwapColors) will swap the Red and Blue pixel colors in an effort to show how image modifications can be performed. Note that the download also includes the Socket and DLL example implementations. The example pipe program is in the single source file called main.cpp. The other files are used for VC++ project configuration.

You will need VC++ or other C++ compiler to build the program.

Note that a pre-built executable is in the Release subfolder.

To configure RoboRealm to use a pipe based program click on the Extensions->Pipe_Program module.

The following interface elements can be specified:

Name - indicates the name or function of the module. This name is used for display in the main RoboRealm program list area for easy identification and is also passed to the Pipe Program as a 'name' variable that indicates the desired functionality. Using the name variable you can use a single program to perform multiple tasks.

Program File - The name of the pipe program. This is a windows console executable program that should create and open a windows named pipe and begin listening on that pipe for data. Note that RoboRealm checks to see if the pipe exists and connects to it if it does. If not RoboRealm will execute the specified program in expectation that the pipe is created and active.

Program Arguments - You can specify arguments that are passed to the pipe based program that will appear in the argc and argv arguments of that program. This can be used to specify additional parameters to your program in order to customize as needed.

Pipe Name - The communications pipe's name that is used to transfer image pixels and data to and from your program. The default used is rpipe.

Available Variables - Shows you the current RoboRealm variables that other modules have created. Note that the variables are NOT reset during each image call and thus can be used to save information from past images.

Returned Variables - Shows the variables that your pipe program has returned which were added to the RoboRealm variables.

Messages - Any system error messages. If your pipe program passes back the 'error' variable the contents are displayed in this text box.

Reload and Run - While RoboRealm attempts to keep the pipe connection open you may need to restart the pipe program due to failure of some kind.

To start the program specify the .exe file using the browse button. RoboRealm will start the program and pass variables and images to it for processing.

processing.

The data format used to pass the image and associated variables to both pipe and socket based programs is based on the a name value pair. The following is an example of the data stream:

```
5 width 4 320 6 height 4 240 5 image 6 230400 xxxxx ... 0
```

The pair format is

length of name | name | length of value | value | next length or 0 to end

and continues until a 0 or -1 is encountered. 0 means end of the current image step while -1 means end of program.

Note that all variable values are sent as text strings with the exception of width and height which are stored as a four byte number and the actual image which is composed on unsigned BGR pixel triplets.

Things to remember:

- Be sure to use a unique pipe name! If you create more than one Pipe program use a different pipe name or combine the two functions into a single pipe program.
- RoboRealm only processes images when needed! Thus if the RoboRealm window is hidden it will NOT process images and will not call your pipe program. If you are processing a static image RoboRealm will call your pipe module when refreshing part of the window.

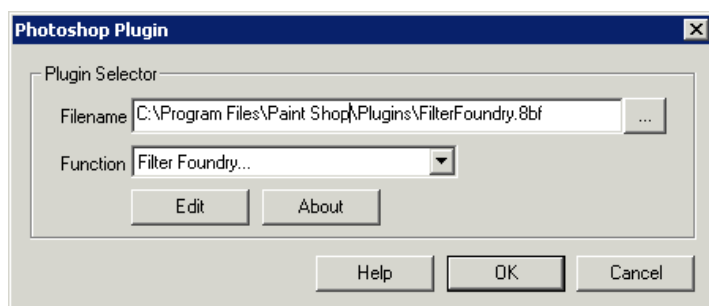
Photoshop Plugin Host

The Photoshop Plugin Host allows you to use a Photoshop plugin within RoboRealm. Note that this module is largely for prototyping purposes as most Photoshop plugins are usually NOT built to be used in video applications. Because of this you will experience:

1. Significant slowdown of the processing pipeline when using a Photoshop plugin. Photoshop plugins are meant to be used on single images. Thus most of the plugins are not high performance filters that can run at video rate.
2. When using a Photoshop plugin that requires a dialog those options selected within the plugin dialog interface will NOT be saved if you exit RoboRealm. This is a limitation on how the plugin filters store and access their own information and do not provide a mechanism for RoboRealm to access the internal information.
3. You may experience significant flashing of the desktop while using plugin filters with dialog interfaces. This is a limitation on how some plugins refresh the desktop screen as apposed to the current image being processed.
4. There are documented routines that the plugins may use that RoboRealm will not support. You may receive "memory errors" from the plugin filters that are you only indication of failure somewhere within the plugin and are not errors generated by RoboRealm.

It is highly recommended that if you use a Photoshop plugin frequently that you either contact RoboRealm or the plugin manufacturer and request the plugin be converted to a native RoboRealm module or refined to work better with video image sequences.

Interface



Instructions

1. Specify the 8bf filename as it resides on your filesystem. Note that this is a file pathname and not a folder. RoboRealm will only access the plugin directly from the 8bf file.
2. Select which function within that 8bf file you wish to execute. Normally this will only include one selection.
3. Press the EDIT button to cause the plugin to display the dialog interface (if one exists).

4. Press ABOUT to read more about the module you have selected.

See Also

[RoboRealm Plugins](#)

For free Photoshop plugins visit

[The Plugin Site](#)

[Graphicextras Free Plugins](#)

[Flaming Pear Software](#)

[Google Search for Plugins](#)

Socket Program Plugin

Please be sure to read the [Overview](#) page to get a general sense of if the Socket RoboRealm Extension is a good fit for you.

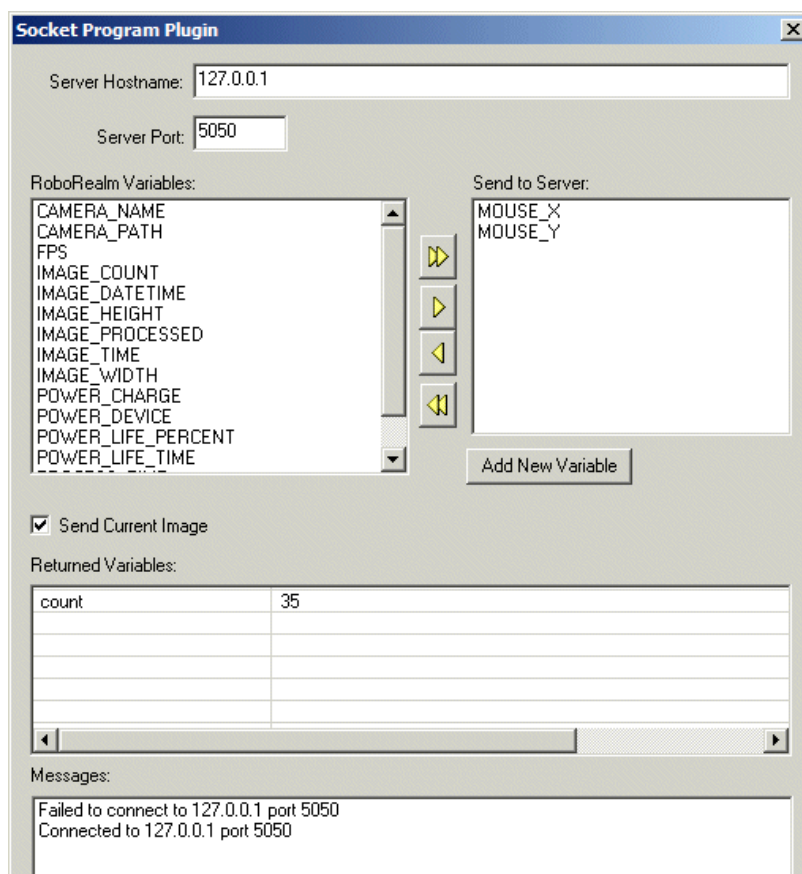
To create your own socket program to extend RoboRealm [download](#) our example socket programs. This example program (SwapColors) will swap the Red and Blue pixel colors in an effort to show how image modifications can be performed. Note that the download also includes the Pipe and DLL example implementations. The example socket program is in the single source file called main.cpp.

There are both C++ and Java based examples for the socket extension. You will need VC++ or other C++ compiler to build the C++ based program and any generic Javac compiler to compile the Java based extension.

Note that there are pre-built executables in each of the examples. If you run these programs (either C++ or Java) you should see a black command dialog with the words 'Running' being displayed. The program is now waiting for a connection from the RoboRealm program on port 5050. Note that if you have any firewall software installed that prevents communication on port 5050 either allow that port or change the port number in the appropriate source file and within the RoboRealm socket module.

To configure RoboRealm to use a socket based program click on the Extensions->Socket_Program module. This module is a CLIENT and thus expects to connect to a network server application.

Interface





Instructions

Server Hostname - This is the computer name or IP address that will play host to the image processing server that you create. This machine needs to be accessible on your network. Localhost - your current computer - is the default. Running your socket server on remote machines will require adequate bandwidth as a single image within RoboRealm is typically 320x240x3 (or larger) in size. Images are NOT compressed in any way when sent across the network.

Server Port - The port number to use when connecting to the host specified above. Please be sure that your Firewall or security proxies allow for that port to be communicated on AND that the TCP protocol is allowed.

Available Variables - Shows you the current RoboRealm variables that other modules have created. Note that the variables are NOT reset during each image call and thus can be used to save information from past images.

Returned Variables - Shows the variables that your socket program has returned which were added to the RoboRealm variables.

Messages - Any system error messages. If your socket program passes back the 'error' variable the contents are displayed in this text box.

Connect - Causes RoboRealm to connect to the specified hostname using the specified port. Use this to test your connection.

While the examples we provide are in C++ using Windows Visual C++ Ver. 6.0 and Java JDK2.1 you are not required to use these platforms in order to extend RoboRealm. By following the same protocol you can write your extensions in other languages like C#, Perl, etc.

The data format used to pass the image and associated variables to both pipe and socket based programs is based on the a name value pair. The following is an example of the data stream:

```
5 width 4 320 6 height 4 240 5 image 6 230400 xxxxx ... 0
```

The pair format is

```
length of name | name | length of value | value | next length or 0 to end
```

and continues until a 0 or -1 is encountered. 0 means end of the current image step while -1 means end of program.

Note that all variable values are sent as text strings with the exception of width and height which are stored as a four byte number and the actual image which is composed of unsigned BGR pixel triplets.

Things to remember:

- If the socket program fails RoboRealm will pause processing and then continue slowly while trying to reconnect each time. Unlike Pipe programs RoboRealm cannot restart the server (we assume the server may not even be on the same machine). Thus you will need to ensure standard monitoring practices to ensure that the system remains stable.
- RoboRealm only processes images when needed! If you are processing a single static image RoboRealm will call your socket module only when refreshing part of the window. So if you are not using a live camera feed you will have to move the RoboRealm window or hide/show that window to ensure that RoboRealm will call your program.

Windows DLL Plugin

Please be sure to read the [Overview](#) page to get a general sense of if the Windows DLL RoboRealm Plugin is a good fit for you.

Extending RoboRealm using a windows DLL is the best way to introduce new functionality into RoboRealm.

To create your own dll plugin [download](#) our example dll program. This example program (SwapColors) will swap the Red and Blue pixel colors in an effort to show how image modifications can be performed. Note that the download also includes the Pipe and Socket example implementations.

You will need VC++ or other C++ compiler to build the program.

Note that a pre-built executable is in the Release subfolder.

To try the example SwapColor plugin click on the Plugins->SwapColor module. This shows a very basic module that provides a GUI interface that can be used to capture user configuration options.

The first Plugins (VBScript_Program, Pipe_Program, etc) are added to the tree menu by default. Afterwards the custom DLL module names will appear. In your current case only SwapColor exists. If you look in your RoboRealm folder you will notice a folder called Plugins which contains the

SwapColor plugin.

Any custom DLL created is registered into RoboRealm by placing it into this folder.

To create your own DLL module use the existing SwapColor example or use VC++ to create a new DLL project.

There are several files in the DLL folder within Plugins.zip. The following are its contents and function:

- HashtableExt - interface class used to provide access into the RoboRealm variable table.
- ModuleDlg - interface class that specifies a RoboRealm module. All of your modules must have this class as its base.
- RoboRealmExt - interface class used to communicate directly back to RoboRealm
- RRModule - container for the DLL
- SwapColorDlg - the actual code for the image processing function to be performed

The main focus of the DLL will be in SwapColorDlg class. This file contains the code to perform the image manipulation routines. There are several calls that RoboRealm makes that you can trap and add functionality. See ModuleDlg.cpp for an explanation of those routines.

For example, the main routine is

```
int SwapColorDlg::Process(unsigned char *data, short int *tmpImage, unsigned char *tmpImage)
```

which is called to process an image. The first parameter is the image data. The second two are temporary buffers that can be used when processing the image. Often an images are processed in destructive ways so a copy needs to be made during processing. Any modifications to the data array will reflect in the actual image displayed by RoboRealm after being passed to all other modules after yours.

Variables

Two types of variable are available to the RoboRealm DLL module. The first is also known by VBScript, Pipe and Socket extensions which are the RoboRealm variables. These variables are shared amongst all RoboRealm modules and can be accessed at any time. Their purpose is to store relevant image information needed to perform processing or for communication between modules.

The second type of variable is the configuration variable that is used to store the configuration specified by users as they interact with the provided module's GUI. It is your responsibility to save this information so that on recall the interface will appear as last configured. This is also needed during program invocations so that users do not need to reconfigure modules during every usage.

Use the SetInt, SetFloat, SetString calls to perform this. See ModuleDlg.dll for additional details.

Things to remember:

- If your DLL crashes it will crash the RoboRealm program too!
- You MUST compile your DLL in Release mode. RoboRealm uses virtual classes to simplify interfacing to the DLL. If you compile your DLL in DEBUG mode the class declarations will NOT be in the same memory alignment as RoboRealm
- When recompiling your DLL you will need to terminate RoboRealm in order for the new DLL to overwrite the existing one. Simply closing RoboRealm will allow it to remember your current configuration (saved in working.xml) and reloaded when you re-run the application. This should minimize the number of clicks in write, compile, and test programming cycles.

CScript (picoc)



Please be sure to read the [Plugins](#) page to get a general sense of if the CScript module is a good fit for your project.

The CScript module provides a way to create custom C scripts that can be used to process image statistics and map them toward servo/motor values. This module is intended to be used as a way to quickly perform custom operations without needing to implement a Plugin or use the API which typically require external tools.

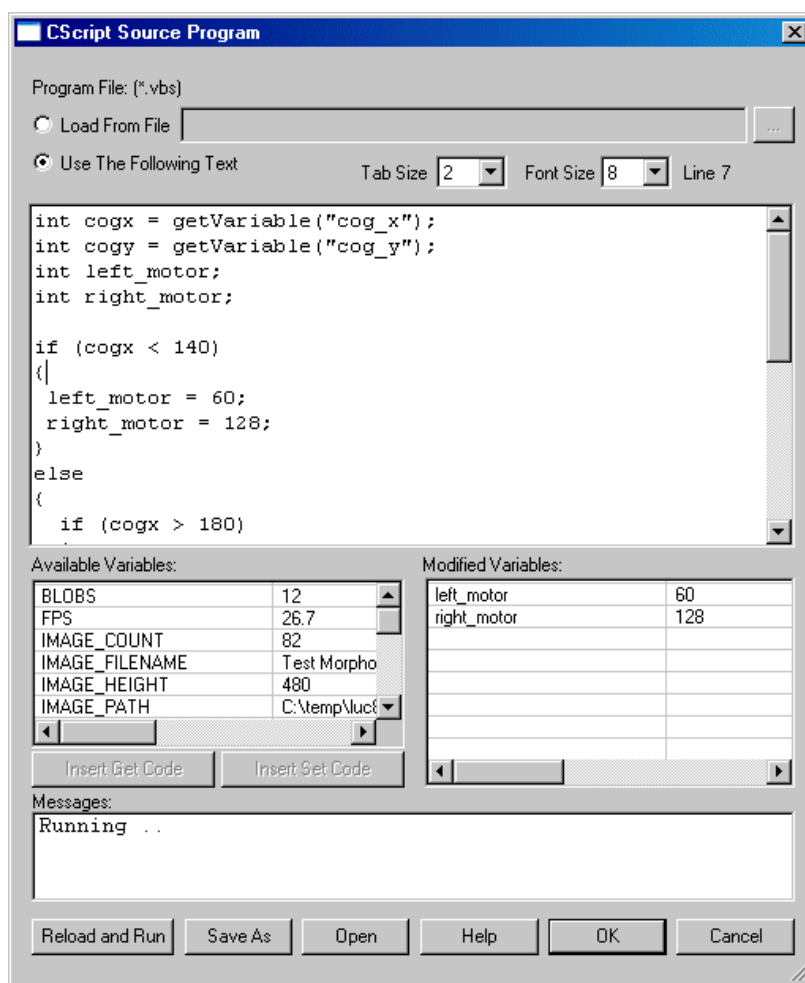
The interface allows two methods for specifying your code. One technique requires that you specify a text file as the source of the program. You can use your favorite text editor to edit/create this file. The file should be a regular text .c file containing CScript commands/functions/operators. The program interface displays the current file being used, available system or user variables (and their values) and any messages the program produces.

The alternate way is to include the CScript text within the provided text box (see below). Using this technique allows you to include the CScript code within the .robo program and not require another user to save the code and reference a text file as needed by the above file based technique. Using the text box provides a quick way to enter in text but is NOT meant as a replacement for a full featured text editor as its editing features are very basic.

The CScript module interprets C code using the [PicoC - A very small C interpreter](#) project and thus will not have the full Windows API available to use. If you need additional capabilities have a look at the [Plugin](#) or [API](#) architecture where you can use the full Visual Studio tools and still communicate with RoboRealm.

As PicoC is interpreted it will run slower than regular C. This will be fine for most applications but if you chose to do image processing directly in this module (similar to the SwapColor example below) the FPS (frames per second) will decrease dramatically as a result.

Interface



RoboRealm Specific Functions

- **int variableExists(char *variable_name);** - returns true (1) or false (0) depending on if the specified variable exists in RoboRealm. Several of the variable return functions below will return a valid value regardless of if the variable exists or not. You can use this function to determine if the value returned from one of these functions is a valid value.
- **int getVariable(char *var);** - returns an integer value of the specified variable
- **float getFloatVariable(char *var);** - returns the float value of the specified variable
- **char *getStrVariable(char *var);** - returns the string value of the specified variable
- **int getArrayVariable(char *var, int *arr, int max);** - returns the int array associated with the specified variable in the array "arr" for max entries. The number of items stored in the provided array is returned. Note that the array NEEDS to already be allocated or statically declared (int arr[64]).

- **int getFloatArrayVariable**(char *var, float *arr, int max); - returns the float array associated with the specified variable in the array "arr" for max entries. The number of items stored in the provided arr is returned by the function.
- **int getStrArrayVariable**(char *var, char *arr, int max); - returns the string array associated with the specified variable in the char buffer "arr" up to max size. Each entry is spaced within 64 characters of arr such that &arr[0] is record 1, &arr[64] is record 2, etc. Note that max is the total maximum number of characters to be used including any padding below 64.

```
int arr[64];

int num = getArrayVariable("BLOBS", arr, 64);

printf("%d %d", num, arr[0]);

arr[0]=33;

setArrayVariable("BLOBS", arr, num);
```

- **void setVariable**(char *var, int value); - sets the integer variable value to the specified variable
- **void setFloatVariable**(char *var, float value); - sets the float variable value to the specified variable
- **void setStrVariable**(char *var, char *value); - sets the string variable value to the specified variable
- **void setTimedVariable**(char *key, int value, int timeout); - - sets the variable value to the specified variable after the timeout (milliseconds) has passed. This allows you to specify when in time a variable's value should be changed. Note that each call to SetTimedVariable will reset the timer so be sure to only call it once to allow the time to lapse and set the specified value. This function is useful when you need an alternative to the Sleep function. By setting a variable to a value in the future you can emulate a Sleep function by allowing something to happen for a period of time, but then switching it off after the time has lapsed. So instead of using

```
setVariable("motor", 100);
sleep(1000);
setVariable("motor", 0);
```

you could use

```
setVariable("motor", 100);
setTimedVariable("motor", 0, 1000);
```

- **void setFloatTimedVariable**(char *key, float value, int timeout); - sets the float variable value to the specified variable after the timeout (milliseconds) has passed
- **void setStrTimedVariable**(char *key, char *value, int timeout); - sets the string variable value to the specified variable after the timeout (milliseconds) has passed
- **void addTimedVariable**("variable_name", variable_value, timeout); - similar to SetTimedVariable, this function will instead add the timing parameter to a queue instead of overwriting the timing information. For example:

```
setTimedVariable("motor", 100, 1000);
setTimedVariable("motor", 200, 1500);
setTimedVariable("motor", 300, 2000);
```

will result in the motor variable being set to 300 in 2 seconds (2000 ms). What happens is that the first two calls to setTimedVariable are **OVERWRITTEN** by the third. Instead, if you use

```
addTimedVariable("motor", 100, 1000);
addTimedVariable("motor", 200, 1500);
```

```
addTimedVariable("motor", 300, 2000);
```

the motor variable will be set to 100 in 1 second, 200 in 1.5 seconds and finally 300 in 2 seconds. The addTimedVariable will add each call into a list that is executed in sequence based on the specified time.

- void **addFloatTimedVariable**("variable_name", variable_value, timeout); - float version of addTimedVariable
- void **addStrTimedVariable**("variable_name", variable_value, timeout); - string version of addTimedVariable
- void **sleep**(X); - suspends processing for X number of milliseconds and returns control to the pipeline to acquire additional images and send values to external devices.
- void **waitVariable**("variable_name", variable_value); - suspends VBScript processing until the specified variable equals the specified value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the variable value.
- void **waitVariableChange**("variable_name"); - suspends VBScript processing until the specified variable changes its value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the value to change.
- void **waitImage**() - suspends VBScript processing and returns control to the pipeline for one pipeline iteration. This allows for the acquisition of additional images and to send values to external devices.
- unsigned char ***getPixels**(int *width, int *height); - returns an array of BGR values of the current image and sets width and height of the image. Note that the returned array is an unsigned char with Blue first.
- void **setPixels**(unsigned char *, int , int); - sets the current image to the specified pixel BGR data
- void **setParameter**(char *module, int count, char *parameter_name, int paramter_value); - changes the specified parameter in the specified module to the specified value. This is useful when a GUI dialog does not have a variable selection as part of the dropdown. Avoid using this function unless necessary as it is slow to update the GUI. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- void **setFloatParameter**(char *module, int count, char *parameter_name, float paramter_value); - float value of above routine.
- void **setStrParameter**(char *module, int count, char *parameter_name, char *paramter_value); - char value of above routine.
- int **getParameter**(char *module, int count, char *parameter_name); - returns the specified parameter in the specified module. This is useful when a GUI dialog does not expose a value within a variable. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- float **getFloatParameter**(char *module, int count, char *parameter_name); - the float version of the above routine.
- char ***getStrParameter**(char *module, int count, char *parameter_name); - the string version of the above routine.
- void **stopProcessing**(); - stops RoboRealm from continuing to process the images with the remaining modules
- void **cameraOff**(); - switches off the image capturing from the current camera
- void **cameraOn**(); - switches on the image capturing using the current camera
- void **pushButton**(char *module, int count, char *key); - automates pushing a button in one of the RoboRealm GUI windows. For example:

```
if (strcmp(getStrVariable("test"), "pushed") !=0)
{
    setStrVariable("test", "pushed");
    pushButton("Read_AVI", 0, "Start");
}
```

will cause the Read_AVI module (assuming one exists in the pipeline) to start playback.

```
pushButton("RoboRealm", 0, "Snap");
```

will cause RoboRealm to create a snapshot of the current image.

- float **atan2**(x,y); - arc tangent function not present in picoC but useful for trigonometric calculations

Examples

EXAMPLES

To access variables from your CScript file/script use the following commands:

```
int value = getVariable("variable_name")
```

Or to create a new variable use

```
setVariable("variable_name", variable_value);
```

Once a new variable is created this variable becomes available to control functions (such as in the SSC module) that can be used to automatically control a servo/motor.

You can also get variable arrays using

```
int list[256]; int num = GetArrayVariable("variable_array_name", list, 256);
```

or get individual entries in an array using

```
int val = getVariable("variable_array_name:2");
```

which would return the second entry in an array.

To change a variable's value in X milliseconds you can use

```
setTimedVariable("variable_name", variable_value, timeout_in_milliseconds);
```

which is very useful for resetting motor values after a period of time.

For example the following program (assuming you have added the [Center Of Gravity](#) module) will map COG to servo motors:

```
int cogx = getVariable("cog_x");
int cogy = getVariable("cog_y");
int left_motor;
int right_motor;
```

```
if(cogx < 140)
{
    left_motor = 60;
    right_motor = 128;
}
else
{
    if(cogx > 180)
    {
        left_motor = 128;
        right_motor = 60;
    }
    else
    {
        left_motor = 128;
        right_motor = 128;
    }
}
```

```
setVariable("left_motor", left_motor);
setVariable("right_motor", right_motor);
```

Note that the COGX values range from 0 to 320. COGY ranges from 0 to 160. These values are bounded by the current image size. These values may change if you crop, shrink, etc. the image dimensions.

You could then use a module like the [SSC](#) module to map the "left_motor" and "right_motor" variables to actual servos.

You can also process the image pixels directly using CScript. This is useful for prototyping but not recommended due to performance reasons for actual usage. The CScript module supports two routines to get and set the image pixels. Following is the example SwapColor module but in CScript.

```
int width, height;
```

```
int i;
unsigned char c;
unsigned char *pixels = getPixels(&width, &height);
```

```
for (i=0;i<(width*height*3);i+=3)
{
    c = pixels[i];
    pixels[i]=pixels[i+2];
    pixels[i+2]=c;
}
```

```
setPixels(pixels, width, height);
```

If you need to wait for a specific condition in a loop you will have to setup a trigger variable. You can't really sleep, wait or loop in the CScript module as that would stop all image processing while it is waiting for the loop or sleep to finish. As the pipeline is essentially an infinite loop (i.e. grab an image, process it, then grab another, etc) adding a sleep would cease all execution which is usually not desired. If you prevent the ending of

the loop by putting a while loop in the code it will not allow the system to capture new images and continue processing. So instead you can use the SetTimedVariable to cause the variable to change value after X number of milliseconds. Your script would look something like

```
int period = 2000;
int step = getVariable("step");
if (step == 0)
{
    setVariable("data", 2);
    setVariable("step", 1);
    setTimedVariable("step", 2, period);
}
else
{
    if (step == 2)
    {
        setVariable("data", 4);
        setVariable("step", 3);
        setTimedVariable("step", 0, period);
    }
}
```

which would cause the variable data to oscillate between 2 and 4 with a period of 2000 seconds. Note that usage of step and setting it to 1 and 3 which are dummy steps where nothing is done.

The list of supported intrinsic C functions are:

```
void printf(char *, ...);
char *sprintf(char *, char *, ...);
void exit();
float atan2(float, float);
float sin(float);
float cos(float);
float tan(float);
float asin(float);
float acos(float);
float atan(float);
float sinh(float);
float cosh(float);
float tanh(float);
float exp(float);
float fabs(float);
float log(float);
float log10(float);
float pow(float, float);
float sqrt(float);
float round(float);
float ceil(float);
float floor(float);
void *malloc(int);
void *calloc(int, int);
void *realloc(void *, int);
void free(void *)
```



```

void strcpy(char *,char *);
void strncpy(char *,char *,int);
int strcmp(char *,char *);
int strncmp(char *,char *,int);
void strcat(char *,char *);
char *index(char *,int);
char *rindex(char *,int);
int strlen(char *);
void memset(void *,int,int);
void memcpy(void *,void *,int);
int memcmp(void *,void *,int);

```

Variables

On each successful run of the script the SCRIPT_COUNT variable is incremented. Thus you can use this variable to indicate a first run state where variables can be initialized to default values. Note that the Reload and Run button will reset this count to 0.

See Also

[VBScript Program](#)
[JScript Program](#)
[Python Program](#)

JScript

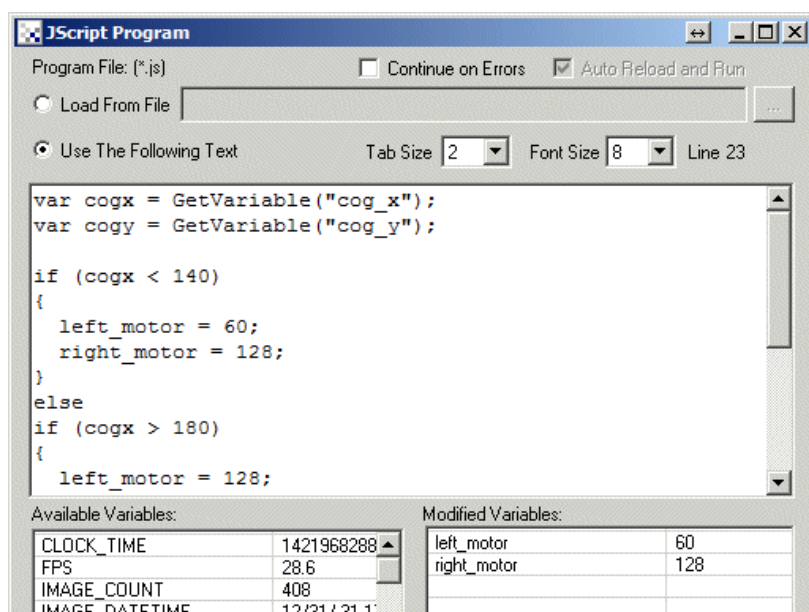
Please be sure to read the [Plugins](#) page to get a general sense of if the JScript module is a good fit for your project.

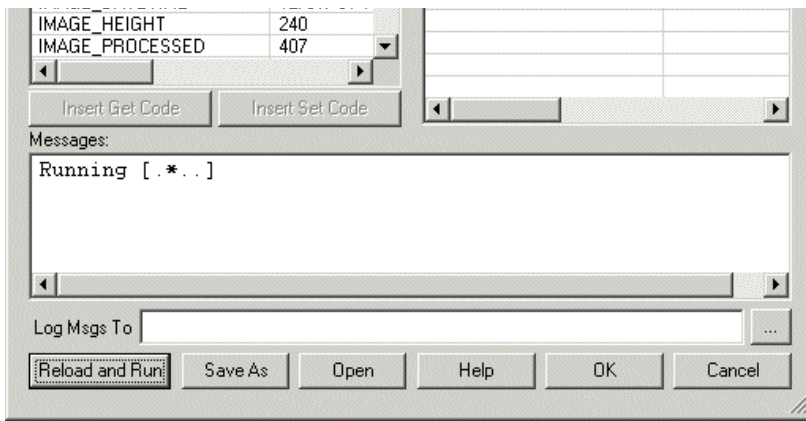
The JScript module provides a way to create custom Javascript scripts that can be used to process image statistics and map then toward servo/motor values. This module is intended to be used as a way to quickly perform custom operations without needing to implement a Plugin or use the API which typically require external tools.

The interface allows two methods for specifying your code. One technique requires that you specify a text file as the source of the program. You can use your favorite text editor to edit/create this file. The file should be a regular text file containing JScript commands/functions/operators. The program interface displays the current file being used, available system or user variables (and their values) and any messages the program produces.

The alternate way is to include the JScript text within the provided text box (see below). Using this technique allows you to include the JScript code within the .robo program and not require another user to save the code and reference a text file as needed by the above file based technique. Using the text box provides a quick way to enter in text but is NOT meant as a replacement for a full featured text editor as its editing features are very basic.

Interface





RoboRealm Specific Functions

- **VariableExists**("variable_name") - returns true or false depending on if the specified variable exists in RoboRealm. Several of the variable return functions below will return a valid value regardless of if the variable exists or not. You can use this function to determine if the value returned from one of these functions is a valid value.
- **GetVariable**("variable_name") - returns an integer value of the specified variable. Zero is returned when the variable does not exist.
- **GetFloatVariable**("variable_name") - returns the float value of the specified variable. A zero is returned when the variable does not exist.
- **GetStrVariable**("variable_name") - returns the string value of the specified variable. An empty string "" is returned when the variable does not exist.
- **GetArrayVariable**("array_variable_name") - returns the array associated with the specified variable. An array is ALWAYS returned but the length (ubound) will be zero if the array does not exist.
- **SetArrayVariable**("array_variable_name", array_variable) - sets an array associated with the specified variable name; for example

```
ReDim points(4)
```

```
points(0) = 100
```

```
points(1) = 100
```

```
points(2) = 150
```

```
points(3) = 200
```

```
SetArrayVariable "waypoints", points
```

- **SetVariable** "variable_name", variable_value - sets the variable value to the specified variable
- **SetTimedVariable** "variable_name", variable_value, timeout - sets the variable value to the specified variable after the timeout (milliseconds) has passed. This allows you to specify when in time a variable's value should be changed. Note that each call to SetTimedVariable will reset the timer so be sure to only call it ONCE to allow the time to lapse and set the specified value. This function is useful when you need an alternative to the Sleep function. By setting a variable to a value in the future you can emulate a Sleep function by allowing something to happen for a period of time, but then switching it off after the time has lapsed. So instead of using

```
SetVariable("motor", 100);
```

```
Sleep(1000)
```

```
SetVariable("motor", 0);
```

you could use

```
SetVariable("motor", 100);
```

```
SetTimedVariable("motor", 0, 1000);
```

- **AddTimedVariable** "variable_name", variable_value, timeout - similar to SetTimedVariable, this function will instead add the timing parameter to a queue instead of overwriting the timing information. For example:

```
SetTimedVariable("motor", 100, 1000);
SetTimedVariable("motor", 200, 1500);
SetTimedVariable("motor", 300, 2000);
```

will result in the motor variable being set to 300 in 2 seconds (2000 ms). What happens is that the first two calls to SetTimedVariable are **OVERWRITTEN** by the third. Instead, if you use

```
AddTimedVariable("motor", 100, 1000);
AddTimedVariable("motor", 200, 1500);
AddTimedVariable("motor", 300, 2000);
```

the motor variable will be set to 100 in 1 second, 200 in 1.5 seconds and finally 300 in 2 seconds. The AddTimedVariable will add each call into a list that is executed in sequence based on the specified time.

- **Sleep(X)** - suspends processing for X number of milliseconds and returns control to the pipeline to acquire additional images and send values to external devices.
- **WaitVariable("variable_name", variable_value)** - suspends processing until the specified variable equals the specified value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the variable value.
- **WaitVariableChange("variable_name")** - suspends processing until the specified variable changes its value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the value to change.
- **WaitImage()** - suspends processing and returns control to the pipeline for one pipeline iteration. This allows for the acquisition of additional images and to send values to external devices.
- **GetPixels()** - returns an array of RGB values of the current image
- **SetPixels(pixel_data_array)** - sets the current image to the specified pixel RGB data
- **SetParameter("module_name", count, "param_name", "param_value")** - changes the specified parameter in the specified module to the specified value. This is useful when a GUI dialog does not have a variable selection as part of the dropdown. Avoid using this function unless necessary as it is slow to update the GUI. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **GetParameter("module_name", count, "param_name")** - returns the specified parameter in the specified module. This is useful when a GUI dialog does not expose a value within a variable.
- **GetFloatParameter("module_name", count, "param_name")** - the float version of the above routine.
- **GetStrParameter("module_name", count, "param_name")** - the string version of the above routine.
- **Write("text")** - sends text to the module message output area in the dialog interface (used for debugging)
- **StopProcessing()** - stops RoboRealm from continuing to process the images with the remaining modules
- **CameraOff()** - switches off the image capturing from the current camera
- **CameraOn()** - switches on the image capturing using the current camera
- **PushButton("module_name", count, "Button")** - automates pushing a button in one of the RoboRealm GUI windows. For example:

```
if (GetStrVariable("test") != "pushed")
{
    SetVariable("test", "pushed");
    PushButton("Read_AVI", 0, "Start");
}
```

will cause the Read_AVI module (assuming one exists in the pipeline) to start playback.

```
PushButton("RoboRealm", 0, "Snap");
```

will cause RoboRealm to create a snapshot of the current image

will cause RoboRealm to create a snapshot of the current image.

- **atan2(x,y)** - arc tangent function not present in JScript but useful for trigonometric calculations
- **sqrt(x)** - square root function not present in JScript but useful for trigonometric calculations

Examples

To access variables from your JScript file/script use the following commands:

```
MyVar = GetVariable("variable_name")
```

Or to create a new variable use

```
SetVariable("variable_name", variable_value)
```

Once a new variable is created this variable becomes available to control functions (such as in the SSC module) that can be used to automatically control a servo/motor.

You can also get variable arrays using

```
MyVar = GetArrayVariable("variable_array_name")
```

or get individual entries in an array using

```
MyVar = GetVariable("variable_array_name:2")
```

which would return the second entry in an array.

To change a variable's value in X milliseconds you can use

```
SetTimedVariable("variable_name", variable_value, timeout_in_milliseconds)
```

which is very useful for resetting motor values after a period of time.

All regular JS Script commands and operators are available for use in the JScript file. See [Microsoft Scripting](#) for complete documentation on how to use these commands or view the sample files for examples.

For example the following program (assuming you have added the [Center Of Gravity](#) module) will map COG to servo motors:

```
var cogx = GetVariable("cog_x");  
var cogy = GetVariable("cog_y");
```

```
if(cogx < 140)  
{  
  left_motor = 60;  
  right_motor = 128;  
}  
else  
if(cogx > 180)  
{  
  left_motor = 128;  
  right_motor = 60;  
}  
else  
{  
  left_motor = 128;  
  right_motor = 128;  
}
```

```
SetVariable("left_motor", left_motor);  
SetVariable("right_motor", right_motor);
```

Note that the COGX values range from 0 to 320. COGY ranges from 0 to 160. These values are bounded by the current image size. These values may change if you crop, shrink, etc. the image dimensions.

You could then use a module like the [SSC](#) module to map the "left_motor" and "right_motor" variables to actual servos.

You can also process the image pixels directly using JScript. This is useful for prototyping but not recommended due to performance reasons for actual usage. The JScript module supports two routines to get and set the image pixels. Following is the example SwapColor module but in JScript.

```
var pixels
```

```
pixels = GetPixels();
```

```
for (i=0;i<(320*240*3)-1;i+=3)
{
    tmp = pixels[i]
    pixels[i] = pixels[i+2];
    pixels[i+2] = tmp;
}
```

```
SetPixels(pixels);
```

To write variables or debug statements to a log file you can use

```
var fso = new ActiveXObject("Scripting.FileSystemObject");
var fi = fso.OpenTextFile("test.txt", 8, true);
fi.WriteLine(GetVariable("cog_x")+" "+GetVariable("cog_y"));
fi.close();
```

Note that JScript can access external activeX objects such as the Microsoft XML Http object. The following example shows this technique to issue a custom URL to a remote machine. Be sure to also have a look at the [HTTP module](#) for such tasks.

```
// Show Status Output 1
```

```
var objHTTP = new ActiveXObject("MSXML2.ServerXMLHTTP");
objHTTP.open("GET", "http://172.17.42.54/axis-cgi/io/output.cgi?check=1");
objHTTP.send("");
```

```
// Activate Output 1
```

```
var objHTTP = new ActiveXObject("MSXML2.ServerXMLHTTP");
objHTTP.open("GET", "http://172.17.42.54/axis-cgi/io/output.cgi?action=1:");
objHTTP.send("");
```

```
// Deactivate Output 1
```

```
var objHTTP = new ActiveXObject("MSXML2.ServerXMLHTTP");
objHTTP.open("GET", "http://172.17.42.54/axis-cgi/io/output.cgi?action=1:\");
objHTTP.send("");
```

Variables

On each successful run of the script the SCRIPT_COUNT variable is incremented. Thus you can use this variable to indicate a first run state where variables can be initialized to default values. Note that the Reload and Run button will reset this count to 0.

Downloads

If RoboRealm complains about missing components you may need to [download the Microsoft Script Control](#).

See Also

[VBScript Program](#)

[CScript Program](#)

[Python Program](#)

Python Script

Please be sure to read the [Plugins](#) page to get a general sense of if the Python module is a good fit for your project.

The Python module provides a way to create custom Python scripts that can be used to process image statistics and map then toward servo/motor values. This module is intended to be used as a way to quickly perform custom operations without needing to implement a Plugin or use the API which typically require external tools.

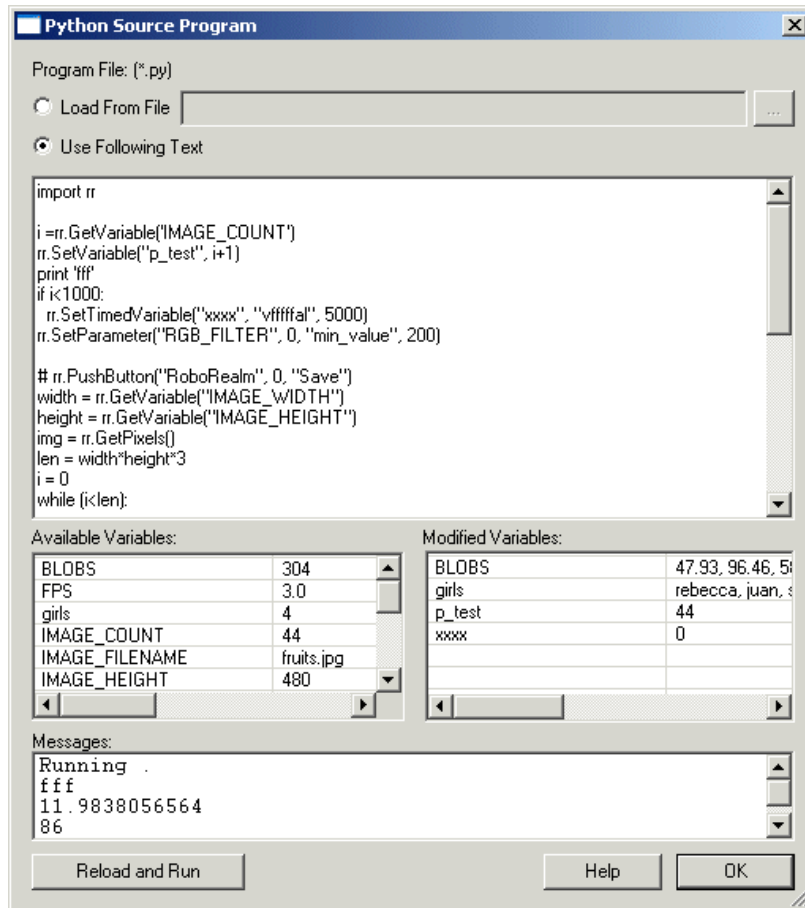
The interface allows two methods for specifying your code. One technique requires that you specify a text file as the source of the program. You

can use your favorite text editor to edit/create this file. The file should be a regular text file containing Python commands/functions/operators. The program interface displays the current file being used, available system or user variables (and their values) and any messages the program produces.

The alternate way is to include the Python text within the provided text box (see below). Using this technique allows you to include the Python code within the .robo program and not require another user to save the code and reference a text file as needed by the above file based technique. Using the text box provides a quick way to enter in text but is NOT meant as a replacement for a full featured text editor as its editing features are very basic.

Note that in any Python script you MUST "import rr" before you can use any of the embedded Python <-> RoboRealm specific functions.

Interface



RoboRealm Specific Functions

- **Print("text")** - sends text to the Python module message output area in the dialog interface used for debugging. Note that lack of "rr." as this function is global.
- **rr.VariableExists("variable_name")** - returns true or false depending on if the specified variable exists in RoboRealm. Several of the variable return functions below will return a valid value regardless of if the variable exists or not. You can use this function to determine if the value returned from one of these functions is a valid value.
- **rr.GetVariable("variable_name")** - returns an integer value of the specified variable
- **rr.GetStrVariable("variable_name")** - returns the string value of the specified variable
- **rr.GetArrayVariable("array_variable_name")** - returns the array associated with the specified variable
- **rr.SetArrayVariable "array_variable_name", array_variable** - sets an array associated with the specified variable name; for example

```
import rr

points = [100,100,150,200]

rr.SetArrayVariable("waypoints", points)
```

- **rr.SetVariable("variable_name", variable_value)** - sets the variable value to the specified variable
- **rr.SetTimedVariable("variable_name", variable_value, timeout)** - sets the variable value to the specified variable after the timeout (milliseconds) has passed. This allows you to specify when in time a variable's value should be changed. Note that each call to

SetTimedVariable will reset the timer so be sure to only call it once to allow the time to lapse and set the specified value. This function is useful when you need an alternative to the Sleep function. By setting a variable to a value in the future you can emulate a Sleep function by allowing something to happen for a period of time, but then switching it off after the time has lapsed. So instead of using

```
rr.SetVariable("motor", 100);  
rr.Sleep(1000);  
rr.SetVariable("motor", 0);
```

you could use

```
rr.SetVariable("motor", 100);  
rr.SetTimedVariable("motor", 0, 1000);
```

- **rr.AddTimedVariable("variable_name", variable_value, timeout)** - similar to SetTimedVariable, this function will instead add the timing parameter to a queue instead of overwriting the timing information. For example:

```
rr.SetTimedVariable("motor", 100, 1000);  
rr.SetTimedVariable("motor", 200, 1500);  
rr.SetTimedVariable("motor", 300, 2000);
```

will result in the motor variable being set to 300 in 2 seconds (2000 ms). What happens is that the first two calls to SetTimedVariable are **OVERWRITTEN** by the third. Instead, if you use

```
rr.AddTimedVariable("motor", 100, 1000);  
rr.AddTimedVariable("motor", 200, 1500);  
rr.AddTimedVariable("motor", 300, 2000);
```

the motor variable will be set to 100 in 1 second, 200 in 1.5 seconds and finally 300 in 2 seconds. The AddTimedVariable will add each call into a list that is executed in sequence based on the specified time.

- **rr.Sleep(X)** - suspends processing for X number of milliseconds and returns control to the pipeline to acquire additional images and send values to external devices.
- **rr.WaitVariable("variable_name", variable_value)** - suspends processing until the specified variable equals the specified value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the variable value.
- **rr.WaitVariableChange("variable_name")** - suspends processing until the specified variable changes its value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the value to change.
- **rr.WaitImage()** - suspends processing and returns control to the pipeline for one pipeline iteration. This allows for the acquisition of additional images and to send values to external devices.
- **rr.GetPixels()** - returns an array of RGB values of the current image
- **rr.SetPixels(pixel_data_array)** - sets the current image to the specified pixel RGB data
- **rr.SetParameter("module_name", count, "param_name", "param_value")** - changes the specified parameter in the specified module to the specified value. This is useful when a GUI dialog does not have a variable selection as part of the dropdown. Avoid using this function unless necessary as it is slow to update the GUI. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **rr.GetParameter("module_name", count, "param_name")** - returns the specified parameter in the specified module. This is useful when a GUI dialog does not expose a value within a variable. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **rr.GetStrParameter("module_name", index, "param_name")** - the string version of the above routine.
- **rr.StopProcessing()** - stops RoboRealm from continuing to process the images with the remaining modules
- **rr.CameraOff()** - switches off the image capturing from the current camera
- **rr.CameraOn()** - switches on the image capturing using the current camera

- **rr.PushButton("Module_Name", Module_Index, "Button")** - automates pushing a button in one of the RoboRealm GUI windows. For example:

```
import rr
if rr.GetStrVariable("test") <> "pushed":
    rr.SetVariable("test", "pushed")
    rr.PushButton("Read_AVI", 0, "Start")
```

will cause the Read_AVI module (assuming one exists in the pipeline) to start playback.

```
import rr
rr.PushButton("RoboRealm", 0, "Snap")
```

will cause RoboRealm to create a snapshot of the current image.

Examples

To access variables from your Python file/script use the following commands:

```
MyVar = rr.GetVariable("variable_name")
```

Or to create a new variable use

```
rr.SetVariable("variable_name", variable_value)
```

Once a new variable is created this variable becomes available to control functions (such as in the SSC module) that can be used to automatically control a servo/motor.

You can also get variable arrays using

```
MyVar = rr.GetArrayVariable("variable_array_name")
```

or get individual entries in an array using

```
MyVar = rr.GetVariable("variable_array_name:2")
```

which would return the second entry in an array.

To change a variable's value in X milliseconds you can use

```
rr.SetTimedVariable("variable_name", variable_value, timeout_in__milliseconds)
```

which is very useful for resetting motor values after a period of time.

All regular Python commands and operators are available for use in the file. See [Official Python Website](#) for complete documentation on how to use Python commands.

For example the following program (assuming you have added the [Center Of Gravity](#) module) will map COG to servo motors:

```
import rr
```

```
cogx = rr.GetVariable("cog_x")
cogy = rr.GetVariable("cog_y")
```

```
if cogx < 140:
```

```
    left_motor = 60
    right_motor = 128
```

```
else:
```

```
    if cogx > 180:
        left_motor = 128
        right_motor = 60
```

```
    else:
```

```
        left_motor = 128
        right_motor = 128
```

```
rr.SetVariable("left_motor", left_motor)
```



```
rr.SetVariable("right_motor", right_motor)
```

Note that the COGX values range from 0 to 320. COGY ranges from 0 to 160. These values are bounded by the current image size. These values may change if you crop, shrink, etc. the image dimensions.

You could then use a module like the [SSC](#) module to map the "left_motor" and "right_motor" variables to actual servos.

You can also process the image pixels directly using Python. This is useful for prototyping but not recommended due to performance reasons for actual usage. The Python module supports two routines to get and set the image pixels. Following is the example SwapColor module but in Python.

```
import rr
```

```
width = rr.GetVariable("IMAGE_WIDTH")
height = rr.GetVariable("IMAGE_HEIGHT")
```

```
img = rr.GetPixels()
len = width*height*3
i = 0
while (i<len):
    t = img[i]
    img[i] = img[i+2]
    img[i+2] = t
    i+=3
rr.SetPixels(img, width, height)
```

To write variables or debug statements to a log file you can use

```
import rr
```

```
f = open('c:\\temp\\test.txt', 'a')
f.write(rr.GetStrVariable("cog_x") + '!' + rr.GetStrVariable("cog_y") + '\n')
f.close()
```

Variables

On each successful run of the script the SCRIPT_COUNT variable is incremented. Thus you can use this variable to indicate a first run state where variables can be initialized to default values. Note that the Reload and Run button will reset this count to 0.

Downloads

If RoboRealm complains about missing components you may need to [download Python](#).

See Also

[JScript Program](#)

[VBScript Program](#)

[CScript Program](#)

VBScript

Please be sure to read the [Plugins](#) page to get a general sense of if the VBScript module is a good fit for your project.

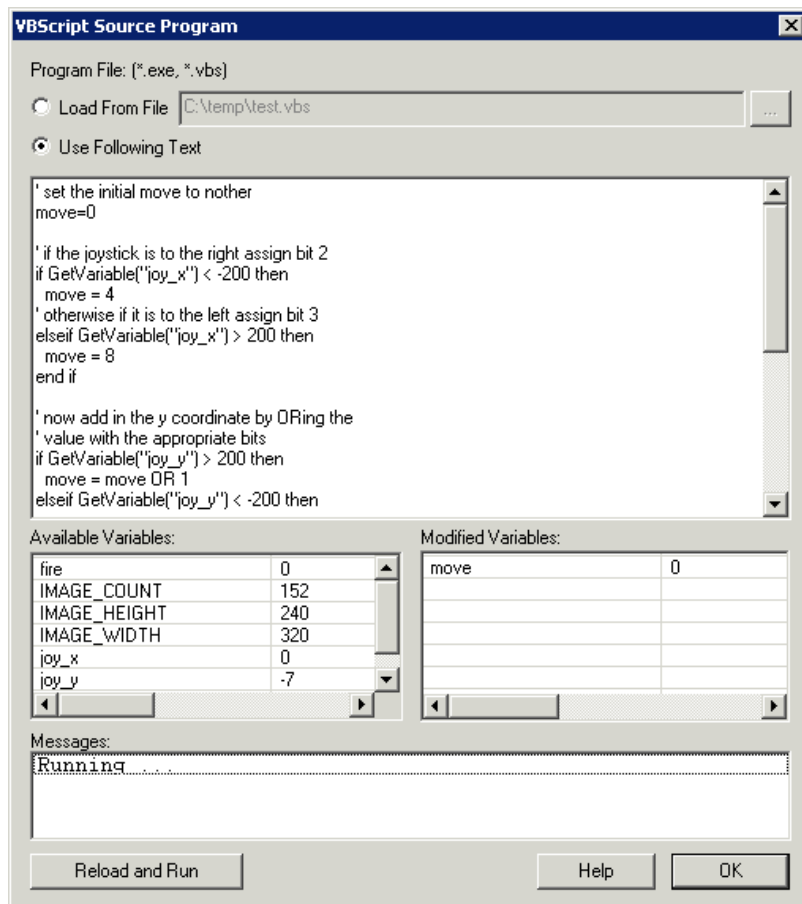
The VBScript module provides a way to create custom Visual Basic scripts that can be used to process image statistics and map them toward servo/motor values. This module is intended to be used as a way to quickly perform custom operations without needing to implement a Plugin or use the API which typically require external tools.

The interface allows two methods for specifying your code. One technique requires that you specify a text file as the source of the program. You can use your favorite text editor to edit/create this file. The file should be a regular text file containing VBScript commands/functions/operators. The program interface displays the current file being used, available system or user variables (and their values) and any messages the program produces.

The alternate way is to include the VBScript text within the provided text box (see below). Using this technique allows you to include the VBScript code within the .robo program and not require another user to save the code and reference a text file as needed by the above file based technique. Using the text box provides a quick way to enter in text but is NOT meant as a replacement for a full featured text editor as its editing features are

very basic.

Interface



RoboRealm Specific Functions

- **VariableExists**("variable_name") - returns true or false depending on if the specified variable exists in RoboRealm. Several of the variable return functions below will return a valid value regardless of if the variable exists or not. You can use this function to determine if the value returned from one of these functions is a valid value.
- **GetVariable**("variable_name") - returns an integer value of the specified variable. Zero is returned when the variable does not exist.
- **GetFloatVariable**("variable_name") - returns the float value of the specified variable. A zero is returned when the variable does not exist.
- **GetStrVariable**("variable_name") - returns the string value of the specified variable. An empty string "" is returned when the variable does not exist.
- **GetArrayVariable**("array_variable_name") - returns the array associated with the specified variable. An array is ALWAYS returned but the length (ubound) will be zero if the array does not exist.
- **SetArrayVariable** "array_variable_name", array_variable - sets an array associated with the specified variable name; for example

```
ReDim points(4)
```

```
points(0) = 100
```

```
points(1) = 100
```

```
points(2) = 150
```

```
points(3) = 200
```

```
SetArrayVariable "waypoints", points
```

- **SetVariable** "variable_name", variable_value - sets the variable value to the specified variable
- **SetTimedVariable** "variable_name", variable_value, timeout - sets the variable value to the specified variable after the timeout (milliseconds) has passed. This allows you to specify when in time a variable's value should be changed. Note that each call to SetTimedVariable will reset the timer so be sure to only call it ONCE to allow the time to lapse and set the specified value. This function is useful when you need an alternative to the Sleep function. By setting a variable to a value in the future you can emulate a Sleep function by allowing something to happen for a period of time, but then switching it off after the time has lapsed. So instead of using

```
SetVariable "motor", 100
Sleep(1000)
SetVariable "motor", 0
```

you could use

```
SetVariable "motor", 100
SetTimedVariable "motor", 0, 1000
```

- **AddTimedVariable** "variable_name", variable_value, timeout - similar to SetTimedVariable, this function will instead add the timing parameter to a queue instead of overwriting the timing information. For example:

```
SetTimedVariable "motor", 100, 1000
SetTimedVariable "motor", 200, 1500
SetTimedVariable "motor", 300, 2000
```

will result in the motor variable being set to 300 in 2 seconds (2000 ms). What happens is that the first two calls to SetTimedVariable are OVERWRITTEN by the third. Instead, if you use

```
AddTimedVariable "motor", 100, 1000
AddTimedVariable "motor", 200, 1500
AddTimedVariable "motor", 300, 2000
```

the motor variable will be set to 100 in 1 second, 200 in 1.5 seconds and finally 300 in 2 seconds. The AddTimedVariable will add each call into a list that is executed in sequence based on the specified time.

- **Sleep X** - suspends processing for X number of milliseconds and returns control to the pipeline to acquire additional images and send values to external devices.
- **WaitVariable** "variable_name", variable_value - suspends processing until the specified variable equals the specified value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the variable value.
- **WaitVariableChange** "variable_name" - suspends processing until the specified variable changes its value. This function returns control to the pipeline to acquire additional images and send values to external devices while it waits for the value to change.
- **WaitImage** - suspends processing and returns control to the pipeline for one pipeline iteration. This allows for the acquisition of additional images and to send values to external devices.
- **GetPixels** - returns an array of RGB values of the current image
- **SetPixels** pixel_data_array - sets the current image to the specified pixel RGB data
- **SetParameter** "module_name", count, "param_name", "param_value" - changes the specified parameter in the specified module to the specified value. This is useful when a GUI dialog does not have a variable selection as part of the dropdown. Avoid using this function unless necessary as it is slow to update the GUI. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **GetParameter**("module_name", count, "param_name") - returns the specified parameter in the specified module. This is useful when a GUI dialog does not expose a value within a variable. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **GetFloatParameter**("module_name", count, "param_name") - the float version of the above routine. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).

- **GetStrParameter**("module_name", count, "param_name") - the string version of the above routine. Count is used to distinguish between multiple uses of the same module (i.e. module_name would be the same for either module).
- **Write** "text" - sends text to the module message output area in the dialog interface (used for debugging)
- **StopProcessing** - stops RoboRealm from continuing to process the images with the remaining modules
- **CameraOff** - switches off the image capturing from the current camera
- **CameraOn** - switches on the image capturing using the current camera
- **PushButton** "module_name", count, "button" - automates pushing a button in one of the RoboRealm GUI windows. For example:

```
if GetStrVariable("test") <> "pushed" then
    SetVariable "test", "pushed"
    PushButton "Read_AVI", 0, "Start"
end if
```

will cause the Read_AVI module (assuming one exists in the pipeline) to start playback.

```
PushButton "RoboRealm", 0, "Snap"
```

will cause RoboRealm to create a snapshot of the current image.

- **atan2**(x,y) - arc tangent function not present in VBScript but useful for trigonometric calculations
- **sqr**t(x) - square root function not present in VBScript but useful for trigonometric calculations

Examples

To access variables from your VBScript file/script use the following commands:

```
MyVar = GetVariable("variable_name")
```

Or to create a new variable use

```
SetVariable "variable_name", variable_value
```

Once a new variable is created this variable becomes available to control functions (such as in the SSC module) that can be used to automatically control a servo/motor.

You can also get variable arrays using

```
MyVar = GetArrayVariable("variable_array_name")
```

or get individual entries in an array using

```
MyVar = GetVariable("variable_array_name:2")
```

which would return the second entry in an array.

To change a variable's value in X milliseconds you can use

```
SetTimedVariable "variable_name", variable_value, timeout_in_milliseconds
```

which is very useful for resetting motor values after a period of time.

All regular VB Script commands and operators are available for use in the VBScript file. See [Microsoft Scripting](#) for complete documentation on how to use these commands or view the sample files for examples.

For example the following program (assuming you have added the [Center Of Gravity](#) module) will map COG to servo motors:

```
cogx = GetVariable("cog_x")
cogy = GetVariable("cog_y")
```

```
if cogx < 140 then
    left_motor = 60
    right_motor = 128
```

```

else
if cogx > 180 then
left_motor = 128

right_motor = 60
else
left_motor = 128
right_motor = 128
end if
end if

```

```

SetVariable "left_motor", left_motor
SetVariable "right_motor", right_motor

```

Note that the COGX values range from 0 to 320. COGY ranges from 0 to 160. These values are bounded by the current image size. These values may change if you crop, shrink, etc. the image dimensions.

You could then use a module like the [SSC](#) module to map the "left_motor" and "right_motor" variables to actual servos.

You can also process the image pixels directly using VBScript. This is useful for prototyping but not recommended due to performance reasons for actual usage. The VBScript module supports two routines to get and set the image pixels. Following is the example SwapColor module but in VBScript.

```

Dim pixels

```

```

pixels = GetPixels()

```

```

for i = 0 to (320*240*3)-1 step 3

```

```

tmp = pixels(i)
pixels(i) = pixels(i+2)
pixels(i+2) = tmp

```

```

next

```

```

SetPixels(pixels)

```

To write variables or debug statements to a log file you can use

```

Dim fi
Dim fso
set fso = CreateObject("Scripting.FileSystemObject")
set fi = fso.OpenTextFile("c:\temp\test.txt", 8, true)
if err.number = 0 then
fi.WriteLine GetVariable("cog_x") & "!" & _
GetVariable("cog_y")
fi.close
end if
set fi = nothing
set fso = nothing

```

Note that VBScript can access external activeX objects such as the Microsoft XML Http object. The following example shows this technique to issue a custom URL to a remote machine. Be sure to also have a look at the [HTTP module](#) for such tasks.

```

' Show Status Output 1
Dim a
Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")
objHTTP.Open "GET", "http://172.17.42.54/axis-cgi/io/output.cgi?check=1", False
objHTTP.send ("")

a = Split(objHTTP.responseText, "=")
SetVariable("Output1"),a(1)

' Activate Output 1
Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")
objHTTP.Open "GET", "http://172.17.42.54/axis-cgi/io/output.cgi?action=1:", False
objHTTP.send ("")

' Deactivate Output 1
Set objHTTP = CreateObject("MSXML2.ServerXMLHTTP")

```

```
objHTTP.Open "GET", "http://172.17.42.54/axis-cgi/io/output.cgi?action=1:\", False
objHTTP.send ("")
```

Variables

On each successful run of the script the `SCRIPT_COUNT` variable is incremented. Thus you can use this variable to indicate a first run state where variables can be initialized to default values. Note that the Reload and Run button will reset this count to 0.

Downloads

If RoboRealm complains about missing components you may need to [download the Microsoft Script Control](#).

See Also

[JScript Program](#)

[CScript Program](#)

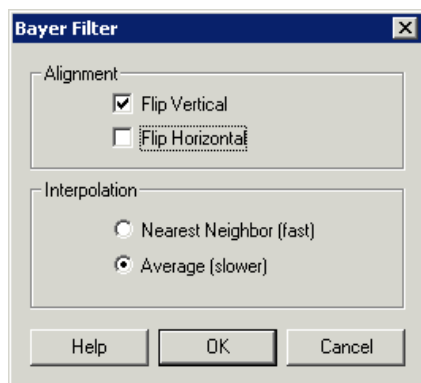
[Python Program](#)

Bayer Filter

The Bayer Filter module will rearrange the current image pixels from a Bayer format to an RGB color format. The Bayer pixel format is used by most CCD cameras to sample red, green and blue colors using different color sensors. These color sensors then produce an image that has the red, green and blue colors interlaced around each other. In order to create a reasonable color representation these single channel color pixels need to be combined in such a way as to produce a color image.

For a more in depth review of the Bayer format in addition to an interactive demo see [The Imaging Source](#).

Interface



Instructions

1. Interpolation - Select which type of interpolation mode you would like to use in creating a color image.

Nearest Neighbor - uses the nearest RGB value for each pixel

Average - uses a combined average of the nearest RGB values for each pixel

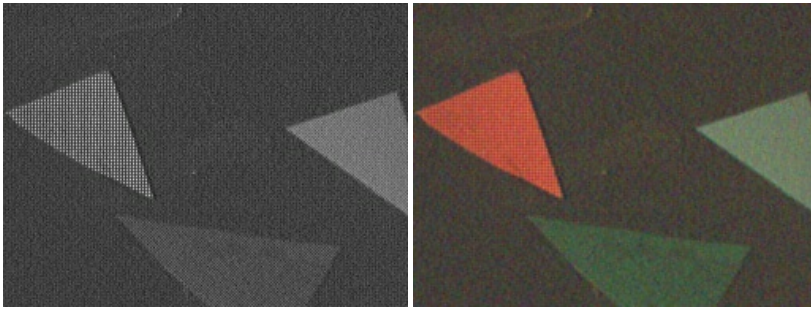
2. Flip - Mirrors the image horizontally or vertically to get the right orientation.

3. Pattern - Specify what Bayer pattern is used in the image. The first two letters represent the labels for the first two pixels on the first row. The second two letters represent the first two pixels on the second row. Note that R=red, G=green, B=blue.

Example

Bayer Source Image

Colorized



Background Removal

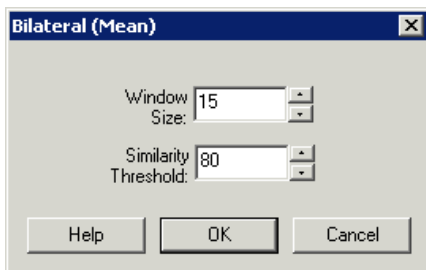
The Background Filter attempts to remove large areas of consistent texture. The window size is used to create an average pixel value (RGB) by summing all pixel values in the window and dividing by the window size. This average pixel value is then compared to the current image pixel value at the center of the window/sample area. If the image pixel value is within a specified distance to the average the pixel is assumed to be a background pixel and set to black. Those pixel values that are not within the mean distance to the average pixel is considered to be a foreground pixel and its value is untouched. The window is then shifted by one pixel and the process repeated to process the entire image.

Bilateral

The Bilateral (also known as conditional mean) filter is very similar to the [Mean filter](#) but better preserves edges while averaging/blurring other parts of the image. The filter accomplishes this task by only averaging the values around the current pixel that are close in color value to the current pixel. The 'closeness' of other neighborhood pixels to the current pixels is determined by the specified threshold. I.e. for a value of 10 each pixel that contributes to the current mean have to be within 10 values of the current pixel.

The Bilateral mean filter can be used to remove image noise but still preserve edge boundaries.

Interface



Instructions

1. Specify the window size to use for the averaging filter. The greater the window size the more blurring will occur in the image
2. Specify the 'closeness' threshold. The smaller the threshold the less distant pixels will contribute to the current mean. Larger thresholds will allow more pixels to contribute to the mean but will also start blurring over less defined edges.

Example

Source



Bilateral



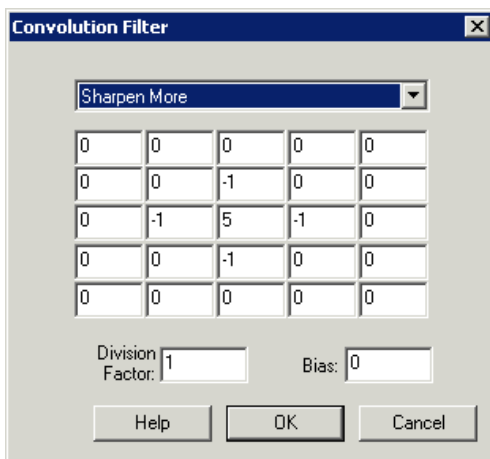
See Also

[Mean](#)
[Median](#)
[Kuwahara](#)

Convolution Filter

Convolution filters are a great way to process images for certain features. Features are defined by an n by m matrix that is applied to the image in the following way: (grayscale only for purposes of example)

Interface



Instructions

1. Kernel - Edit the 5x5 textbox grid to add in your convolution values OR
2. Dropdown - Select a pre-created filter using the dropdown menu to help you get started
3. Divisor/Bias - Specify the divisor and/or bias. The result of the kernel above will be divided by the divisor to help keep the pixel within the 0 to 255 range. The bias will be added to the result to also keep it within the positive 0 to 255 range.

Example

Source



Sharpen More



An example small grayscale image (10x10):

34	22	77	48	237	205	29	212	107	41
50	150	77	158	233	251	112	165	47	229
93	0	77	219	43	56	42	113	140	94
32	19	44	30	36	94	151	101	28	84
10	90	48	73	63	148	159	183	99	22
192	70	27	88	20	230	53	34	38	106

239	202	196	205	50	123	192	88	41	37
230	174	14	22	127	100	189	186	214	187
227	86	195	6	53	168	46	166	36	249
215	165	237	110	125	191	191	94	123	8

An example convolution filter for line detection:

-1	-1	-1
-1	8	-1
-1	-1	-1

The row=2, column=2 pixel and its neighborhood from the image above: The row=2, column=2 pixel and its neighborhood from the image above:

34	22	77
50	150	77
93	0	77

To apply the convolution filter multiply the filter values with the image data block. Work with each pixel and its 3x3 neighborhood:

-1*34	-1*22	-1*77
-1*50	8*150	-1*77
-1*93	-1*0	-1*77

Then sum all the values:

$$(-34)+(-22)+(-77)+$$

$$(-50)+(1200)+(-77)+$$

$$(-93)+(0)+(-77) = 770$$

Divide by the divisor and add the bias.

$$(770/\text{divisor})+\text{bias}=770 \text{ (in this example divisor}=1, \text{ bias}=0)$$

If the new pixel value is > 255 set it to 255

If the new pixel value is < 0 set it to 0

The new pixel value is 255. Store that in a new image:

34	22	77
50	255	77
93	0	77

Continue with all other 3x3 blocks in the image using original values. For example the next image block could be

22	77	48
150	77	158
0	77	219

Note the 3x3 "window" is shifted to the right by one and that the new pixel value is NOT used but stored as a second new image.

Most of the image is processed in this manner. Image borders create problems and are ignored.

Many other filters can easily be defined for other purposes

Blur:

1	1	1	1	2	1
1	1	1	2	4	2
1	1	1	1	2	1

Sharpen:

-1	-1	-1	0	-1	0
-1	9	-1	-1	5	-1
-1	-1	-1	0	-1	0

Edee Enhancement:

```

0 0 0    0 -1 0    -1 0 0
-1 1 0    0 1 0     0 1 0
0 0 0    0 0 0     0 0 0

```

Find Edges:

```

0 1 0    -1 -1 -1    1 -2 1
1 -4 1    -1 8 -1    -2 4 -2
0 1 0    -1 -1 -1    1 -2 1

```

Emboss:

```

-2 -1 0
-1 1 1
0 1 2

```

The convolution matrix is displayed in the convolution interface. Changing any number will alter the matrix and change the image as a result of applying that matrix.

Select a matrix from the pull down menu which will populate the matrix with those values.

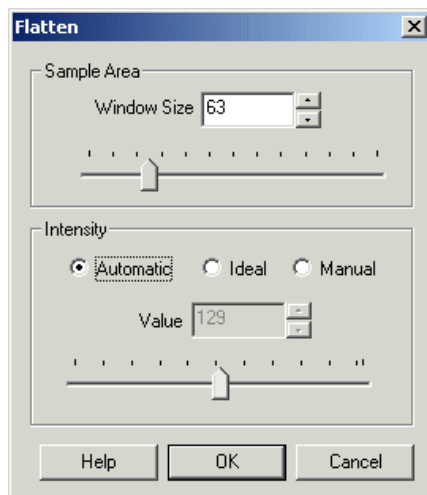
Flatten

The Flatten module (aka Pixel Division) attempts to deal with uneven lighting within an image. Images are taken in low light situations such as with flash or with cameras with uneven lighting distributions or scenes with an overhead cast shadow are uneven in lighting distribution throughout the image. This significant artifact reveals itself once you attempt to segment the image using the flood fill, threshold, RGB Filter, etc. modules in that the segmentation appears uneven across the entire image as the lighting changes.

The traditional technique used to deal with such situations require averaging over a large area of the image followed by subtraction of that average from the original image. This technique was illustrated in our [Surveyor Trail](#) following tutorial. As this technique works well for many images that have highlights or large shadow areas this technique with several refinements has been added as a distinct module.

This module is particularly useful when removing highlights or shadows. The examples below show flash light correction, shadow removal, highlight correction and another shadow removal.

Interface



Instructions

1. Sample Area - The sample area defines how much the a localized area the module should look at when determining the appropriate region intensity. Decreasing the size will reduce the image to the result of an edge detection method, whilst increasing the sample area will focus more on larger areas of the image.
2. Automatic Intensity - The automatic intensity mode will attempt to produce an image that averages in intensity at 128. This creates a intensity image that allows for good contrast.
3. Ideal Intensity - The Ideal Intensity mode will check the image for the most common intensity mode and produce a final image that is close to that intensity mode. This is useful if your image appears too dark using the automatic mode. Note that this mode can become confused and produce an overlv dark or bright image.

4. Manual Intensity - There are times when both previous intensity modes will fail. At that point you can select the manual intensity mode to produce images that appear correct.

5. Max Contrast - The module will also apply a localized contrast enhancement to pixel in order to bring out details. Sometime this adjustment can create too extreme an enhancement. The Max Contrast value can be used to restrict this contrast enhancement to something more reasonable. Note, this only happens in images which are predominately dark.

Example

Source

Light Flattened



See Also

- [Equalize](#)
- [Normalize](#)
- [Color Balance](#)

Gaussian Filter

Similar to the [mean filter](#) the Gaussian filter will smooth an image but will preserve edges better than the more basic mean filter.

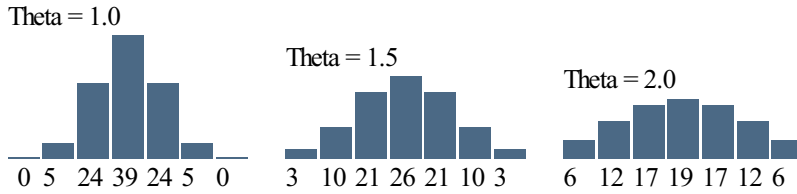
The Gaussian filter gives more weight to the current pixel position and then tapers the weights as distance increases according to the Gaussian

formula. By weighting a pixels contribution to the final pixel value this filter can better preserve edges than the mean filter which specifies equal weights to all pixels within the filter window.

The Gaussian Filter is a [convolution filter](#) whose convolution matrix is a Gaussian distribution. For a 1-D Gaussian filter the single filter values are defined as

$$G(x) = \frac{1}{\sqrt{2\pi\theta^2}} \exp\left(-\frac{x^2}{2\theta^2}\right)$$

For example the following charts represent a Gaussian filter of window size 7 and differing thetas.



Couple points to note:

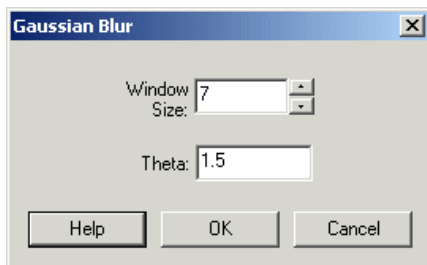
1. The values have been scaled to add to 255.
2. Below theta 1.0 the filter can be reduced to size 5 since the endpoints have fallen off to zero.
3. As theta increases the filter becomes more flatter as neighboring pixels are given more weight.

Thus, if theta is less than 0.5, the filter becomes a single point (and not very useful). Likewise, if theta is greater than the (window size)/6, then the filter becomes more like a mean filter.

This filter is applied to an image in a two phase approach. First the horizontal direction is filtered using the above filter in a similar manner to a [convolution filter](#) by taking each pixel in the image, centering the filter on that pixel (the middle value) and then multiplying the pixel values by the weight at each filter location followed by a final divide to get the resulting new pixel value. This process is then repeated vertically on the horizontally processed image to create the final image.

The Gaussian filter is one of the more popular blurring filters as it has its basis in the human visual perception system. It has been found that neurons create a similar filter when processing visual images.

Interface



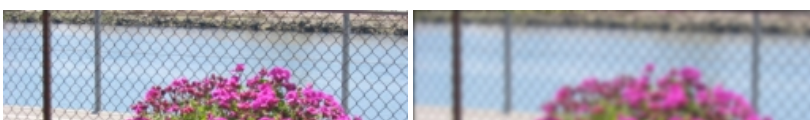
Instructions

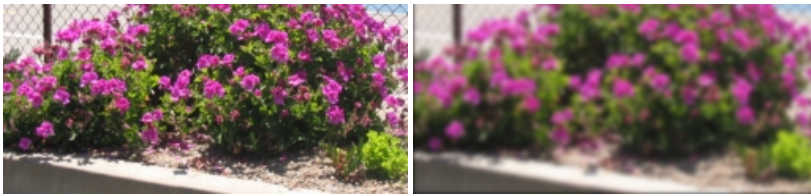
1. Window Size - Specify the window size of the filter. This is often referred to as the kernel size. Larger windows require more processing but can achieve higher levels of blurring.
2. Theta - Specify the theta or sigma of the filter. This controls the relative weights or effect the surrounding pixels have on the final pixels. Smaller theta results in more weight on the center pixel and results in less blurring, larger theta causes less weight on the center pixel and causes more blurring.
3. Grayscale - To increase speed, you can select the grayscale option which will only examine the green channel to produce the blur results.

Example

Source

Gaussian Blur





See Also

[Mean Filter](#)

[Difference of Gaussian Filter](#)

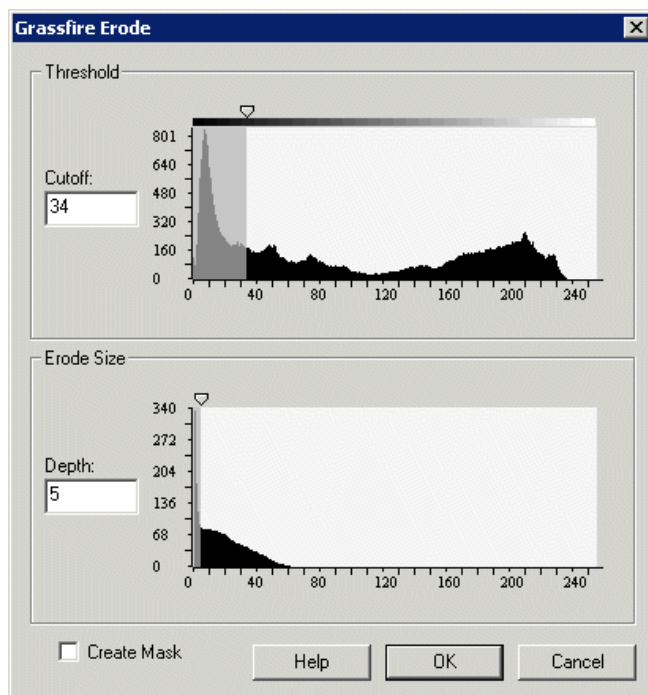
Grassfire Thinning

The Grassfire effect is used to erode images by a set amount. When processing images a lot of small artifacts (erroneous pixels) can be present that create noise in the final results. Eroding the image can remove single pixels artifacts and smooth edges of larger objects.

As the Grassfire algorithm operates on a binary image you will need to specify a threshold value before eroding can begin. The Threshold histogram seen in the Grassfire interface will cause all pixels below the threshold to black and all pixels above the threshold to white. Note that the threshold operates only on intensity (grayscale) values. To segment color values use the RGB Threshold function and then the Grassfire threshold.

Below the threshold histogram is the erosion histogram. This histogram maps the object thickness on the X axis with the number of objects of that size on the Y axis. Moving the slider will cause all objects with thickness below that size to disappear and will shrink all larger objects by that amount. For example, if the slider is set to 10 then all objects with thickness (radius) less than 10 will disappear, all objects thicker than 10 pixels will be thinned (shrunk) by 10 pixels.

Interface



Instructions

1. Specify the intensity threshold cutoff
2. Specify the erosion depth. This is the amount of erosion that remaining objects will undergo.
3. If you want to create a mask out of the remaining objects select the "Create Mask" checkbox.

Example

Source



Eroded Image





See Also

[Erode](#)
[Threshold](#)
[Auto Threshold](#)

Kuwahara Filter

The Kuwahara filter is an edge-preserving filter that softens the current image but attempts to preserve edges. Similar to the mean filter the Kuwahara filter replaces the current pixel with the mean of a neighboring 3x3 block that has the least variance.

Example

Source Image

Kuwahara



See Also

[Mean](#)

For More Information

[Image Processing Fundamentals - Smoothing Operations](#)
[NEC Incubation Center - IMAP VISION - Kuwahara Filter](#)
[Image Processing with gtuas](#)

Max Filter

The Maximum filter enhances bright values in the image by increasing its area. Similar to a dilate function each 3x3 (or other window size) is processed for the brightest surrounding pixel. That brightest pixel then becomes the new pixel value at the center of the window.

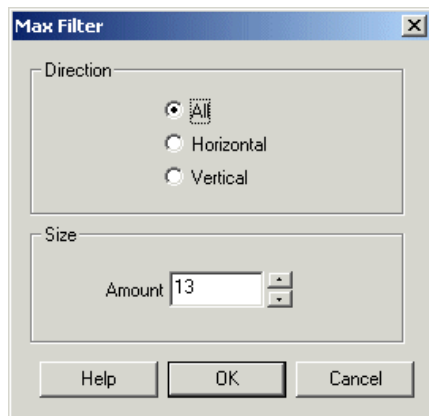
For example, given the grayscale 3x3 pixel neighborhood;

22	77	48
150	77	158
0	77	219

The center pixel would be changed from 77 to 219 as it is the brightest pixel within the current window.

Interf

Interface



Instructions

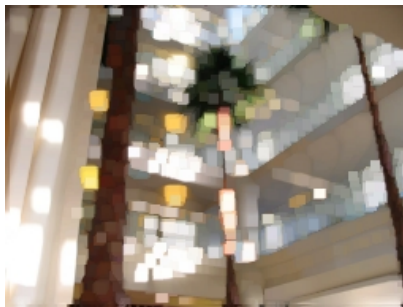
1. Direction - Select which axis you want the max filter to operate on
2. Size - Select the window size of the max filter

Example

Source



Max Filter



See Also

[Min Filter](#)

[Midpoint Filter](#)

Max - Min Filter

The "Max - Min" filter blurs the image by replacing each pixel with the difference of the highest pixel and the lowest pixel (with respect to intensity) within the specified window size.

For example, given the grayscale 3x3 pixel neighborhood;

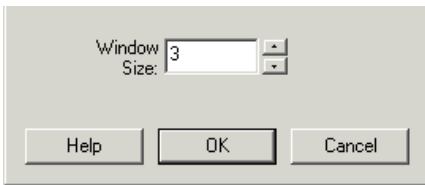
22	77	48
150	77	158
10	77	219

The center pixel would be changed from 77 to 209 as it is the difference between the brightest pixel 219 and the darkest pixel 10 within the current window.

It is interesting to note that at a small window size (2 or 3) this filter is an effective edge detection filter as well.

Interface





Instructions

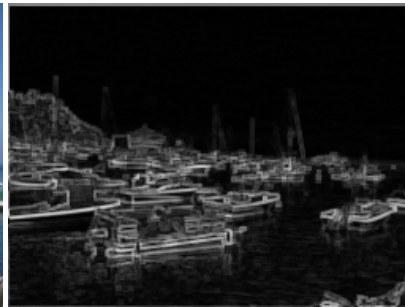
1. Select the window size of the Max - Min filter

Example

Source



Max - Min Filter (after greyscale applied)



See Also

[Midpoint Filter](#)

[Min Filter](#)

[Max Filter](#)

Mean Filter

The Mean or Average filter is used to soften an image by averaging surrounding pixel values.

For example, given the grayscale 3x3 pixel window;

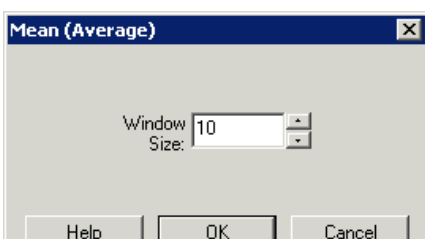
22	77	48
150	77	158
0	77	219

Center pixel = $(22+77+48+150+77+158+0+77+219)/9$

The center pixel would be changed from 77 to 92 as that is the mean value of all surrounding pixels.

This filter is often used to smooth images prior to processing. It can be used to reduce pixel flicker due to overhead fluorescent lights.

Interface



Instructions

1. Select the window size of the mean filter

Example

Source



Mean



See Also

[Median](#)

Median Filter

The Median filter is used to remove noise from an image by replacing pixels with the middle pixel value selected from a certain window size.

For example, given the grayscale 3x3 pixel window;

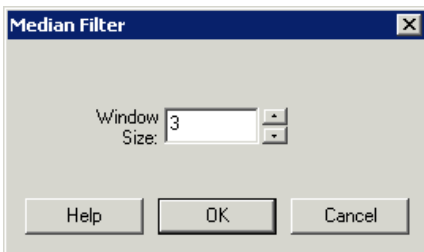
22	77	48
150	77	158
0	77	219

Pixels sorted based on intensity: 0, 22, 48, 77, [77], 77, 150, 158, 219

The center pixel would be left at 77 since 77 is the middle value of the sorted list of pixels.

The median filter is very effective at removing noise while not destroying sharp edges in an image.

Interface

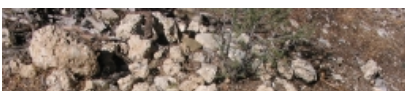


Instructions

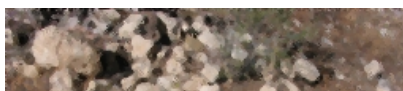
1. Select the window size of the median filter

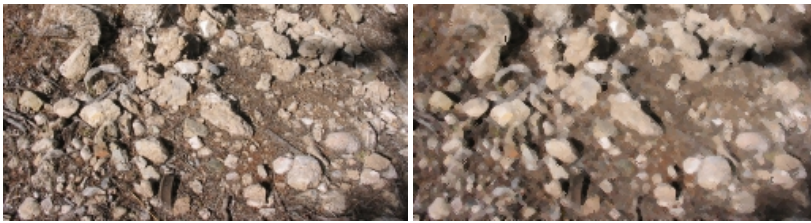
Example

Source



Median





See Also

[Mean Filter](#)

[Mode Filter](#)

Midpoint Filter

The Midpoint filter blurs the image by replacing each pixel with the average of the highest pixel and the lowest pixel (with respect to intensity) within the specified window size.

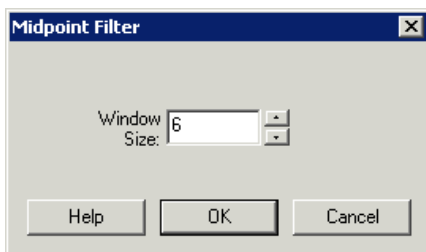
For example, given the grayscale 3x3 pixel neighborhood;

22	77	48
150	77	158
0	77	219

The center pixel would be changed from 77 to 109 as it is the midpoint between the brightest pixel 219 and the darkest pixel 0 within the current window.

Midpoint = (darkest+lightest)/2

Interface



Instructions

1. Select the window size of the midpoint filter

Example

Source

Midpoint Filter



See Also

[Min Filter](#)
[Max Filter](#)

Minimum Filter

The Minimum filter enhances dark values in the image by increasing its area. Similar to a dilate function each 3x3 (or other window size) is processed for the darkest surrounding pixel. That darkest pixel then becomes the new pixel value at the center of the window.

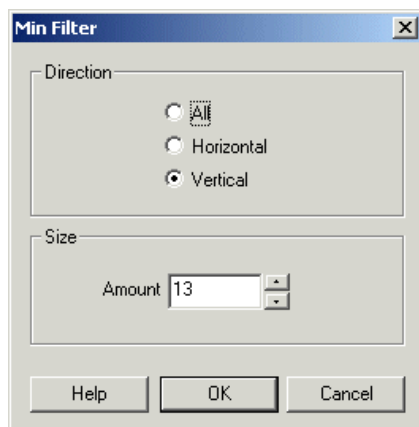
For example, given the grayscale 3x3 pixel neighborhood;

22	77	48
150	77	158
0	77	219

The center pixel would be changed from 77 to 0 as it is the darkest pixel within the current window.

Try this filter on images containing peoples faces. It can be used as a simple eye detector.

Interface



Instructions

1. Direction - Select which axis you want the min filter to operate on
2. Size - Select the window size of the min filter

Example

Source

Min Filter



See Also

[Max](#)
[Midpoint Filter](#)

Mode Filter

The Mode filter is used to remove noise from an image by replacing pixels with the most frequently occurring pixel value selected from a certain window size.

For example, given the grayscale 3x3 pixel window;

22	77	48
150	77	158
0	77	219

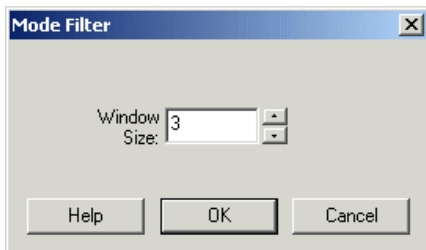
Pixels based on frequency:

1 - 0
1 - 22
1 - 48
3 - 77
1 - 150
1 - 158
1 - 219

Thus the center pixel would be left at 77 since 77 is the most frequently occurring value in the list of pixels.

The mode filter (like the median filter) is very effective at removing noise while not destroying sharp edges in an image.

Interface



Instructions

1. Select the window size of the mode filter

Example

Source

Mode



See Also

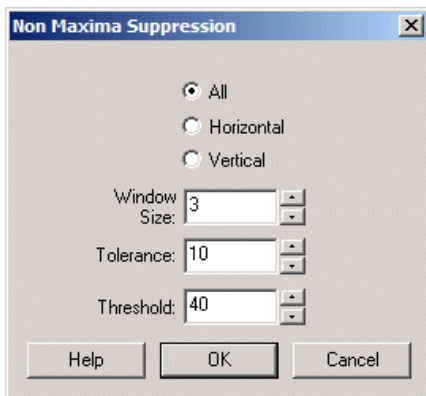
[Mean Filter](#)
[Median Filter](#)

Non-Maxima Suppression

The Non-Maximum Suppression (NMS) module will set all pixels in the current neighborhood window that are lower than the maximum value in that window to zero (or black). The module is similar to the [Max Filter](#) in that the maximum value for the specified window size (or current ROI area) is calculated. The current pixel is then compared to this maximum value. If lower it is set to black otherwise the value is unchanged.

Non-Maximum Suppression is a useful processed after other filters that produce a feature space image (see [Hough Transform](#)) and the tops or troughs of the created image need to be isolated as they may indicate the presence of significant feature types in an image.

Interface

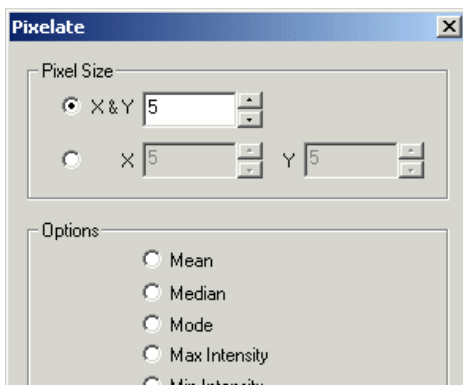


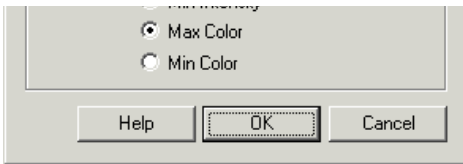
Pixelate

The Pixelate module will reduce the apparent resolution of the current image by replacing a specified window size block of pixels with their average value. As all pixels in a block get replaced with the average value the effective resolution of the image decreases to mimic how images look when zoomed to a very high magnification.

This module is useful when you want to reduce the amount of detail information into a more grid like structure.

Interface





Instructions

1. Pixel Size - specify the "pixel" or window size that is used to reduce the image detail. You can either modify by the X and Y size (as a square) together or you can modify each one individually to create a non-square pixel

2. Method - specify the type of sampling method to use when pixelating the image.

Mean - replace the superpixel with the mean or average color of all pixels it covers

Median - replace the superpixel with the median or middle color of all pixels it covers

Mode - replace the superpixel with the mode or most frequent color of all pixels it covers

Max Intensity - replace the superpixel with the highest intensity pixel of all pixels it covers (a form of dilation)

Min Intensity - replace the superpixel with the lowest intensity pixel of all pixels it covers (a form of erosion)

Max Color - replace the superpixel with the most colorful pixel of all pixels it covers (a form of dilation)

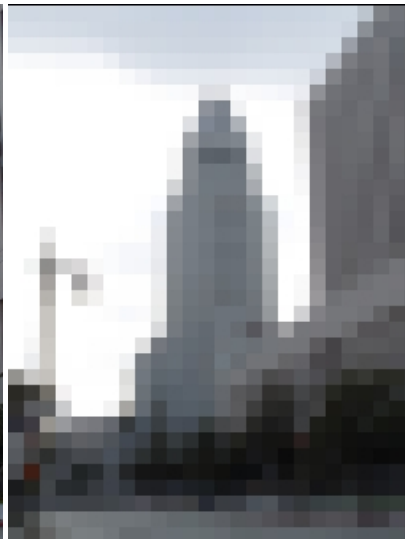
Min Intensity - replace the superpixel with the least colorful pixel of all pixels it covers (a form of erosion)

Example

Source



Pixelate



See Also

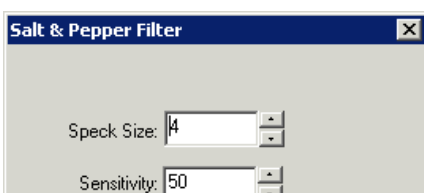
[Scale](#)

Salt & Pepper

The Salt and Pepper filter will remove "hotspots" in the image. Hotspots are defined as those pixels whose intensity values are abnormally higher (salt) or lower (pepper) than the immediate surrounding. The filter will determine what the average intensity is around a particular pixel and then based on a similarity set the pixel to that average value if the pixels value is above that similarity threshold.

This filter is most useful in astronomic images that contain noise from long exposure CCD devices.

Interface



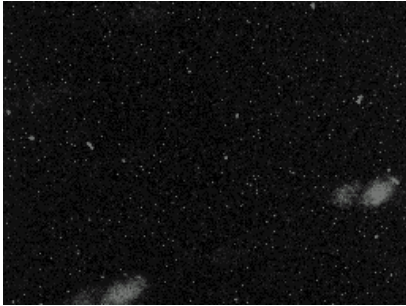


Instructions

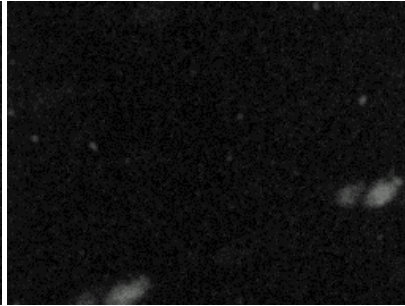
1. Speck Size - Select the speck size that you want to remove from the image.
2. Sensitivity - Select how much a suspect pixel needs to be different from the neighborhood mean to be considered a speck. The smaller the number the more sensitive it will be. At a level of 1 the filter will act as a mean filter, at 255 the filter will produce minimal change.

Example

Source



Salt & Pepper



See Also

[Median](#)

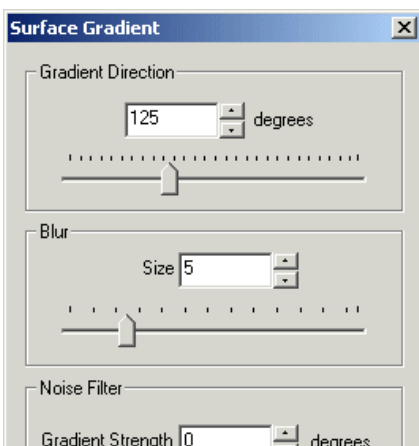
[Mean](#)

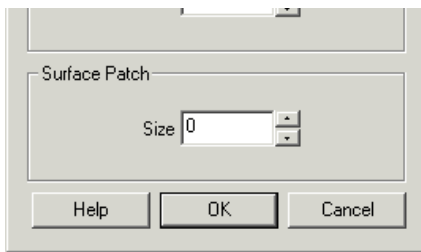
Surface Gradient

The Surface Gradient module provides a way to highlight the intensity gradient within an image. Often due to shadows and other lighting distortions the actual intensity value of an object can change. The gradient or direction caused by the intrinsic intensity of color of the object stays relatively the same under these circumstance. Using the Surface Gradient module it is possible to highlight gradients of a specific angle that can be used for successive processing.

A surface gradient is similar to an edge but instead operates specifically on the direction of the slope of that edge. Instead of generating a high pixel value when a high transition (like white to black) occurs as is done in an edge detection routine the Surface Gradient module looks at the normal of the planar surface created by that same edge. Thus a direction of gradient (0 - 360) becomes the most relevant result of gradient detection. Note that even edges that are small in transition (i.e. gray to black) will still cause a planar surface with a gradient at a particular direction. This ability is one of the Surface Gradient's advantage over the edge detector in that it is not as dependant on actual image intensity.

Interface





Instructions

1. Gradient Direction - Specify the gradient's degree that you want to highlight. This is not unlike adjusting the desired lighting direction to highlight a particular gradient. For example, specifying 90 will cause those surface areas of a gradient of 90 to become mostly white (255) with other gradients further from 90 degrees being represented with a darker intensity.
2. Blur - To reduce noise within the image it is recommended to blur the image slightly. This will help to reduce the spurious surface gradients caused mostly by noise and produce a smoother gradient image.
3. Noise Filter - To further reduce gradients whose angles are created from low intensity gradients you can set a gradient threshold to exclude those surface patches whose gradient vector strength is not significant. Increasing the noise filter value will cause only those gradients with high a high intensity shift to be considered. Note this value depends on the intensity of the image and thus is not invariant to intensity changes.
4. Surface Patch - To average a gradient over a larger patch size in order to smooth the gradient you can increase the gradient patch size under consideration. Increasing this value has a similar effect to increasing the blur but has a more accurate representation but at the cost of CPU processing.

Example

Source



Surface Gradient



The example above shows how the steps can be highlighted using a gradient of 180 or 0. This image can then be thresholded to better isolate the staircase.

See Also

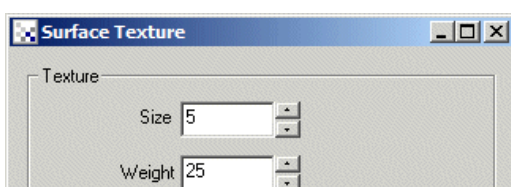
[Sobel Edge](#)

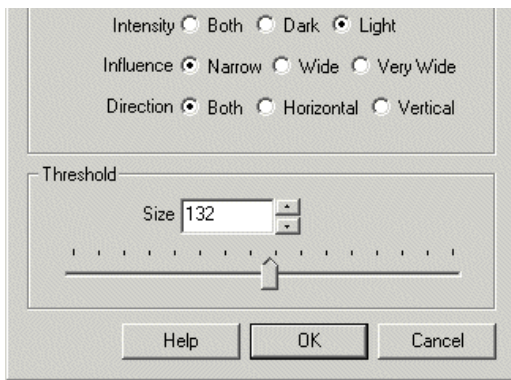
Surface Texture

The Surface Texture module is used to highlight surface irregularities that define texture. Similar to edge detection, the surface texture module will also highlight edges but normalized to their surroundings. The surface texture module can be tuned to specific frequencies and texture directions which also make it insensitive to lighting changes.



Interface





Instructions

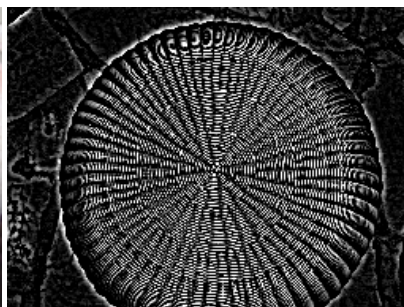
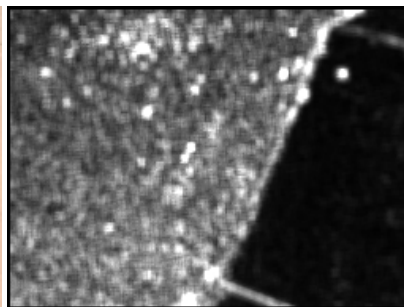
1. Size - The Size attribute specifies the area of consideration that the filter will analyze for intensity changes. The larger the size the more subtle textures will be highlighted.
2. Weight - The sensitivity to light that is taken into account when determining texture. The larger the value the more average intensity of the window size is preserved. This ensure that darker areas which will have more random texture can be reduced.
3. Intensity - When determining texture the module will regard pixel intensities above and below the average intensity within the window size. You can select that only lighter than average or darker than average intensities are reflected. This allows the module to also segment small parts from the image as texture.
4. Influence - To reduce noise, you can select that the results of neighboring pixels be combined into the final result. This tends to smooth the results by averaging from the surrounding pixels.
5. Direction - For Wide and Very Wide Influences you can select a specific direction to amplify.
6. Threshold - For a non-zero threshold value you can select which pixels to preserve above the specified value. Use zero to disable the thresholding to view the gray level response.

Example

Source



Surface Texture





See Also

[Sobel Edge](#)

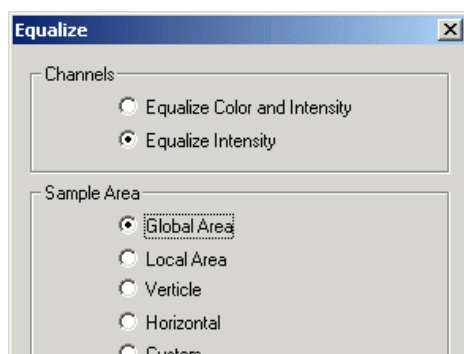
[Adaptive Threshold](#)

Equalize

The Equalize module equalizes an image's contrast based on various sample sizes. The module attempts to equalize the number of pixels in a given color and will tend to flatten and raise an image's histogram. Selecting global will equalize the image based on all pixels in the image. Vertical equalization uses only a single column of pixels into the equalization process. Horizontal equalization uses a single row of pixels.

The equalization module can operate on just the pixel intensities or on all RGB channels. This allows the module to modify the colors of the image to better spread them across the spectrum. If you find that the colors become too skewed when using the Color mode switch to the intensity mode. The Intensity mode will just modify the pixel intensities and not alter the color.

Interface



Custom Sample Area

Area of consideration

Width Height

Area of modification

Width Height

Instructions

1. Channel - Specify which channels (color and/or intensity) should be modified by the equalization module.
2. Sample Area - Specify which area is checked when performing the histogram equalization.

Global Area - consider the entire image when performing histogram equalization.

Local Area - consider a 20x20 area when performing histogram equalization. Note that this mode can be VERY SLOW due to the local operations required for each pixel.

Vertical - consider the vertical column when performing histogram equalization.

Horizontal - consider the horizontal column when performing histogram equalization.

Custom - create your own block sizes.

3. Custom Sample Area - If you select Custom as the Sample Area you can specify the size of the equalization windows.

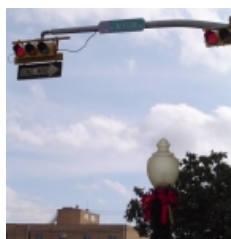
The 'Area of Consideration' specifies the window size of those pixels that will contribute to the equalization process. The 'Area of Modification' specifies the window of those pixels that will be modified based on the histogram calculated from the 'Area of Consideration'. The two window sizes allow you to specify an overlapping processing which can help to smooth the individual results better.

Example

Source Image



Equalized Image





See Also

[Normalize](#)
[Flatten](#)

Exponential Histogram

The Exponential Histogram will counter the effects of the [Logarithmic Histogram](#) module and allows you to transform an image back into linear lighting space. Performing an exponential adjustment of the histogram will tend to darken brighter pixels while keeping dark pixels dark.

Example

Source Image



Exponential



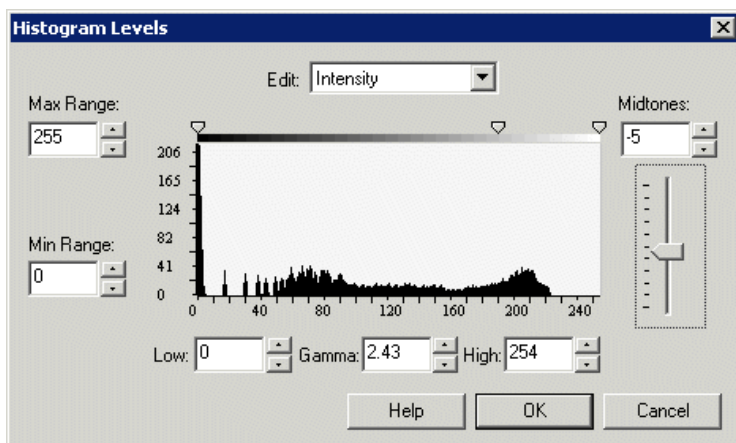
See Also

[Logarithmic Histogram](#)
[Normalize](#)
[Equalize](#)

Histogram Levels

The Histogram Levels module provides a way to control the histogram of an image in a more customized way by compressing, stretching and bending the histogram to achieve a desired effect. The modification can be done on multiple channels or a single channel.

Interface



Instructions

1. Edit - you can edit all RGB channels together, the Intensity only, Red channel, Green channel, or Blue channels. Selecting RGB will change all the RGB values of the histogram simultaneously. Selecting Intensity will only change the luminosity of the image. Selecting Red, Green or Blue will only

RGB values of the histogram simultaneously. Selecting Intensity will only change the luminosity of the image. Selecting Red, Green or Blue will only change that particular color channel. Note that each channel modification is performed in this order. Thus if the Intensity channel is chosen and a modification is made the subsequent channels (Red, Green, Blue) will make their modifications after the intensity is adjusted. This allows for each channel to specify its modifications prior to other channels being processed.

2. Range max/min - changing the values in the output range will compress the histogram values within the max-min range. Reducing the max will prevent pixel values from exceeding that value. I.e. it reduces the maximum intensity of the image. Similarly increasing the min range will increase the lowest possible intensity value which increases the overall intensity of the image.

3. Midtones compress - to bend the image towards lighter and darker intensity values select a new midtone by entering a value into the text box or by dragging the slider up or down. Increasing the midtones moves the image closer to a high contrast image, reducing the midtones reduces the contrast towards a more gray image.

4. Stretching - by moving the triangle markers below the histogram you can threshold the image to the specified minimum or maximum values. All pixels that are below (darker) than the minimum (dark triangle) will be set to zero. All pixels whose values are greater than the maximum (white triangle) will be set to 255. Instead of dragging the markers you can enter in values into the text boxes below the histogram. You can change the values directly in the provided text boxes or by sliding the white and black markers up and down the intensity range.

5. Gamma - you can change the image gamma value by dragging the middle triangle marker (gray). The Gamma value can also be set by entering a number (typically around 1.0) into the provided text box.

Example

Source



Histogram Levels (gamma 2.43, midtone -5)



See Also

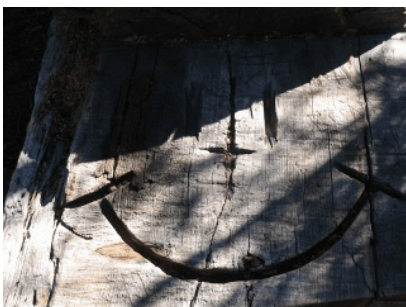
[Equalize](#)
[Normalize](#)

Logarithmic Histogram

Performing a logarithmic adjustment of the histogram will tend to brighten darker pixels while keeping bright pixels bright. The logarithmic module helps to bring out objects in shadows.

Example

Source Image



Logarithmic



See Also

[Exponential Histogram](#)
[Normalize](#)
[Equalize](#)

Normalize

The Normalize module stretches an image's pixel values to cover the entire pixel value range (0-255).

The function processes each color band (RGB) and determines the minimum and maximum value in each of the three color bands. Once these values are computed the image is reprocessed by subtracting the minimum value of each band from each pixel and dividing by its max-min range (3 times for each RGB pixel). This has the effect of stretching the pixel value to the full 0-255 pixel value range. Visually the image appears to have increased in contrast.

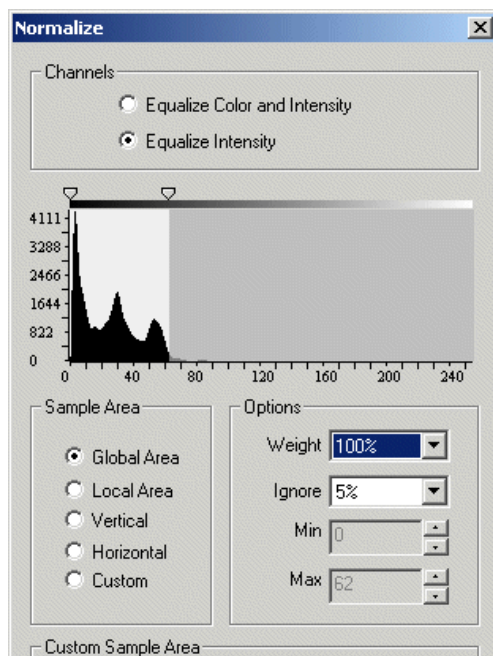
$$R = ((R - R_{min}) / (R_{max} - R_{min})) * 255.0$$

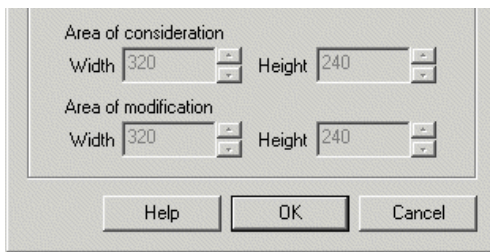
$$G = ((G - G_{min}) / (G_{max} - G_{min})) * 255.0$$

$$B = ((B - B_{min}) / (B_{max} - B_{min})) * 255.0$$

Normalization is a good tool to combat lighting changes as the camera moves. Many filters rely on absolute pixel values (such as threshold) that are easily changed in different lighting conditions. Normalization is one of the tools used to reduce this sensitivity to lighting conditions.

Interface





Instructions

1. Channel - Specify which channels (color and/or intensity) should be modified by the normalization module.
2. Sample Area - Specify which area is checked when performing the histogram equalization.

Global Area - consider the entire image when performing histogram normalization.

Local Area - consider a 20x20 area when performing histogram normalization. Note that this selection will execute VERY SLOWLY due to the per pixel computational requirements.

Vertical - consider the vertical column when performing histogram normalization.

Horizontal - consider the horizontal column when performing histogram normalization.

Custom - create your own block sizes. The 'Area of Consideration' specifies the window size of those pixels that will contribute to the normalization process. The 'Area of Modification' specifies the window of those pixels that will be modified based on the min/max values calculated from the 'Area of Consideration'. The two window sizes allow you to specify an overlapping processing which can help to smooth the individual results better.

3. Weight - specifies how much normalization the image should receive. At 100% the image is changed to the normalized values. At 20% the image pixels are changed by 20% towards the normalized values. The weighting is most useful when in the "Local Area" mode since a 100% normalization can produce undesired effects.

4. Ignore - specify the lower percent values of the histogram in which to ignore when determining the minimum and maximum levels to use in normalization. At 5% any pixels that are less than 5% in count relative to the maximum pixel count is ignored and not considered for the minimum and maximum range. This allows low pixel counts to be ignored during normalization. For example, this helps to prevent a single white pixel from forcing the maximum normalization value to be 255.

5. Min/Max - enforces limits on how wide the min and max range of the pixel values will be. For images that require normalization only in the high white pixels you can decrease the Max value which according to the above formula will limit the amount of normalization that can occur. The effect (similar to the Ignore setting) will reduce normalization noise in lower/darker pixels.

Example

Source Image

Local Normalized Image at 80% weight



Intensity Only Normalized Image



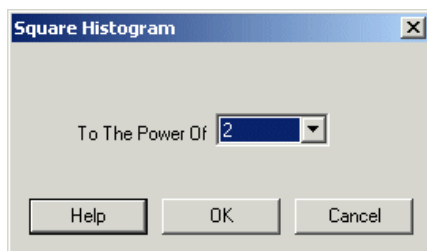
See Also

[Equalize](#)

Raise Histogram

The Raise Histogram module will square each pixel value in the current image. The squaring of pixel values will tend to make darker pixels darker while brighter pixels remain bright if raised to a integer value. The opposite effect occurs when raised to a decimal number (similar to the [logarithmic module](#)). Raising an image to an integer number can be an effective way to reduce background noise as indicated in our [Line following tutorial](#)

Interface



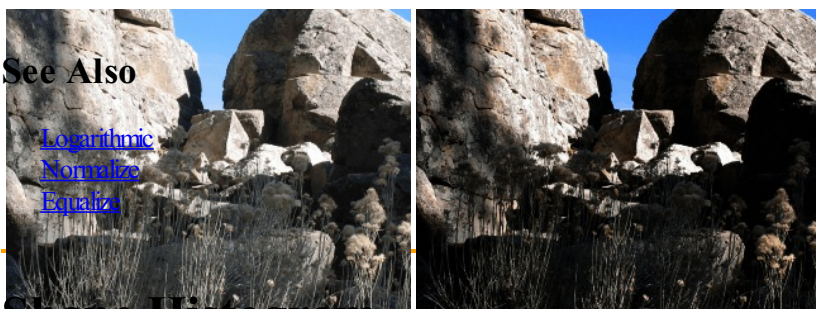
Instructions

1. Select using the dropdown menu the appropriate number to raise each pixels value to. Note the integers reflect squaring, cubing, etc while the fractional values represent square root, etc.

Example

Source Image

Raised to 2



See Also

[Logarithmic](#)
[Normalize](#)
[Equalize](#)

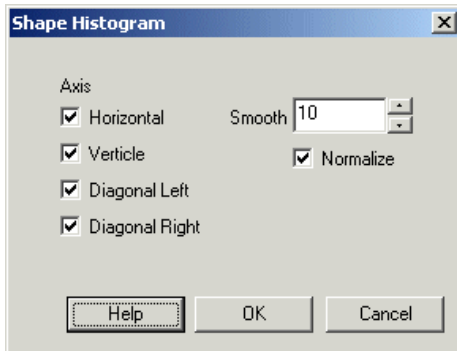
Shape Histogram

The Shape Histogram module is a type of histogram transform and can be used as part of an object classifier. A binary image is used to generate a histogram that represents the run-length values of the given image in each of 4 direction.

The advantage of creating a histogram based on a shape's pixel-length span in many directions is that it reduces orientation dependency and produces a similar histogram regardless of the shape's orientation.

For example, if a square shape is being processed (100 width by 200 height) the algorithm will encounter the top left corner of the square first and proceed to the right corner counting how many pixels the top run length of the square is. Once a black pixel is encountered the span is considered terminated and the span length is added into the histogram by incrementing the bucket in the histogram that represents the span size (in this case 100) by 1. This process of counting the span lengths is then continued for each horizontal line. Depending on the selected checkboxes further orientations may be processed in a similar manner.

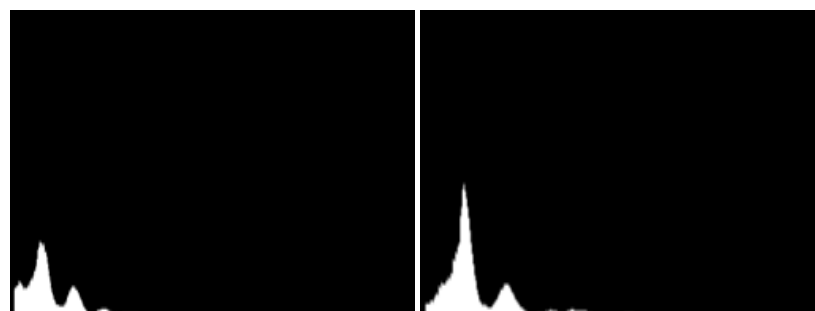
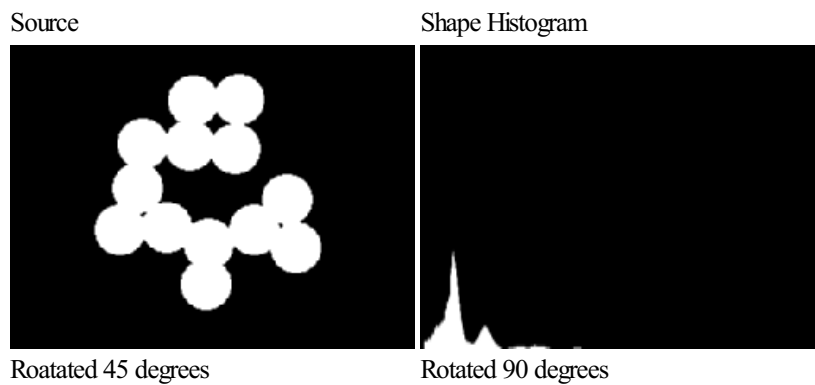
Interface



Instructions

1. Specify which axis to analyze. Using fewer axes will result in histograms that are more dependent on orientation.
2. To smooth the histogram to a more usable shape select the amount of smoothness to apply to the histogram.
3. Normalizing the histogram will scale the values to occupy the maximum range of the histogram and can be used to view more detail of the histogram.

Example



See Also

[Histogram](#)

View Histogram

The view histogram interface provides you with a view of the image's current histogram graph depending on where the view histogram is inserted into the processing pipeline. A histogram is a graphical display of the number of pixels (the Y or vertical axis) with the pixel value (the X or horizontal axis 0-255). Using an images histogram you can quickly see the nixel color distribution and how the various image processing

When using image histograms you can query the pixel color distribution and see the various image processing components effect an images histogram.

It essentially provides another way of looking at the same image information. The vertical axis representing the number of pixels at that intensity and the horizontal axis representing the color intensity of the image's pixels.

Instructions

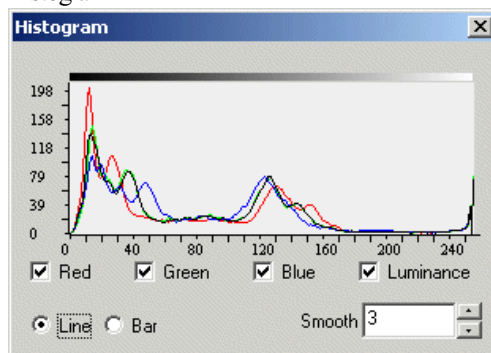
1. Select which color channel (red, green, blue, intensity) you would like to see displayed in the histogram.
2. Select the chart type - line or bar.
3. Select the amount of histogram smoothing to perform. Smoothing the histogram allows you gain a better understanding of the histogram's shape by reducing the amount of minor spikes that typically occur in a histogram.

Example

Image



Histogram



See Also

- [Normalize](#)
- [Equalize](#)

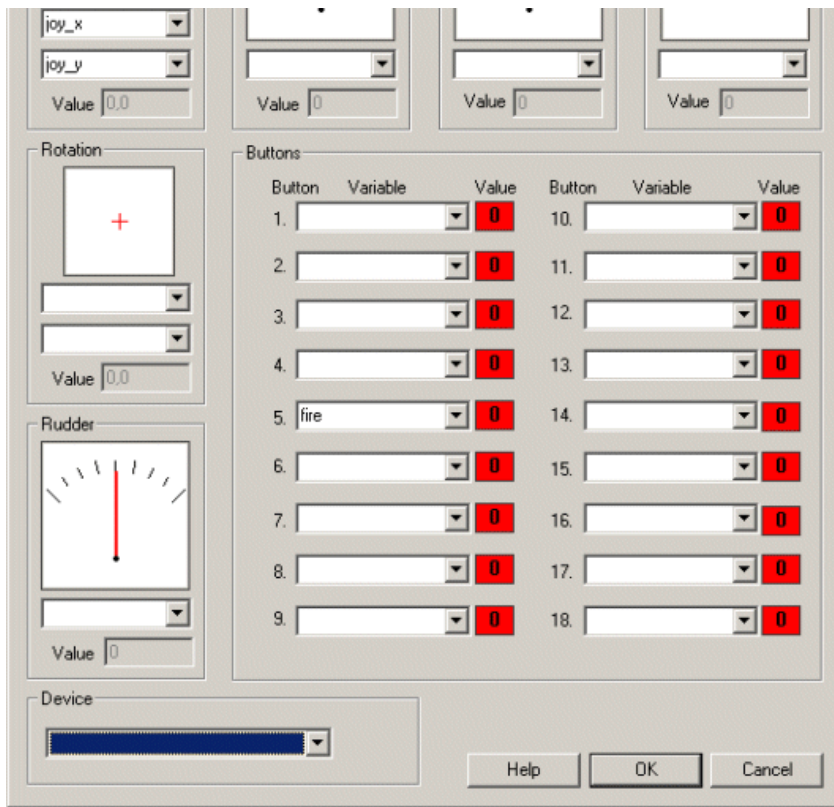
Joystick

The Joystick module provides you a way to interface a joystick to your robotic projects. This can be useful if you need a manual override or to provide suggestions to your control program. It is also useful for testing out mechanical designs prior to automatic control.

As there are many joysticks with different capabilities this module attempts to capture the basics of what might be needed. You can edit the interface and try moving your joystick around to see how the stick, buttons and levers change values in the interface. From that you can map the values into RoboRealm variables that are used within other modules to perform certain actions.

Interface





Instructions

1. Device - Select your appropriate joystick device from the dropdown list. If your joystick does not appear close this interface, plug in your joystick and double click on the joystick module to reshown the GUI interface. Then select your joystick from the dropdown menu.
2. You should now be able move your joystick and see the gauges moving appropriately. By pressing the joystick buttons you should see the red 0 numbers change to a green 1 when pressed.
3. Joystick, Twist, Throttle, etc - Specify the variable that you would like to be set with the corresponding joystick value. You can either type in a new variable name or use an existing one from the dropdown menu.
4. Buttons - specify which variables should be set to 0 or 1 depending on the current status of the appropriate button.

Example

The interface above has been configured to assign the joystick X coordinates to a variable called "joy_x", Y to "joy_y" and the fire button to the variable "fire".

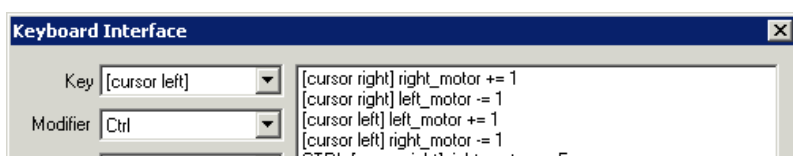
Keyboard

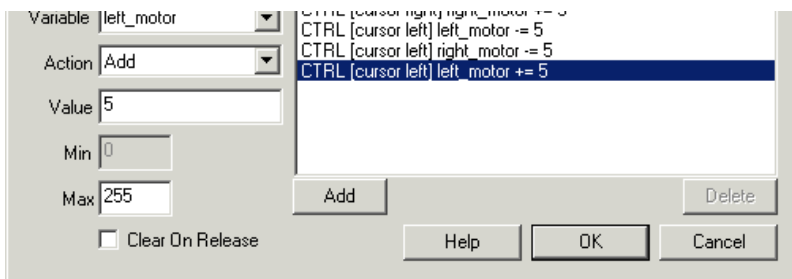
The Keyboard module provides a way to interface key strokes (from your keyboard) to change specific variables within RoboRealm. For example, using this module you can assign that when F1 is pressed the variable left_motor is set to 0 and when F2 is pressed that same variable will be set to 255. This is a nice way to add options within your program to perform different behaviours without needing to load other programs or reconfigure interfaces.

The key to interfacing with control elements (like motors, servos, etc.) within RoboRealm are via variables. If you change a variable that is used within one of the control interfaces that maps to a servo you are in effect controlling that servo. Thus, by changing a variables value you can change the behavior of a servo, usb missile launcher, pan/tilt camera, etc.

The Keyboard module is also a good replacement for the Joystick or Mouse interface modules.

Interface





Instructions

1. To create a new key/variable combination click on the Add button. This will create a new default key/variable mapping. Use the interface items on the left side of the interface to change the properties of the current mapping.
2. Key - the keypress you want to use
3. Modifier - select if one of the modifier keys should be used (SHIFT, ALT, CTRL, WIND) in combination with the selected key.
4. Variable - the variable you want modified when the above key is pressed.
5. Action - how the variable should be modified when the above key is pressed.

Set - the value is assigned to or overwrites the above variable.

Add - the value is added to the current value of the variable

Subtract - the value is subtracted from the current value of the variable

Or - the value is bitwise ORed with the current value of the variable

And - the value is bitwise ANDed with the current value of the variable

Xor - the value is bitwise XORed with the current value of the variable

6. Value - the value either assigned to the variable or combined with the variable based on the action selected above.
7. Min - the minimum value the variable can be reduced to.
8. Max - the maximum value the variable can be increased to.
9. Set on Release - if selected this will cause the variable's value to be set to the value below once the key has been released. This is useful if you need to terminate an action based on a variable when the user releases a key. The default (off) will not alter the variables value once the user has released the key.
10. Continue to edit, add and delete key/variable combinations by selecting them in the right hand side list and pressing delete to remove the currently selected combination or by using the left hand side controls to change values.

Example

The interface above shows the configuration need to change a left_motor and right_motor variables based on the cursor keys. Note that selecting CTRL while using the cursor keys will change the values quicker than without.

Try this configuration yourself. [Click on this link](#) and RoboRealm will start running the above configuration. Edit (or double click on) the Watch_Variables module in the processing pipeline to see the left_motor and right_motor variables change when you either cursor left or cursor right. To complete the example you will need to attach those variables to another control module. That part of the exercise is left up to you.

See Also

[Joystick](#)

Keyboard Send

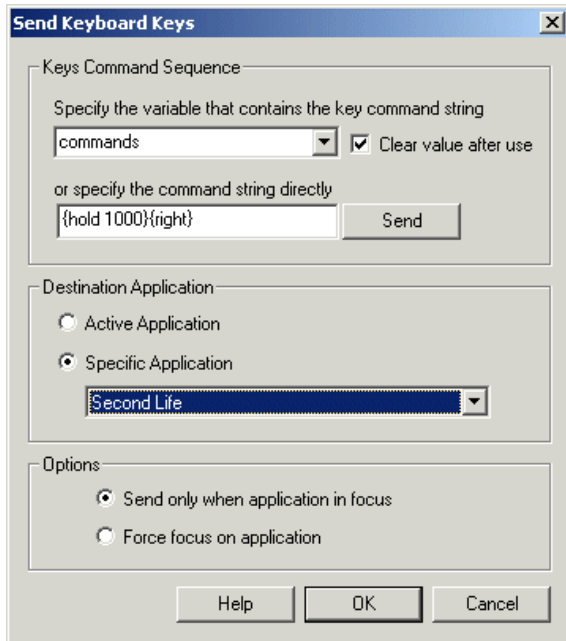
The Keyboard Send module creates a way to simulate keypresses as if you were typing into a different application other than RoboRealm. This method provides a way to integrate the results processed by RoboRealm into other applications without requiring any custom integration between the two applications.

Similar to the [Mouse](#) module this module accepts a user specified variable that contains keys to be set to another application. This is unlike the [Keyboard](#) module that instead does the opposite by reading keystrokes.

This module can be used as a simplistic way to control other applications when used in conjunction with the [Screen Capture](#) module.

WARNING - if you cannot get this module to work you may not have sufficient permissions to allow for one application to send key strokes to another. You can try to run RoboRealm as Administrator by right clicking on RoboRealm.exe and selecting "Run As Administrator" which may solve this issue.

Interface



Instructions

1. Variable - Specify the variable that will contain the keystroke commands that will be sent to the application. Note that this variable can be programmatically created using the [VBScript](#) module, the [Set Variable](#) module, the [API](#), etc.
2. Clear on use - Select this checkbox if after the command is sent the variable's value should be cleared. If the value is not cleared the variable contents will continuously be sent to the application.
3. Direct - To test sending keys to the selected application type in the key command text and press Send. This will send the keys to the specified application after forcing the focus to that application (or desktop). This is a convenient way to determine what commands the variable might use to cause certain actions to happen.
4. Active Application - Select if the key strokes should be sent to whatever the currently focused application is.
5. Specific Application - Select if you only want to send the keystrokes to a specific application seen in the dropdown list. If your program is not already running, close this GUI by pressing OK, start your application and then double click/edit this module again. Your application's name should appear in the dropdown.
6. Send when in focus - Select if you only want the keys to be sent when the specified application is in focus. In this mode RoboRealm will not send out keys until it realizes the application that you specified currently has the keyboard focus. This allows you to retain control of your desktop and only start the automated keyboard strokes when focused on a specific application.
7. Force Focus - Select if you want to ensure that RoboRealm forces the desired application to the foreground and give it focus. RoboRealm will then immediately start sending the keystrokes to that application. If the application loses focus, RoboRealm will once again force the focus back onto the application. This mode is great to ensure only one application is in focus at a time but can become confusing when focus is needed elsewhere. If you use this mode you will have to terminate the specified application in order to stop RoboRealm from forcing focus back to that application in order to make any changes to RoboRealm.

Command String

The variable content or direct command string is a sequence of keystrokes that are sent to the specified application. Because many keystrokes cannot be typed as text the command string has many embedded commands that can be used to enter in a particular character. This list is very similar to the [SendKeys](#) functionality embedded within Microsoft with a couple of additions around keystroke timing.

The following text should appear in {}'s in order to type that character into the selected application.

Command	Meaning
{DEF AY}	

{PAUSE 100}	Causes a 100ms delay and then continues
{HOLD 100}	Causes each key to be held down for 100ms
{SPEED 100}	Causes a 100ms delay between typing in each character
{FOCUS Notepad}	Sets the focus to the specified application. Useful to switch keypresses to another application. Note that you only need to include one unique word that appears in the title of the application to switch to it. It is not necessary to type in the full title.
{BEEP 477 250}	Beeps with a tone of 477Hz for 250ms.

In order to simulate key presses that have no characters the following commands can be used:

Command	Key
{SHIFT}	+
{CTRL}	^
{ALT}	%
{WINDOWS}	@

Naturally with these keys being redefined, if you need to type a plus and mean just the character '+' then you would need to use the table below to look up its sequence which is {ADD} or {+}.

You can also group a couple of characters together after a modifier key seen above. For example, +(abcd) would translate to ABCD as the '(' and ')' are grouping characters that will in this case cause all letters to be typed while the SHIFT key is pressed.

To specify more than one keystroke simply include the number of times the key is to be typed after the key. For example, {LEFT 10} means press the LEFT CURSOR KEY 10 times. {a 5} would produce "aaaaa".

Letters	Meaning
{AT}	@
{CARET}	^
{LEFTBRACE}	{
{LEFTPAREN}	(
{PERCENT}	%
{PLUS}	+
{RIGHTBRACE}	}
{RIGHTPAREN})
{TILDE}	~
{ADD}	+
{MULTIPLY}	*
{DIVIDE}	\
{UP}	CURSOR UP
{LEFT}	LEFT CURSOR
{RIGHT}	CURSOR RIGHT
{DOWN}	DOWN
{BACKSPACE}	BACKSPACE
{BKSP}	BACKSPACE
{BREAK}	
{BS}	BACKSPACE
{CAPSLOCK}	CAPS LOCK
{CLEAR}	CLEAR
{DECIMAL}	DECIMAL
{DEL}	DELETE
{DELETE}	DELETE

{END}	END
{ENTER}	RETURN
{ESC}	ESCAPE
{ESCAPE}	ESCAPE
{F1}	F1
{F10}	F10
{F11}	F11
{F12}	F12
{F13}	F13
{F14}	F14
{F15}	F15
{F16}	F16
{F2}	F2
{F3}	F3
{F4}	F4
{F5}	F5
{F6}	F6
{F7}	F7
{F8}	F8
{F9}	F9
{HELP}	HELP
{HOME}	HOME
{INS}	INSERT
{LWIN}	LEFT WINDOWS KEY
{NUMLOCK}	NUMLOCK
{NUMPAD0}	NUMPAD0
{NUMPAD1}	NUMPAD1
{NUMPAD2}	NUMPAD2
{NUMPAD3}	NUMPAD3
{NUMPAD4}	NUMPAD4
{NUMPAD5}	NUMPAD5
{NUMPAD6}	NUMPAD6
{NUMPAD7}	NUMPAD7
{NUMPAD8}	NUMPAD8
{NUMPAD9}	NUMPAD9
{PGDN}	PAGE DOWN
{PGUP}	PAGE UP
{PRTSC}	PRINT SCREEN
{RWIN}	RIGHT WINDOWS KEY
{SCROLL}	SCROLL
{SNAPSHOT}	SNAPSHOT
{SUBTRACT}	SUBTRACT
{TAB}	TAB
{WIN}	LEFT WINDOWS KEY

Example

{DELAY 100}@rnotepad~hello world&ha - Waits for 100ms, then invokes Win-r or the Run Dialog, types in 'notepad' followed by enter which starts that application, types in 'hello world' into notepad, invokes the Help menu using ALT-h and then selects the About menu item using the letter 'a'.

{DELAY 100}

See Also

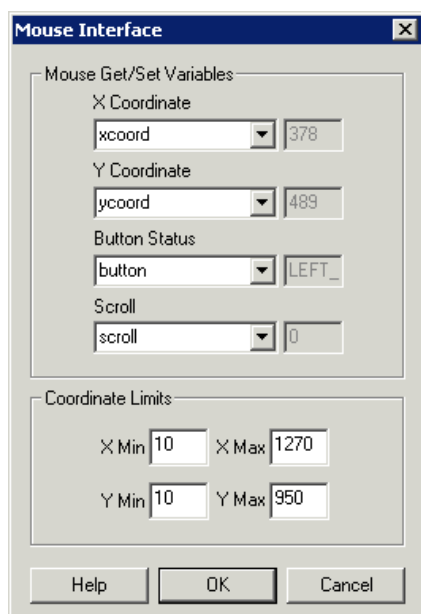
[Mouse](#)
[Screen Capture](#)

Mouse

The Mouse module provides access and control of your desktop mouse. Using the Mouse module you can read in the current mouse coordinates and button status. Using the exact same interface you can also control the mouse position and button status. The interface is bidirectional in that sense.

If the variables specified in the interface change based on some other module or VBScript program those values will be used to position the desktop mouse to those coordinates/button status. If, however, the current pipeline does NOT change the variables specified then AND the mouse changes position then those values are put into the specified variables. Thus, if you change the variables specified the mouse will respond accordingly, but if you only read the variables then the values of the actual mouse position and button status will be used to populate the variables. The direction is based on how you use the variables.

Interface



Instructions

1. Get/Set - Specify the variables that will either contain the values to set the mouse cursor or are used to read in the current mouse position and button status.
2. Three second stabilize delay - Normally the module will not change the mouse position (Set) unless the mouse has stopped moving for 3 seconds. This is to ensure that you do not have to fight over control over the mouse. If you uncheck this checkbox then the module will attempt to set the mouse position each time. Keep in mind the ALT-R key which will toggle the run button and cause the pipeline to stop allowing you control back over the mouse.

3. Limits - To ensure that the mouse coordinates remain within a specified boundary specify the minimum and maximum limits that the mouse values can range from.


You can set the button status by using one of:

- LEFT_CLICK
- RIGHT_CLICK
- LEFT_DOUBLE_CLICK
- RIGHT_DOUBLE_CLICK
- MIDDLE_CLICK
- LEFT_DOWN
- RIGHT_DOWN
- MIDDLE_DOWN
- LEFT_UP
- RIGHT_UP
- MIDDLE_UP

When reading the button status the value will be one of:

-
- LEFT_DOWN
- RIGHT_DOWN
- MIDDLE_DOWN

Example

 [Click here](#) to load a configuration to move the Mouse based on the position of a red object! We configured the mouse button to fire when the ball becomes > 50 pixels. You may need to change that based on your red object size and how close you are to the camera.

Moving a red object in front of your camera will cause the mouse to also move. Note that moving the object up moves the mouse down. If you move the object towards the camera (zooming) and the COG_BOX_AREA is above 50 then a mouse button click is simulated. So if you start out with a really large red object your desktop will probably get quite crazy as simulated mouse clicks are spread around. The best way to start this demo is to not have any red objects around, then introduce a small red object (like a object the size of a gold ball) and then gradually move the object towards the camera to click.

Clicking on a window is very tricky. You have to keep your hand steady with respect to the camera while approaching the camera to ensure that the mouse does not move away from the window bar.

You can still use your mouse even when RoboRealm is trying to control it. Once RoboRealm sees the mouse moving it will wait until the mouse is stable for 3 seconds and then start setting its position again. When you run the above example, wait for 3 seconds, and you should then start to see your mouse jumping around. The above configuration is created for a 320x240 camera meant for a 1280x960 screen. Thus there is a scale factor of 4 in the VBScript that you may change based on your screen and camera size. The larger the camera size the more stable the mouse will become.

Variables

SCREEN_WIDTH - specifies the desktop screen width

SCREEN_HEIGHT - specifies the desktop screen height

See Also

[Joystick](#)

Ftp Images

The Ftp Images module provides a way to transmit images from RoboRealm to an FTP server. This module can be used to update remote sites with images based on a specific frequency or time criteria. You can also overwrite the same image in the remote FTP site each time or cause the filename to change on each upload.

Note that your network must allow for outbound FTP network connections in order for this module to function. Also note that the FTP uses

passive mode for uploading images.

Interface

The screenshot shows a dialog box titled "FTP Images" with a close button (X) in the top right corner. The dialog is organized into several sections:

- Image to Save:** A dropdown menu currently showing "Current".
- Ftp Connection:** A group box containing:
 - Hostname: localhost
 - Port: 21
 - Username: test
 - Password: user
 - Filename: image.jpg
- Frequency:** A group box containing:
 - Send Every: 20 (dropdown), Seconds (dropdown)
 - Between: 3:38:25 AM and 3:38:25 AM
- Image Numbering:** A group box containing:
 - Number Image Filenames
 - Start numbering from: 1
 - Limit to: 0 frames.
 - Stop after: 0 (dropdown)
 - Frames: 0
 - Elapsed Time: 0 Seconds

At the bottom of the dialog are four buttons: Start, Help, OK, and Cancel.

Instructions

1. Image to Save - Specify which image you would like to send to the remote FTP site

Source - the original image that was initially loaded or captured into RoboRealm

Current - the currently processed image within RoboRealm

Last - the last image processed by RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created.

If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

2. Hostname - The name of the FTP server to send the image to

3. Port - The command port of the FTP server (default is 25)

4. Username - The username of the login required to access the remote FTP server

5. Password - The password for the login required to access the FTP server

6. Filename - The path/filename to save the image as in the FTP server

7. Send Every - As you will probably not want to send each image to the FTP server you can specify a periodic frequency in which to send the image. For example, sending an image every minute or every 20 seconds would help to reduce bandwidth issues and still provide an image that updates.

8. Between - You can also specify that images will only be uploaded during a specified time range. This can be useful if you wish to only upload images during a certain time period (for example, only upload images after office hours).

9. Number Image Filenames - If you wish to keep all the images uploaded to the FTP server you can chose to use a different name for each upload. Selecting the numbering option will cause a number to be attached to the filename that is incremented for each new uploaded image.

Because you could fill up hard drive space quite quickly in doing this you also have the option to limit the number of uploaded images to X number of images or after a set amount of time (whichever comes first).

10. START - To start sending frames press 'Start'. You can also manually stop the recording by pressing 'Stop'. Once START is pressed you can save the robofile or exit RoboRealm as on reentry or reloading of the robofile the START button will already be depressed and immediately start sending images (as appropriate).

See Also

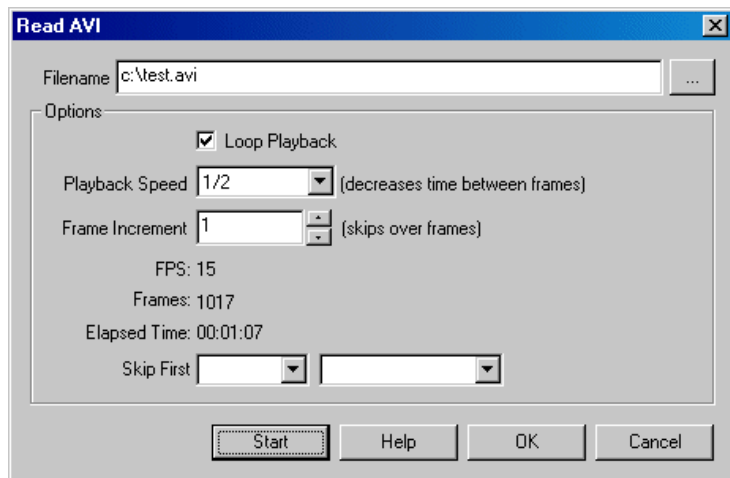
[Write Images](#)

Read AVI

Note that this module is only provided for backward compatibility. For playback of AVI files please see the [Media Reader](#) which supports more formats and different video sizes.

The Read AVI module allows you to process an image sequence without having a live camera feed. Looping playback will restart the video stream once the stream has ended.

Interface



Instructions

1. Filename - Specify the file to read in the "Filename" text box. Click on "..." to browse your file system
2. Start - To start playback press "Start". You can also manually stop the playback by pressing "Stop".
3. Loop Playback - If you want to ensure that the video loops around and around click on the "Loop Playback" checkbox.
4. Playback Speed - to speed up the playback of a slow video select a playback speed. This is useful if you want to process a video and produce the frames as quickly as possible.
5. Frame Increment - if you do not need to process every frame you can cause frames to be skipped in order to proceed through the video quickly.
6. Skip first - if the start of your video contains unnecessary frames select how many seconds to skip into the video before starting playback.

Note that you must press start for the playback to start streaming. Simply pressing OK will NOT begin playback.

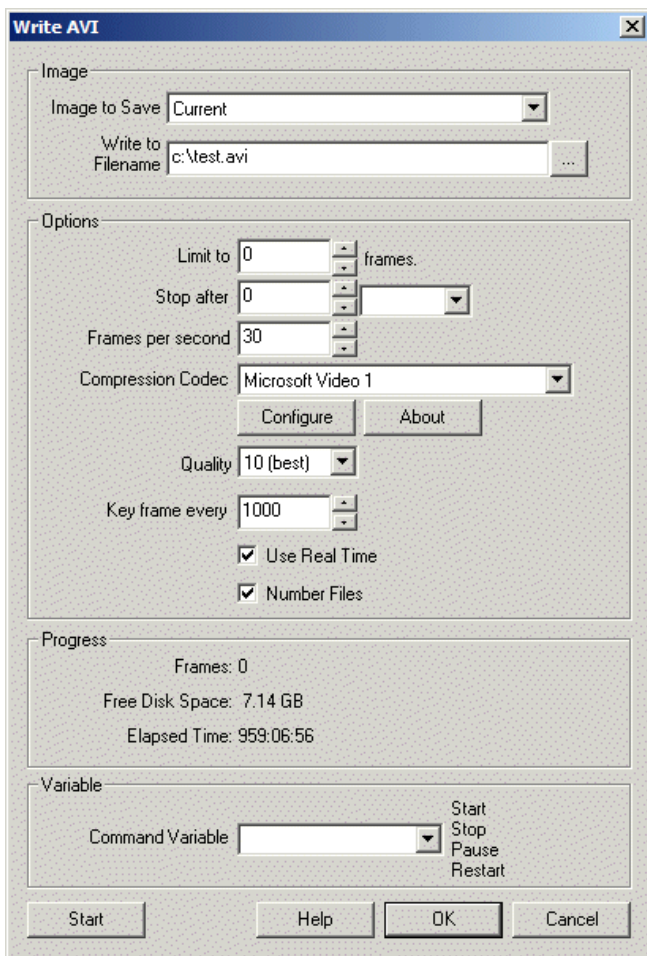
See Also

- [Write AVI](#)
- [VLC Player](#)
- [Media Reader](#)

Write AVI

The Write AVI function allows you to save the processed video to an AVI video file. This file can then be played back using any AVI player (with the appropriate decompressor) when needed without running RoboRealm.

Interface



Instructions

1. Image to Save - Specify which image to save

1. Image to Save - Specify which image to save.

Source - the original image that was initially loaded or captured into RoboRealm

Current - the currently processed image within RoboRealm

Last - the last image processed by RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created.

If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

2. Write to Filename - Specify the file to save the recording to.

3. Limit to - To stop the recording you have two options. You can limit recording to a set number of frames and/or stop after a set amount of time (whichever comes first).

4. Stop after - Specify the time to stop recording.

5. Frame per second - Specify the number of frames per second the video should record at. Note that 30 frames a second is standard for most movies. However, if you do not need very fluid movement you can reduce the number of frames per second in order to reduce video size.

This can also be used to record a movie slower or faster than being played.

6. Compression Codec - Specify the type of compression you want to use to record the video. Note that the video size will depend greatly depending on what type of compression you use. If you are not familiar with compression codecs chose "Microsoft Video 1" which is a good general codec.

7. Configure button - To configure the compression options to a finer degree you can use the Configure button to show the compression dialog specific to the selected compression driver.

8. About button - To know more about what Compression Codec you are using press the About button.

9. Quality - Many of the codecs will allow a quality measure to be provided on how well the video will represent the true pixels. Less quality will create a smaller video file whilst higher quality video will create much larger files.

10. Keyframe every - Some compression drivers use temporal encoding which may need more keyframes when playback occurs online or with some viewers. A keyframe is a frame within the video that includes all information to reproduce that frame without having any dependent information stored in previous or next frames.

11. Use Real Time - If you use a function that may slow the image processing pipeline (such as the movement detector) you should uncheck the "Use Real Time" checkbox. This removes the actual timing of the image frames so that frames can be played back in a sequential time frame irrespective of the actual time the image frame was saved. With the checkbox checked the actual time while recording is used. This may cause the original timing not to be preserved when processing videos.

12. Number Files - To avoid overwriting files that already exist the number files checkbox will append a number to the end of the filename if the file already exists. This helps to avoid destroying existing data.

13. Command Variable - To automate the starting and stopping of video recording (i.e. through the API) you can specify a variable whose value will contain either Start, Stop, Pause or Restart as a command word which will cause the module to perform the appropriate action. Note that once read the variable will be set to blank to avoid repeating the same action more than once.

To start the recording press "Start". You can also manually stop the recording by pressing "Stop".

Note that any recording will terminate if less than 20 Megs are available on your hard disk to prevent the computer from running out of disk space.

Also note that if you do not have permissions to write to a particular location (for example `c:\test.avi` which is the default location) the OS may redirect that to `C:\Users\Your UserName\AppData\Local\VirtualStore` instead. So if you cannot find your video that you just recorded, check in that location too!

The Write AVI module will cycle files (close and open a new file) approaching 2 gigs to avoid the 2 gig limitation imposed by VFW (Video for Windows).

See Also

[Media Reader](#)

[Read AVI](#)

[Write Images](#)

Write Clipboard

The Write Clipboard module provides a way to "paste" an image from RoboRealm into the system keyboard. Note that any current image into the clipboard is emptied and replaced with the current image within RoboRealm.

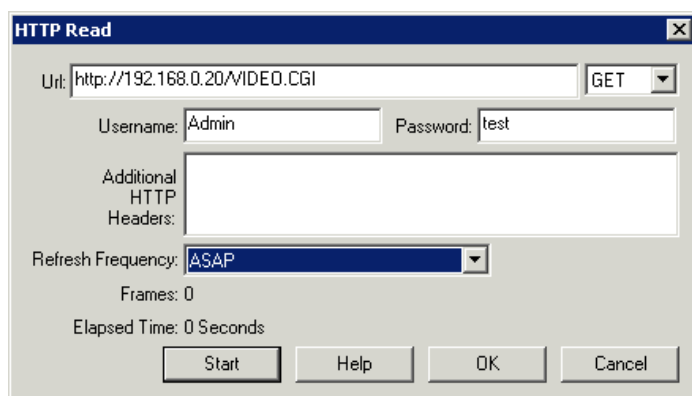
See Also

[Read Clipboard](#)

HTTP Read

The HTTP Read module allows you to process images from remote site across the internet instead of from your connected webcam. This module can also be used to access internet enabled webcams that may be connected on your local network.

Interface



Instructions

1. Camera Type - select which type of camera you wish to connect to.
2. Url - Enter the web URL in the provided box. The url should be a HTTP Push server or to a single web image that when reloaded would provide a different image. Any url provided via the Camera Type dropdown will need to be configured for your domain. If your camera type does not appear in the Camera Type list you will need to find the appropriate url to use that when access will deliver a MJPEG image over a http protocol.
3. Enter the HTTP Method in the dropdown
4. Username/Password - Enter any Basic HTTP authentication in the username and password text boxes.
5. Headers - Enter any additional HTTP headers needed
6. Refresh Frequency - Select how frequently the image should be reloaded. In the case of HTTP Push the image will always be updated as quickly as the remote server provides the image.
7. Erase on Error - if the remote camera fails to send images (during a network/power/etc failure) clear the image to black to indicate an error.
8. Timeout - the number of milliseconds to wait for a remote image. When this fails the image will be cleared based on the above "Erase on Error" and a system variable SYSTEM_ERROR will be set to indicate this timeout failure. Note that the rest of the pipeline DOES execute even after this error.

Example

1. Insert the HTTP_READ module (located under the Loading/Saving level) into the RoboRealm processing pipeline.
2. Enter the url specific to your camera. For the Axis 206 you would enter in your url looking something like

`http://camera.ip.address/mjpg/video.mjpg`

but note that your IP number will most likely be different. If you do not know the url of your camera image you may need to look under the hood of a HTML page that shows the streaming image. You most likely have that page installed with the software that comes with the camera. Using your browser select 'view source' and look for the url.

3. Press the "start" button and if you don't get any errors then you should then see the image start streaming.

The HTTP_READ module does NOT read HTML pages. You need to provide it with the specific url that will return a jpeg image. It will understand both moving jpeg images (mjpg) OR still jpeg images that refresh each time they are reloaded.

Note that [DLink](#), [TRENDnet](#) and [Linksys](#) cameras have their own modules specific to those cameras to allow for more of the camera features to be accessed in RoboRealm.

VLC Media Player

If you want to stream from a VLC server, you need to start the capture using VLC and then select the Stream option under the Media menu. You will need to Add Http as a Destination, edit a profile to select MJPEG encapsulation and M-JPEG as the Video codec in order for VLC to stream using MJPEG. Once you press the Steam button at the end of the Wizard, you can now use this module with `http://localhost:8080/` as the URL assuming you didn't change the defaults. This should then start showing the image within the main RoboRealm window.

Your final streaming output string should be something like

```
:sout=#transcode{vcodec=MJPEG,vb=800,scale=Auto,width=320,
height=240,acodec=none}:http{mux=mpjpeg,dst=:8080/} :sout-keep
```

Note that while streaming VLC may not show the image being streamed.

See Also

[RTSP Player](#)

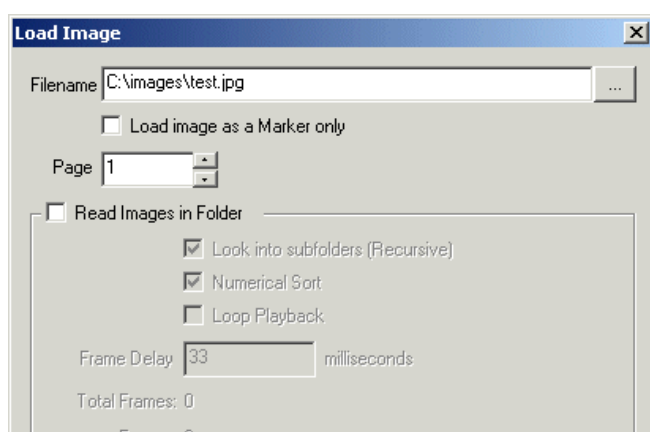
[VLC Player](#)

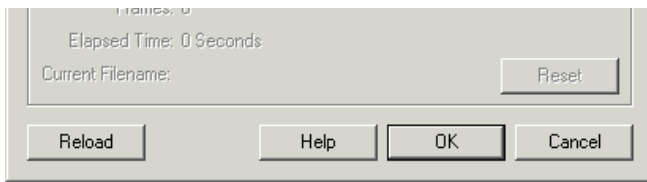
[HTTP Text Module](#)

Load Image

The Load Image module loads in an external image file (ppm, gif, jpg, etc) from your filesystem and overwrites the current image within the processing pipeline with that image. This module is useful if you need to refer to an external image outside of RoboRealm. For example, the Load Image module can be used to load in an image mask.

Interface





Instructions

1. **Filename** - Specify the filename of the image to load.
2. **Load as Marker** - Select the "Load as a Marker only" if you want to load the image into memory but not replace the currently viewed image. If this is unselected the current image in RoboRealm will be replaced with this image.
3. **Page** - For some formats there can be more than one page in the file (specifically PDFs). Specify the page number to load in a multi-page file.
4. **Reload on file change** - When selected the module will monitor the file date/time and reload the file if this changes. This allows the file to be reloaded if you change it outside of RoboRealm. With this unselected, the system will not attempt to reload the file unless relevant configuration changes are made to conserve disk activity.
5. If you want to load multiple images in sequence select the "Read Images in Folder" checkbox and the options below will become enabled. To read in multiple images be sure to have selected an appropriate file in the folder of those images you want to read. RoboRealm will then search the folder for appropriate images and load then in sorted succession.
6. **Recursive** - Select the checkbox if you want images in subfolders to also be loaded. Selecting this option will search from the specified folder and all other subfolders for images.
7. **Numerical Sort** - Often images have numbers in them and there is a desire to load in the images in accordance to the number within the filename. Selecting Numerical Sort will ensure that filenames that contain XXX_1.gif, XXX_01.gif, XXX_010.gif are sorted with respect to the numerical 1, 1, and 10 instead of the characters ASCII code.
8. **Loop Playback** - Select if you want the images to repeat after the last image has been loaded.
9. **Monitor Folder** - Select if you want new images added to the specified folder to be automatically loaded.
10. **Frame Delay** - To change the playback speed increase or decrease the Frame Delay. That alters how quickly RoboRealm will load in the next frame.
11. **Higher Bit Images** - RoboRealm operates on 24 bit RGB for performance reasons. When loading in images with higher bit depths RoboRealm needs to know how to process the image into a 24 bit image. There are many ways this can be accomplished:
 - **High** - Uses the upper 8 bits of the image
 - **Low** - Uses the lower 8 bits of the image
 - **Sqrt** - Square root's the image pixel to the 8 bit range
 - **Inv Sqrt** - The Sqrt function will favor darker pixels, the Inv Sqrt favors lighter
 - **Center Mean** - Forces the high bit range to be centered at the image mean
 - **Below Mean** - Shows only pixel below the image mean compressed into 24 bits
 - **Above Mean** - Shows only pixel above the image mean compressed into 24 bits
 - **Around Mean** - Similar to Center Mean but thresholds values on either side to improve contrast
 - **Normalize** - Determines image low and high values and scales to 24 bit
 - **Pseudo X** - Translates image intensity into a higher color range for improved visibility
 - **Reinhard/Drago** - High Dynamic Range reduction techniques as specified in the FreeImage.dll
12. **Process Color Channels Together** - Specifies that color channels will be considered a single channel such that the relative color amounts will not change. Unselected, each color channel will be processed independently which can improve or worsen an images overall appearance.

Note that the load image module is different from just loading an image to process in RoboRealm. The load image module will always load in the specified image regardless of what image is being processed or if live video is being played. The load image module can be used to load in images that are used for masking or other overlay techniques. Using the multiple image load capabilities you can also cause RoboRealm to act as if a movie file had be loaded instead.

Also note that the images loaded using the multiple image load are sorted with respect to numeric comparisons. Thus, image2.gif will be recognized as coming after image1.gif instead of image100.gif.

Variables

IMAGE_FILENAME - The currently loaded filename. Removed if no image

loaded or failed to load.

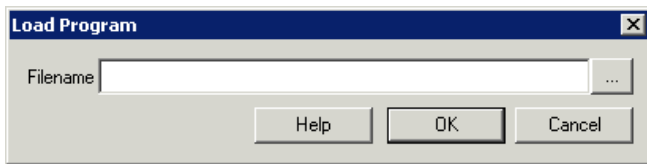
See Also

[Read AVI](#)

Load Program

The Load Program module loads in a new program that overwrites the current executing program. For one time loading you can use the Load button in the main RoboRealm dialog instead. The load program module is typically used within the [if statement](#) to jump to a different execution pipeline.

Interface



Instructions

1. Specify the filename of the program to load. Once this module is executed the current program is replaced with the loaded program and begins execution.

See Also

[Load Image](#)

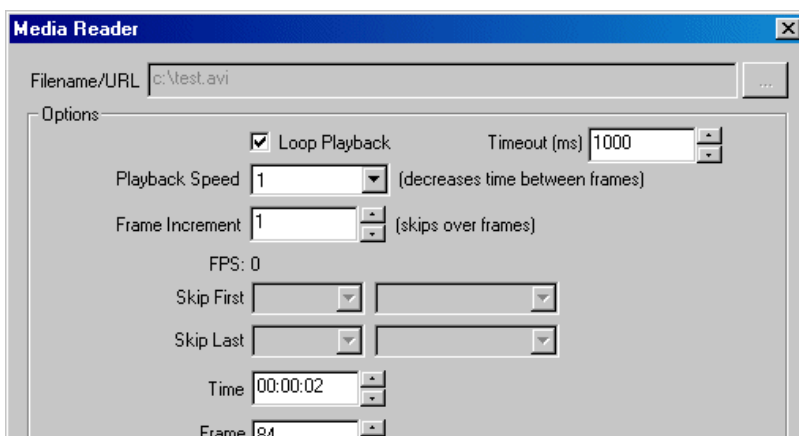
Media Reader

The Media Reader module allows you to process an image sequence without having a live camera feed. The module play accept AVI, WMV, WMA, MPG files and ASF streams. This is useful for processing the same image sequence over and over in order to test out new modules or experiment with other modules over the same sequence of images.

Note that with the addition of the ASF stream you can also hook into image streams broadcasts over the Internet in ASF format. This is similar to the [Http Read](#) component but can accept a different streaming format.

If you have issues playing videos due to missing codecs we recommend that you install the K-Lite Codec pack which includes many codecs needed for most videos. You can download the codec from [here](#). And/Or alternatively from [FDDShow](#).

Interface





Instructions

1. Filename/Url - Specify the file or url to read in the "Filename/URL" textbox. The media reader can accommodate .avi, .wmv, .mpg, and .asf files.
2. Loop Playback - If you want to loop the playback (only valid for files) then check the "Loop Playback" checkbox.
3. Timeout - On playing videos being streamed over the network a delay is needed in order to allow for network speed fluctuations. If you are playing videos that are stored locally you can reduce the timeout delay which will also be noticed when the video loops. If you are playing files over the network and the timeout is too small the video may accidentally restart as a result of a timeout. Increasing the timeout tells the Media Reader module to wait longer until a new frame arrives over the network before resetting.
4. Playback Speed - To change the playback rate of the video select the appropriate playback speed. Numbers lower than 1 reduce the speed while numbers greater than one increase it. This reduces or increases the time delay between frames.
5. Frame Increment - If you want to skip over the video and only process every other frame set the increment to 2. This will cause every other frame to be skipped.
6. FPS - Displays the current Frames Per Second that the video is recorded in. This number is needed when using the [Write_AVI](#) module to ensure consistent timing for reading and writing.
7. Skip - To Skip the first or last part of the video configure the "Skip First" and "Skip Last" accordingly. When the video loops the first X seconds, milliseconds, etc. will be jumped over. Similarly the video will end X seconds before the actual end of the video. Note that this is not cropping or modifying the video in any way.
8. Time - Displays the current frame time of the video. Changing this value will move the video to that time point.
9. Frames - Displays the current frame number of the video. Changing this value will move the video to that time point.
10. Start - You MUST press the "Start" button to begin playing. Once pressed you can also manually stop the playback by pressing "Stop" (the "Start" button will change to "Stop" once playing starts). You can also Pause the video and use the Time, Frames or Slider controls to move the video to a point of interest.

Note that you must press Start for the playback to start streaming. Simply pressing OK will NOT begin playback. Also be aware that for Internet based streams you need to wait about 20-30 seconds after pressing the start button for playback to actually start.

Note that if RoboRealm appears to freeze the connection may be down. Wait for about 20 seconds and you will regain control after the network connection times out.

Variables

VIDEO_FRAME - specifies the frame number of the current video image.

VIDEO_SECONDS - specifies in seconds where the current video frame is within the video.

VIDEO_TIME - specifies the HH:MM:SS (hours:minutes:seconds) where the current video frame is within the video.

See Also

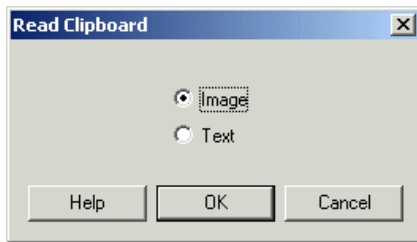
[VLC Player](#)
[Read AVI](#)
[HTTP Read](#)

Read Clipboard

Read Clipboard

The Read Clipboard module provides a way to read images into RoboRealm from the system keyboard. When the clipboard is updated RoboRealm receives that image and will continue to process the image.


Interface



Instructions

1. Select which type of data you would like to read from the clipboard. 'Image' refers to reading bitmap images from the clipboard whereas 'Text' refers to ascii text. An image read from the clipboard will replace the current image in the main RoboRealm GUI view. Any text that is read from the clipboard is placed in a CLIPBOARD_TEXT variable.

Example

 [Click Here](#) to run a robofile that will process an image in the clipboard and update its content with the processed results. Thus while running you can open a paint program, copy some image content into the clipboard and then paste the processed results back into the paint program without even accessing RoboRealm between the copy and paste.

Variables

CLIPBOARD_TEXT - the text read in from the clipboard. Maximum length that can be read is 32768.

See Also

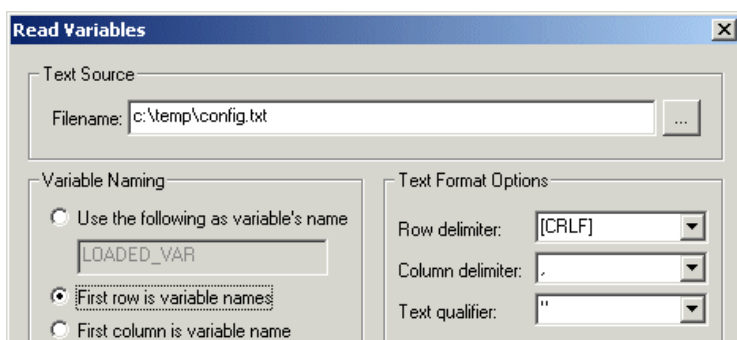
[Write Clipboard](#)

Read Variables

The Read Variables module allows you to read in RoboRealm variables from the disk drive in various formats. This can be used to load configuration or discrete data into RoboRealm for use with its modules. Be aware that this module only reads in the file once it has changed on disk. If you change a variable that has been read in from disk it will be overwritten once the file changes and is re-read into RoboRealm.

For more direct methods of changing variables within RoboRealm see the [API](#).

Interface



Help

OK

Cancel

Instructions

1. **Filename** - Specify the file that contains the information to be read into RoboRealm. Note that RoboRealm does NOT enforce a filename extension as it is expected to be a text file of some sort.

2. **Variable Naming** - Select how the name of the variables should be determined from the text file

Use the following ... - This assumes the file contains no variable names and thus you need to specify the one variable name that will be used to contain the text information in the specified file. This is typically used to bulk load in a string or sequence of numbers from a file into a single variable.

First row is variable names - Specifies that the first row in the text file contains the variable names to be used to contain the data in the subsequent rows. This format is typically for CSV (comma separated files) that are typically produced from programs like Excel. This format is effective for variables that have lists of data.

First column is variable name - Specifies a file format whose first column is the variable name followed by a delimiter and the actual variable value. This structure is similar to the INI files found in Windows and is effective for one-off variables that contain a single value/string. This is most commonly used as a configuration format.

3. **Row delimiter** - Specifies the delimiter (separator) that indicates the start of a new row (record). This will typically be CRLF (linefeed, carriage return) for most text files but may differ occasionally.

4. **Column delimiter** - Specifies the delimiter (separator) that indicates the start of a new column. This is typically a comma for CSV type files but might instead be a space for INI type files. (The variable name is followed by a space which is followed by the value which may include spaces too.)

5. **Text qualifier** - Specifies the symbol that groups text together disregarding the earlier delimiters. For example, a value in a CSV file may have a comma in it which would normally mean that a new column has been indicated unless the value is surrounded by quotes. The quote is the text qualifier which ensures that the value is specified outside of the row and column delimiter criteria.

Example

For the following data file:

```
X, Y, Z
100, 100, 100
200, 200, 200
300, 300, 300
```

Specify that the first row contains the variable names, the row delimiter is CRLF, the column delimiter is a comma and the text qualifier is not used. This would create three variables with a list of 3 items each.

Next:

```
X 100
Y 200
Z 300
```

Specify that the first column contains the variable names, the row delimiter is CRLF, the column delimiter is a space and the text qualifier is not used. This would create three variables each with a single value.

Next:

```
option1 some option text
option2 some more text
```

Specify that the first column contains the variable names, the row delimiter is CRLF, the column delimiter is a space and the text qualifier is not used. This would create two variables each with a single line of text. In this case even though the column delimiter is set to a space the values of the variables are NOT split. In other words only the first column delimiter is respected when using this style of data file.

Next:

100,200,300

Specify a variable name such as `LOADED_VAR`, the row delimiter is empty, the column delimiter is a comma and the text qualifier is not used. This would create a single variable (`LOADED_VAR`) with three values.

Next:

this is some text.

Specify a variable name such as `LOADED_VAR`, the row delimiter is empty, the column delimiter is also empty and the text qualifier is not used. This would create a single variable (`LOADED_VAR`) with a single text string.

See Also

[Write Variables](#)

RTSP Player



The RTSP Player module uses the Open Source FFmpeg RTSP player to play RTSP network streams with minimal latency. This is accomplished by removing any buffering that typically is used in streaming players to smooth out image streaming. While useful in streaming pre-recorded files, these delays introduce huge problems when actions need to be taken due to what is present within a transferred image.

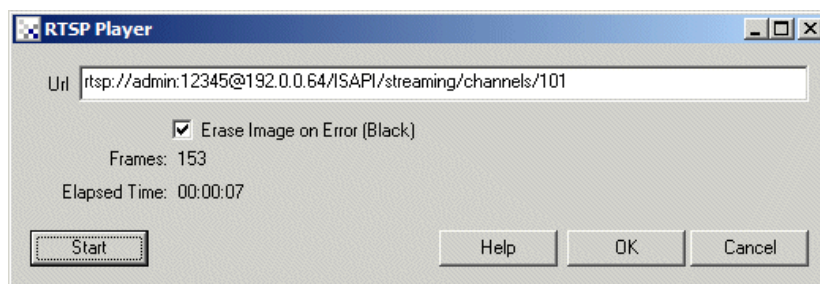
FFmpeg ***MUST*** be installed in order for this module to work. Please download the shared 32-bit FFmpeg libraries from the [Zeranoe 32bit FFmpeg](#) or [Zeranoe 64bit FFmpeg?](#) website. Please note, chose 32bit or 64bit to match RoboRealm's bit size NOT your system size. 32 bit applications can only load 32 bit DLLs. 64 bit applications can only load 64 bit DLLs.

To extract any of the builds you will need to download and install [7-Zip](#). Once extracted navigate to

```
C:\Users\Your_Username_Here\Downloads\ffmpeg-20150610-git-913685f-win32-shared\bin
```

and copy all the .DLL files from this folder into the RoboRealm folder or your `c:\Windows\System32\` folder or to `C:\Windows\SysWOW64\` folder if you are running RR x32 on a x64 system

Interface



1. Url - Specify the url to stream in the URL textbox.
2. Erase Image on Error - When selected this will zero out the current image when a network timeout occurs.
3. Low Delay - For even lower latency you can select the Low Delay checkbox. Note, this may cause artifacts in larger images that cannot get the data quick enough to decode.
4. Start - You **MUST** press the "Start" button to begin playing. Once pressed you can also manually stop the playback by pressing "Stop" (the "Start" button will change to "Stop" once playing starts).

Note that you must press Start for the playback to start streaming. Simply pressing OK will NOT begin playback. Also be aware that for Internet based streams you need to wait about 20-30 seconds after pressing the start button for playback to actually start.

Variables

VIDEO_SECONDS - specifies how long in seconds the current stream has been playing

VIDEO_TIME - specifies how long the current stream has been playing

See Also

[VLC Player](#)
[Media Reader](#)
[Read AVI](#)
[HTTP Read](#)

Screen Capture

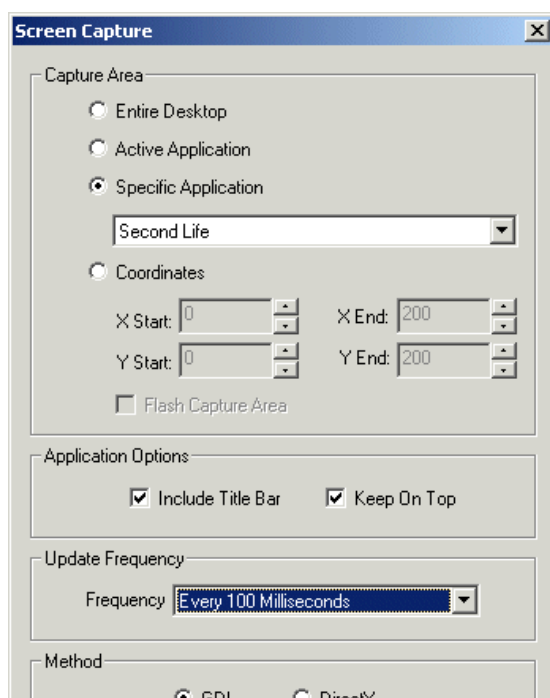
The screen capture module provides you with the ability to capture the video output of any application that has a display on your computer screen and process that output within RoboRealm as if it were just another webcam video stream. For example, the very popular virtual environment Second Life can be used as a simulator to test out computer vision algorithms without needing a robot or scenery to physically exist. The benefits of using simulators during robotic development has been well known (see the [MSRS simulator](#)) but having multiple games and simulators act as robotic hosts allows one much more flexibility in testing and designing for the physical world. Plus games already have a particular goal surrounding the environment and many human individuals to contend with, learn from and stimulate challenging situations.

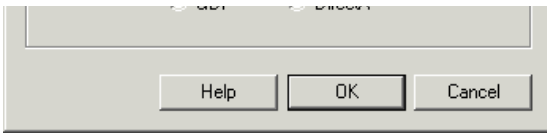
Including this module with the [Keyboard Send](#) or [Mouse](#) module provides a method to automate the interaction with virtual environments in a way that has previously not been possible without significant integration work.

While convenient, the capturing of application interfaces in this way is not ideal. If you elect to capture a specific application the application always needs to be at the foreground of the desktop otherwise the screen capture will show the other overlapping applications within the captured image of your application. In the GUI controls for this module you can specify to have your application always on top which should help to reduce this issue. Note that this module is best used with a very large screen where you can have the RoboRealm application and the desired to be captured application running side by side.

Note that applications that run within a Window will work best to automate. Games that switch resolution into fullscreen may not work correctly due to DirectX conflicts. Check your specific game as to how to run it within a Window. For example, using "halo -window" will run Halo within a window and allow the interaction between programming an automated character using RoboRealm much easier to do.

Interface





Instructions

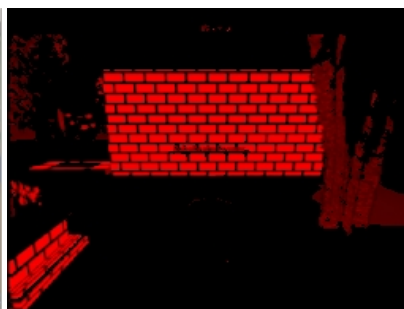
1. Entire Desktop - Select if you wish to capture your entire desktop and start processing it with RoboRealm.
2. Active Application - This selection will always cause the current focused application to be captured. Note that as you switch from application to application the captured image will adjust to the new window size. Also note that offscreen parts of the captured window will appear black.
3. Specific Application - Select the application you wish to capture. This will ensure that only the application specified will be captured. Note that if the "Keep on Top" checkbox is NOT selected the application may disappear behind other applications and the captured image will appear to be incorrect. This is not an error of the capture function but due to the Z-buffering that is used in your windowed desktop environment. To ensure that the window is captured completely ensure that the window is always fully visible.
4. Coordinates - Specify the specific screen coordinates that you wish to capture. This may or may not include several applications and will not adjust to follow moved applications as option #2 and #3 will do.
5. Flash Capture Area - Select this checkbox if you want to check what coordinate area you are capturing by flashing a box along the specified coordinates.
6. Include Title Bar - When #2 or #3 above is selected you can elect to not include the title bar as that is often not an essential part of the image to be captured.
7. Keep On Top - To ensure that option #3 only captures the selected application select the "Keep On Top" checkbox to force the specified application to always be on top. This causes other windows to move "under" the selected window ensuring that every capture does not include parts of other applications.
8. Frequency - As the computer screen does not have a specified "frames per second" you can elect the speed at which the frames should be grabbed from the screen. If you specify a very very high frequency your computer may become sluggish or stop due to the high CPU requirements for the capture. Normally, a capture around 100 milliseconds which translates to roughly 10 fps should do the trick between keeping the frame motion smooth and your CPU rate at a workable speed.
9. Method - Select which method to perform the screen capture. The GDI or Graphics Device Interface should work well with most windowed applications whereas the DirectX method may be needed for some applications or games that change video resolutions.

Example

Second Life Screen Capture



Processed for Red



See Also

[Mouse](#)
[Keyboard_Send](#)

SQL Interface

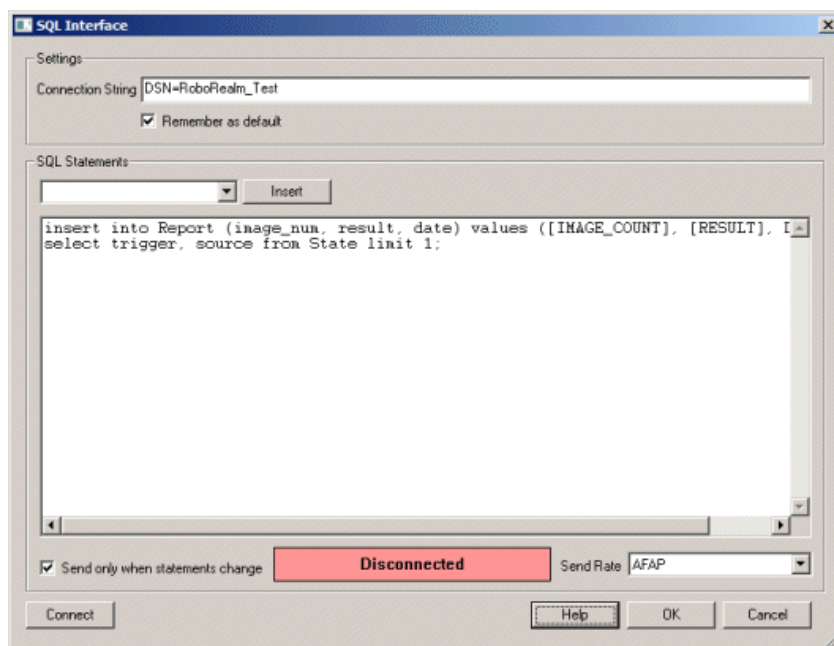


The SQL Interface module provides access from RoboRealm into an ODBC compatible SQL database. Using the provided interface you can execute statements against an ODBC database to insert RoboRealm variables into SQL tables and read back information from the database into RoboRealm.

NOTE: As this module runs each time an image is processed, it is very likely that you can fill a table with many values very quickly. Care should be

take to determine when a value should be written either by selecting the "Send only when statements change" or by surrounding this module with the [If Statement](#) module that will prevent execution on every pipeline iteration.

Interface



Instructions

1. Connection String - The connection string is where you specify how you want to connect to your database. If you already have a DSN setup you can specify DSN=name to utilize that DSN to connect.
2. Remember As Default - Select the "Remember as Default" checkbox if you would like the current Connection String to be remembered by RoboRealm such that whenever the module is inserted into the RoboRealm pipeline the Connection String will be auto-populated to the current setting. This ability allows you to not have to constantly type in the Connection String setting when loading in successive RoboRealm robofile configurations as long as the connection properties remain the same.
3. SQL Statements - Enter in the SQL statements that you want to execute. This interface supports multiple SQL statements separated by a semicolon (;) and can include INSERT, DELETE, UPDATE and SELECT statements. Any information returned by a SELECT statement is added into the RoboRealm variable namespace with a "SQL_" prefix. To populate SQL statements with RoboRealm variables you can include RoboRealm variables within [] which will be processed as specified in the [Expressions](#) page.

As SELECT statements can include field names that are not compatible with the RoboRealm variable naming, any character within the field string that is not a letter or digit will be replaced with an underscore. Thus if you query for count(*) it will be translated into SQL_count__ where the last __ are the translated (*) characters.

4. Insert - A convenience button is provided to allow you to specify a variable from within RoboRealm to be inserted into the Text editing. This provides no more functionality than just adding in the variable text name into the SQL Statement editor to ensure correct spelling.
5. Connect - Press the Connect button when you have specified the Connection String and SQL statements to execute. This will connect to the database and begin executing the SQL statements after parsing for [expressions].

6. Send only when statements change - To help reduce the load on the database you can chose to only execute the SQL statements when the values specified in the SQL statements change. This works for INSERT and UPDATE statements (assumed to be the more common use of the module) since they specify values to be saved to the database but will NOT work for SELECT or DELETE statements unless something in the query is changing (perhaps a WHERE id = [my_id]) otherwise the statements will never change their text and therefore not execute during successive pipeline iterations.

7. Send Rate - To further help reduce the load, you can select to only execute the SQL statements at a defined timing rate.

Examples

```
insert into Report (image_num, result, date)
values ([IMAGE_COUNT], [RESULT], NOW())
```

would insert the current image count (IMAGE_COUNT), the value of a variable called RESULT and the current date/time (MySQL function) into a Report table. Likewise,

```
select config_z, config_a from Config_Table limit 1;
```

would read in two configuration items (perhaps values to be used in processing modules) and enter them into the RoboRealm variable table as SQL_config_z and SQL_config_a. Finally,

```
select count(*) as total from reports
```

would insert the number of records in the reports table within a SQL_total (as apposed to SQL_count___) variable.

Variables

SQL_ERROR - Set when an error is encountered.

See Also

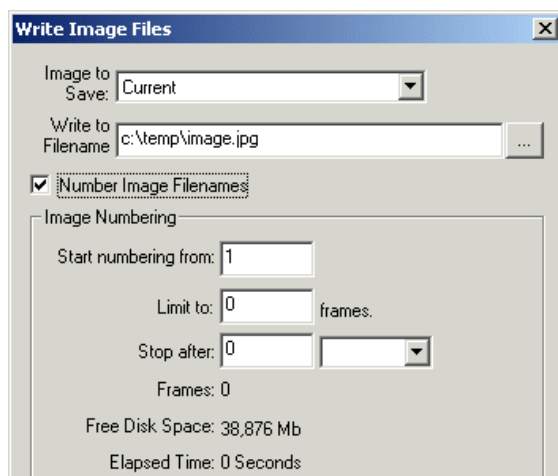
[Write Variables](#)

[Write Text](#)

Write Images

The Write Images component allows you save the processed image stream to separate jpeg image files. The files can then be used by other display or processing programs.

Interface





Instructions

1. Image To Save - Specify which image to save.

Source - the original image that was initially loaded or captured into RoboRealm

Current - the currently processed image within RoboRealm

Last - the last image processed by RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module.

The Marker labels represent images at the time markers were created. If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

2. Write to Filename - Specify the base filename to save the images to. For example if you specify c:\temp\image.jpg then the first image will be called c:\temp\image1.jpg, the second will be c:\temp\image2.jpg, etc. This can also be a variable of the form [my_variable] where my_variable is a variable that contains a fully qualified filename (see Notes below).

2. Options / Create Folder - if the folder doesn't exist it will be created if selected.

3. Options / Silent Errors - to avoid stopping the processing due to an error select the Silent Error checkbox. This will prevent popup errors from happening

4. Options / Jpeg Quality - if saving Jpeg images specify the quality to use. Default is superb. The lower quality settings will result in more artifacts within the image but result in much better compression (i.e. smaller file size).

5. Options / TIFF Compression - if saving TIFF images specify the type of compression to use within the TIFF file. The default LZW provides the good compression with standard usage, while the Jpeg option provides the best compression but may not work with some image editing/loading applications.

6. Image Numbering - If you want the first image to be c:\temp\image10.jpg then specify the "Start numbering from" to the appropriate value.

Notes

To stop the recording you have two options. You can limit to a set number of frames and/or stop after a set amount of time (whichever comes first).

To start generating frames press 'Start'. You can also manually stop the recording by pressing 'Stop'.

Note that any recording will terminate if less than 20 Megs are available on your hard disk to prevent the computer from running out of disk space.

You can also create a custom filename using one of the scripting modules like [VBScript](#). For example using [my_variable] in place of a filename and :

```
rr.SetVariable "my_filename", "c:\\temp\\my_image" & (rr.GetVariable("image_count") mod 1000 ) & ".jpg"
```

would cause the images to overwrite each other every 1000 images. This is useful for logging images but preventing too much disk space from being used.

Likewise

```
d = Now()  
stamp = Month(d) & "_" & Day(d) & "_" & Hour(d) & "_" & Minute(d)  
SetVariable "my_filename", "c:\\temp\\image_log_" & stamp & ".jpg"
```

will create a variable my_filename as

```
c:\temp\image_log_11_8_15_38.jpg
```

which is a unique filename for Nov 8th, at 3:38pm. This will change every minute to a different filename. Using a variable as a filename and a bit of programming one can create any filename needed.

See Also

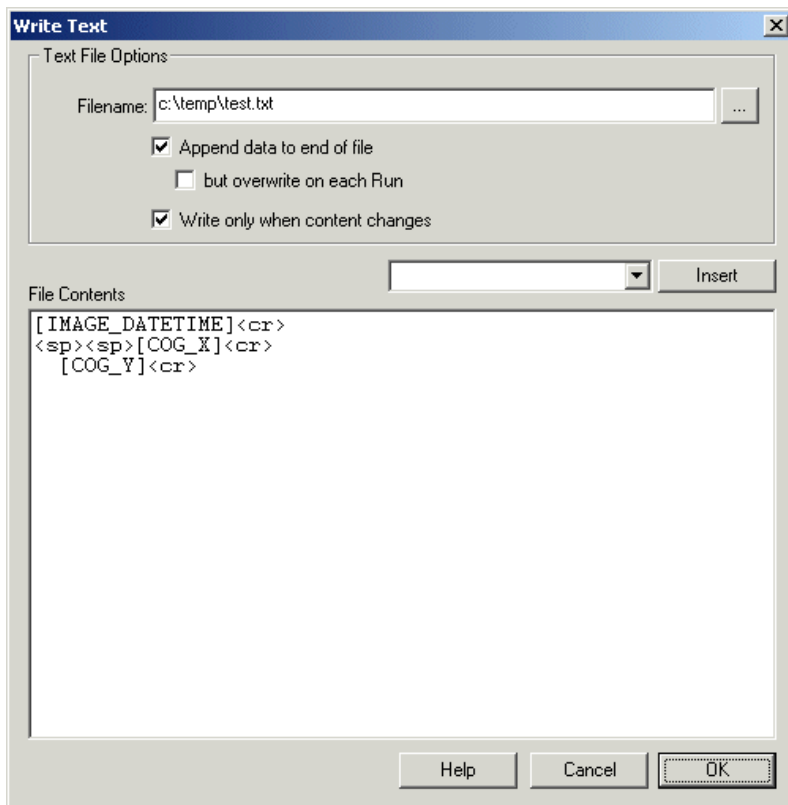


Write Text

The Write Text module provides a more freeform way of writing a text file to disk. This is similar to the [Write Variables](#) module but allows for a more customized format like those typically used in log files.

The provided text field allows for typing of variables and text to be included in the file. The format follows similar to the [Serial](#), [Socket Client](#), [USB](#) and [Execute Program](#) modules. You can type in text as normal and include variables using the [variable] notations.

Interface



Instructions

1. Filename - specify the filename you wish to write the text too. This will typically be a .txt extension but can be any you chose.
2. Append data to end of file - select if you wish to append the text data to an existing text file. If this is unchecked the file will be cleared before any data is written.
3. but overwrite on each Run - this will ensure that the file is cleared/tuncated each time the Run button is toggled.
4. File Contents - specify the text that you would like to write to the text file. For example, if you specify

```
[COG_X], [COG_Y]<cr>
```

then the two COG X and Y variables will be logged to the text file. This is functionaly similar to the [Write Variables](#) module. However, you can also specify more flexible formatting such as

```
[IMAGE_DATETIME]<cr>  
[COG_X]<cr>  
[COG_Y]<cr>
```

See the [Text Formats](#) page for additional information about the text string format.

See Also

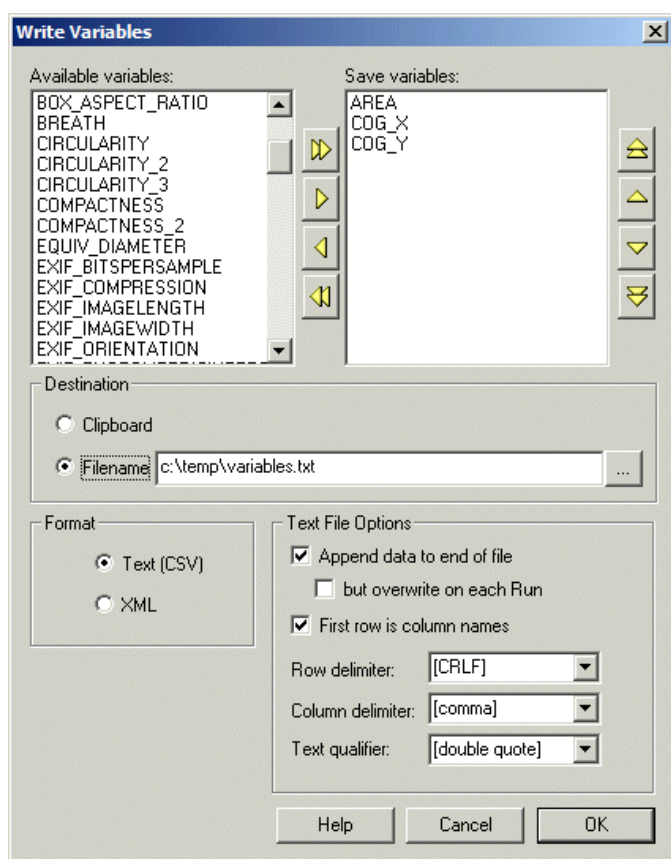
[Write Variables](#)
[Write Clipboard](#)

Write Variables

This module provides an interface to write RoboRealm variables to disk. The interface allows the selection of the variables that will be saved. Note that the order specified in the interface will be the order saved on disk.

This is the simplest way to export variables from RoboRealm. For more advanced, efficient and interactive methods see the [Plugins](#) documentation.

Interface



Instructions

1. Available Variables - Select the variables that you want to save. The left list shows all the current RoboRealm variables available to save given the current processing pipeline. Use the arrows (or doubleclick on a variable) to move it to the 'save variables' list. Use the up/down arrows to reorder that list.
2. Clipboard - If you want to write the values to the Windows Clipboard select this radio button. Note that you can test this by pasting into a text editor to see the values RoboRealm is placing into the Clipboard. Be aware, this overwrites any existing text in the clipboard. Variables are written as name value pairs with each on its own line. Thus you can use a newline (\bLF or \n) to break the text string into individual parts.
3. Filename - Specify which filename you would like the variable values to be saved to. Be sure to have created any needed directories/folders along the filename specification otherwise the module will NOT be able to update the file.
4. File Type - If you chose a text file select if you'd like the variable values to be appended to the end of the file. If chose append you should be ready for the file to become really large really quickly!

5. Text File Options - Specify the appropriate row, column and text delimiters. The text delimiters are used when a variable containing a text value (non-numerical) is saved. You can also type in characters to use as delimiters. Using the default delimiters will allow you to load the file directly into Excel or other spreadsheet programs.

You can choose to append data to a single file (but erase the file between runs) or just overwrite the entire file out each time the frame is processed.

Note that when reading the saved file you will need to check for the end of row delimiter (text file) or the </ROBOREALM> tag (xml file) to ensure that you have read a complete file. Due to the speed of the writing (once per frame) you are not assured that reads are atomic. RoboRealm is configured to write data in one write to help reduce this issue.

See Also

[Write Images](#)

[Plugins](#)

Video Recorder

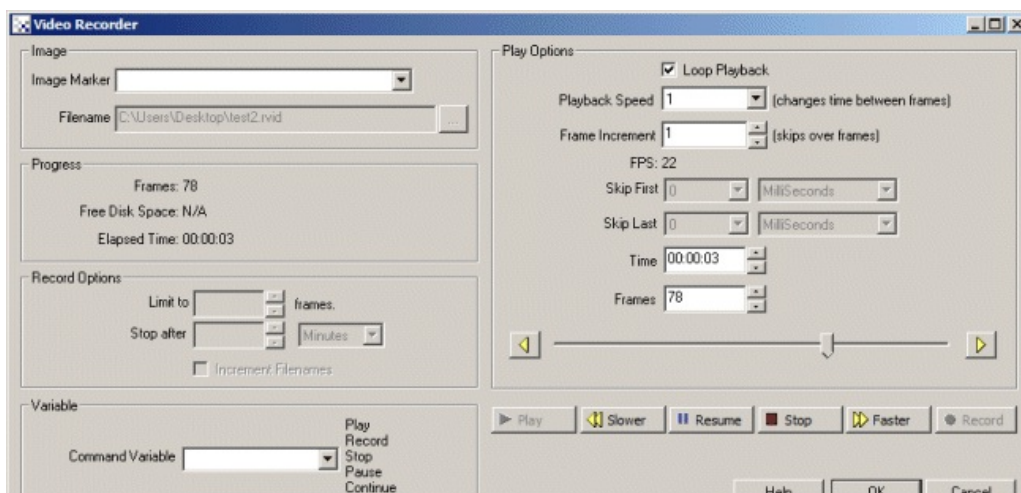


The Video Recorder module provides a similar function to the Write_AVI/Write_Images and Media_Reader/AVI_Reader in that it will record and playback video. The difference between this module and the others is that the recorded video:

- is lossless - There is no data loss between recording and playback to ensure that any processing done while recording can be reliably repeated during playback. This is crucial when testing for errors that may happen during the course of system testing that are hard to repeat.
- contains ALL color - Typically video recording encoders will remove a lot of color in order to save space. This module includes all color information to allow for color processing to proceed as if working with the direct source stream.
- is fast - Its not always possible to record on high end machines that have ample processing ability to keep up with MPEG or even JPEG compression. Having the ability to record using lower end tablet devices but maintain the video quality and speed is important for system testing.
- saves every frame - Current video encoders may drop frames if encoding or playback cannot keep up with realtime speeds. This module will ensure that every frame is recorded AND played back to ensure that the playback is exactly the same as the source video stream.
- does NOT include audio - As imaging applications rarely use audio as part of the application function no audio is recorded using this module.
- includes random access - Many video encoders will only record keyframes (full video images) every couple of frames to increase compression. This can cause issues when requiring the ability to seek to a specific frame or move backwards through video.
- plays and records without additional codecs - This module ensures that all those capabilities are provided without having to install many codecs (encoder/decoder) drivers to test quality and features.

An unfortunate side effect of these requirements is that the recorded video will be *MUCH* larger than typical video files meant for Internet distribution. If your intent is to use this recording in other applications or for online distribution do *NOT* use this module. While you can always convert this video format to other formats this module records video in a proprietary format only for use with RoboRealm and sacrifices disk space in order to ensure the best quality.

Interface



Instructions

1. Image Marker - The image to play into or record.

Source - the original image that was initially loaded or captured into RoboRealm

Current - the currently processed image within RoboRealm

Last - the last image processed by RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created.

If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

When recording, the marker is the image to record. When playing the marker is the image to update. If playback is into a marker the image will NOT appear as the current image.

2. Filename - Specify the file to open/save. Note that the file extension is expected to be .rvid.

3. Loop Playback - If you want to loop the playback then check the "Loop Playback" checkbox.

4. Relative Frame Delay - To change the playback rate of the video select the appropriate playback speed. Numbers lower than 1 reduce the speed while numbers greater than one increase it. This reduces or increases the time delay between frames.

5. Absolute Frame Delay - Replaces the time delay between frames to the specified amount. This ignores the recorded delay to allow you to playback video at a different rate than recorded. For example, if you recorded a single frame very hour you can playback the video at 30 frames per second using a 33 millisecond delay.

6. Frame Increment - If you want to skip over the video and only process every other frame set the increment to 2. This will cause every other frame to be skipped.

7. FPS - Displays the current Frames Per Second that the video is recorded in.

8. Skip - To Skip the first or last part of the video configure the "Skip First" and "Skip Last" accordingly. When the video loops the first X seconds, milliseconds, etc. will be jumped over. Similarly the video will end X seconds before the actual end of the video. Note that this is not cropping or modifying the video in any way.

9. Time - Displays the current frame time of the video. Changing this value will move the video to that time point.

10. Frames - Displays the current frame number of the video. Changing this value will move the video to that time point.

11. Sliderbar - Shows the current frame location within the video. Clicking or dragging this toolbar will move the video frame accordingly. You can also use the buttons surrounding the slider bar to advance or retreat a frame at a time.

12. Play - You MUST press the "Play" button to begin playing. Once pressed you can also manually stop the playback by pressing "Stop". You can also Pause the video and use the Time, Frames or Slider controls to move the video to a point of interest.

Note that you must press Play for the playback to start showing. Simply pressing OK will NOT begin playback.

13. Number Files - To avoid overwriting files that already exist the number files checkbox will append a number to the end of the filename if the file already exists. This helps to avoid destroying existing data.

14. Limit to - To stop the recording you have two options. You can limit recording to a set number of frames and/or stop after a set amount of time (whichever comes first).

15. Stop after - Specify the time once lapsed to stop recording.

16. Record Every - If you only need frames every X seconds you can use this configuration to change how often frames are recorded to video. This allows you very fine control over a time lapse functionality that can help reduce file size if realtime recording is not needed.

17. Increment Filenames - To avoid overwriting files that already exist, the increment filenames checkbox will append a number to the end of the filename if the file already exists. This helps to avoid destroying existing data and record multiple files with minimal changes.

18. Command Variable - To automate the starting and stopping of video recording (i.e. through the API) you can specify a variable whose value will contain either Play, Record, Stop, Pause or Resume as a command word which will cause the module to perform the appropriate action. Note that once read the variable will be set to blank to avoid repeating the same action more than once.

Variables

VIDEO_FRAME - specifies the frame number of the current video image.

VIDEO_SECONDS - specifies in seconds where the current video frame is within the video.

VIDEO_TIME - specifies the HH:MM:SS (hours:minutes:seconds) where the current video frame is within the video.

See Also

[VLC Player](#)
[Read AVI](#)
[HTTP Read](#)

VLC Player

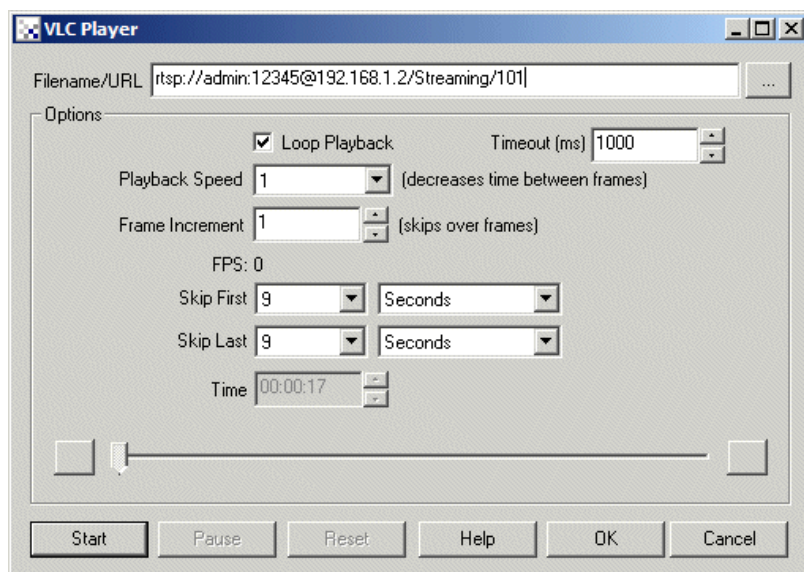


The VLC Player module uses the Open Source VideoLan VLC media player to play video files within RoboRealm that can be processed by additional modules.

Note that with the addition to video files you can also hook into image streams broadcasts over the Internet in RTSP format. This is similar to the [Http Read](#) component but can accept a different streaming format.

VLC *MUST* be installed in order for this module to work. Please download VLC from the [VideoLan](#) website.

Interface



1. Filename/Url - Specify the file or url to read in the "Filename/URL" textbox. The VLC player can accommodate .avi, .wmv, and .mpg files to name only a few of the supported formats.

2. Loop Playback - If you want to loop the playback (only valid for files) then check the "Loop Playback" checkbox.

3. Timeout - On playing videos being streamed over the network a delay is needed in order to allow for network speed fluctuations. If you are playing videos that are stored locally you can reduce the timeout delay which will also be noticed when the video loops. If you are playing files over the network and the timeout is too small the video may accidentally restart as a result of a timeout. Increasing the timeout tells the VLC module to wait longer until a new frame arrives over the network before resetting.

3. Playback Speed - To change the playback rate of the video select the appropriate playback speed. Numbers lower than 1 reduce the speed while numbers greater than one increase it. This reduces or increases the time delay between frames.

4. Frame Increment - If you want to skip over the video and only process every other frame set the increment to 2. This will cause every other frame to be skipped.

5. FPS - Displays the current Frames Per Second that the video is recorded in.

6. Skip - To Skip the first or last part of the video configure the "Skip First" and "Skip Last" accordingly. When the video loops the first X seconds, milliseconds, etc. will be jumped over. Similarly the video will end X seconds before the actual end of the video. Note that this is not cropping or modifying the video in any way.

7. Time - Displays the current frame time of the video. Changing this value will move the video to that time point.

8. Start - You MUST press the "Start" button to begin playing. Once pressed you can also manually stop the playback by pressing "Stop" (the "Start" button will change to "Stop" once playing starts). You can also Pause the video and use the Time or Slider controls to move the video to a point of interest.

Note that you must press Start for the playback to start streaming. Simply pressing OK will NOT begin playback. Also be aware that for Internet based streams you need to wait about 20-30 seconds after pressing the start button for playback to actually start.

Note that if RoboRealm appears to freeze the connection may be down. Wait for about 20 seconds and you will regain control after the network connection times out.

Variables

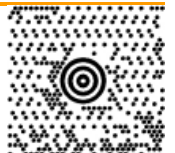
VIDEO_SECONDS - specifies in seconds where the current video frame is within the video.

VIDEO_TIME - specifies the HH:MM:SS (hours:minutes:seconds) where the current video frame is within the video.

See Also

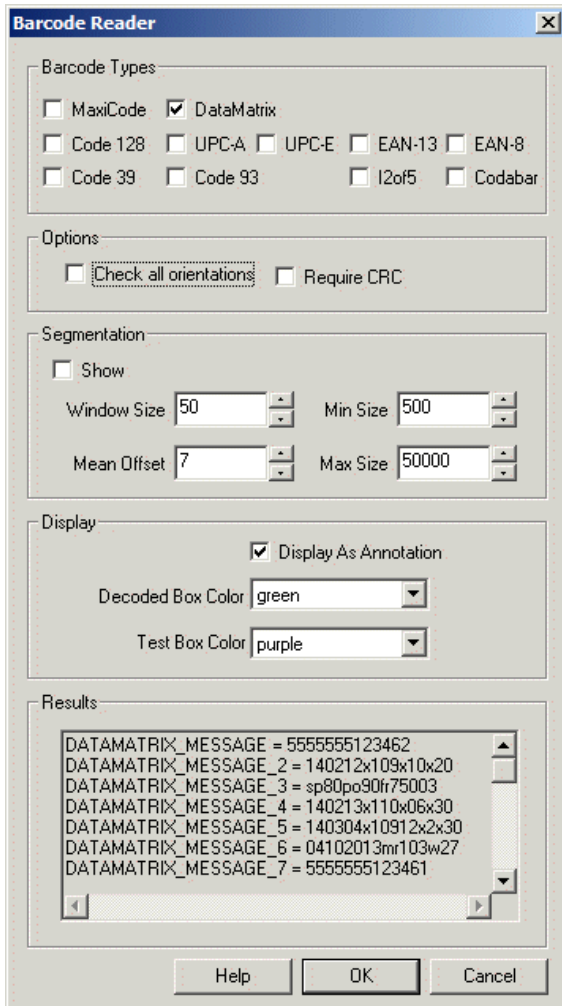
[Media Reader](#)
[Read AVI](#)
[RTSP Player](#)
[HTTP Read](#)

Barcode Reader



The Barcode Reader module provides a way to detect and decode barcodes that are visible in the current image. If you would like to use RoboRealm for additional barcode detection please let us know and we will add the particular barcode type to this module. We only add barcode types on customer request.

Interface



Instructions

1. Decode Box Color - select the color of successfully detected and decoded barcodes.
2. Test Box Color - select the color of barcodes there were detected but could not be decoded.
3. Barcode Types - (2D) MaxiCode, (1D) Code 128, UPC-A, UPC-E, EAN-13, EAN-8, Code 39, i2of5, Codabar

MaxiCode is a fixed-size (1.11 inch x 1.054 inch nominal) two-dimensional symbology made up of offset rows of hexagonal elements around a unique circular finder pattern. A MaxiCode symbol has 884 hexagonal modules arranged in 33 rows with each row containing up to 30 modules. The maximum data capacity for MaxiCode is 93 characters. MaxiCode is used by United Parcel Service (UPS) for package tracking.

While any data can be contained in a MaxiCode the typical UPS MaxiCode contains a Postal code, Country Code and Service mode in addition to a string of text that contains additional information such as street address. Because the MaxiCode has two error correcting modes it is possible to detect the Postal, Country and Service codes without being able to detect the rest of the street address. RoboRealm indicates this situation using the MAXICODE_MESSAGE_RAW variable which provides the rest of the textual information that could not be correctly decoded.

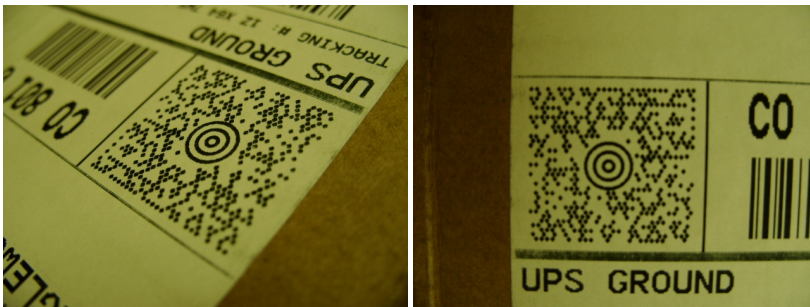
Note that due to the complexity and detail required for a MaxiCode you will need to place the webcam close to the MaxiCode in order to successfully read the code. We recommend a minimum resolution of 640x480 in order to ensure that enough detail is represented in order to decode.

The 1D barcodes are traditionally seen on products that one would typically buy at a store. Most of them include a CRC digit which helps to

reduce accidental recognition. Code 39 does NOT have a CRC check and therefore can cause many false matches. In this case you may need to create additional filters prior to the barcode module to isolate just the barcode.

4. Check all Orientations - Some 1D barcodes can be checked to exist in multiple orientations. Selecting this checkbox will check multiple directions for the presence of barcodes. Note, this slows processing speed.
5. Require CRC - Some 1D barcodes have optional CRC checking. This can help to ensure correct decoding of barcodes but will remove partial decoding of some barcodes.
6. Show Segmentation - Embedded within the module is an adaptive threshold method to isolate 2D barcodes like DataMatrix. Selecting this checkbox will show you what the current segmentation parameters are working with.
7. Window Size/Mean Offset - Parameters exposed from the [Adaptive Threshold](#) technique
8. Min/Max Size - Specifies the size of the bounding box of the code that will be considered. Those objects with size less than min and greater than max will be excluded from the decoding process. This filter can be used to speed up the decoding process by eliminating objects early in the analysis process.
9. Display As Annotation - Select if you want the graphic boxes to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
10. Decoded Box Color - The color that is used to indicate a correctly decoded barcode.
11. Test Box Color - The color to use to indicate a possible barcode shape but with an incorrect decoding (i.e. the CRC or other failed).
12. Results - The detected codes. Note, these are also created as RoboRealm variables (see [Watch Variables](#) module) that can be used in other modules.

Test MaxiCodes (right click and select "Copy image" to copy these images and paste use CTRL-V to paste into RR)



Variables

MAXICODE_POSTAL_CODE - the zip code destination for the package

MAXICODE_COUNTRY_CODE - the country code (840 for the USA)

MAXICODE_COUNTRY_NAME - the country code expanding into a readable name.

MAXICODE_SERVICE_CLASS - the package service class

MAXICODE_MESSAGE - other information include street address

MAXICODE_MESSAGE_RAW - contains the raw message if it could

not be decoded correctly

DATAMATRIX_COUNT - Number of recognized datamatrix codes

DATAMATRIX_MESSAGE - the content of the datamatrix code

DATAMATRIX_ORIENTATION - the detected orientation (degrees) of the code

BARCODE - the first decoded 1D barcode information

BARCODE_X - the X decoded 1D barcode information

See Also

[Fiducial](#)

Blob Inspection

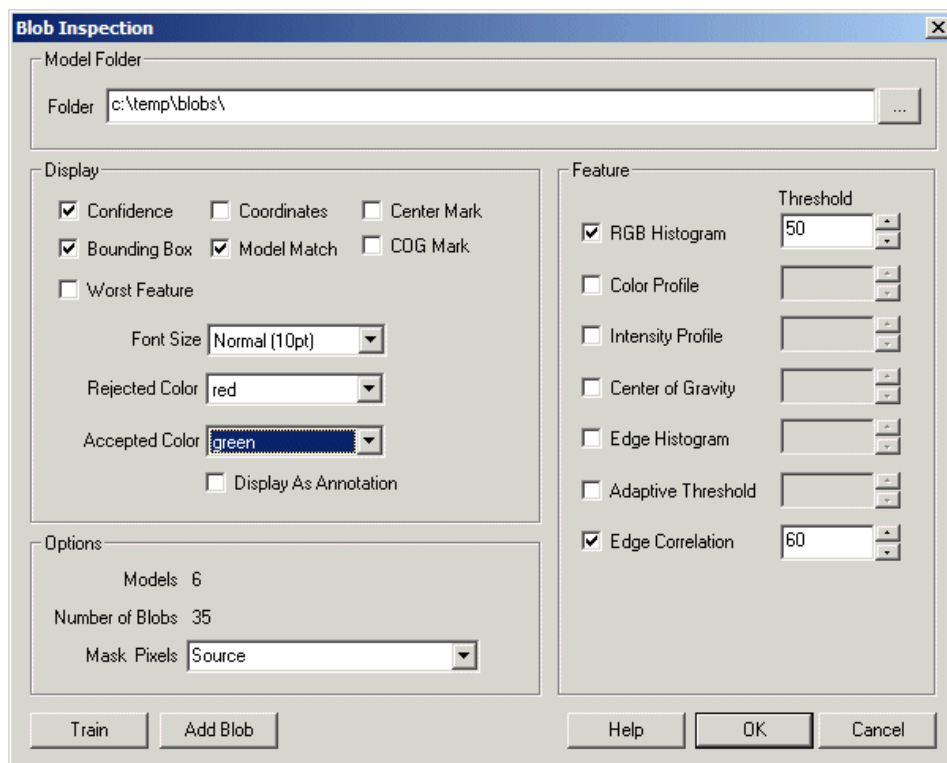


The Blob Inspection module is meant to be used with verification systems that require comparing new objects with model or template objects. This module allows you to specify a database (collection of image files on the filesystem) of golden or ideal images that are compared in various ways to new blobs. There are several features which can be used that highlight or ignore aspects of the object to allow for some features to be very rigid while others to be more flexible. This is necessary as each object may be allowed variance in one aspect but not another.

Similar to the [Blob Filter](#) module, this module has various features that it creates given a specified blob (collection of pixels) and the associated raw pixel image (source). By using the binary blob image it collects statistics from the source image and compares these statistics to the model images to determine similarity. If any of the features falls below a specified threshold, that object is considered rejected and does not pass the verification process.

Using multiple features one can create a verification system to ensure that quality is maintained for a certain aspect of a product along an assembly line.

Interface



Instructions

1. Pre-Requisite - Ensure that you have loaded an image (most likely using [Load Image](#)) that is black and white with white being the signal/blobs/object that you want to run through the verification process. Typically this means using the [Threshold](#) or [Auto Threshold](#), [Adaptive Threshold](#), etc. modules to create a mask image that indicates where the blobs are within the current image that you want to verify. Note, the dimensions of the mask image and the current image MUST be the same as they are simply overlays and need to match. The image entering this module would be the raw colored pixels.

2. Model Folder - Specify where the module has or will store all the model images that are created or will be created. The model folder specifies where this module should find its model database/images that constitute the standard to which all other blobs are compared against. Note, this folder should be writeable by you.

3. Add Blob - Select this button to view the current image and in order to select which blobs/objects are considered ideal or model examples of what you are inspecting. Once selected that blob will change color and after pressing OK will be added to the model database and saved in the Model Folder. Note, images within the model folder are regular images and can be loaded in any image editor. The major difference is that the non-object areas are colored in pure green (0,255,0) which indicates the Mask or non-object areas. Its important to preserve these areas to avoid causing the background of an object to be included in the analysis.

4. Feature - Select which features you want to use for the comparison process. You can enable a particular feature by clicking on the associated checkbox and specifying a threshold value. All objects that fall below this threshold when compared to the images in the model database will be considered rejected. If you are unsure of the value, specify a non-zero value and then see which objects are passed versus failed and their values.

By quickly scanning the results you can get a good idea of if a particular feature is useful or not in segmenting the pass versus fail objects.

RGB Histogram - Specifies that the histogram (summation of) raw RGB values of the object will be compared to the models. This comparison will ensure that the relative amounts of a particular color are also in the blob in question. Because it only compares the summation of the color values, change within the blob is possible without adversely affecting the comparison value. For example, a Pepperoni Pizza has about the same amount of red and yellow colors as other Pepperoni Pizza's do while the Pepperoni can be in different locations on each pizza. However, when compared to a Margarita Pizza, the absolute amounts of yellow and red will be very different. Note that the RGB Histogram will be sensitive to light levels too.

This can be used to eliminate objects of different color and intensity than the templates.

Color Profile - Specifies that the histogram of colors is used as a comparison metric. Similar in comparison to the RGB Histogram but with a less lighting sensitive metric. As each pixel's color is only used the dependency on absolute intensity is reduced. Note that as pixels become darker the actual color represented within the pixel is also reduced and typically favors blue. Thus at extreme light and dark levels color is absent.

This can be used to eliminate objects of different color than the templates.

Intensity Profile - Specifies that the histogram of pixel intensities be used as a metric. This is opposite of the Color Profile in that only intensity values are used with color being ignored. Combining the Intensity Profile with the Color Profile will result in the same comparison as the RGB Histogram

This can be used to eliminate objects of different intensity than the templates.

Center of Gravity - The center of gravity or center of mass is calculated for each object and compared to the distance from the center of the object. The center of gravity takes into account edge intensities such that more textured parts of the object will exhibit more 'pull' towards that direction. The pixel center is defined as the center of the object disregarding the edge values. The distance between these two points is often a good indication of where a label/graphic/text is located.

This can be used to eliminate objects with un-centered labels.

Edge Histogram - The edge representation of the object is calculated to highlight sudden pixel changes. The magnitude/strength of these pixel changes can be compared from one object to another while disregarding the actual location of these sudden pixel changes. This will highlight differences between objects where the label/graphic/print is not as light or dark as the model images are.

This can be used to eliminate objects where labels/graphics/text is different than the templates.

Adaptive Threshold - The [adaptive threshold](#) algorithm is run over the object to produce an image that indicates points that are much lighter or darker than their surroundings. This image will segment areas that are different from their surroundings. The histogram of this resulting image is then compared to other model images to determine texture similarity with the model images.

This can be used to eliminate objects where labels/graphics/text is different than the templates.

Edge Correlation - The edge representation of the object is calculated, aligned to eliminate rotational dependencies and then compared with the model images. This feature directly compares in a template like fashion text/labels/print against the models to ensure that they are point by point similar to the model images.

This can be used to eliminate objects where labels/graphics/text is different than the templates.

Low Intensity Area - The object intensity changes are split into two groups based on the most frequent intensity value. The Low Intensity Area represents the number of pixels that are below this most frequent value peak. It best represents a count of outliers that are the least frequently occurring intensity values.

This can be used to eliminate objects where labels/graphics/text is different than the templates.

High Intensity Area - A count of the other side of the frequency peak. I.e. the count of the least frequent pixels above the most frequent peak.

This can be used to eliminate objects where labels/graphics/text is different than the templates.

Intensity Correlation - A straight pixel to pixel comparison of the template images to the current object.

This can be used to eliminate objects where objects are overall different than the templates.

X/Y Offset - Specifies how far off from center the inner parts of the object are. The inner parts of the object are identified as significant edges but does NOT include object borders. Note that even if the object is rotated, the offset will be normalized with respect to the object inner edges. This ensures that comparisons between different rotated objects can be done.

This can be used to eliminate objects where labels/graphics/text are centered differently than the templates.

5. Source - If you are getting your image from a different you may need to specify the Image [Marker](#) that contains the source pixels. Both the binary image that indicates a blob/object and the raw pixel values have to be in the same image location as the binary image is used as a mask over the raw pixels. This allows a blob to be specified in many different ways.

6. Display - Select which options you want for display purposes. This will annotate the current image with the results generated from the comparisons.

Confidence - The lowest feature confidence. When this confidence value falls below the specified threshold in the Feature list the object will fail. Note, this confidence value is the *lowest* confidence value and thus the feature that is the most likely to cause the object to fail.

Worst Feature - Specifies which feature produced the lowest confidence.

Model Match - Shows the model number that produced the lowest confidence value (i.e. the model which the current object is least similar to).

Coordinates - The center of gravity of the object in coordinates (useful for identification).

Center Mark - Show the center of the object.

COG Mark - Show the center of gravity of the object.

Bounding Box - Shows a bounding box around the object extents.

7. Rejected/Accepted Color - Indicates which colors to use to display the above information which indicates if an object was accepted or rejected.

8. Display As Annotation - Draw the annotations but only on the second pipeline pass so as not to affect subsequent processing due to the drawn graphics.

Variables

BLOB_INSPECTION_RESULTS - Contains the results of the inspection module. This is an integer array of values in blocks of 12 per result. The individual elements are

Offset Contents

0 Pass/Fail - Blob is accepted or not. 1: pass, 0: fail

1 Confidence - Confidence of blob associated to model (0 to 100)

2 Worst Feature - The feature that generated the lowest confidence value.

1 - RGB Threshold

2 - Color Profile

- 1 - Blob Filter
 - 2 - Blob Recognition
 - 3 - Intensity Profile
 - 4 - Center Of Gravity
 - 5 - Edge Histogram
 - 6 - Adaptive Threshold
 - 7 - Edge Correlation
 - 8 - Low Area
 - 9 - High Area
 - 10 - Intensity Correlation
 - 11 - X Offset
 - 12 - Y offset
- 3 Worst Model - The Index of the model that created the worst comparison
 - 4 Start X bounding box coordinate
 - 5 Start Y bounding box coordinate
 - 6 End X bounding box coordinate
 - 7 End Y bounding box coordinate
 - 8 X center of gravity
 - 9 Y center of gravity
 - 10 X blob center
 - 11 Y blob center

For example, to count all failed objects the VBScript code would be

```
blobs = GetArrayVariable("BLOB_INSPECTION_RESULTS")
for i = 0 to ubound(blobs)-1 step 12
    if blobs(i) <> 1 then
        count = count + 1
    end if
next

SetVariable "total_failed", count
```

Variable Commands

BLOB_INSPECTION_DELETE_ALL - Causes the module to remove all images and its cache `roborealms.inspect` file from the model folder.

BLOB_INSPECTION_ADD - Causes the module to add a new model to its model database based on the coordinate variables below. Note, this is as if a user clicked on a blob within the "Add Blob" button.

BLOB_INSPECTION_ADD_X - The X coordinate of the blob to add.

BLOB_INSPECTION_ADD_Y - The Y coordinate of the blob to add.

BLOB_INSPECTION_TRAIN - Triggers a retraining of the image database. (In case the model folder changed externally to RoboRealm).

See Also

[Blob Filter](#)

[Blob Recognition](#)

[Blob Size](#)

Blob Recognition

The blob recognition module allows you to train a detection engine on specific blob based on supplied training images. The supplied training images (in sets of 2) should consist of the "raw" pixel image as taken by a camera and the manually annotated "mask" image showing which parts of the image are to be recognized. The "mask" image is typically a binary image where black areas are concluded to be background data with non-black (normally white) pixels specify the locations of blobs that should be detected.

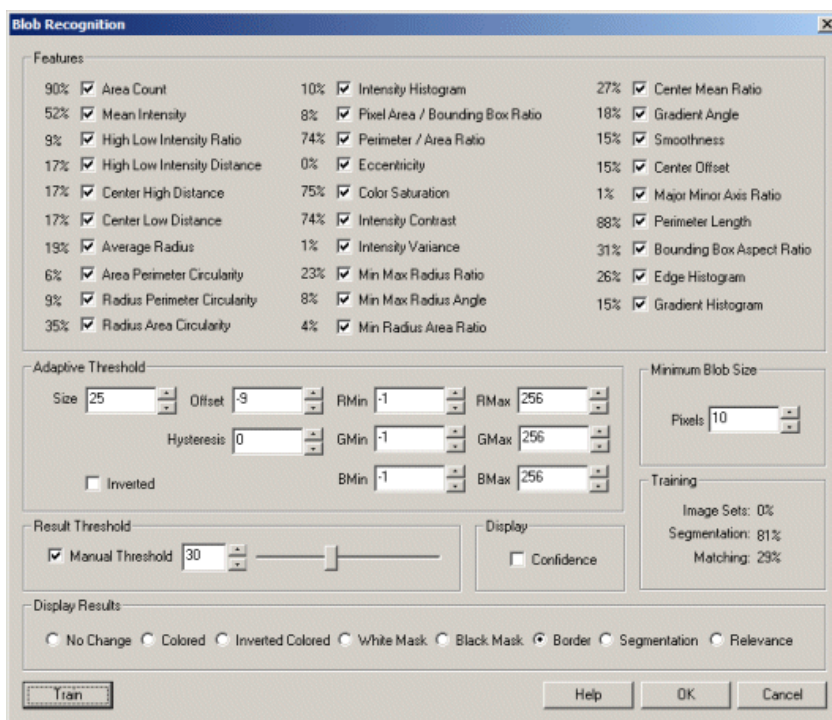
The module's interface allows you to specify where the training folders is located and which features are used in detection. Once training is complete the weight or relevance of that feature in performing the detection is displayed next to the feature. The module then utilizes this weight information to display recognized objects in subsequent images for testing purposes. The overall goal of the module is to automate the detection of trained blobs without having to determine the relevance of any specific feature such as that performed in the [Blob Filter](#) module.

The usage of the module first requires you to train on a set of images (one raw pixels and one binary mask) which will configure the threshold and other configuration values based on that training set. You can then make any changes as needed (most likely this is NOT recommended) and then just drag in images that you want classified. These numbers reflect the TRAINING set and NOT the current image since the current image will typically not have an annotated mask in which to compare the results to in order to determine the correctness of the classification.

Keep in mind the two different approaches to the system, training has the "answer key" but when a new image is run through you do not have an "answer key".

Once training is complete a ".blob" file is created in the source folder that contains the model generated from the training images. Once this file is created you can restart RoboRealm and not have to re-train in order to load in the classification model.

Interface



Instructions

1. Features - The Features list are those features calculated from the mask/annotated image using the raw image to actually determine the numbers. It is important to use the raw data for the statistics otherwise when a new image is shown it needs to work on that data and not on the mask. Thus it is VERY important that the raw image and the mask overlay exactly otherwise the statistics and verification will be wrong.

The checkboxes controls which, if any, features are used for identification for objects. If you uncheck one of the features, it will be removed from the analysis and not used to determine a valid or invalid object.

The percentages next to each feature reflects the relevance of that feature. It is an indication of how powerful/useful a feature is to distinguishing an object from a non-object. So if a feature is 0 then it is essentially useless in distinguishing an object from the background. If it is 100% then that feature is a perfect classifier in determining object versus background.

For example, if an animal has fur, does that uniquely identify a dog? Not quite, but it can be used to narrow the search by say 50%. Barking, that

may be 60%, and by combining fur and barking that helps narrow it down even more.

Following is a brief description of the features:

- Size/Area - the pixel count of the blob
- Mean Intensity - the average intensity value of each pixel within the blob
- High Low Intensity Ratio - The intensity difference between the average intensity below the mean and above the mean intensity.
- High Low Intensity Distance - The distance in pixels between the average intensity below the mean and above the mean intensity.
- Center High Distance - the distance between high cog and mean cog divided by the average ratio (to normalize)
- Center Low Distance - the distance between low cog and mean cog divided by the average ratio (to normalize)
- Average Radius - The average distance from the center of gravity of the object and each perimeter pixel
- Circularity - The ratio of a radius determined by the perimeter and the radius determined by the area of the blob.
- Center Offset - The distance in pixels between the gray level center of gravity and the binary (mask) center of gravity.
- Major Minor Axis Ratio - minor axis divided by the major axis as determined by moment analysis
- Perimeter Length - the number of pixels (count) of the perimeter of the blob
- Bounding Box Aspect Ratio - bounding box width divided by height
- Intensity Histogram - distance to trained 16 valued intensity histogram
- Pixel Area / Bounding Box Ratio - pixel area count divided by the size of the bounding box
- Perimeter / Area Ratio - perimeter count divided by pixel area count
- Eccentricity - A measure of circular versus oval shape
- Color Saturation - the maximum color value minus the minimum color value (assumes RGB image)
- Intensity Contrast - the minimum intensity value as compared to the maximum intensity value
- Intensity Variance - difference of pixel intensities around mean intensity
- Min Max Radius Ratio - The ratio of the minimum radius over the maximum as found when traversing the blob's perimeter
- Min Max Radius Angle - The angle between the nearest point and furthest point on the perimeter relative to the blob's center of gravity
- Min Max Radius Area - The ratio between actual area and the area defined between the min and max radius
- Center Mean Ratio - The ratio between the intensity value at the center of gravity relative to the mean intensity value of the blob
- Gradient Angle - Average Surface Gradient of blob (0-90 deg)
- Smoothness - The average difference in intensity within neighboring pixels
- Edge Histogram - distance to trained edge changes (pixel neighbors) in a 16 bucket histogram
- Gradient Histogram - distance to trained surface gradient (angle) in a 16 bucket histogram

The numbers next to each of the features are relevance or the feature's contributing weight. Basically the weight is calculated by determining how well that individual feature separates signal from background (i.e. egg versus garbage). If the signal and background have the same response then the weight is close to zero. If it does a good job then the weight will be near 100%.

2. Threshold - based on the match quality of all selected features the threshold can be used to eliminate those bad matches. This can be done automatically or manually depending on which features are used.

The threshold is for all combined features. It is the minimum weight necessary for a blobs combined feature weight to be classified as an object. The automatic value is determined by running the training set through the current classification weights and determines the best weight were most of the blobs are classified correctly. The manual threshold is just for testing how close object and background are in terms of classification. I.e. you can move the threshold around and see what is close to being object or not.

3. Display - There are various ways to display the match results including showing the actual confidence value of each blob. This can aid in determining the correctness of the match.

4. Train - The train button brings up an interface to select where the raw and mask combinations of training images are located in the filesystem. On specifying that the module will use those images to create a recognition model using all the selected features in order to discern between good and bad detections. The train button brings up the train interface that allows you to specify the folder that contains raw and mask images on which to train.

5. Segmentation - This implies how well the current threshold technique is at segmenting the current image into signal blobs. Any threshold will also cause a lot of noise or garbage blobs to appear. This number reflects how well candidate objects are extracted from the image. After candidates are identified the features are then used to classify and eliminate background blobs.

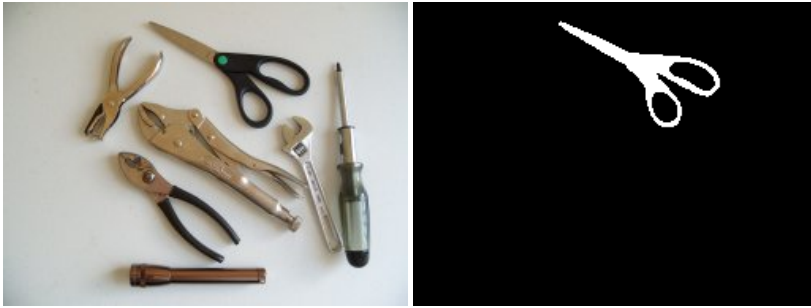
6. Matching - The final score of how well the blobs are classified as objects or background based on the training set.

Training Sets

The Blob Recognition module expects two images for a training set. One image is the raw pixel information as you would expect to see directly from a camera or imaging device. The second image should be a binary black and white image that has white pixel that depict where the blob you want to train on is located within the image. For example,

Source

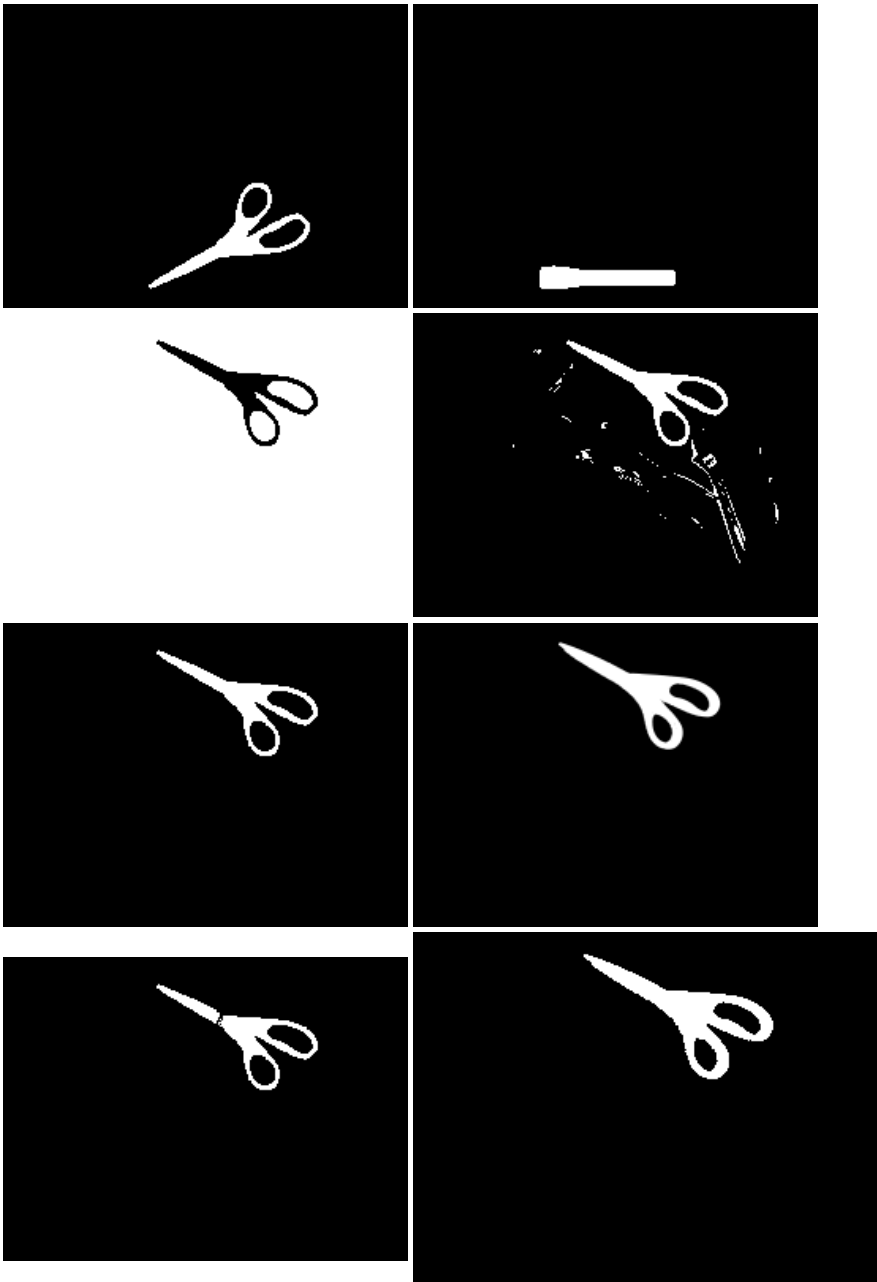
Mask



is a good training set. The mask image **MUST** align perfectly (pixel to pixel) with the original raw image. If this is even off by a couple pixels the training will not return good results. The following are **NOT** good masks.

Source

Mask



A couple points about creating a good mask:

- Be sure to align the mask with the original image. Often the easiest way is to paint over the original image (using your favorite paint program) to ensure that the mask will align with the original.
- You can include several of the objects in one image training set, just be sure that they do NOT touch.
- Ensure that the image is binary (black and white) and does not include grayscale pixels. Anything non-black will be treated as white.

Grayscale masks are not recommended.

- Use a non-lossy image format such as GIF, PNG, BMP, or TIFF for the mask. This ensures 100% mask accuracy and will not introduce noise into the image due to lossy compression (such as that used in JPG).
- The mask image MUST be exactly the same size as the raw image. Even if the object's size is the same, if the image is 1 pixel larger than the raw image the module will ignore the training set.

You will have to name the raw image and the mask image using the same filename (the extensions can be different though) and ensure that some term is present in each that can be used for identification. For example if your original file is called somethingABCD.jpg you will need to rename it to somethingABCD-raw.jpg for the raw file and create somethingABCD-mask.gif for the mask file. Note that addition of the words "raw" and "mask" into the filename. These terms help the module determine which file is which. These terms are then entered into the training interface (seen when the train button is pressed). It is important that most of the name elements of the filenames are approximately the same in order for the module to match raw and mask files.

See Also

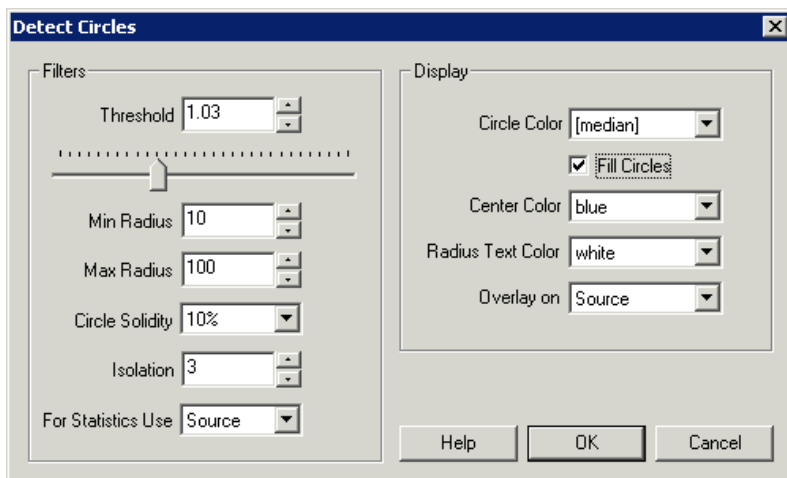
[Blob Filter](#)

Circles

The Circles module is a shape detection module that identifies contours within an image that are circular. The Circles module is meant to run after a contour based module (like [Canny](#), [DOB](#), [Sobel](#), etc.) that identify edges within an image. Using these edges the Circles module will determine which contours are part of a circular shape and identify that shape for successive processing.

Note that since the Circles module is contour based several edge processing routines can be applied prior to the Circles module to best create distinct contours.

Interface



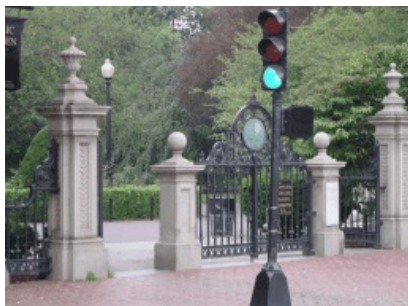
Instructions

1. Threshold - Select the circle threshold. The threshold value sets the sensitivity of the circle detector. The lower the threshold the more partial circles will be detected. The higher the value the more complete and defined the circles will need to be.
2. Min Radius - Select the minimum radius of detected circles. Any circles detected whose radius is less than the minimum radius will be eliminated.
3. Max Radius - Select the maximum radius of detected circles. Any circles detected whose radius is more than the maximum radius will be eliminated.
4. Circle Solidity - Select how solid a detected circle needs to be. The circle solidity is a measure of standard deviation of a circle's pixel values. The lower the solidity percentage the larger the standard deviation. The higher the solidity percentage the lower the standard deviation which translates into a more purely solid circle (i.e. a flat same color circle).
5. Isolation - Select how close to each other can circles can be detected. The larger the isolation the more space between successive circles needs to exist. If you find that too many overlapping circles are detected increase the isolation number as this will reduce overlapping circles.
6. Isolation Scope - Determine what area is considered when determining if two circles overlap. If set to radius then the isolation is determined by adding the isolation pixels to the current circle size. Thus if any circles include each other the less stronger circle will be eliminated. If center is chosen then circles that are within X pixels of each other's center will be tested for elimination.

7. Statistics Use - Select which image should be used to calculate the solidity and other circle statistics. Since the current image needs to be a contour image you will need to select which image can be used to access the circles original pixel data.
8. Circle Color - Select which color you would like to replace a detected circle with. The values [mean], [median] and [mode] relate to a circles statistical values of the original pixels. Circles will then be draw in that appropriate color.
9. Fill Circles - Select if you want to fill the detected circles with the appropriate color selected above. If "Fill Circles" is not selected then circles will not be filled and remain as thin circles.
10. Center Color - Select which color the X in the center of the circle will be draw using.
11. Radius Color - Select which color the circles radius will be draw in.
12. Overlay on - Select which image the circle will be draw on. If none is selected the circles will be draw on a black cleared image.

Example

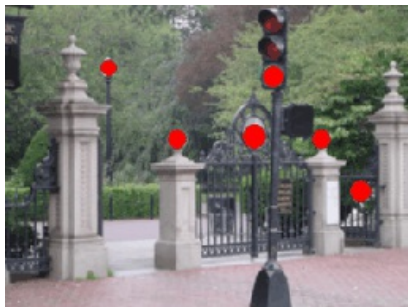
Source



Canny Edges



Detected Circles in Red



Edges

Variables

CIRCLES_COUNT - the number of detected circles

CIRCLES - contains an array of detected circles.

The array contains blocks of 13 values per detected circles. The elements are as follows:

Offset	Contents
0	x coordinate of the center of gravity of the circle
1	y coordinate of the center of gravity of the circle
2	radius of the circle
3	standard deviation of the circle pixel colors
4	mean red color of the circle
5	mean green color of the circle
6	mean blue color of the circle
7	median red color of the circle

```
8 median green color of the circle
9 median blue color of the circle
10 mode red color of the circle
11 mode green color of the circle
12 mode blue color of the circle
```

For example, to add up all the radii of detected circles the VBScript code would be

```
circles = GetArrayVariable("CIRCLES")
for i = 0 to ubound(circles)-1 step 13
    radius = radius + circles(i+2)
next

SetVariable "total_radius", radius
```

To calculate the mean radius and standard deviation of the radius download the  [Circle Statistics Example](#) robofile. Note that this includes an example image.

See Also

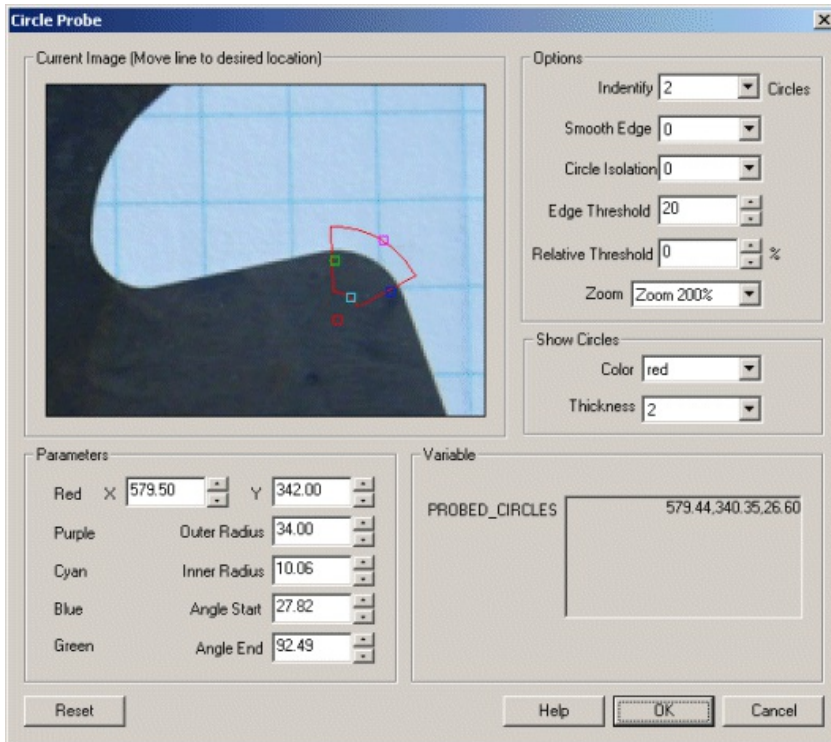
[Ellipses](#)
[Sobel](#)
[Canny](#)
[DOB](#)

Circle Probe



The Circle Probe module provides a way to extract out a circle from a given image assuming that the placement of the probe is in the approximate location of the circle. This module is used in machine vision quality assurance techniques in order to determine if a particular part meets specific quality assurance levels. For example, if your part requires that an inner cut hole be of about 12 pixels in radius the Circle Probe module can be used to extract out that diameter from the current image into a variable array. This variable array then contains the center and radius of the circle which can then be examined by other modules or exported to external applications.

Interface



Instructions

1. Current Image - move the probe graphic to the location which you want to probe for circles. You drag the red square to move the entire probe, the cyan and purple squares to expand or shrink the minimum and maximum radius, or the green and blue squares to specify the angular range of the probe. Likewise, you can move each endpoint individually by changing the coordinates using the text boxes below in the Parameters area.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the probe of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

3. Options Identify - select how many circles you want to detect. Note that if you select many circles to be identified you are not guaranteed to get that many circles depending on the thresholding information below. The "Identify" number specifies a maximal number of circles to detect.

4. Smooth Edge - to reduce noisy edges select the amount of smoothing that should be applied to the edge prior to edge detection.

5. Circle Isolation - to avoid multiple circles from being detected together select how many pixels each circle's center should be isolated from the next detected circle center.

6. Edge Threshold - to eliminate very weak edges from being detected as part of a circle select an appropriate edge threshold (0-255) that will remove edges whose edge intensity is below the threshold. Note that smoothing the edge will also reduce the edge intensity and thus the Edge Threshold will need to be adjusted after the smoothness is specified.

7. Relative Threshold - to only select those edges that are significant you can specify the relative threshold (0-100) that will remove successive edges that are the relative threshold percent less than the previous edge. For example, consider the highest 3 edge values as 130, 120 and 40. If the relative threshold is 50% then the last edge 40 would be eliminated since 40 is less than 50% of 120. As 120 is 92% of 130 it is not eliminated (unless the threshold were set at 95%).

8. Show Circles - to visually see which edges are detected (in the main RoboRealm interface) select the appropriate Color and Thickness of the circle that will indicate where in the image the edges have been detected.

Example

Detected circle using above probe



Variables

PROBED_CIRCLES - an array consisting of the X, Y, and Radius of each detected circle.

PROBED_CIRCLES_COUNT - the number of detected circles.

PROBED_CIRCLES_POINTS - an array consisting of the X, Y coordinates of each detected edge point that was used to calculate the circles.

See Also

[Origin Probe](#)

[Edge Probe](#)

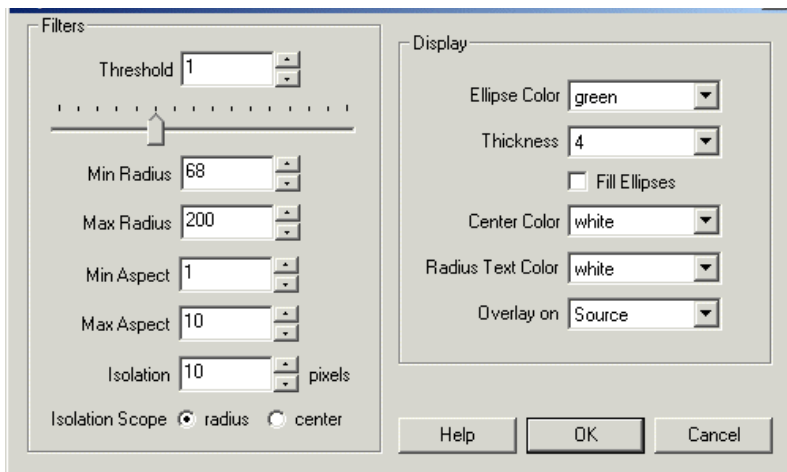
[Line Probe](#)

Ellipses

The ellipses module is a shape detection module that identifies contours within an image that are elliptical. The ellipses module is meant to run after a contour based module (like [Canny](#), [DOB](#), [Sobel](#), etc.) that identify edges within an image. Using these edges the ellipses module will determine which contours are part of an elliptical shape and identify that shape for successive processing.

Note that since the ellipses module is contour based several edge processing routines can be applied prior to the ellipses module to best create distinct contours.

Interface



Instructions

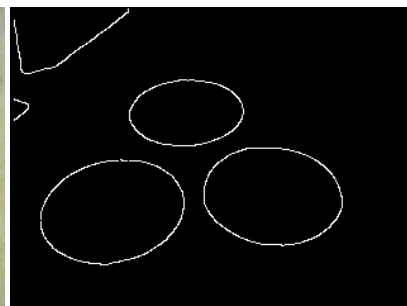
1. Threshold - Select the ellipse threshold. The threshold value sets the sensitivity of the ellipse detector. The lower the threshold the more partial ellipses will be detected. The higher the value the more complete and defined the ellipses will need to be.
2. Min Radius - Select the minimum radius of detected ellipses. Any ellipse detected whose radius is less than the minimum radius will be eliminated. Note that this is compared with the major radius of the ellipse.
3. Max Radius - Select the maximum radius of detected Ellipses. Any ellipse detected whose radius is more than the maximum radius will be eliminated. Note that this is compared with the major radius of the ellipse.
4. Min Aspect - Select the minimum aspect ratio of the ellipse. The ratio is defined by the major axis radius divided by the minor axis radius. Any ellipse detected whose aspect ratio is less than the minimum radius will be eliminated.
5. Max Aspect - Select the maximum aspect ratio of the ellipse. The ratio is defined by the major axis radius divided by the minor axis radius. Any ellipse detected whose aspect ratio is greater than the maximum radius will be eliminated.
6. Isolation - Select how close to each other can ellipses can be detected. The larger the isolation the more space between successive ellipses needs to exist. If you find that too many overlapping ellipses are detected increase the isolation number as this will reduce overlapping ellipses.
7. Isolation Scope - Determine what area is considered when determining if two ellipses overlap. It set to radius then the isolation is determined by adding the isolation pixels to the current ellipse size. Thus if any ellipses include each other the less stronger ellipse will be eliminated. If center is chosen then ellipses that are within X pixels of each other's center will tested for elimination.
8. Ellipse Color - Select which color you would like to replace a detected ellipse with.
9. Thickness - Select how thick the line that draws the ellipse should be.
10. Fill - Select if you want to fill the detected Ellipses with the appropriate color selected above. If "Fill" is not selected then ellipses will not be filled and remain as a thin ellipses.
11. Center Color - Select which color the X in the center of the ellipse will be draw using.
12. Radius Color - Select which color the ellipses radius will be draw in.
13. Overlay on - Select which image the ellipse will be draw on. If none is selected the ellipses will be draw on a black cleared image.

Example

Source



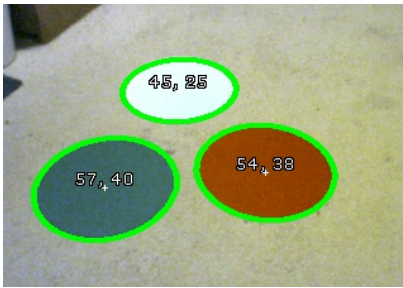
Canny Edges



Detected Ellipses in Green



Edges



Variables

ELLIPSES_COUNT - the number of detected ellipses

ELLIPSES - contains an array of detected ellipses.

The array contains blocks of 6 values per detected ellipse. The elements are as follows:

Offset	Contents
0	x coordinate of the center of the ellipse
1	y coordinate of the center of the ellipse
2	major radius of the ellipse
3	minor radius of the ellipse
4	orientation in radians of the ellipse
5	x axis rotation of the ellipse

For example, to add up all the radii of detected Ellipses would be

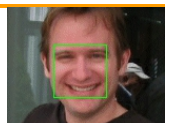
```
ellipses = GetArrayVariable("ELLIPSES")
for i = 0 to ubound(ellipses)-1 step 6
    radius = radius + ellipses(i+2)
next
SetVariable "total_radius", radius
```

See Also

[Circles](#)
[Sobel](#)
[Camy](#)
[DOB](#)

Face Detection

The face detection module is used to detect the presence of faces within an image.



Interface



Instructions

1. Display - You should now notice faces being recognized in the main RoboRealm window. Matches are displayed as green boxes.

You can add or remove information from being displayed by changing the checkboxes in the Display area. Note if too much information is displayed you might want to change the Font size to a smaller size.

- X Translation - the X location with respect to the lower right corner of the object.
- Y Translation - the Y location with respect to the lower right corner of the object.
- Size - the size of the object relative to that in the database.
- Orientation - the Z rotation or angle of the object (spin)

2. Box Color - You can change the match box color by changing the Box Color.

3. Tracking - Face detection is limited to full frontal non-rotated faces. To ensure more stable detection you can chose to have tracking enabled which will quickly look for a face in the previous image in its immediate area and adjust for any movement. Tracking is much faster than recognition and can also track in more dimensions (rotation).

4. Tracking Confidence - Once a face is detected the actual tracking can use a lower confidence since the movement of an object can cause temporary distortions or highlights/shadows within the image. You can use the Tracking Confidence threshold to be lower than the recognition threshold to ensure that once objects are recognized they will still be tracked under harsher image conditions.

5. Check Sizes - The image is processed for various face sizes. Each additional size will require additional scans. You can restrict the detection of face sizes to a smaller range that will help improve the scan speed or can also be used to eliminate smaller or large false detections. Note that the size reflects the size in pixels of a face which is slightly smaller than the size of the actual head.

6. Stabilization - When moving the camera motion blur can cause incorrect low confidence matches to appear. By increasing the "Present After" you are requiring a face to appear for a set number of frames before being reported as a match. This essentially removes the "flickering" of low matched faces that can cause noise in the results. Note that if you are working with static images you will need to set this to zero! In the same way the "Absent After" number will only remove an object from being tracked after the specified number of frames. This will help prevent the momentary disappearance of a face due to pixel noise.

7. Variables - To use the results of the object module in your own application or in other RoboRealm modules you can select that the object array be created. This array has all the collected information about a detect object. See below for the array format.

8. Display Results - By default No Change is selected which does not modify the current image in any way other than annotating the objects with the above colors. If you want to use the results in other modules you can select to see

- Colored - displays only the faces with their original pixel colors
- Inverted - displays everything else but the face (which is blacked out).
- White Mask - white where the faces are, black everywhere else
- Black Mask - black where the faces are, white everywhere else

Variables

FACES - contains an array of detected faces.

FACES_COUNT - the number of faces detected and the total size of the FACES array (FACES_COUNT*13)

The array contains blocks of 13 values per detected circles. The elements are as follows:

The FACES array is composed of 13 numbers as follows:

Offset	Contents
0	Tracking Confidence 0-100
1	Point 1 X coordinate
2	Point 1 Y coordinate
3	Point 2 X coordinate
4	Point 2 Y coordinate
5	Point 3 X coordinate
6	Point 3 Y coordinate
7	Point 4 X coordinate
8	Point 4 Y coordinate
9	Translation in X
10	Translation in Y
11	Tracking Scale 0-100
12	Tracking Orientation 0-2PI (radians)

For example, to print out all found faces use the VBScript module and the following script.

```
faces = GetArrayVariable("FACES")

if isArray(faces) then
    if ubound(faces) > 0 then
        for i=0 to ubound(faces)-1 step 13
            write "Face At: " & objects(i+9) & ", " & objects(i+10) & vbCRLF
        next
    end if
end if
```

See Also

[Object Recognition](#)



The Fiducial module provides you with an easy way to identify a particular marker within the current image view. Fiducials are commonly known as "markers that are easy to identify" and are typically added to an environment where localization and navigation are needed for robots using machine vision.

Fiducial detection and recognition differs from that of generic object recognition as some assumptions are made on the type of object being detected. For example, a black and white fiducial can be detected without needing color, and can be easily separated from the background due to its high contrast nature. Fiducials are also typically 2D planar objects that have very distinct corners or shapes. Identifying these traits in the environment helps to reduce the "clutter" of other objects and ensures better identification reliability.

Thanks to the planar nature of fiducials they can be placed on the floor, ceiling or walls and still be detected correctly without any recalibration of the camera. By knowing the basic shape of a fiducial machine vision techniques can account for the perspective and affine distortion (i.e. a fiducial rotated in X and Y plane) to a considerable degree while still maintaining recognition. It is for this reason, along with their barcode like nature, that fiducials are frequently used for navigation and localization.

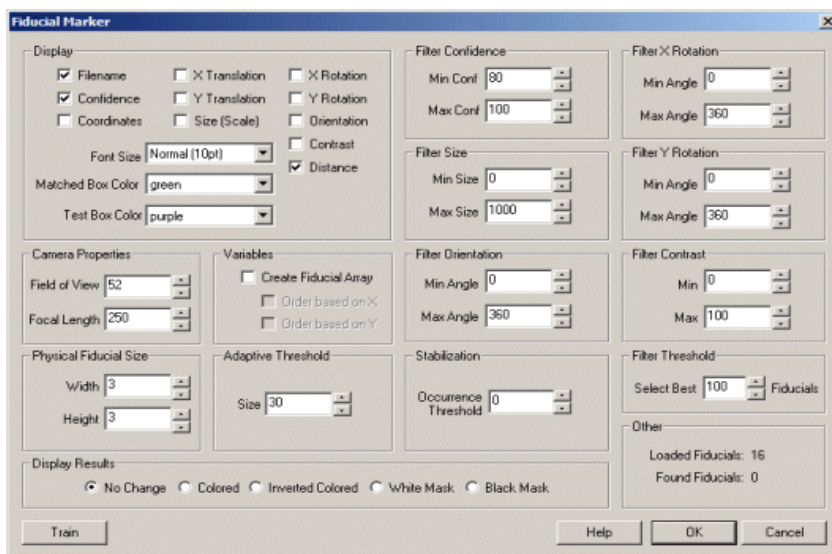
There are 6 parameters that localize a fiducial in space. Luckily, due to the shape and planar nature of the fiducial these 6 parameters can be determined very quickly and allow the processing to run at very near camera speed. This has an advantage on moving robots that need to process their environment before it changes.

Once you know the identity of a fiducial you can know approximately where you are. Having the fiducial in sight helps the robot to know where it is with respect to the fiducial and can use the fiducial as a target for navigation based on scale, rotation and translation.

The Fiducial module provides a way to easily recognize fiducials in your environment. We have included 15 test fiducials in the Fiducials folder along with the RoboRealm download that you can print out to experiment with. If you don't have a printer just bring them up on your screen and point your camera towards it!

Note that these are graphical files that can be modified (or more created) as needed. Also note the one poster like fiducial. The only requirement for the fiducial is that it be surrounded by a white surface with an inner black square and contain inner elements that can distinguish it from other Fiducials. If you have posters that satisfy this requirement then you already have fiducials on your walls!

Interface



Instructions

1. Train - Press the train button and enter the full path to the Fiducials folder that you downloaded from RoboRealm. This will typically be c:\Program Files\RoboRealm\Fiducials\. Press the START button to start training. After it is complete press the ok button. If you would like training to happen automatically whenever a file is added, updated or deleted from the specified folder select the 'Monitor Folder' checkbox.

2. Display - You should know notice the fiducials being recognized in the main RoboRealm window. Matches are displayed as green whilst candidate regions that do not match are in purple. Note that the 'name' of a fiducial is the name of the image filename without the extension. So you can change the names by changing the filenames and re-training.

3. Attributes - By default the Filename and Confidence are displayed. You can add or remove information from being displayed by changing the checkboxes in the Display area. Note if too much information is displayed you might want to change the Font size to a smaller size.

- Filename - the name of the fiducial.
- Confidence - how strongly the module believes this fiducial matches one in the database folder.
- Coordinates - the location of the matched fiducial.
- X Translation - the X location with respect to the center of the screen of the fiducial.
- Y Translation - the Y location with respect to the center of the screen of the fiducial.
- Size - the size of the fiducial relative to that in the database.

USE THE SIZE OF THE FIDUCIAL SQUARE TO SET IN THE DATABASE.

- X Rotation - the rotation in the X axis of the fiducial (tilt)
- Y Rotation - the rotation in the Y axis of the fiducial (pan)
- Orientation - the Z rotation or angle of the fiducial (spin)
- Contrast - range between lowest and highest intensity value within fiducial. The higher the better.

4. Box Color - You can change the match and test box color by changing the Matched Box Color and Test Box Color respectively.

5. Filter Confidence - Many candidate squares do pass the basic fiducial requirements but do **not** match with a known fiducial very well. You can remove those unwanted matches by increasing the Min Confidence. This removes bad matches from being made.

6. Filter Size - Each recognized fiducial has a scale associated with it that can be used to remove larger or smaller matches.

7. Filter Orientation/X/Y - by default fiducials are recognized in any orientation. You can use the Filter Orientation to remove those fiducials outside of your desired orientation. For example, 15, 345 would remove any fiducial ± 15 degrees from that what is in the database. You can use the X and Y rotation filters to remove fiducials that are rotated into the image plane (i.e. slightly twisted or flipped).

8. Filter Threshold - you can decide on how many fiducials you want to work with. Specifying a number in the Filter Threshold will display the best X fiducials found in the image.

9. Stabilization - When moving the camera motion blurring can cause incorrect low confidence matches to appear. By increasing the Occurrence Threshold you are requiring a fiducial to appear and continue to appear for a set number of frames before being reported as a match. This essentially removes the "flickering" of low matched fiducials that can cause noise in the results. Note that if you are working with static images you will need to set this to zero!

10. Filter Contrast - As the module attempts to ignore lighting as much as possible it is capable of pulling in fiducial matches that have very little contrast. Using the contrast filter can help to remove those false recognitions by ensuring that enough contrast exists in the fiducial before attempting recognition.

11. Field of View - In order to provide accurate distance information the module needs to know how large a field of view the camera can see. If you do not know this value, hold the fiducial at a known distance from the camera and adjust the field of view until the distance is correct. Note that this is the Vertical Field of View.

12. Focal Length - Due to the focal length of the camera perspective will be apparent in the fiducial square. You need to specify what this focal length is for your camera in order for the fiducial's angles to be extracted correctly. The manufacturer of your camera will normally know this value. If you do not have access to this value you can empirically determine this by placing a fiducial at a known out of plane angle to the camera (for example 45 deg X rotation) and adjusting the focal length FL value up or down until the X rotation correctly reads 45.

13. Focal Length Error Color - When a fiducial is detected and matches but cannot create a correct parameterization of the fiducial (i.e. what rotations, translations, perspective is the fiducial experiencing) the module will NOT record the fiducial and display the specified error color around the Fiducial. When this happens, you will need to review your focal length setting as this is often the cause of the parameterization failure. This is typically experienced with Fiducials that are rotated out of plane (X or Y axis) where perspective becomes apparent and the correct FL setting becomes required.

14. Physical Fiducial Width/Height - In order to determine the distance to the fiducial the actual size (in whatever units you want the distance to be) needs to be entered. For example, if you print out a fiducial such that it takes up most of a page, then the actual measurement would be around 7 inches. You can simply use a ruler to measure the actual size (width and height are normally the same) and enter those measurements. If you enter inches then the distance value will be in inches, if you enter meters, then the distance value will be in meters.

15. Variables - To use the results of the fiducial module in your own application or in other RoboRealm modules you can select that the fiducial array be created. This array has all the collected information about a detect fiducial. See below for the array format.

16. Display Results - By default No Change is selected which does not modify the current image in any way other than annotating the fiducials with the above colors. If you want to use the results in other modules you can select to see

- Colored - displays only the fiducials with their original pixel colors
- Inverted - displays everything else but the fiducial (which is blacked out).
- White Mask - white where the fiducials are, black everywhere else
- Black Mask - black where the fiducials are, white everywhere else

Notes

Why do you need to enter in both Camera Field of View and Focal Length? Are they not related to each other? Yes they are, the issue is that in order to relate them we'd need to know the actual sensor size in your camera (e.g. a 35mm camera has a film size of 35mm). As this is difficult to estimate having both parameters estimated using the actual camera and know angle/distance of the fiducial is an easier approach.

Example

Source

Fiducial



Variables

FIDUCIALS - contains an array of detected fiducials.

The array contains blocks of 17 values per detected fiducial. The elements are as follows:

The FIDUCIALS array is composed of 17 numbers as follows:

Offset	Contents
0	Match Confidence 0-100
1	Point 1 X coordinate
2	Point 1 Y coordinate
3	Point 2 X coordinate
4	Point 2 Y coordinate
5	Point 3 X coordinate
6	Point 3 Y coordinate
7	Point 4 X coordinate
8	Point 4 Y coordinate
9	Translation in X
10	Translation in Y
11	Scale 0-100
12	Rotation in X (radians)
13	Rotation in Y (radians)
14	Orientation (Rotation in Z, radians)
15	Path start index
16	Length of path

For example, to print out all found fiducials use the VBScript module and the following script.

```

fiducials = GetArrayVariable("FIDUCIALS")
names = GetStrVariable("FIDUCIALS_PATH")

if isArray(fiducials) then
  if ubound(fiducials) > 0 then
    for i=0 to ubound(fiducials)-1 step 17
      nstart = fiducials(i+15)
      nend = fiducials(i+16)
      write "Conf: " & fiducials(i) & "% Path: " & _
        mid(names, nstart, nend) & vbCRLF
    next
  end if
end if

```

For ease of access the largest fiducial's information is also available in the following variables

FIDUCIAL_CONFIDENCE - the confidence of the best matched fiducial

FIDUCIAL_CONTRAST - the amount of contrast in the fiducial

FIDUCIAL_DISTANCE - the distance to the fiducial given the specified camera properties

FIDUCIAL_FILENAME - the filename of the fiducial

FIDUCIAL_FOLDER_X - the folder sequence that the best fiducial is in (replace X with 1, 2, etc.)

FIDUCIAL_NAME - the name of the matched fiducial

FIDUCIAL_ORIENTATION - the orientation (spin) of the best fiducial

FIDUCIAL_PATH - a string array of all the names of matched fiducials as specified by entry #15 in the FIDUCIALS array

FIDUCIAL_SIZE - the size of the best matched fiducial

FIDUCIAL_WIDTH - the width of the best matched fiducial

FIDUCIAL_HEIGHT - the height of the best matched fiducial

FIDUCIAL_X1_COORD - x coordinate of first box corner of the best matched fiducial

FIDUCIAL_X2_COORD - x coordinate of second box corner of the best matched fiducial

FIDUCIAL_X3_COORD - x coordinate of third box corner of the best matched fiducial

FIDUCIAL_X4_COORD - x coordinate of fourth box corner of the best matched fiducial

FIDUCIAL_X_COORD - the x location relative to center screen of the best matched fiducial

FIDUCIAL_X_ROTATION - the x rotation (tilt) of the best fiducial

FIDUCIAL_X_TRANS - x axis translation of the fiducial

FIDUCIAL_Y1_COORD - y coordinate of first box corner of the best matched fiducial

FIDUCIAL_Y2_COORD - y coordinate of second box corner of the best matched fiducial

FIDUCIAL_Y3_COORD - y coordinate of third box corner of the best matched fiducial

FIDUCIAL_Y4_COORD - y coordinate of fourth box corner of the best matched fiducial

FIDUCIAL_Y_COORD - the y location relative to center screen of the best matched fiducial

FIDUCIAL_Y_ROTATION - the y rotation (pan) of the best fiducial

FIDUCIAL_Y_TRANS - y axis translation of the fiducial

FIDUCIAL_CONFIDENCE_ARRAY - an array of confidence values of all matched fiducials

FIDUCIAL_CONTRAST_ARRAY - an array of contrast values of all matched fiducials

FIDUCIAL_DISTANCE_ARRAY - an array of distance to the fiducial given the specified camera
properties of all matched fiducials

FIDUCIAL_NAME_ARRAY - an array of all the names of matched fiducials

FIDUCIAL_ORIENTATION_ARRAY - an array containing the orientation (spin) of the all matched fiducials

FIDUCIAL_SIZE_ARRAY - an array containing the size of the all matched fiducials

FIDUCIAL_WIDTH_ARRAY - an array containing the width of the all matched fiducial

FIDUCIAL_HEIGHT_ARRAY - an array containing the height of the all matched fiducial

FIDUCIAL_X1_COORD_ARRAY - an array containing x coordinate of first box corner of all matched fiducials

FIDUCIAL_X2_COORD_ARRAY - an array containing x coordinate of second box corner of all matched fiducials

FIDUCIAL_X3_COORD_ARRAY - an array containing x coordinate of third box corner of all matched fiducials

FIDUCIAL_X4_COORD_ARRAY - an array containing x coordinate of fourth box corner of all matched fiducials

FIDUCIAL_X_COORD_ARRAY - an array containing the x location relative to center screen of all matched
fiducials

FIDUCIAL_X_ROTATION_ARRAY - an array of the x rotation (tilt) of all matched fiducials

FIDUCIAL_X_TRANS_ARRAY - an array of the x axis translation of all matched fiducials

FIDUCIAL_Y1_COORD_ARRAY - an array containing y coordinate of first box corner of all matched fiducials

FIDUCIAL_Y2_COORD_ARRAY - an array containing y coordinate of second box corner of all matched fiducials

FIDUCIAL_Y3_COORD_ARRAY - an array containing y coordinate of third box corner of all matched fiducials

FIDUCIAL_Y4_COORD_ARRAY - an array containing y coordinate of fourth box corner of all matched fiducials

FIDUCIAL_Y_COORD_ARRAY - an array containing the y location relative to center screen of all matched
fiducials

FIDUCIAL_Y_ROTATION_ARRAY - an array of the y rotation (pan) of all matched fiducials

FIDUCIAL_Y_TRANS_ARRAY - an array of the y axis translation of all matched fiducials

See Also

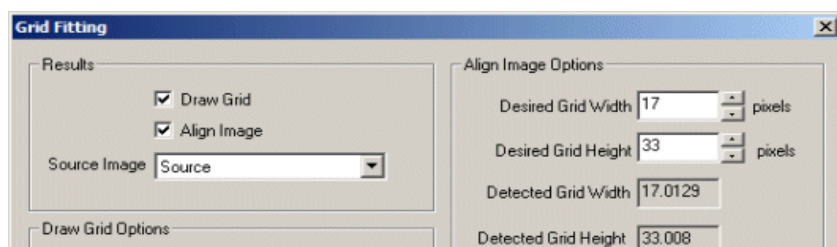
[Shape Matching](#)

[Blob Filter](#)

Grid Fitting

The Grid Fitting module provides a way to align a grid on top of many individual points to aid in alignment of the image or to create a concise separation of objects in a grid pattern. This can be used as part of a segmentation process to break multiple periodic objects into individual cells that can then be further processed.

Interface



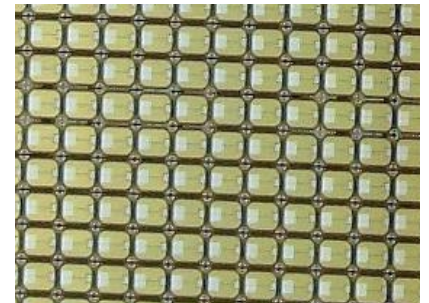


Instructions

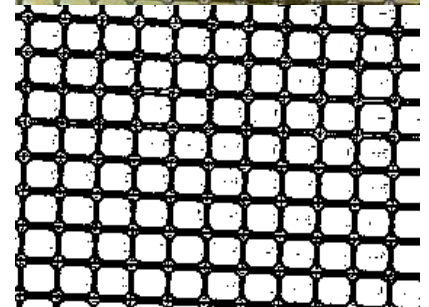
1. Precursor - Ensure that the image before the Grid Fitting module contains white objects in a grid or periodic pattern. The module will calculate the center of gravity of each object to use as an anchor for the grid alignment process. Note, this is assume to be a black and white image.
2. Draw Grid - Specifies that the found grid should be draw ontop of the Source Image. This is useful when attempting to break apart objects for additional processing where their boundaries are not well defined.
3. Align Image - Causes the Source image to be aligned in accordance to the detected Grid. This is helpful when the grid needs to be repositioned in a known orientation for further processing.
4. Source Image - The image that constitutes the pixels behind the current black and white blob image. The module expects a black and white image with white blobs being points that constitute a grid. Once the grid has been found applying the transform to the source image is typically what the module is used for.
5. Desired Grid Width/Height - When aligning the grid you can specify the desired size of the grid elements. This is useful when you want successive images of the grid to be scaled to the same dimensions.
6. Out of bounds Color - When the image is aligned parts of the resulting image may not map from the source image. The out-of-bounds color specifies what color to make these parts of the resulting image.
7. Draw Grid Color - The color to draw the detected grid.
8. Thickness - The line thickness to use when drawing the grid. Note, a single pixel line may not be sufficient to disconnect blobs.
9. Display As Annotation - Draw the grid, but only on the second pipeline pass so as not to affect subsequent processing due to the drawn grid.

Example

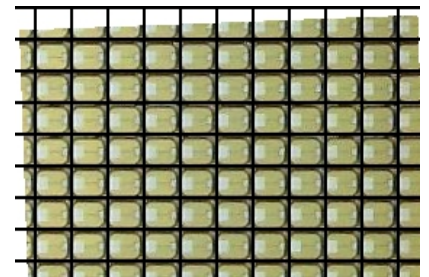
The source image to be aligned based on a grid like pattern.



Use adaptive threshold module to binarize the image and identify the cells that are part of the grid pattern.



Use the Grid Fitting module to detect the grid and align the image correctly with 20x25 pixel cells.



Variables

GRID_FITTING_ERROR - Number of pixels off the fitting of the Grid is.

GRID_FITTING_RZ - Rotation of Grid in the Z plane

GRID_FITTING_RZ - Rotation of Grid in the Z plane
GRID_FITTING_RX - Rotation of Grid in the X plane
GRID_FITTING_RY - Rotation of Grid in the Y plane
GRID_FITTING_TX - X Translation
GRID_FITTING_TY - Y Translation
GRID_FITTING_HORIZ_FREQ - Detected X unit size
GRID_FITTING_VERT_FREQ - Detected Y unit size



See Also

[Orient Image](#)

[Rotate](#)

[Transform Image](#)

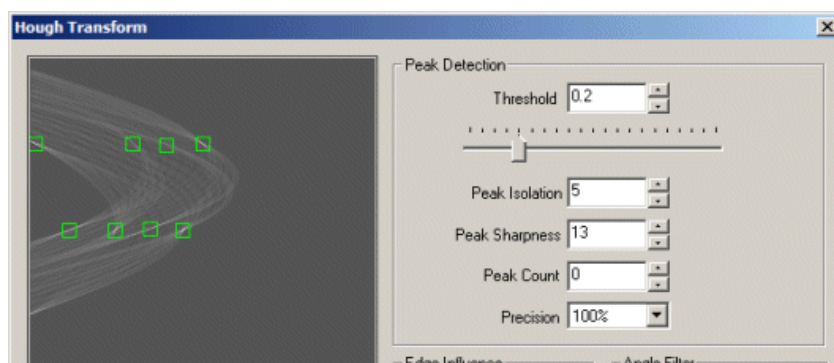
Hough

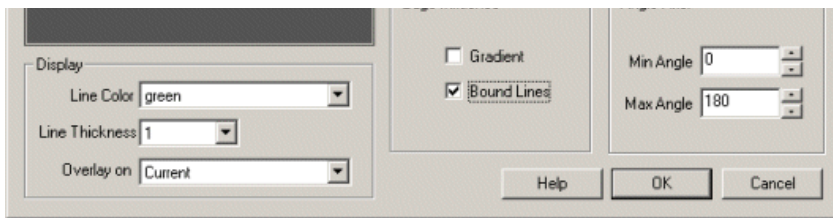
The Hough transform is a technique for creating lines based on points. The results of a typical edge detection routine are many unconnected points. To us it is obvious that these points represent shapes but because the points are not connected it is difficult for a machine to understand the underlying shape. The Hough transform takes as input many points and will generate guesses for what lines those points represent.

The Hough module interface shows the parameter space image that represents the Hough transform in a visual manner. The image axis are the angular amount (theta) and the distance value (r) used to represent a line using polar notation " $r = x \cos(\theta) + y \sin(\theta)$ " as apposed to coordinate notation " $y = mx + b$ ". The Hough transform converts potential lines into peaks within this image. The task of detecting lines now instead becomes a task of detecting peaks within this image. The interface below shows green squares around those detected peaks which indicate the parameters (theta,r) of the lines seen in the below example. By thresholding and isolating these peaks it is possible to get a reasonably good approximation of the actual figure (source image -> Hough transform).

For a detailed description of how the Hough transform works see the links below.

Interface





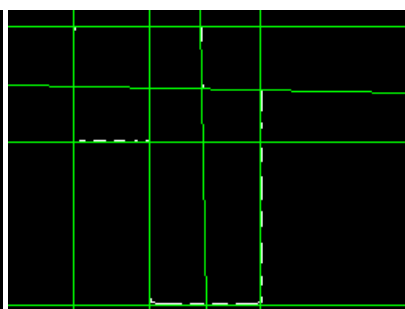
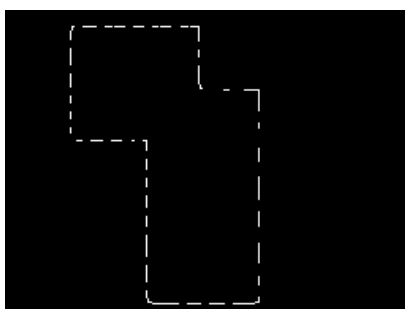
Instructions

1. Setup - To use the Hough transform it is typical to first use an edge detection routine like the [Sobel Edge](#). This helps identify points to use in the transform and can speed up processing by eliminating points (i.e. setting them to black) that do not need to be processed.
2. Threshold - Increase or decrease the "Threshold" number field or scroll the horizontal scroller until a number of lines appear in the image.
3. Isolation - Often many lines that are detected are very close in slope and proximity to each other and can create a meaningless image. Use the peak isolation number to remove lines whose slopes are close to each other. See below for a visual example.
4. Sharpness - Not all detected Hough Peaks are very sharp which produce lines which are created from noise within the image. Increase the sharpness value until lines that do not have sharp support are removed from the display. These errors happen often when the prior edge detection module creates a high response in a flat area due to texture noise. As that is not a true edge the resulting peak's shape will not be sharp.
5. Peak Count - If you know how many lines you wish to detect enter that number in the Peak Count textbox. This will ensure that the module selects only the X highest peaks/lines as a result of the hough transform. This can be more adaptive than setting the threshold as that can change based on image size, edge detection method used, etc.
6. Precision - If the module causes your machine to slow down or hang use the "Precision" to reduce the computational load on your machine. The Hough transform is VERY cpu intensive and not all the points need to be analyzed in order for a useful result to appear. Changing the "Precision" value will reduce the number of points used in the transform by skipping over X number of points.
7. Edge Gradient - If you have a gradient image (as apposed to a binary image) then checking the gradient will cause the transform to take into account the intensity of the edge. Thus strong edges will result in a higher peak value than weak edges which can make threshold detection easier. Note that this disables the bounding of lines.
8. Bound Lines - Select the bound lines checkbox if you want to bound each line to those points that define the extreme points of the detected line. (Black/White Images only)
9. Min/Max Angle Filter - Specify the line angles that you want to ignore from the results. Note that the range is 0 to 180 degrees with 0 and 180 being horizontal lines and 90 being vertical. Thus if you wanted to eliminate horizontal lines you could use 20 and 160 as the range. If you wanted to eliminate vertical lines then using 100 and 70 as the range would remove angles 70 to 100. Note that if min is larger than max, the longer arc (i.e. from 100 to 180 and then 0 to 70) would be in range) is used due to circular math.
10. Line Color - The color of the detected lines drawn in the main image (and also the peak indicators).
11. Line Thickness - The thickness to use when drawing the detected lines in the main image.
12. Overlay On - Select if you want to draw the detected lines ontop of your source image to help you understand why lines are appearing.

Example

Source

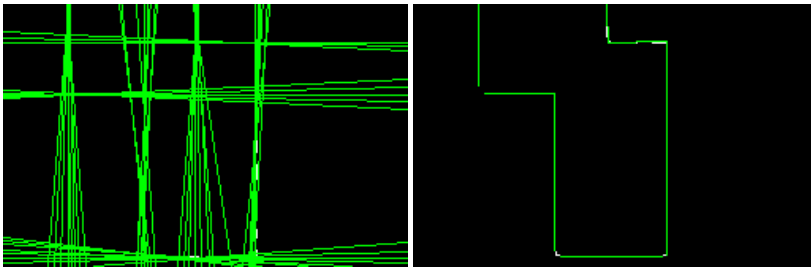
Hough Lines



Hough Lines with peak isolation of 1

Bounded Hough Lines





Variables

HOUGH_LINES - contains the lines found from the Hough transform. Using code similar to the following within a VBScript module will allow you to further process the line segments.

HOUGH_LINE_INTENSITIES - contains the strength or height values for the lines found from Hough transform. Using these codes you can determine a level of confidence associated with the line. Note that this array is 4x smaller than the **HOUGH_LINES** array.

```
line = GetArrayVariable("HOUGH_LINES")
inten = GetArrayVariable("HOUGH_LINE_INTENSITIES")

' write back to messages the first line ..
' note that line segments are defined by two endpoints
Write("First Coord: " & line(0) & "," & line(1) & vbCRLF)
Write("Second Coord: " & line(2) & "," & line(3))
Write("Strength: " & inten(0))
```

Image Matching

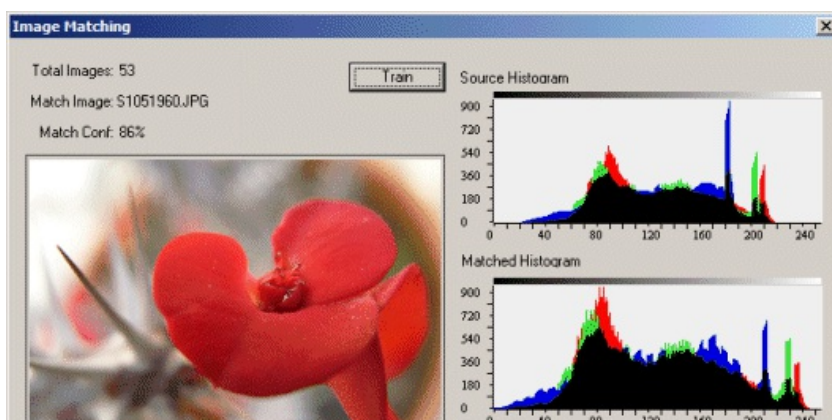


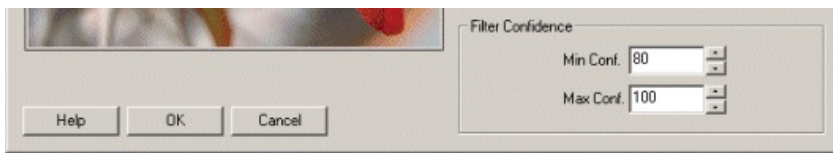
The Image Matching module is a color based matching module that attempts to match the current image with the closest image in its database (i.e. folder of images) based on color. This module is built to handle hundreds of images quite quickly but is based solely on color/intensity.

The image's histogram is determined and quickly compared to all other images in its database. Some lighting variance is adjusted for prior to this comparison to avoid lighting changes from causing false matches. Because color and intensity is reduced to a one dimensional histogram there is no comparison of features and/or shapes being made. Thus, it is very possible to be presented with an image that appears quite different to one in the database but whose global color and intensity is quite close to the presented match.

If shape and/or other features are needed when matching please see the [Object Recognition](#) module which has more advanced comparison techniques. Note that this module matches the entire image as apposed to individual objects within the image.

Interface





Instructions

1. Train - First you will need to train the module on images that need to be matched to. Press Train to begin that process and select the folder where all your images are stored. Once specified press Start to start the training process.
2. After training is complete you will be presented with that image in the database that most closely matches that in the current main RoboRealm interface based on the number of color and intensity pixels that are very close to the matched image.
3. Filter Confidence - Select the confidence amount that is allowed to match. The higher the number the more the image will need to match the one in the database.

Variables

The module creates a number of variables that can be then utilized in other ways to create an action based on the matched results.

`MATCH_CONFIDENCE` - how well the presented matched image matches
the current image

`MATCH_NAME` - The full path of the matched image relative to the
training folder

`MATCH_FOLDER_X` - The specific folders/categories that the matched
image is in or belongs to.

`MATCH_FILENAME` - The filename of the matched image (without path
information)

`MATCH_LABEL` - The name of the matched image without path or file
type extension information.

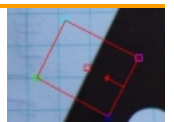
Images

`MATCH_IMAGE` - the image matched in the database

See Also

[Object Recognition](#)
[Shape Match](#)
[Fiducial](#)

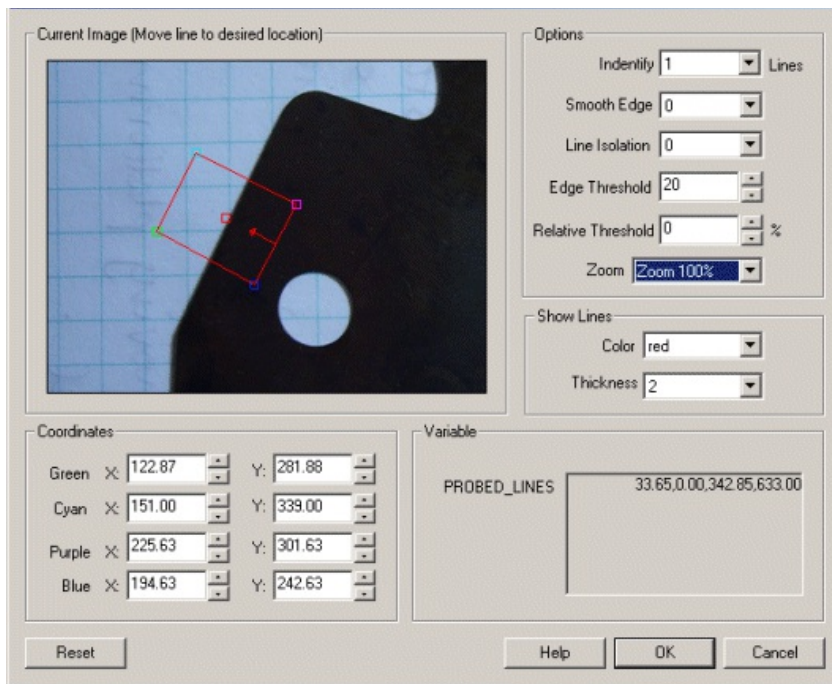
Line Probe



The Line Probe module provides a way to extract out a line from a given image assuming that the placement of the probe is in the approximate location of the line's edge. This module is used in machine vision quality assurance techniques in order to determine if a particular part meets specific quality assurance levels. For example, if your part requires that a side of the part needs to be a particular slope or intersect another line at a particular point then the Line Probe module can be used to extract out that line. The variable array then contains the start and end coordinates of the line which can then be examined by other modules or exported to external applications.

Interface





Instructions

1. Current Image - move the probe graphic to the location which you want to probe for lines. You drag the red square to move the entire probe or move each of the corners to change the shape and size of the probe. Note that the arrow dictates the search direction and should always be perpendicular to the edge being detected. You can also move each endpoint individually by changing the coordinates using the text boxes below in the Coordinates area.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the probe of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

3. Options Identify - select how many lines you want to detect. Note that if you select many lines to be identified you are not guaranteed to get that many lines depending on the thresholding information below. The "Identify" number specifies a maximal number of lines to detect.

4. Smooth Edge - to reduce noisy edges select the amount of smoothing that should be applied to the edge prior to edge detection.

5. Line Isolation - to avoid multiple lines from being detected together select how many pixels each line's center should be isolated from the next detected line center.

6. Edge Threshold - to eliminate very weak edges from being detected as part of a line select an appropriate edge threshold (0-255) that will remove edges whose edge intensity is below the threshold. Note that smoothing the edge will also reduce the edge intensity and thus the Edge Threshold will need to be adjusted after the smoothness is specified.

7. Relative Threshold - to only select those edges that are significant you can specify the relative threshold (0-100) that will remove successive edges that are the relative threshold percent less than the previous edge. For example, consider the highest 3 edge values as 130, 120 and 40. If the

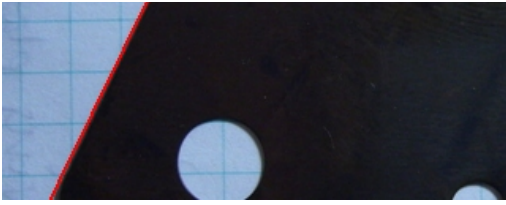
relative threshold is 50% then the last edge 40 would be eliminated since 40 is less than 50% of 120. As 120 is 92% of 130 it is not eliminated (unless the threshold where set at 95%).

8. Show lines - to visually see which edges are detected (in the main RoboRealm interface) select the appropriate Color and Thickness of the line that will indicate where in the image the edges have been detected.

Example

Detected line using above probe





Variables

PROBED_LINES_COUNT - the number of lines detected.

PROBED_LINES - an array consisting of the X_START, Y_START, X_END, Y_END of each detected line.

PROBED_LINES_POINTS - an array consisting of the X, Y coordinates of each detected edge point that was used to calculate the lines.

See Also

- [Origin Probe](#)
- [Edge Probe](#)
- [Circle Probe](#)

OCR



The OCR (Optical Character Recognition) module provides a way to convert text represented within an image into ascii text. Once converted it becomes easier to interpret an image by analyzing and processing digital text instead of pixels.

The OCR module is meant to be used with short phrases that have been segmented from the background and represented as a black and white image with the 'to-be-extracted' text in white.

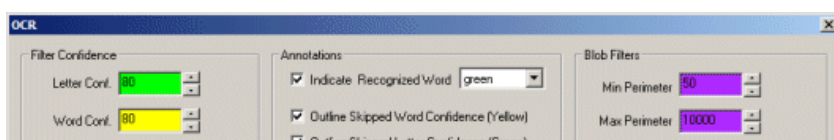
The OCR module uses font images stored in the OCR folder alongside the RoboRealm install folder. These font files are used to compare against blobs located within the current image to determine the appropriate match. These files are 24x48 graphic files which can be managed from the file explorer interface in Windows and edited in any graphic editor. For performance reasons all these files are combined into an OCR/ocr_database.dat file. Should you wish to modify this database, you can delete this file, [download the original GIF files](#) and extract them in the OCR folder. Once you make any modifications and restart RoboRealm again, the ocr_database.dat will be regenerated from those files. Please note the fontnames.txt can also be added to in order to classify new font additions. When restarting, there will be pause while the module recreates the database file. The application will appear frozen during that time.

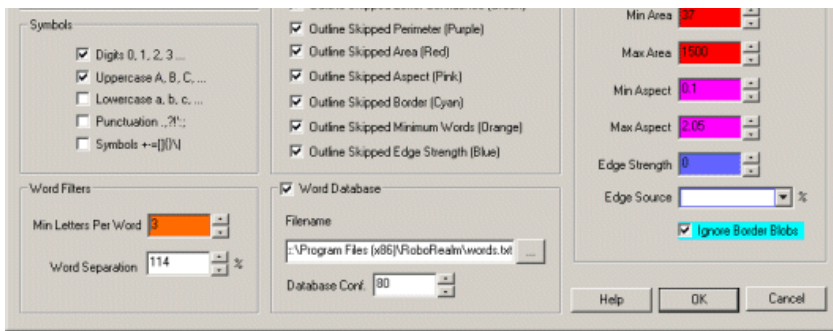
To improve performance, the module attempts to reduce the false tests on blobs that are not actual letters, digits, etc. Included in the module are several filters that help to remove non-symbols. These filters are similar to those contained in the [Blob Filter](#) module. By eliminating all non-symbols the allowed error for comparisons between know fonts and those contained in the image can be made very low to allow for different font comparisons not to require exact matches.

To provide immediate feedback for which filter is active on a particular blob, each filter will outline a blob that it eliminates. Using this color key, you can quickly adjust the filters to eliminate or preserve a blob for testing against the template database.

The final extracted text is placed into a OCR_RESULT variable. Words that are on the same line are separated by a space with words of different lines being separated with a newline character.

Interface





Instructions

1. **Letter Confidence** - The per symbol confidence threshold. Symbols that are matched to known letters, digits, etc. that fall below this confidence value are not recorded as part of the result. Very low values will include undesirable parts whilst making this too high will eliminate letters within words.
2. **Word Confidence** - The per word confidence threshold. Groups of symbols are combined together into words with their individual confidence values are averaged to create a word confidence.
3. **Symbols** - Select which groups of characters you want to recognize. Fewer groups will yield faster and more accurate results as less symbols will be compared against the font templates.
4. **Annotation** - To better review what blobs are being removed by which filters you can use the Outline checkboxes to color the blobs that get removed. The colors relate to which filter was used to remove that blob from the final result.
5. **Word Database** - To better match against a known list of words you can specify a text file that contains words to be recognized with one word per line. This list will then focus the recognition on a known list instead of determining a best match. Currently the default uses a large English dictionary to help English common words get recognized correctly.
6. **Database Confidence** - When comparing best matches to the database, the module will replace best matches with secondary, or tertiary, etc. matches. This lowers the overall confidence of the final word. The database matching will stop if this value falls below the specified value to prevent completely wrong matches from being assumed correct when they exist in the database.
7. **Min/Max Perimeter** - Blobs whose perimeter (the blob's outline) that fall outside of this specified minimum and maximum range will be excluded from recognition.
8. **Min/Max Area** - Blobs whose area (total number of pixels a blob has) that fall outside of this specified minimum and maximum range will be excluded from recognition.
9. **Edge Strength** - Blobs whose edge's transition is below the specified threshold will be excluded from recognition. The calculation is done by investigating the blobs border transition. Those blobs that are well defined (have a sharp edge) will result in a stronger value.
10. **Edge Source** - As the image fed into the OCR module is a binary image using it will not yield the correct results for the edge strength calculation (all blobs have sharp edges). Instead you need to specify the original image that contains the gray values that created the binary image in order to calculate the blob's relative edge strength. This will typically be the Source image.
11. **Ignore Border Blobs** - Select to remove blobs located on the border of the image that may be mistakenly recognized as incorrect symbols.
12. **Minimum Letters Per Word** - This specifies how many symbols/letters need to be recognized in order for a word to be recognized. This ensures that multiple symbols are needed in order to create a word. This provides a form of context that helps to isolate and remove incorrect entries.
13. **Maximum Letters Per Word** - This specifies how many symbols/letters need to be recognized in order for a word to be recognized. This helps to minimize long strings of characters from being identified as a word. 0 indicates that this filter is inactive.
14. **Word Separation** - In order to determine what makes a word instead of an individual symbol a separation amount that below which defines a joined symbol and above which defines a new word needs to be specified. The amount is based on a percentage of the previous symbol. For example, if the previous symbol is 30 pixels high and the Word Separation is 125% then any symbol within 38 pixels will be considered part of the same word.
15. **Heuristics** - When extracting out text from a full scene its likely that irregular combinations of text are extracted. For example, often in parallel bars (such as those seen in a fence) the module may extract out a word such as "lllll". Enabling the heuristics checkbox will check for these irregular matches and remove them from the results. Letters/digits removed using these rules are outlined in dark red to indicate why they have been disqualified from the end results.

Examples

Examples using Adjusting Threshold to segment text from background

Examples use Adaptive Threshold to segment text from background.



Variables

OCR_RESULT - The text extracted from the image.

OCR_BOUNDING_BOX - Array of integers that specify an index into OCR_RESULT followed by 4 coordinates (2 numbers each) that specify the words bounding box.

See Also

[Shape Matching](#)

Object Recognition



The Object Recognition module provides a way to identify specific trained objects within the current image. Once the module is trained with sample template images it will identify those objects within the current image depending on the filtered parameters of confidence, size, rotation, etc.

Several of the techniques will account for different object sizes, location and in plane rotation (roll) of the object as well as variations in lighting and contrast. It will NOT account for significant rotation of the object in the X and Y (pan and tilt) directions. Should you need to identify a 3D object in any orientation you will need to include template examples of each orientation.

Templates are created by including images into a folder that is used as the training samples for this module. Thus you can use any image editing application to edit and manage those templates as needed. Note, it is always best just to include the object to be identified without any background parts of the image. Only one folder at a time can be selected into a single object Recognition module. By changing the folder you change those objects that are to be identified. Note that you can also use more than one Object Recognition module within the pipeline.

It is recommended to use as large a template image as possible that will appear in the scene. One that encompasses as much of the image size as possible is best. This is because the Object Recognition module will only seek out objects from the template size down to a minimum of 1/3th the template size. Thus having the template contain the most amount of detail will provide the best results. If you specify too large an object then smaller versions of the objects may not be recognized as they may fall below the 30% size limit.

Several recognition methods are provided. As many objects/environments differ in task the module provides various methods that can be switched between in order to determine the best technique for your use. Note that while you can switch between the techniques by selecting the appropriate radio button the module will NOT update the template database when you switch to that method. Thus if you add a new image and want to experiment with all techniques you will have to switch to each method and you MUST press the Train button in order to ensure that the template database is up-to-date.

Feature Points

This method will identify interesting points within the template using a modified fast [Harris](#) feature detector and match those points with those

detected within the current image. The identified points are typically corner-like points that exhibit restraining forces in each direction (i.e. the highest edge signal in both X and Y directions will be maximal at the point's position). This helps to stabilize point choices in both the template and image such that most (but not all) points will be detected between the template and image. Once this correlation has been done the most likely template is then tested for at that location using a slower cross correlation technique. This technique is a good standard technique assuming there is enough internal texture within the object (think of a book cover) and is fast enough for most purposes.

Keep in mind that as this technique mainly uses corners as features motion blur will cause all those features to disappear and therefore not match correctly. Thus if you have a lot of motion blur within images you wish to match against you may need to decide on an alternative technique.

Objects to be detected MUST have internal texture as this module relies in inner feature points as the primary object identification technique. If you are interested in just the shape of an object the [Shape Matching](#) module will be of better use or the Shape method mentioned below.

This method will check for translation, scale and orientation.

Shape

The shape matching method is similar to the [Shape Matching](#) module but this method works on intensity images contrary to the Shape Matching module which uses binary (Black&White) images. The shape method will analyze the template and current image for correspondences in shape and determine which parts of the image best contain a particular template.

Due to its reliance on shape the shape matching method can determine any size and orientation (including X and Y location) of the template image within the current image in about the same time as the cross correlation method does just X and Y location. Its speed and flexibility make the shape matching method very useful when limited internal template texture is available. For example, labels, street signs, text, logos, fiducials, etc. are all ideal templates for the shape matching method as they consist of non-textured areas and rely on shape as their primary form of identification.

It is recommended to use templates that are as large as possible with regards to what may be seen in the current image. If small templates are used and matching to larger possible targets in the current image is attempted a mismatch will occur as more detail will be in the current image than what is contained within the template. Thus it is important to use a template by cropping the largest size seen of the template in any test or production images.

This method will check for translation, scale and orientation.

Haar

A very popular face recognition technique uses Haar like filters to determine a set of pixels comparisons that best represents the concept of a face. While this has its uses the Haar technique can also be applied to specific object recognition. In this scenario many high intensity versus low intensity checks are created that when run in sequence will identify a template with a high probability. These checks can be run across the image very quickly and even adapt to size differences but they cannot be quickly rotated and thus are restricted to a single orientation.

Note that contrary to typical Haar training this method only requires ONE sample image to recognize the image and does NOT create a generic class based on that image. Thus if you want to recognize a class of objects you will need to include a couple images to best represent that class.

This method will check for translation and scale.

Cross Correlation

This method uses a well known and established technique for object recognition. The normalized cross correlation method has been widely used in many applications and can be one of the more stable techniques to use. The algorithm behind the cross correlation technique is, unfortunately, very CPU intensive and therefore should be used with care otherwise significant time may be spent on searching for objects within an image.

The cross correlation algorithm will use the template as saved in the trained folder and compare the template pixel by pixel at each pixel location within the current image. For example, if your template is 50x50 pixels large and your current image is 320x240 large this will mean there are approximately $2500 \times 76800 = 192,000,000$ pixel comparisons to make. This brute force method is entirely too slow for reasonable usage so tradeoffs between the size of the template versus the size of the image is made in order to speed up comparisons. While this does significantly speed up the algorithm some accuracy is forfeited as a result of that speed.

While normalized cross correlation does attempt to best deal with lighting changes between the template and the current image it is not performed within the image itself, thus while the template and current test image can be overall darker or lighter if a shadow is cast across the template that is not also within the current image the recognition process will fail.

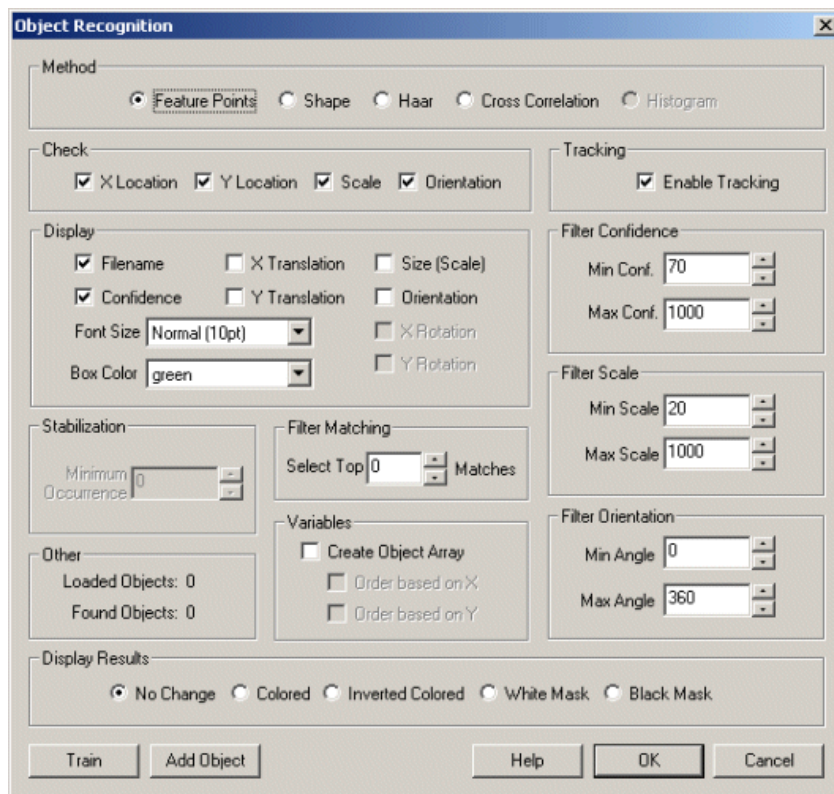
To further reduce performance requirements the cross correlation method ONLY checks for templates of the same size AND orientation. Thus a template is only searched in the X and Y direction. If the current image contains the template at a different size it will NOT be recognized (but can still be tracked). If this is a requirement for your project and you decide to use cross correlation you will need to add the template in different sizes to the training folder. We recommend changing template sizes in increments of 10% which typically will provide adequate coverage.

If orientation is required you may attempt to use the [Orient Image](#) module prior to cross correlation which will help to specify a standard orientation prior to matching. Note that you will need to add the 180deg rotated template within the training folder if you decide on this direction as the Orient Image will most likely have a 180deg symmetry and therefore not always align to the same 180 direction.

For comparisons that are image to image (meaning NO change in X or Y) the cross correlation technique can be one of the fastest. Thus if you goal

is to find an image that exists within a known database without any size, orientation, horizontal or vertical shifts the cross correlation technique is a good technique to attempt.

Interface



Instructions

1. Train - Press the train button and enter the full path to the templates folder that you want to train on. Press the START button to start training. After it is complete press the ok button.

2. Display - You should now notice the objects being recognized in the main RoboRealm window. Matches are displayed as green boxes. Note that the 'name' of a object is the name of the template image filename without the extension. So you can change the names by changing the filenames and re-train. Note that size is based on the size of the trained template where 100% is the same size as the trained template.

By default the Filename and Confidence are displayed. You can add or remove information from being displayed by changing the checkboxes in the Display area. Note if too much information is displayed you might want to change the Font size to a smaller size.

- Filename - the name of the object.
- Confidence - how strongly the module believes this object matches one in the database folder.
- Coordinates - the location of the matched object.
- X Translation - the X location with respect to the center of the screen of the object.
- Y Translation - the Y location with respect to the center of the screen of the object.
- Size - the size of the object relative to that in the database.
- Orientation - the Z rotation or angle of the object (spin)

3. Box Color - You can change the match box color by changing the Box Color.

4. Check - If you want to restrict where the module will search for a template you can restrict the search space by un-checking the appropriate checkboxes. For example, if you know the template will always appear in the same size you can uncheck the Scale checkbox. Removing search requirements will help to speed up the search. Note that if you remove X then the module will search along the vertical center of the image for the template, likewise with removing Y will cause the search in the horizontal direction. If you remove both X and Y then the center of the image will be checked for the presence of the template.

5. Tracking - Recognizing objects can be a very taxing process on your CPU. To remove some of this load you can chose to have tracking enabled which will quickly look for the recognized object in the previous image in its immediate area and adjust for any movement. Tracking is much faster than recognition and can also track in more dimensions than the original recognition method. For example, if you use the Cross Correlation technique to recognize an object and have tracking turned on, once recognized the template can now be scaled and rotated and still be identified even though Cross Correlation does not support scale or orientation changes!

6. Tracking Confidence - Once an object is recognized the actual tracking (recognizing from one image to the next) can use a lower confidence since the movement of an object can cause temporary distortions or highlights/shadows within the image. You can use the Tracking Confidence

threshold to be lower than the recognition threshold to ensure that once objects are recognized they will still be tracked under harsher image conditions.

7. Filter Confidence - Many candidate objects do pass the basic match requirements but do **not** match with a known object very well. You can remove those unwanted matches by increasing the Min Confidence. This removes bad matches from being made.

8. Filter Scale - Each recognized object has a scale associated with it that can be used to remove larger or smaller matches.

9. Filter Orientation - by default objects are recognized in any orientation. You can use the Filter Orientation to remove those objects outside of your desired orientation. For example, 15, 345 would remove any object ± 15 degrees from that what is in the database.

10. Stabilization - When moving the camera motion blur can cause incorrect low confidence matches to appear. By increasing the "Present After" you are requiring a object to appear for a set number of frames before being reported as a match. This essentially removes the "flickering" of low matched objects that can cause noise in the results. Note that if you are working with static images you will need to set this to zero! In the same way the "Absent After" number will only remove an object from being tracked after the specified number of frames. This will help prevent the momentary disappearance of the object due to pixel noise.

11. Filter Matching - If you know you only want to track a single object you can specify a 1 in the top matches box which will only return a single best recognized template from the current image. Likewise, if you know that you will never have more than 3 recognized objects you can specify 3 in the same box. This helps to remove lower matching templates without having to adjust the confidence threshold.

12. Overlap - Specifies the amount of allowed overlap between objects. Often similar objects may be detected in a similar location which may not be desirable. The amount of overlap will remove those less confident objects to result in a single dominant recognition. You can use this (default setting of 20%) to ensure that no objects overlap by more than 20% or by whatever percentage you use.

13. Variables - To use the results of the object module in your own application or in other RoboRealm modules you can select that the object array be created. This array has all the collected information about a detect object. See below for the array format.

14. Treat black as mask - Select if your templates contain pure black areas that should NOT be used in matching a template with a potential match. This helps to eliminate parts of a non-square template and create a higher confidence match result.

15. Display Results - By default No Change is selected which does not modify the current image in any way other than annotating the objects with the above colors. If you want to use the results in other modules you can select to see

- Colored - displays only the objects with their original pixel colors
- Inverted - displays everything else but the object (which is blacked out).
- White Mask - white where the objects are, black everywhere else
- Black Mask - black where the objects are, white everywhere else

Example

Source

Object_Recognition

Recognition of a basic shape. As the object becomes less planar the match confidence will decrease.



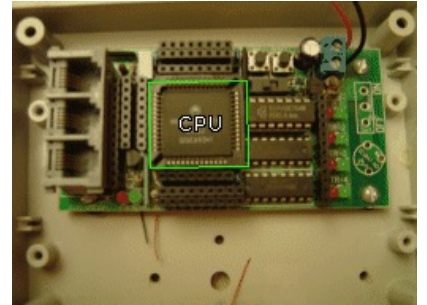
Scale invariant recognition of the same object. Notice significant lighting change.



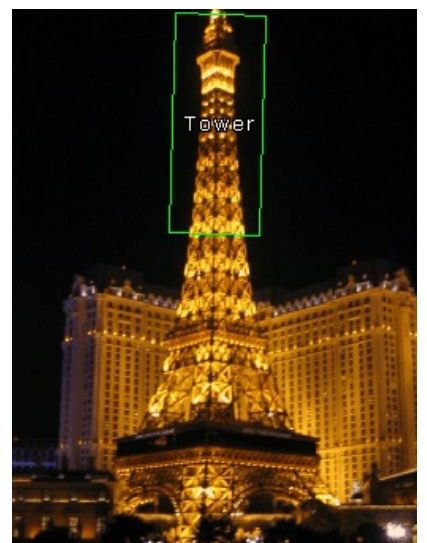
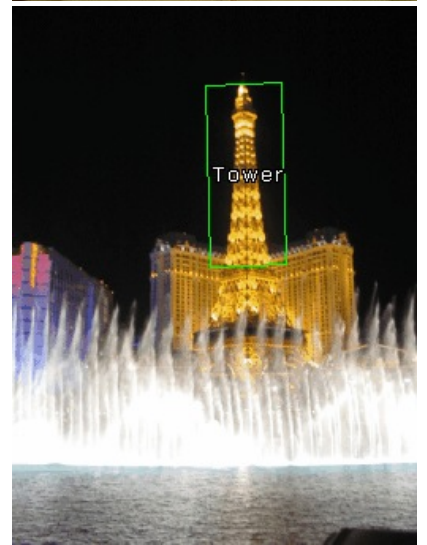
180 degree rotated objects are still recognized.

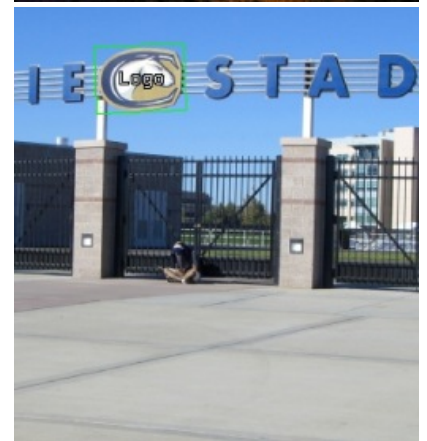
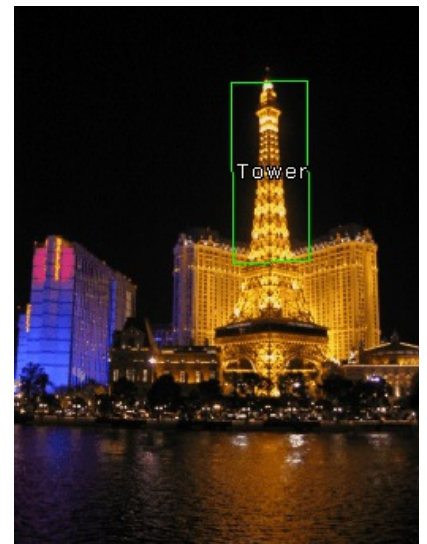


These recognition methods are useful for parts identification and verification. But try to keep a perpendicular view without too much camera distortions.



These techniques can work on 3D objects but not if the view angle (X&Y) changes significantly. Taking images of far away objects will normally work better as you'd have to move quite a bit to change the view angle.





Different techniques will work better on different objects. In this case there are not enough feature points to detect the object ... but the Shape technique has plenty to work with.



These techniques are good for specific object identification but do NOT work as a general classifier for objects. In this case while very similar the open window is different enough to fail as being identified as a window. You can always include more templates but there is a limit.

Variables

OBJECTS - contains an array of detected objects.

OBJECTS_PATH - string that contains the names of the objects.

This single string contains all the names and is used

by #13 and #14 below to extract out the actual name.

OBJECT_COUNT - the number of objects recognized and the
total size of the OBJECTS array (OBJECT_COUNT*15)

The array contains blocks of 15 values per detected circles. The
elements are as follows:

The OBJECTS array is composed of 15 numbers as follows:

Offset	Contents
0	Match Confidence 0-100
1	Point 1 X coordinate
2	Point 1 Y coordinate
3	Point 2 X coordinate
4	Point 2 Y coordinate
5	Point 3 X coordinate
6	Point 3 Y coordinate
7	Point 4 X coordinate
8	Point 4 Y coordinate
9	Translation in X
10	Translation in Y
11	Scale 0-100
12	Orientation 0-2PI (radians)
13	Path start index
14	Length of path

For example, to print out all found objects use the VBScript module
and the following script.

```
objects = GetArrayVariable("OBJECTS")
names = GetStrVariable("OBJECTS_PATH")

if isArray(objects) then
  if ubound(objects) > 0 then
    for i=0 to ubound(objects)-1 step 15
      nstart = objects(i+13)
      nend = objects(i+14)
      write "Conf: " & objects(i) & "% Path: " & _
        mid(names, nstart, nend) & vbCRLF
    next
  end if
end if
```

If you wanted to write the COG to a file you can VBScript to format and write it to a file.

```
objects = GetArrayVariable("OBJECTS")
```

```

if isArray(objects) then
  if ubound(objects) > 0 then

    Dim fi
    Dim fso

    set fso = CreateObject("Scripting.FileSystemObject")
    set fi = fso.OpenTextFile(CStr("c:\temp\test.txt"), 8, true)

    if err.number = 0 then

      fi.WriteLine "x,y"

      fi.WriteLine (ubound(objects)+1)/15

      for i=0 to ubound(objects)-1 step 15

        x = (objects(i+1)+objects(i+3)+objects(i+5)+objects(i+7))/4
        y = (objects(i+2)+objects(i+4)+objects(i+6)+objects(i+8))/4

        fi.WriteLine x & "," & y & "," & objects(i+12)

      next

    end if

    fi.close

    set fi = nothing
    set fso = nothing

  end if
end if

```

For ease of access the largest object's information is also available in the following variables

OBJECT_CONFIDENCE - the confidence of the best matched object

OBJECT_NAME - the name of the matched object

OBJECT_FILENAME - the filename of the object

OBJECT_FOLDER_X - the folder sequence that the object is in
(replace X with 1, 2, etc.)

OBJECT_SIZE - the size of the best matched object

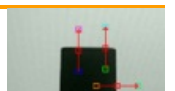
OBJECT_X_COORD - the x location relative to center screen of
the best matched object

OBJECT_Y_COORD - the y location relative to center screen of
the best matched object

OBJECT_ORIENTATION - the orientation (spin/roll) of the largest object

See Also

[Shape Match](#)
[Fiducial](#)
[Image Matching](#)

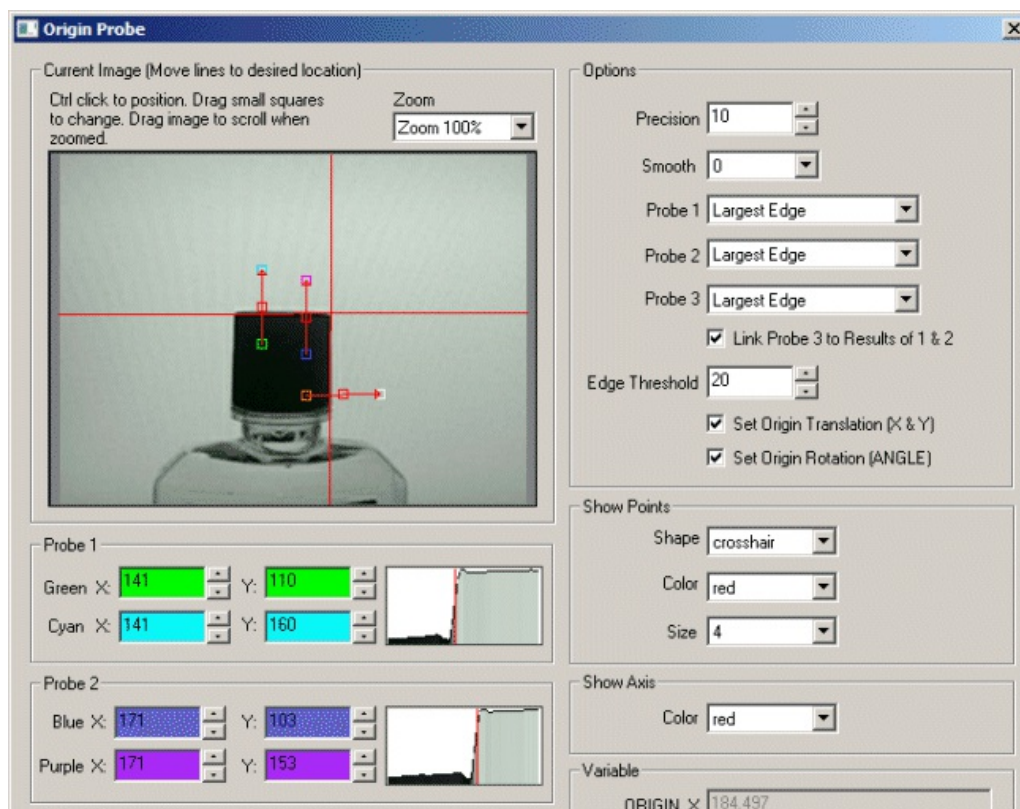


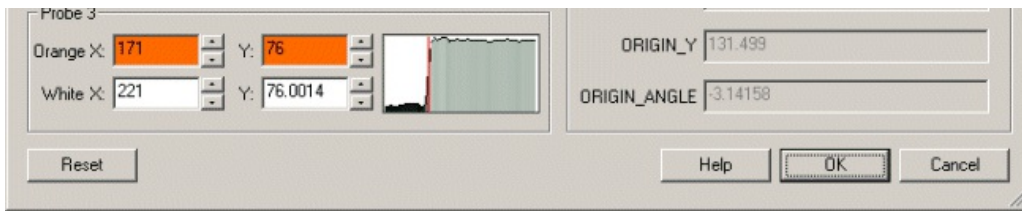
The Origin Probe module allows you to specify a reference position that can be used in other probe modules to specify relative placement based on the origin. This is useful if you cannot use long measurement probes with your object due to size constraints but can use long probes to create a reference position. This enables the system to compensate for large movements without compromising measurement positioning.



The three probes are means to be used to specify two lines whose intersection is considered the origin or 0,0 position. Use the two probes to specify the X axis and the third to specify the Y axis. By dragging the probes around your image you will see the generated axis lines in the main RoboRealm window being generated. Once complete you can then add additional probe modules into the processing pipeline and select the "Use Origin Variable" within that probe to base its position from the origin specified in this module.

Interface





Instructions

1. Current Image - move the three lines such that they specify the 0,0 location correctly. Note that the two parallel lines should be used to specify a single axis with the third being used to specify an orthogonal axis. You can use the red square to move the entire line, the endpoint squares to move each endpoint individually or change the coordinates using the text boxes below in the Coordinates area.

Use CTRL-click to move the entire probe to a different location. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Precision - the precision specifies how each pixel should be divided in order to determine the exact place that the edge occurs. If you have a blurred image you may need to reduce the precision to negative numbers. Negative numbers mean that an edge is comprised of several pixels (i.e. white to black transition occurs slowly over several pixels). If positive the precision will determine where the edge most likely exists between two successive pixels by interpolating values between the two pixel intensities.

3. Smooth Edge - to reduce noisy edges select the amount of smoothing that should be applied to the edge prior to edge detection. This prevents a single sharp noise pixel from being detected as an edge.

4. Probe Types - Specifies how the points within each probe are located.

Largest Edge - The point is determined by the largest edge encountered along the probe line. The largest edge is defined by the different from one pixel intensity to the next.

Largest White to Black Edge - The largest transition from a light to dark pixel.

Largest Black to White Edge - The largest transition from a dark to light pixel

First Edge - The first edge that is larger than the specified threshold

First White to Black Edge - The first light to dark pixel edge that is larger than the specified threshold

First Black to White Edge - The first dark to light edge that is larger than the specified threshold

Darkest Pixel - The darkest pixel along the probe from an intensity perspective

Lightest Pixel - The brightest pixel along the probe from an intensity perspective

5. Link Probe 3 - Specifies that probe 3's (orange to white) position is relative to the line found by probe 1 and 2. The reason for this is that a small corner may not be easily detected if rotation of the part is present. Linking the probe to current results allows for smaller corners to still be detected through a larger range of rotation.

6. Edge Threshold - to eliminate very weak edges or pixels select an appropriate edge threshold (0-255) that will remove edges whose edge intensity is below the threshold and pixels whose intensities are below the threshold. Note that smoothing the edge will also reduce the edge intensity and thus the Threshold will need to be adjusted after the smoothness is specified.

7. Set Origin Translation/Rotation - By default the origin probe will set both translation and rotation variables (ORIGIN_X, ORIGIN_Y, ORIGIN_ANGLE) but sometimes only one or the other is needed. For example, depending on your particular image the angle may cause more vibration than it benefits. These checkboxes allow you to disable the setting of these variables to avoid undue noise.

8. Show Points - to visually see which edges are being detected select the appropriate Shape, Color and Size of the marker that will indicate where in the image the edges have been detected. Note that this appears in the main RoboRealm GUI window.

9. Show Axis - Specify the color you want to use to show the origin reference frame in the main RoboRealm GUI window.

10. Probe Coordinates - To fine tune the probe positions you can type in numbers into the appropriate coordinate boxes (they are labeled with the color of the box as seen in the Current Image area) or use the up/down spin controls to increment or decrement the numbers accordingly. Note that the text box colors correspond to the probe drag square colors.

11. Variables - The variables generated specify the translation and rotation needed in order to create a position relative to this new origin. The three variables are then used by other modules to create the reference position. Note that you can also generate or modify these variables in other modules as needed as they are just regular RoboRealm variables.

Variables

ORIGIN_X - the X (horizontal) offset of the generated origin

ORIGIN_Y - the Y (vertical) offset of the generated origin

ORIGIN_ANGLE - the angle of the generated origin

ORIGIN_Y - the Y (vertical) offset of the generated origin

ORIGIN_ANGLE - the radian angle (rotation) of the generated origin

See Also

[Circle Probe](#)

[Line Probe](#)

[Edge Probe](#)

[Thickness Probe](#)

[Peak Valley Probe](#)

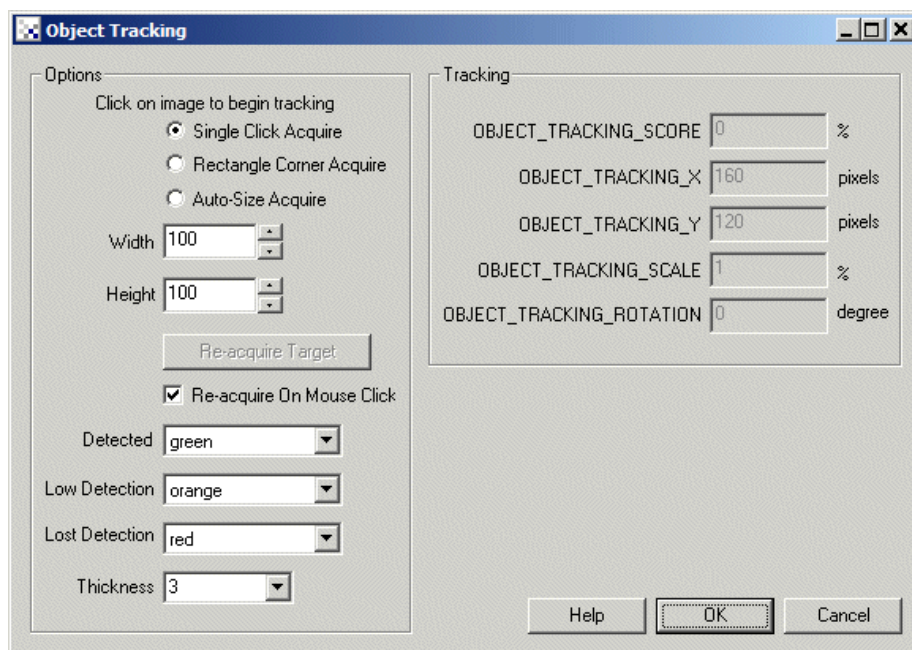
Object Tracking



The Object Tracking module provides a way to track a particular object within an image by manually identifying the initial location of the object and its approximate size. Once this is done, the module attempts to correlate that object in successive images to the originally selected object to produce new location coordinates. As the object moves in 3D the module will record new views of the object in an attempt to update its internal model to avoid object tracking loss.

Unlike Blob Tracking this module expects that the raw image (without thresholding) is used to track objects. Thus objects need to be easily differentiated from their background by either intensity (texture) or color.

Interface



Instructions

1. Options Acquire - Select how you would like to manually specify the location and size of the object to track

Single Click - By clicking on the main GUI image the selected location and specified Width and Height will create an object window which initiates tracking. This mode is appropriate when the approximate size of the object remains constant but tracking is lost due to obstructions.

Rectangle Corner - By clicking on the upper left and lower right or upper right and lower left corners (diagonally) you can specify the location and size of the tracking window. This works best when the object is not moving very quickly and you need to specify a different size each time tracking is initiated.

Auto-Size - By clicking on the main GUI image the module will analyze the clicked location to find a natural (i.e. low textured area) that encompasses the clicked location. This is an attempt to automatically identify the object size. Note this will only work in those locations where the object exists in a low textured background.

2. Re-acquire Target - To reacquire the target when not in single mouse click mode (to allow for image scrolling) press the Re-acquire Target button. When this is pressed the next click (or clicks) in the main GUI image will be used to locate the object.
3. Re-acquire On Mouse Click - When selected this mode will cause each mouse click on the main GUI image to become the new location to track. This is very useful if the object tends to get lost frequently and you do NOT need to scroll the image within the GUI. When this mode is enabled you can still scroll the image using the scroll bars but any click in the main image (in an attempt to drag the image) will result in a new location for tracking.
4. Detected Color - The Color used when an object is well detected (>70 score).
5. Low Detected Color - The Color used when an object's detected score is low (<70).
6. Lost Detection Color - The Color used when an object detection is low (=0)
7. Thickness - The thickness of the detection window/box when graphically drawn on the image.

Variables

OBJECT_TRACKING_SCORE - How well the object is being recognized by the module (assuming previous image model).

OBJECT_TRACKING_X - The X pixel coordinate of where the tracking currently identifies the object.

OBJECT_TRACKING_Y - The Y pixel coordinate of where the tracking currently identifies the object.

OBJECT_TRACKING_SCALE - The scale change of the current object relative to the initially selected window.

OBJECT_TRACKING_ROTATION - The rotation change of the current object relative to the initially selected window.

See Also

[Blob Tracking](#)
[Line Counter](#)

Shape Matching

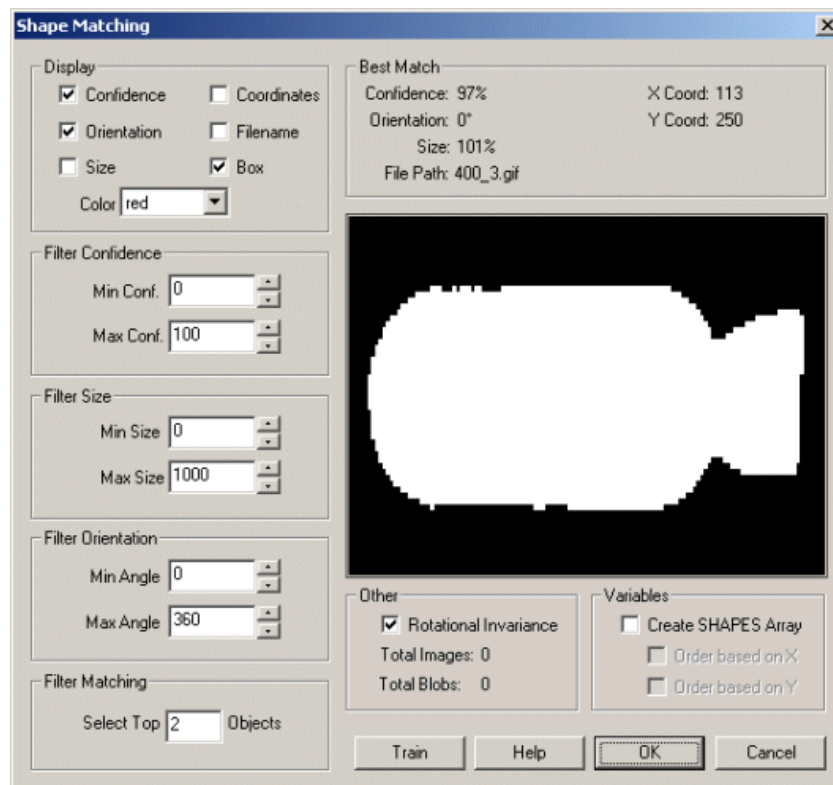
The Shape Matching module provides a way to match a binary image to a known database of images. The module is used to recognize shapes and provides statistical relationships between the currently viewed object and that stored in the image database. Note that the shape matching is based on actual image shape and NOT on direct pixel matching as done in image template matching. In this way the matches are invariant to shape translation, size and orientation.

The module uses a specified folder for training. It expects to see black and white images in that folder that are processed for shape information. (White being the shape, black being the background). Using this database of images, new images fed in from the camera (after being thresholded) will be compared to the known database with resulting similarity being displayed.

The module does NOT adapt to changes in perspective. Thus if your shape is on the floor with your camera pointing outwards, you will probably not get very good matches. You'll need to use the [Perspective module](#) to adjust the image such that the shape appears as if the camera was looking

right down on the shape. Alternatively, you can snap many images of the shape given perspective distortion which would provide some stability ... but not as reliable as warping the image prior to matching.

Interface



Instructions

1. Press the 'Train' button to create a new image database. This will popup the training dialog. Training images should be located in a specified folder. If you need to train on a live image from the camera, take a snapshot of that image, threshold to black and white (white being active signal) and save that image to a known folder. This folder becomes the 'training' database. When new images are added or old ones deleted you will need to retrain.
2. Click on '...' to select the folder where the images you would like to train on are. NOTE: the shape matching only works on **binary** black and white images with the object being in white and the background in black.
3. Select 'Start' to start training. If you would like training to happen automatically whenever a file is added, updated or deleted from the specified folder select the 'Monitor Folder' checkbox.
4. After training click on OK and begin viewing which images match the current image.
5. Click on OK to remove the matching dialog. This window can be viewed again at any time by 'editing' the module again.
6. Depending on your matches you may want to filter out specific objects. In order to get a sense of what you can use to filter out objects click on the Display checkboxes which will show the match information about each object in the main RoboRealm image window.

Confidence - how well the shape matches those in the database. Note that this refers to the best available match in the database of trained shapes.

Orientation - how the shape would have to be rotated to match the closest match in the database. The angle of rotation is relative to the orientation provided in the shape database. Thus exact shape matches will be 0 degree rotation whereas mirrored objects would be rotated 180 degrees.

Size - the relative size of the shape to that in the database. 100% means the shape is the same size as that in the database. 150% means that the shape is much larger than that in the database.

Coordinates - the center of where the detected shape and the match in the shape database best correlate. This is the center of the boxes draw around the matches.

Filename - the name of the image file that represents the best match found in the database. I.e. this was the original filename of the image as it went into the shape database.

Box - shows a bounding box around a matched shape that represents the scaled dimensions of the entire shape image in the database

BOX shows a bounding box around a filtered shape that represents the scaled dimensions of the entire shape image in the database.

7. Filter Confidence - to remove shapes that are below a confidence threshold set the number in the Min Conf text area. If you wish to eliminate shapes of high confidence then enter a lower number in the Max Conf.
8. Filter Size - used to remove those shapes that are either much larger or smaller than the original used in the database.
9. Filter Orientation - used to remove shapes that are different in orientation to the shapes in the database. For example, to remove a 180 degree rotated image enter 170 for Min Angle and 190 for Max Angle to provide 20 degrees of noise tolerance. To do the same around 0 degrees use 350 in the Min and 10 in the Max. Note that angles reset to 0 above 360.
10. Filter Matching - if you just want to select the top 1 or 2 matches enter that number in the text box. This will remove all but the top X number of matches.
11. Other - unselect the "Rotational Invariance" checkbox if you do not want the system to ignore orientation during matching. This can be illustrated when matching the digits '9' and '6' which are often a 180 degree rotation of each other depending on font used. Unchecking the rotational invariance checkbox will ensure that templates match based on their current rotation.
12. Shapes Array - for further processing or exporting of data you can select the "Create SHAPES Array" which will create an array of the results of this module to be accessed by VBScript or external modules. On selecting this checkbox the following VBScript program could function.

```
shapes = GetArrayVariable("SHAPES")
names = GetStrVariable("SHAPES_PATH")

if isArray(shapes) then
  if ubound(shapes) > 0 then
    for i=0 to ubound(shapes)-1 step 9
      nstart = shapes(i+7)
      nend = shapes(i+8)
      write "Conf: " & (shapes(i)/1000) & "% Path: " & _
        mid(names, nstart, nend)
    next
  end if
end if
```

The SHAPES array is composed of 9 numbers as follows:

Offset	Contents
0	Match Confidence 0-100
1	Orientation 0-360
2	Relative Size
3	X min coordinate of bounding box
4	X max coordinate of bounding box
5	Y min coordinate of bounding box
6	Y max coordinate of bounding box
7	Path start index
8	Length of path

The Path index is a number that defines an offset into another variable SHAPES_PATH that contains all the path information for a particular match. Using the SHAPES_PATH and the length of the path you can extract out the path for the match as illustrated in the small script above.

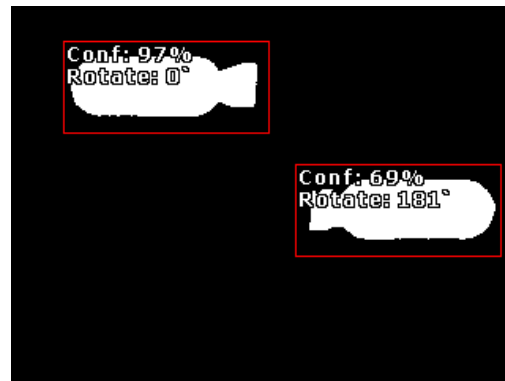
Note that the creation of a binary image shape matching database will create a roborealms.shape file in that folder. This file contains the image information needed for matching and is a more compact form than the original image pixels.

You can change the ordering of the shapes array by selecting the appropriate checkbox. The default order of the array is in order of encounter from

a bottom to top, right to left scanning. Selecting either checkbox will order the array based on the particular axis with respect to the center of the object.

Example

The user interface image above shows the closest match to



Note that the appropriate confidence, orientation, size, and coordinates are provided in the interface. You can access these statistics using RoboRealm variables within the appropriate modules (like the VBScript module). The following variables (in addition to SHAPES Array) are defined for the top match only:

SHAPE_FILENAME - filename of matched shape

SHAPE_FOLDER_1, SHAPE_FOLDER_2, etc - sub folders of matched shape

SHAPE_ORIENTATION - orientation of current shape relative to match

SHAPE_SIZE - relative size of match to that stored in database

SHAPE_X_COORD - x axis offset of shape relative to that in database

SHAPE_Y_COORD - y axis offset of shape relative to that in database

SHAPE_CONFIDENCE - how close the current object matches that

stored in database

See Also

[Object Recognition](#)

[Image Matching](#)

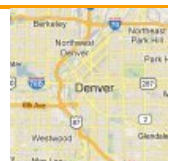
[Fiducial](#)

[Auto Threshold](#)

[Center of Gravity](#)

Target Localization

The Target Localization module provides detection of a specified target to calculate the position of the camera/robot based on the orientation and size of the target. In order for the target to be accurately detected a couple assumptions are made.



1. The target is composed of horizontal and vertical lines. Some curved edges are acceptable but the majority of the target is composed of straight lines.

2. The target's edges are sharp and of high contrast with respect to the rest of the image.

3. The target is perpendicular to the plane of movement. I.e. the camera/robot moves on the ground/horizontal plane in front of the target. While

3. The target is perpendicular to the plane of movement. i.e. the camera/robot moves on the ground/horizontal plane in front of the target. While some tilt and roll is permitted, significant tilt or roll will cause detection failure. It is assumed that your camera/robot stays right side up.

4. Most of the target is in view. Some obstructions are permitted but enough corners must be visible for detection to work.

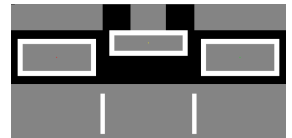
5. The target is planar, i.e. it must be on a flat surface. If the target is bent or curved in any way detection will fail.

Given these assumptions the module will return with X and Y coordinates of the camera relative to the target such that you can know your approximate location.

The module has no specific units specified (i.e. meters, feet, etc.). The results will be in whatever units you use for the size of the target. Thus, if you use feet for the size, the X and Y results will be in feet. If meters are used, the results will be in meters.

Target Image

The target image specifies what pattern to detect. The white area represents the target whose outline consists of straight lines. These edges will be detected within the current image. The black areas represent the background of the target where no edges are expected to be present. This is used to provide detection context within a cluttered scene.



The gray areas represent areas that can be disregarded since they are unknown and can change depending on the background scene.

The intensity levels of the image are very important. If the target is not perfectly white (255,255,255), the black area not perfectly black (0,0,0) and the gray area not (128,128,128) the template will be incorrectly interpreted and fail. For this reason, only lossless image files (GIF, PNG, PPM, TIFF) are accepted as template files. The use of lossy formats (JPG) would cause the intensity to change.

Hotspots

Once the target is detected, you may need to know the position of points within the target. To do so, you can specify hotspots. There are 5 hotspots that you can define. They are represented by a + pattern within the template in unique colors. The 5 permitted colors are Blue (0,0,255), Green (0,255,0), Cyan (0,255,255), Red (255,0,0), Purple (255,255,0) and Yellow (255,0,255). In order for detection of hotspots to work, they MUST be in a + formation

[] [*] []

[*] [*] [*]

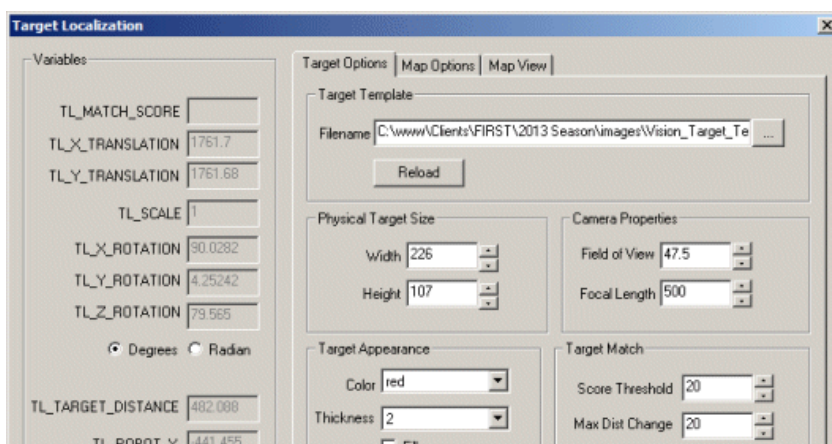
[] [*] []

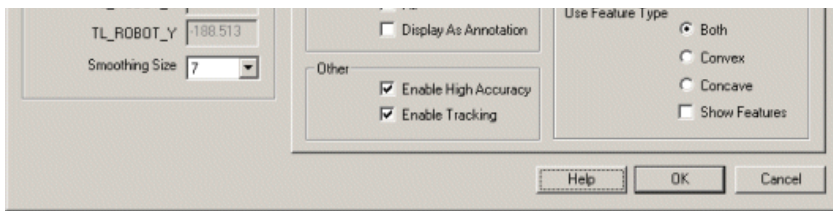
and be the EXACT color. Failure in the pattern or color will stop the hotspot from being recognized and position provided. For all hotspots, the distance, horizontal and vertical angles are reported based on the color of the hotspot, i.e. the variable name will include the color (red, blue, etc.) of the hotspot.

Map

You can specify a additional image that provides context for the X and Y coordinates. This background image can help to visually identify where you are with respect to the target. This is a pixel based image that can be any image format loaded by RoboRealm. The dimensions of the are important. Depending on what units you use for the target size, a pixel within that image will represent a square unit of that dimension. For example, if you specify the target size in inches, each pixel within the map image represents a square inch in real space. Keep in mind that position 0,0 is right in the middle of the target (which is an unrealistic point since you'd not be able to see the target and therefore not know where the camera is). The map may be larger than this space which requires the use of the map target X and Y points to adjust to this location.

Interface

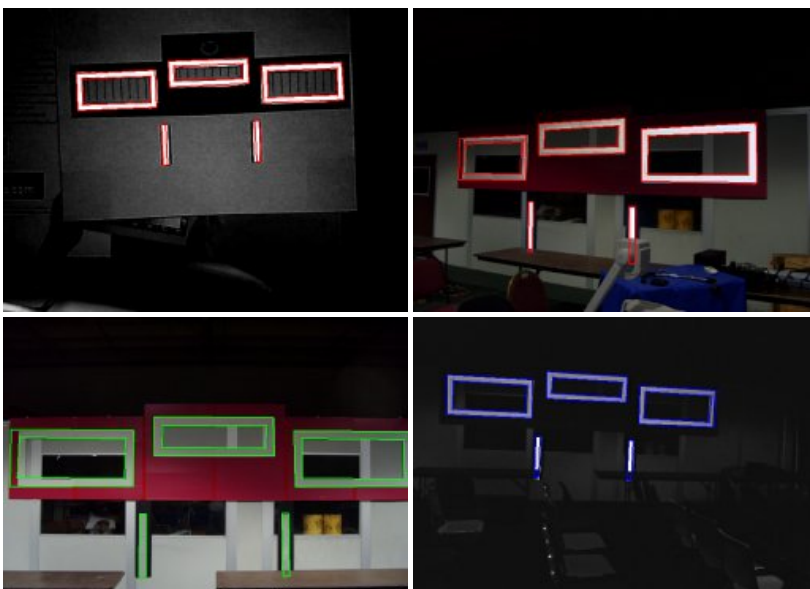




Instructions

1. Target Template - Specifies the filename of the target image.
2. Physical Target - Specify the actual size of the target in whatever units are most convenient.
3. Camera Properties Field of View - Specifies how many degrees the camera can see/image. The larger the field of view the more context a camera can see.
4. Camera Properties Focal Length - The focal length determines the amount of perspective distortion that occurs when looking at the target from the side. Check with your camera vendor for the focal length or use a side view of the target and change the focal length until you get a good match.
5. Target Appearance - Specify how the detected target should be identified in the current image.
6. Target Match Threshold - Specify how well the match should be in order to be detected.
7. Max Dist Change - Specify the maximum distance the robot can move in a short period of time. This value is used to ignore sudden incorrect detections of the target which can dramatically move the position very quickly.
8. Use Feature Type - Specify which feature type should be used. The target localization module looks for corners in order to determine the mapping between actual and template. Sometimes, the outer (convex) corner is more reliable than the inner (concave) corner depending on the target used. Typically corners that are created within the target as apposed to on the border of the target are more reliably detected since the background may change.
9. Show Features - This checkbox will help indicate the detected corners.
10. Map Options Image - The bitmap of the background map.
11. Map Target X/Y - The offset within the map that defines the 0,0 point (i.e. middle of target).
12. Robot & Route Color - The respective color of the robot and taken route.
13. Map View - The robot position on top of the bitmap map.
14. Center on Robot - Specifies that the map should center on the robot location. When this is unchecked you can scroll the map area by dragging it with the mouse to view other positions.
15. Smoothing Size - The raw number of the X and Y position of the target can be very noisy (sporadic) if not smoothed with previous values. The more smoothing applied the more stable the position is. However, with more smoothing comes slower updates in that the reported position will lag a little behind reality.

Examples



Variables

TL_MATCH_SCORE - The match score of the detected target

TL_X_TRANSLATION - The number of pixels the target is off from horizontal center of screen

TL_Y_TRANSLATION - The number of pixels the target is off from vertical center of screen

TL_SCALE - The scale of the target relative to the template

TL_X_ROTATION - The amount of tilt rotation the target is relative to the camera view (up or down of target)

TL_Y_ROTATION - The amount of pan rotation the target is relative to the camera view (left or right of target)

TL_Z_ROTATION - The amount of in plane rotation of the target

TL_TARGET_DISTANCE - The distance of the center of the target from the camera

TL_ROBOT_X - The X coordinate of the camera relative to the target

TL_ROBOT_Y - The Y coordinate of the camera relative to the target

See Also

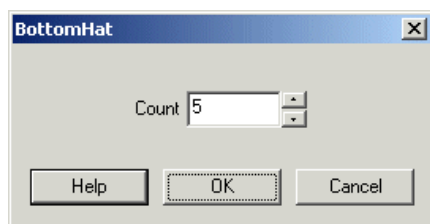
[Fiducials](#)

[Shape Match](#)

Bottom Hat

The BottomHat module performs a dilation routine (grows the current white image), followed by an erosion (shrinks the current white image) and then subtracts the original image. Dilation followed by erosion will connect objects close to each other. Subtracting the original will result in the display of just the connection points between close objects.

Interface



Instructions

1. Count - Specify the number of times to perform dilation before erosion. Higher numbers will connect more distant objects which will result in larger connections being displayed in the final image.

Example

Source Image



Bottom Hat with count of 2



BottomHat results shown in white. Original image in red.

See Also

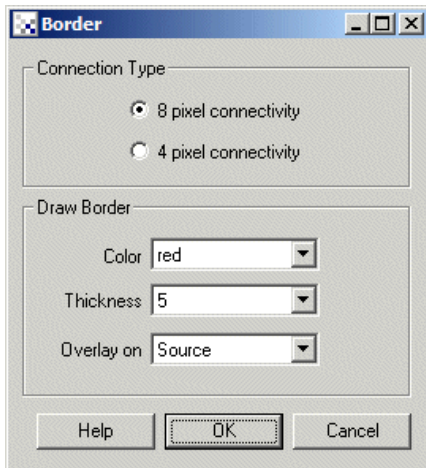
[TopHat](#)

[Close](#)

[Open](#)

Border

The Border module extracts the outline or border of a binary black & white image. The background is assumed to be black whereas the foreground image are non-black pixels.



Instructions

1. Connection Type - Specify which pixels need to touch in order for the border to be marked. 8 pixel connectivity will create a thinner border where pixels need only touch diagonally in order to be considered connected. 4 pixel connectivity will ensure that the resulting border pixels always touch vertically or horizontally.
2. Color/Thickness - Specify the color, thickness (beyond the connection type) of the border to be draw.
3. Overlay On - Specify which image the border should be drawn on. Blank will draw the border on a blank black image.

Example

Source Image



Border Image



Note that the extracted border is the border within the image. If you wish to extract a border outside of the current white area use the [Negative](#) module to switch the colors before using this module.

See Also

[Canny](#)

[Sobel](#)

Bridge

The Bridge module connects pixels that are one pixel away from each other. For example

```
0 0 1          0 0 1
1 0 1    becomes  1 1 1
0 0 1          0 0 1
```

Example

Source Image



Bridged Image



Note that the bridge is only 1 pixel in size.

See Also

[Close](#)

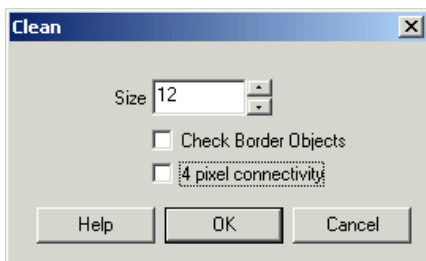
[Dilate](#)

Clean

The Clean module remove single isolated pixels from the current image.

```
0 0 0          0 0 0
0 1 0    becomes  0 0 0
0 0 0          0 0 0
```

Interface



Instructions

1. Size - Specify the size of the blob that should be removed. Zero indicates that all blobs detected will be removed.
2. Check Border Objects - Specify if blobs that include the edge of the image should be removed too.
3. 4 pixel connectivity - By default RoboRealm uses 8 pixels around the current pixel to test if the pixel is connected to another pixel. Selecting 4

4 - pixel connectivity - By default RoboRealm uses 8 pixels around the current pixel to test if the pixel is connected to another pixel. Selecting 4 pixel connectivity restricts RoboRealm to only check the North, South, West and East pixels for connectivity.

Example

Source Image



Cleaned Image with count = 1



Note that 0 removes all non-border objects.

See Also

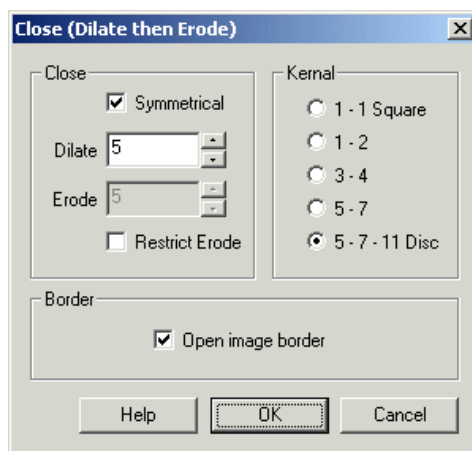
[Blob Size](#)

[Fill](#)

Close

The Close module performs a dilation routine (grows the current white image), followed by an erosion (shrinks the current white image). Dilation followed by erosion will connect objects close to each other.

Interface



Instructions

1. Dilate - Specify the number of times to perform dilation before erosion. Higher numbers will connect more distant objects.
2. Symmetrical - if you wish to dilate and then erode an unequal number of times (leaving the resulting blob smaller or larger than the original) unselect the "Symmetrical" checkbox. This will allow you to specify a different number for the Erode Count.
3. Erode - only active if module is not symmetrical meaning the image will be dilated and then eroded an unequal number of times.
4. Restrict Erode - if using an asymmetrical closing you may want to restrict the eroding process to NOT erode more than the original image. Selecting "Restrict Erode" will ensure that the eroding part of the processing does not erode parts of the original blob.
5. Kernel - Specify which kernel the close operation should use. Using a square kernel will keep square shapes square but cause round shapes to become more square. Specifying the Disc shape will cause round shapes to stay round whilst square shapes will become rounded in the corners. In between these two shapes are various other shapes that cause different side effects during closing. Chose one that best suits your needs.
6. Border - Select if you want the border to act as an object boundary and also be closed.

Example

Source Image



Close with count of 2



See Also

[Open](#)
[Bottom Hat](#)
[Dilate](#)

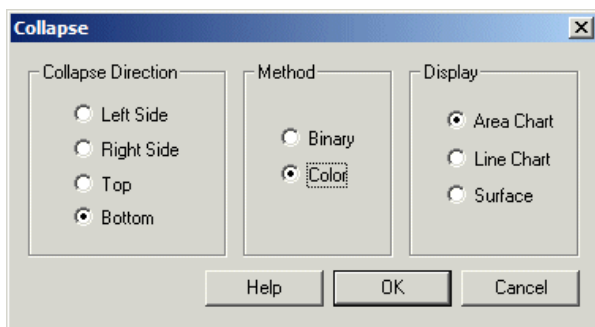
Collapse

The Collapse module is a morphological module that can be used to analyze an object's shape. The module will collapse a binary image by shifting the non-black pixels to the side of the image specified in the user interface. The resulting image appears like a histogram but instead of representing the number of colors within an image (a histogram) it relates to how many non-black pixels are in each vertical or horizontal scan line (when in Binary mode). Basically each scan line (either horizontal or vertical) is scanned for non-black pixels. If one is found the resulting height is incremented by 1. This continues for all scan lines. Note that left & right and top & bottom are symmetrical views of the same information.

This module is useful in shape analysis where decomposing the shape of an image reduces the shape detection problem to a 1 dimensional problem rather than a 2 dimensional problem. While the reduction in dimension does come at a shape uniqueness price it can be a sufficient model in many cases.

This module is often combined with the peak detection module to determine if a shape is present.

Interface



Instructions

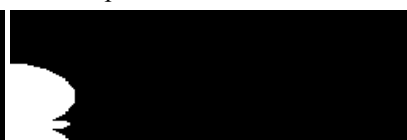
1. Direction - Specify which direction you want the current pixels to collapse/shift towards
2. Method - If you have black and white pixels you can use Binary mode to create the histogram. If instead you have grayscale or colored pixels that represent the intensity value of the pixel then selecting analog mode will sum up all those pixels and create a white pixel to represent values greater than 255.
3. Display - Choose how you'd like to display the results. Area Chart will display a solid chart in the appropriate direction. Line Chart is similar to an Area Chart but only includes the chart border. Surface will display the accumulated values as a pixel intensity line that fills the entire image. This can be thought of as looking at the results from above versus from the side like the Area and Line Charts do.

Example

Source



Left Collapsed





Bottom Collapsed



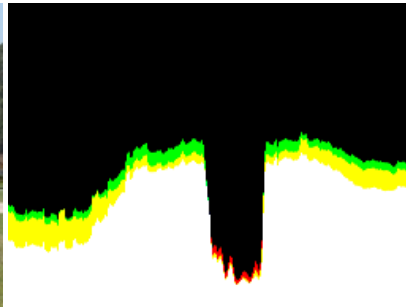
Right Collapsed



Source



Bottom Analog Collapsed



Variables

COLLAPSE_MAX - maximum collapsed value in binary mode

COLLAPSE_MAX_POSITION - position (X or Y) of maximum value in binary mode

COLLAPSE_MIN - minimum collapsed value in binary mode

COLLAPSE_MIN_POSITION - position (X or Y) of minimum value in binary mode

COLLAPSE_AVERAGE - average collapsed value in binary mode

COLLAPSE_MAX_RED - maximum collapsed red value in color mode

COLLAPSE_MAX_POSITION_RED - position (X or Y) of maximum red value in color mode

COLLAPSE_MIN_RED - minimum collapsed red value in color mode

COLLAPSE_MIN_POSITION_RED - position (X or Y) of minimum red value in color mode

COLLAPSE_AVERAGE_RED - average collapsed red value in color mode

COLLAPSE_MAX_GREEN - maximum collapsed green value in color mode

COLLAPSE_MAX_POSITION_GREEN - position (X or Y) of maximum green value in color mode

COLLAPSE_MIN_GREEN - minimum collapsed green value in color mode

COLLAPSE_MIN_POSITION_GREEN - position (X or Y) of minimum green value in color mode

COLLAPSE_AVERAGE_GREEN - average collapsed green value in color mode

COLLAPSE_MAX_BLUE - maximum collapsed blue value in color mode

COLLAPSE_MAX_POSITION_BLUE - position (X or Y) of maximum blue value in color mode

COLLAPSE_MIN_BLUE - minimum collapsed blue value in color mode

COLLAPSE_MIN_POSITION_BLUE - position (X or Y) of minimum blue value in color mode

See Also

[Threshold](#)

Diagonal Fill

The Diagonal Fill module adds white pixels to enforce 8-connectivity in the current image.

```
0 1 0           0 1 0
1 0 0   becomes 1 1 0
0 0 0           0 0 0
```

Example

Source Image



Diagonal Filled Image



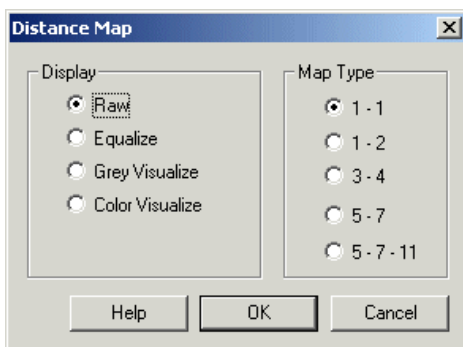
See Also

[Dilate](#)
[Bridge](#)

Distance Map

The Distance Map module performs a Euclidean distance map calculation. The distance map module replaces each foreground pixel with a number that represents the distance from that pixel to the border of the object. The distance map is useful in improving performance when calculating morphological operations such as erosion, dilation, etc.

Interface



Instructions

1. Specify the how you would like to see the distance map.

Raw - display with actual calculated numbers.

Equalized - the raw numbers are spread across a 0-255 range to improve visibility of the different numbers.

Grey Visualize - the distance map bands are colored as grey values to better show the distance map elevations.

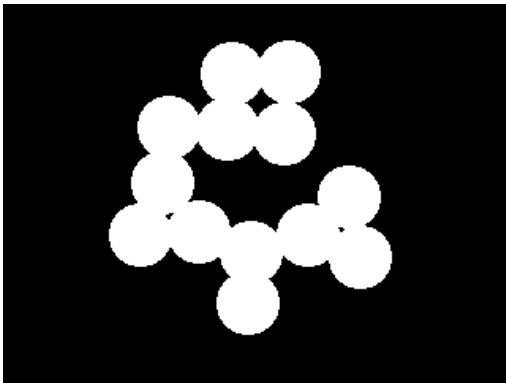
Color Visualize - like the Grey Visualize but with colored distance bands

2. Specify the map type the distance calculation should use. The map types relate to the approximated integer euclidean distances that each pixel represents. For example 1 - 2 represents the following map

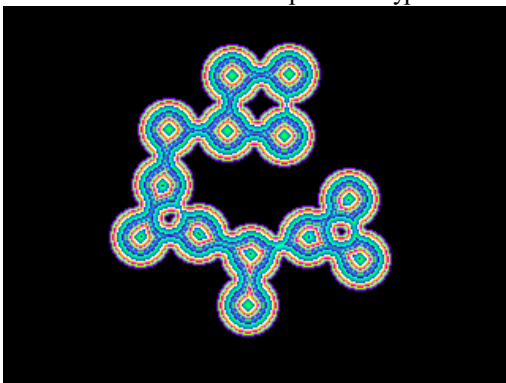
```
2 1 2
1 0 1
2 1 2
```

Example

Source Image



Color Visualized Distance Map of 3 - 4 type

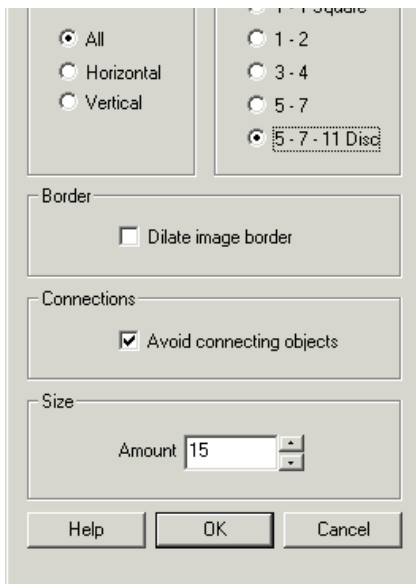


Dilate

The Dilate module performs a dilation routine (grows the current white image). Objects that are close to each other will merge based on the dilation size amount. This is useful for growing an object to a larger size than it actually is to either eliminate holes, noise or increase coverage of the underlying object.

Interface





Instructions

1. Direction - Specify which direction to dilate.

All - dilate the object on all sides

Horizontal/Left/Right - dilate only in the horizontal direction

Vertical/Top/Bottom - dilate only in the vertical direction

2. Kernel - Specify which kernel the dilation should use to perform the dilation. Using a square kernel will keep square shapes square but cause round shapes to become more square. Specifying the Disc shape will cause round shapes to stay round whilst square shapes will become rounded in the corners. In between these two shapes are various other shapes that cause different side effects during dilation. Choose one that best suits your needs.

3. Border - Select if you want the border to act as an object and also become dilated. This is useful if you wish for objects to become merged with the image boundaries.

4. Connections - Specify if you want separate objects to be merged when dilating. Selecting this checkbox will prevent objects from merging.

5. Size Amount - Specify the number of times to perform dilation. Higher numbers will connect more distant objects and/or fill larger holes.

Example

Source Image

Dilate with count of 2

See Also

[Erode](#)

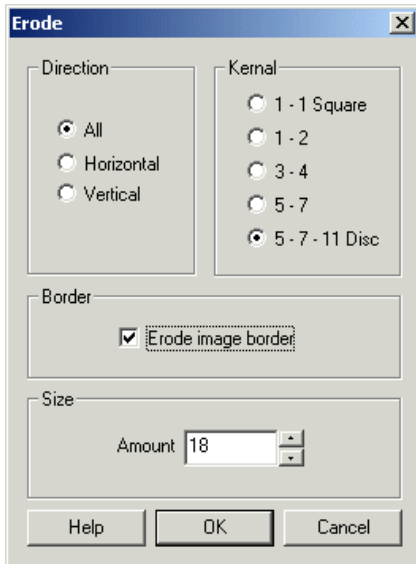
[Close](#)

[Open](#)

Erode

The Erode module performs an erosion routine (shrinks the current white image). Objects that are connected with other objects will become separated. Objects that are too thin may disappear entirely. This module is useful for removing noise from an image. Eroding an image by a large amount will remove all small objects and cause remaining larger ones to have smoother boundaries.

Interface



Instructions

1. Direction - Specify which direction the object should be eroded from.

All - erode the object from all sides

Horizontal/Left/Right - erode only in the horizontal direction

Vertical/Top/Bottom - erode only in the vertical direction

2. Kernal - Specify which kernal the dilation should use to perform the erosion. Using a square kernal will keep square shapes square but cause round shapes to become more square. Specifying the Disc shape will cause round shapes to stay round whilst square shapes will become rounded in the corners. In between these two shapes are various other shapes that cause different side effects during dilation. Chose one that best suits your needs.

3. Border - Select if you want the border to act as an object boundary and also be eroded. This is useful if you wish for objects to become separated from the image boundaries.

4. Avoid Removing Objects - Select if you want to keep at least one pixel erasian per object. This will ensure that regardless of erasian amount, objects that would be completely removed are still preserved with a couple pixels.

5. Removal Amount - Specify the number of times to perform erosion regardless of if objects are removed or not. This helps to remove noise surrounding objects that would be preserved due to the Avoid Removing Object selection.

6. Amount - Specify the number of times to perform erosion. Higher numbers will cause smaller objects to disappear (unless Avoid Removing checkbox is selected) but will disconnect larger objects.

Example

Source Image

Erode with count of 1

See Also

[Dilate](#)

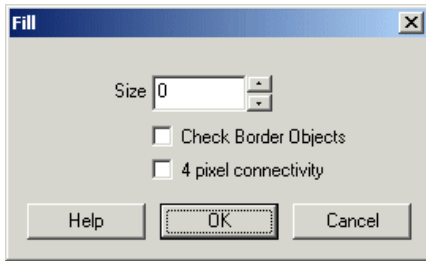
[Open](#)

[Close](#)

Fill

The Fill module fills any holes in the current image. You can specify a specific object size to fill.

Interface



Instructions

1. Size - Specify the size of the object that would be filled. Zero indicates that all holes detected should be filled.
2. Check Border Objects - Specify if holes that include the edge of the image should be filled too.
3. 4 pixel connectivity - By default RoboRealm uses 8 pixels around the current pixel to test if the pixel is connected to another pixel. Selecting 4 pixel connectivity restricts RoboRealm to only check the North, South, West and East pixels for connectivity.
4. Fill with average color - Fills objects holes based on the surrounding color of the hole instead of just using white. This is useful if your image has already been segmented down to a few colors and you would like to fill object holes with the same color as the object.

Example

Source Image



Fill with size of 0 (fill any non-border holes)



See Also

[Clean](#)

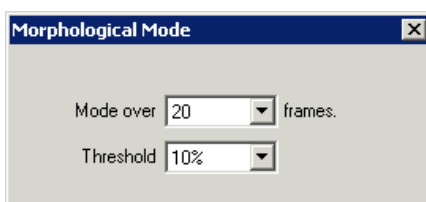
Flicker

The Flicker module is used to stabilize your image when in lower light situations. Similar to the [frame averaging module](#) the Flicker can filter out those values that are sporadic and mainly serve as image noise. In particular, when detecting the color blue in low light situations black objects can appear to have a blue hue to them. This can be seen when using the [RGBFilter](#) set to filter out blue in a dark environment. You will notice that many of the black areas will be detected as having some blue components that flicker violently within the image.

To stabilize the video image and identify solid blue areas (such as a blue ball) the Flicker module will monitor the past few frames and will set pixels as being 'on' only if the past frames have had the same pixel on (think of a temporal mode filter). In effect this module will filter out pixels that are changing very rapidly from frame to frame. This flickering of colors is exaggerated in lower light situations.

Note that this filter only operates on binary images. It considers any pixel value < 128 to be off and ≥ 128 to be on.

Interface



Help

OK

Cancel

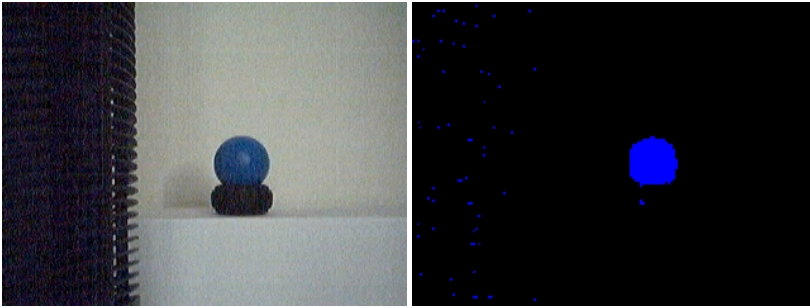
Instructions

1. Select how many frames you would like to monitor when trying to reduce the image flicker. Note that the more frames you select the larger the image updates will lag. This can be seen when moving a detected object across the video frame by the presence of a movement tail lagging behind the actual object.
2. Select the threshold that below which the pixel is considered off and above which the pixel is considered on. You can change this value to reduce the flicker of the image while keeping the image lag low but you will sacrifice parts of the object that are not completely stable.

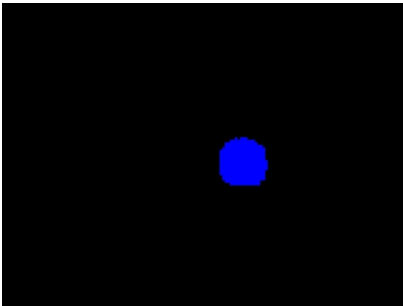
Example

Source

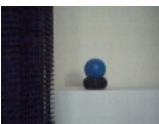
RGBFilter Blue with noise



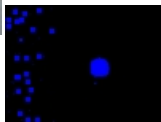
Flicker



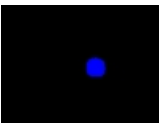
It is a little difficult to understand the filter from a static point of view. Below we have included 15 second clips from the video camera to better demonstrate the effect.



The original video without any filters. Notice the flickering of pixels even when encoded in MPEG. The dark area to the left is also flickering but since the intensity is very low you will not notice.



The image RGB filtered to blue. You can now easily see the issue with the image. The dark area on the left side has many small noise pixels that flicker violently. You can also notice the area just around the ball is also flickering and does not define the circular quality of the ball as best it could.



Finally, the processed image sequence using the Flicker filter. This video sequence is not particularly interesting except unless you see the previous video. This sequence just shows the blue ball with almost no flickering around the ball or even in the dark area to the left. We now see a nicely defined blue ball that can be easy to detect even in lower light conditions.

The Flicker module is a great example of how you can use multiple video frames to produce a better image than a single frame can provide. Note that similar (but not as stable) effects can be created using the frame averaging and frame integration techniques.

See Also

[Frame Averaging](#)
[Frame Integration](#)

HBreak

The HBreak module breaks H-connected pixels

```
1 1 1          1 1 1
0 1 0    becomes  0 0 0
1 1 1          1 1 1
```

Example

Source Image



H-Break Image



Note that the h-break is only 1 pixel big.

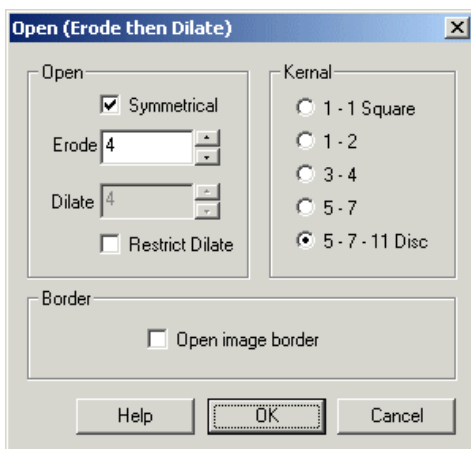
See Also

- [Open](#)
 - [Erode](#)
-

Open

The Open module performs an erosion (shrinks the current white image), followed by a dilation routine (grows the current white image). Erosion followed by dilation will open objects closed by a thin boundary.

Interface



Instructions

1. Erode - Specify the number of times to perform erosion before dilation. Higher numbers will open thicker objects.
2. Symmetrical - if you wish to erode and then dilate an unequal number of times (leaving the resulting blob larger or smaller than the original) unselect the "Symmetrical" checkbox. This will allow you to specify a different number of the Dilate Count.

3. Dilate - only active if module is not symmetrical meaning the image will be eroded and then dilated an unequal number of times.
4. Restrict Dilate - if using an asymmetrical opening you may want to restrict the dilation process to NOT become larger than the original image. Selecting "Restrict Dilate" will ensure that the dilation part of the processing does not dilate parts beyond that of the original blob.
5. Kernal - Specify which kernal the open operation should use. Using a square kernal will keep square shapes square but cause round shapes to become more square. Specifying the Disc shape will cause round shapes to stay round whilst square shapes will become rounded in the corners. In between these two shapes are various other shapes that cause different side effects during opening. Chose one that best suits your needs.
6. Border - Select if you want the border to act as an object boundary and also be opened.

Example

Source Image



Open with count of 1



See Also

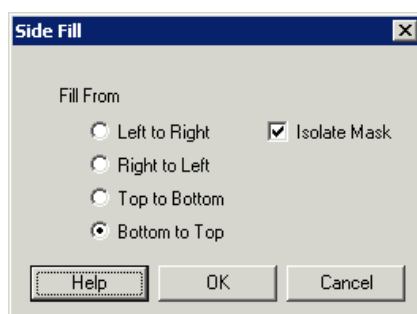
[Blob Split](#)
[Watershed](#)
[Close](#)
[Top Hat](#)
[Erode](#)

SideFill

The SideFill module fills the black area of an image starting from the specified side and proceeds until a non-black pixel is found. In the case of filling from the top side to the bottom of the image you can think of water drops starting from the top of the image and stopping when they encounter a non-black pixel. As the water fills the image the dark area on top becomes white.

This function is very useful to create a silhouette outline of an edge boundary used for skyline or ground plane detection.

Interface



Instructions

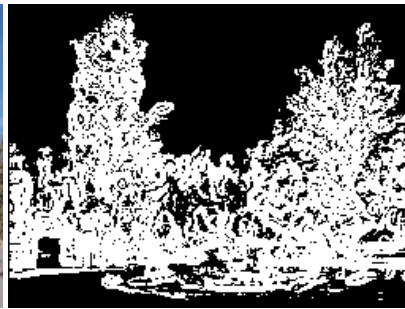
1. Direction - Specify which direction you would like to fill from. The most common for sky detection is "Top to Bottom". The most common for ground plane detection is "Bottom to Top".
2. Isolate Mask - To remove the existing image in order to show a detailed silhouette check the 'Isolate Mask' checkbox. This will remove the current image that is used to stop the filling process and leaves just the 'filler' visible.
3. Skip Border - At times a single pixel line will be created around the image as an artifact due to module processing. Selecting 'Skip Border' will ignore this single pixel border when filling the image.

Example

Source



Thresholded Edge



SideFill Top to Bottom



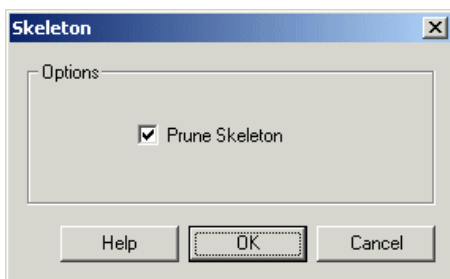
See Also

[Fill](#)
[Open](#)

Skeleton

The Skeleton module extracts the skeleton of an image. The skeleton of an image is the medial axis or the thinnest representation of the current image. The skeleton can be used as a basic shape analysis tool.

Interface



Instructions

1. Prune - select if you want the created skeleton to be pruned. Pruning will remove those parts of the skeleton which are created due to thick objects.

Example

Source Image



Skeleton Image



Source Image



Skeleton Image



Pruned Skeleton Image



Note that the skeleton of an image is very sensitive to image noise. You may need to erode by 2 before creating an image skeleton to help remove any noise.

See Also

[Border](#)

[Outline](#)

[Sobel](#)

Spur

The Spur module removes spur pixels.

```
0 0 0          0 0 0
0 1 0    becomes  0 0 0
1 0 0          1 0 0
```

Example

Source Image



Skeleton Image



See Also

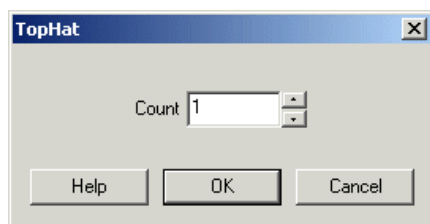
[Erode](#)

[HBreak](#)

Top Hat

The Top Hat module performs an erosion (shrinks the current white image), followed by a dilation routine (grows the current white image) and then subtracts the original image. Erosion followed by dilation will open closed objects. Subtracting the original will result in the display of just the pixels that close the object.

Interface



Instructions

1. Specify the number of times to perform erosion before dilation. Higher numbers will open thicker connected objects which will result in larger connections being displayed in the final image.

Example

Source Image



Top Hat with count of 1



Top Hat results shown in white. Original image in red.

See Also

[BottomHat](#)

[Open](#)

[Close](#)

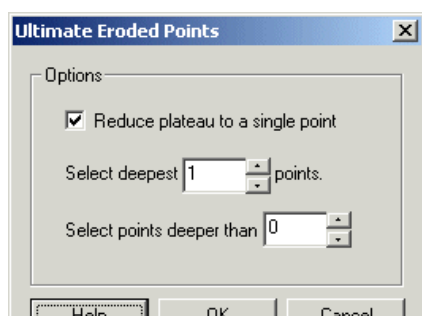
Ultimate Points

The Ultimate Points module extracts the last points that would be removed if the object were eroded to completion. They represent the seeds of an object.

Note that for objects of uniform shape the ultimate points module can approximate the skeleton module.

This module can be helpful in determining where free space is located. For example, if you have several objects in an overhead view and you want to find the location in the image most distant from all other objects taking the UEP of the images negative (i.e. make the black pixels white and then find the ultimate points) you can determine the point most distant from all other detected objects.

Interface

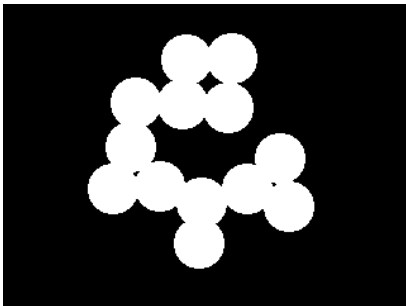


Instructions

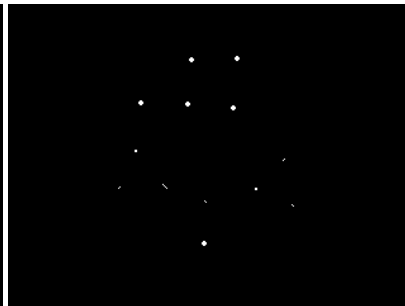
1. Reduce Plateau - Often when eroding an object down to its ultimate points the last level to be eroded to completion is actually not a point but several points generating a small plateau that will be removed on the next erosion cycle. Selecting the Reduce Plateau checkbox will reduce these plateaus to a single point in the center of the plateau to ensure that only a single point is recorded.
2. Select deepest - Specify how many of the deepest eroded points you want to keep. Often when calculating the Ultimate Eroded Points a lot of shallow points are generated by small objects due to noise. Selecting only the deepest points will ensure that only those larger blob's UEP's are recorded.
3. Select points deeper than - Once again to ensure that smaller eroded blobs do not contribute towards the final results you can threshold the selection to ensure only large blobs (deeper UEPs) will be recorded.

Example

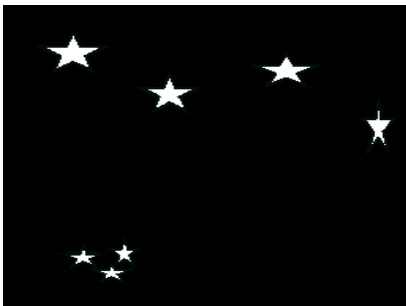
Source



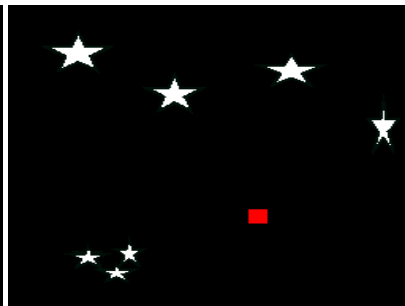
Ultimate Points



Source



Most distant point from stars



Variables

ULTIMATE_POINTS - a 3 by N array containing the X,Y and depth of each calculated Ultimate Point.

See Also

[Erode](#)
[Skeleton](#)

Watershed

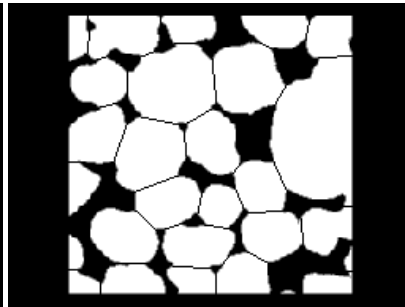
The Watershed module will split objects within a binary image into individual objects based on how they are connected to each other. This module allows for separation of connected blobs so that they can be processed individually. Objects are split based on a circular compactness routine that will attempt to create objects that are as compact as possible.

Example

Source



Watershed



See Also

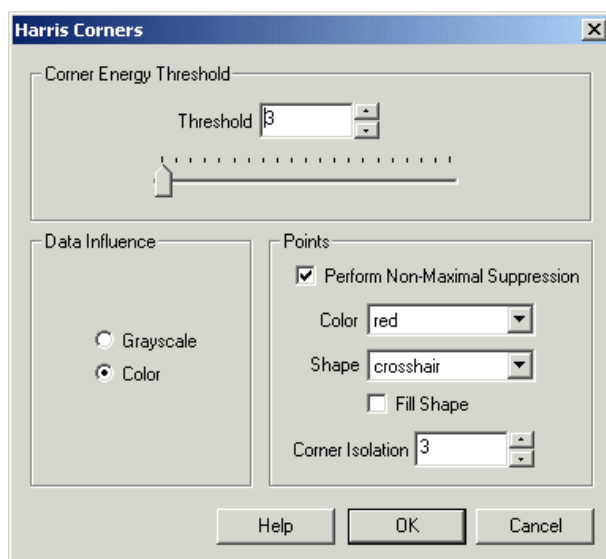
[Blob_Split](#)
[Open](#)

Harris Corners

The Harris Corners module identifies corners present within the image using the Harris Corner detection technique. The technique first identifies vertical and horizontal edges using a Sobel type edge detector. Those edges are then blurred to reduce the effect of any image noise. The resulting edges are then combined together to form an energy map that contains peaks and valleys. The peaks indicate the presence of a corner.

Note that a maximum of 15,000 points will be detected. If your image is very large and requires an even spread across the image increase the threshold to reduce the total number of detected points.

Interface



Instructions

1. Threshold - Select the threshold above which a corner is considered present. Lower values allow more subtle corners to appear whereas higher values reduce the number of corners identified.
2. Data Influence - Select if you want the algorithm to consider color or not. Color can help identify corners between colors but adds additional processing.
3. Fast Harris - While the traditional Harris corner detector can work well it can also be slow for some applications. If speed is a concern of accuracy select the Fast Harris checkbox. This will enable the fast Harris corner detector which optimizes the Harris detector by approximating some steps to achieve additional speed.
4. Perform Non-Maximal Suppression - Unchecking the Non-Maximal Suppression will display the normalized Harris corner energy map instead of the detected corners.
5. Color - Select a color to display the detected corner points ontop of the current image.

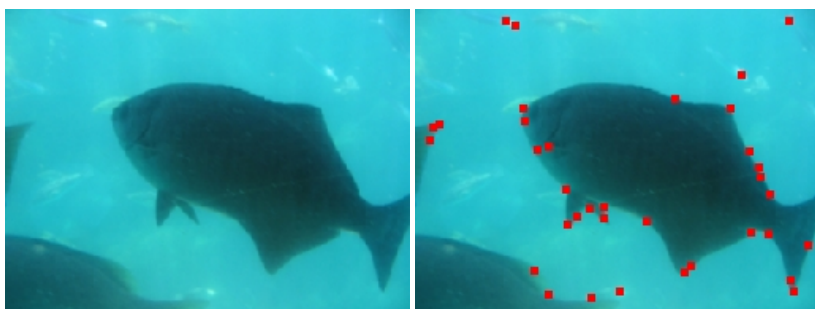
6. Shape - Select the shape to use when displaying the detected corner points.

7. Corner Isolation - to eliminate points close to each other select an appropriate corner isolation number. This will ensure that points are at least X pixels from each other and prevents tight clustering of detected points.

Example

Source

Detected Corners



Variables

HARRIS_CORNERS - Each corner XY coordinate is saved into an array accessible by the HARRIS_CORNERS variable. Using code similar to the following within a VBScript module will allow you to further process the corner points.

```
corners = GetArrayVariable("HARRIS_CORNERS")
```

```
' write back to messages the first line ..
```

```
' note that line segments are defined by two endpoints
```

```
Write("First Coord: " & corners(0) & "," & corners(1) & vbCRLF)
```

```
Write("Second Coord: " & corners(2) & "," & corners(3))
```

See Also

[Moravec Interest Operator](#)

[Sobel](#)

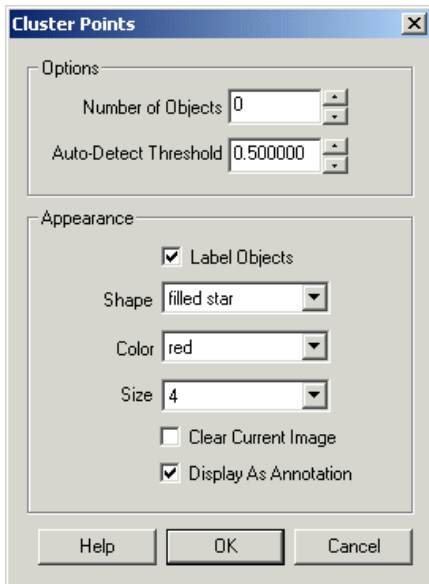
Cluster Points

The cluster points module provides a way to group pixels into larger objects that may not be connected or might not be separated correctly. The clustering of points into larger objects is somewhat of a subjective choice as given a set of pixels different people may make different choices in what constitutes a larger object. Nevertheless, automating that subjective process is essential when working with pixels that need to be grouped into meaningful objects that can be processed from a higher point of view. Note that this module expects a binary (black/white) image.

The clustering algorithm used is a variant of the k-means point clustering. The variations allow for increased speed and an auto-detect "number of objects" feature.

It is recommended to first try a combination of morphological techniques like [Erosion](#), [Dilation](#), [Open](#) or [Close](#) modules to join objects into distinct blobs for further processing.

Interface



Instructions

1. Number of Objects - If you know the number of objects that need to be grouped from the current pixels specify it in the Number of Objects textbox. If you do not know how many objects are to be detected enter in a zero (0) and set the threshold to a value between 0 and 1 that divides the current image into reasonable groups.
2. Auto-Detect Threshold - In deciding on how many objects are to be created from the current image the threshold is used to terminate the addition of more objects.
3. Label Objects - If you want to see which pixels are associated together select the "label objects" checkbox. This will colorize pixels that belong to the resulting object. Note that pixels that are connected directly to other pixels can belong to different objects. This feature is what makes the clustering module similar to erosion and dilation techniques in order to split connected objects.
4. Shape, Color, Size - Specify the shape, color and size of the graphic that is used to indicate the Center of Gravity point of the new clustered object.
5. Clear current image - Select to clear the current image and draw the graphics on a black image.
6. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

Example

Source



Clustered (Labeled) Points to 3 Objects



Variables

CLUSTER_POINTS - the resulting center of gravity of detected clustered objects.

See Also

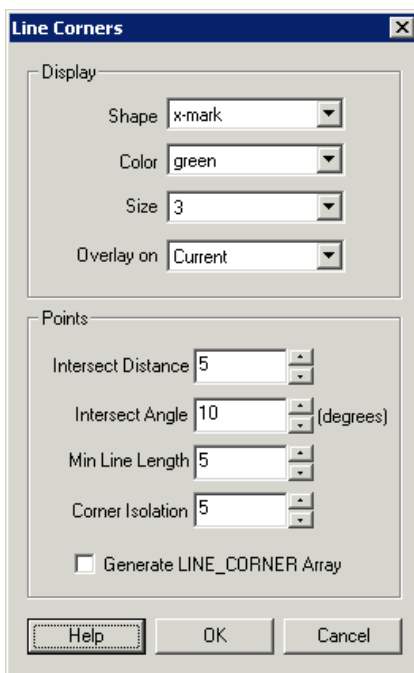
[Erode](#)
[Dilate](#)
[Open](#)
[Close](#)

Line Corners

The Line Corner module is a curvature corner detector. The module works by identifying straight lines within the image and then plots points at the intersections of these lines. The benefit to using intersecting lines to generate interest points is that they are typically quite stable and can exist in areas of implied corners (i.e. where soft corners exist).

The Line Corner module expects an edge extracted image to work correctly. Thus you must use Canny or other edge finding modules before running the Line Corner module.

Interface



Instructions

1. Be sure to have an edge detected image (using something like [Canny](#)) before using the Line Corner detection module.
2. Intersect Distance - how far from the end of the line should an intersection be considered. The larger this number the further away the detected intersections will appear from the originating lines. The smaller the number the closer the line intersections will need to be in order to trigger a point display.
3. Intersect Angle - the minimum angle the two lines need to make at the intersection for the point to be considered. See the last example image below that filters all but about 90 degree line intersections.
4. Min Line Length - the minimum line length that should be considered for intersection. As shorter lines have less defined slope they should

normally be eliminated otherwise they may cause spurious line intersections.

5. Corner Isolation - many points may be detected close to each other (especially in a noisy image). Increasing this number will ensure that detected corners are far enough away from other corners to avoid creating a pack of corners.

6. Generate LINE_CORNER Array - creates a RoboRealm variable that contains the x,y coordinates of each detected corner point. Note that the array is even size with LINE_CORNER(0) being the X coordinate and LINE_CORNER(1) being the Y coordinate. LINE_CORNER(2) would then be the X coordinate of the next point.

7. Display Shape, Color, Size - the shape, color, and size of the graphic corner indicators

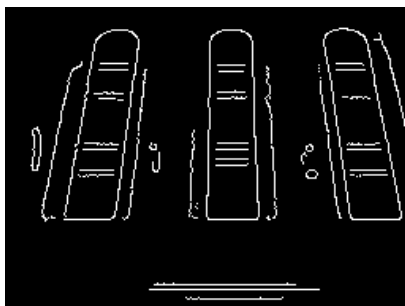
8. Overlay On - as you will be working on a edge detected image you can select the final image on which to overlay the graphics in order to better understand where the corners are being detected.

Example

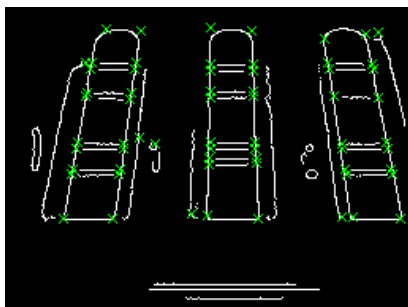
Source



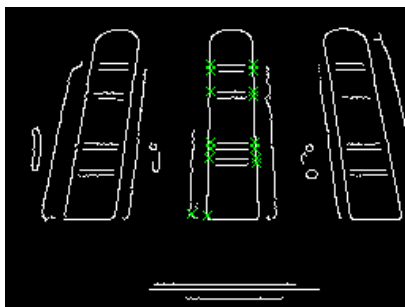
Canny Edge



Line Corners



Intersect Angle at 87.5



Notice the middle lines of the windows in the Canny image. Those lines are not connected to the edges of the windows (due to bad edge detection) but the Line Corner module recognizes these as corners due to the line intersection between those horizontal middle lines and the outer vertical lines. (Assuming the Intersect Distance allows an intersection at a length > 10).

Variables

LINE_CORNER - contains x,y coordinates of detected corners

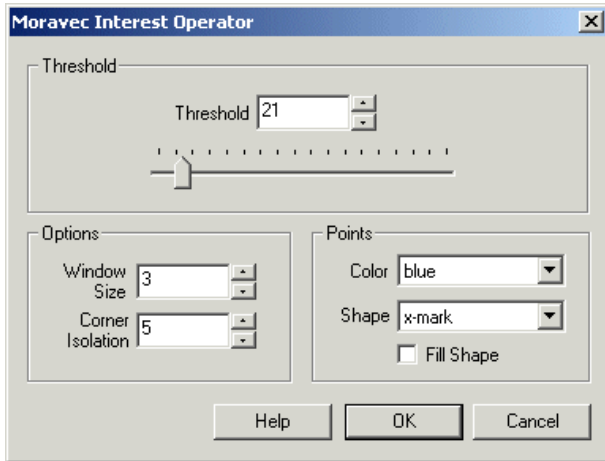
See Also

[Straight Lines](#)

Moravec Interest Operator

The Moravec Interest Operator module identifies "interesting" points within the image using the Moravec Interest Operator detection technique. These points normally identify corners of objects as corners are good points to detect and track. The technique first calculates an edge map based on horizontal, vertical and both diagonal directions. From this edge map the maximum response of all these edges is recorded. Once all maximal edge responses have been calculated the point that is maximal with respect to its neighbors is identified as "interesting" and marked.

Interface



Instructions

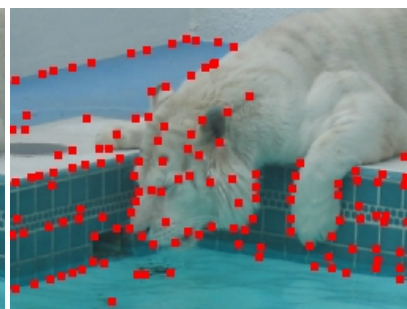
1. Threshold - Select the threshold above which a point is considered interesting. Lower values allow less anchored points (less repeatability) to appear whereas higher values reduce the number of points identified but improve the repeatability (i.e. identifying the same point in successive images).
2. Window Size - Select the window size for the point detection. This will almost always be 3.
3. Corner Isolation - To eliminate points close to each other select an appropriate corner isolation number. This will ensure that points are at least X pixels from each other and prevents tight clustering of detected points.
4. Color - Select a color to display the detected corner points ontop of the current image.
5. Shape - Select the shape to use when displaying the detected corner points.

Example

Source



Detected Points (threshold=10)



Variables

MORAVEC_CORNERS - Each corner XY coordinate is saved into an array accessible by the MORAVEC_CORNERS variable. Using code similar to the following within a VBScript module will allow you to further process the corner points.

```
corners = GetArrayVariable("MORAVEC_CORNERS")
```

```
' write back to messages the first line ..
```

```
' note that line segments are defined by two endpoints
```

```
Write("First Coord: " & corners(0) & ". " & corners(1) & vbCRLF)
```


Write("Second Coord: " & corners(2) & "," & corners(3))

See Also

[Harris Corners](#)

[Sobel](#)

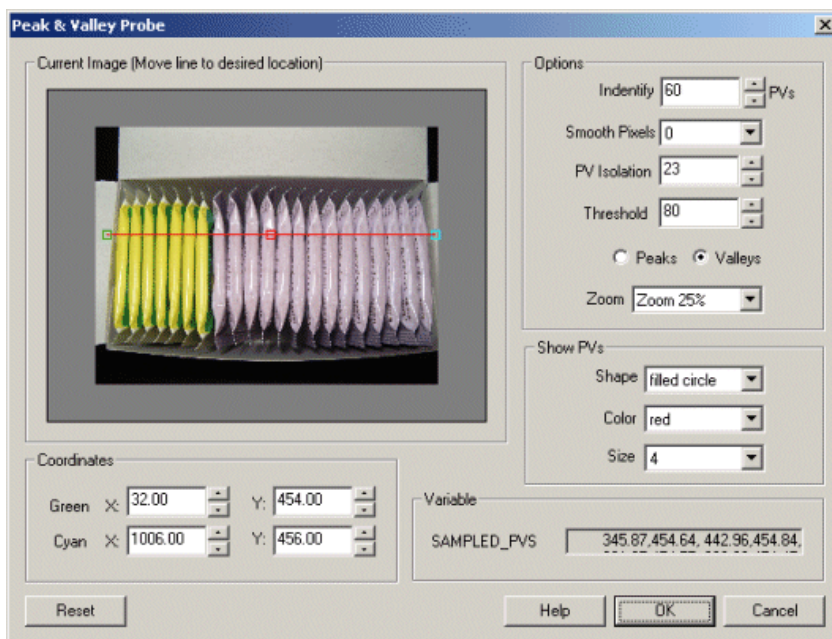
Peak & Valley Probe



The Peak & Valley Probe provide a way to locate peaks (maximum points) and valleys (minimum points) along a specified line in an image. This is similar to the [Edge Probe](#) but instead of detecting large changes in intensity this module will detect the change in direction of intensity. This is useful in detecting patterns, frequencies or recurring objects.

To better visualize these peaks and valleys you can use the [Line Profile](#) module that displays the pixel intensities along a line. If you see a nice peak and valley curve then it is a good candidate for the Peak & Valley Probe module.

Interface



Instructions

1. Coordinates - Specify the line location to probe for peaks or valleys. You can either drag the squares in the Current Image display or specify the numbers in the provided text boxes. Keep in mind that the [\[expression\]](#) notation also applies in the text box.

Use CTRL-click to move the entire probe to a different location. Use SHIFT-drag to create the probe of a certain size in the clicked position. When zoomed in, drag the image around to view different parts of the image or expand the dialog window to view more of the image.

2. Use Origin - You can specify that the current coordinates are relative to the Origin Variables created by the [Origin Probe](#) Module (or by setting these variables yourself). This allows the specified coordinates to move relative to the detected origin in case what you are sampling is not always in the same absolute image location. When you select this checkbox the current origin values are subtracted from the currently specified coordinates to create a relative position. If you have not yet set the origin, you can come back later and adjust the coordinates as appropriate.

3. Options Identity - Specify the maximum number of peaks or probes you want to detect.

4. Options Smooth - If you are detecting too many peaks or valleys try smoothing the pixels. This will reduce intensity noise and produce better results.

5. Options Isolation - To avoid detecting too many points close to each other increase the isolation count. This will force only the best detection per X pixels to be recorded.

6. Options Abs Threshold - Select how high a peak needs to be before it could be qualified as a peak OR how low a valley needs to be before it could be qualified as a valley.

7. Options Relative Threshold - Select how low a valley must exist between two peaks or how high a peak must exist between two valleys. This

helps to reduce local dips or bumps that might trigger an incorrect peak or valley detection.

8. Select if you are detecting a peak or valley.

9. Zoom - Select the zoom for the Current Image in case it is too large or small for the provided viewing area.

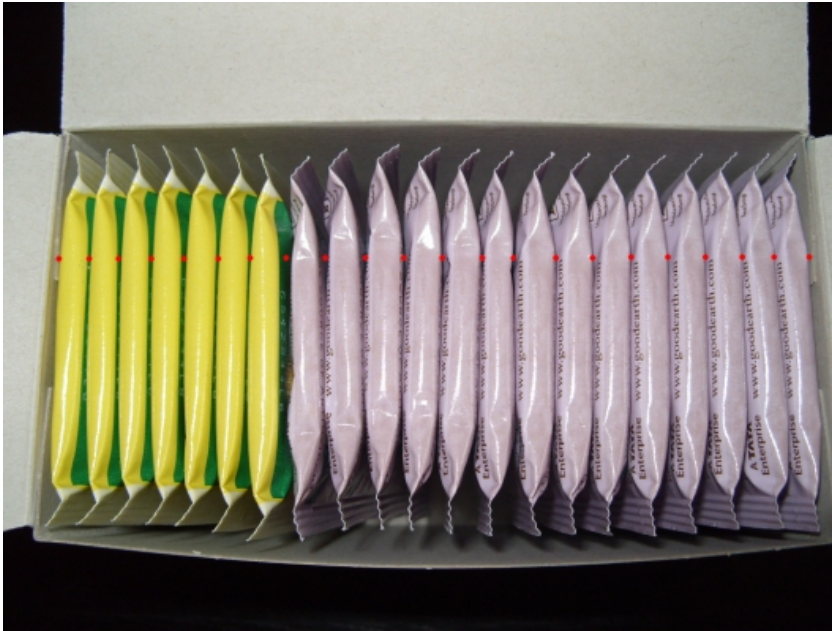
10. Show PVs - to visually see which points are detected (in the main RoboRealm interface) select the appropriate Color and Thickness of the point that will indicate where in the image the peak or valley has been detected.


11. Include Intensity - If you want to know the intensity value in addition to the peak or valley coordinates select the Include Intensity checkbox. This will add the intensity value after the X and Y coordinates in the recorded array PROBED_PEAK_VALLEY.

Example

Red spots are used to detect the valleys inbetween the tea bags. The total count-1 is the number of tea bags.

Peak & Valley Probe



 [Click here](#) to download this example and run in RoboRealm.

Variables

PROBED_PEAK_VALLEY_COUNT - the number of peaks or valleys found

PROBED_PEAK_VALLEY - an x,y array of the location of the detected peaks
or valleys.

OR

PROBED_PEAK_VALLEY - an x,y,intensity array of the location of the detected peaks
or valleys along with the intensity value depending on the setting of
Include Intensity checkbox.

See Also

[Origin Probe](#)

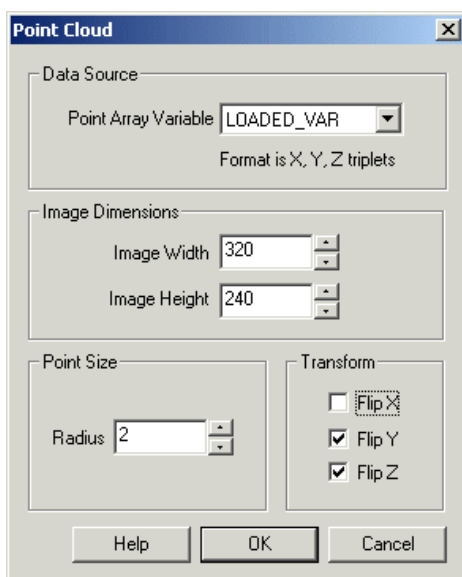
Point Cloud

The Point Cloud module is used to create a solid structured based on a cloud of points. This is normally needed after sampling certain images devices such as LIDAR type systems whose end result is a set of sampled points at non-discrete intervals. The Point Cloud module is one way of joining these sparse points into a uniform solid in order to provide additional processing.

Once these points are accessed they are normalized to fit into an image with the specified dimensions. Thus even if your point x range is from -100 to -50 the resulting image will represent that range but with x coordinates translated to 0 to 320 assuming that you specified an image width of 320.

The resulting graphic image is created by normalizing the data points and drawing an averaged circle at each point in the new image. The size of the circle (seen below as radius) is relevant in that you want the size to be large enough to overlap with nearby points but not too large as to obscure the object boundaries. It is best to interactively play with the radius to see which size suits your needs given your desired image dimensions.

Interface

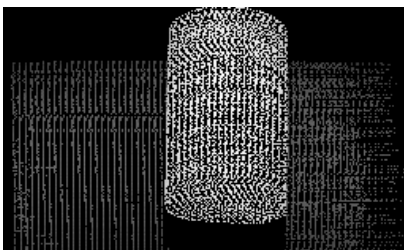


Instructions

1. Point Array Variable - Specify which RoboRealm variable contains the x,y,z point coordinates to be draw as an image. Note that this data triplet is contained in a SINGLE variable as apposed to three separate variables. This variable can be created from any other module within RoboRealm including the [Read Variables](#), [VBScript](#) or specified through the [API](#).
2. Image Width - Specify the width of the resulting image you wish to create.
3. Image Height - Specify the height of the resulting image you wish to create.
4. Radius - Specify the size of the circle drawn at each point. Overlapping circles are averaged to provide a smooth point to point transition.
5. Transform - As point coordinates are not always in the right orientation in the data file you can select a combination of coordinate flips that will help to orient the final image. Note that flipping Z is actually inverting the intensity at that point. If your image appears too dark or too light try flipping the Z.

Example

Points with radius 0



Points with radius 2





Produced from a text file with 21,985 points in the following format:

```
22 159 500
24 164 514
26 162 508
39 159 500
52 162 508
...
```

Note that the Z or intensity is above the normal 255 image intensity limit but as this number is also normalized the resulting image is acceptable.

Once represented as a graphic image all the other RoboRealm modules can now be used to segment or quantify objects as would be the case with webcam images.

See Also

[Read Variables](#)

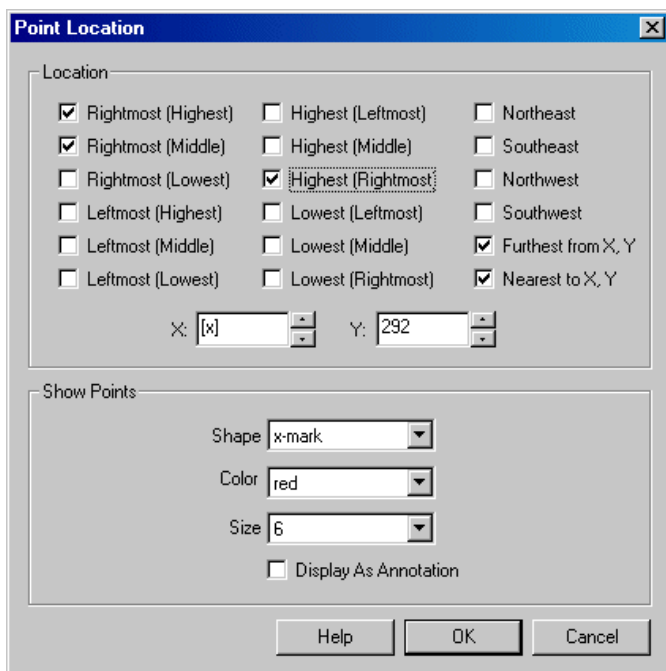
Point Location

The Point Location module provides a quick way to identify specific coordinates within the image based on their location. For example, you may want to know the position of the highest point of any non-black pixel within the image to identify a maximum peak. Note that this module operates on a binary image.

The module is often used after numerous morphological operations that have thresholded the image and changed the image shape.

Note that the north, south, east, west points are calculated relative to the object's Center of Gravity point.

Interface



Instructions

1. Location - Select each point that you would like to indicate with the image. Note that for each point a variable with a similar name is created with the X and Y coordinates of the point.

2. X,Y - Once Nearest X,Y and/or Furthest X,Y is selected the X and Y coordinate text boxes will be enabled. You can specify the X and Y coordinates of the point that should be used when comparing for furthest and nearest object points to the specified point. Keep in mind that the [\[expression\]](#) notation also applies in the text boxes.

3. Shape - The shape of the marker that indicates where a point has been sampled.

4. Color - The color of the marker that indicates where a point has been sampled.

5. Size - The size of the marker.

Example

Leftmost, Rightmost and highest points indicated in red X



Variables

HIGHEST_LEFTMOST_X

HIGHEST_LEFTMOST_Y

HIGHEST_MIDDLE_X

HIGHEST_MIDDLE_Y

HIGHEST_RIGHTMOST_X

HIGHEST_RIGHTMOST_Y

LEFTMOST_HIGHEST_X

LEFTMOST_HIGHEST_Y

LEFTMOST_MIDDLE_X

LEFTMOST_MIDDLE_Y

LEFTMOST_LOWEST_X

LEFTMOST_LOWEST_Y

LOWEST_LEFTMOST_X

LOWEST_LEFTMOST_Y

LOWEST_MIDDLE_X

LOWEST_MIDDLE_Y

LOWEST_RIGHTMOST_X

LOWEST_RIGHTMOST_Y

RIGHTMOST_HIGHEST_X

RIGHTMOST_HIGHEST_Y

RIGHTMOST_MIDDLE_X

RIGHTMOST_MIDDLE_Y

RIGHTMOST_LOWEST_X

RIGHTMOST_LOWEST_Y

NORTHEAST_X

NORTHEAST_Y

NORTHWEST_X

NORTHWEST_Y

SOUTHEAST_X

SOUTHEAST_Y

SOUTHWEST_X

SOUTHWEST_Y

FURTHEST_X

FURTHEST_Y

NEAREST_X

NEAREST_Y

See Also

[Sample Line](#)

[Sample Curve](#)

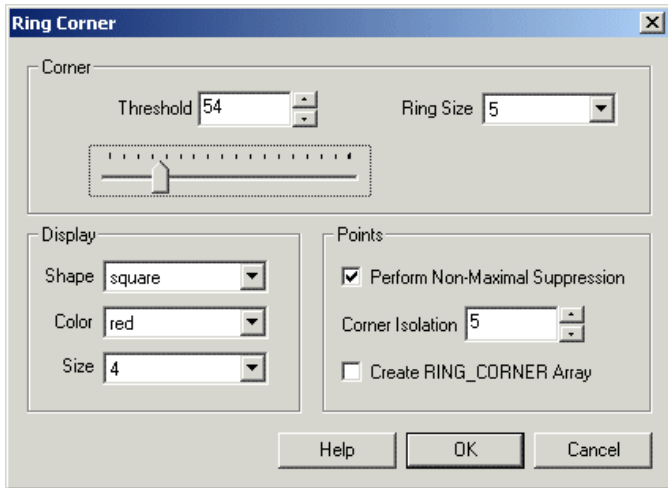
[Ultimate Points](#)

Ring Corner

The Ring Corner module is a point feature module that identifies corners based on how the surrounding pixels compare with the center pixel. This is functionally similar to [Edward Rosten's FAST](#) corner detector which has been known to be one of the fastest corner detectors to date.

The corner detector provides a way to identify points within an image that are hopefully repeatable in successive frames. As corners are scale, translation and rotationally invariant they provide valuable keypoints that can be used to compare one image with another

Interface



Instructions

1. Corner Threshold - defines what range of pixel values are considered equal. For example, a pixel at value 100 and 130 will be considered equal if the threshold is set at 40 but different if set at 20.
2. Ring Size - the filter width used to analyze the surrounding pixels with regards to the center pixel.
3. Display Shape, Color, Size - configures the point indicator graphics.
4. Perform Non-Maximal Suppression - corner points are often identified in clumps of pixels. The non-maximal suppression will "suppress" or remove those pixels that are not the maximum value in a set filter ring size.
5. Corner Isolation - despite non-maximal suppression additional space between points is sometimes needed. The corner isolation number will ensure that a point is isolated within the specified number of pixels.
6. Create RING_CORNER Array - creates a VBScript accessible array of the point coordinates identified by the ring corner module.

Example

Source



Ring Corners Identified



Variables

RING_CORNER - x_coord, y_coord array list of identified points

See Also

[Harris](#)

Sample Curve

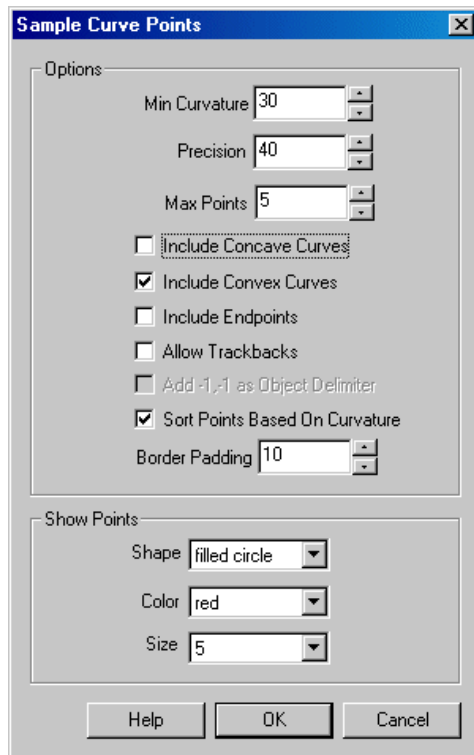


The Sample Curve module expects a binary (black and white image) with an object whose outline has curves that need to be quantified in some way. This module is similar to the [Sample Line](#) module but instead of sampling a line based on distance the Sample Curve module does so based on the amount of curvature and edge exhibits at a specific pixel. In this way you can extract out those points in the outline of the object that are high curvature points.

For example, the points at the tips of a hand are very high in curvature and can be readily extracted using this module.

Extracted points are placed into a `SAMPLE_CURVE_POINTS` array.

Interface



Instructions

1. **Min Curvature** - Specifies the minimum curvature that will be sampled. Straight lines will have 0 curvature while a sharp wedge will have close to 180. Note that the curvature values are specified as degrees.
2. **Precision** - To define the curvature of each point on a edge some pixels before and after the point need to be analyzed to determine the point's curvature. The Precision value defines how many neighboring pixels are used to define the points curvature. Smaller values will detect smaller more finer curves, whereas larger values will detect larger curves.
3. **Max Points** - Specify how many points you want to sample. If the maximum points to be sampled is smaller than the actual sampled points the points with the highest curvatures will be selected.
4. **Include Concave Curves** - Select if you only want to sample concave points.
5. **Include Convex Curves** - Select if you only want to sample convex points.
6. **Include Endpoints** - When working with single pixel lines it is often required to also have the line endpoints included in the sample regardless of their actual curvature.
7. **Allow Trackbacks** - Single pixel lines will track back on themselves (think of the outline of a line with 3 endpoints). Selecting "Allow Trackbacks" will include the trackback points that were traversed to get to the next endpoint regardless of if the outline points have already been checked.
8. **Add -1,-1 as Object Delimiter** - if you have more than one object in the current image a -1,-1 will be added into the sample point array to indicate that a new object is about to be specified. If you are just using the points in other modules you may want this unchecked to remove the -1,-1 coordinates from being specified in the `SAMPLE_CURVE_POINTS` array.

9. Sort Points Based on Curvature - The sampled points are order as they are encountered. Select this checkbox if you would prefer to have the points order from smallest to largest curvature.

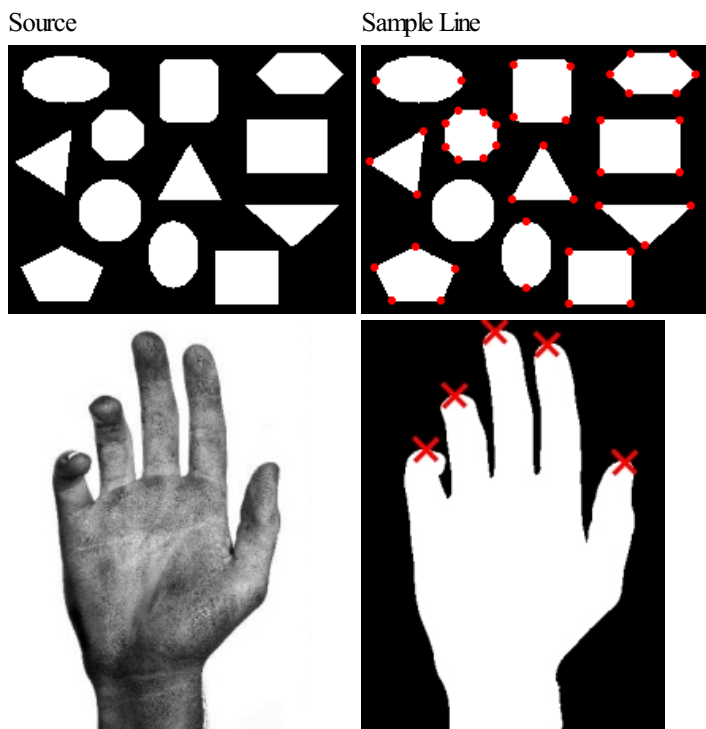
10. Border Padding - Specify the border amount that will NOT be included in the sample curve points array. This can be used to prevent sampling along image borders that may create invalid curvature due to image clipping. Any point that within X pixels close to the border will not be considered.

11. Shape - The shape of the marker that indicates a point has been sampled.

12. Color - To see what points are being sampled you can select the color of the marker that indicates a point has been sampled.

13. Size - The size of the marker.

Example



[Click Here](#) to load a robofile configuration that will generate the above image. The included image shows a hand against a white background. In this case we just use intensity thresholding to segment the hand from the background. For color images using the [RGB Filter](#) may be of better use. After some processing to remove smaller unwanted pixels the Sample Curve module is used to identify the fingertips of the hand as they are typically high curvature points when compared to the rest of the hand. Note that only convex points are sampled to eliminate those points that would be detected in between the finders which are also of high curvature.

Variables

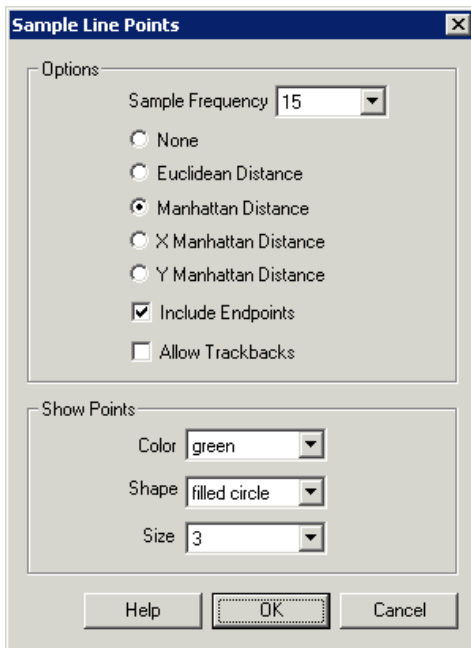
`SAMPLE_CURVE_POINTS` - the array that is created based on the points being sampled. The array format is `x,y,curvature` (3 numbers repeating for each point) that indicate the points `x` and `y` coordinates and the amount of curvature detected at that point. Curvature units are `degrees*100`. So a curvature value of `12456` is actually `124.56` degrees. Note that the end of a line is marked with a triple `-1,-1,-1`. This indicates that one object has completed and another is about to start and depends on the `-1,-1` checkbox (see above) if these will be included.

See Also

Sample Line Points

The Sample Line Points module will detect one or more lines within the current image and create an array of points along that line. This module can be used to generate an array of points to be used by other modules or saved for further processing. The included example shows how to generate a list of points from a laser scan of a 3D object.

Interface



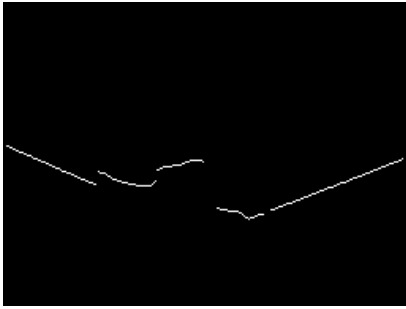
Instructions

1. Sample Frequency - Specify the sampling frequency. The frequency is the spacing between point samples. A smaller frequency will create many more tightly packed points than a larger frequency.
2. Select the sample technique.
 - None - no sampling is done. Useful when just the endpoints are desired
 - Euclidean - if the frequency is x then every point x pixels from the last one is sampled using the Euclidean $\sqrt{(x*x)+(y*y)}$ distance formula.
 - Manhattan - if the frequency is x then every point x vertical or horizontal pixels from the last one is sampled.
 - X Manhattan - only the horizontal pixel distance is checked against the frequency. This will produce points that are multiples of the frequency on the X axis but not on the Y.
 - Y Manhattan - only the vertical pixel distance is checked against the frequency. This will produce points that are multiples of the frequency on the Y axis but not on the X.
3. Include Endpoints - select if you wish to include the line endpoints in the final point array. Including endpoints will ignore the sample frequency settings to ensure that the endpoints are included.
4. Allow Trackbacks - often lines will track back on themselves (think of a line with 3 endpoints). Selecting "Allow Trackbacks" will include the trackback points that were traversed to get to the next endpoint regardless of if the values have already been included in the array.
5. Add -1,-1 as Object Delimiter - if you have more than one object in the current image a -1,-1 will be added into the sample line array to indicate that a new object is about to be specified. If you are just using the points in other modules you may want this unchecked to remove the -1,-1 coordinates from being specified in the SAMPLE_LINE_POINTS array.
6. Border Padding - Specify the border amount that will NOT be included in the sample line points. This can be used to prevent sampling along image borders.
7. Color - to see what points are being sampled you can select the color of the marker that indicates a point has been sampled.
8. Shape - the shape of the marker that indicates a point has been sampled.

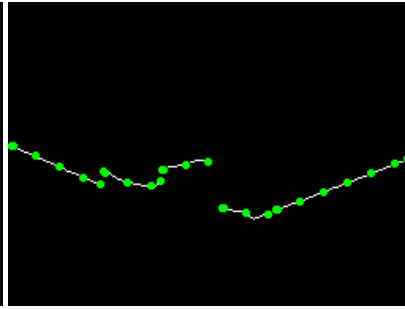
9. Size - the size of the marker

Example

Source



Sample Line



[Click here](#) to load a robofile configuration that will generate the above image. The included image shows a laser scan of a 3D object and how the scan is processed to refine the edge in order to be sampled correctly.

Variables

`SAMPLE_LINE_POINTS` - the array that is created based on the points being sampled. The array format is `x,y` (2 numbers) that indicate the points `x` and `y` coordinates. Note that the end of a line is marked with a double `-1,-1`. This indicates that one line has completed and another is about to start and depends on the `-1,-1` checkbox (see above) if this will occur.

See Also

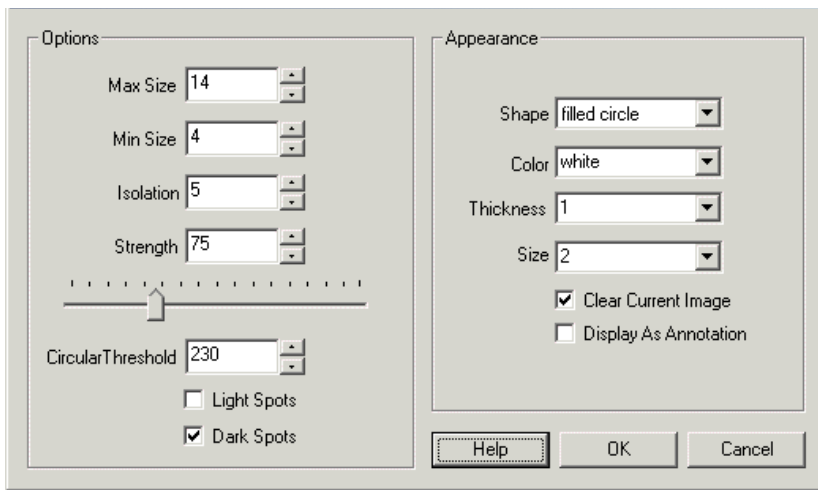
[Sample Curve](#)
[Point Location](#)

Spot Detector



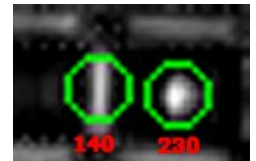
The Spot Detector module provides several configuration options for detecting spots/dots within an image. Unlike the corner detector modules, the Spot Detector module will identify the middle of a spot instead of its edges. The provided parameters allow you to specify what types of spots to identify as well as how large and strong the spots need to be with respect to the background in order to be identified.

Interface



Instructions

1. Max Size - Specifies the largest spot size that will be detected (its diameter)
2. Min Size - Specifies the smallest spot size that will be detected (lowest is 3). Together with the Max size this specifies the search size range that will be analyzed. Decreasing this size range will increase performance of this module.
3. Isolation - Defines how closely spots can be. If two spots are closer than the number of pixels specified in the isolation parameter the less stronger spot will be removed.
4. Strength - Defines how strong the spot needs to be in order to be identified. This is loosely related to the contrast of the depression (black dot) or elevation (white dot) and its surrounding area. The higher this contrast the more 'strength' a dot has.
5. Circular Threshold - Because many lines may have slight bumps in them that can be strongly identified as dots, the circular threshold parameter defines how much variation the circular border of the dot can be before being identified. This ensures that a dot is truly disconnected from other parts of the image. Note the image on the right that shows 2 potential spots with differing circularity threshold. Setting the threshold to 200 would eliminate the line segment.
6. Light Spots/Dark Spots - Select which type of dot you want to detect.
7. Appearance - Select how you'd like the results to be displayed.
8. Display As Annotation - Select if you want the results to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

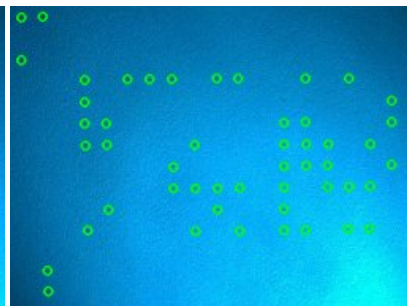


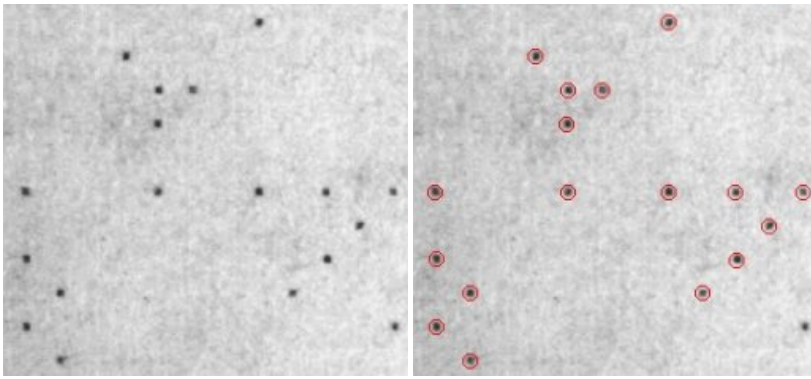
Examples

Source



Spot Detector





Note that dots near image edges may not be detected due to max window size.

Variables

SPOT_POINT_COUNT - number of detected spots

SPOT_POINTS - array of x, y coordinates of each spot

See Also

[Laser Spot](#)

[Harris Corners](#)

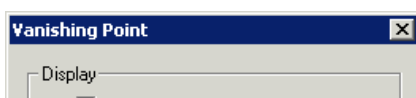
Vanishing Point

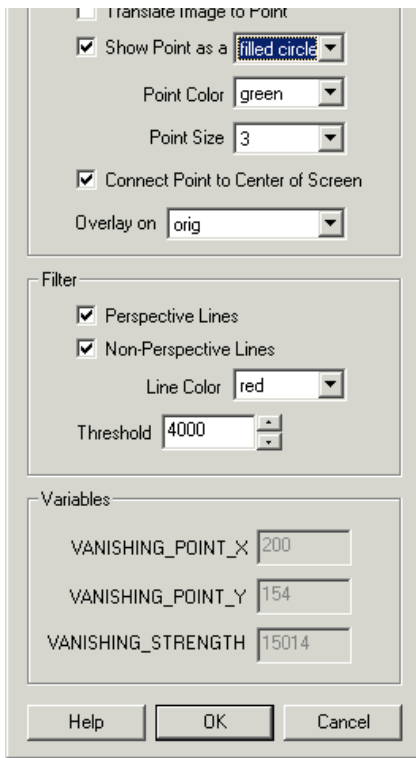
The Vanishing Point module determines where in the current image the vanishing point is. Vanishing points are the points that lines are drawn towards in order to induce perspective. Using lines within the image we can reverse the process of perspective to understand where the vanishing point is.

The value of knowing where the vanishing point is in understanding more about the scene that the robot is faced with. The vanishing point is also quite stable in a scene and can be used as a target similar to the [Visual Anchor](#) module but without any need for a "target" image. Vanishing points are an artifact of depth. Any image that has sufficient depth present will indicate perspective lines that can be used to identify the vanishing point.

If you have a hallway that the robot needs to move down simply track the vanishing point and steer the robot towards that point. It will work with most indoor hallways.

Interface



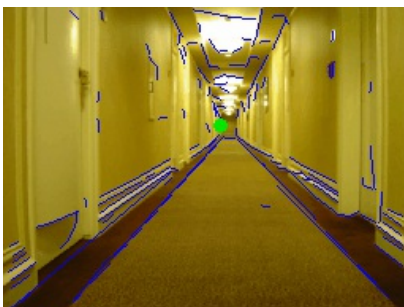


Instructions

1. Edge - Ensure that an edge detected image is created before running the vanishing point image. You can use [Canny](#) to generate the edges first.
2. Translate Image - will translate the current image such that the vanishing point is in the center of the image. This is a kind of image stabilization but only in the X axis.
3. Show Point - shows the vanishing point in a particular shape, color and size. Note that sometimes the vanishing point may not even be in the image. (see example below)
4. Connect Point to Center of Screen - connects the vanishing point to the center of the screen. This helps to visualize how the point is moving with respect to the screen center.
5. Overlay on - used to overlay the resulting graphics generated by this module onto different images. As the current image needs to be an edge detected image you can select to overlay the graphics on the original source image instead of the current edge image.
6. Filter Perspective Lines - when selected shows the perspective lines detected that are contributing to the vanishing point location.
7. Non-Perspective Lines - when selected shows the straight lines that were detected but did NOT contribute to the vanishing point location.
8. Threshold - The threshold provides a way to ignore vanishing point locations that are not strong enough due to the lack of perspective lines in the image.

Example

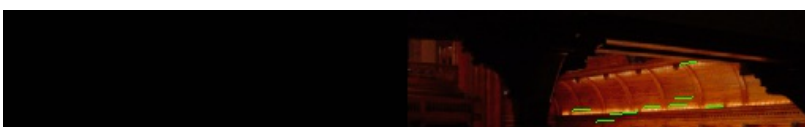
Hallway

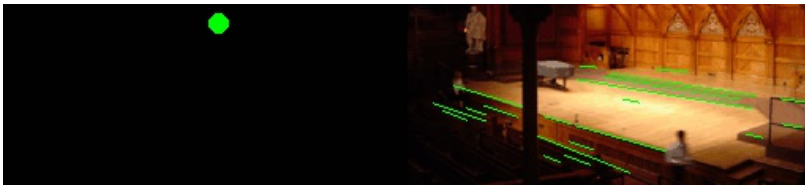


Hallway#2




Point outside of picture





Have a look at our [hallway video](#) while the vanishing point detector is running.

 [Click here](#) to load a configuration that detect the vanishing point in an image. Either switch on your camera and look down a hallway or drag an image into RoboRealm. Note that this example uses the Canny edge detection to find edges.

Variables

VANISHING_POINT_X - the X location of the vanishing point

VANISHING_POINT_Y - the Y location of the vanishing point

VANISHING_STRENGTH - an indication of how strong the point is (also used in the filter threshold interface)

See Also

[Visual Anchor](#)

AVM Navigator



The **AVM Navigator** module is a **third party** module that provides object recognition functionality that allows you to program your robot to recognize objects in the environment. Using the AVM Navigator module you can change robot behavior based on recognized objects or navigate the robot relative to those objects.

The AVM Navigator plugin is distributed outside of RoboRealm. To Install

1. Please [download](#) the zip file.
2. Extract the files in the c:\Program Files\RoboRealm\ folder. You will need admin permissions to do so. You may need to unzip to your desktop first and then copy the files to the RoboRealm folder.
3. Verify that the avm061.dll is in the RoboRealm folder and that the Navigator.dll and DVR_Server.dll is in the RoboRealm\Plugins folder.
4. Restart RoboRealm.
5. Look in the Plugins section in the Contents tab or type in Navigator in the Search tab. (Note, AVM is NOT the right name for the module).

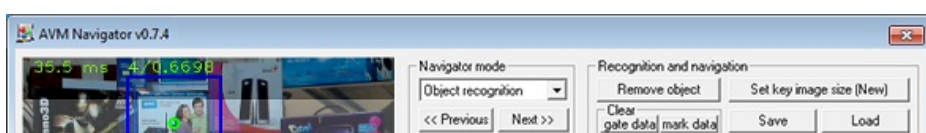
The **AVM Algorithm** uses a principle of multilevel decomposition of recognition matrices, is steady against camera noise, scales well with additional objects and is simple and quick to train. It also performs well on higher image resolutions (960x720 and more). The algorithm works with grayscale images.

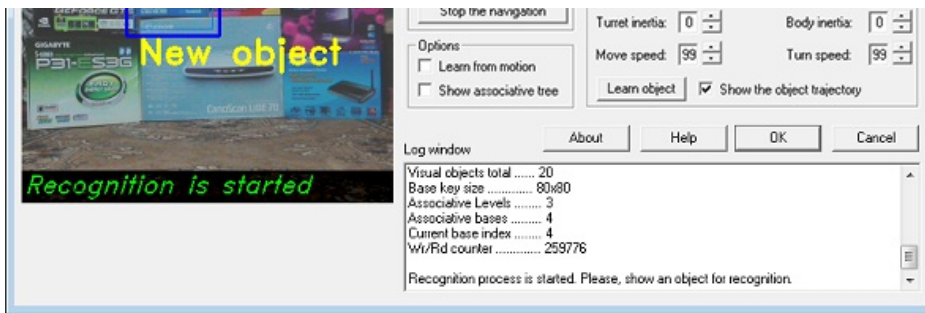
Detailed information about the AVM algorithm can be found [here](#).

The AVM Module has **several modes** that instruct the module how to perform.

Object Recognition

This mode allows you to train on a particular object as seen within the camera. Once trained the object will be identified when seen again.





1. Initialize AVM search tree by clicking of "**Set key image size (New)**" if it needed (this action clears all recognition data that were previously saved).
2. Press "**Learn object**" button to learn a new object for recognition.
3. You can hide an indication of object trajectory by disabling of "**Show the object trajectory**" option.
4. If you want to remove some object data from AVM search tree then you should press "**Remove object**" button.
5. All current data about learned objects, gates and marks is placed in the file "avm.dat". This file is located in user folder (for example: "C:\Documents and Settings\user\AVM\avm.dat"). You can save and load this data to/from another file by clicking "**Save**" and "**Load**" buttons and also you can remove data of objects, gates and marks selectively by clicking of "**Set key image size (New)**" that clear objects data, "**Clear/gate data**" or "**Clear/mark data**" buttons.
6. You can set "**Learn from motion**" option [for training on some movable object](#).
7. Choose "**Show associative tree**" option if you want to see [diagram of AVM search tree](#).

*How to make an object/face tracking

First clear the AVM search tree (if it needed) by click on button "**Set key image size (New)**" and further press "**Yes**".

Now you can train AVM on some faces like in video below:

When training will be done you should use **variables** that described below for your **VBScript** program:

NV_OBJECTS_TOTAL - total number of recognized objects
 NV_ARR_OBJ_RECT_X - left-top corner X coordinate of recognized object
 NV_ARR_OBJ_RECT_Y - left-top corner Y coordinate of recognized object
 NV_ARR_OBJ_RECT_W - width of recognized object
 NV_ARR_OBJ_RECT_H - height of recognized object

As example you can use these VBScript programs that was published in [this topic](#).

*Speak an Object's Name

The variable NV_ARR_OBJ_IDX is an array and can be used to construct a string to speak using the following VBScript:

```
TotalObj = GetVariable("NV_OBJECTS_TOTAL")
ObjIdx = GetArrayVariable("NV_ARR_OBJ_IDX")
```



```

ObjName (0) = "1"
ObjName (1) = "2"
ObjName (2) = "3"
ObjName (3) = "4"
ObjName (4) = "5"

SpeakStr = "I see"

for i = 0 to TotalObj-1 step 1
SpeakStr = SpeakStr + " " + ObjName (ObjIdx (i))
next

if TotalObj = 0 then SpeakStr = SpeakStr + " nothing"

SetVariable "SPEAK_STR", SpeakStr

```

For the complete example [download](#) the robofile.

Navigate mode & Nova gate mode

Navigate mode - This mode is similar to the Object Recognition mode but it provides variables that specify which direction the object is in relation to the robot. Using these variables you can steer the robot towards a particular object. The algorithm attempts to align the position of a turret (with a camera fixed on top) and body of the robot to the center of the recognized object. If the object is far away it will alter the NV_FORWARD variable to signal a move forward. If it is too close the NV_BACKWARDS variable will be set to move the robot backwards. If multiple objects are recognized the object with the highest index (most recently learned object) is chosen.

Also this mode provide a walking of robot from gate to gate by the route.

Walking to the point number - Activating of route that was selected by user for autonomous navigation and sets destination checkpoint.

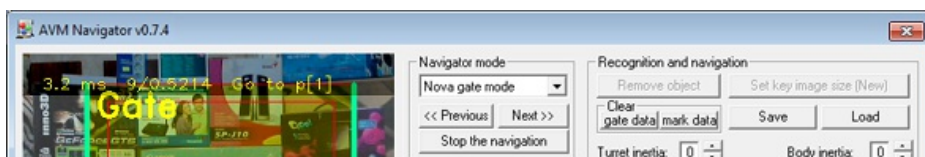
Turret/Body inertia - Different robots require different timings based on their mass and size. Use the Inertia numbers to adjust the movement of your robot to compensate for different sizes.

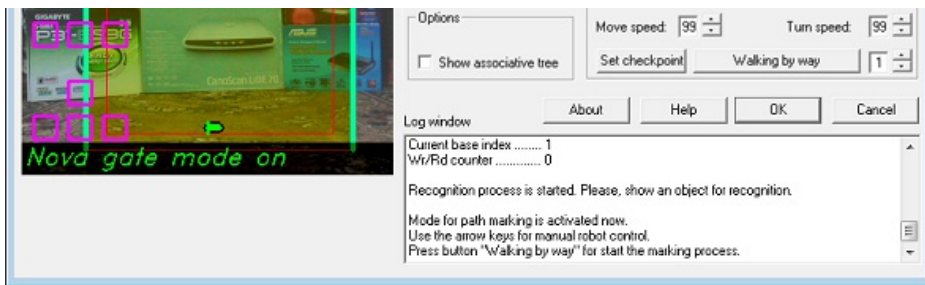
Move/Turn speed - Definition of maximal speed for robot moving and turning. These parameters can limit the values of control variables.



Nova gate mode - Similar to the Navigate mode this mode provides a visual stepping stone functionality by identifying 'gate' objects in succession.

By sequentially identifying objects you can lead a robot along a path of visual markers.





To train a route press the **"Walking by way"** button and then drive the robot manually through the route. The appropriate gates will be created automatically as the robot progresses through its route. When you reach the end of the route press the **"Set Checkpoint"** button which will cause the robot to turn on that spot and mark the spot as a checkpoint.

Once a route is complete, the robot can re-travel that route by identifying the same gates and use these gates to steer the robot along the correct path.

If the robot loses track of where it is along the route (i.e. no gates are in view) it will turn around on one spot to search for the next gate in the route sequence. As targeting gates may be momentarily lost from time to time this behavior may cause a slight twitching of the robot from time to time.

Historically first was developed *"Navigate mode"* and *"Nova gate mode"* that provided a walking of robot from gate to gate. The gate is an image (from robot camera) that associates with specific data inside AVM tree. The gate data contains weights for the seven routes that indicate the importance of this gateway for each route. At the bottom of the screen is an indicator "horizon" which shows the direction to adjust the robot's motion for further progress along the route. The gates are painted blue if the gates do not participate in the current route (weight rate 0) with warmer colors (up till yellow) show a gradation of "importance" of the gate in the current route.

These old modes were left within *AVM Navigator* just for compatibility with previous versions:

You should use more advanced navigation solution such as **"Navigation by map"** with **"Marker mode"** for route recording.

But you can use *"Nova gate mode"* for manual robot control with helping of arrow keys.

*How to get start

1. Open RoboRealm dialog window.
2. Make sure that "Camera" button is pressed in RoboRealm dialog window and also you should check out the camera resolution (it must be 320x240 pixels).
3. Call the dialog window of AVM Navigator (click on it at video-processing pipeline) and then switch to *"Nova gate mode"*. Now you can control your robot by arrow keys and also you can turn robot camera by "Delete" and "Page Down" keys ("End" key will set the camera in front position). If camera was turned you have to press "End" for alignment before continuing of robot moving.

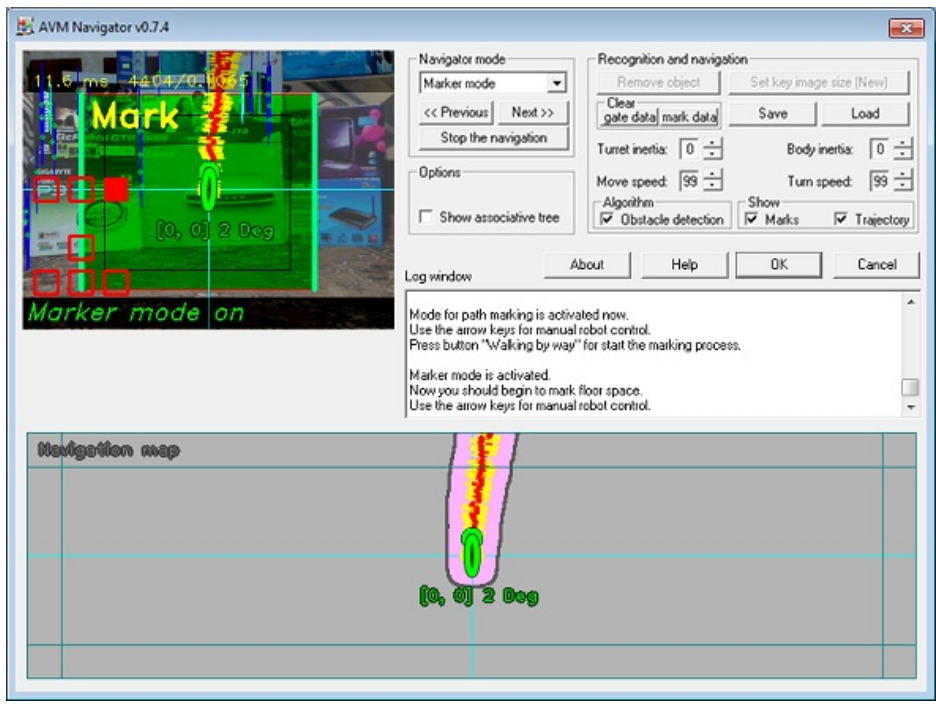
...

Find out more [here](#).

Also you can try to train with ["AVM Quake 3 mod"](#) for acquaintance with *"Marker mode"* and *"Navigation by map"* modes.

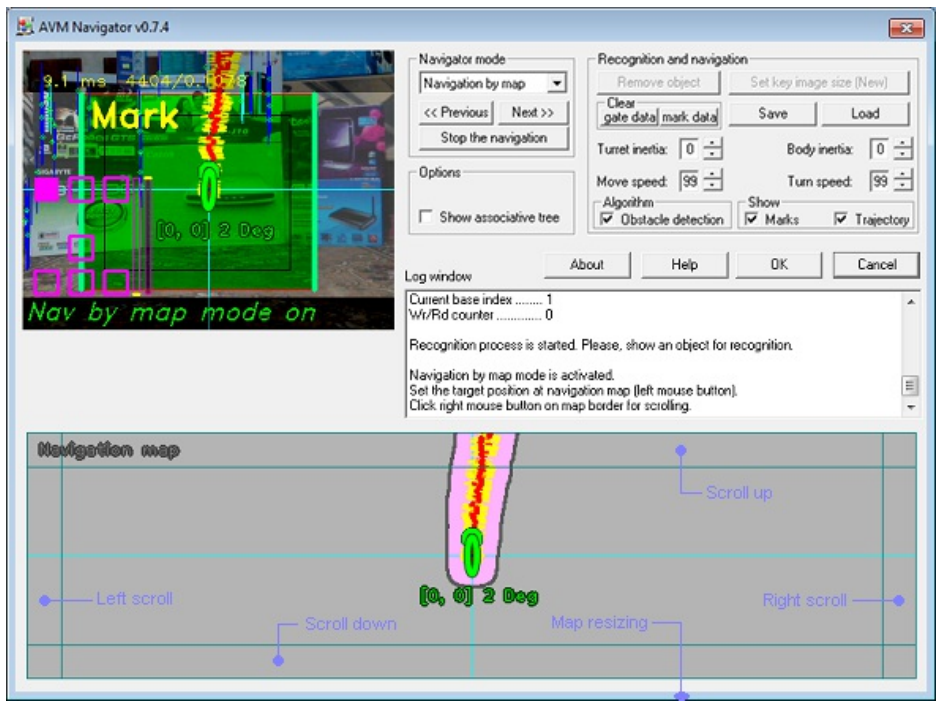
Marker mode

The marker mode provides for the creation of the navigation map that will done automatically by manually leading the robot along a path. For best results you should repeat the path several times in order to create appropriate map details.

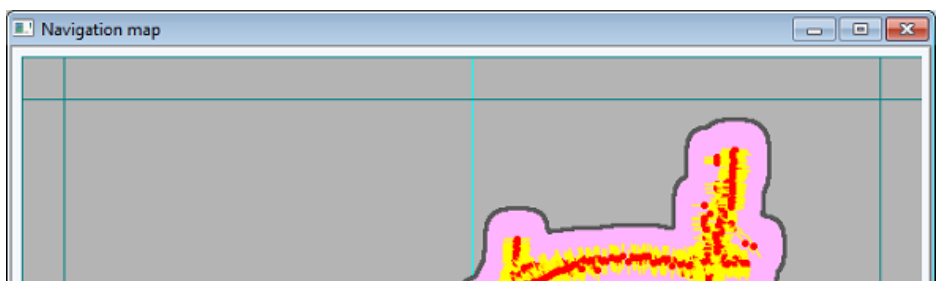


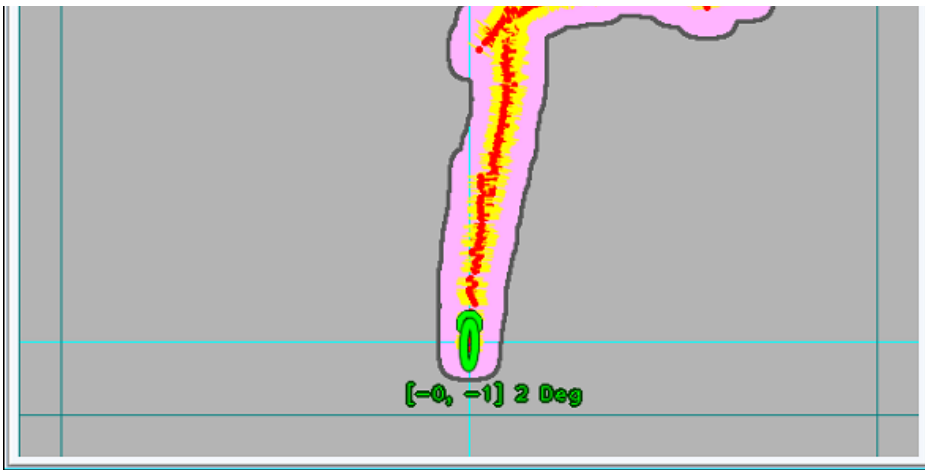
Navigation by map

In this mode you should first point the robot to the target position.



The robot then plans the path from the current location to the target position (big green circle) and then begins walking to the interim point of path (big red circle). When the interim point is achieved the robot gets the new direction to the next waypoint and so forth.





*How it works?

In our case the visual navigation for robot is just sequence of images with associated coordinates that was memorized inside AVM tree. The navigation map is presented at all as the set of data (such as X, Y coordinates and azimuth) that has associated with images inside AVM tree. We can imagine navigation map as array of pairs: [image -> X,Y and azimuth] because tree data structure needed only for fast image searching. The AVM algorithm can recognize image that was scaled and this image's scaling also is taking into consideration when actual location coordinates is calculating.

Let's call pair: [image -> X,Y and azimuth] as location association.

So, each of location association is indicated at navigation map of AVM Navigator dialog window as the yellow strip with a small red point in the middle. You also can see location association marks in camera view as thin red rectangles in the center of screen.

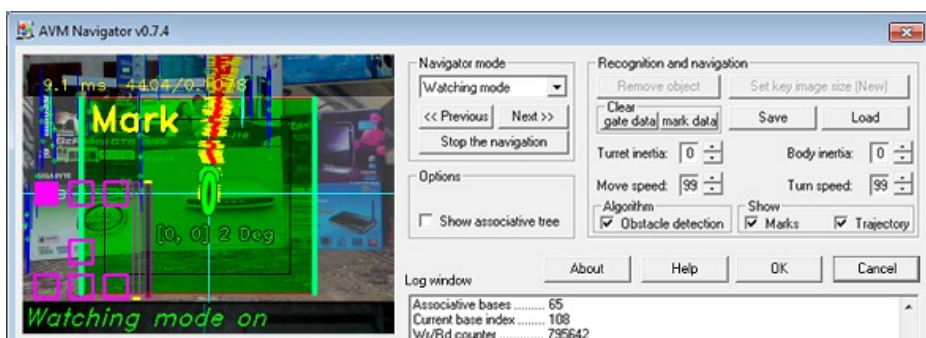
And so, when you point to target position in "Navigation by map" mode then navigator just builds route from current position to target point as chain of waypoints. Further the navigator chooses nearest waypoints and then starts moving to direction where point is placed. If the current robot direction does not correspond to direction to the actual waypoint then navigator tries to turn robot body to correct direction. When actual waypoint is achieved then navigator take direction to other nearest waypoint and so further until the target position will be achieved.

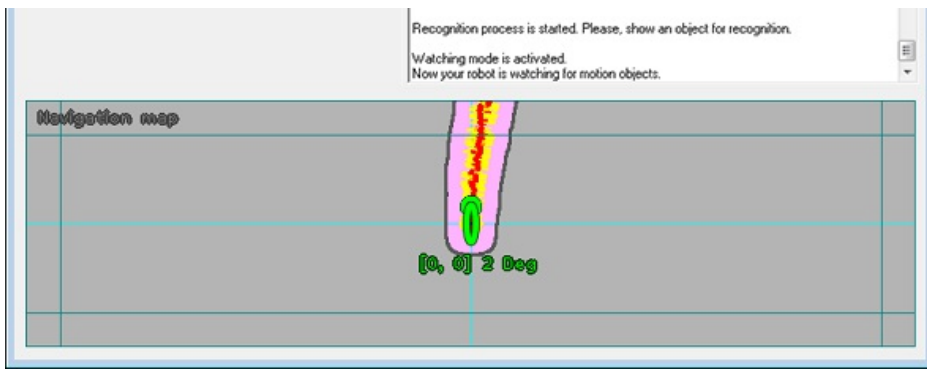
See [here](#) for more details.

First in video the robot has received command: "go to the checkpoint" and when robot arrived to the checkpoint then he was brought back to the start position of learned route. When robot noticed the changes then it indicated that robot was displaced because any commands were not given by robot to his motors however changes were seen in the input image. Then robot started looking around and **localized his current position**. Further the robot just calculated path from current position to the checkpoint and went there again.

Watching mode

The robot can move to direction where motion was noticed in this mode.





If robot see motion then he go to this direction but if further there is no motion then robot will come back to previous position.

*Obstacle avoidance

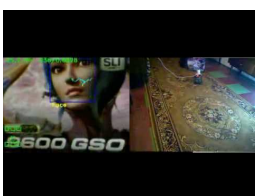
The robot processes the motion history of input images for the previous second. If any commands were given (i.e. the "forward" command) but no changes were seen in the input image (nothing happened) then the robot is assumed to be stuck (i.e. the robot is pushing against a wall). When stuck the module will set the movement variables to move backwards. The stuck situation is indicated by a red rectangle with a circle in the center.

*Odometry / localization

The robot sets the marks (it writes central part of the screen image with associated data to AVM tree). Marker data (inside AVM) contain horizon angle (azimuth), path length from start and X, Y location position (relative to the start position). Information for the marker data is based on marks tracking (horizontal shift for azimuth and change of mark scaling for path length measurement). Generalization of all recognized data of marks in input image gives actual value of azimuth and path length. If we have information about motion direction and value of path length from previous position and x, y coordinates of previous position then we can calculate the next coordinates of the current position. This information will be written to the new mark (inside AVM) when it is created and so forth.

The set of marks inside the AVM gives a map of locations (robot see the marks and recognize its location).

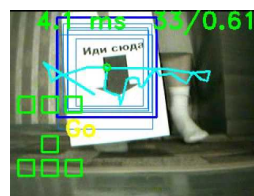
Examples



[Robot & objects #1](#)



[Robot & objects #2](#)

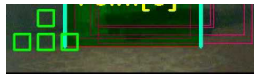


[Follow Me](#)





[Walking by gates #1](#)



[Walking by gates #2](#)



[Orientation Based on Beacons](#)



[Target Training](#)



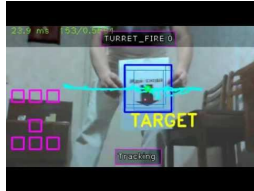
[Visual Odometry](#)



[Navigation by map](#)



[Playing with robot](#)



[Object tracking](#)



[Back to checkpoint](#)

Quake 3 Mod



Don't have a robot just yet? Then click [here](#) to view the manual that explains how to setup RoboRealm with the AVM module to control the movement and processing of images from the Quake first person video game. This allows you to work with visual odometry techniques without needing a robot!

The additional software needed for this integration can be downloaded [here](#).

Variables

The *AVMNavigator* module produces several variables that can be used to guide a robot.

NV_FORWARD - set when the robot should move forward

NV_BACKWARDS - set when the robot should move backwards

NV_LEFT - set when the robot should move forward or backwards to the left side

NV_RIGHT - set when the robot should move forward or backwards to the right side

NV_TURRET_RIGHT - set when the robot should move the turret (with an attached camera) to the right ("Delete" key in Nova gate mode)

NV_FIRE - reserved signal (however it can be activated if you press the "End" key in Nova gate mode). If the "End" key is pressed in "Marker mode" it will set NV_TURRET_BALANCE to zero (turns the camera to face the front).

NV_TURRET_LEFT - set when the robot should move the turret to the left side ("Page down" key in Nova gate mode)

NV_TURRET_BALANCE - indicates the turn degree amount. This value range from -100 to 100 with forward being zero.

NV_OBJECTS_TOTAL - total number of recognized objects

NV_OBJECT_IDX - index (identifier) of object

NV_ARR_OBJ_IDA - INDEX (IDENTIFIER) OF OBJECT

NV_ARR_OBJ_RECT_X - left-top corner X coordinate of recognized object

NV_ARR_OBJ_RECT_Y - left-top corner Y coordinate of recognized object

NV_ARR_OBJ_RECT_W - width of recognized object

NV_ARR_OBJ_RECT_H - height of recognized object

NV_ARR_OBJ_SIM - similarity rate (0...1) of recognized object

NV_ARR_OBJ_CORNER - left-top corner coordinate of recognized object. This variable contains both the x and y value. (X,Y)

NV_ARR_OBJ_RECT - the rectangle coordinates of the recognized object consisting of 8 numbers comma seperated. (X1,Y1,X2,Y2,X3,Y3,X4,Y4).

NV_IN_WAY_NUMBER - the cureernt route number (from 1 to 7)

NV_IN_SUBMIT_WAY - submitting of route number (value should be set 0 -> 1 for action)

NV_IN_CHECKPOINT - the current setting of checkpoint in Nova gate mode. It also needs to set 0 -> 1 for action.

NV_GATE_HORIZON - indication of gate horizons (float from -1 to 1) which shows the direction to adjust the robot's motion (0 - no adjustment)

NV_CHECKPOINT_NOW - indicates the checkpoint number when the robot arrives at the end of the current route (1-7). This variable is 0 if the checkpoint has not yet been recognized.

NV_LOCATION_X - current location X coordinate;

NV_LOCATION_Y - current location Y coordinate;

NV_LOCATION_ANGLE - horizontal angle of robot in current location (in radians).

NV_IN_TRG_POS_X - target position X coordinate of the navigation map

NV_IN_TRG_POS_Y - target position Y coordinate of the navigation map

NV_IN_SUBMIT_POS - submitting of target position (value should be set 0 -> 1 for action).

NV_IN_SET_MODE - the current setting of module's mode:

- 0 - Object recognition
- 1 - Navigate mode
- 2 - Nova gate mode
- 3 - Marker mode
- 4 - Navigate by map
- 5 - Watching mode

- Additional control variables (scaled to 255) -

NV_L_MOTOR_128

NV_R_MOTOR_128 - motors control

NV_TURRET_128 - control of camera turning

NV_TURRET_INV_128 - inversed control of camera turning

NV_DAT_FILE_NAME - data file name

NV_LOAD_DATA - the non-zero value of this variable
provides a signal for data loading

NV_REMOVE_IDX - the object index for removing. This variable provides removing of
recognition data from AVM with object index that was specified by the
user and further it will be switched to -1.


NV_WAYPOINT_X - current waypoint X coordinate;


NV_WAYPOINT_Y - current waypoint Y coordinate;

Recommended use of control variables

Use variable NV_TURRET_BALANCE for camera turning: NV_TURRET_BALANCE - indicates the turn degree amount. This value range from -100 to 100 with forward being zero. Use for motor control NV_L_MOTOR and NV_R_MOTOR variables that have range from -100 to 100 for motion control ("100" - full power backwards, "100" - full power forwards, "0" - motor off).

Note that the commands NV_LEFT and NV_RIGHT should be interpreted along with the commands NV_FORWARD and NV_BACKWARDS.

 [Click Here](#) to load a robofile that shows the object variables.

 [Click Here](#) to load a robofile that shows the control variables.

Digital Video Recording system (DVR)

The **DVR** module is a **third party** module that provides video recording functionality.

The DVR plugin is distributed outside of RoboRealm To Install

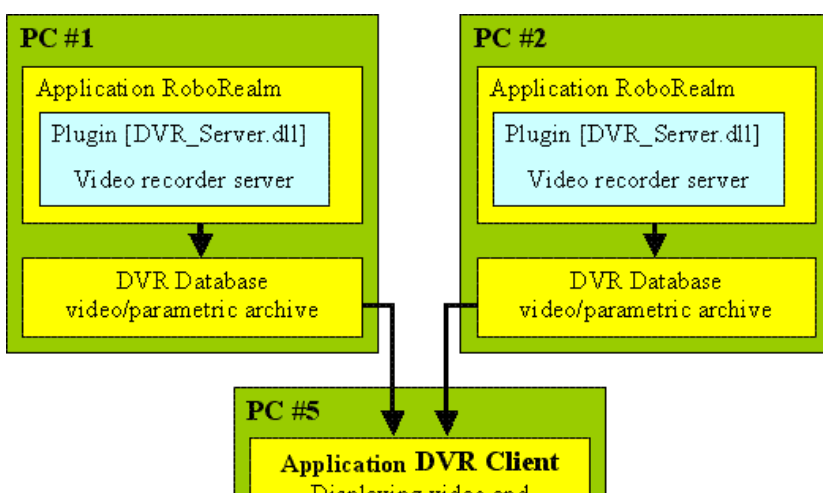
1. Please [download](#) the zip file.
2. Extract the files in the c:\Program Files\RoboRealm\ folder. You will need admin permissions to do so. You may need to unzip to your desktop first and then copy the files to the RoboRealm folder.
3. Verify that the avm061.dll is in the RoboRealm folder and that the Navigator.dll and DVR_Server.dll is in the RoboRealm\Plugins folder.
4. Restart RoboRealm.
5. Look in the Plugins section in the Contents tab or type in DVR in the Search tab.

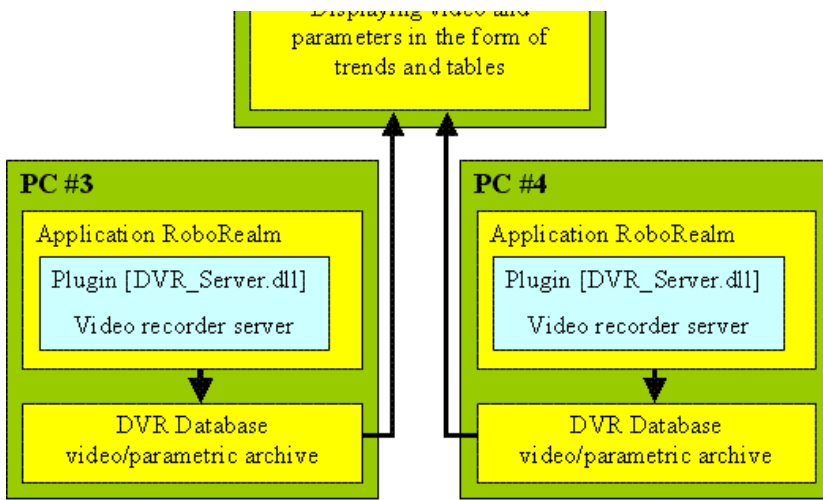
You can use the "**DVR Client-server**" package as a **Video Surveillance System** in which parametric data (such as VR_VIDEO_ACTIVITY) from different video cameras will help you search for a video fragment that you are looking for.

You can use the "**DVR Client-server**" package as a powerful instrument for debugging your video processing and control algorithms that provides access to the values of your algorithm variables that were archived during recording.

Technical Details

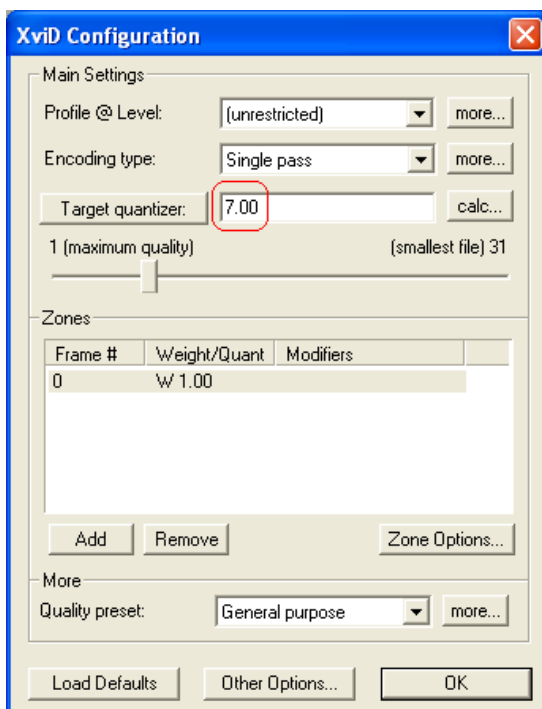
- **ring video/parametric archive** with duration of 1 - 12 months;
- **configurable database record** (for parametric data) with maximal length of 190 bytes;
- writing of parameters to database with discretization **250 ms**;
- the **DVR Client** can work **simultaneously with four databases** that can be located at remote computers.



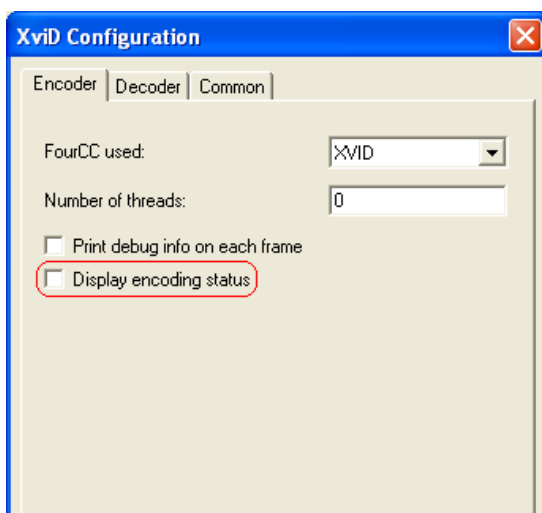


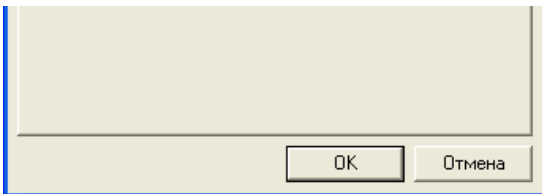
Setup

1. The "DVR Client-server" package uses the [XviD codec](#) for purpose of video archiving. Therefore, you should install the [XviD codec](#) before using the DVR module.
2. Activate the dialog window of **XviD Configuration** (choice a "Xvid / Configure Encoder") and then set the "Target quantizer" parameter to 7.0 or higher.

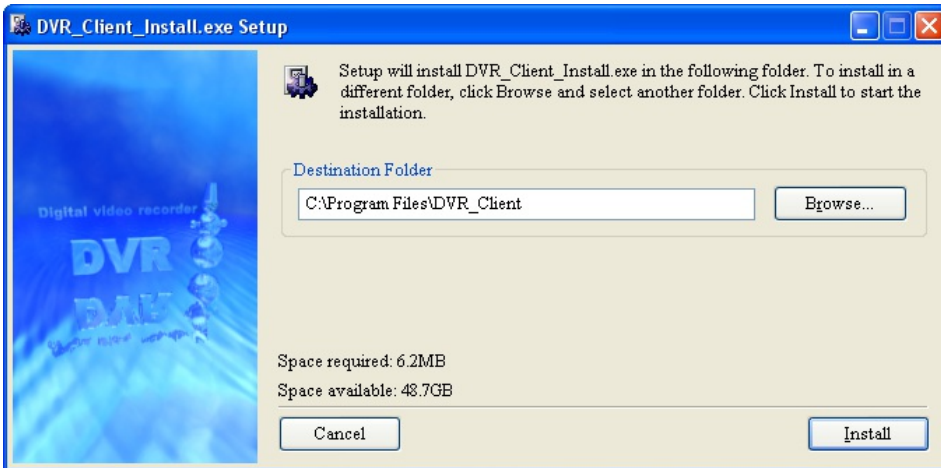


3. Click the "Other Option" button and then disable the option "Display encoding status".



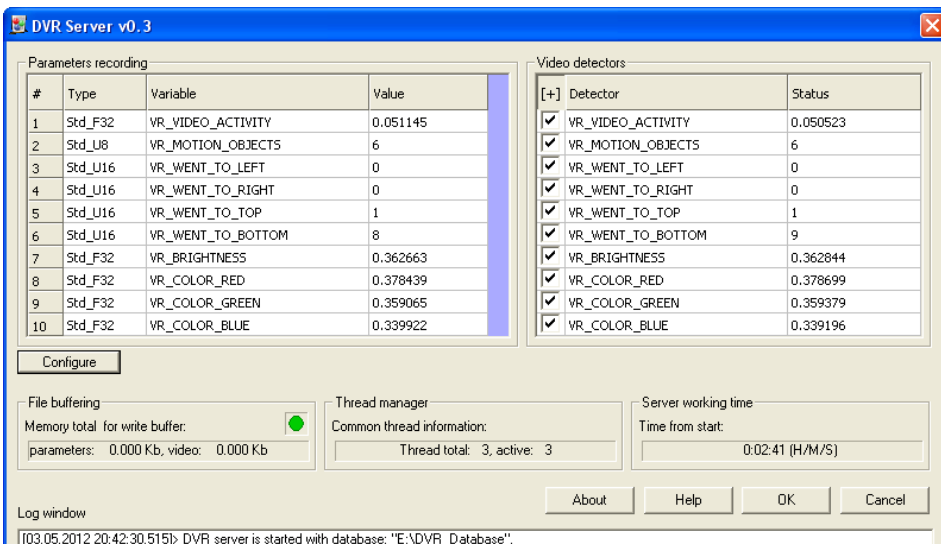


4. Install the RoboRealm package with DVR Server v0.1 (or higher) plugin.
5. **Start RoboRealm** after installation and **add DVR Server** plugin to video-processing pipeline.
6. Activate the dialog window of DVR Server (click on it at video-processing pipeline). Then choose the database directory location.
7. After the RoboRealm installation you should install [DVR_Client_Install.exe](#) package. When installation is complete you should set the database directory path in the client program (chose a "Service/Configuration/Camera #1 database path"). Now the DVR Client is ready to work.

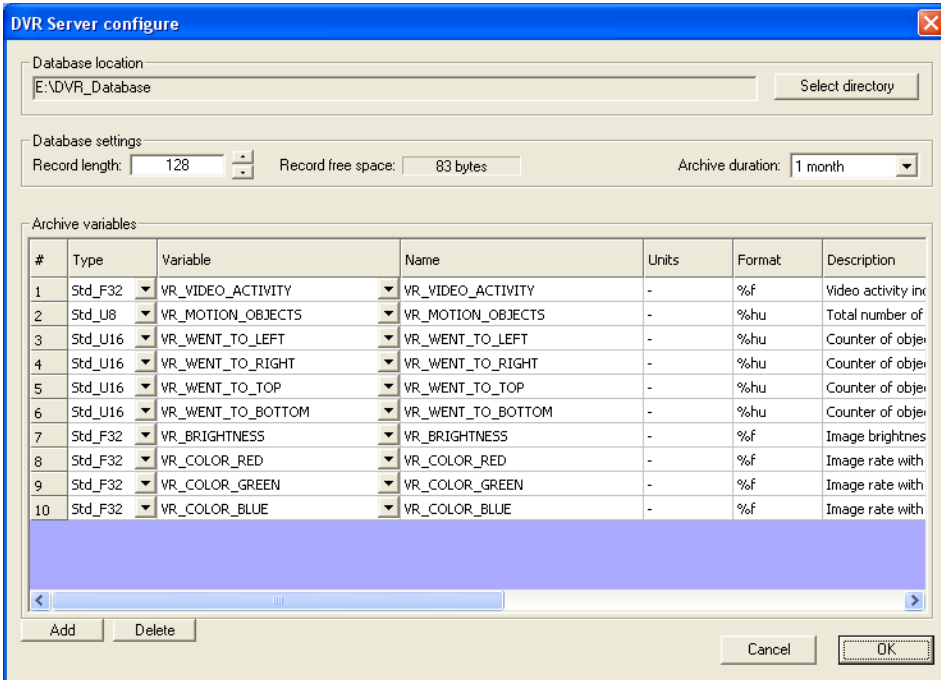


DVR Server configuration

1. Call the dialog window of DVR Server (click on it at video-processing pipeline) and click on the "Configure" button.



- Now you can configure record length (16 - 190 bytes) of parametric database and change archive duration (1 - 12 months). Also you can change the database directory location by clicking on the "Select directory" button.



- You can then create a variable list of database records. Click "Add" button for adding of new description entry for the archive variables table and then select a variable from drop-list and choose the **variable type**:

Name	Description	Size	Range
Std_U8	Unsigned small integer	1	0 to 255
Std_S8	Signed small integer	1	-128 to 127
Std_U16	Unsigned short integer	2	0 to 65535
Std_S16	Signed short integer	2	-32768 to 32767
Std_U32	Unsigned integer	4	0 to 4294967295
Std_S32	Signed integer	4	-2147483648 to 2147483647
Std_U64	Unsigned long integer	8	0 to 18,446,744,073,709,551,615
Std_S64	Signed long integer	8	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807
Std_F32	Floating point number	4	+/- 3.4e +/- 38 (~7 digits)
Std_F64	Double precision floating point number	8	+/- 1.7e +/- 308 (~15 digits)

- You can edit a variable name that will be displayed in tables and trends (if it needed).
- Specify **units** in which variable is measured.
- Specify the printing **format** for the variable for displaying in tables and trends (see [sprintf format specifiers](#)).
- Specify variable's **description** for displaying as additional information in DVR Client program.
- Specify **Min** and **Max** limitation for correct scaling of variable in trends.
- Specify **"Show"** flag for variable displaying in tables or trend or both.

DVR Client configuration

- You should call the **client configuration** dialog window (choice a "Service/Configuration") for the configuration of the database path and displaying variables in the main **DVR Client** window.

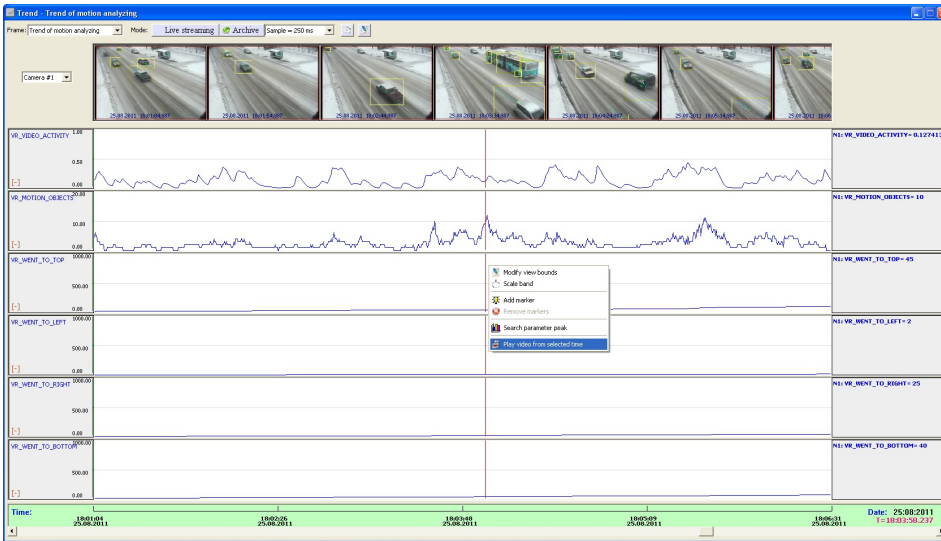


2. You can **configure** staff of **displaying variables** in tables and trends from "Create frame" dialog window (choice a "View archive/New frame").

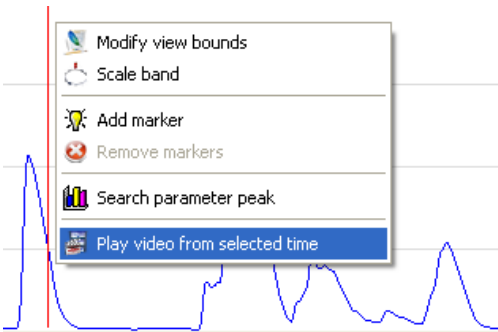
You have to press "Save frame" button for accepting of changing and then you should close and open again table or trend window for refreshing of variable staff.

3. Choose the "View archive/Trends" or "View archive/Tables" for calling of **tables** or **trends** dialog window.

	18:11:26.250	18:11:26.750	18:11:27.000	18:11:27.250	18:11:27.500	18:11:27.750	18:11:28.000	18:11:28.250	18:11:28.500	18:11:28.750	18:11:29.000	18:11:29.250
NI_VR_VIDEO_ACTIVITY	0.342026	0.345979	0.342794	0.330009	0.311910	0.302427	0.288250	0.272268	0.255656	0.235865	0.219276	0.202010
NI_VR_MOTION_OBJECTS	3	3	3	3	3	3	3	2	2	2	2	2
NI_VR_WENT_TO_LEFT	1	1	1	1	1	1	1	1	1	1	1	1
NI_VR_WENT_TO_RIGHT	15	15	15	15	15	15	15	15	15	15	15	15
NI_VR_WENT_TO_TOP	32	32	32	32	32	32	32	32	33	33	33	33
NI_VR_WENT_TO_BOTTOM	33	33	33	33	34	34	34	34	34	34	34	34
NI_VR_BRIGHTNESS	0.625998	0.644038	0.644033	0.640099	0.627174	0.613222	0.604669	0.605961	0.620710	0.630099	0.640194	0.644891
NI_VR_COLOR_RED	0.631607	0.632984	0.632681	0.629718	0.626023	0.619981	0.621786	0.624904	0.626954	0.628322	0.629630	0.630214
NI_VR_COLOR_GREEN	0.631621	0.651172	0.655683	0.647138	0.643460	0.637491	0.640723	0.642166	0.643756	0.644908	0.646234	0.647028
NI_VR_COLOR_BLUE	0.625905	0.649503	0.645340	0.637025	0.633291	0.627662	0.631295	0.632109	0.633095	0.634758	0.635956	0.636992

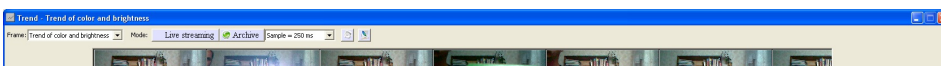


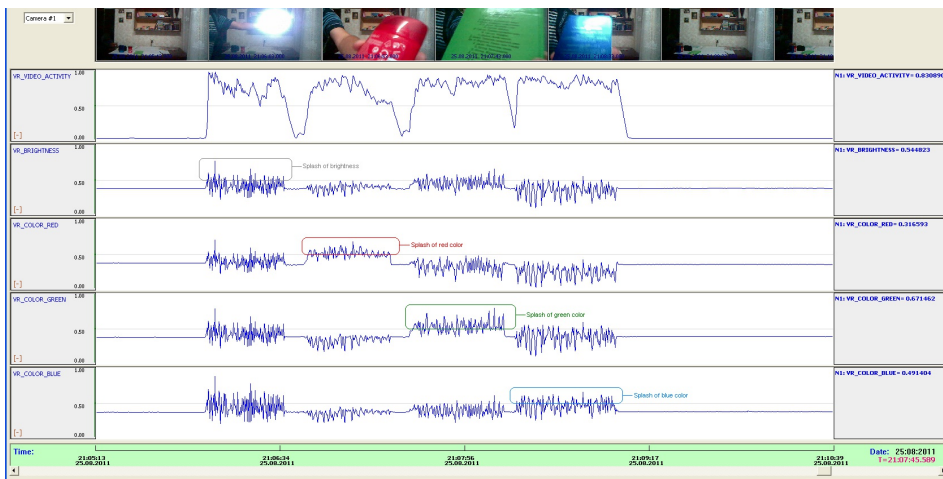
4. **Double-click** the left mouse button on video bar for playback or click right mouse button on the trend for popup menu activation and then select "Play video from selected time".



Video detectors

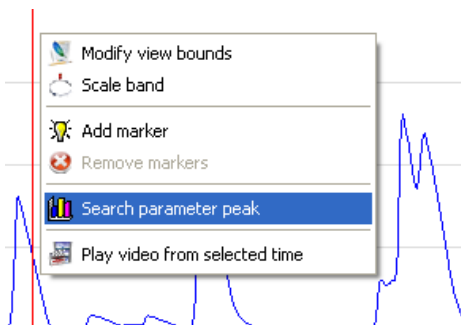
1. You can use information that was archived from video detectors for **quick searching** of video fragment that you need. For example it can be splash of brightness, color or number of motion objects.



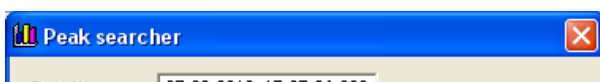


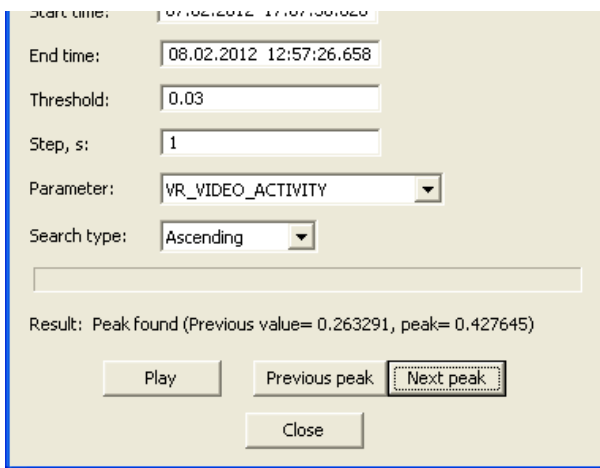
Video detector	Description
VR_VIDEO_ACTIVITY	Video activity indicator
VR_MOTION_OBJECTS	Total number of motion objects
VR_WENT_TO_LEFT	Counter of objects that went to the left
VR_WENT_TO_RIGHT	Counter of objects that went to the right
VR_WENT_TO_TOP	Counter of objects that went to the top
VR_WENT_TO_BOTTOM	Counter of objects that went to the bottom
VR_BRIGHTNESS	Image brightness rate
VR_COLOR_RED	Image rate with red color
VR_COLOR_GREEN	Image rate with green color
VR_COLOR_BLUE	Image rate with blue color

2. For searching of the **parameter peaks** you should click right mouse button on the trend for popup menu activation and then select "**Search parameter peak**".



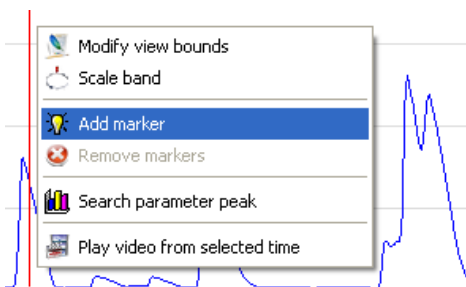
The "**Peak searcher**" dialog allows you to select time interval for searching (*Start/End time*), select an acceptable rate *threshold* of parameter acceleration and a search *step*. Also you can choose a *parameter* and the *search type* (ascending or descending). You can use buttons "*Previous peak*" and "*Next Peak*" for navigation between peaks and button "*Play*" for viewing of the video from position where peak was found.



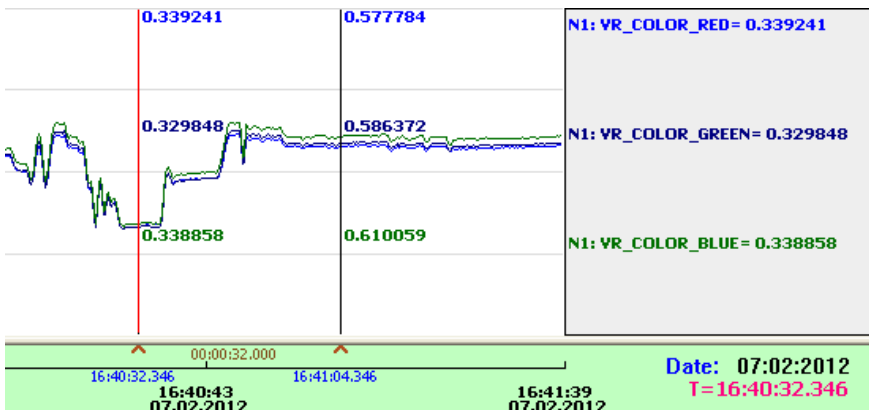


Trend markers

1. You can use markers for more obvious presentation of parameter values in the trends.

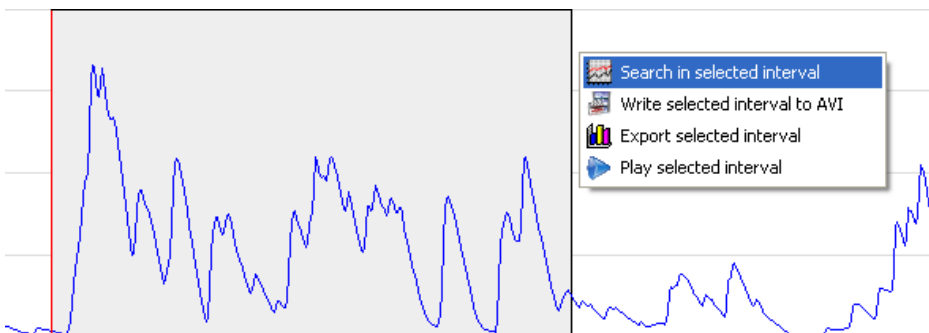


You should use "Add marker" or "Remove markers" popup menu items for markers setting or removing.



Interval selection

1. **Press** the left mouse button and **pull** mouse cursor to the right side of the trend for activation of interval selection.



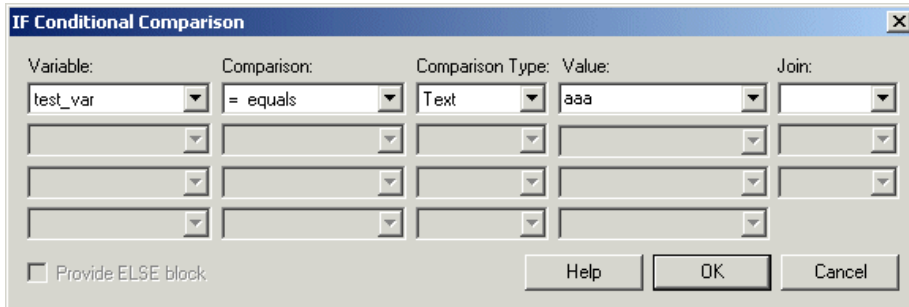
Further in popup menu you can choose: peak searching, writing video to AVI, exporting parameters to Excel or viewing of selected fragment of video.

See [DVR-Client demo](#) video for more details regarding user interface.

If Statement

The If Statement module allows you to create a condition on which the enclosed modules will get executed or not. This is similar to the conditional statements seen in the VBScript and other scripting languages. Using the interface you can compare RoboRealm variables against other variables or values to determine if the following modules should be executed or not.

Interface



Instructions

1. Variable - Specify the variable to test using the dropdown menus. Select [new] to create a new variable that is not already in the list.
2. Comparison - Select the appropriate comparison.
3. Type - Select the comparison type. The comparison type helps specify how the comparison is performed. For example, if two strings "100" and "2" are compared as text "2" will be greater than "100", but if compared as integers then "100" will be greater than "2".
4. Value - Specify the value to compare to. For variables use the dropdown list. For other static values such as numbers or strings simply type in the value.
5. Join - Specify the type of join to perform on multiple conditions
6. Latch - Select the Latch Condition if you only want the If Statement to trigger when the condition transitions from false to true, i.e. once true, the condition will not trigger until the condition evaluates to false and then true again.

See Also

[Set Variable](#)

Exit Statement

The Exit Statement module stops the image processing pipeline and allows RoboRealm to grab the next image frame and start the pipeline again. This happens regardless of which tab current execution is in.

See Also

[Return Statement](#)
[If Statement](#)

Return Statement

The Return Statement module stops the image processing pipeline in the current tab and returns execution to the calling tab. If the current tab is the Main tab, execution stops and restarts the pipeline with the next available image.

See Also

[Exit Statement](#)
[If Statement](#)

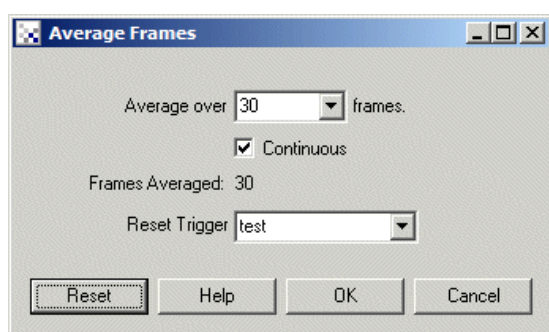
Image Averaging

Frame Averaging provides a way to average multiple video frames to create a more stable image. This module can be used to eliminate pixel vibrations or high frequency image changes.

The frame averaging works by adding each frame into a moving average of frames. This effectively creates the same effect of averaging many frames without the significant memory and time that averaging hundreds of frames would take.

The averaging effect can be used to remove fast moving objects from frames if a high frame averaging is used. If fewer frames are averaged the effect creates a washing or strobe like effect (way cool!!).

Interface



Example



[Averaging of 4 frames. Click to view \(345 Kb AVI\)](#)

Instructions

1. Average over - Specify the number of frames to average.
2. Continuous - Specify if the averaging should be continuous or stop after the specified number of frames have been averaged.
3. Reset Trigger - Similar to the Reset button the reset variable when nonzero will cause the averaging to reset to zero and start accumulation of frames again. This can be used to cause a reset to happen based on a variable value.

5. Reset Button - You can press Reset to start the averaging from zero again.

See Also

[Temporal Median](#)

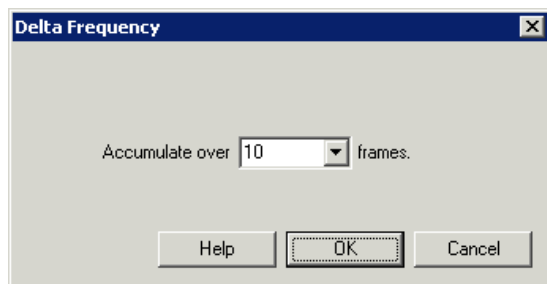
[Temporal Mode](#)

[Movement](#)

Delta Frequency

The Delta Frequency module is used to detect quick movement, i.e. when a pixel is changing rapidly from frame to frame. This happens when either quick movement is present or a grated object is passing through the field of view of the camera.


Interface



Instructions

1. Select how many frames the delta should be accumulated over. The fewer the frames the slower the movement need to be to trigger a visible effect.

Example

 [Click here](#) to load a configuration that will say and print hello on the current image when you wave quickly at your camera. Note that if you don't have MS Speech installed you will only see the "hello" text on the screen.

See Also

[Movement](#)

Light Strobe



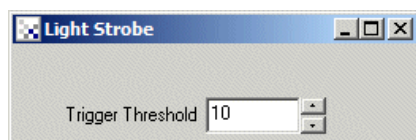
The Light Strobe module is meant to be used to capture images from a strobe light that does NOT have a light strobe trigger mechanism. This is the case with most webcams and some machine vision cameras. Despite the lack of a hardware trigger it is still desirable to use a strobe to correctly light an object for a moment. Without a hardware trigger this module is required to capture the best image as possible. The module will monitor the light level of each pixel and save the maximum value. When the light level falls below the specified trigger threshold (i.e. light levels are dropping) the module will trigger a variable and update the current image.

Using this method this ensures that the best lighting from multiple images is captured and retained. Even if the frame capture rate is slightly slower than the strobe pulse, the module can grab lighting from multiple frames to create a single image.

In some cases with moving objects creating a single frame from multiple frames will cause significant image artifacts. If this is the case, selecting "full frame" checkbox will ensure that the lightest entire frame is used which will remove the artifacts but sacrifice some of the potential light stabilization.

This module will also autodetect when an image is the same as before (as in the case of using a static test image). If this is detected the trigger variable is activated as would be in a live image. This ensures that single images will be handled in the same way as a live image.

Interface





Instructions

1. Activate Variable - Specify a variable at when contains the value 1,on,active, or true will cause the module to start sampling the image data for the lightest value per pixel. This is to ensure that accidental noise doesn't trigger an image update unintentionally. Without this activate variable, the system will continue to monitor the image stream and update the max values as appropriate.
2. Trigger Threshold - The amount of lighting drop off that triggers a 'new' image. The module will sample each pixel for its maximal lighting value. When this value drops off (is less than before) by the Thredhold percentage the module will assume the lighting stobe has been switched off and additional max values are not expected. This will cause the module to update its current image and make the image available to successive modules.
3. Full Frame - Best lighting is per frame instead of per pixel.
4. Timeout - There are cases when the module will not be able to correctly detect a drop in lighting due to very low light being available in the image. In these cases its good to set a timeout that will trigger regardless of image content after X milliseconds. This is useful when adjusted camera properties during a stobe event that could eliminate the image content and cause the module not to respond. During normal operation this feature is less relevant but can be used as a failsafe mechanism if your application is waiting for the LIGHT_STROBE_TRIGGER to change.

Variables

LIGHT_STROBE_TRIGGER - set to 1 when a new image is updated, 0 otherwise.

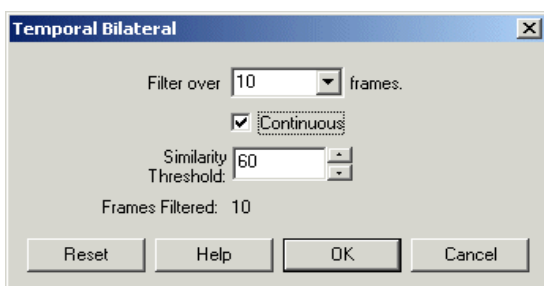
See Also

[Max](#)

Temporal Bilateral Filter

This module is similar to the [Average](#) module in that it operates on past frames in order to produce a more stable current frame. The difference with the Average module is that the Temporal Bilateral filter will perform a bilateral filter over the past X frames. This basically means that the current pixel value for a particular x,y coordinate is compared with the past X frames. Where the difference is lower than the specified amount the pixel is added into an accumulator to create the final new value of the pixel. This has a similar stabilizing effect as with the Averaging module but preserves moving edges without the ghosting effect seen with the Average filter.

Interface



Instructions

1. Filter over - Select how many past frames should be considered when creating the new pixel value
2. Continuous - Select continuous if the frames should be keep updating with scene changes
3. Threshold - The similarity threshold used when comparing the current image frame with the past images. Pixels that are within the difference threshold are averaged together to form the current pixel. Those that are outside of the range are ignored.

Note that in order to memorize past frames significant memory resources may be utilized to create enough room to store the past X frames. If your computer does not have enough memory resources for the given frame number either decrease the frame size or the number of frames used in the

analysis.

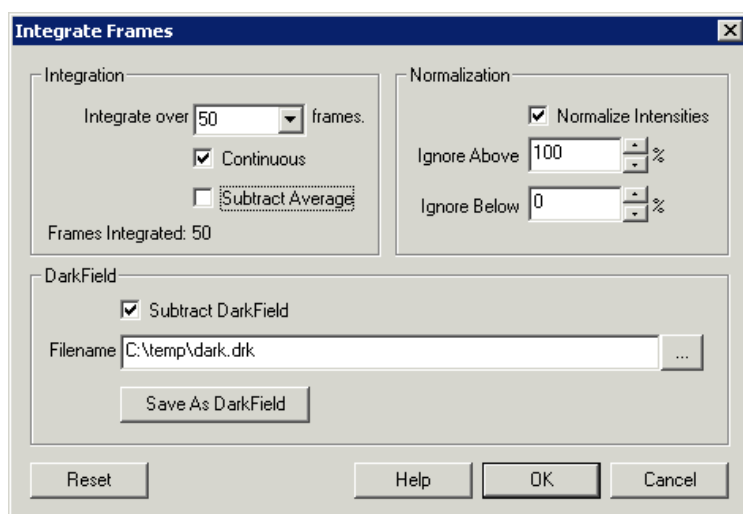
See Also

[Average](#)

Image Integration

The Image Integration module will keep adding images to the current image in an effort to increase the light in an image. Note that the image content will need to be very still during the light integration period. The Image Integration is similar to the Image Averaging but will instead normalize the image based on the lightest pixel seen in the image. To ignore situations where hot or cold pixels exist you can adjust the "Ignore Above" and "Ignore Below" percentages to disqualify the hot or cold pixel values from the normalization process.

Interface



Instructions

1. Specify the number of frames to integrate.
2. Specify if the addition should be continuous or stop after the specified number of frames have been integrated.
3. Select the subtraction checkbox if you want to subtract the current mean intensity value from the image prior to integration. This can be used in place of normalization to ensure that the image does not become a white wash.
4. Click on "Normalize Image" if you want the resulting integrated image to be normalized with respect to the pixel intensity range (0-255). This process can be modified by the following text boxes where you can modify the analyzed range.
5. Specify what high intensity white pixels should be ignored when normalizing the image. For example, if your CCD has a bright noise pixel that creates a bright pixel spot in the resulting image it will cause the normalization to suppress any darker areas of the image. Reducing the percentage from 100% will shrink that range and bring darker areas into view.
6. Similar to the "Ignore Above" you can also ignore dark pixel values that may cause the normalization range to hide darker areas.
7. If your CCD has any 'hot spots' you can use dark field/image subtraction to remove those high pixels. These are pixels that always appear brighter than other pixels and are typically seen as white pixels in the resulting image where no light was present. These pixels are always in the same spot and are caused by noise due to heat generated within your Camera. Using the DarkField subtraction you can reduce the noise generated by these pixels.
 - To generate a darkfield image put the lens cap on your camera and press the reset button in this interface to start integration. Once the number of frames specified has been integrated press the "Save As DarkField" button. This will save the current image as a darkfield image. The format of the darkimage is a 32 bit RGB value that also includes the integration frame count. This inclusion allows you to use the same darkfield image for different integration lengths during subtraction. Note that if you change the image dimensions you will need to recreate a new darkfield image.
 - To then use that image remove your camera lens cap and click on the "Subtract Darkfield" checkbox.

- Specify the darkfield image in the filename textbox or click on the "..."/> to browse to that file. Set it to the file you just saved to be subtracting it from the currently seen image.

Note that the "frames integrated" value shows how many frames have currently been integrated.

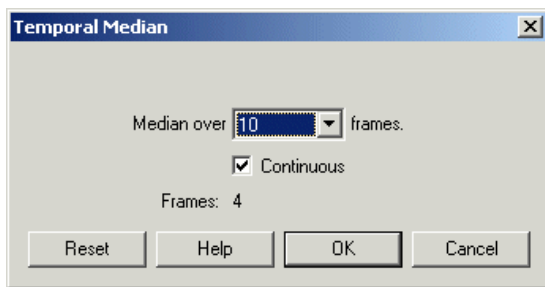
See Also

[Image Averaging](#)
[Movement](#)

Temporal Median

Similar to the [Average module](#) the Temporal Median module will determine the median pixel based on a number of frames (as apposed to the average module that calculates the mean value). This module is useful over the average module when only a limited number of frames are available to create a background image. For example, if you need to create a background image and can average over 100's of frames then the average module will work, however, if you only have 10 or less frames to create the background and all the frames have non-background objects in them then the Temporal Median module will work better to eliminate non-background based objects.

Interface



Instructions

1. Median over - Select how many frames should contribute to calculating the median. More frames are normally better. Note that there is a maximum limitation to 10 frames using the temporal median module. If you have access to more background images see the Average module (Temporal Mean).
2. Continuous - Select if the background image creation should keep using new images and removing old images. This allows for a background image to be updated over 10 images. If you just wish to sample the first X images as the background keep this unchecked.

Example

Source Images



Temporal Median (5 frames)



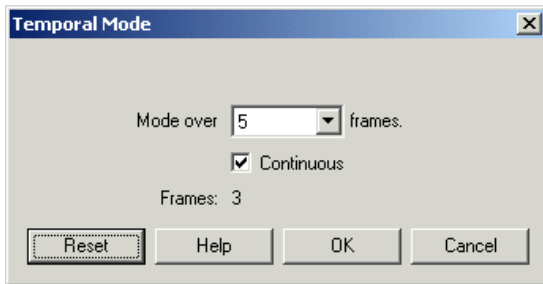
See Also

[Average](#)
[Temporal_Mode](#)

Temporal Mode

Similar to the [Average module](#) the Temporal Mode module will determine the most frequent pixel color within the specified number of frames (as opposed to the average module that calculates the mean value). This module is useful over the average module when only a limited number of frames are available to create a background image and the [Temporal Median](#) module failed to perform.

Interface



Instructions

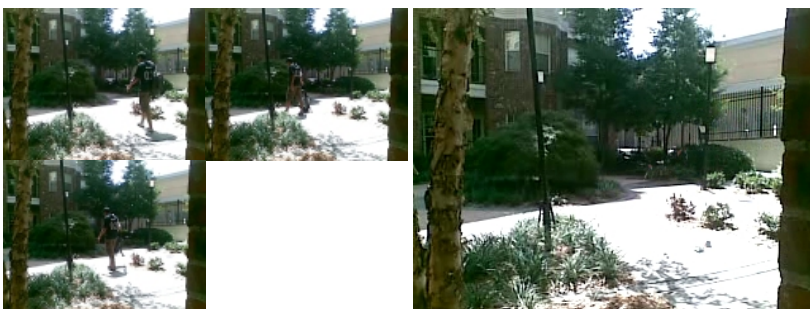
1. Mode over - Select how many frames should contribute to calculating the mode (most frequent) pixel. More frames are normally better. Note that there is a maximum limitation to 10 frames using the temporal mode module due to memory constraints. If you have access to more background images see the Average module (Temporal Mean).
2. Continuous - Select if the background image creation should keep using new images and removing old images. This allows for a background image to be updated over 10 images. If you just wish to sample the first X images as the background keep this unchecked.

Example

Source Images



Temporal Mode (5 frames)



See Also

[Average](#)

[Temporal Median](#)

Adaptive Threshold

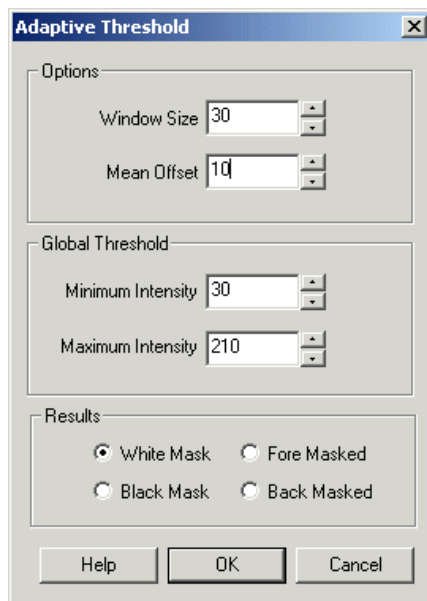
The Adaptive Threshold module is used in uneven lighting conditions when you need to segment a lighter foreground object from its background. In many lighting situations shadows or dimming of light cause thresholding problems as traditional thresholding considers the entire image brightness. Adaptive Thresholding will perform binary thresholding (i.e. it creates a black and white image) by analyzing each pixel with respect to its local neighborhood. This localization allows each pixel to be considered in a more adaptive environment.

The algorithm will consider each pixel one at a time, calculate the mean of the local neighborhood 'window size' ($x\text{-windowSize}/2$, $y\text{-windowSize}/2$).

$x+windowSize/2$, $y+windowSize/2$) and thresholds the current pixel to white if the difference between the calculated mean and the current pixel value is lower than the 'mean offset'.

The example below shows an image from our [line following tutorial](#). In this image the sides of the line image are dimmed due to uneven lighting. Adaptive Thresholding can solve this problem as long as the neighborhood considered (the pixel window) is large enough.

Interface

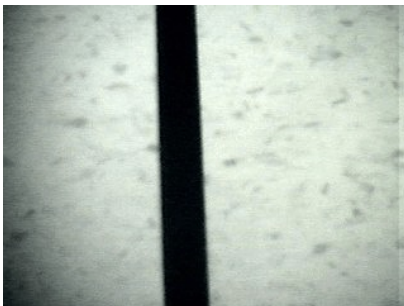


Instructions

1. Window Size - Specify a window size large enough to cause a good separation between background and foreground object.
2. Mean Offset - Specify how much the current pixel should differ from the mean in order to signal as 'on'. This helps to ensure stability of the white pixels by ensuring that they white pixels are higher than the mean by X amount. A low offset value will cause some pixels to vibrate between black and white if they are near the intensity edge.
3. Global Threshold - As the adaptive threshold technique is a local technique (i.e. it resolves the intensity levels within a particular area of the image) it can cause very low intensity areas to become white and very high intensity areas to become black based on the amount of texture with that area. Using the global threshold values you can specify that really dark or really white areas are ALWAYS dark or light respectively. This helps to reduce the noise caused by the local analysis of un textured light or dark areas. This is functionality similar to the [Threshold](#) module.
4. Results - Select how the results should be represented
 - White Mask - values above local mean are white, below are black
 - Black Mask - values above local mean are black, below are white
 - Fore Masked - values above local mean retain original value, below are black
 - Back Masked - value below local mean retain original value, above are white

Example

Source



Adaptive Threshold



See Also

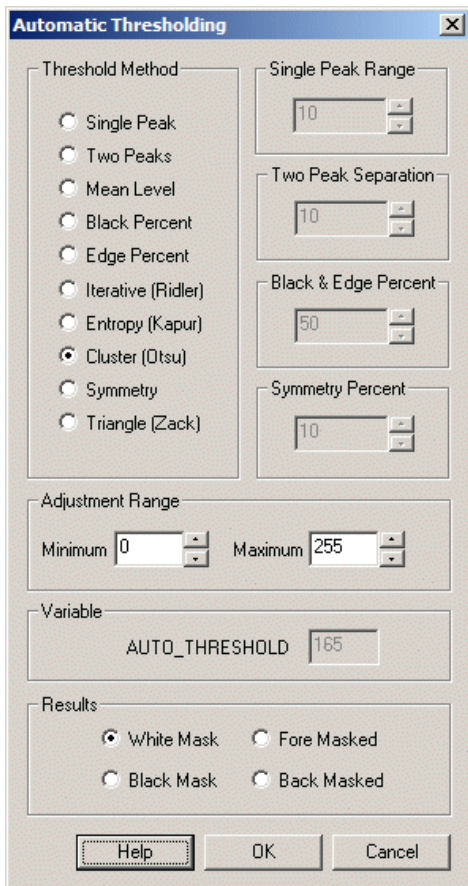
[Threshold](#)

Auto Threshold

The Auto Threshold module will automatically threshold the current image into a binary black and white image. The threshold level is automatically determined based on the method selected. The appropriate method to use will depend on your application. Select Cluster (Otsu) if you are looking for a standard technique that is most often referenced by the current machine vision literature.

This module is useful when working with blob analysis or shape recognition whose background image can change and a manual threshold is not reliable enough.

Interface



INSTRUCTIONS

1. Threshold Method - The following briefly outlines the algorithms used by the thresholding methods to allow you to chose the most appropriate for your application. Note that they all operate on the image's [histogram](#).

- Single Peak - Detects the highest peak in the histogram and preserves pixels around that peak as defined by the Single Peak Range.
- Two Peaks - Detects the two highest peaks in the histogram separated by the distance specified. The distance will ensure that peaks close to each other are not selected. The threshold is then found by finding the deepest valley between these two peaks.
- Mean Level - the average pixel value is determined using the image histogram. All pixel intensities below that value are set to black with all pixel intensities above the mean set to white.
- Black Percent - Also known as P-Tile. The threshold level is set based on the specified percent of suggested dark pixels (or background) there are in the image. The histogram is used to indicate how much of the image at a certain threshold would be set to black. Once this amount exceeds the specified percent the current histogram index (0-255) is used as the threshold.
- Edge Percent - Similar to the black percent the edge percent threshold is determined by the specified percent of edge pixels that exist below the threshold. Instead of just counting every pixel the edge percent bases its measurement on how much a pixel is part of an edge by performing a laplacian filter prior to threshold determination.
- Entropy (Kapur) - Utilizes Kapur's entropy formula to find the threshold that minimizes the entropy between the two halves of the histogram created by a threshold.
- Cluster (Otsu) - One of the most popular threshold techniques that creates two clusters (white and black) around a threshold T and successively tests the within-class variance of the clusters for a minimum. This algorithm can also be thought of as maximizing the between-class variance.
- Symmetry - Assumes that the largest peak in the histogram is somewhat symmetrical and uses that symmetry to create a threshold just before the beginning of the largest peak. This technique is particularly useful to segment objects from large background planes.
- Triangle - Works well with histograms that don't have well defined peaks. This technique finds the maximum distance between a suggested threshold value and a line that connects the first non-zero pixel intensity with the highest peak. Inherent in this technique is the distance of a point to a line equation.
- Clip - Removes pixels in the upper and lower X percent of intensity range. This is useful if you have noise in the higher and lower intensity ranges.

2. Single Peak Range - Defines how wide the range is for the single peak. For example, if the current image has the most pixels at 160 then a value of 10 would preserve those pixels whose values are from 150 to 170.

3. Two Peak Separation - Defines the required distance between the two selected highest peaks. The larger the value the more separated the peaks need to be.

4. Black & Edge Percent - Defines the percentage amount as needed by the Black Percent and Edge Percent methods.

5. Symmetry Percent - Specifies the percent amount when calculating the start and end of the highest peak.

6. Adjustment Range - Specifies the limits as to how the threshold can be generated. For example, if the image is very bright the threshold might be set to 240 in order to break the image into to. Setting the Maximum range to 170 would force that setting back down to 170 instead of 240 which would then prevent really white images from being thresholded. This global restriction can be used to prevent the auto-threshold from using a too aggressive value when working with darker or lighter images.

Example

Source Image



Thresholded



See Also

[Manual Thresholding](#)

Color Threshold

The Color Threshold module is used to remove parts of the image that fall within a specified color range. This module can be used to detect objects of consistent color values.

The interface displays the Red, Green and Blue histograms. Histograms chart the pixel value (0-255) on the X axis with the number of pixels (0-image size) having that color value on the Y axis. Using the histograms you can filter pixels with those values out of the image leaving the desired object in view.

If $R < R_{min_thres}$ or $R > R_{max_thres}$ then $R=0$

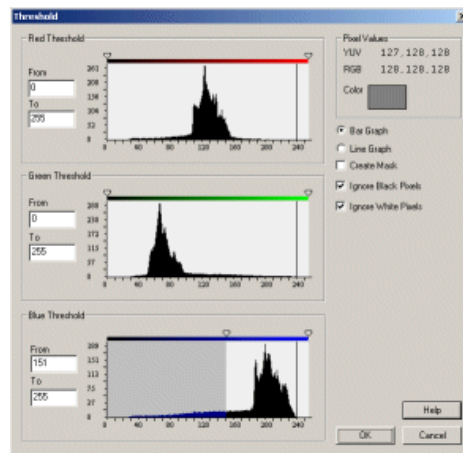
If $G < G_{min_thres}$ or $G > G_{max_thres}$ then $G=0$

If $B < B_{min_thres}$ or $B > B_{max_thres}$ then $B=0$

The Histograms are set to ignore all black and all white pixels in the chart. This allows for somewhat dark or light images to still show a useful histogram. These values can be added back in if needed but they do not affect the threshold operation.

Note that when lighting conditions change the values you used may fail to segment the image correctly. Be sure to specify as wide a threshold value as possible to account for these changes.

Interface



Instructions

1. Enter the From and To range to specify the part of the histogram you would like to preserve. You can also use the scroll triangles on top of the histograms to drag the mask to the appropriate spot. Dragging the scroll or specifying the range will change the current image to reflect the masked out places.
2. Enter the appropriate range for all Red, Green and Blue colors.
3. If you would like to see the histogram in a line format select the "Line Graph" radio button.
4. If you would like to create a mask (i.e. a white pixel) instead of preserving the current pixel values select 'Create Mask'. This will replace all non-masked pixels with a white color.
5. Often image histograms can become somewhat unusable due to excessive 0 (black) or 255 (white) pixels. To remove these two values from the histogram graph and increase the resolution of lower count pixels select the "Ignore Black Pixels" or "Ignore White Pixels". This will remove those values from the histogram to reveal better detail for other pixel counts.

Example

Source Image



Blue Threshold



See Also

[Threshold](#)
[Adaptive Threshold](#)

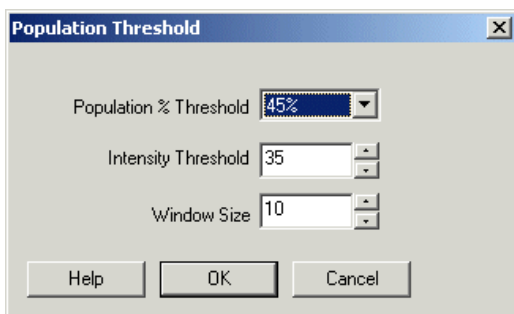
Population Threshold

The Population Threshold module will threshold an image based on the number of pixels within the specified window. Similar to the 'game of life' each pixel's neighborhood is scanned to determine if the non-black pixel lives in a neighborhood whose population is greater than some specified threshold. If it does then the pixel is set to white, otherwise it is set to black.

This procedure will join areas of high population and eradicate areas of lower population to create an image that is more global in features with much less detail. The algorithm is useful when an image contains many small blobs that need to be connected to form larger areas that better represent the overall structure of the image.

This module is useful in reducing image noise by producing a threshold that is sensitive to population boundaries.

Interface



Instructions

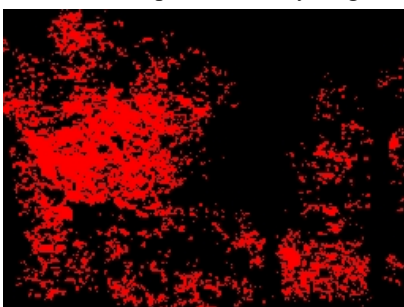
1. Population Threshold - Select the threshold used to test if a pixel lives in a populated neighborhood. Lowering the threshold will set more pixels white, increasing the threshold will cause more pixels to disappear.
2. Intensity Threshold - Select the threshold used to determine if a pixel is to be considered on or off. You can use other thresholding (like auto-threshold) to create a binary image that is then used by the population threshold.
3. Window Size - select the size of the neighborhood that will be scanned for 'on' pixels. Note that the larger the window size the more connected the resulting blob will become but the more inexact edge boundaries will become.

Example

Source



RGBFiltered to produce a noisy image



Result of Population Threshold



Variables

See Also

[Blob Size](#)

[Blob Filter](#)

Threshold

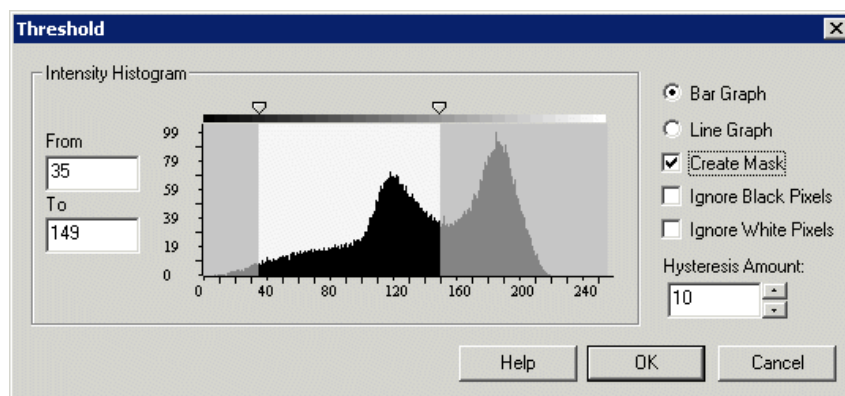
The Threshold module is used to remove parts of the image that fall within a specified intensity range. This module can be used to segment objects of consistent intensity values (i.e. remove black or white parts of the image).

The interface displays the intensity histogram. Histograms chart the pixel value (0-255) on the X axis with the number of pixels (0-image size) having that intensity value on the Y axis. Using the histograms you can filter pixels with those values out of the image leaving the desired object in view.

The Histogram is set to ignore all black and all white pixels in the chart. This allows for somewhat dark or light images to still show a useful histogram. These values can be added back in if needed but they do not affect the threshold operation.

Note that when lighting conditions change the values you used may fail to segment the image correctly. Be sure to specify as wide a threshold value as possible to account for these changes.

Interface



Instructions

1. Enter the From and To range to specify the part of the histogram you would like to preserve. You can also use the scroll triangles on top of the histograms to drag the mask to the appropriate spot. Dragging the scroll or specifying the range will change the current image to reflect the masked out places.
2. If you would like to see the histogram in a line format select the "Line Graph" radio button.
3. If you would like to create a mask (i.e. a white pixel) instead of preserving the current pixel values select 'Create Mask'. This will replace all non-masked pixels with a white color.
4. Often image histograms can become somewhat unusable due to excessive 0 (black) or 255 (white) pixels. To remove these two values from the

histogram graph and increase the resolution of lower count pixels select the "Ignore Black Pixels" or "Ignore White Pixels". This will remove those values from the histogram to reveal better detail for other pixel counts.

5. Select the Hysteresis amount. See below for a discussion on Hysteresis.

6. If you want to threshold color saturation select the appropriate radio button on top of the histogram. This will cause the module to use the color intensity as the histogram comparison and not the image intensity. This can be used as an effective way to remove non-colored objects from the image.

Hysteresis

One of the issue with thresholding based on absolute values are pixels that fall just below or above the thresholding limit. Live images from cameras (esp WebCams) will flicker a lot in terms of pixel color and intensity. This gives the image a 'live' feeling even when viewing a stationary object. When the image is thresholded large portions of the image may be just at the threshold boundary and in some frames will be in the valid part of the image and sometimes not. To reduce this problem one can widen the threshold values encompassing more of the pixels that are flickering from image to image. This, however, causes much more of the image to not be thresholded which may render the action of thresholding useless.

To solve this issue hysteresis can be used. Using a hysteresis amount will act similar to widening the threshold values BUT only for pixels connected to the current hard threshold. Thus, you can really tighten the thresholded values and then increase the hysteresis amount which will cause the object the increase in area but NOT include other parts of the image that would have been if the threshold were simply increased.

As a simple explanation think of hysteresis are widening the area of the object after thresholding is complete.

Example

Source Image

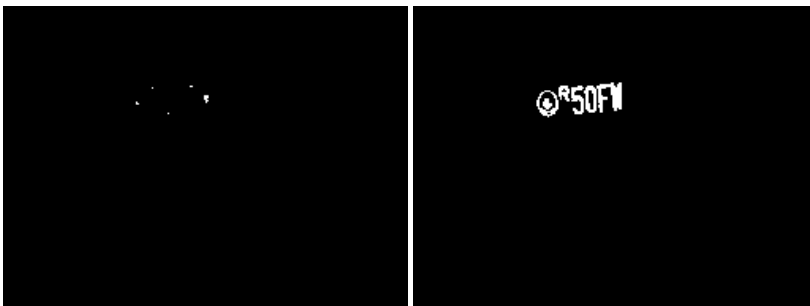
Red Filtered and Thresholded



From the example above you can see that the wood chips below the car have a lot of red components in them. If we want to remove that area from the image we have to threshold the image with a very high tight range which removes most of the license plate (see first image below). This renders the thresholding someone useless. Introducing hysteresis, however, allows us to include most of the license plate back into the image without including the wood chip areas since the license plate has the reddest parts of the image in comparison to the wood chips.

Extreme Thresholded Image (Masked)

With Hysteresis (Masked)



See Also

[Color Threshold](#)

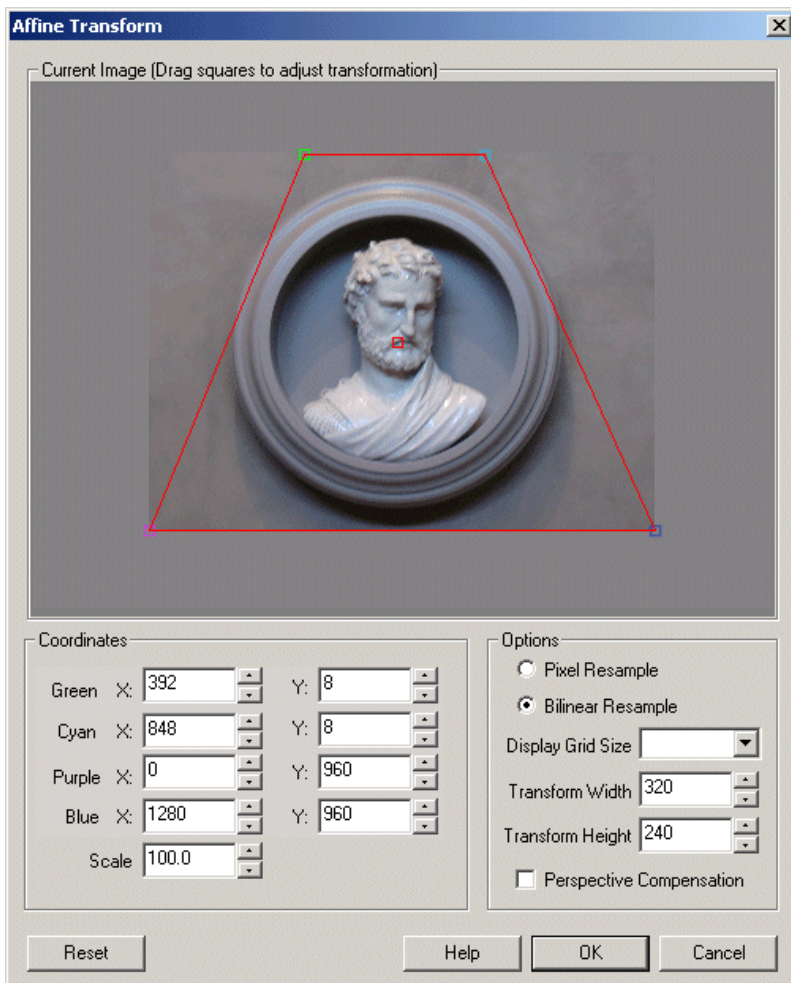
[Adaptive Threshold](#)

[RGBFilter](#)

Affine

The Affine Transform module provides a way to warp an image into a different scale, rotated and warped image. This module can be used to extract part of an image and align it to a canonical (known size and orientation) view. The affine transform will use the four selected points to create a new image that is scaled, rotated and translated based on the four coordinate points. This transform is useful for removing perspective distortion or for straightening objects within the field of view.

Interface



Instructions

1. Green. Cyan. Purple. Blue squares - drag the squares to create a new rough transform

2. Coordinates - Use the text boxes to refine the coordinates
3. Options Resample - select the interpolation mode. Pixel is fastest but causes blocky results. Bilinear is smooth but slower.
4. Options Display Grid Size - to overlay a grid used when aligning to specific angles select the grid size to use for the overlay.
5. Options Transform Width & Height - Specify the resulting size you would like the selected part of the image to be transformed into.
6. Options Perspective Compensation - If you are transforming an image which exhibits depth perspective the resulting image will have corrected most of the distortion but will not change how much a pixel represents in the real world. Because of this objects nearer to the camera will demand more pixels whilst objects further back in the scene (think of a road) will retain less. Once corrected for the affine distortion this relationship will remain. The Perspective Compensation checkbox will change this relationship and provide equal amounts of pixels for foreground and background objects. Thus transformed images will show objects in the foreground and background as being the same size.
7. Grid Size - specify a grid size that will be superimposed over the current image to help in manual alignment.

Example

Source



Affine Transform



Perspective Correction



Object Alignment



Variables

AFFINE_X1,AFFINE_Y1 - the first point used in the affine transformation
AFFINE_X2,AFFINE_Y2 - the second point used in the affine transformation
AFFINE_X3,AFFINE_Y3 - the third point used in the affine transformation
AFFINE_X4,AFFINE_Y4 - the fourth point used in the affine transformation
Note that instead of coordinates in the coordinate text boxes you can enter variable values using [variable_name] instead of a number.

See Also

[Rotate](#)
[Scale](#)
[Translate](#)

Align Image



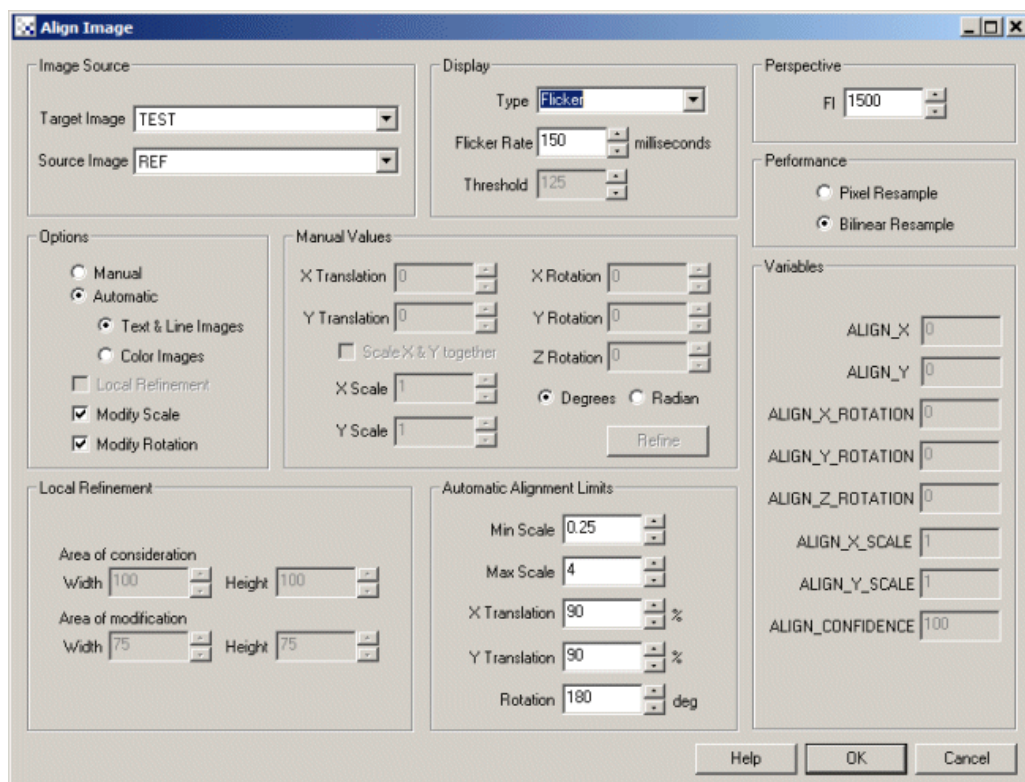
The Align Image module provides a way to align two images. Similar to the [Visual Anchor](#) and [Stabilize](#) module the Align_Image module will look for characteristic properties between two images in an effort to determine how they overlap.

Note that if too much of the target image is not included in the image to align then the system will produce erratic results as any detected alignment between the two images will be random.

This module will determine the translation (X,Y) movement, the Z plane rotation or orientation and the scale changes between the two images. Note that in order to associate the placement of one image over the next you will need some texture within the image that is the same in both in order to determine how to overlay the image.

The module was developed to allow for comparison between two images that may be somewhat misaligned such as in a security setting to determine abandoned or removed objects. It is also very useful in document comparison such as comparing differences between scanned or otherwise photographed documents. Both automated and manual modes of alignment are supported.

Interface



Instructions

1. Target Image - Select the "Target Image" in the Image Source dropdown. The target image is your model image that typically does not change. The source image will be stretched, rotated, etc. in order to make it as close as possible to the Target Image.
2. Align Image - Select the image you want to align to another image. Typically this will be the current video image from your webcam or come from a loaded image (see [Load Module](#)) for image loading.
3. Display - Select how you want the two images to be displayed relative to each other. As there are many ways to do this to highlight specific

aspects of each image, you can select which method should be used.

Aligned - Shows the Source image aligned to look like the target image.

Target - The target image to align to (i.e. a good template).

Source - The source image to align (i.e. an image rotated, skewed, etc. that is to be aligned to the target).

Flicker - Causes the two images to flicker back and forth between each other. This helps the eye to understand the movement between the two images and can help to determine alignment or rotational issues. No glasses are required for this mode.

Side by Side - Produces a single image with both images displayed side by side.

Edges - Shows the detected edges of each image. This can be useful for manual tuning as Blue and Green edges from the two images becomes yellow when aligned correctly.

Transparency - Shows the two images averaged over each other like a transparency. Also used for manual alignment.

Red Green - Target image is represented by the red channel, while the Source image is represented by the Green channel. When the intensities are the same yellow will appear. This is similar to the Edges mode but instead shows the image intensities instead of their edges.

Difference - Target image and Aligned Source image are subtracted from each other to show the differences between the two.

Wide Difference - Target image and Aligned Source image are subtracted from each other to show the differences but with a wider search area. A wide difference will cause each pixel to check its immediate neighborhood for a good match and then use that as the difference amount rather than an exact match. For example if you have a source pixel value of 156 and the corresponding target image's pixel is 14 but inside

200 66 44

156 (14) 0

40 0 0

instead of producing the result $156 - 14 = 142$ the module will instead look around the 14 value pixel for a better match and instead use $156 - 156 = 0$. Thus if the alignment is still off by about a pixel that would not matter much.

Target Difference - The Difference mode performs a subtraction between the two images and then negates any negative values back to positive to ensure that regardless of values a difference is show. The Target difference will subtract the target from the aligned source but zero out any negative values. Values below the selected Threshold will also be set to zero.

Source Difference - Functionally the same as Target Difference except that the aligned Source image is subtracted from the target image with any negative values being set to zero. Values below the selected Threshold will also be set to zero.

Red Blue - Target image is represented by the red channel, while the Source image is represented by the Blue channel. When the intensities are the same purple will appear.

Red Blue - Target image is represented by the red channel, while the Source image is represented by the Blue channel. When the intensities are the same purple will appear.

Purple Green - Target image is represented by the purple channel, while the Source image is represented by the Green channel. When the intensities are the same gray will appear otherwise the differences will appear in color.

Green Red - The inverse of the Red Green mode. (Target = Green, Aligned Source = Red, Aligned = Yellow)

Blue Red - The inverse of the Red Blue mode. (Target = Blue, Aligned Source = Red, Aligned = Purple)

Green Purple - The inverse of the Purple Green mode . (Target = Green, Aligned Source = Purple, Aligned = Gray)

4. Perspective - While Rotating in the X and Y plane (spin and flip) you can specify the amount of perspective to incorporate into the transform. The amount of perspective will change how much the image shrinks at points far away in depth and how much the image enlarges when closer in depth. A high perspective number causes things to seem very fat while a smaller number will cause more and more warping of the image at higher X or Y rotations. The amount of Perspective loosely correlates with the amount of warping seen in the image which is interpreted as depth by us.

5. Performance Pixel, Bilinear Resample - Select how precise the Source image should be transformed into the Target image. Pixel Resample refers to a nearest neighbor approach where a pixel from the Source image is transformed and placed into the nearest pixel of the transformed image. This creates a jagged effect but provides a quicker transform process. The Bilinear will interpolate the pixels position and smooth out the final image to avoid jaggies and create a more natural final transform but at the expense of reduced performance.

6. Options - Select which mode you would like to use for image Alignment. The Automatic mode seeks to find the correlation between the two images without needing any feedback from you. There are cases that may occur when you will need to adjust the values manually. In those cases

images. When selecting any alignment mode you should be aware that you may need to adjust the "Manual Values" section. In automatic mode you can select the Manual radio button which will enable all the value editing interfaces in the Manual Values section.

In automatic mode the "Text & Line Images" selection will work better with images that are mainly black and white and contain lines, drawings. The "Color Images" works better on images that are of real scenes and contain texture/gradients within the image.

Note that in order for the automatic mode to work correctly the majority of the image content needs to be shared between the two images.

7. Manual X, Y Translation - These are the values associated with the horizontal and vertical movement of the Source image to align it with the Target image. Modifying this value will move the Source image in relation to the Target image in efforts to align them better. Note that these values are in pixels.

8. Manual X,Y Scale - These are the values associated with the size change from the Source to the Target image. For example, an X or Y scale of 2 means that the Source image is enlarged to twice its size in order to align with the target image. Similarly, a value of 0.5 means the Source is reduced to half its size in order to align with the target image. One can interpret this as meaning how close or far are you from the Target image in terms of depth.

Typically images scale in the same amount for X and Y. But sometimes due to a non- perpendicular photographic plane this may not be true. If you select the Automatic mode or press the Refine button both X and Y will be optimized separately just in case. While manually editing you may find it easier for them to have the same value which can be enforced by checking the "Scale X&Y together".

9. Manual X, Y Rotation - These are known as 'out of plane' rotation in that they will spin and flip the image (X and Y Rotation) such that the image appears to exist into and out of the computer screen (ie. the Z Plane). Note that the automatic mode does NOT currently change these values.

10. Manual Z Rotation - This is also known as orientation and fixes situations where the Source and Target image are upside down with respect to each other. This parameter will twist the Source image until it better aligns with the Target image. If you are more comfortable in working with Radians as apposed to degrees you can select the Radian radio button to switch between Degrees and Radians.

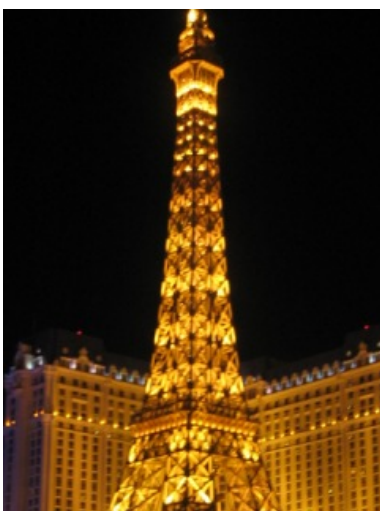
11. Manual Refine - The refine button will execute an automated alignment just once on the image. This is helpful when using the manual mode of image alignment such that when you get things close you can try the Refine button in order to tweak the final values. Note that the alignment should be close in order for the Refine button to work. If the images are far apart in terms of alignment then the Refine button may not find a match or make things even worse.

12. Local Refinement - Currently disabled but will be used to refine alignment locally (as apposed to globally) where the image is divided into smaller sections and aligned further. This help to align local issues such as skewed or crumpled images. If you have such requirements please [contact us](#) about this functionality.

13. Automatic Alignment Limits - Limits the range that the automatic method will allow when testing alignments. This ensures for really bad alignments that the produced values are still within a known range and don't cause the aligned image to disappear from the result completely. Limiting the allowed transform ranges will ensure that bad matches will show bad results as apposed to no results at all. Note that the Translation is specified as a percentage of the source image. A value of 50% means that the system can allow a translation of 1/2 the size of the image in order to find a best match. Likewise, if you know that images will never be up-side-down with respect to each other you can probably reduce the Rotation to 45deg which will speed up alignment.

Example

Source



Target



Aligned Image





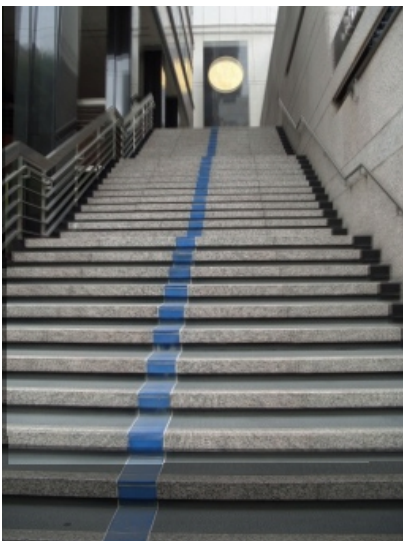
[Download](#) the zip file containing the above two images and the robofile used to generate the aligned image.

Source

Target



Aligned Image



[Download](#) the zip file containing the above two images and the robofile used to generate the aligned image.

Variables

ALIGN_X - the X translation that aligns the image to the target.

ALIGN_Y - the Y translation that aligns the image to the target.

ALIGN_X_ROTATION - the X or spin angle that aligns the image to the target (degrees)

ALIGN_Y_ROTATION - the Y or flip angle that aligns the image to the target (degrees)

ALIGN_Z_ROTATION - the Z or rotation angle that aligns the image to the target (degrees)

ALIGN_X_SCALE - the X or width scale factor that relates the image to the target

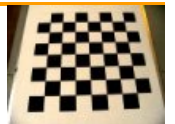
ALIGN_Y_SCALE - the Y or height scale factor that relates the image to the target

ALIGN_CONFIDENCE - a percent confidence of how well the Source image matches the Target image after transformation. 100% is very confident, 0% is not.

See Also

[Visual Anchor](#)
[Stabilize](#)

Automatic Image Calibration



The Automatic Image Calibration module provides an easy way to quickly calibrate an image view based on a rectangle calibration grid. The module expects a rectangular checkered grid of any size to be visible in the image. Once the grid is detected the module will warp the current image such that the grid will become square within the image. Namely, each checker box will be the same number of pixels regardless of rotation from the camera imaging plane. This ensures that objects within the calibration plane will be the same relative size. This can also be used to independently calibrate two separate images to be combined into a single larger image.

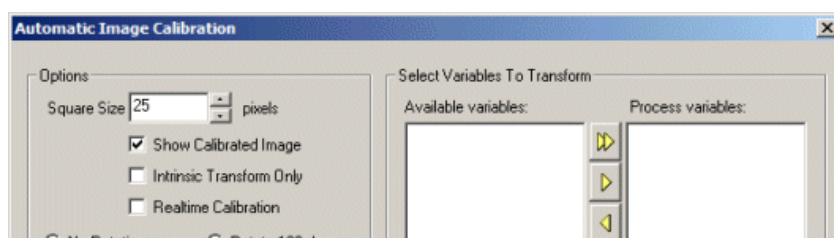
Once calibrated, the checker board can be removed and the transform will remain such that any objects on the same plane will be transformed in the same way as the grid was.

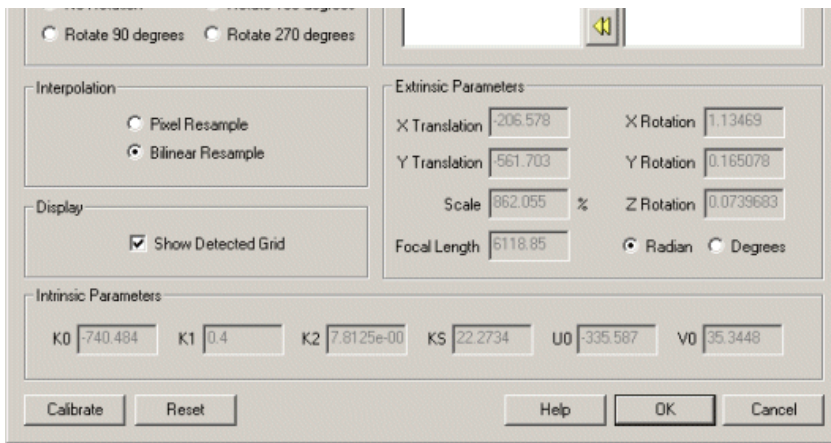
If you select the "View Grid" checkbox you should see a green overlay on the recognized grid. By pressing the calibrate button the image will be automatically warped such that the checker grid will be square with respect to the image plane. This warping process will remove perspective, scale, and lens distortions that can cause problems when measurements or recognition tasks need to be performed on surfaces not aligned to the imaging plane.

This module has two functions, one to determine calibration parameters based on a checker board calibration grid and then to warp subsequent images to that setting. Thus the intended use is to first calibrate your camera using a calibration grid and then have all subsequent images be transformed in the same way as the calibration board dictated.

If you need a grid to print out you can use [this one](#).

Interface





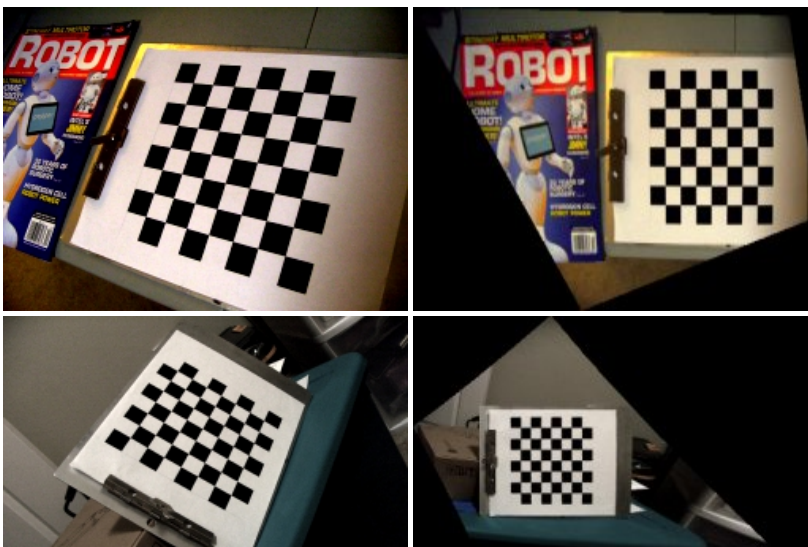
Instructions

1. Square Size - This setting indirectly affects the size of the image. The specific property refers to how large one of the checkered squares should be made in the resulting image. By choosing a convenient number you can also ensure a simple real world to pixel coordinate conversion. For example, if your square is 1 inch long and you set the square size to 20 pixels then each pixel will represent 0.05" in real world. Taking any measurement in pixels from the image thereafter can be multiplied with 0.05 (1/20) to get real world inches.
2. Intrinsic Transform Only - If you are just interested in removing lens distortion properties of the image you can select that only intrinsic transformation be applied to the image. This will retain scale and perspective but remove the pincushion or barrel warping of the image.
3. Realtime Calibration - If you are not sure on how the final image looks with regards to a particular orientation of the calibration grid you can select the Realtime Calibration which will calibrate the image to the visible checker grid as-fast-as-possible to allow you to settle on an appropriate view. Note that this will create a very slow frame rate as the calibration process will not permit a very high frame rate. Once you are happy with the image you can unselect this checkbox and the frame rate will dramatically increase but still warp the image as appropriate.
4. No Rotation, Rotate 90, 180, 270 Degrees - Since the calibration grid does not have a obvious orientation there are cases where when calibrated the image will be incorrectly oriented with respect to this 90 degree ambiguity. You can select the appropriate rotation to correct for this ambiguity.
5. Pixel/Bilinear Resample - If you need quicker but a less precise/smooth transform you can select "Pixel Resample" which will perform the resulting image transformation quicker but with a blockier feel.
6. Show Detected Grid - Select if you are not sure if the calibration grid you are holding is being detected by the module. Keep in mind that reasonable lighting and size are expected in order to calibrate correctly. Any image containing a grid whose squares are less than 20 pixels in size in the image will probably NOT be detected.
7. Extrinsic Parameters - Those values calculated given the calibration grid that will warp the image to align the calibration grid with the imaging plane. Note that you can chose to view the rotation as degrees or radians.
8. Intrinsic Parameters - Those values that best describe the lens distortion present in the calibration image. These parameters describe the pincushion or barrel warping obvious in less expensive len's but may also be subtly present in any camera.

Example

Source

Calibrated Image





Notes

For best results

- Use as large a calibration grid as you can to cover most of the image. Smaller grids will not capture the lens distortion parameters well enough as they are mainly pronounced at the image edges.
- Avoid using grids that are out of alignment by more than 45 degrees as that will start to distort the squares too much for an accurate rectification.
- Ensure that the grid is a matte (non-reflective) surface (like printed blank ink) otherwise the square detection will not work very well.
- If you have detection issues enable the grid display and move the grid around a little until you see the green overlay.
- Ensure that each square is at least 20+ pixels big otherwise they will not be detected. Try moving the grid closer to the camera to see if that helps with detection.
- Avoid any shadows that would break the grid into shaded and non-shaded parts.
- Ensure that there is sufficient contrast (white is much whiter than black) in order to get the most precise position information.
- Keep the calibration grid as flat as possible. If you print something out paste it onto a more solid surface to ensure it remains planar. If you happen to have a chess set lying around the board makes for a good calibration grid!

Note that there is no manual mode for this module (it is really just too laborious a process!). If you have problems in detection try moving it around a bit to change the angles which will eventually favor detection.

Also note that the grid size is auto-adjusted for. You can use a 12x12 or a 6x18 grid without changing any parameters.

Variables

AUTO_CALIBRATION_ERROR - The average pixel distance error between the image grid and the matched warped grid based on the calculated values

AUTO_CALIBRATION_X_TRANSLATION - The horizontal translation from center of screen the detected grid is at

AUTO_CALIBRATION_Y_TRANSLATION - The vertical translation from center of screen the detected grid is at

AUTO_CALIBRATION_SCALE - The size difference between the desired square size and the actual detected square size

AUTO_CALIBRATION_PERSPECTIVE - The focal length used to unwarp the detected grid to bring it back to the image plane

AUTO_CALIBRATION_X_ROTATION - The radian amount of pan rotation to bring the grid back into the image plane

AUTO_CALIBRATION_Y_ROTATION - The radian amount of tilt rotation to bring the grid back into the image plane

AUTO_CALIBRATION_Z_ROTATION - The radian amount of in plane (orientation) rotation to bring the grid back into vertical/horizontal alignment

AUTO_CALIBRATION_K0 - The K0 camera property used to invalidate len's distortion

AUTO_CALIBRATION_K1 - The K1 (multiplied by radius) camera property used to invalidate the lens distortion

AUTO_CALIBRATION_K2 - The K2 (multiplied by radius²) camera property used to invalidate lens distortion

AUTO_CALIBRATION_KS - The K scale/zoom factor used to invalidate len's distortion

AUTO_CALIBRATION_U0 - The X principal location (Cx) of lens distortion center

with respect to center screen

AUTO_CALIBRATION_V0 - The Y principal location (Cy) of lens distortion center

with respect to center screen

See Also

[Radial Calibration](#)

[Perspective Calibration](#)

[Transform Image](#)

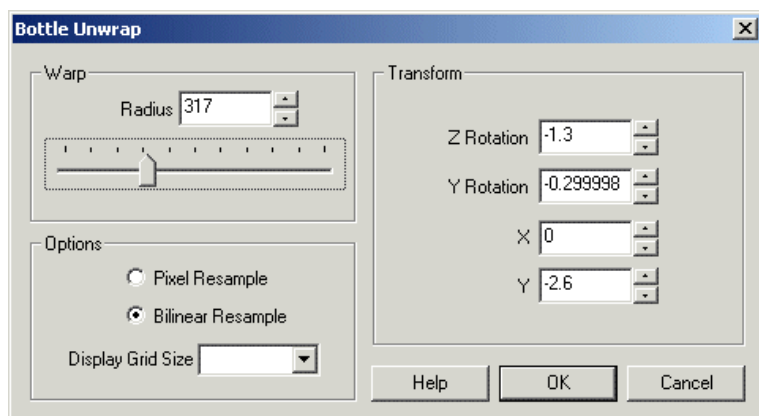
[Affine Warp](#)

[Bottle Unwrap](#)

Bottle Unwrap

The Bottle Unwrap module transforms an image that is wrapped around a bottle surface into a flat surface. This module is often used as a precursor to an OCR module in order to correct for curve distortion when imaging a curved surface. For example, the label placed on a cylindrical container can be unwrapped in order to recognize the words, images, etc. on that surface in an easier planar way.

Interface



Instructions

1. Warp - Adjust the radius of the wrap such that the image appears to flatten out. Numbers larger than the image width will shrink or compress the image whilst numbers smaller than the current image width will spread out the image by expanding the image extremes.
4. X - If the label in the image is not in the image center use the X transform to move the transform center into the center of the label. This will ensure that the image unwraps from the correct center point. If your object is not in the image center and the unwrap is applied you will notice more distortion on one side of the object. Move the X center until the image appears equally transformed on the left and right sides.
5. Y - The Y transform will shift the image center up or down in order to center the image but it does not affect the unwarping.
6. Pinch - Most images will require some adjustments in order to straighten horizontal lines after warping. This value increases with the camera's field of view.
7. Z Rotation - If the image is slightly tilted you might want to rotate the image into a more horizontal/vertical orientation. This will ensure that the bottle unwrap will operate on an aligned image.
8. Y Rotation - If the image is tilted forward or backwards (as if the camera is pointed up or down towards the label) you might need to adjust the Y Rotation which will rotate the image accordingly.
9. Pixel Resample - specifies that the image will be transformed on a per pixel basis. This is a faster technique and will make the image appear more crisp but can suffer from being too crisp in that edges may appear too sharp.
10. Bilinear Resample - provides smoother results than the Pixel method but is a slower transform
11. Display Grid Size - the goal of this module is to align the image into a flat rectilinear surface. In order to help you tweak the parameters you can enable the Display Grid size to superimpose a grid structure on top of the image in order to better compare image lines with actual horizontal and

vertical lines. Using the grid as a comparison you can change the values of the transforms in order to achieve the alignment goal.

Example

[Click here](#) to load a robofile that contains the example transform below. Note that this example includes the image below and also uses the Radial Distortion module to undistort radial effects due to the camera lens.

Source

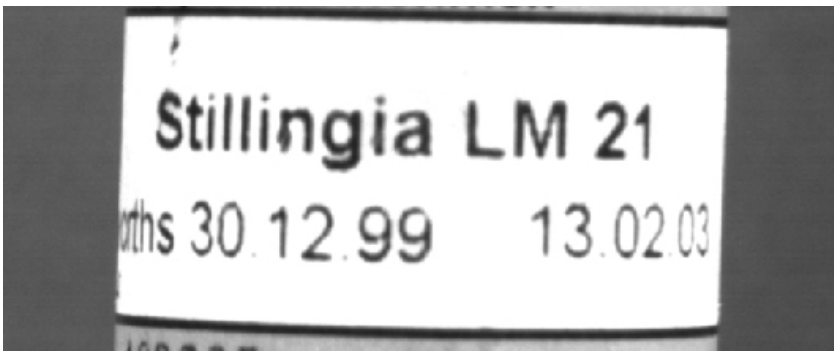


Transformed



Source

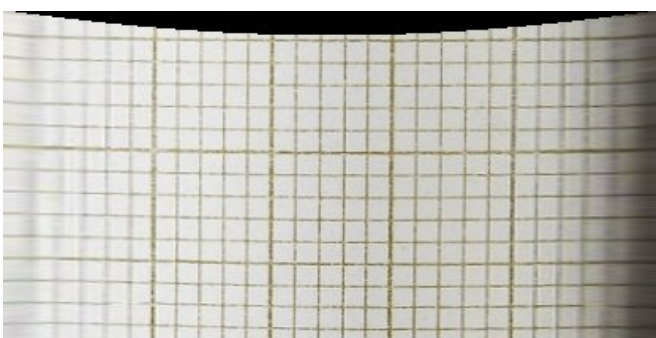
zum Einnehmen



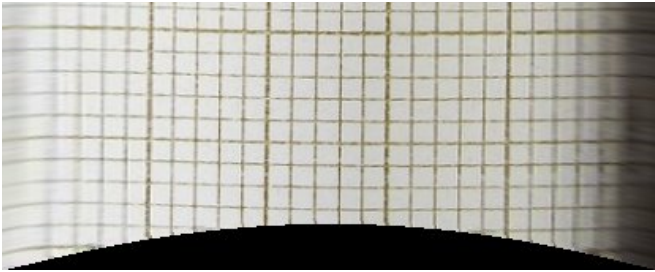
Transformed



Source



Transformed



The unwarping process will straighten horizontal lines and allow text characters to occupy the appropriate space. Note in the last example, how the corrected transform appears like a standard grid. This is a good way to calibrate the system to ensure that you have the optimal configuration for your situation.

See Also

[Radial Distortion](#)

Crop

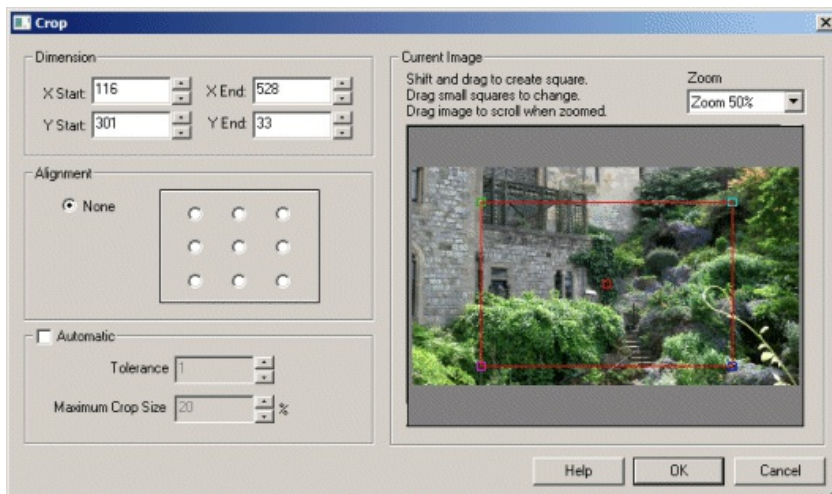
Crop allows you to remove parts of the image that are not of interest. Often due to NTSC conversion camera's have spurious beginning or ending lines that are not colored correctly. To remove those erroneous lines from the processing pipeline use the crop function to cut those parts of the image out. For example, if the top line of the image is bad in a 320x240 image use

```
X Start 0
Y Start 0
X End 320
Y End -1
```

Note that negative or zero numbers in the "End" coordinates will cause the current width or height to be added before cropping to that number. This allows the crop module to crop relative to an images size even if that size changes.

If you would like to use variables to control the crop parameters simply type in the variable name surrounded by [] as in a [variable] [expression](#). For example, if you enter in [image_count] into one of the provided text box the image will be cropped according to the image count. I.e. as the image is updated image_count is updated which will cause the cropping to change. See [Set Variable](#) to create new variables.

Interface



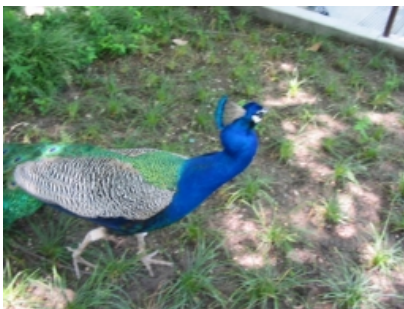
Instructions

1. Dimensions - Specify the crop dimensions by either typing in the appropriate number or use the up/down buttons to adjust the numbers. Note that numbers larger than the current image size are ignored and do not contribute to cropping. Note that you can use the [variable] [expression](#) format in each of the coordinate boxes in order to programmatically change the values.
2. Resize canvas - Normally the crop will resize the current image to include only those cropped areas. As this does change coordinates within the image it is sometimes desirable to keep the image dimensions and just black out the non-active areas. When this checkbox is selected the image retains its dimensions and blacks out the rest of the image.

3. Alignment - Given the crop coordinates specify the alignment bias of the crop. This allows you to move the crop window relative to different sides of the image. For example, if you wanted the right 100 pixels of the image regardless of image size you would specify xstart:0, ystart:0, xend:100, yend:0 and any of the right sided radio buttons under alignment. Note that a ending coordiante of 0 implies the width or height of the image (it is assumed a size of 0 would not be desired).
4. Automatic - Select this checkbox to turn on automatic cropping. Images often taken as screen shots, scanned images or images from devices with extra borders around them can be automatically cropped by detecting the box around the image in which to crop to. The Automatic option causes the Crop module to automatically detect this box and crop the image to those dimensions. Note that autocrop assumes that the image is surrounded by perfectly horizontal/vertical lines having been introduced by GUI interfaces or are digitally introduced.
5. Automatic Tolerance - When searching for the border lines you can increase the tolerance which will allow for pixels in vertical and horizontal lines that are slightly different in color to be considered a continuous line.
6. Automatic Maximum Crop Size - Sometimes the autocrop routine will detect a much smaller box within the image and crop too much out of the image. Specifying a low crop percentage will ensure that the image is not cropped too much. I.e. it is assumed that the crop coordinates will be more towards the border of the image rather than in the center.

Example

Source



Cropped



Variables

CROP_X_START - holds the x start (left) of the crop transform used

CROP_Y_START - holds the y start (bottom) of the crop transform used

CROP_X_END - holds the x end (right) of the crop transform used

CROP_Y_END - holds the y end (top) of the crop transform used

See Also

[Scale](#)

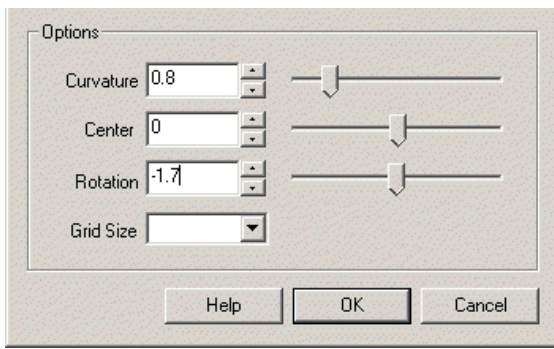
Cylindrical Unwarp

The Cylindrical Unwarp module performs an unwarping operation on an image in order to remove warping on the sides of an image. This warping is caused by lens distortion that causes objects towards the edges of an image to become stretched. This unwarping operation is needed for panoramic stitching or for consistent measurements of objects near the edges of images.

The module is used in cases of panning the camera should match with previous images during the pan. Note that this module only affects movement in the horizontal direction and does NOT work with vertical movements. If you need vertical warp see the [Spherical Unwarp](#) module.

Interface



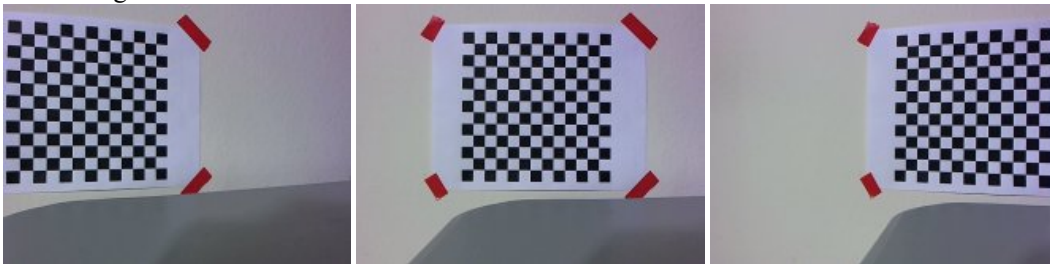


Instructions

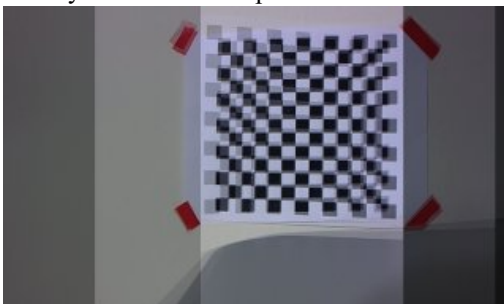
1. Curvature - Specifies how much curvature to introduce to the image left and right edge in order to correct for stretching of images due to lens distortions. Use the slider or up/down buttons to warp the image into correcting edge stretching. By ensuring that horizontal lines in the image remain horizontal towards the edges will help to correct for this form of distortion.
2. Center - The horizontal center of the transform. This helps to correct distortions that are off-center from the middle of the image. Often the lens is not perfectly centered on the CCD which would cause the distortion to be off-center. By correctly aligning the transform the edge warping will shift and will become more consistent.
3. Rotation - Corrects for rotation of the CCD such that panning motion causes horizontal motion to remain horizontal. Note that the module is configured for -45 to 45 degree rotation as most CCDs are only misaligned by a few degrees.
4. Grid Size - To help you view how the warping adjustments affect alignment you can use the grid to better understand what numbers to use.
5. Crop - To remove the black borders around the image you can select the Crop checkbox. This will crop the image to the point that no black edges exist. Note that this changes the image size and removes pixel data that is otherwise correct.

Example

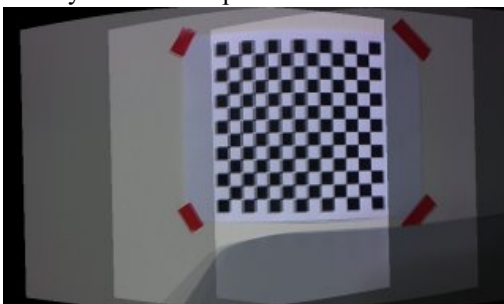
Source Images



Overlaid **without** Unwarp



Overlaid **with** Unwarp



See Also

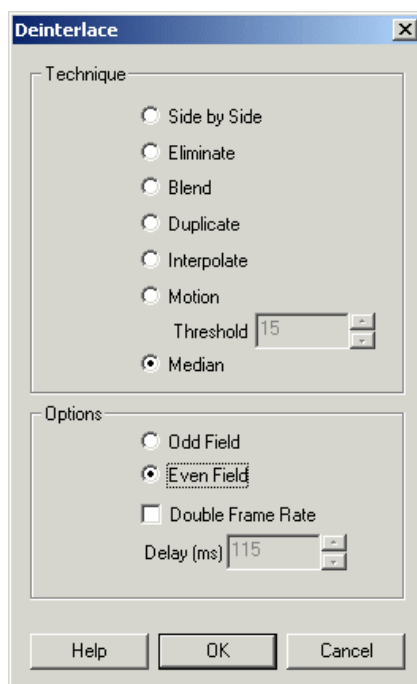
[Bottle Unwrap](#)
[Radial Distortion](#)
[Fisheye Transform](#)
[Polar Transform](#)

Deinterlace

The Deinterlace module will un-interlace an image produced by a camera or video source that has weaved the even and odd rows of two images together. This weaving was done to reduce the overall bandwidth required for the transmission of images but causes "jaggies" in images when reproduced on a computer for analysis. Note that this only valid for moving images as an interlaced image is in fact two images spliced together. If you are using such a video source for a static scene or one where the camera does not move then interlacing will not be an issue as the current and next frame will be so similar that the interlacing will not be noticeable.

For those video sources that are filming rapid movements the deinterface module will help to reduce or eliminate those artifacts. Due to the interlacing being two images, each image only has half the number of rows that it normally does. This causes resolution issues when trying to compensate for interlacing by separating them. The missing row in each image needs to be artificially created in order for a "reasonable" approximation of the original frame to be created. Note that if you have the possibility to not use an interlaced video source we recommend you attempt that route as the resolution and information content of those images are always superior to their interlaced version (about 2x).

Interface



Instructions

1. Technique - Select the Appropriate Technique to use to eliminate the jaggies.

Eliminate - Removes the even or odd rows of the image. In effect this is ignoring one of the frames.

Side by Side - show the two deinterlaced frames next to each other.

Blend - blurs the two frames into a less jaggied version. Note this will not effect the frame rate and in effect causes a frame to be lost. This does have the nice effect of causing motion blur where motion is present.

Duplicate - simply duplicates the even or odd row to create the missing row in each image. This is the fastest way to separate each image and double the frame rate.

Interpolate - Instead of just copying one row to the next the interpolation technique will smooth the transition from one actual row to the next by creating the new row as an interpolation (average) of the previous and next row. This causes smoother row transitions than the duplicate technique but with the addition of some blurring.

Motion - If one row (even) is similar enough to the next interlaced row (odd) then its value is used. Otherwise if the next row is above the difference threshold then the interpolated value of the two rows (even+even2) is used. This has the advantage of providing full resolution on parts of the image that are not in motion or do not include a moving object.

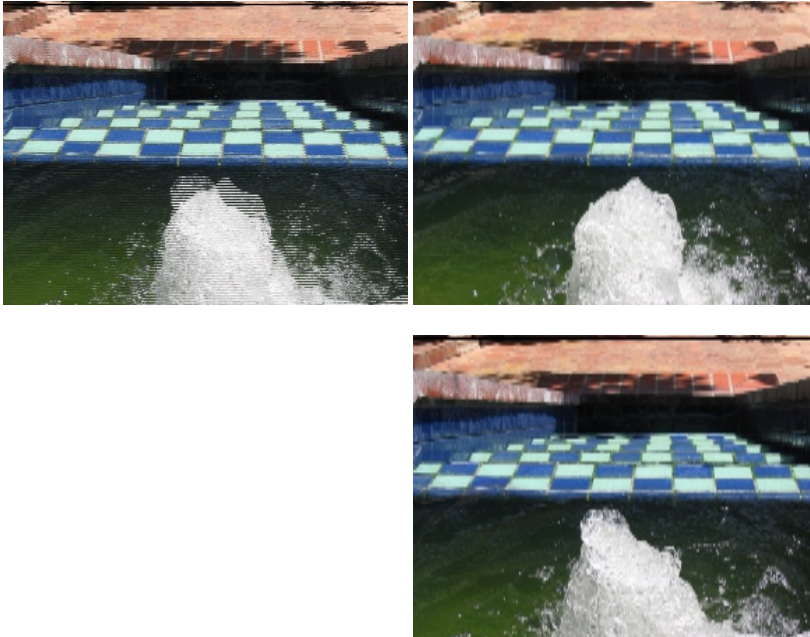
Median - The median value of the interpolated pixel, the current pixel (even) and the next row (odd) is used.

2. Options - Select which row field you want to use. Odd specifies that only the odd row is deinterlaced. Even specifies that the even row is used. Double Frame rate means that two frames are extracted from a single interlaced image. Each frame is then added as a new frame into the pipeline. Thus you can increase the frame rate of the video by selecting this option. Please note that while the frame rate will be increased the actual image content is 50% of what reality is.

Example

Source

Deinterlaced



FFT - Fast Fourier Transform

The FFT module provides a way to transform the current image from spatial (x,y intensity) space into frequency space. The FFT is one of the cornerstone routines use in signal processing as it can be used to eliminated repetitive signals from the source data. The FFT module will decompose an image into its fundamental intensity frequencies that can be filtered and recombined to create a new image.

The results of a Fourier Transform are two data channels. The first channel that contains the intensity or real numbers is often referred to as the power spectrum. This is the channel that can be processed to remove certain unwanted artifacts from an image. The second channel is called the phase channel. This channel contains the imaginary or phase information resulting from the Fourier transform and is only needed if a reverse transform is desired. Typically this channel is not processed and simply passed through unchanged to the inverse transform.

The main use of the FFT in image processing is for the removal of repetitive noise from an image. This is accomplished by transforming the image into the frequency space using the FFT module and then masking out the points within the power spectrum that reflect the presence of a particular signal. Once transformed into the frequency space it is easy to see points or lines in the power spectrum that are created due to sinusoidal noise in the original image.

Combined with the arithmetic module you can create an image mask (using an external image paint program) that can be used to eliminate certain hotspots or lines from the power spectrum and then perform an inverse FFT to regain the original image.

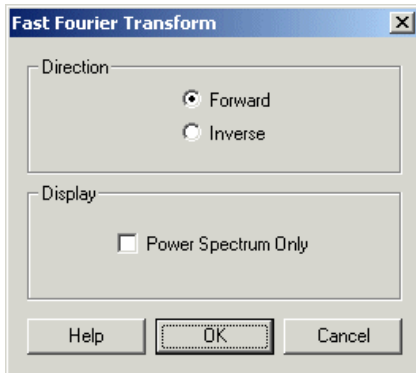
Note that the FFT module produces the power and phase in different channels. After the FFT module has run the RED channel will contain the power spectrum whilst the green channel contains the phase (or imaginary) information. Both channels are required for if an inverse transform is performed. Note that doing a forward transform, followed by an inverse transform will result in the original image. To view the power spectrum you would need to add in the RGB_Channel module and filter just for Red.

Once the power spectrum has been processed you can use the Combine_RGB module to combine color channels back into a single image to view the results.

The FFT module operates in grayscale only and requires a power of 2 image (width and height are 128, 256, 512, etc.) to process the image. You

The FFT module operates in grayscale only and requires a power of 2 image (width and height are 128, 256, 512, etc.) to process the image. You can add back in color information from the original image if desired (see example below) by separating RGB into a HLS color space and replacing the L space with the FFT power spectrum.

Interface



Instructions

1. Select if you want to perform a forward or inverse FFT transform.
2. If you do not need the phase information you can select the "Power Spectrum Only" checkbox which will remove the phase information from the computed image. Note that the phase information is required if an inverse transform is eventually desired.

Example

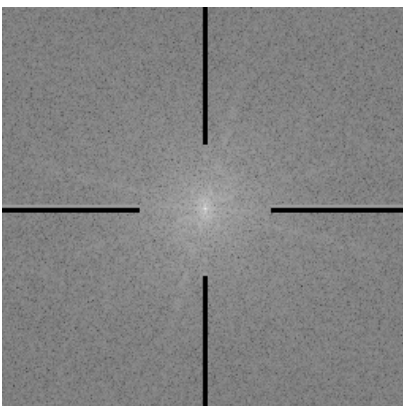
Source



Processed FFT



Power Spectrum with Mask



Note that horizontal and vertical striations in the source image above. These frequencies correspond to a horizontal and vertical lines in the power spectrum. Using the above mask removes those frequencies which when inverted to RGB space no longer exist. The full RGB color information is acquired by converting the image to HLS color space and then substituting in the inverse FFT into the 'L' channel. The resulting image is the converted back to RGB color space to produce the final image.

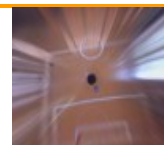
Example

[Click here](#) to load a configuration that uses the FFT module to eliminate horizontal and vertical noise signals as seen above. You will also need to [download](#) the mask image as seen below.

See Also

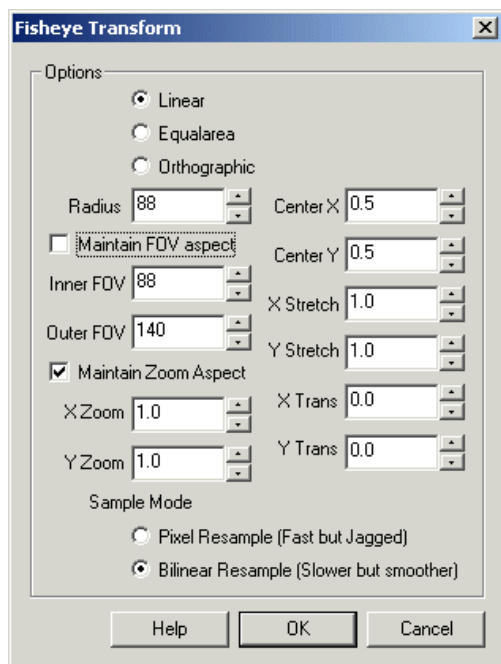
[Math](#)
[Combine RGB](#)
[Load Image](#)

Fisheye Transform



The Fisheye Transform performs an unwarping of an image in order to produce a more usable image with regards to lines and distortion effects produced by wide angles lens. Similar to the [Radial Distortion](#) and [Spherical Transform](#) this module assumes a circle image taken from a wide angle lens and provides you with many image warping controls in order to rectify the image. Note that this module is functionally the same as the [Radial Distortion](#) but offers more intuitive controls for manipulating the image.

Interface



Instructions

1. Linear, Equalarea, Orthographic - Select what type of lens you are using in order to unwarp the image. If you are not sure which type you have simply try each and see which provides the best results.
2. Radius - Specify the radius of the circle within the image. The objective here is to increase or decrease this value until the edges of the image become as straight as possible. Either changing the number directly or by clicking on the up and down arrows you will be able to interactively explore which number works best.
3. Inner and Outer FOV - The FOV parameters define the amount of curvature the lens creates. The inner FOV will warp the inner most part of the image towards the center of the image (to undo the enlargement of objects in the image center) whilst the Outer FOV will increase objects size in the outer parts of the image. Together they provide an un-warping effect due to the wide angle lens.
4. Zoom X,Y - If the transform creates a resulting image that is larger than the current image use the Zoom number to reduce (or increase) the size of the image such that you can see as much of the valid image data within the current image viewing area. Note that you can uncheck the "Maintain

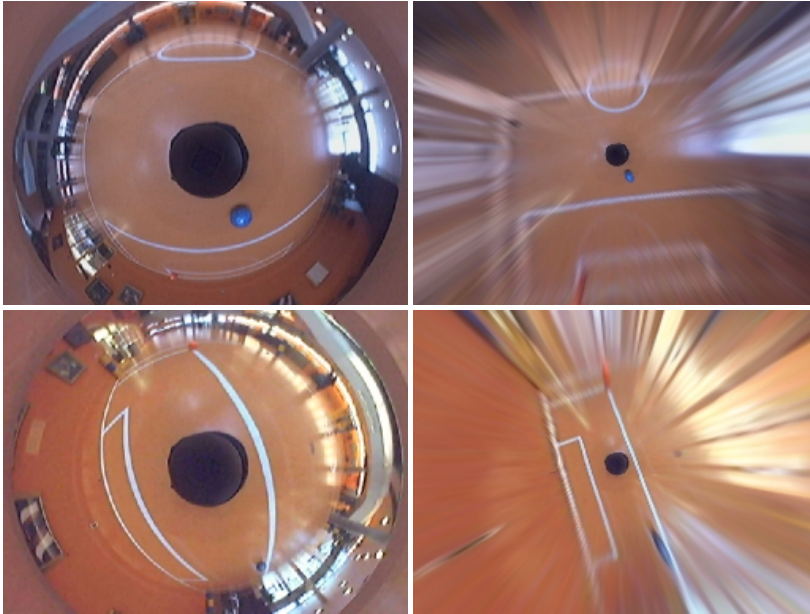
Zoom Aspect" to zoom the X and Y directions independently. This may be needed if your lens is slightly oval rather than a perfect circle.

5. Center X,Y - Use the Center values to move the center of the sphere to the right location. If your resulting transform appears to be lopsided try increasing/decreasing the center values to find the exact middle of the lens within the image.
6. Stretch X,Y - To stretch the image in the X and/or Y direction change the numbers accordingly. Note that this does not affect the transform in the way that the previous numbers do. Stretching simply resizes the final image transform to other dimension and adjusts the pixels accordingly.
7. Trans X, Y - If your resulting transform is not quite centered in the viewable image area use the Trans X,Y to shift the image in the appropriate direction to maximize the image area taken by the transform.

Example

Source

Unwarped



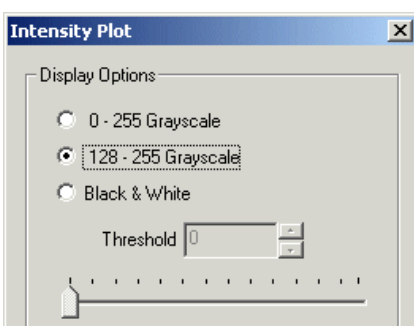
See Also

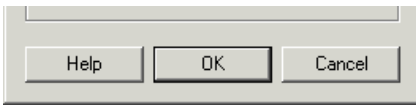
- [Radial Transform](#)
- [Spherical Transform](#)
- [Polar Transform](#)

Intensity Plot

The Intensity Plot module provides a way to view the intensity profile of an image based on the values in a vertical column. This is different than a [histogram](#) transformation of the image in that the resulting image is of the same width as the original image but with a height of 256 to represent the 256 intensity values of each column. To generate the image each column of the original image is converted to grayscale and the resulting value for each pixel is recorded. Once the column has been processed the chart is generated with each resulting pixel's intensity relating to the amount of original pixels that are at that intensity value (0-255).

Interface





Instructions

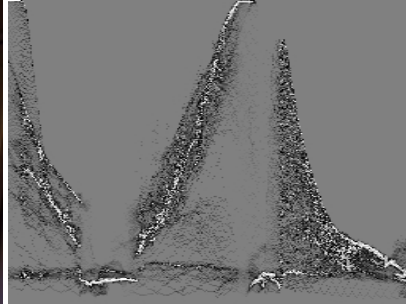
1. 0 - 255 Grayscale - This display shows the number of pixels per vertical row scaled from 0 to 255
2. 128 - 255 Grayscale - This display shows the number of pixels per vertical row scaled from 128 to 255
3. Black & White - This display shows pixels that exceed the threshold count as white.

Example

Source



Intensity Plot



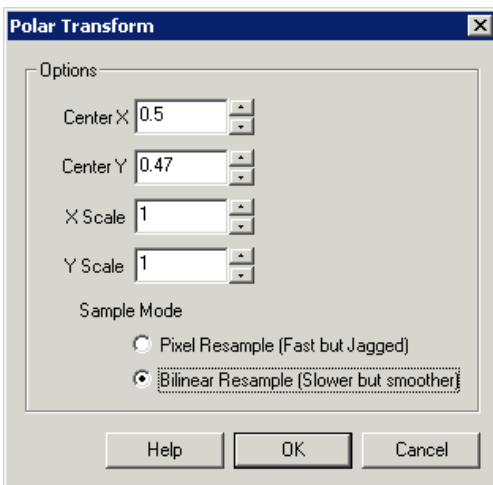
See Also

- [Histogram](#)
- [Surface Plot](#)

Polar Transform

The Polar Transform module will unwrap an omnidirectional parabolic image such that the resulting image is a vertical interpretation of the original radial image. This can be useful to transform the image into a more human understandable form.

Interface



Instructions

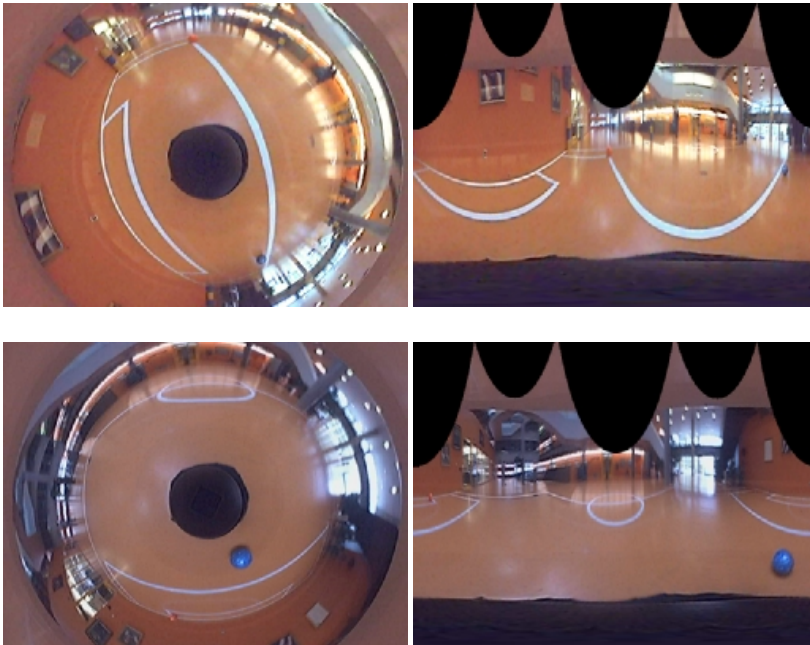
1. Center X,Y - Adjust the center coordinate of the image. If the bottom edge of the image seems curved adjust the Center X and Y coordinates to move the center of the image to the image center. This is used to compensate for a parabolic image where the mirror and camera are not perfectly aligned.
2. Scale X, Y - Adjust the scale of the image to change the size and aspect ratio of the transformed image to the desired dimensions.

3. Sample Mode - Specify if the transformed image should use nearest neighbor pixel selection or be created using interpolation. Using interpolation produces a nicer smoother image but at the cost of some computational speed. The nearest neighbor image is faster but is prone to more jagged edges.

Example

Source

Polar Transform



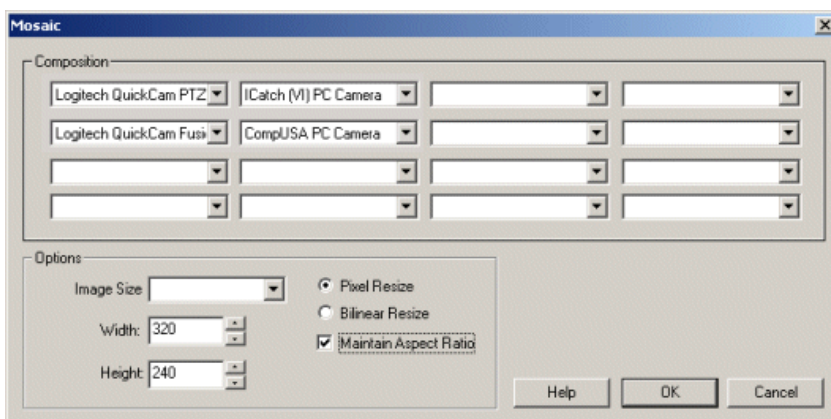
See Also

[Radial Distortion](#)

Mosaic

The Mosaic module demonstrates the multi-camera functionality of RoboRealm in that each camera can be configured to occupy a certain block within an image mosaic. Using the Mosaic module you can arrange each camera in a lightbox type arrangement of your choosing.

Interface



Instructions

1. Composition - Select the appropriate image to display in the corresponding block within the composed mosaic image. The default selections are

Source - the source image that was initially loaded into RoboRealm

Current - the currently processed image within RoboRealm

CameraX - a list of attached and active USB camera devices

MarkerX - a list of created marker images using the [Marker](#) module. The Marker labels represent images at the time markers were created. If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

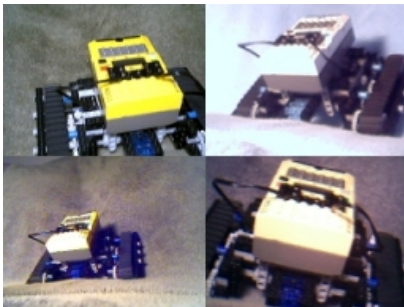
2. Image Size - Select the size of each of the images to be used in the mosaic image. This will auto-populate the width and height sizes below.

3. Resize - Select the scaling technique that is to be used. Pixel resize is faster and is sharper than other techniques but can cause jagged edges on high contrast lines. Bilinear resize will use linear interpolation to determine the actual resulting pixel values. This technique is slower but will create smoother more pleasing results.

4. Aspect Ratio - While editing either the pixel size or percent size the module will adjust the width and height to keep the same aspect ratio. To remove this restriction uncheck the "Maintain Aspect Ratio" checkbox.

Example

4 camera view



2 camera view



The above example shows 4 cameras looking at the Lego Mindstorms RCX robot. Note the color and quality difference between the images. The second image shows a larger two camera view of the same scene (scaled down for web viewing).

Note - multi-camera view is ONLY available for DirectX compatible devices. VFW does not generically support multiple camera devices and is therefore not compatible with using more than one camera at a time.

Speed considerations

For large images you may want to ensure that the resulting image size per composite block is a factor of 2 of the original image. Scaling by a factor of 2 will result in better performance and higher quality results (i.e. for factors of 2 pixel and bilinear modes produce the same results).

See Also

[Display Image](#)

[Align Image](#)

Optical Flow

The Optical Flow module is used to determine the movement of objects from the current image with respect to the last image. Note that this means more than one image is needed in order to use Optical Flow.

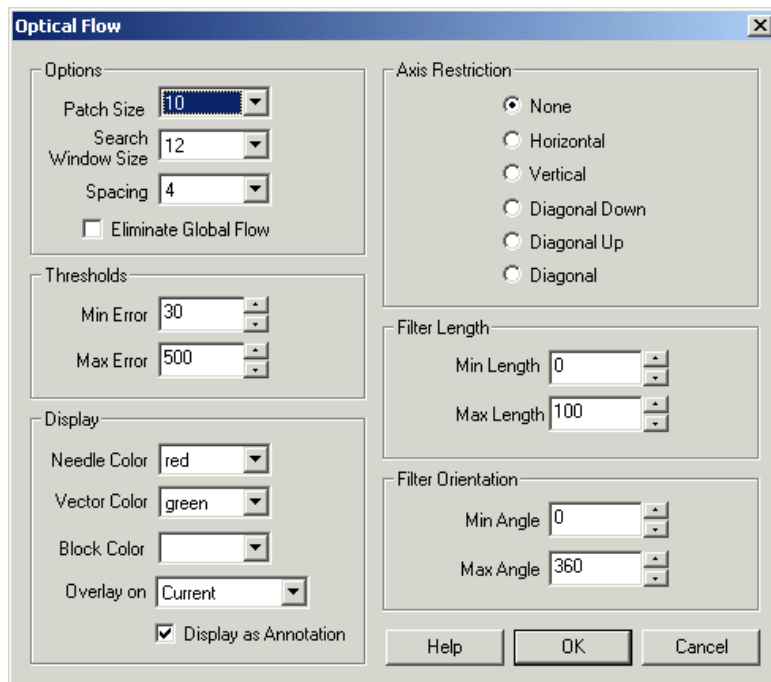
The Optical Flow of an image is defined as how parts of the image (in this case square patches) move with respect to the previous image. Each part of the image is broken into square patches and searched for the best match within the previous image. Once the best match has been identified a connection between where that patch current is and where it was is draw as a line. These lines are often referred to as needles due to their appearance when viewed in a moving image.

While very powerful the optical flow technique does suffer from some fundamental issues. One such issue is the nature of the matching that is performed on each patch. If there is not enough texture or pixel intensity changes within a given patch that patch could potentially match many patches in the previous image. Thus if a patch would match with too many previous patches the match would be very unstable, not contribute much to the overall flow and therefore is eliminated from the result. This eliminations causes large flat planar areas (such as in the middle of a white piece of paper) to not indicate optical flow. It becomes apparent that optical flow really likes edges and corners as the matching of those image features are more often unique than other parts of the image.

Optical flow can be used to calculate the overall image movement. At times this is needed as often the camera is stationary with a moving object in

view. But in the case of a moving camera the overall global flow will "hide" objects moving against the motion of the camera. The issue is to eliminate the global flow in order to isolate the localized object motion. This procedure of calculating the egomotion or global motion can be done using the checkbox seen in the GUI interface.

Interface



Instructions

1. Patch Size - The size of the patch to match in the current image to the previous image. Note that the larger the patch the better and more unique the match will be but the flow will be slower to calculate and less precise.
2. Search Window Size - The size of the search window in the previous image in which to look for the current patch. It is assumed that a patch in the current image will be found in a similar but not exact location in the previous image. The Search Window Size specifies how much of that surrounding space will be searched. The larger the window the more likely the patch will be found but this will increase the CPU requirements and therefore decrease the frame rate. If you notice a lot of errors in the flow try increasing the search window size. If you notice a slow frame rate try reducing the search size assuming your flow is relatively small.
3. Spacing - As it is not necessary to calculate the flow for every pixel in the image (as the CPU requirements would be too great) the spacing will only process every X pixels for their flow. This in effect reduces the resolution of the flow but can increase the frames per second.
4. Precision - In addition to spacing, you can specify a lower precision for flow detection which will further reduce the search requirements by moving the search window over multiple pixels at a time instead of 1 (High). For Medium, 1 pixel is skipped, for Low, 3 pixels are skipped.
5. Eliminate Global Flow - Eliminates the egomotion or global flow of the image to reveal objects that are not moving in coordination of the global flow. An example of this is happens when you pan the camera to the left and a person is walking to the right.
6. Min Error - Before a patch in the current image is matched with the previous image a quick check to determine if the image under the patch has actually changed is performed. If the error between this patch and the patch at the same location in the previous image is below the Min Error the patch is determined to have not moved and is disregarded from the flow. If you see changes to an image with no optical flow being performed try to reduce this value. If you require increased frames per second try increasing this value as it will allow the optical flow calculation to only focus on those patches that have moved by more than a small amount.
7. Max Error - When each patch is matched an error is calculated that determines the goodness of fit. Often a patch in the current image CANNOT be matched well in the previous image. This happens in many cases but is often related to objects overlapping each other or suddenly moving off screen. The Max Error defines the maximum error that is allowed in order for two patches to be considered a match. If you notice that there are many bad needles being created try decreasing this value. Increase this value if areas of the image that are changing are not generating any optical flow.
8. Needle Color - The color of the drawn optical flow needles
9. Vector Color - As an additional graphic an overall vector (average) of all the small needle flows is calculated. This is useful if you are trying to determine the direction of how an object in view is moving with respect to the camera. The Vector Color specifies in which color this vector is

drawn.

10. Block Color - Instead of viewing the optical flow as needles you can also chose to view the patch movements in an intensity mode. Select the appropriate color to view each block movement amount as an intensity value.

11. Overlay On - Specify which image the graphics would be draw on.

12. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.

13. Axis Restriction - If you are expecting movement only in a particular direction such as Vertical up and down movement you can restrict the optical flow checking to a particular axis. Axis restriction helps to eliminate those incorrect needles by enforcing movement checks only in the specified orientation. These irregular needles are mainly due to image noise but requires some knowledge about the potential image movement.

14. Filter Length - You can remove those needles that are too small or too long depending on the expected movement. For example, with sufficient image noise a lot of small one or two length needles may be creating. Setting a min of 3 would eliminate those needles and clean up the reported needle array.

15. Filter Orientation - Similar to filter length, the Filter Orientation will remove those needles that are not off the specified orientation. For example if you want to focus on North East needles you would use $\text{min} = 40$ $\text{max} = 50$ to eliminate all needles are that not within 40 to 50 degrees. Note that this is functionally similar to Axis Restriction above but allows for the needles to chose the closest match in any direction and then be removed if not matching a desired orientation. The Axis Restriction will ONLY match needles in the specified direction regardless of if a better match is in a slightly different orientation.

Example

Source



The lion is sipping at the water. Only the head is moving but camera angle was shifted.



The bird's head moved from being tucked under its wing. Note that the water in the background required the min error to be set very high due the time difference between shots

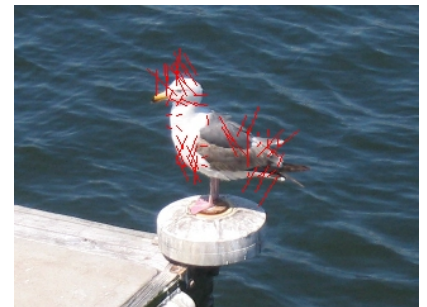
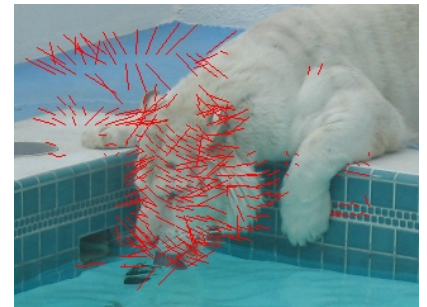


Almost stereo shot of the beachfront. Note the nice linear movement of the rocks but noise in the water. Again, due to the time different in shots the water has changed texture.



Moving forward down a forest path. Note the expanding optical lines showing a zooming motion

Optical_Flow



Variables

OPTICAL_FLOW_NEEDLES - An X_START,Y_START,X_END,Y_END list of lines that specify the start and stop coordinates of each optical flow needle. Note that this array is modulus 4 which means that each needle information occupies the next 4 numbers.

OPTICAL_FLOW_VECTOR_X, OPTICAL_FLOW_VECTOR_Y - The X,Y coordinate of the global vector that was calculated as the average from all the needles.

See Also

[Movement](#)

Orient Image

The orient image module will align an arbitrary image with respect to major gradient direction of the image. Basically this means that if an image has a lot of straight lines in one direction the entire image will be reoriented such that those lines are vertically straight. This is functionally similar to the [Visual Anchor](#) module but will align an image based on just the image's content as apposed to with respect to another image such as in the Visual Anchor.

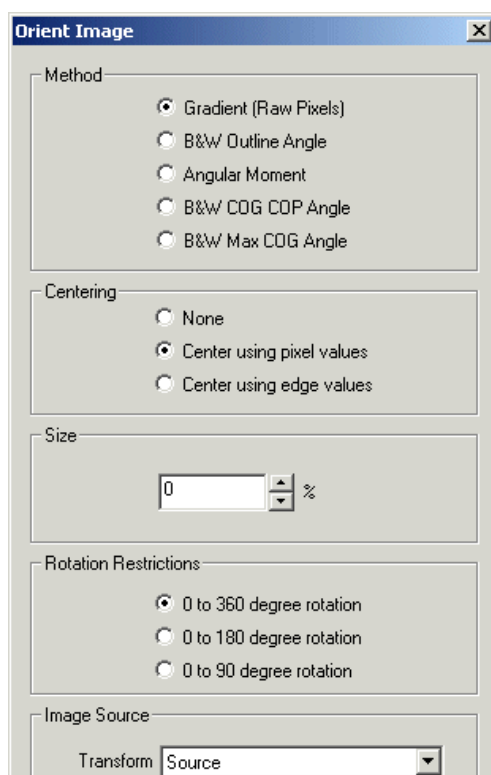
This is useful when image that you are working with need to be aligned to a canonical view (i.e. any image containing the same content needs to be aligned to a known orientation). For example, images of barcodes, playing cards, text, room scenes, etc. all exhibit enough straight lines for this module to work. Once these image have been aligned further processing is typically done to extract out other features or as precursor to modules such as barcode recognition, OCR, template matching, etc.

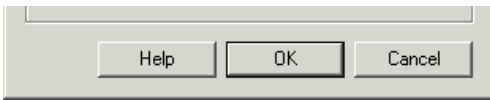
Straight lines are great to use for aligning images but do have a 180 degree ambiguity. Thus this module will tend to align images in either vertical direction and may flip the image even on very similar content. Thus you will need to allow for two final orientations for each image.

Note that by canonical orientation we mean an orientation that each image with the same content will end up in. This angle is not specified by you but determined intrinsically by the image contents. The actual angle it ends up in is not important, the fact that whenever the same image appears it is rotated to the *same* orientation is important.

Also note that this module looks for white pixels as being foreground and black pixels as being background. If you need you can add the [Negative](#) module to invert an image.

Interface





Instructions

1. Inserting the module will produce images aligned to their intrinsic orientation. Keep in mind that the orientation of the image is based on the image content. Not all images will have a stable alignment.

2. Method - There are many ways to determine an objects orientation. As no one method covers all cases well select an appropriate method for your image. Keep in mind that each method will chose a different final orientation for your image. This is normal. What you are looking for is that all images you test result to the same orientation whatever that might be. If you know that each image is oriented in the same way after this module you are use the [Rotate](#) module to rotate the image to a desired rotation.

- Gradient - This technique will look at a full color image and determine its canonical orientation based on the gradient (surface slope). This works best for images with lots of texture.
- B&W Outline Angle - This technique will threshold the image into a black and white object and then investigate the curvature of the outline of the resulting blobs in order to determine the canonical orientation. This is a good general technique but also suffers from a 180 degree ambiguity (i.e. the image may be rotated to 0 or 180 degrees at random) unless the objects shape is asymmetrical.
- Angular Moment - This uses the second order Moment of Inertia technique to determine the orientation. To see more about moments have a look at the [Moment Statistics](#) module.
- B&W COG COP Angle - This technique determines the objects COG (center of gravity) and its COP (center of perimeter points) and uses the angle between them to determine the canonical orientation. This works best for objects that have holes in them.
- B&W Max COG Angle - This technique determines the objects COG (center of gravity) and the point along the perimeter furthest from the COG. The angle between these two points is what determines the objects canonical orientation.

3. Centering - To center the image select an appropriate technique. "Center using pixels" will calculate the Center of Gravity of pixel values (i.e. white is high relevance whilst black is very low). This will cause an image with a white object to center in the middle of the image. Use "Center using edge values" to center an object based on the edges of an object. This will center an object based on what parts of the object have more texture or more edges.

4. Size - As an objects size can change in addition to its orientation the size percent can be used to create a canonical object size. If non-zero this field specifies the size to scale the image relative to this canonical size as the canonical size can sometimes be larger than the current image size and the need to shrink the object slightly is needed. Thus using the size parameter at 100% will show all the same objects with the same size.

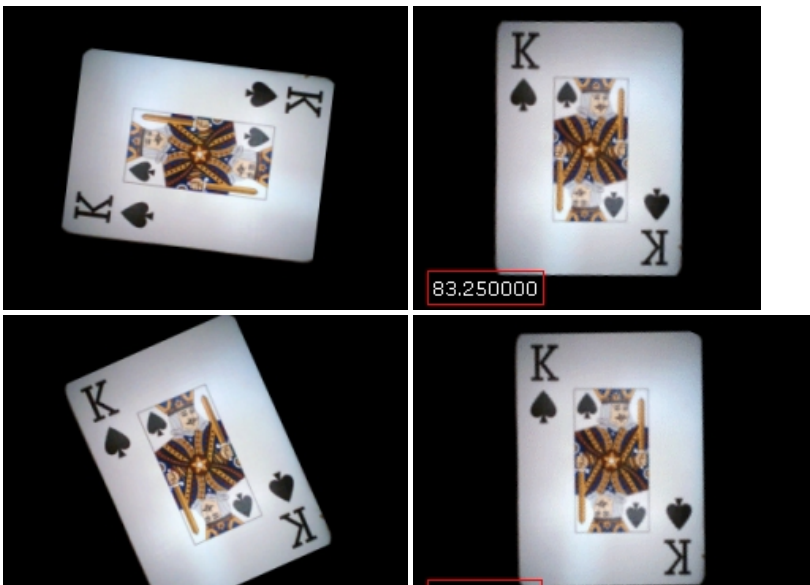
5. Rotation - Some objects will be asymmetric and always rotate to exactly the same orientation regardless of their initial orientation. But some images (like a square) are not asymmetric and may appear in 0,90,180,270 degree orientations. If you have an object that appears to flip orientation randomly try to reduce the allowed rotation amount to a reduced degree. This will stabilize the final rotation based on the initial orientation of the object. For example, if you have a square object and know that for the most part the object always appears ± 45 degrees in the desired orientation restricting the rotation from 0 to 90 will ensure that the final object rotation will be biased towards the initial rotation. Naturally, if your square object does not appear in about the right orientation then you will have to accept a 90 degree ambiguity.

6. Image Source - Select which image should the final transformation be performed on. As the current image might be pre-processed before entering into this module you can also specify "Source" in the dropdown which will transform the original image to its canonical rotation and size.

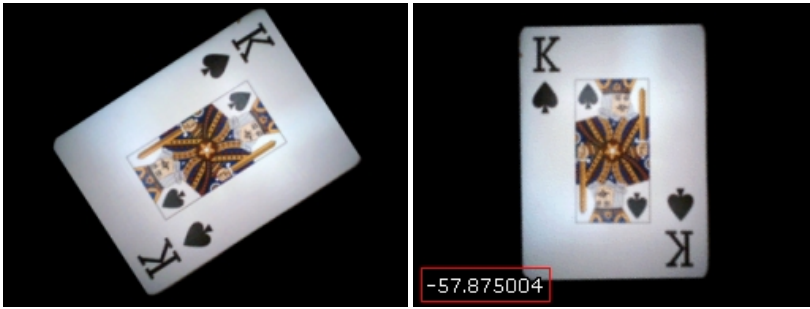
Example

Source

Oriented Image



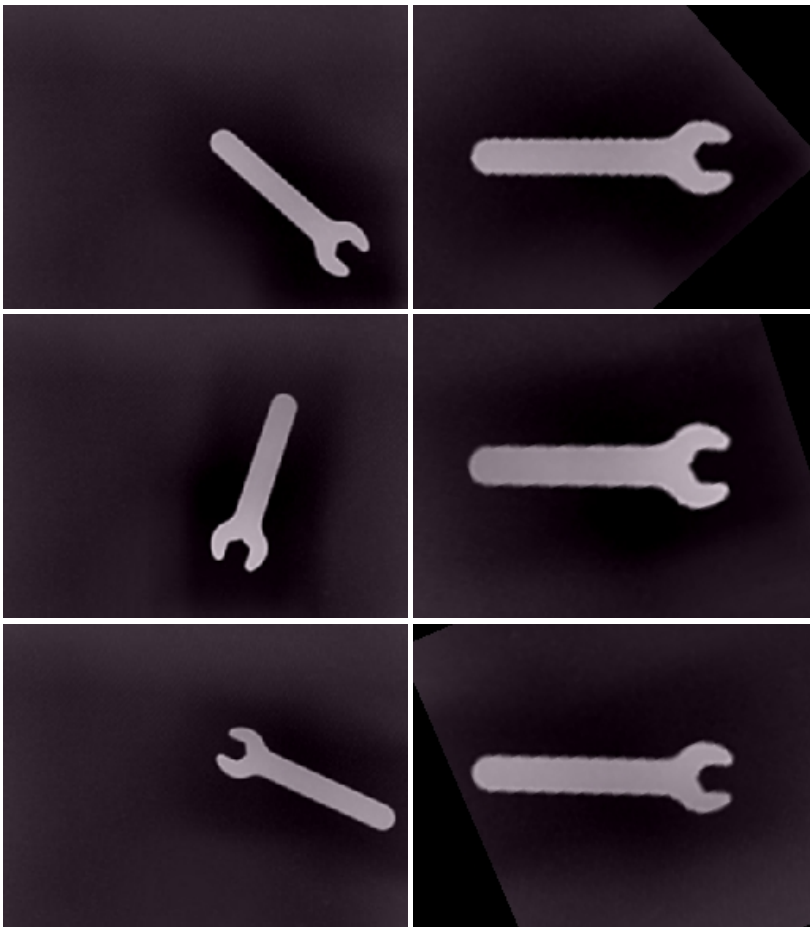
23.125000



For symmetrical objects that can be easily thresholded the B&W COG COP Angle performs better.

Source

Oriented Image



Variables

ORIENT_IMAGE_ROTATION - specifies what degree of rotation was applied to the image in degrees.

ORIENT_IMAGE_X_TRANSLATION - if centering is enabled this variable contains the horizontal translation in pixels of the image needed to center it.

ORIENT_IMAGE_Y_TRANSLATION - if centering is enabled this variable contains the vertical translation in pixels of the image needed to center it.

See Also

[Visual Anchor](#)
[Rotate](#)

Perspective Correction

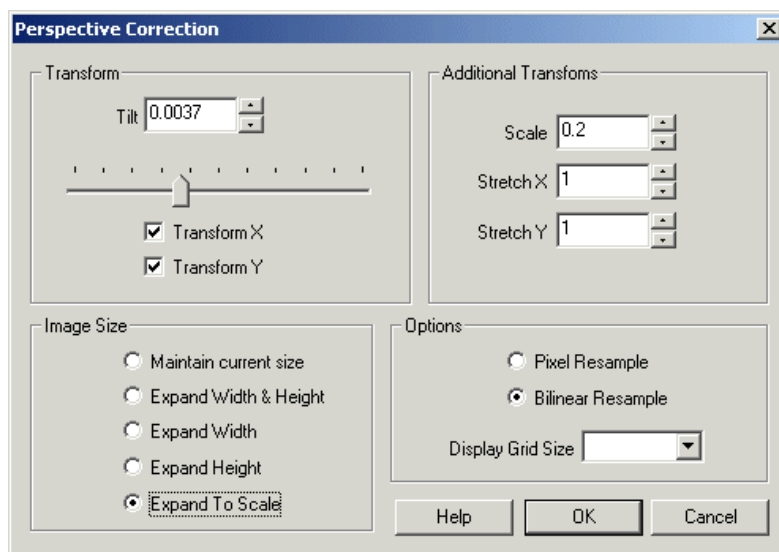
The Perspective Correction module provides an image transform that corrects for the effects of perspective. Perspective is created whenever a scene has depth and will distort objects to appear larger the closer they are to the camera source and smaller when further away. The Perspective Correction module can undistort some of the perspective effects to provide a more calibrated view for successive processing.

The effects of perspective can be readily seen when viewing a grid like surface stretching out into the distance. Measuring the pixel size of any of the square boxes will reveal that they shrink with distance. This can cause problems when attempting to map out obstacles in front of the robot as their sizes cannot be easily determined unless perspective is taken into account.

The Perspective Correction module works by effectively rotating the image such that it appears that you are looking straight down at the scene instead of at an acute angle. When viewing straight down at the scene square objects will appear square and object sizes relative to each other are comparable.

Note that the Perspective Module is only working with the current 2D image and is unable to perform a true 3D rotation due to the lack of data. However, despite this limitation a perspective undistorted routine is a very valuable tool to have for image processing.

Interface



Instructions

1. Tilt - specify the amount of perspective correction to apply to the current image. Note that the perspective correction is quite sensitive so the tilt number will be very small typically around 0.001 to 0.003.
2. Transform X - select to apply the perspective transform to the horizontal axis. Applying the transform to the horizontal axis will expand the image towards the top to undo the horizontal shrinking of objects due to perspective.
3. Transform Y - select to apply the perspective transform to the vertical axis. Applying the transform to the vertical axis will lengthen objects that are shortened due to perspective.
4. Scale - specify a number greater than 1 to zoom into the image, specify a number less than 1 to zoom out of the image.
5. Stretch X - manually stretches the image in the horizontal direction.
6. Stretch Y - manually stretches the image in the vertical direction.
7. Image Size - specifies how to crop the image dimensions based on the result of the transform. Note that the transform can create very VERY large images if a high tilt rate is specified. Use the Image size selections to automatically crop the image to a workable size.

Maintain current size - centers the transformed image into the current image size

Scale to current size - automatically scales transformed image into current image size

Expand Width & Height - grows the image to fit the entire transform into view. Note that this can cause delays in display while calculating such a big transformation.

Expand Width - only expand the image horizontally based on the resulting transformed size.

Expand Height - only expand the image vertically based on the resulting transformed size.

Expand to Stretch - expand the image size based on the stretch factors used in relation to the current image size.

8. Pixel & Bilinear Resample - select the interpolation mode. Pixel is fastest but blocky. Bilinear is smooth but slower.

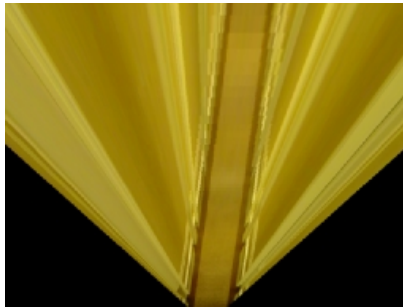
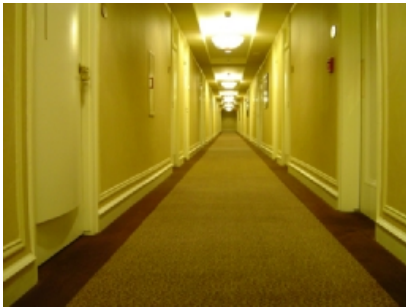
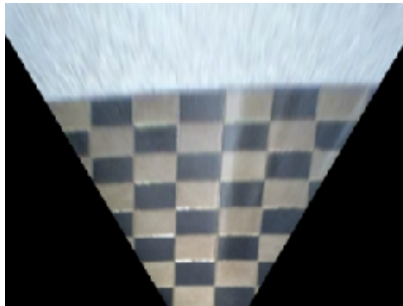
9. Grid Size - specify a grid size that will be superimposed over the current image to help in alignment.

10. Fill Color - specify the color used to fill border areas that are out of bounds of the image. As the image is warped in a non-square way there will be gaps between the image and the edge borders. The fill color specifies what color is used to fill in these areas.

Example

Source

Corrected



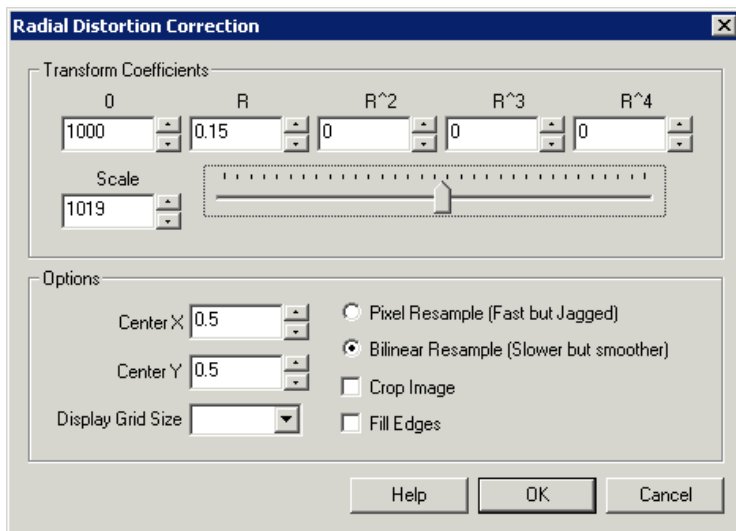
See Also

[Affine](#)
[Radial](#)

Radial Distortion

The Radial Distortion module provides a way to correct for Barrel and Pincushion distortion often seen when capturing images with webcams or pinhole type cameras. This is also a necessary precursor to performing image measures that can be obscured by lens distortions. Note that once a corrected image is used the coefficients can be used on any other scene with that camera as the distortion coefficients are NOT content related but are related to the intrinsic camera properties.

Interface



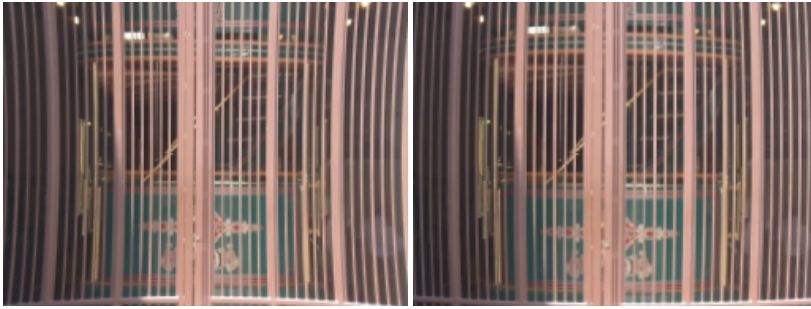
Instructions

1. Coefficients - specify the intrinsic camera coefficients. These can be obtained from the camera manufacturer or by experimentation. To manually correct use an image that has straight lines in it. For pincushion distortion the lines will bend outwards, for barrel distortion they will bend inwards. Adjust the coefficients until the lines appear straight.
2. You can use the slider to quick change coefficient R which will help you find the crude coefficients that can then be fine tuned using the above text boxes. Note that the higher order coefficients will normally be very very small.
3. Options Resampling - Select how you would like the transform to be performed. The Pixel Resample will use the closest pixel to the desired transform while the Bilinear method will use the pixel neighborhood to determine a new pixel value. The Pixel Resample is faster but less accurate. Most applications should use the Bilinear method unless performance is critical.
4. Crop Image - given a specific transform the image may contain areas that do not map correctly and will either appear as black borders or if the "Fill Edges" is on appear smeared. Selecting the "Crop Image" option will crop the image to a fully contained area. This has the unfortunate result of cropping actual image areas but is often not serious due to the minor amount of cropping.
5. Fill Edges - as apposed to cropping the image to a solid area you can chose to fill the unmapped edges to the closes border pixel. This will help to eliminate the dark borders but in extreme cases will appear to smear the image.
6. Display Grid Size - use to overlay a grid on the current image so that distortions become easier to see. Distortions will result in straight lines that appear bent. The grid draws many straight lines for you to compare with the lines in the image. Using this comparison you can better determine the optimal coefficients for the transform

Example

Pincushion

Barrel

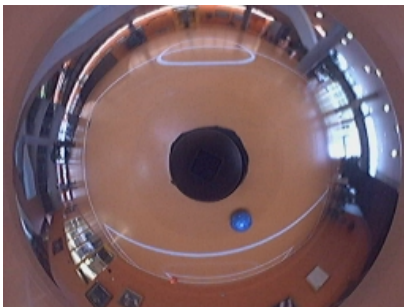


Corrected

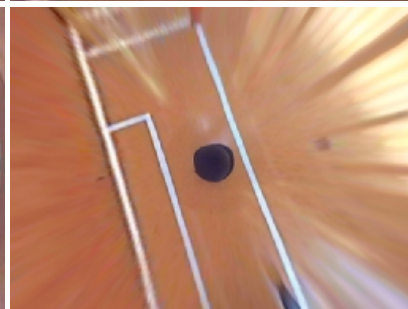
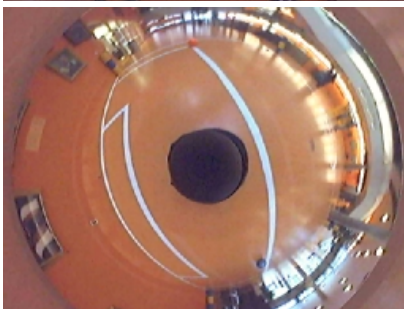
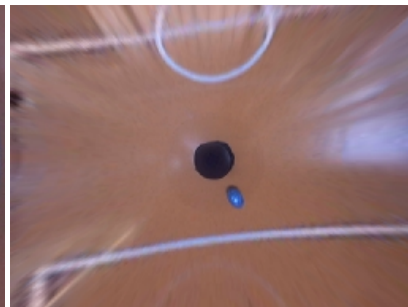


You can also use this module to unwrap 360 degree panoramic omnidirectional images.

Original



Undistorted



Polynomial

The following is a code snippet of how the distortion calculation works in order for you to better understand how the arguments are used. The following transforms a coordinate x,y into X,Y . Note that this module performs the inverse of this function in order to guarantee full pixel coverage.

```
SCALE = 1000;
P0 = 1000.0; // order zero coefficient
PR = 1.0; // first order coefficient
PR2 = 0.0; // second order coefficient
PR3 = 0.0; // third order coefficient
PR4 = 0.0; // fourth order coefficient

r = sqrt(x^2+y^2);
z = P0 + PR*r + PR2*r*r + PR3*r*r*r + PR4*r*r*r*r;
```

$s = -SCALE/z;$

$X = x*s;$

$Y = y*s;$

See Also

[Cylindrical Unwarp](#)

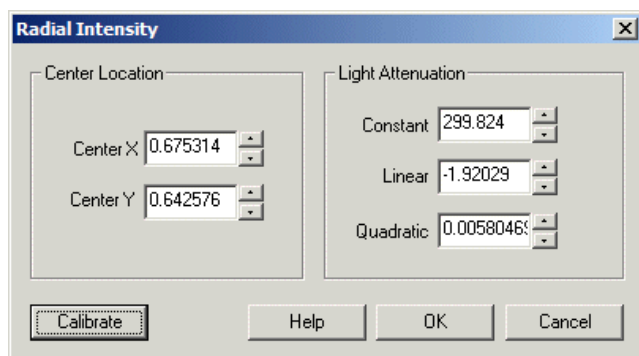
[Fisheye Transform](#)

[Polar Transform](#)

Radial Intensity

The Radial Intensity module is used to compensate for radial lighting, the kind that is generated from a single point light source and diffuses over a surface area. In many applications keeping the lighting as flat as possible can make segmentation or analysis of foreground objects much easier if the lighting/intensity is similar throughout the image.

Interface



Instructions

1. Center X - Specify the x coordinate of the highest light point in the image. This is the point that is closest to the light source.
2. Center Y - Specify the y coordinate of the highest light point in the image. This is the point that is closest to the light source.
3. Constant - Amount of intensity to adjust the entire image by.
4. Linear - Amount of intensity to adjust towards the center location by a linear amount.
5. Quadratic - Amount of intensity to adjust towards the center location by a quadratic amount.

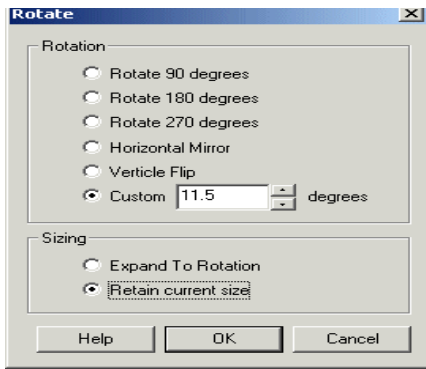
See Also

[Flatten](#)

Rotate Image

Rotate allows you to rotate the image by 90, 180, 270 degrees or mirror the image horizontally or vertically. Rotating the image is useful if the hardware does not permit the correct side mounting orientation.

Interface



Instructions

1. Rotation - To rotate, mirror or flip the image select the appropriate radio button
2. Rotation - To rotate using a custom degree click on the custom radio button and type in the degrees that you want to rotate the image by. Note that the range goes from -360 to 360 in degrees.
3. Sizing - The default rotation will enlarge the image when rotated or adjust the width, height to the new rotated image. There are situations when this is not desired and the original size is needed. If this is the case select the "Retain current size" radio button to keep the image the same size even after rotation. Note that when this is enabled parts of the image that will not contain pixel data will be set to black.

Example

Source



Rotate 90



See Also

[Shift](#)

Scale Image

The Scale module will scale an image by a percent factor. You can use this module to reduce or enlarge an image within the processing pipeline. Reducing an image can help to speed up the processing frame rate by reducing the amount of data to be processed. However, this is best achieved by changing the image capture format size in the OPTIONS button in the main RoboRealm dialog. Doing so causes the camera driver to scale down the image and send less data. This causes the frame rate to go up significantly.

Interface



Pixel Size

Width: Height:

Percentage

Width: Height:

Pixel Resize

Bilinear Resize

Maintain Aspect Ratio

Help OK Cancel

Instructions

1. Pixel Size / Percentage - Select the final width and height size as a percent or actual pixel size that you want to scale the image to. This can be smaller or larger than 100% or the current image size.
2. Options - Select the scaling technique that is to be used. Pixel resize is faster and is sharper than other techniques but can cause jagged edges on high contrast lines. Bilinear resize will use linear interpolation to determine the actual resulting pixel values. This technique is slower but will create smoother more pleasing results.
3. Other - While editing either the pixel size or percent size the module will adjust the width and height to keep the same aspect ratio. To remove this restriction uncheck the "Maintain Aspect Ratio" checkbox.

Example

Original



Scaled to 160x120



See Also

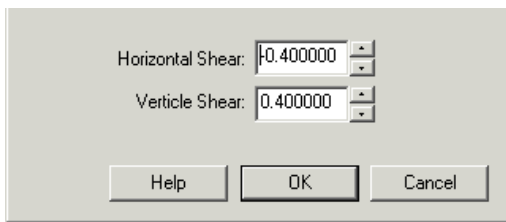
- [Translate](#)
- [Resize Canvas](#)

Shear

The Shear module will shear the current image in the horizontal (X) and vertical (Y) direction. This is useful if your images are coming in a little lopsided and you need a quick routine (other than rotation) to straighten them out.

Interface





Instructions

1. Specify the amount of shear (can be a decimal number) for horizontal and/or verticle shear.

Example

Source



Shear of .4 in both directions



Note that you can also specify variables instead of numbers in the text boxes, eg. [my_horiz_shear_amount].

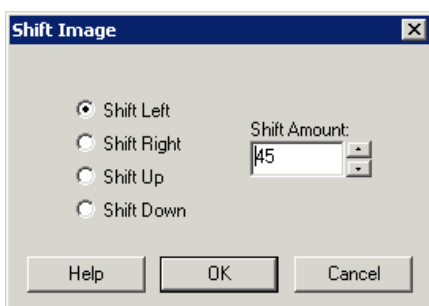
See Also

[Rotate](#)
[Shift](#)

Shift

Shift allows you to slide the image vertically or horizontally by x pixels. Shifting the image to the right by 10 pixels will cause the rightmost 10 pixels to become the leftmost 10 pixels; effectively sliding the image around by 10 pixels.

Interface

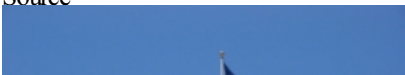


Instructions

1. Select which direction the shift should occur in by selecting the appropriate radio button
2. Type in the size or amount of shifting that should occur or use the up/down buttons to increase/decrease the shift amount

Example

Source



Left Shifted by 45

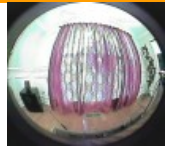




See Also

[Translate](#)

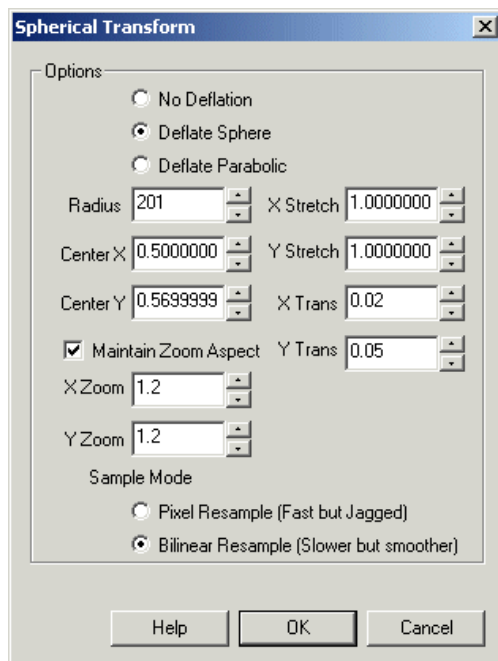
Spherical Transform



The Spherical Transform performs a circular unwarping of an image in order to produce a more usable image with regards to lines and distortion effects produced by wide angles lens. Similar to the [Radial Distortion](#) and [Polar Transform](#) this module assumes a circle image taken from a wide angle lens and provides you with many image warping controls in order to rectify the image.

The spherical transform is different than traditional image rectification performed by the [Radial Distortion](#) and [Fisheye Transform](#) modules in that it attempts to preserve more of the border objects rather than eliminating them from the resulting image. This helps to preserve the wide angle nature of the lens in that more objects are kept in view but adjusted for the lens distortion.

Interface



Instructions

1. Deflation - Select what type of lens you are using in order to start the initial deflation. In essence Deflation "pops" the balloon under the image by shrinking objects in the center of the image to a more normal size. Objects in the middle of a wide angle lens will become much larger than surrounding objects due to the shape of the lens.
2. Radius - Specify the radius of the circle within the image. The objective here is to increase or decrease this value until the edges of the image become as straight as possible. Either changing the number directly or by clicking on the up and down arrows you will be able to interactively explore which number works best.
3. Center X,Y - Use the Center values to move the center of the sphere to the right location. If your resulting transform appears to be lopsided try increasing/decreasing the center values to find the exact middle of the lens within the image.
4. Zoom X,Y - If the transform creates a resulting image that is larger than the current image use the Zoom number to reduce (or increase) the size of the image such that you can see as much of the valid image data within the current image viewing area. Note that you can uncheck the "Maintain

Zoom Aspect" to zoom the X and Y directions independently. This may be needed if your lens is slightly oval rather than a perfect circle.

5. Stretch X,Y - To stretch the image in the X and/or Y direction change the numbers accordingly. Note that this does not affect the transform in the way that the previous numbers do. Stretching simply resizes the final image transform to other dimension and adjusts the pixels accordingly.

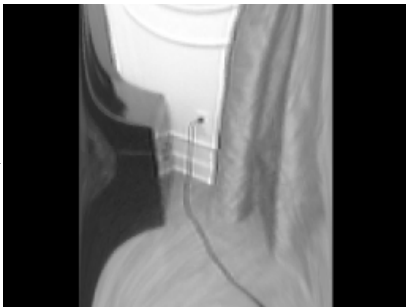
6. Trans X, Y - If your resulting transform is not quite centered in the viewable image area use the Trans X,Y to shift the image in the appropriate direction to maximize the image area taken by the transform.

Example

Source



Default Transform



Set Radial



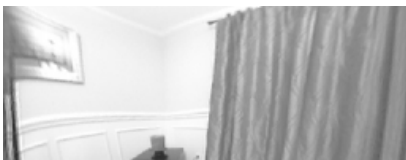
Set Center X,Y



Set Zoom X,Y



Set Trans X,Y





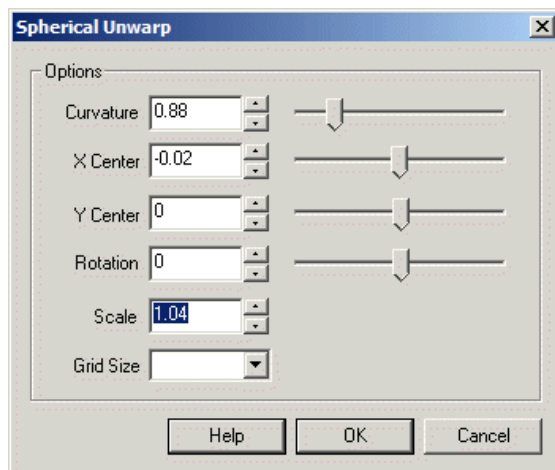
See Also

[Fisheye Transform](#)
[Polar Transform](#)
[Radial Transform](#)

Spherical Unwarp

The Spherical Unwarp module performs an unwarping operation on an image in order to remove warping on the border edges of an image. This warping is caused by lens distortion that causes objects towards the edges of an image to become stretched. This unwarping operation is needed for panoramic stitching or for consistent measurements of objects near the edges of images.

Interface

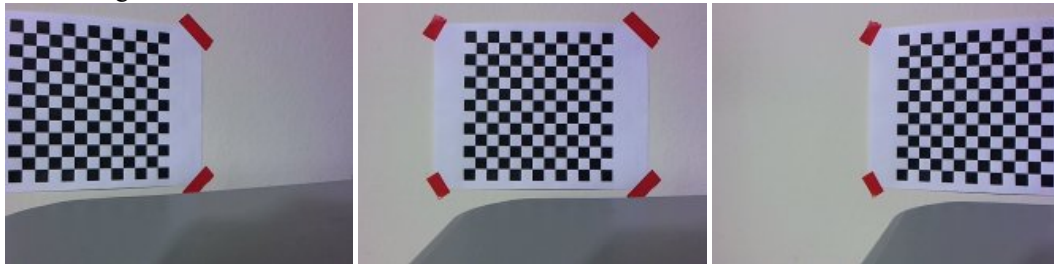


Instructions

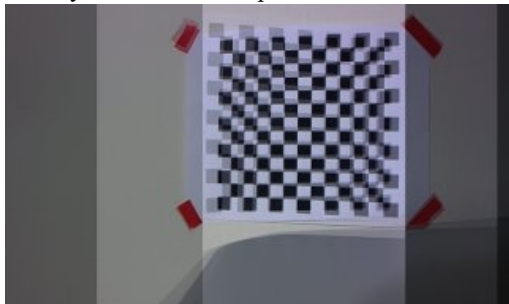
1. Curvature - Specifies how much curvature to introduce to the image left and right edge in order to correct for stretching of images due to lens distortions. Use the slider or up/down buttons to warp the image into correcting edge stretching. By ensuring that horizontal lines in the image remain horizontal towards the edges will help to correct for this form of distortion.
2. X Center - The horizontal center of the transform. This helps to correct distortions that are off-center from the middle of the image. Often the lens is not perfectly centered on the CCD which would cause the distortion to be off-center. By correctly aligning the transform the edge warping will shift and will become more consistent.
3. Y Center - The vertical center of the transform. This helps to correct distortions that are off-center from the middle of the image. Often the lens is not perfectly centered on the CCD which would cause the distortion to be off-center. By correctly aligning the transform the edge warping will shift and will become more consistent.
4. Rotation - Corrects for rotation of the CCD such that panning motion causes horizontal motion to remain horizontal. Note that the module is configured for -45 to 45 degree rotation as most CCDs are only misaligned by a few degrees.
5. Scale - On warping the image it will tend to compress and create black borders around the image. You can use the scale measure to increase the image size back up to touch the borders so that the absolute size of objects in the image will remain the same post warping.
6. Grid Size - To help you view how the warping adjustments affect alignment you can use the grid to better understand what numbers to use.
7. Crop - To remove the black borders around the image you can select the Crop checkbox. This will crop the image to the point that no black edges exist. Note that this changes the image size and removes pixel data that is otherwise correct.

Example

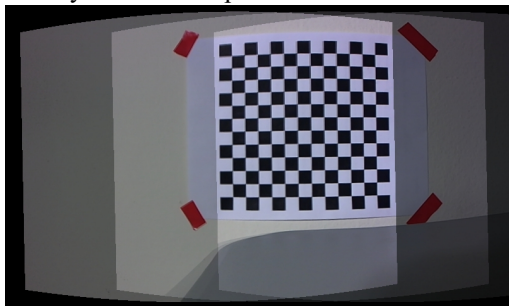
Source Images



Overlaid **without** Unwarp



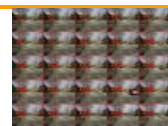
Overlaid **with** Unwarp



See Also

- [Cylindrical Unwrap](#)
- [Spherical Transform](#)
- [Radial Distortion](#)
- [Fisheye Transform](#)
- [Polar Transform](#)

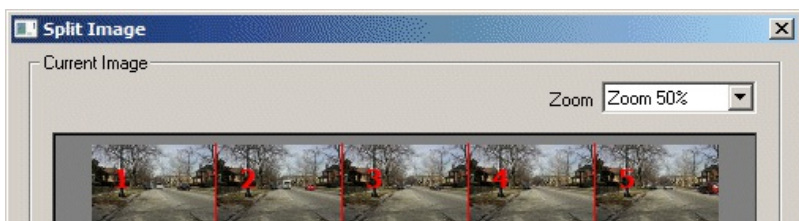
Split Image

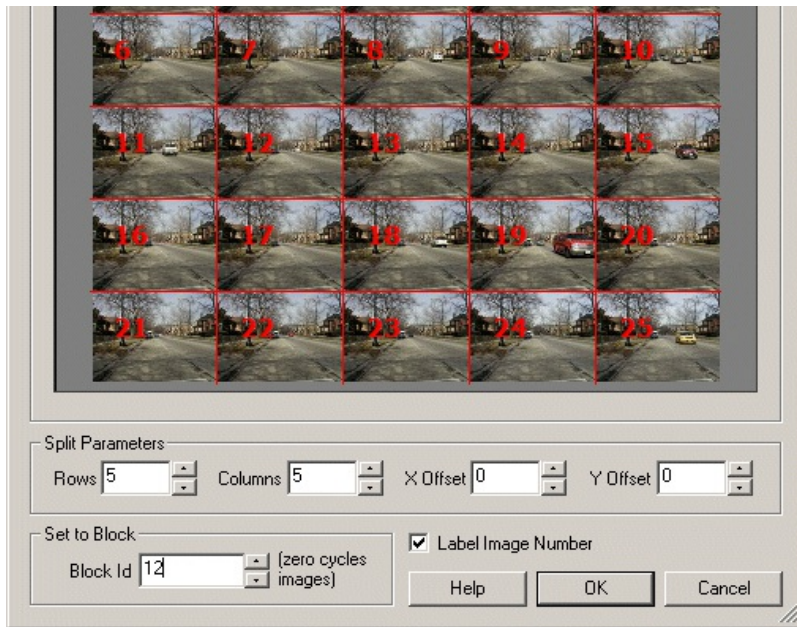


The Split Image module breaks a single large image into sub images and presents each of those smaller images into the pipeline one at a time. This allows you to break an image into rows and columns and process each part of the image as if it were a single image presented into the pipeline. This is very useful in those cases where processing a cropped part of the image is desired but the cropped area needs to move through the image in known increments.

This is often required in applications that present a tray of samples that need to be individually tested. Note that because a single image can become many additional individual images the frames per second recorded by RoboRealm will increase by how many sub images the single image is split into.

Interface





Instructions

1. Split Rows/Cols - Specify how many rows and columns you want to split the current image into.
2. X/Y Offset - Sometimes the image isn't aligned to structured splitting along rows and columns. The Offsets allow you to move the image to be better aligned with the break lines.
3. Set to Block - For review purposes, it is often required to stabilize on one part of the image to adjust parameters. Setting the Block ID will cause the module to only return the specified sub image block. This is useful when cycling quickly through all blocks is not desired. Note, setting this to 0 will cause the module to cycle through all blocks as expected.
4. Label Image Number - To better indicate what sub image block you are currently looking at this checkbox will annotate the block ID in red in the upper left corner of the image. Note that this is done as an annotation which will not affect further processing of image data (i.e. it floats above the pixel data).

Variables

SPLIT_IMAGE_ID - the current block ID of the sub image (increases from left to right, top to bottom)

SPLIT_IMAGE_ROW - the current block row of the sub image

SPLIT_IMAGE_COL - the current block column of the sub image

SPLIT_IMAGE_X_START - the start x coordinate of the sub image within the single source image

SPLIT_IMAGE_Y_START - the start y coordinate of the sub image within the single source image

SPLIT_IMAGE_X_END - the end x coordinate of the sub image within the single source image

SPLIT_IMAGE_Y_END - the end y coordinate of the sub image within the single source image

Makers

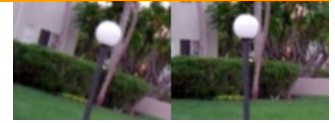
split_image - creates a marked image (in-memory saved image) of the current sub image block.

Because the 'Source' image will remain as the single large image, this marked image can be used instead of the 'Source' image to refer to the raw image data in successive modules that allow for image usage.

See Also

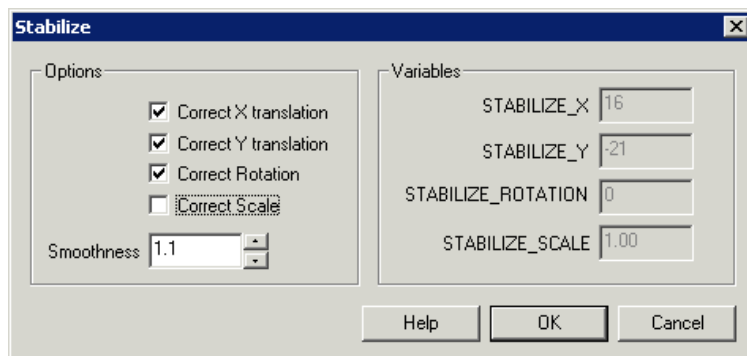
[Crop](#)

Stabilize



The Stabilize module provides a video stabilization function to eliminate camera shake. Removing video shake is an important component when comparing video sequences from frame to frame.

Interface



Instructions

1. Options - Select which correction you want to use to stabilize the video.

Correct X translation - will correct for horizontal movements

Correct Y translation - will correct for vertical movements

Correct Rotation - will correct for rotational movements

Correct Scale - will correct for zoom

2. Variables - Select the smoothness (greater than 1.0) that is used to smooth the frame sequence. At 1.0 frames will be kept stable with no adjustment. Thus if the video sequence has global movement that leads the image to the extreme right the video stabilization will eventually cause the video to disappear. At high numbers 5.0+ the video sequence will update very abruptly and essentially invalidate any smoothing that would be performed. Thus if your video has many twists and turns a number above 1.1 should probably be used otherwise quick turns can cause a large amount of the stabilized video to be offscreen.

3. Limits - Specify the maximum transform limits to avoid over adjustment of very unstable video. This will limit the range of adjustments and lower the ability of the module to stabilize the video. Using these limits you can confine the adjustment to the specified pixels/rotation/scale in order to be able to crop the image such that the black borders can be eliminated.

4. Reset - Reset alignment when limits are exceeded. Selecting this checkbox will cause the video to snap back into the original video if the limits are exceeded. This provides a way for rotations of the camera to not be overadjusted when rotating on purpose.

5. Background - Selects how to fill in the black borders created due to offsets of the current frame from previous frames that have not yet been filled.

Use Last Transformed Image - Draws the current aligned image ontop of the past transforms. Can create slurring due to previous frames

Use Last Image - Draws the current aligned image ontop of the last image without alignment. Helps to reduce the evidence of the black borders but can create choppiness in the border areas.

Mirror - Causes the black portions to be fill using a mirror image into the currently transformed image.

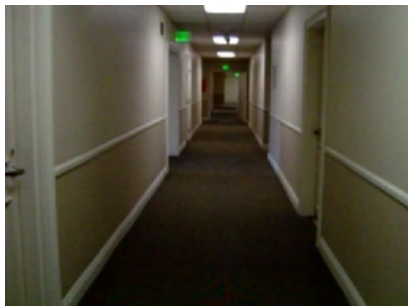
Examples

The examples show video taken while walking with a camera through a couple different environments. No effort during the filming of these videos was taken to stabilize the video and thus the individual steps taken during the filming are evident in the video. Each video shows the original images

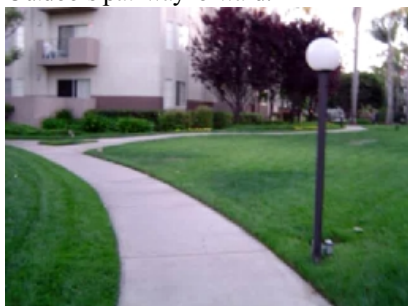
was taken to stabilize the video and thus the individual steps taken during the filming are visible in the video. Each video shows the original images on the left side whilst the stabilized video is on the right side. Note the black areas around the images in the stabilized video. These are the offscreen parts of the image that are created due to stabilizing the video. Whenever video is stabilized the current image needs to be realigned with the previous image. This alignment process may cause most of the image to be offscreen and thus black areas are used to fill in the remaining area.

Note the example robofile below that can be used to process your own videos, or simply delete the Media Reader module and switch on the camera to perform this stabilization processing live on your webcam! Note that the videos below show the stabilization process at frame rate, i.e. stabilization is performed while video is recording!

Indoor corridor scene.



Outdoors pathway forward.



Outdoors pathway reverse.



[Click here](#) to load how we created the above videos. Once loaded double click on the first module (Media Reader) and specify a video to stabilize. Press the Start button in that interface to start the stabilization process and then click on the Mosaic module to see the rendered results. (Note that clicking on individual modules shows the processing up to that point including that module.)

Variables

STABILIZE_X - contains the horizontal offset applied to
stabilize the video

STABILIZE_Y - contains the vertical offset applied to
stabilize the video

STABILIZE_ROTATION - contains the degree of rotation to
stabilize the video

STABILIZE_SCALE - contains the scale amount applied to
stabilize the video

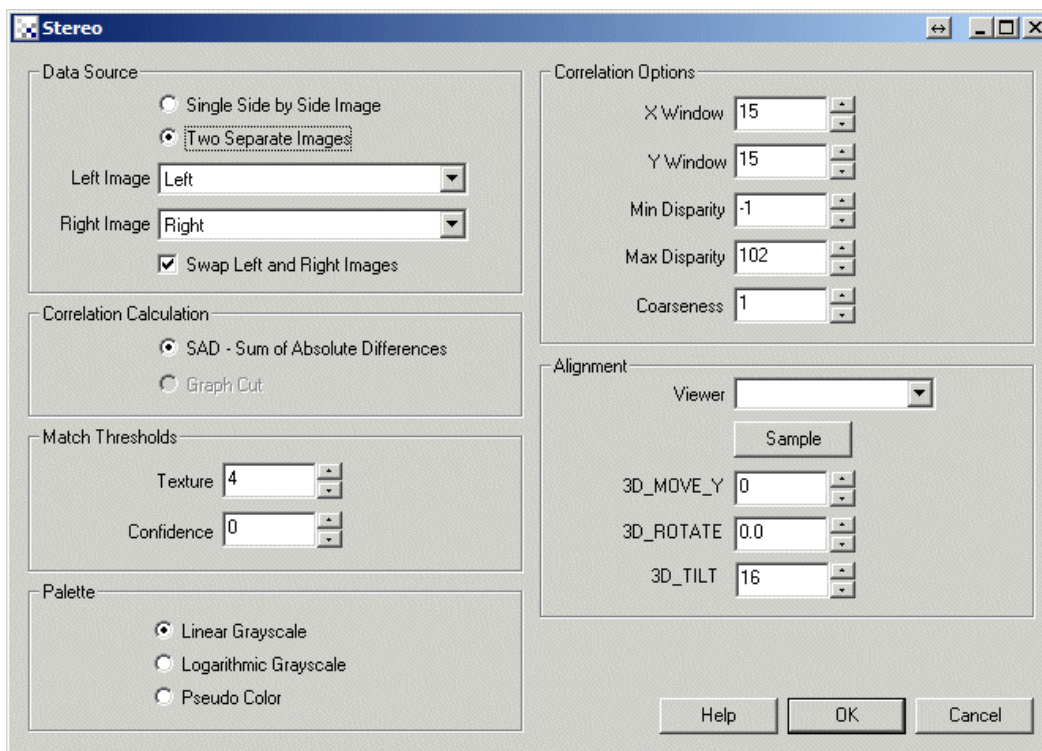
See Also

Stereo Depth

The Stereo Depth module uses two images to calculate a depth or distance image whose intensities reflect the distance to that point in the image. With two cameras one can use this module to determine nearby obstacles or know when objects are close by. Depth maps can also be used for depth segmentation where near objects can be removed from distance objects based on their depth values (not unlike color or intensity segmentation).

Stereo depth calculation is still a very active research area. With new methods and techniques appearing frequently new methods will be added when appropriate. The current technique combines speed with simplicity to produce usable results.

Interface



Instructions

1. Data Source - Select if the two stereo images will come from a single image or two separate images. Many stereo images are sent as a single image that needs to be split down the middle in order to create the two sides.

2. Left/Right Image - If two separate images are to be used select which ones are which. You can select two cameras or two markers that have been saved as images from other modules.

3. Match Thresholds - You can eliminate bad stereo matching areas by eliminating matches that are low in texture or confidence. While this will create unmatched areas within the image it will reduce false depth matches.

4. X/Y Window - The size of the block matching window that is used to determine correspondence between the left and right images.

5. Min/Max Disparity - The minimum and maximum disparity between the left and right image. The higher the max disparity the larger the depth disparity but the longer processing is required to determine that depth.

6. Coarseness - If you don't need a full depth map you can specify a larger coarseness to increase processing speed but with a lower resolution.

7. Viewer - Specifies how the two images should be viewed together

- Edges - Shows the combined edges of both images superimposed on each other.
- Flicker - Causes the two images to flicker back and forth between each other. This helps the eye to understand the movement between the two images and can help to determine alignment or rotational issues. No glasses are required for this mode.
- Anaglyph - produces an with the left or right channels combined into the color channel of a single image. You will need the red/green,

red/blue, etc. colored lens glasses to see the actual 3D image.

- Side by Side - Shows the left and right images combined together in a single side by side image.

8. Sample - Attempts to determine global rotation and Y movement automatically.

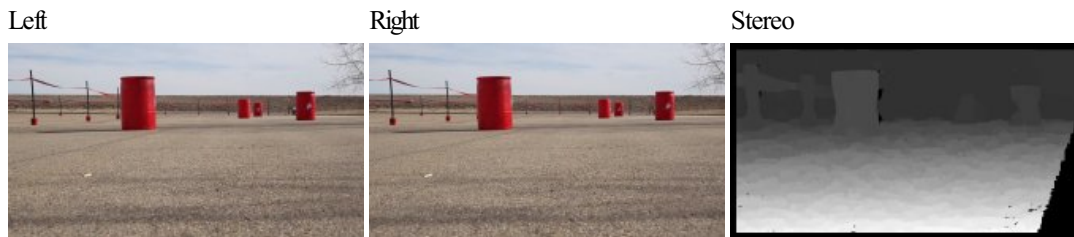
9. 3D_MOVE_Y - Specifies how many rows the left and right image are misaligned. In order for the stereo correspondence to work well the left and right image correspondence needs to be on the same row/line. Because camera alignment may not be perfect adjustment of the Y correspondence may need to be adjusted.

10. 3D_ROTATE - Specifies the rotational difference between the left and right images.


11. 3D_TILT - Specifies the tilt amount difference between the left and right images.

12. Palette - Specifies how the depth map is displayed. Linear correlates directly to the image displacement. Pseudo Color will color different depths with different colors.

Example



[Download](#) the above example robofile and images taken from the Sparkfun AVC 2012.

 [Download](#) another example robofile that shows how to use two separate webcams to create a stereo image. This uses the [Camera Properties](#) module that you will have to edit to specify your cameras. These images are saved as [Markers](#) and then used in the Stereo module. Don't see anything useful? Try swapping the Left and Right image (checkbox in the Stereo module). Because the images are saved to memory prior to usage in the Stereo module you can also apply other modules such as the [Radial Distortion](#) in order to correct images prior to correlation in the Stereo module.

See Also

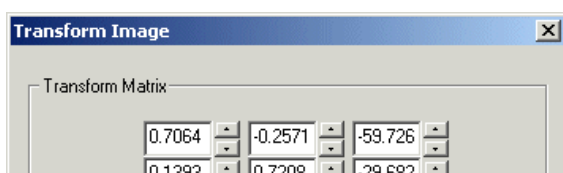
- [Creative Senz3D](#)
- [OpenNI Kinect](#)
- [LeapMotion Controller](#)
- [Hokuyo Laser](#)
- [Visible Laser Line](#)

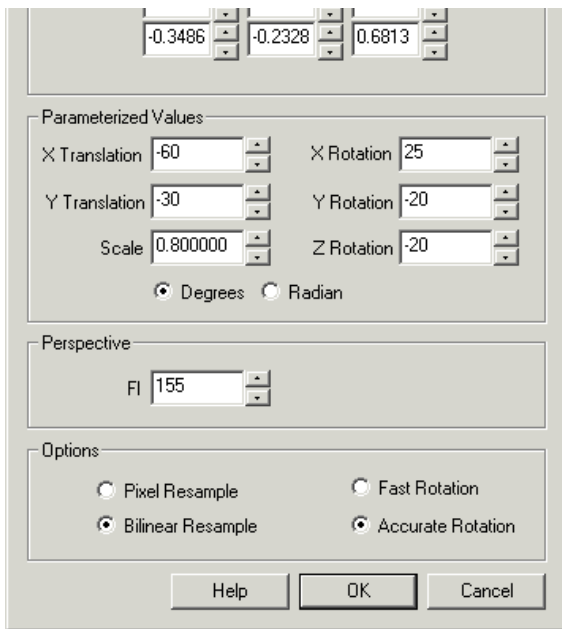
Transform Image

The Transform Image module provides a way to transform an image in terms of X,Y,Z rotation, scale, X,Y translation and the addition of perspective. This module provides a way to better align 2D images such that they can be further processed by other modules which might be sensitive to image transforms.

The transform image module provides a more comprehensive image transform module that just the [rotation](#), [scaling](#), [shearing](#) and [translation](#) modules alone. Note that if your sole task is just rotation, scaling or translation it is recommended to use the individual modules as they are more specialized to their task and therefore faster operations. However, the Transform Image module does provide more options in performing these tasks together in terms of the addition of perspective and the ability to manipulate the homogeneous matrix directly. The module also attempts to keep the parameterized values (rotation, translation, scale) in sync with changes to the homogeneous matrix. Note that as the matrix can represent more transforms than just the three parameterized values can the synchronization will only be approximate. Thus making changes to the matrix and then making changes to the parameterized values may cause a slight jump in the image as the values settle to the new inputs.

Interface





Instructions

1. Parameterized Values - Change the appropriate value to transform the image in the specific way. Note that you can either use Degrees or Radians to update those values.

X, Y Translation - moves the image up or down and left or right

Scale - shrinks or enlarges the image

X Rotation - Rotates the image in the X plane by twisting the image left and right.

Y Rotation - Rotates the image in the Y plane by tilting the image up and down.

Z Rotation - Rotates the image in the Z plane (aka orientation) by spinning the image around.

2. Transform Matrix - when updating the Parameterized values the transform matrix will also update. You can change any of the matrix values to see how the changes affect the image and how they update the parameterized values. Note that the matrix has much more flexible transform abilities than the parameterized values do so the updates may not be 100% reflected in the parameterized values.

3. Perspective - to increase the perspective of the image decrease the value. If you do not want any perspective to be applied to the image set this value to zero. Perspective will cause pixels further away from the image plane (your computer screen) to shrink towards each other, with closer values being enlarged. This effect causes a sense of depth in a 2D image.

4. Options - chose the appropriate sample and speed options. Pixel Resampling is not as accurate than the bilinear resampling but is much faster to perform. Likewise Fast rotation allows quicker updates when the parameterized values or matrix values change on each new image whereas Accurate rotation will provide a better final image quality after the transform is performed. Typically the default values should provide the best quality and speed possibility but if you use variables as values using the [variable] notation in any of the parameterized or matrix values you may want to switch to Fast Rotation.

Example

Source

Transformed Image



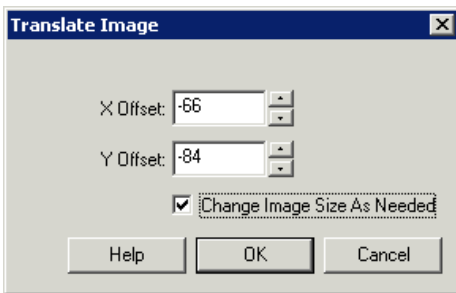
See Also

- [Rotation](#)
- [Scaling](#)
- [Shearing](#)
- [Translation](#)

Translate Image

The Translate module allows you to move the image vertically or horizontally by a number of pixels. Once the image has been translated to another location the processing will continue with the image in that location.

Interface

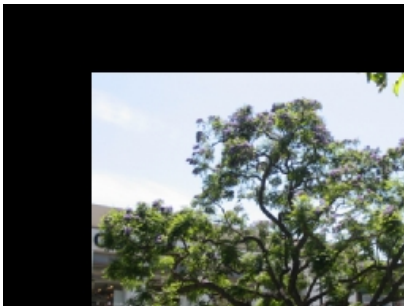


Instructions

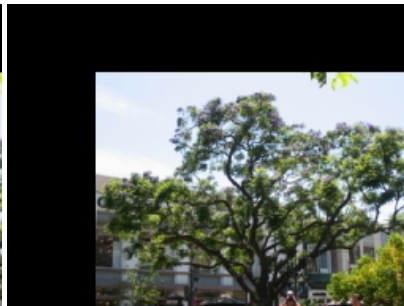
1. Use the buttons or text areas to type in the translation desired. Negative X values moves the image to the left, positive values to the right, negative Y values move the image down, while positive values move it up. The offsets are limited in X from -1280 to 1280 and -960 to 960 for Y
2. If you want the image canvas size to expand when translating select the "Change Image Size as Needed" checkbox. This will allow the image to grow to still accommodate the original image while translating it by the offset amounts.

Example

Translated (56, -43)



Translated (Change Size)



See Also

- [Shift](#)

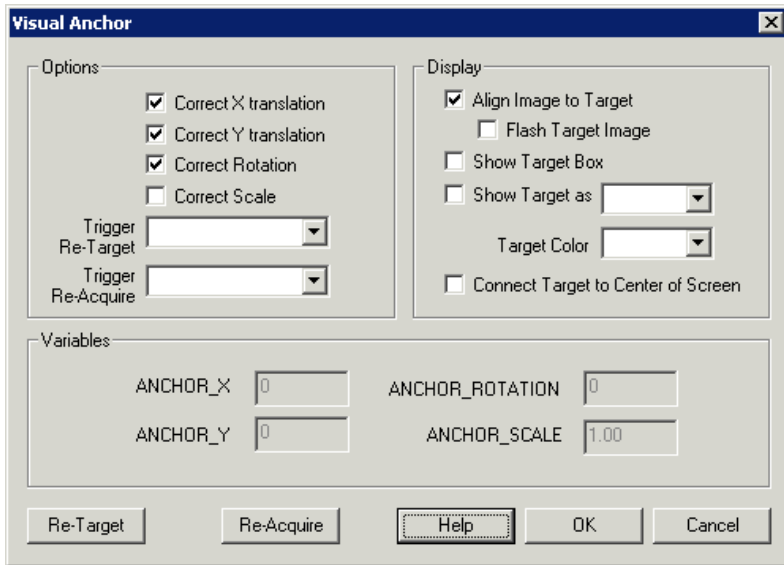
Visual Anchor

Similar to the [Stabilize](#) module, the Visual Anchor module provides a stabilization functionality used specifically for targeting scenes.

The Visual Anchor module will align current images with a targeted image taken as soon as the module is inserted or when the Re-Target button is pressed. The result of the alignment is then set into variables that can be used in other modules. These variables can then be used to steer the robot directly towards the target, stop at a certain zoom factor, or just keep the servo head pointing in the same direction.

Interface

INTERFACE



Instructions

1. Correct X translation - aligns the current image to the target image along the X axis/horizontal direction.
2. Correct Y translation - aligns the current image to the target image along the Y axis/vertical direction.
3. Correct Rotation - aligns the current image to the target image around the Z axis/rotational direction.
4. Correct Scale - aligns the current image to the target image along the Z axis/zoom direction.
5. Trigger Re-Target - specify which variable will contain a non zero/false value that when present will trigger a new target anchor to be acquired from the current image stream. This variable will immediately be set to "" once the trigger has executed and a new target has been selected. This resets the variables to zero.
6. Trigger Re-Acquire - specify which variable will contain a non zero/false value that when present will trigger a new offset to be set from the current image stream. This variable will immediately be set to "" once the trigger has executed and a new image has been acquired. This is functionally the same as re-target but does NOT set the variables to zero. This allows you to grab fresh images from the camera but still keep the original target offsets in mind. This is useful when extreme rotation or scale cause the original target to have little overlapping pixels with the current image.
7. Align Image to Target - select this checkbox if you want the current image to be aligned with the target image. This causes the current image to undergo translation, rotation, scale and zoom transforms in order to best align with the original target image.
8. Flash Target Image - if you would like to see the original target in the context then select the "Flash Target Image" checkbox. The original target will flash every 1 second to show how the current image aligns with the target image.
9. Show Target Box - draws a box that represents where the target image is within the currently viewed image. It makes best sense if the "Align Image to Target" is off.
10. Show Target as - draws a shape in a specific color at the center of the target image.
11. Connect Target to Center of Screen - draws a line from the target offset to the current screen center.

Performance Note

This module can require a significant amount of computation in order to align images correctly. To improve performance use smaller images or remove some of the alignment criteria. For example, if you are using the Visual Anchor module in a robot traveling on the ground with a camera pointing forward you may be able to remove the Y and Rotational alignment as the robot only moves around the planar floor and does not rotate. This will help to improve performance.

Example





Aligned (gray is target)

Current



First test: Switch on your camera, insert the module, press Re-Target and then move your camera slowly. Notice the black areas around the image that are the artifacts of the alignment process? Try rotating the camera and notice the image stays the right way up!

[Click here](#) to load a configuration to move the Traxster Robot forward until the scene scale is 2x. You will have to edit the Traxster module and specify your appropriate COM port AND edit the Visual Anchor and press Re-Target when the Traxster is positioned where you want it to move forward.

[Click here](#) to load a configuration that will track the X axis movement as you twist the camera. This process can be used as a visual compass to tell how far you have turned. We found 90° to be around 600 pixels.

Variables

ANCHOR_X - the X pixel translation is needed from current to target

ANCHOR_Y - the Y pixel translation is needed from current to target

ANCHOR_ROTATION - the rotation in degrees (0-360) needed from
current to target

ANCHOR_SCALE - the scaling needed from current to target. 1.0 is no
scale.

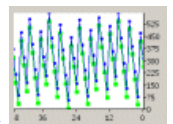
See Also

[Stabilize](#)

[Align_Image](#)

Chart Variables

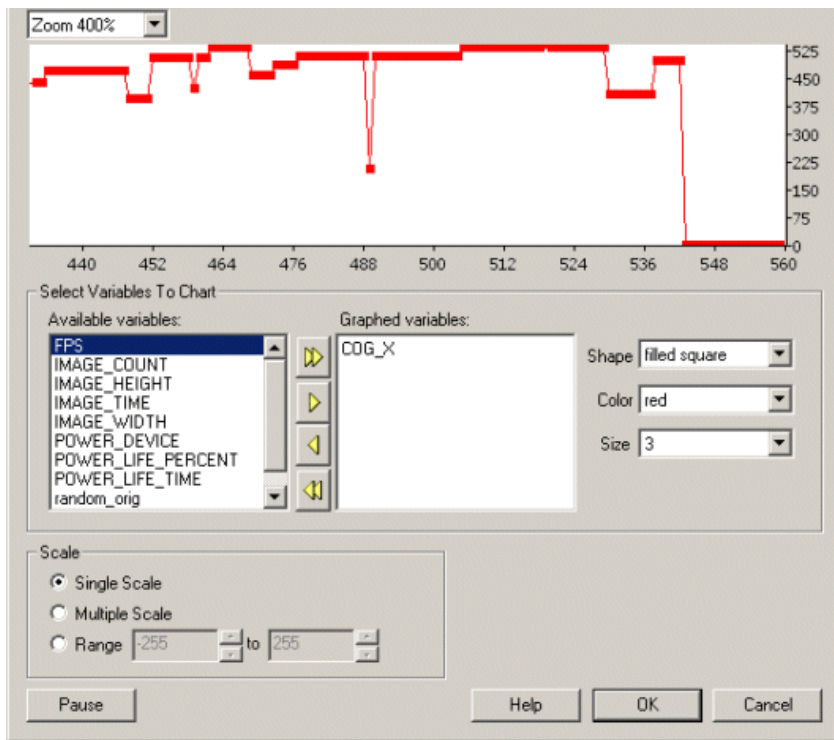
The Chart Variables module provides a way to better view the changes that occur for a given variable over time. While the [Watch Variables](#) module shows the current value of a variable at the current time it is sometimes preferable to view how a variable changes value over time or be able to view more of the context of a variable when changes happen. The Chart Variable will read the currently specified variables values and plot them on a simple chart scrolling on each new image capture. Note that the placement of the Chart Variable module is relevant as is with the Watch Module. If you change a variable AFTER the chart variable then the module will not see the change until the next pipeline iteration when a new image is captured.



Currently only 1000 values are held for charting purposes. Values older than 1000 samples are dropped.

Interface





Instructions

1. Variables to Chart - Select which variables you wish to chart using the provided "Available Variables" listbox. Either double click on a variable or select a variable and click on the appropriate yellow arrow to move the variable into the Graphed Variables listbox. As soon as the variable shows in the Graphed Variables listbox it will be charted. Note that the default chart color is red for all new variables so be careful when adding multiple variables to select different chart colors.

2. Shape / Color / Size - Select the appropriate shape, color and size of the small square that indicates a value for each sampled value of the graphed variable. Note that the color is also used to connect the values to form a line chart.

3. Scale - Select how the chart should reflect the different vertical values for the graphed variables.

- Single Scale - Use a single scale to encompass all the variable values into a single graph. Note that if one variable has a high range whilst another has a very low range then the second variable will appear to be unchanging as the first variable will define the vertical scale of the chart. I.e. the first variable's values will overpower the scale of the second and make it appear unchanging. If this happens select the Multiple Scale option.
- Multiple Scale - This scaling scheme will determine an effective vertical scale for each variable individually and plot that variable with respect to its own scale. Thus if one variable has a very high range it will NOT affect another variable with a smaller range. All variables are plotted with respect to only themselves and NOT each other.
- Range - Lastly you can specify your own scale to use when viewing the variable values. This causes each variable to be plotted with respect to ymin and ymax values. If a variable's value is above or below this range it will NOT appear on the chart. You are responsible to determine the best range to plot all variables being monitored.

4. Zoom - select an effective zoom or horizontal scale. When viewed at + 200% each value is marked with a square. You can also scroll the chart by grabbing and moving the chart left or right using the mouse. Moving the mouse over the chart will highlight the nearest value and display the value at that point.

See Also

[Watch Variables](#)

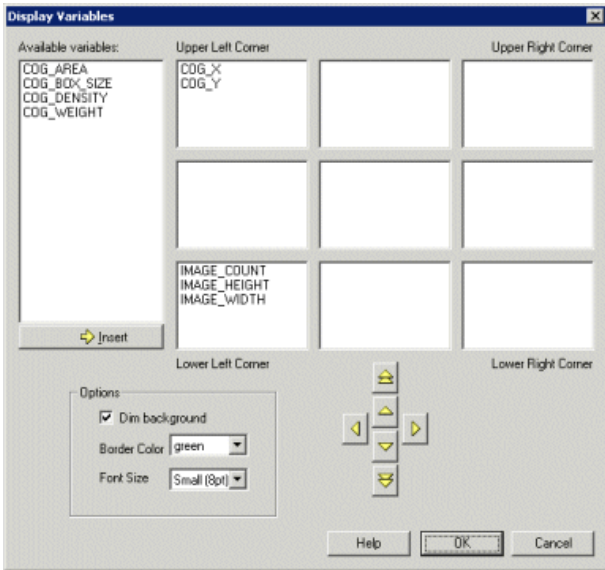
Display Variables

The Display Variables module will draw variables and their values into the current video stream for use in recording and display purposes. Note that the graphics are overlaid after all processing is completed to ensure that added graphics do not become part of the processed image.

When recording a filtered sequence it can be useful for end viewers to include any statistics within the image that you are trying to achieve

WHEN RECORDING A FILTERED SEQUENCE IT CAN BE USEFUL FOR END VIEWERS TO INCLUDE ANY STATUSES WITHIN THE IMAGE THAT YOU ARE TRYING TO ACHIEVE.

Interface

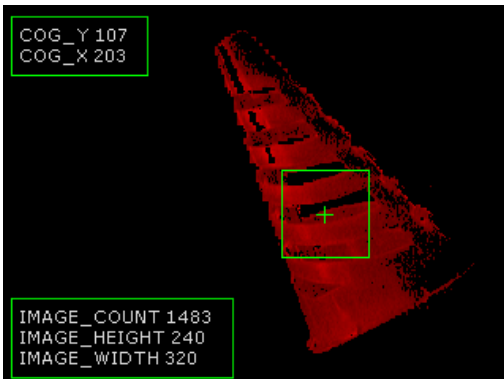


Instructions

1. Insert a variable by double clicking on an available variable or select a variable and click on the insert button.
2. Once a variable is in a slotted box it can be moved to alternate locations using the yellow arrow keys. Note that the double up and down arrows move the variable to the next up or below location box.
3. Dim background - Select if you would like to "dim" the background of the text area. This reduces the background of the text intensity to make it easier to read the white text.
4. Border Color - Select the color of the box that surrounds the text.
5. Font Size - Select the font size to use for displaying the variables.
6. Font Color - Select the font color to use for displaying the variables.
7. Display as Annotation - Select if you want the graphic to be draw after all processing has been completed. If this is NOT selected then the next module in the processing pipeline will see the graphic as if it were part of the image and process it accordingly.
8. When done press ok to save the current configuration. Note that any changes are immediately visible in the image preview in the main dialog.

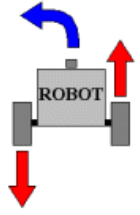
Example

Annotated image



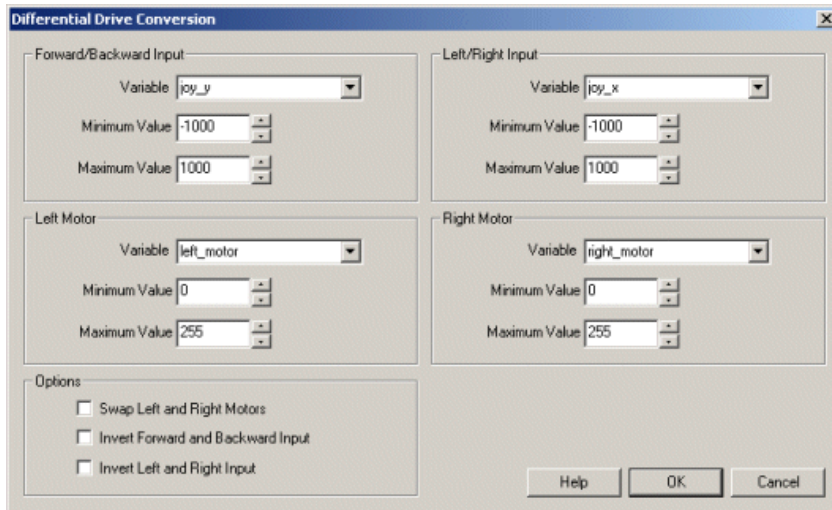
See Also

Differential Drive



The Differential Drive module provides a quick and easy way to convert from a forward/backward and left/right input into a left forward/backward and right forward/backward drive. Differential drive is most common in two wheeled robots which are controlled by two servos or motors and have forward and reverse capabilities. The module solves the problem of converting from one directional specification into a differential one.

Interface



Instructions

1. Forward/Backward Input - The variable that contains the value represents the forward and reverse movement of the robot. The minimum and maximum values define the full range of what those values can be. Typically, the higher this value the faster the robot should move, the lower the slower. Negative values or values below the mid point would cause the robot to move in reverse.
2. Left/Right Input - The variable that contains the value represents the left and right movement of the robot. The minimum and maximum values define the full range of what those values can be. The extremes are defined as the robot spinning in one direction versus the other. Values close to the mid point would stop the robot turning.
3. Left and Right Motors - The variables that WILL contain the value for each motor. The minimum and maximum values specify the full range of possible values for that motor and can be different for left and right sides (but not usually). These values are then used in other motor drive modules like the [Dimension Engineering Sabertooth](#) motor drive or the [Pololu Maestro](#) servo controller (to name a few) which would send that value straight to the motor or servo which would cause the robot to move.
4. Invert Forward/Backward - If you find that the robot is moving forward when it should be backward select the "Swap Left and Right Motors" checkbox. This will invert the Forward/Backward input (i.e. joy_y) to cause the robot to move in the opposite direction.
5. Invert Left/Right - If you find that the input forward/backward direction (i.e. joy_y) is moving forward when it should be backward select the "Swap Left and Right Motors" checkbox.
6. Oppose Left/Right - If you find that you are moving forward when you should be moving left or right it is possible that the motors/servos were not mounted opposing each other. I.e. to move forward the values need to be opposite each other (-255, 255) instead of the same (255, 255). Selecting the Oppose checkbox will correct this issue but will then likely need to have Inverted Left/Right or Forward/Backward changed too.
7. Predictive Behaviour - As soon as the input values disappear the motor values will immediately become neutral (i.e. stop). To avoid this and have the motor values slowly approach neutral you can enable the motor value Decay which will decrease the values towards neutral at the specified rate. This allows you to add some lag into the system which causes the robot to continue to move in its last direction for a while longer in hopes to reacquire a lost target that has moved off screen.

Note, the decay value will linearly decrease the values over each frame where the input values are empty (i.e. undefined/removed or have no value/empty string ""). This is different than a zero 0 value.)

Notes

The code used to perform the conversion in VBScript is

```
' amount of y joystick determines speed
speed = 128 - CInt(((GetVariable("joy_y") * 128) / 1000))


' amount of x joystick determines rotation
turn = CInt(((GetVariable("joy_x") * 128) / 1000))

' determine intermediate values
mleft = speed + turn
mright = speed - turn

'ensure they are within the range 0 - 255
if mleft<0 then mleft=0
if mright<0 then mright=0
if mleft>255 then mleft=255
if mright>255 then mright=255

' finally set the variables to be used in the joystick module
SetVariable "left_motor", mleft
SetVariable "right_motor", mright
```

Example

 [Click Here](#) to download a robofile that shows how to use the Differential Drive module with a joystick and Create Robot.

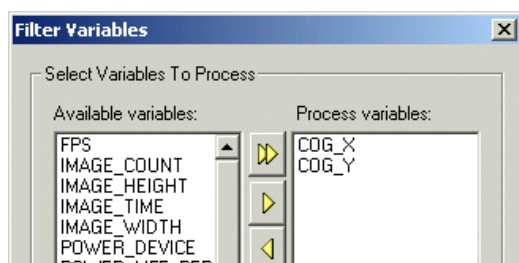
See Also

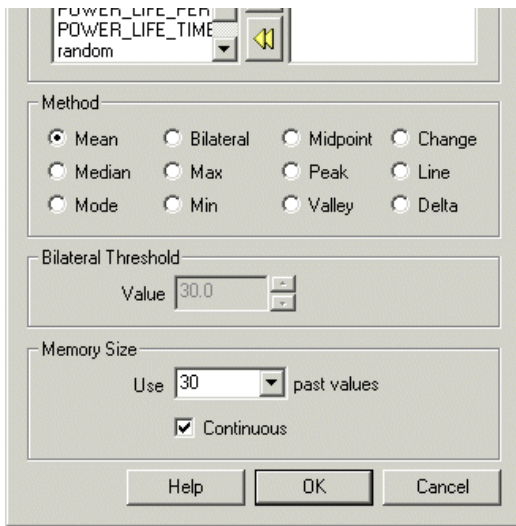
- [Scale Variable](#)
- [VBScript Program](#)
- [CScript Program](#)
- [Python Program](#)

Filter Variables

The Filter Variables module provides a way to filter a variable's value over time in order to remove noise generated in a given image. For example, when tracking a color object using the [Center of Gravity](#) module the actual COG_X and COG_Y reported will tend to vibrate somewhat even in a stable image due to pixel noise. Using the Filter Variable module you can specify that the current value is in fact the current value plus a combination from previous values of that variable. This allows you to smooth the current value based on previous values to provide a more stable representation of what the current value is.

Interface





Instructions

1. Variables to Process - Select which variables you wish to filter using the provided "Available Variables" listbox. Either double click on a variable or select a variable and click on the appropriate yellow arrow to move the variable into the Process variables listbox. As soon as the variable shows in the Process Variables listbox it will be filtered. Note that if you wish to preserve the current unfiltered value you will need to create a copy of the variable. See the [Set Variable](#) module to do so.

2. Method - Once your variables are selected you will need to select which filtering method to apply to the variable.

- Mean - Calculates the mean of the last X and current variable values and substitutes the current value for that result.
- Median - Calculates the middle value of the last X and current variable value from a sorted list of the past values and substitutes the current value for that result.
- Mode - Determines the most frequent value of the last X values and substitutes the current value for that result. If no values occur more than once then the Median is used instead.
- Bilateral - Calculates the bilateral value of the last X values and substitutes the current value for that result. Note the Bilateral threshold is used during this calculation.
- Max - Substitutes the current value for the maximum value seen in the last X and current values.
- Min - Substitutes the current value for the minimum value seen in the last X and current values.
- Midpoint - Substitutes the current value for the midpoint value seen in the last X and current values. The midpoint value is calculated by determining the average of the max and min value of the last X values.
- Peak - Determines the value of the last peak seen in the last X values. A peak is defined as an increase followed by a decrease in value around a single point. Due to this the current value can NEVER be a peak as the definition requires at least one previous and one next value around a point to determine if it is a peak or not.
- Valley - Opposite of a peak, a valley is defined as a decrease followed by an increase in value around a single point. Again, the current value cannot be defined as a valley as the next value is not yet known.
- Change - A combination of a peak or valley that indicates the last value when either a peak or a valley was detected.
- Line - Calculates a line that best fits the last X values and uses that equation to determine what the current value would be if placed on that line.
- Delta - Calculates the difference between the current value and the average of the last X values.

See Also

[Set Variable](#)
[Chart Variables](#)

Format Date/Time

The "Format Date/Time" module provides a way to format a datetime value stored in a variable into another. For example, you specify an

THE FORMAT DATE/TIME MODULE provides a way to format a datetime value stored in a variable into another. For example, you specify call IMAGE_DATETIME as the input variable, then select or type what output variable you want to use and then the date format type desired. The module will take care of parsing the date/time and format that as specified. Note that the output variable can then be displayed, saved, etc. within other modules.

Interface

The screenshot shows a dialog box titled "Format Date/Time Variable". It is divided into two main sections: "Variables" and "Format".

- Variables:** Contains two dropdown menus. "Input Variable" is set to "IMAGE_DATETIME" and "Output Variable" is set to "test".
- Format:** Contains a list of radio buttons for predefined date/time formats:
 - Monday, December 18, 2006
 - Mon, Dec 18, 2006
 - 12/18/2006 10:00:00 PM
 - 2006/12/18 10:00:00 PM
 - 10:00:00 PM
 - 22:00:00
 - CustomThe "Custom" option is selected, and its text field contains "MON Month D'Y". To the right of this field is a ">>" button.

At the bottom of the dialog are three buttons: "Help", "OK", and "Cancel".

Instructions

1. Input Variable - The variable that holds the datetime value that is to be formatted. If this is empty the current time is assumed.
2. Output Variable - The variable that will contain the formatted date string.
3. Format - Select the time/date stamp format that you would like to use. If you do not find an appropriate format select the Custom radio button and type in the appropriate format string. You can use the button to the right of the custom text field to select appropriate date/time strings. Note that any text not recognized as a time/date placeholder will be printed as text.

The following outlines the time/date stamp placeholders that can be used:

- SPACE
- DASH
- M - month number
- MM - month number - zero padded
- MON - 3 letter monthN
- Mon - 3 letter monthN2
- mon - 3 letter monthN3
- MONTH - full monthNTH
- Month - full monthNTH2
- month - full monthNTH3
- DDD - days in year
- DD - days in month - zero padded
- D - days in month
- LD - last day in month
- DW - day in week
- DY - 3 letter weekday
- Dy - 3 letter weekday
- dy - 3 letter weekday
- DAY - full weekday
- Day - full weekday
- day - full weekday
- YYYY - 4 letter year
- YY - 2 letter year
- WW - week in year
- W - week in month
- HH - hour
- HH24 - 24 hour
- MI - minutes

SS - seconds
MS - milli-seconds
PM - AM / PM indicator
pm - am / pm indicator

See Also

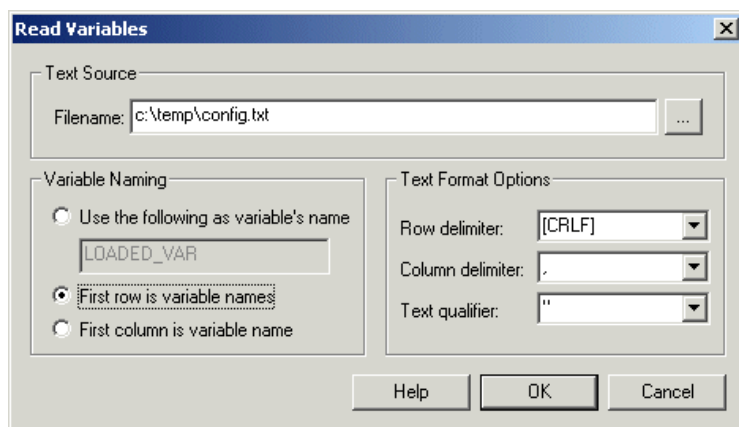
[Display Date Time](#)
[Display Variables](#)

Read Variables

The Read Variables module allows you to read in RoboRealm variables from the disk drive in various formats. This can be used to load configuration or discrete data into RoboRealm for use with its modules. Be aware that this module only reads in the file once it has changed on disk. If you change a variable that has been read in from disk it will be overwritten once the file changes and is re-read into RoboRealm.

For more direct methods of changing variables within RoboRealm see the [API](#).

Interface



Instructions

1. **Filename** - Specify the file that contains the information to be read into RoboRealm. Note that RoboRealm does NOT enforce a filename extension as it is expected to be a text file of some sort.
2. **Variable Naming** - Select how the name of the variables should be determined from the text file
 - Use the following ... - This assumes the file contains no variable names and thus you need to specify the one variable name that will be used to contain the text information in the specified file. This is typically used to bulk load in a string or sequence of numbers from a file into a single variable.
 - First row is variable names - Specifies that the first row in the text file contains the variable names to be used to contain the data in the subsequent rows. This format is typically for CSV (comma separated files) that are typically produced from programs like Excel. This format is effective for variables that have lists of data.
 - First column is variable name - Specifies a file format whose first column is the variable name followed by a delimiter and the actual variable value. This structure is similar to the INI files found in Windows and is effective for one-off variables that contain a single value/string. This is most commonly used as a configuration format.
3. **Row delimiter** - Specifies the delimiter (separator) that indicates the start of a new row (record). This will typically be CRLF (linefeed, carriage return) for most text files but may differ occasionally.
4. **Column delimiter** - Specifies the delimiter (separator) that indicates the start of a new column. This is typically a comma for CSV type files but might instead be a space for INI type files. (The variable name is followed by a space which is followed by the value which may include spaces too.)
5. **Text qualifier** - Specifies the symbol that groups text together disregarding the earlier delimiters. For example, a value in a CSV file may have a comma in it which would normally mean that a new column has been indicated unless the value is surrounded by quotes. The quote is the text qualifier which ensures that the value is specified outside of the row and column delimiter criteria.

Example

For the following data file:

```
X,Y,Z  
100,100,100  
200,200,200  
300,300,300
```

Specify that the first row contains the variable names, the row delimiter is CRLF, the column delimiter is a comma and the text qualifier is not used. This would create three variables with a list of 3 items each.

Next:

```
X 100  
Y 200  
Z 300
```

Specify that the first column contains the variable names, the row delimiter is CRLF, the column delimiter is a space and the text qualifier is not used. This would create three variables each with a single value.

Next:

```
option1 some option text  
option2 some more text
```

Specify that the first column contains the variable names, the row delimiter is CRLF, the column delimiter is a space and the text qualifier is not used. This would create two variables each with a single line of text. In this case even though the column delimiter is set to a space the values of the variables are NOT split. In other words only the first column delimiter is respected when using this style of data file.

Next:

```
100,200,300
```

Specify a variable name such as LOADED_VAR, the row delimiter is empty, the column delimiter is a comma and the text qualifier is not used. This would create a single variable (LOADED_VAR) with three values.

Next:

```
this is some text.
```

Specify a variable name such as LOADED_VAR, the row delimiter is empty, the column delimiter is also empty and the text qualifier is not used. This would create a single variable (LOADED_VAR) with a single text string.

See Also

[Write_Variables](#)

Set Variable

The Set Variable module allows you to assign other variables, time settings, or direct number/text values to variables. The module will also create new variables and assign them the specified value. You can use this module as a quick way to assign values to variables that can be used within the [IfStatement](#) or in VBScript modules.

Interface



Variable:	Value:	Set Once
my_new_var	test	<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>
		<input checked="" type="checkbox"/>

Help OK Cancel

Instructions

1. Select the variable to set the value of. If the variable does not yet exist simply type the name into the provided combo box.
2. Type in the value to assign to the variable. If you wish to assign the value of another variable to this variable then select that variable in the dropdown list as [variable name]. If you wish to set the variable to a particular time do so by selecting the appropriate time specification with [].
3. If you wish to set this variable as an initial configuration but then allow the value to change on each successive evaluation of the pipeline then select the "Set Once" checkbox. This checkbox will ensure that the variable is only assigned the specified value ONCE on program start. If unselected a variable will be set each time a new image is processed and will overwrite the current value.
4. Is Array - Specifies that the value for the variable is an array. Each item is separated by a comma. On assignment this list is read, split into individual elements and assigned as an array.

Variables

Depends on the configuration. All new variables specified are assigned the given values.

See Also

[Watch Variables](#)
[If Statement](#)

Scale Variable

The scale variable module provides a quick and easy way to convert a single variable from one range to another. This is frequently needed when associating variable values from an input device to an output device that do not share the same value range. For example, the joystick module produces values that range from -1000 to 1000 and but most servo controllers use a range from 0 to 255. You can use this module to convert between the two ranges by specifying the two ranges.

Interface

Scale Variable [X]

Source

Source Variable: test

Minimum Value: -1000

Maximum Value: 1000

Destination

Destination Variable: test

Minimum Value: 0

Maximum Value: 255

Help OK Cancel

Instructions

1. Source Variable - The variable that contains the value you want to convert. (Most likely JOY_X in the example above.)
2. Min/Max Value - The expected minimum and maximum value the variable will range between. In the above example this would be -1000 and 1000 respectively.
3. Destination Variable - Where you want the result placed. Note this can be the same variable as the source variable.
4. Min/Max Value - The destination range that you want to convert the value into. In the above example that would be 0 and 255.

Notes


The formula used to perform the conversion is

```
source_range = source_max - source_min
```

```
destination_range = destination_max - destination_min
```

```
destination = (((source - source_min) * destination_range) / source_range) + destination_min;
```

Example

 [Click Here](#) to download a robofile that shows how to use the Scale Variable module inbetween a joystick and SSC module.

See Also

[Differential Drive](#)
[VBScript Program](#)
[CScript Program](#)
[Python Program](#)

Sort Variables

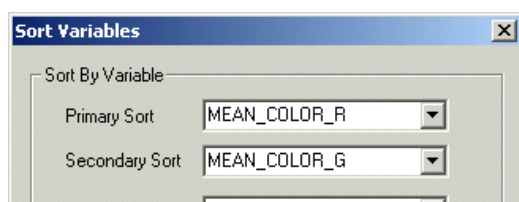
The Sort Variables module accepts three variables to which to sort by. The Primary, Secondary and Tertiary variables are used to create a sort index that can then be used to sort additional variables in order to keep the relative position the same within different arrays. This module is useful if you need to reorder arrays produced from modules like the [Geometric Statistics](#), [Color Statistics](#), etc.

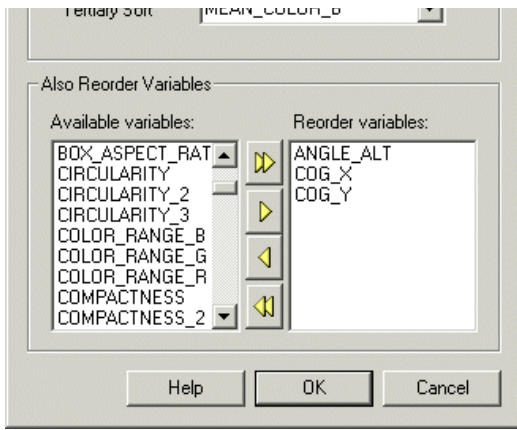
Using this module you can sort based on any single blob array within RoboRealm. For example, you can sort arrays based on an attribute that will create the same ordering based on the color. The interface below shows sorting based on the average color of the blobs. This will always sort in the same order with white (255,255,255) being the last object and blue being the first object. Due to this forced ordering you can now confidently access the first entry of any sorted variable and know that it refers to the blue object.

Care needs to be taken when choosing the attribute to order on. Color works but only if you have used the [Color Filter](#), [RGB Filter](#), [Replace Blob](#), [Sample Color](#), etc modules to ensure that the colors are replaced with solid. If the colors of two objects are very close they may sometimes sort in order but may other times sort out of order. There would be no guarantee that the order is always the same.

Note that this module helps to solve the object id order issue. When processing an image there is NO guarantee that an object's index within a statistical array will be the same from one frame to the next. If it appears to be that is largely due to luck. You need to use a feature/characteristic/attribute of the object compared to the others in order to ensure that the ordering is the same. These features can be color, size, location, etc. For more complex tracking of objects see the [Blob Tracker](#).

Interface





Instructions

1. Primary Sort - Select the variable array to use as the primary ordering.
2. Secondary Sort - Select the variable array to use as the secondary ordering. The secondary ordering is sorted before the primary sorting such that values that are the same in the primary sort will be sorted with respect to the secondary values.
3. Tertiary Sort - Select the variable array to use as the tertiary (third) sorting. Values that are the same for the secondary and primary sorting will then be ordered as specified by the values in the tertiary array.
4. ASC/DESC - Select which direction you want the sort order to be. Use ASC for ascending (increasing) and DESC for descending (decreasing).
5. Also Reorder Variables - Select those variable arrays that you want to also be reordered based on the sorting results of the Primary, Secondary, and Tertiary arrays. These arrays will then keep in sync with those values in the sorted arrays such that an index to a value before the sort will contain the same correlated values in all other arrays.

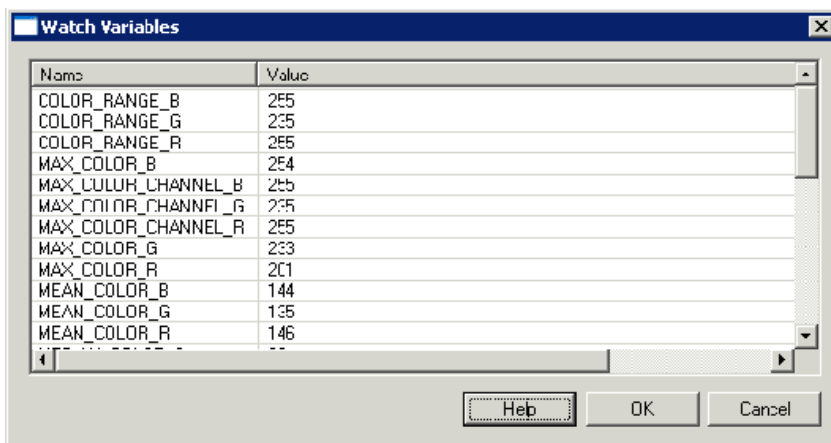
See Also

- [Blob Tracking](#)
- [Filter Variables](#)
- [Chart Variables](#)

Watch Variables

The Watch Variables module allows you to peek into the current variable state being maintained by RoboRealm. This module will list out all current variables that are accessible by other modules and plugins.

Interface



Instructions

1. Insert the watch variable module where you would like to watch a variable's value. Note that the placement of the module is relevant. If variables

1. Insert the watch variable module where you would like to watch a variable's value. Note that the placement of the module is relevant. If variables change before the module then the change will be reflected in the list, however, if a variable changes AFTER this module then the change will not be reflected until the next pipeline iteration after a new image capture.

See Also

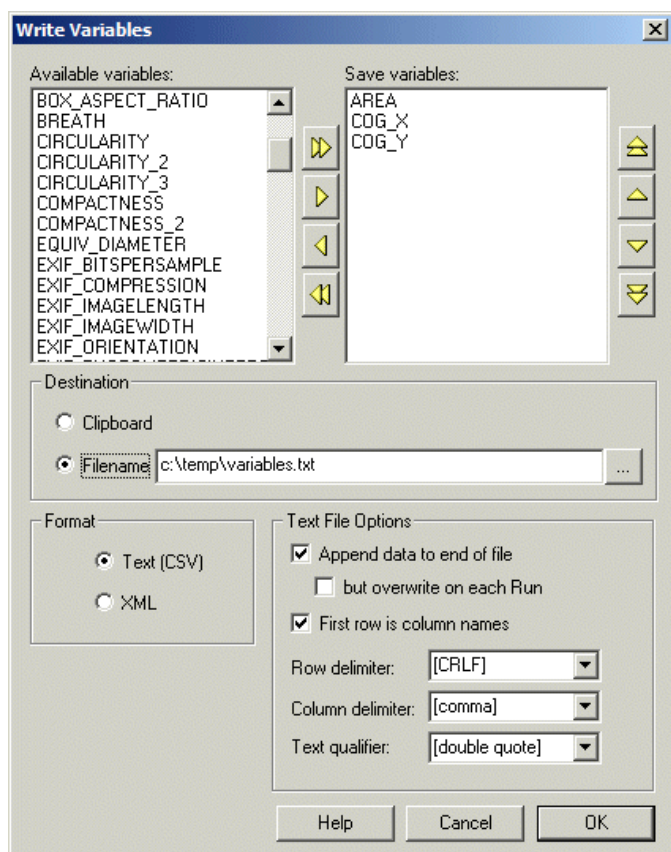
[Set Variable](#)
[Chart Variables](#)

Write Variables

This module provides an interface to write RoboRealm variables to disk. The interface allows the selection of the variables that will be saved. Note that the order specified in the interface will be the order saved on disk.

This is the simplest way to export variables from RoboRealm. For more advanced, efficient and interactive methods see the [Plugins](#) documentation.

Interface



Instructions

1. Available Variables - Select the variables that you want to save. The left list shows all the current RoboRealm variables available to save given the current processing pipeline. Use the arrows (or doubleclick on a variable) to move it to the 'save variables' list. Use the up/down arrows to reorder that list.
2. Clipboard - If you want to write the values to the Windows Clipboard select this radio button. Note that you can test this by pasting into a text editor to see the values RoboRealm is placing into the Clipboard. Be aware, this overwrites any existing text in the clipboard. Variables are written as name value pairs with each on its own line. Thus you can use a newline (\bLF or \n) to break the text string into individual parts.
3. Filename - Specify which filename you would like the variable values to be saved to. Be sure to have created any needed directories/folders along the filename specification otherwise the module will NOT be able to update the file.
4. File Type - If you chose a text file select if you'd like the variable values to be appended to the end of the file. If chose append you should be ready for the file to become really large really quickly!
5. Text File Options - Specify the appropriate row, colom and text delimiters. The text delimiter are used when a variable containing a text value (non-numerical) is saved. You can also type in characters to use as delimiters. Using the default delimiters will allow you to load the file directly into Excel or other spreadsheet programs.

You can chose to append data to a single file (but erase the file between runs) or just overwrite the entire file out each time the frame is processed.

Note that when reading the saved file you will need to check for the end of row delimiter (text file) or the </ROBOREALM> tag (xml file) to ensure that you have read a complete file. Due to the speed of the writing (once per frame) you are not assured that reads are atomic. RoboRealm is configured to write data in one write to help reduce this issue.

See Also

[Write Images](#)
[Plugins](#)

3D Viewer

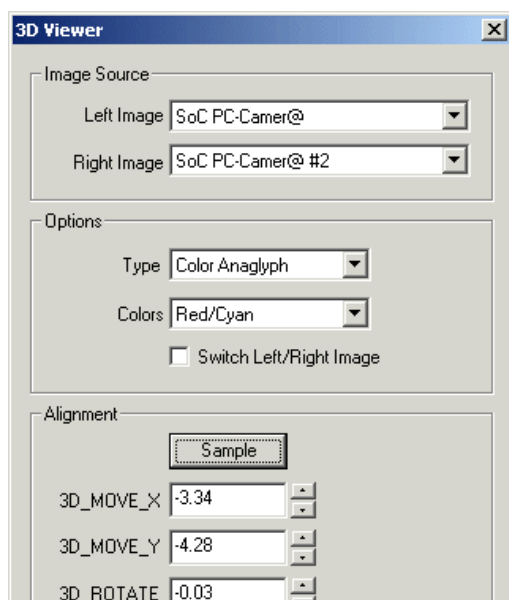
The 3D Viewer module provides you a way to better visualize three dimensional image captures. The setup requires two webcams (or other image sources) placed about 6.5cm apart. The webcams then feed two images into this module which then produces a single image based on your

specification. The options below provide a wide format choice in order to determine the format that best suits your need.

Note that this module is for viewing ONLY and does not create stereoscopic depth maps.

Also note that the images created by this module can be exported as normal by other modules or distributed by other built in mechanisms such as the webserver.

Interface





Instructions

1. Left Image - Specify the left camera source
2. Right Image - Specify the right camera source
3. Type - Specify the 3D image type you would like to create from the two images.

Color Anaglyph - produces an image with the left or right channels combined into the color channel of a single image. You will need the red/green, red/blue, etc. colored lens glasses to see the actual 3D image.

Half Color Anaglyph - produces an image similar to the color anaglyph but reduces the red channel in such a way to reduce retinal rivalry. Retinal rivalry occurs when one eye sees a more dominant color than the other and causes a ghostly type of image.

Optimized Anaglyph - similar to a Half color Anaglyph the Optimized Anaglyph attempts to reduce retinal rivalry.

Gray Anaglyph - produces an anaglyph using gray scale which also reduces retinal rivalry but removes all color from the image.

True Anaglyph - Similar to the above techniques but creates a pure left and right color channel whilst removing the third unused channel. This causes the image to look visually different when not using the Anaglyph glasses.

Flicker - Causes the two images to flicker back and forth between each other. This helps the eye to understand the movement between the two images and can help to determine alignment or rotational issues. No glasses are required for this mode.

Side by Side - Produces a single image with both camera images displayed side by side. This type of image requires a cross-eyed viewing or separate eye channels to view. Often this can be achieved by placing a divider (such as a piece of cardboard) between each eye with the cardboard extending to the screen ending inbetween the two sides of the image. This trick helps to create the cross-eyed effect that one needs to create a 3D image.

Interlaced - Creates an image where one side is interlaced with the other. This format requires the use of shutter glasses that are synched with the screen refresh rate. One interlaced image is seen by one eye while the other is covered (thus the need for shutter glasses) with the shutter switching back between left and right to alternate the eyes. This causes a 3D image to be seen as the right eye sees a different image than the left.

4. Colors - For the color anaglyphs you can specify what color channels are used to contain the image. The default and most common will be Red/Cyan as this works for most colored stereoscopic glasses but other options are available in case you glasses require different channels to be used.

5. Flicker Rate - When the flicker mode is enabled this specifies the millisecond time between flickers or the visible time for each image before switching to the next. Smaller values approaching zero will cause the flicker to speed up. Larger values will slow the switching down. Note that at 0 the flicker rate will be dependent on that of the pipeline speed. Thus if the overall pipeline is processing at 10fps then the flicker rate cannot be less than 100ms per image. You can specify values lower than the pipeline fps but the flicker rate will not increase.

6. Switch Left Right - If the resulting image does not appear 3D or confuses your eyes try to switch the two images by clicking on the switch left right image. This will swap the right and left image. As we cannot see stereo when the left camera is exposed to the right eye and the right camera to the left eye this checkbox allows you to quickly test which way round the cameras are relative to how you are viewing the image.

7. Alignment - Stereo viewing is very sensitive to the alignment of the two cameras. If you move one image out of phase of the other you will lose the stereo effect or cause your eyes (and head) to become very taxed to preserve the 3D viewing. As moving the cameras manually to align the images requires very small and precise movements it is easier to move the camera's into general alignment and then let the software do the rest.

The SAMPLE button will analyze 10 frames from your current setup in order to determine the best translation (x,y movement) and rotation (degrees) that will align the two images. The resulting information is then populated in the text boxes below and entered into variables (3D_MOVE_X, 3D_MOVE_Y, 3D_ROTATE) for other modules to refer to or for export into alternate systems. The current images are then aligned according to this information.

Note that before pressing the alignment SAMPLE button you should ensure that the scene is somewhat stable AND that most of the scene is at the desired neutral depth. Neutral depth is where the left and right image are the most similar with objects in front of this depth will appear in front of the screen whereas objects behind this depth will appear behind the screen (in 3D). Thus it is best to align on a scene which best represents the average depth that you intend to be using in your stereoscopic viewing.

Tips

1. Make sure your camera images are the same size. You can do this by selecting the Options Button->Video tab, selecting the appropriate camera and then clicking on the Format button. Within that dialog you should be able to change the camera size. Do this for both cameras.

- camera and then clicking on the format button. Within that dialog you should be able to change the camera size. DO this for both cameras.
- Place your camera about 6.5cm from each other and try to align them as best as possible. Note that even the slightest bump can cause them to go out of alignment unless they are secured to a shared structure.
 - The best 3D experience can be seen using shutter glasses but that is also the most expensive and most may not work on LCD screens that are polarized differently than the shutter glasses (also LCD) which can cause a permanent dark image on both eyes.

Example

The following shows the Traxster robot spotting a coke can.

Color Anaglyph



Optimized Color



Color Anaglyph



Half Color



Interlaced



Color Anaglyph



Optimized Color



To understand the flicker mode [here](#) is a 10 second video of the surveyor.

Variables

3D_MOVE_X - contains the X movement needed to align left and right images

3D_MOVE_Y - contains the Y movement needed to align left and right images

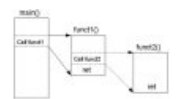
3D_ROTATE - contains the rotation in degrees needed to align left and right images

See Also

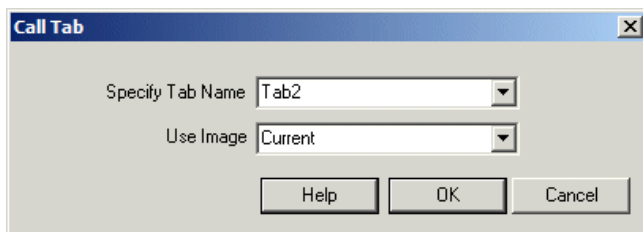
[Mosaic](#)

Call

The Call module provides a way to call a tab from another tab as if it were a function call. This allows you utilize tab pipelines as functions/subroutines that can be executed from another tab (presumably the Main tab).



Interface



Instructions

1. Tab Name - Specify the tab name that should be called. Execution of the pipeline will pause in the current tab, call the specified tab, execute its contents and resume automatically from the calling point.
2. Use Image - Specify what image the called tab should use for processing. If Current is specified then the tab will receive the current image at the point in the pipeline that the call is executed. If Source is used, the called tab will execute against the original image that was captured.

Tabs other than the Main tab are executed when:

1. You are currently in that tab (i.e. that tab is selected)
2. You've set it to executing using the tab properties (white button on the right of the tabs).
3. You are calling it from another tab using this **Call** module.

Note that to change which tab is called used [my_variable] in the Tab Name and set the variable my_variable to the name of the tab you want to run.

Examples

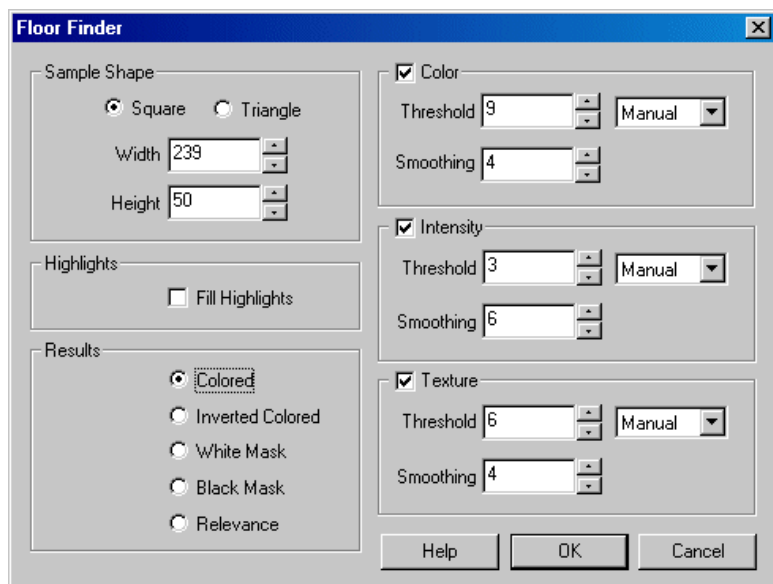
[Click Here](#) to download an example that calls the Red tab from the Main tab.

[Click Here](#) to download an example that calls the a tab based on which button you press.

Floor Finder

The Floor Finder module is used to identify the floor area within an image. The assumptions that make this possible are that the robot or camera is on a planar floor that extends from the bottom of the camera image outwards away from the camera. This assumption is required since the Floor Finder module will sample the pixels in the bottom of the image area (the sample space) and use those pixels to identify similar pixels in the rest of the image. Thus it is assumed that the floor space right in front of the robot is relatively free from obstacles and represents a portion of the floor. Keep in mind that different colored lighting or intensity levels on the floor can change the color and cause a break in the resulting extracted floor.

Interface



Instructions

1. **Sample Shape** - select the sample shape that you would like to use. The default is the square shape that will sample more pixels than the triangle shape but may be affected by darker pixels or non-floor pixels at the image borders. To reduce image border issues (depending on the style of camera you are using) you can switch to the triangle sample area that includes more pixels in the middle of the image.

You can change the width and height of the shape to include more or less pixels. The more representative pixels that are used in the sample space the better. Increasing the sample space will also include more pixels that are not floor space so a compromise needs to be reached between floor and object pixels.

2. **RGB** - Select this checkbox to enable RGB (red, green, blue) processing of the sample area. This is useful when the obstacles are better identified using a combination of color and intensity. Note that this works well on evenly lit areas but will fail in areas of shadows.

3. **Color** - Select this checkbox to enable color processing of the sample area. This works very well if your floor is a different color (hue) than the obstacles to be avoided. This works even when areas are somewhat shaded from direct light. The more colorful your carpet the better the color processing will work. The less color, the more you will need to rely on intensity or the combination of both in RGB.

4. **Intensity** - Select this checkbox to enable intensity (brightness) processing of the sample area. This works well if your floor is much lighter or darker than the obstacles to be avoided. But be careful that shadows may cause problems.

5. **Texture** - Select this checkbox to enable texture processing of the sample area. This works well if your floor has a very different texture (rough versus smooth) than the obstacles to be avoided.

6. **Color/Intensity/Texture Threshold** - Select the threshold level that specifies how many instances of a certain attribute need to exist in the sample area before they are considered 'representative' of the floor. If your immediate floor space includes obstacles setting the threshold higher will help remove those obstacle pixels from consideration. This, however, can also remove floor pixels that are slightly distance from the selected feature from the rest.

For a more adaptive way of setting the threshold chose a value from the dropdown (besides manual) to select only the top X percent of features seen in the sample area as a guide for the rest of the image. Setting this value will find which feature occurs the most frequently (its count) in the sample area and then set the threshold to be X percentage of that count. This allows the threshold to change based on how consistent the sample area is and change based on the image properties within the sample space.

7. **Color/Intensity/Texture Smoothing** - As the sample space is smaller than the rest of the image it often does not include features that are very close to features within the image space. Increasing the smoothing will widen the influence that features in the sample space have in the rest of the image. In this way a feature in the sample space can also affect other pixels that are near in color/intensity/texture to it and help fill out more valid pixels in the rest of the image. If you increase this too much it will encompass all features and not result in any image segmentation.

8. **Pixel Smooth** - Its best to smooth the image prior to actually checking pixel colors. Using the pixel smooth value you can smooth the image to help prevent small holes within non-obstacle areas.

9. **Allowed Break** - The final object free path is determined as non-obstacle pixels starting from the bottom of the image moving towards the top without hitting any obstacle pixels. As soon as it does it will remove all pixels higher than that point to indicate they are not part of the floor. Small groups of pixels can cause this to happen prematurely so the allowed break value ensures that there are many obstacle pixels grouped together before an obstacle is determined and higher pixels removed. If you set this value to zero you will see pixels similar in color to the sample area appear which would not be possible to traverse too.

10. Momentum - While the robot is moving it may accidentally look off the desired path which would immediate affect the module to look for the features within the sample area. To prevent accidental attachment to the wrong surface you can increase the momentum from 0 to a larger number like 200. This means that the module will remember the statistics from the previous 200 frames (a couple seconds at 30fps) and use that to look for the floor surface. Even if the robot accidentally moves over a different area that immediate frame will only marginally affect what the module is attracted to. Using this feature memory will help to stabilize the robot movement on the desired path.

11. Continuous - If you know that the first couple frames will contain exactly the model to use you can uncheck the continuous checkbox which will freeze the model after that number of frames.

12. Highlights - Often floor space (especially reflective floors) will tend to have white highlights in them due to overhead lights. These highlights have very specific properties that can be used to partially include them back into the floor space. As highlights are normally bright intense spots they are normally not included within the sample space beyond very low threshold values. Selecting the "Fill Highlights" checkbox will analyze the current image for highlights and fill them in with white pixels. This helps to complete a full floor space and eliminate the appearance of obstacles in the middle of the floor.

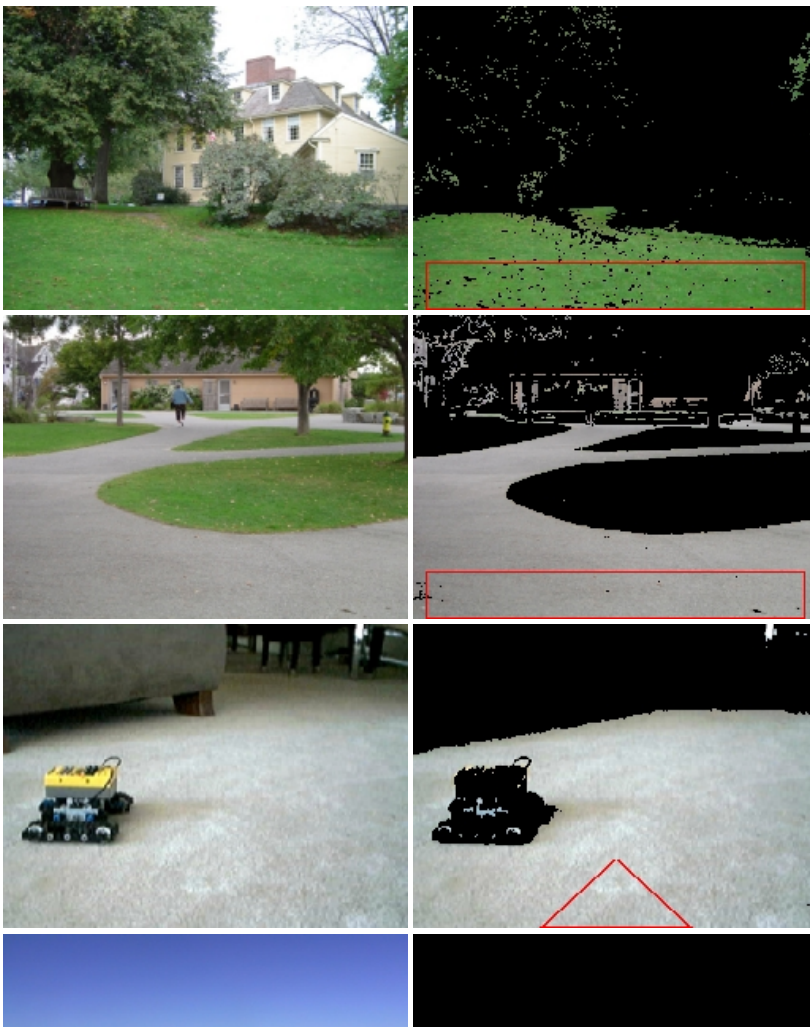
13. Result - select how the results should be presented. "Colored" refers to the original pixel values (excluding the highlights which are always in white). "White" or "Black" mask will set pixels to white or black depending on if they are represented in the sample space or not.

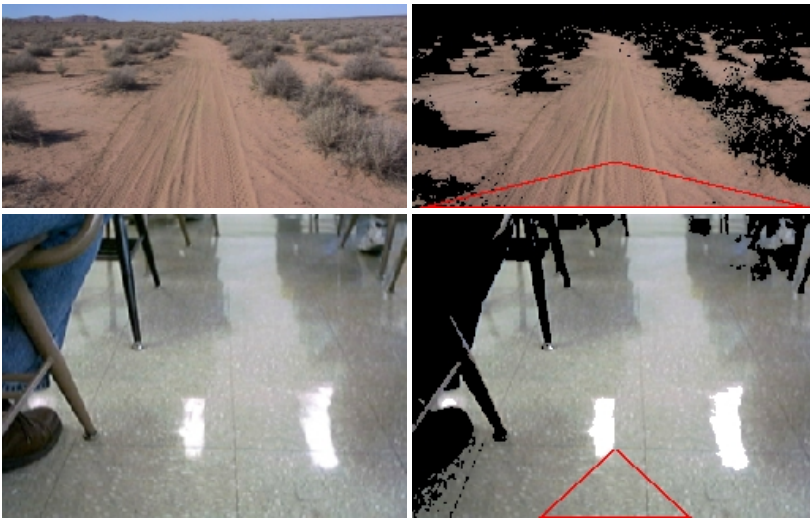
The Relevance Result will indicate from black to white how relevant a particular pixel is to the sample space given the smoothing and threshold values. Note that when more than one feature is enabled the relevance is the addition of all those features.

Examples

Source

Floor Found





Note that last image is the floor of a tiled reflective floor. The two large highlights are filled in using the "Fill Highlight" option.

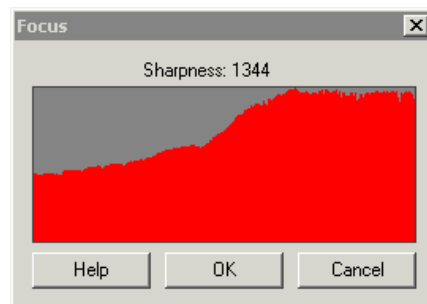
See Also

[Wall Finder](#)

Focus Feedback

The Focus Feedback module will help you to achieve the best focus for an image. The module displays a single number above a moving chart of that number. The number represents the sharpness of the image. Comparing the number with previous numbers one can determine if the image is getting more or less focused. Note that the defined sharpness WILL change when the image content changes. Therefore the module is ONLY effective when the content remains stationary with only the lens focus being changed.

Interface



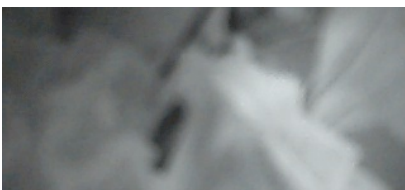
Instructions

1. Watch the progress of the chart while changing the focus of the camera. Note that you need to hold the camera VERY steady while doing this. Once the focus chart reaches an apex and begins to decrease you will know that you've just passed the optimal focus point. Reverse a little on the focus to ensure best results.

Example

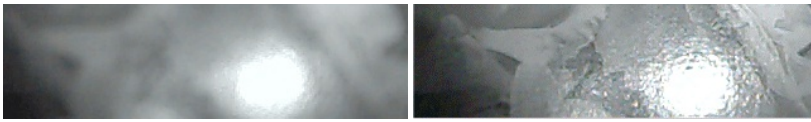
The interface image above displays the processing from the below starting image to the final ending image. You can easily see that as the image focus improves the chart trends upwards. Note that the absolute number of the focus is not important, only the relative increase or decrease is.

Starting Image



Ending Image





Variables

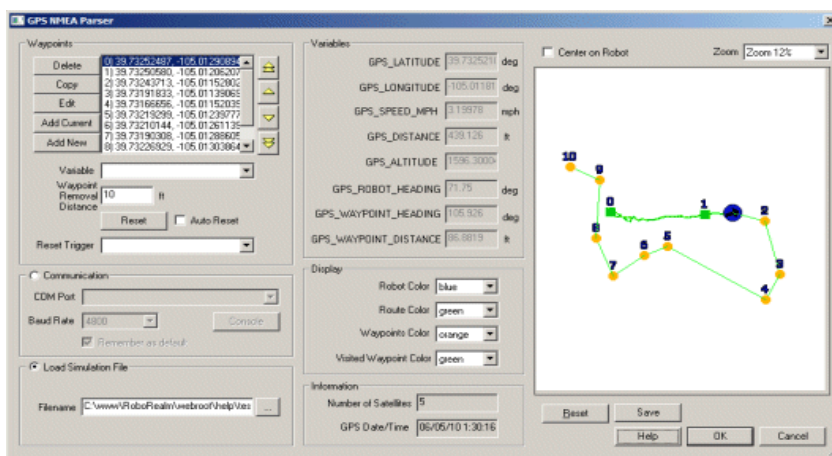
FOCUS_VALUE - the focus value created by this module. This variable is accessible from the VBScript or other modules as appropriate.

GPS Reader



The GPS Reader module provides a way to interface RoboRealm to most NMEA GPS devices that can be hooked up to a PC using either a USB or serial connection. GPS devices provide a wealth of information including latitude and longitude that can be used to localize where your robot is, it's heading and where it should go. The GPS Reader module interprets the NMEA text from the GPS device to provide you with a visible interface of the course your robot has taken and allows you to specify waypoints that can be used to create a path. Utilizing the variables produced by this module you can then steer your robot towards its desired path.

Interface



Instructions

1. **Communication** - select the appropriate COM port that your GPS is connection to. Note that 4800 baud is set as a default communication rate which most GPS devices will use.
2. **Load Simulation File** - if you do not currently have a GPS device available (or are located indoors) then you can instead specify a file that contains recorded NMEA statements. Once specified this file will be read in and acted on just as if the information was being passed from a GPS device.
3. **Map** - the white area will display the points specified by the GPS device in black. The current robot location is specified in blue with the waypoints in orange and the planned route in green. If you uncheck the "Center on Robot" checkbox you can move the map around using the mouse to investigate the route. You can also use the zoom dropdown to move towards and away from the map. If you want to clear the map and start from a blank page press the Reset button. If you want to use the map for your own purposes you can use the Marker module to pull in the GPS_MAP image that is produced by this module and kept in memory.
4. **Information** - The number of satellites and date received by the GPS is displayed in the information section. Use this interface to determine how well your connection is to any satellites. Remember, if you go indoors you will lose all connections.
5. **Waypoints** - The main purpose of using GPS is to be able to steer the robot along some known path. Using the waypoints interface you can enter in a new Latitude and Longitude coordinate such as 5408.0079 and 01345.6965 and add that as a waypoint. Note that the first 2 and 3 digits

are the location degrees with the remainder being seconds. This is the format produced by most NMEA GPS devices. Once this waypoint is added into the list you will see it displayed in the map and additional variables will become active. The two most important variables are the `GPS_ROBOT_HEADING` which gives you your current heading and `GPS_WAYPOINT_HEADING` which is the heading you should be on in order to meet with the waypoint. Using these two variables you can decide on how to move the robot to get to the waypoint. Once the waypoint is reached the module will automatically switch to the next.

6. Waypoints Variable - You can also specify a variable that will contain the list of waypoints. Please note that this variable needs to be created from VB or Python using the `SetArrayVariable` function in order for the waypoints to be accessed correctly. It is assumed that this array includes sets of latitude and longitude numbers (i.e. lat1, lon1, lat2, lon2, etc.) that have been converted to a decimal degree. This is a different format than that used in raw NMEA data (degree minute second) but is what is displayed in the Waypoints list. When you add a new waypoint, you can type in the raw NMEA (degree minute second) coordinates that then get converted into a single decimal degree number.

7. Waypoint Removal Distance - The Waypoint Removal Distance (in feet) defines how close the robot needs to be in order for the waypoint to be considered visited. Because GPS is inherently noisy, this number should be large enough to allow for a certain amount of error. If you notice that your robot is searching and missing waypoints (i.e. it turns around and approaches the same point again) you may have to tight a waypoint removal distance. Likewise, if you notice that the robot does not get near enough to a particular waypoint and seems to bypass it then reduce the removal distance amount.

8. Reset - Resetting the waypoints by pressing the Rest button resets the waypoints that have been visited back to zero and repeats the process. I.e. your robot will now head to waypoint 0 again.

9. Auto Reset - You can chose to loop again and again through waypoints (by selecting the Auto Reset checkbox). This allows for a patrolling of waypoints that once complete, should trigger the first waypoint to be visited again.

10. Reset Trigger - Specify a variable that when non-zero will cause the waypoints to reset in that the robot will then target waypoint 0 again. This is functionally the same as pressing the Reset button but instead happens based on a non-zero value of the reset trigger variable. For example, if you have an external application that wants to reset the waypoints and start from the beginning it can send the specified variable to RoboRealm with a non-zero value (say 1) via the [API](#). This variable can change can happen at any time to reset to the first waypoint.

11. Robot Color - the color marker used to identify the robot (circle with an arrow)

12. Route Color - the route between waypoints

13. Waypoints color - the color of the square that identifies the targeted waypoints

14. Visited color - the color of the waypoints that have already been visited

Variables

`GPS_LATITUDE` - the current GPS latitude value in degrees

`GPS_LONGITUDE` - the current GPS longitude value in degrees

`GPS_DISTANCE` - the total distance in meters that the robot has traveled

`GPS_ALTITUDE` - the current height of the robot (note this is typically very inaccurate)

`GPS_ROBOT_HEADING` - the current orientation of the robot, i.e. direction it is moving

`GPS_WAYPOINT_HEADING` - the orientation of the waypoint relative to the robot

`GPS_WAYPOINT_DISTANCE` - the distance in meters till the next waypoint

`GPS_WAYPOINT_COMPLETE` - signals (becomes 1) when ALL waypoints have been visited

`GPS_WAYPOINT_CURRENT` - the current waypoint being targeted

`GPS_WAYPOINT_REACHED` - signals (becomes 1) when the current waypoint has been reached.

Note, this will remain 1 until you clear the variable as it only gets set when a waypoint is reached and is NOT cleared. This allows you to perform other tasks at a waypoint and continue targetting the next waypoint when you chose to by setting it to zero.

`GPS_NUMBER_SATELLITES` - the number of satellites currently being used by

the GPS device

GPS_DATE_TIME - the date time stamp as reported by the GPS satellites

If you need a sample GPS NMEA file to run via the simulation you can download [this file](#), save it to a known location and then specify that location in the File Simulation in the GUI dialog. This file is the raw NMEA recorded from a GPS device while walking around the block in Denver, Colorado.

See Also

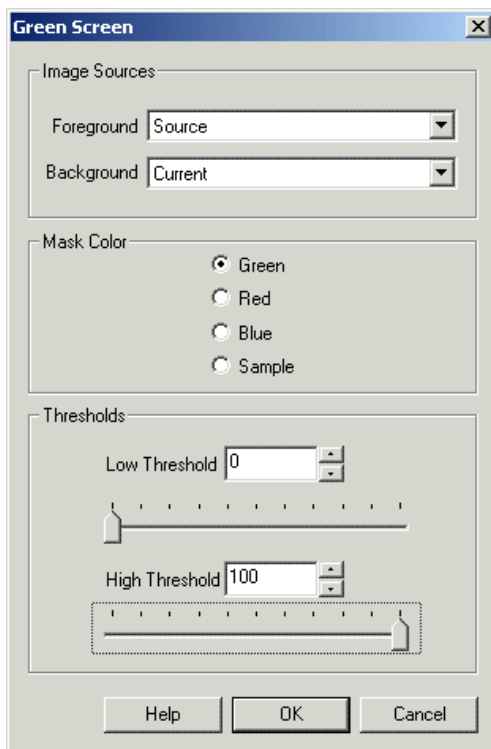
[Path Planning](#)

Green Screen

The Green Screen module provides a way to filter out a GreenScreen (or BlueScreen) and replace that color with another graphic. This filming technique is often used in news broadcasts to provide a news announcer in front of weather graphics or other animations.

Be sure to have adequate lighting of the greenscreen with no shadows. Also be sure not to stand too close to the screen >5 feet otherwise you will cast shadows and cause bleeding of the green color around your subject. Always light the greenscreen and the subject using two lighting sources in order to remove shadows and reduce color bleeding.

Interface



Instructions

1. Foreground - Select the image that contains the green, blue or red color to replace.
2. Background - Select the image that will be placed into the green, blue or red area in the foreground.
3. Mask Color - Select which color serves as the mask. Normally this will be green or blue. As red is very close to human skin you should not use Red unless you are screening something else other than people.

Chose Sample Most Common if your color in use is not a primary color. This will sample the current image for the most common color and use that as the mask color. This will NOT work well if there is no dominant color in the current image. Note, this samples every image!

4. Thresholds - As lighting and background colors are not always perfect in realistic conditions the threshold values are used define the transition range from background to foreground. Setting these thresholds correctly will allow your foregrounds and backgrounds to be sharper while the transition color range will serve to fade the background into the foreground.

Example

Foreground



Background



Result



Bad Foreground



Result



Bad Result with Adjusted Threshold



Comments

As you can see from the second foreground it is very important to get the green color to be as close to perfect green as possible. This is very problematic as the above images use the SAME background color but two different cameras. Each camera will have a different response to green and thus you will need to find the right green for your particular camera. Otherwise a faded background will occur as seen in the above images.

If you find this to be the case try adjusting the threshold values to compensate for the irregular color or lighting conditions.

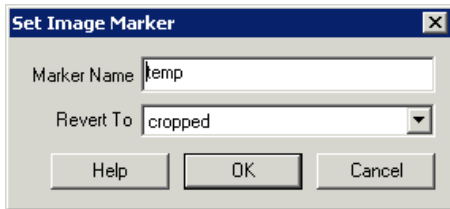
See Also

[Mask](#)

Marker

A Marker provides you with a way to identify or label a spot within the processing pipeline. Adding a marker into the pipeline will allow you to refer to the image currently being processed at that point. It can also be used to specify a location in the pipeline for other modules.

Interface



Instructions

1. Specify the marker name. This name will appear in other modules as needed to refer to this spot within the processing pipeline.
2. Select which image you would like to switch.

Source - the source image that was initially loaded into RoboRealm
Current - the currently processed image within RoboRealm
CameraX - a list of attached and active USB camera devices
MarkerX - a list of created marker images using the this module.

The Marker labels represent images at the time markers were created. If you wish to process the image at a certain point within the image processing pipeline create a marker at that point. The marker will then be included in the dropdown image list.

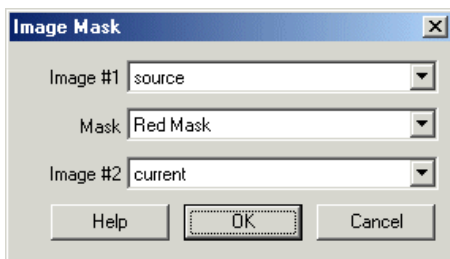
See Also

[Image Math](#)

Image Masking

The Mask module allows you combine two images together based on the intensity of another image (i.e. the mask). Using this module you can create greenscreen, bluescreen, etc effects that combine live images with background still images.

Interface



Instructions

1. Select image #1 to mask. This image will typically be the source image that is acquired using your webcam or digitizer.
2. Select the mask to apply to #1 before merging with image #2. The mask is an image that has typically been thresholded or processed in some way as to indicate a certain area within image #1. For example, an image with all pixels below intensity 50 removed can be used as a mask.
3. Select image #2. This is the image that is combined with image #1 based on the mask to create the final combined image.

The 'source' and 'current' image selections are always present. *source* always refers to the image that is either loaded into RoboRealm or is acquired from your video camera. *current* always refers to the image that has been processed up to this point in the processing pipeline. Other names

that are seen in the dropdowns come from [markers](#) that are setup elsewhere in the pipeline that indicate different images or images at a different processing step.


Example

Source



Red Masked



 [Click here](#) to download the robo-file that converts the image to gray except for red areas.

Movement/Motion Detection

The Movement detection module provides a way to detect image changes as a result of movement. Movement can be an easy way to segment an object of interest from the background.

Any pixel that is different enough in color or intensity is preserved; pixels that have not changed are set to black (default). To change the sensitivity of this comparison increase or decrease the 'Difference Amount' within the movement dialog interface.

To create a movement mask (i.e. set changed pixels to white) click to 'Set Movement to White'.

If you want to show the pixels that are still uncheck the 'Set Still to Black'.

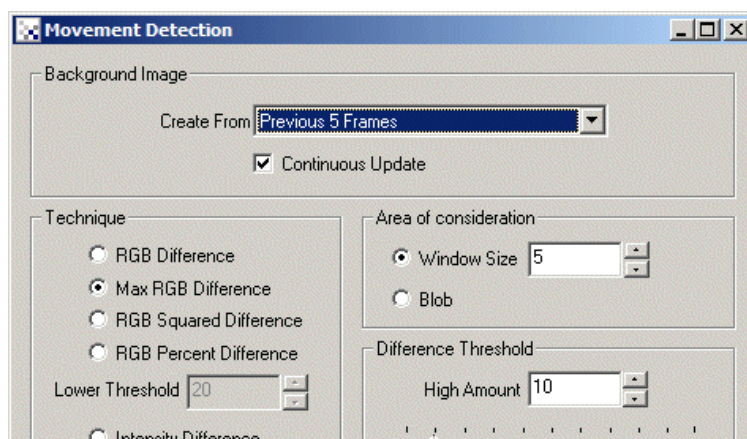
To prevent still images from being further processed you can chose to stop processing of the image processing pipeline if the movement detected is below a certain percent of the image size or if no movement is detected above a certain threshold. Setting either will stop image processing and continue with the next frame grabbed from the video source. This is particularly useful when streaming the video source to an AVI file where you only want to record images with movement and not still images.

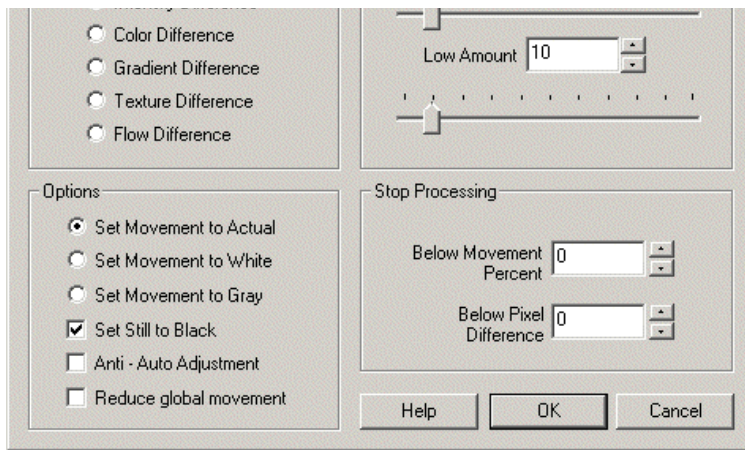
For global movements or large objects use the 'Stop below Movement Percent' as this ensure that small image aberrations do not make it through. Often due to overhead lights or specula effects pixels can abruptly change for an instance to a different color/intensity. However, if small movements need to be detected use the 'Stop below Pixel Difference' which will stop processing as long as all pixel differences are below the specified amount. With this condition even a single pixel above the specified value will allow the image to continue thought the processing pipeline.

Unchecking both 'Set Movement to White' and 'Set Still to Black' is a great way to stabilize a still image and remove the flickering effects due to overhead lighting or poor quality cameras.

Note that using a mean filter before the movement detected is often recommended as another way to reduce any abrupt pixel changes. Spurious pixel values will be more common on lower quality cameras so the values you use may need to be updated when changing cameras.

Interface





Instructions

1. Background Image - Select the reference frame to use when comparing frames for movement. The more frames used the more stable the reference image will be but changes are detected slower.
2. Continuous Update - Specify if the reference frame should be continuously updated or stop after the specified number of frames have occurred.
3. Threshold - Specify the pixel difference amount that needs to be exceeded in order for movement to be detected.
4. Technique - Select which type of comparison you want.

RGB Difference - sum of differences of each pixel RGB channel

Max RGB Difference - sum of maximum red or green or blue difference between pixels

RGB Squared Difference - sum of squared 'RGB Difference' that helps to better reduce low noise

RGB Percent Difference - sum of percent difference between RGB pixels. Since darker pixels contain more noise use the threshold value below to ignore pixel values lower than the specified threshold. The percent mode helps to eliminate white or black background pixel issues.

Intensity Difference - sum of intensity differences only (grayscale comparison)

Color Difference - sum of CrCb (chroma) values. Note that color differences are VERY small so you will need to set the 'Difference Threshold' to a very small amount (1 - 3)

Texture Difference - a small window is used to calculate a measure of texture within the window. This amount is also calculated in the background image. The difference of the two is then used as the threshold value. This allows you to tell when a part of an image's texture (i.e. smooth or not smooth) changes. Note that this is less lighting sensitive as the texture measure is relative to the local window.

Flow Difference - the translation of a pixel is summed over the entire image. The translation of a pixel is defined as the location in the next image that is closest in value to the previous pixel. This is also known as [optical flow](#)

5. Set Movement to Actual - sets any detected movement to the current pixel values.
6. Set Movement to White - sets any detected movement to white. This can be used to create a movement mask.
7. Set Movement to Gray - sets any detected movement to an intensity that represents the amount of detected movement.
8. Set Still to Black - sets any non-movement pixels to black.
9. Anti - Auto Adjustment - if your camera does not provide the option to freeze white balance or auto exposure select this checkbox. The current image's lighting will then be adjusted (reversed) to the previous image's lighting parameters to ensure that comparisons are done within the same lighting conditions. Ideally you can switch these off on the camera to avoid sudden light adjustments that cause a global difference in images.
10. Reduce global movement - if your camera moves in any way the entire image may suddenly not compare well with the previous image and cause a large amount of detected movement. This checkbox activates a method to attempt to reduce this issue such that only local movements within the image is detected. Note, this method is CPU intensive and should only be used if needed.
11. Area of consideration Window - for all techniques checking a single pixel at a time will typically cause a lot of noise to be produced depending on the camera that you are using. Normally a single pixel's color/intensity will change significantly from one image to the next unless you are using a higher end machine vision camera. Because of this, it is best to sample a group of pixels (i.e. small window) and run the comparison against that group of pixels rather than an individual one. The Window Size parameter allows you to specify the size of this window. Smaller windows will

produce more noise, larger windows will filter out smaller changes.

12. Area of consideration Blob - similar to the Window Size this selection will group pixels of similar color together before running the analysis. This will ensure that objects comprised of similar pixels will be analyzed together.

13. Difference Threshold High Amount - specifies the threshold amount for many of the techniques. The lower the value the smaller the difference between the images need to be in order to signal movement. The higher the value the larger the difference needs to be in order to signal movement.

14. Difference Threshold Low Amount - once a pixel is determined as being movement, this threshold allows other pixels that are neighbors to the pixel just detected to also be trigger as movement despite them having a lower activation value. This hysteresis allows moving objects to grow larger to include more appropriate borders where pixel comparisons are not as significantly different than the "High Threshold Amount".

15. Stop Below Movement Percent specifies how much movement needs to be detected for the module to signal that something has changed in the image. If nothing is detected then the entire image is set to black.

16. Stop Below Pixel Difference specifies that if all pixel difference values are below the specified amount then the image will be considered still even if the percentage of pixels changed exceeds the above percent.

Notes

Ideally to detect movement your camera needs to maintain the lighting conditions from one image frame to the next. You may be able to turn off auto-exposure or auto-shutter speeds by examining your camera options. Select the options button in the main RoboRealm dialog, then one of the Video Capture or Video Format buttons to see if your camera provides a manual override of setting these parameters. You can test the lighting issue by setting the movement detector to use the last 50 previous frames to calculate the RGB difference. Once these settings have been made the current image will eventually go to black assuming no movement is within the camera's view. If you then hold up a white sheet of paper in front of the camera you will suddenly see the entire background change since the white sheet of paper will have caused your camera to shift intensity values. This changing of pixel values moves the pixel intensity outside the "difference amount" and therefore the entire image will appear to have moved. Switching the camera to manual mode will prevent this intensity shift from happening. If you are not able to switch to manual mode select the 'Anti - Auto Adjustment' checkbox which will attempt to undo these lighting changes by your camera.

When using the movement detector you may notice issues with black streaks or spots in the resulting movement mask. These spots are caused by dark background pixels. Ideally to create an appropriate mask (similar to green screen or blue screening) your background should be as planar as possible and not include the color that the foreground will. For example, using a white sheet as a background and then standing in front of the background with a white shirt on will not allow you to segment your white shirt ... the shirt and white background are considered as the same color and therefore no movement will be detected. This also frequently happens for black areas since the absence of light in shadows will cause confusion if your foreground object has any shadows in it.

Variables

MOVEMENT_PIXELS - number of pixels that have changed from frame to frame

MOVEMENT_PERCENT - percent of pixels that have changed from frame to frame

See Also

[Optical Flow](#)

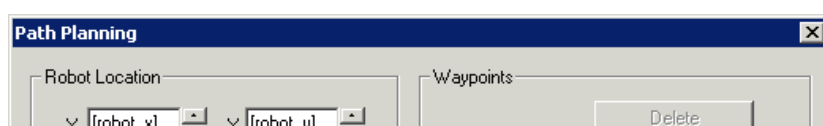
[Average](#)

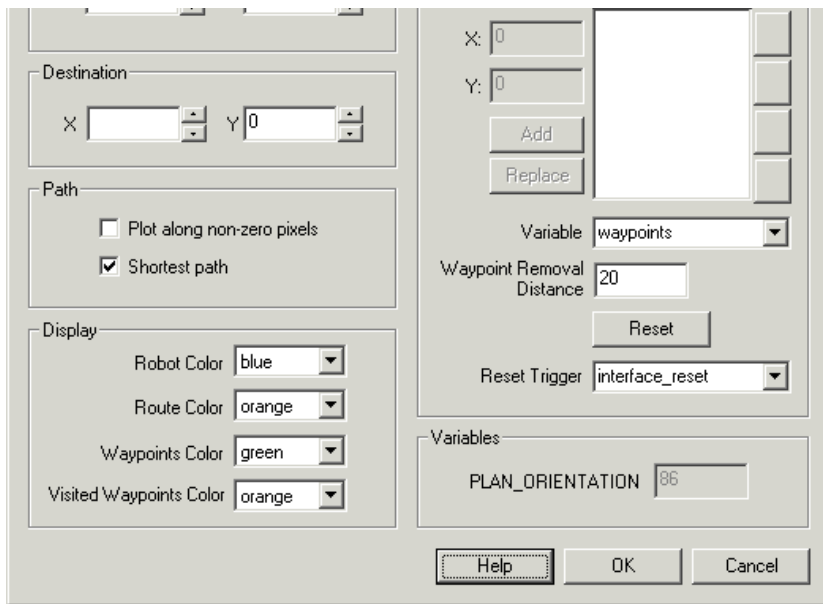
Path Planning

The Path Planning module is used to determine a route from one coordinate location to another along a set of waypoints. For example, if you had an image of a maze and you needed to determine the best path from where the robot is currently located to where it needs to be you would use the Path Planning module to determine the shortest or best path to the desired location.

Note that the robot's location is labeled as a circle with an X in it.

Interface





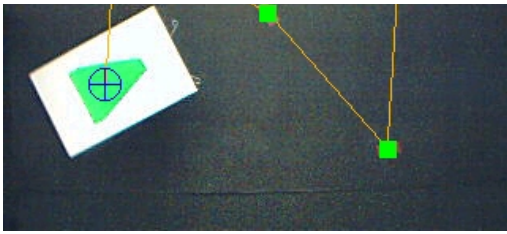
Instructions

1. Robot Location - Specify the start location (X,Y) coordinate of where the robot is currently located. Note that you can type in variables using [variable_name_x] into these dialog entry boxes. This allows you to use the RoboRealm application to determine where the robot is visually located and where the robot needs to go and feed that information into the path planner.
2. Destination - Specify the end location (X,Y) that specify the robots destination.
3. Waypoints - Add in waypoint coordinates that the robot is suggested to go through. Note that if the waypoints are NOT in a traversable path they will be ignored. The waypoints can either be manually entered using the provided interface or configured to read the points from a specified variable.
4. Removal Distance - Specify how close the robot need to be to a waypoint (in pixels) before it is considered traversed and removed from the waypoint target list.
5. Reset - Once all waypoints have been visited you can press the Reset button to reset all waypoints to a non-visited state.
6. Reset Trigger - You can also select a variable when non-empty will also cause the reset to trigger. This is handy if you want programmatic control over clearing all visited waypoints.
7. Path - select "Plot along non-zero pixels" if the current image contains the path boundaries as non-black pixels. This allows you to restrict the robot movement to a selected path or line instead of considering all parts of the image traversable. With this checkbox selected the path planning module operates within the 'white' or on pixels within an image. Any pixel that is non-black is considered part of the potential path. Any pixels that are black are considered non- traversable.
8. Optimal Shortest Path - select this checkbox if you want the system to automatically determine the optimal shortest path to all waypoints. If this is not selected then waypoints are visited in the order they appear in the manual waypoints list or in the waypoint variable. Note that due to the complexity of finding the shortest path only 17 waypoints can be used in this algorithm.
9. Greedy Shortest Path - select to automatically determine a non-optimal but fast shortest path for more than 17 points. This algorithm selects the next nearest waypoint as the direction of travel. While not optimal it does work well for most cases.
10. Robot Color - the color marker used to identify the robot (circle with a X)
11. Route Color - the planned route between waypoints
12. Waypoints color - the color of the square that identifies the targeted waypoints
13. Visited color - the color of the waypoints that have already been visited

Example

Source





See the [Path Planning tutorial](#) for more information.

Variables

`PLAN_ORIENTATION` - the suggested direction the robot should move to stay on the current path. If the robot is NOT currently on the path the `path_direction` will be the direction to the nearest point that is on the path in an attempt to get the robot back on track.

`PLAN_POSSIBLE` - 1 when a path between the robot and destination exists and 0 otherwise.

`PLAN_COORDINATES` - An array that includes all the path points as displayed by this module.

`WAYPOINT_DISTANCE` - the distance between the robot and the next waypoint. Note that this distance is calculated based on a straight line and does NOT respect any path confinements.

`WAYPOINT_NUMBER` - the current waypoint number being traversed to.

`WAYPOINT_X` - the X coordinate of the current waypoint number being traversed to.

`WAYPOINT_Y` - the Y coordinate of the current waypoint number being traversed to.

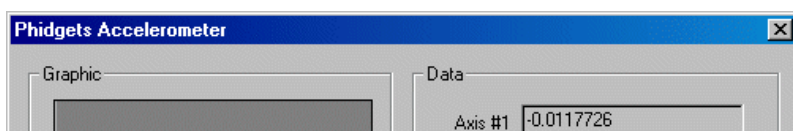
Phidgets Accelerometer

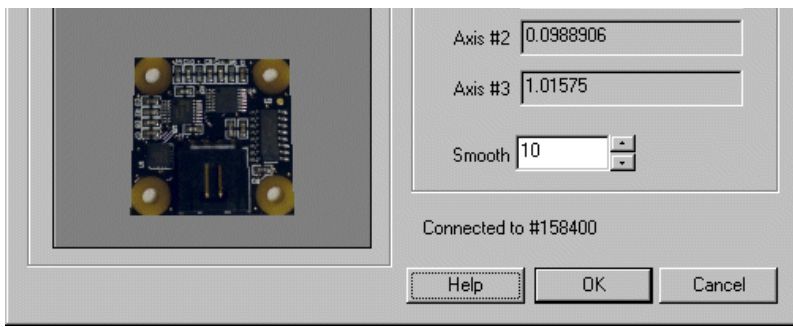


The Phidgets Accelerometer module provides an interface to the Phidgets Accelerometer. Using this module you can read all 3 axis of the Accelerometer by accessing the appropriate variables created by this module.

Note that the simulation graphic uses Axis1 and Axis2 for tilt sensing and checks Axis 3 to determine if the device is upside down. Keep in mind that the Accelerometer only measures tilt and NOT orientation (Z axis) and thus the simulation graphic will assume you are holding the device with the USB cable connection oriented towards you.

Interface





Instructions

1. Plug in the Accelerometer and tilt the device forwards and backwards. You should see the Axis values change and the graphic simulation move in response to the device. The simulation attempts to indicate what orientation the device is as if it is held flat to the ground.
2. Smooth - if the device is not as responsive as you would like try reducing the smooth value. This will allow the numbers to be much more reactive but may also cause some unwanted jitter due to noise in the determined values. By decreasing the smooth value you make the device much more responsive to movement, increasing the value will delay the sharpness of the movement but also make it much more smooth. Note this is reflected in the simulation graphic.

Variables

PHIDGET_AXIS_1 - the current value of Axis 1

PHIDGET_AXIS_2 - the current value of Axis 2

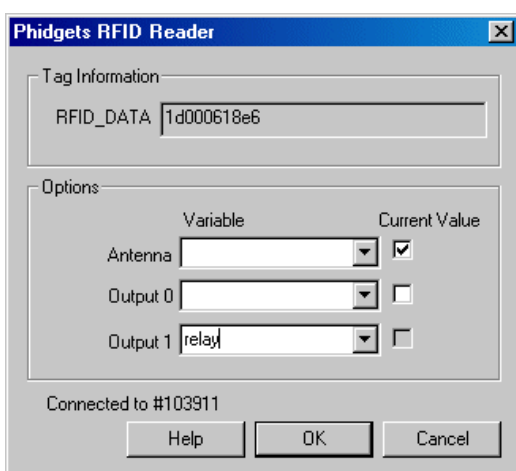
PHIDGET_AXIS_3 - the current value of Axis 3

Phidgets RFID



The Phidgets RFID module provides an interface to the Phidgets RFID tag reader. Using this device any RFID tag placed about 3 inches from the reader will respond with a unique identification number. This number can then be used to localize the robot, detect the presence of specific objects or be used to indicate tasks to perform.

Interface



Instructions

1. Antenna - Ensure that the antenna checkbox is on or that a variable with a non-zero value is selected in the variable dropdown. If the antenna is off no ids will be recognized but power consumption will be at a minimum. Turning on the antenna will allow the device to pick up nearby RFID tags.
2. Connect to #XXXXXX - Ensure that the module is actually connected to the Phidgets device. The devices serial number will be displayed after the #. If N/A is displayed check your connections and/or Phidget drivers.

3. Output - The RFID device has 2 digital outputs on the board that can be controlled via the PC. To switch on/off the digital out select the appropriate checkbox. To control the output automatically select an appropriate variable that contains or will contain the position value that will be sent to the board. This is used to automatically change the digital output values based variables created by [VBScript](#) (using the SetVariable function), [Plugin](#), [API](#), [Set_Variable](#) module, etc.

Variables

PHIDGETS_RFID_TAG - contains the tag number read from the last card waved near the reader.

PHIDGETS_RFID_SERIAL - contains the serial number of the reader that the PHIDGETS_RFID_TAG value is associated with.

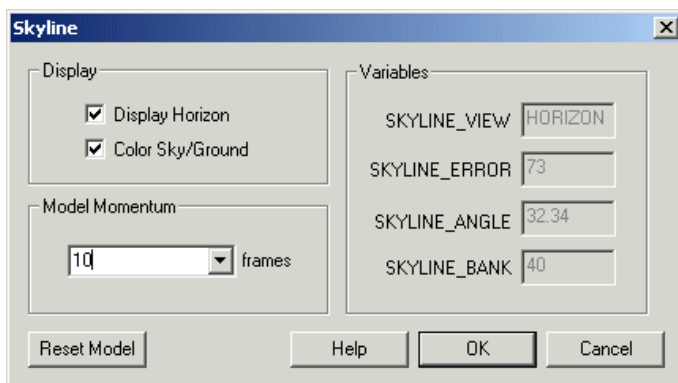
Skyline



The Skyline module analyses the current image and proposes where the skyline or horizon might be. It does this task by minimizing the variance between two halves of the image to the point that a settled state is achieved. This essentially breaks the image into two parts. As most skies are of very different texture than the ground this is an effective technique of determining the split.

For those RC plane enthusiasts it should be noted that an image in one second may contain a horizon but in the next all ground or all sky. This renders the image division useless unless the Model is provided with several frames from previous "good" horizons. Thus, you should start off with the horizon in view which will allow the model to build up a sampling of what is sky and what is ground. When only sky or only ground is suddenly in view the module will indicate this using situation in the SKYLINE_VIEW variable.

Interface



Instructions

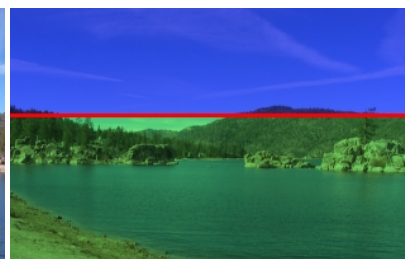
1. Display - you can chose to turn off the horizon and ground/sky indicators by unselecting the checkboxes.
2. Model - specify the number of frames that are used to create the ground/sky model. Note that too many frames will cause the model not to update soon enough if the image content changes drastically, too little and the model might update when only looking at ground or sky. At 30 frames a second 30 would sample about 30 frames but would then also require you to move out of all sky or all ground images in a second before the model updates. Try 120 which would build a model over 4 seconds.

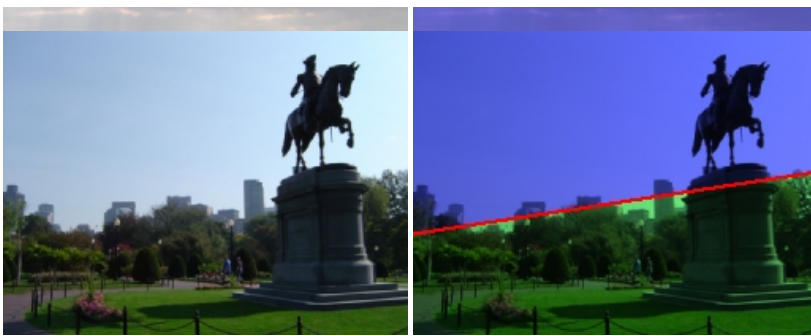
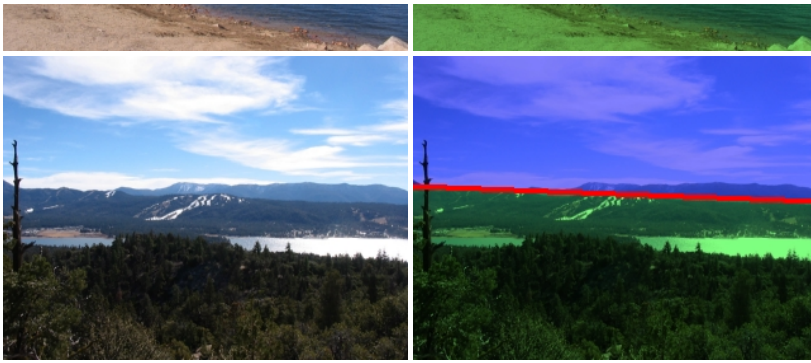
Example

Source



Skyline





Variables

SKYLINE_VIEW - either HORIZON, SKY or GROUND depending on what the module determines

SKYLINE_ERROR - the goodness of split between the two image halves. Lower is better.

SKYLINE_ANGLE - the horizon's angle.

SKYLINE_BANK - the y coordinate of the horizon line with respect to the center height.

SKYLINE_X1

SKYLINE_Y1

SKYLINE_X2

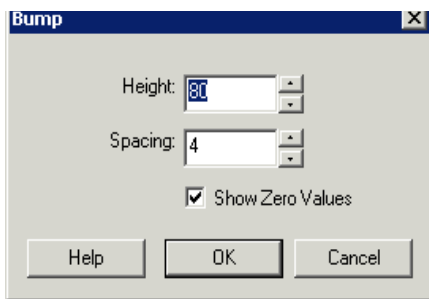
SKYLINE_Y2 - the starting and ending line coordinates as seen on the screen.

Surface Plot

To better visualize the pixel values in the image use the surface plot function. The surface plot uses the pixels intensity as a Y axis offset and plots the location using a white pixel. The end effect is a 3-D like surface plot that can be used to see how processing is affecting pixel rows in more detail.

Interface





Instructions

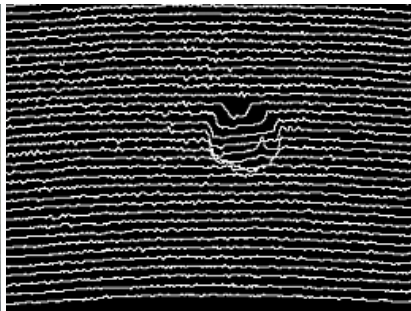
1. To increase/decrease the maximum Y offset change the "Height" value.
2. To increase/decrease the row sampling change the "Spacing" value.

Example

Source



Surface Plot

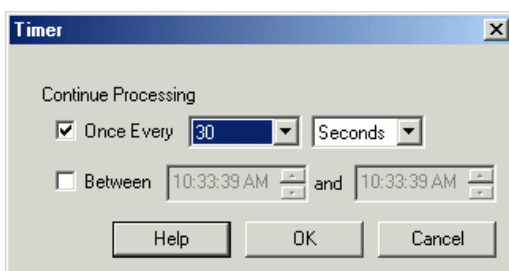


To see how filters can affect pixel values crop the entire image to a single row and use the surface plot to plot the single row. Prefixing filters before the crop and surface plot can help indicate how filters affect pixel values.

Timer

The Timer module allows you to continue processing images based on certain time constraints. Namely, you can proceed every 10 seconds, or 5 minutes or only between 9:00am to 5:00pm, etc. These time constraints allow you to specify a slower than realtime capability for those actions that don't need it. For example, you can specify that an image be saved to disk (in various formats) once every 30 seconds.

Interface



Instructions

1. Select the appropriate timing condition using the checkboxes.
 - Once Every - specifies that processing should continue based on a periodic time frequency
 - Between - specifies that processing should continue only during the specified time range. Remember 12:00pm is noon and 12:00am is midnight.
2. Select the appropriate dropdown boxes to yield the desired timing frequency. Note that once every 60 seconds is the same as once every 1 minutes.
3. Select the time range that processing should continue.

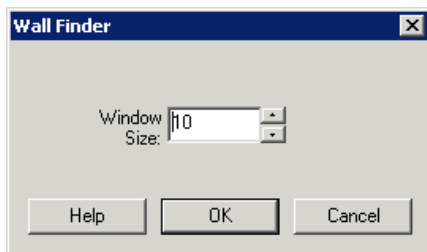
If neither checkbox is selected processing continues as normal. When processing of images is stopped due to one of the time constraints the image displayed will reflect processing up to that point. You will still see the image change and update based on the camera update rate but no processing will be performed after the Timer module insertion point

will be performed after the final module insertion point.

Wall Finder

The Wall Finder function attempts to determine where the corner of the floor and the wall meet. It assumes that the camera is looking at the floor such that the floor is seen at the bottom of the image. The function looks for the transition between the floor texture and the wall texture and indicates that corner using a red dot.

Interface

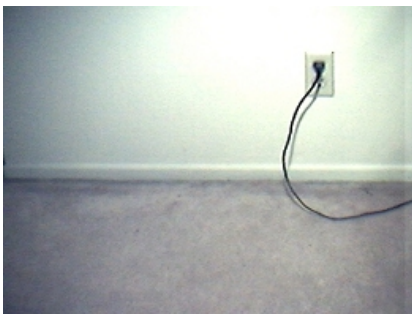


Instructions

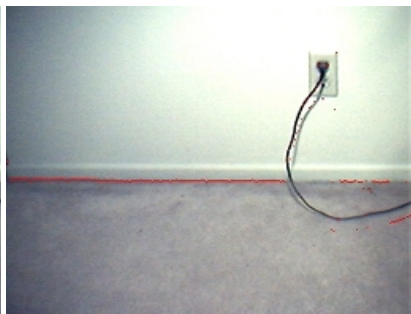
1. You can change the window size that is used to determine the floor/wall texture in case you need to increase/decrease the sensitivity.

Example

Source



Wall Finder

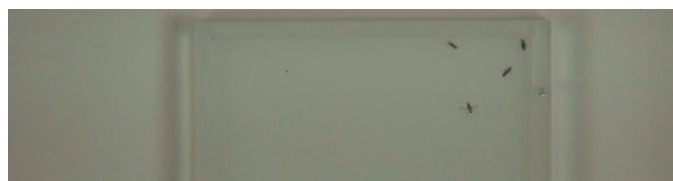


Blob Tracking



This tutorial shows one possible solution for tracking insect paths. The tutorial surrounds the Blob_Tracking module which provides various parameters for connecting blobs on successive images in order to determine movement. The goal of the tutorial is to identify the individual paths of the insects in order to further analyze their interactions and quantify their movement. The path coordinates are provided for external applications to process the results in the desired way.

Movement tracking requires successive images which are best represented in movie files. For this tutorial we use the following raw video file provided by [Juan Pablo Busso](#) from the University of Zürich.





(click to download/view)

As we are working with multiple frames in a video we use the [Media Reader](#) module to play the video frame by frame. Note that there are many other modules that can be used to play videos or stream network images. Multiple modules ensure that many formats can be used within RoboRealm to analyze frames. ([Media Reader](#), [VLC_Player](#), [Read_AVI](#), [HTTP_READ](#), [RTSP_Player](#))

From this video we pick an individual frame to test processing. Prior to tracking blobs we need to prepare the image for best results. While this is not a huge issue, it can cause the moving blobs to be broken into smaller blobs that could not be tracked. By zooming into one part of a single image we see our first problem.



The video signal sampled is from an NTSC source that has interlaced the image. Left and right rows are successively updated with the current live image which causes a broken object in areas of high movement. Interlacing images was common when video transmission capabilities were more limited and while still present today are slowly being replaced by high resolution single image frames.

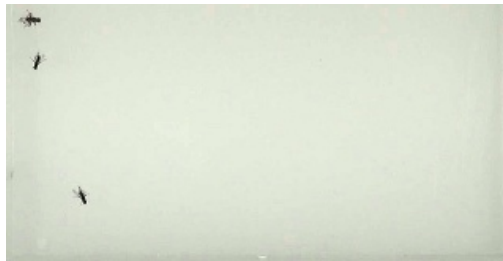
The solution to this is to use the [Deinterlace](#) module which uses various techniques to eliminate the jaggies caused by movement. Using this module our zoomed single image now appears as:



As the insects are contained within a particular area we can [Crop](#) the image to just focus on that area the insects will be contained within. Cropping the image to focus on just the area you want to investigate is always recommended since cropping is quick and it reduces the amount of data to be processed in successive modules.

In addition to cropping, we also use the [Normalize](#) module to correct the lighting to increase the contrast between the insects and the white background. Given these two additions the image appears as:

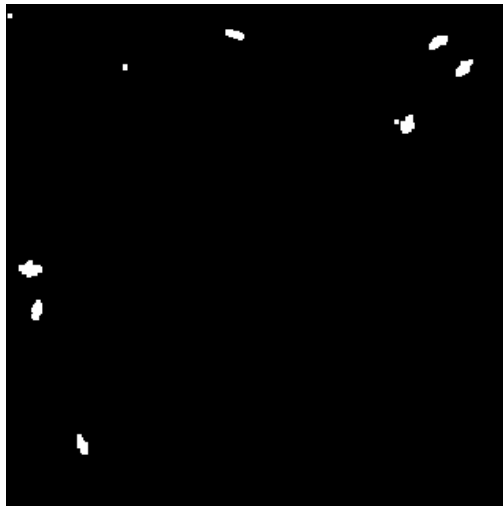




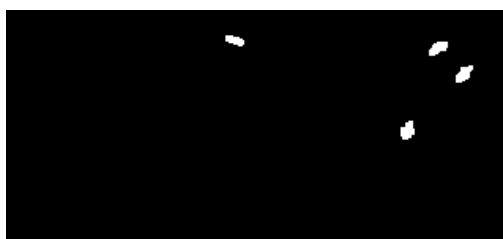
We are now ready to extract just the insects from the white background. As we have such a high contrast between the insect and the back ground we can use the [Auto Threshold](#) module to separate foreground and background. Because RoboRealm processes blobs as white objects we select the Black Mask to invert intensities, i.e. black objects become white. While its easy to switch from one thresholding method to another, care should be taken to watch the entire video using any particular thresholding technique. A particular method may work for the current frame but fail in previous or successive frames.



As we are working with relatively small objects we use the [Dilate](#) module to help consolidate those insect objects that may be broken due to the thresholding. This ensures that bug parts are combined into a single representative blob.

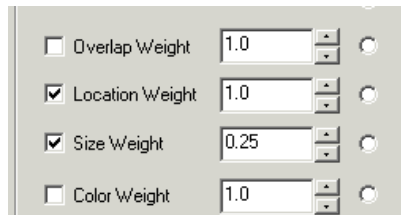


We still have some parts that are not actual insects. In the upper right corner there are two black dots that are not insects and stationary. We can remove those using the [Blob Size](#) module since they are smaller than the actual insects. While these would remain still and would be ignored during tacking, we chose to remove them prior to tracking.



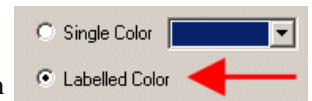


As we now have a black and white image, we can finally use the [Blob Tracking](#) module to connect the blobs in each frame with the frames before. There are many options to choose from in terms of how to associate a blob with its previous image blob. In our case, the insects move quite a bit from image to image so checking for overlapping blobs will not work for the fast moving insects. Instead we use the Location or Proximity and Size attribute to connect blobs. The location assumes that blobs that are closest to previous blobs are the same blob. The Blob tracking module is able to determine the correct association by analyzing all blobs in the image for their best match. Thus if a fast moving insect moves close to a stable insect, the global solution will determine that the stationary insect will not match to the fast moving one. We use the size attribute at a diminished weight (25%) to ensure that in situations where two fast moving insects cross each others path the actual size of each insect will help resolve the ambiguity.

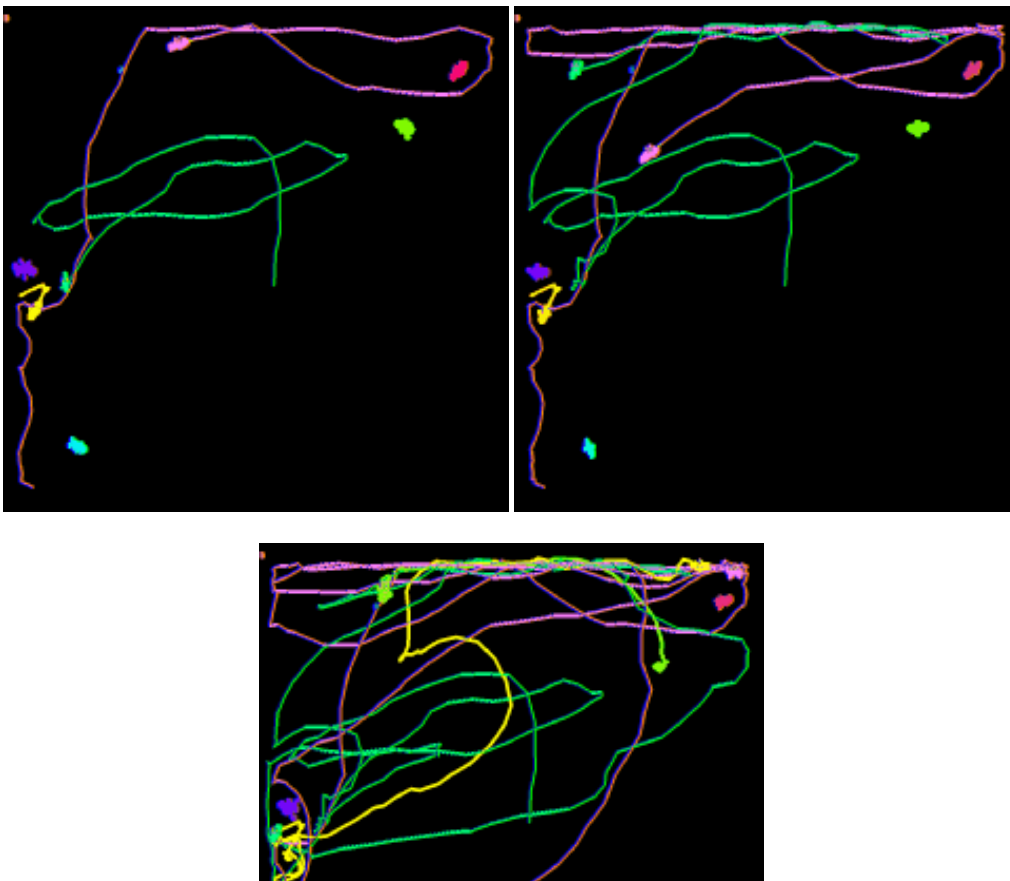


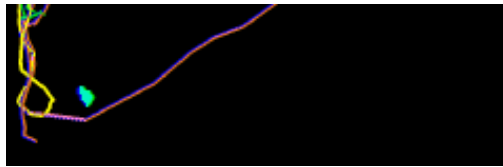
Your specific project may require additional attributes to be considered in order to resolve ambiguities. A simpler solution for this tutorial would be to increase the frame rate of the camera to capture smaller movements. The location/proximity measure will then become more effective due to the smaller movement and the overlap attribute might become valid.

While experimenting, be sure to have the module set to label colors. This will color each blob with a distinct color that if suddenly changes indicates that the tracking was lost for some reason. Slowing down the Media Reader (or VLC or AVI_Reader or ...) to play frames slower can help provide some insight as to why tracking may have been lost.



A couple frames from the playback show the growing paths of the insects:





The module monitors the VIDEO_FRAME variable (selected in the interface) to check when the video loops. Once it does, the tracking information is cleared and restarted. This ensures looping video is processed correctly by restarting the tracking on each loop.

Once the information is generated exporting the path point data for each object we can use the [Write Variables](#) module. By inserting the BLOB_TRACKING variable into that module a CSV file will be generated that contains the movement in each image frame. The format of that variable is BLOB_ID, BLOB_X, and BLOB_Y which means that after the full run we will have all path points of each blob and also in what frame they moved.

```
BLOB_TRACKING
```

```
10;30;380;2;74;410;11;56;535
```

```
10;39;385;2;77;409;11;58;534
```

```
10;39;384;2;76;406;11;57;534
```

```
10;40;382;2;75;406;11;56;533
```

```
10;54;377;2;79;405;11;55;533
```

```
10;74;373;2;85;411;11;60;540
```

Once tracking is complete, we may want additional statistics about the tracked blobs beyond what the Blob_Tracking module provides. In this case, we add the [Geometric_Statistics](#) module that generates an AREA array of the number of pixels of each blobs. Note, we add this module just AFTER the Blob_Tracking module (with Annotation enabled) to ensure that we see the same image as the Blob Tracking module. This is necessary since the two modules need to see the same blobs in the same order otherwise we cannot correlate the Blob_Tracking ids with the Geometric_Statistics array. As the Blob_Tracking module can remove blobs (Remove Un-Tracked option) its important to process the same image in order to get the blob order correct.



To perform the annotation we use a [VBScript](#) module to create an array of X, Y coordinates for each blob followed by a Text string of "Area: Z" which can be used in the [Display_Text](#) module to display an area value above each blob.

The key to this array is that

```
BLOB_TRACKING_IDS - array created from blob tracking with zero entries for non-tracked blobs
```

```
AREA - array created from Geo Stats module of each blob
```

are two arrays generated by different modules but are in the same order. Most modules in RoboRealm analyse the image from the left to right, bottom to top order. As long as the same image is used in both modules, the ordering of blobs WILL be the same. Given these two arrays, we know have enough information to correlate a blob id with its area.


```
ids = GetArrayVariable("BLOB_TRACKING_IDS") ' from blob tracking'  
  
area = GetArrayVariable("AREA") ' from Geo Stats module'  
  
ReDim areaList (ubound(ids)*3)  
  
if isArray(ids) then  
    if ubound(ids) > 0 then  
  
        j = 0
```

```
for i = 0 to ubound(ids)
  if ids(i) <> 0 then
    areaList(j) = GetVariable("BLOB_TRACKING_" & ids(i) & "_X")
    areaList(j+1) = 20 + GetVariable("BLOB_TRACKING_" & ids(i) & "_Y")
    areaList(j+2) = "Area: "&area(i)
    j=j+3
  end if
next

end if
end if

SetArrayVariable "AREA_LIST", areaList
```

Note that with any solution it is very important to test enough images to validate correctness. Testing on only a couple seconds of video is not sufficient for a reliable solution.

Download the  [Blob Tracking robofile](#) to see the steps and more about the modules used. You can download the original video used in this tutorial [here](#).

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Blister Inspection



This tutorial shows one possible method of checking for the presence of pills within a blister pack. Often during the manufacturing process it is possible for a pill not to be inserted correctly into a cavity which needs to be detected before packaging. A vision system is an ideal solution for detecting issues in blister packs.

Lighting

The most important aspect of any vision project is lighting. Bad lighting leads to bad results. In our case of blister inspection since we are working with highly reflective metallic packs the best lighting is a diffuse lighting with a high contrast background. Diffuse lighting (versus direct) will produce a more even illumination and reduce specular reflections that can cause the absence of pill color from certain angles.



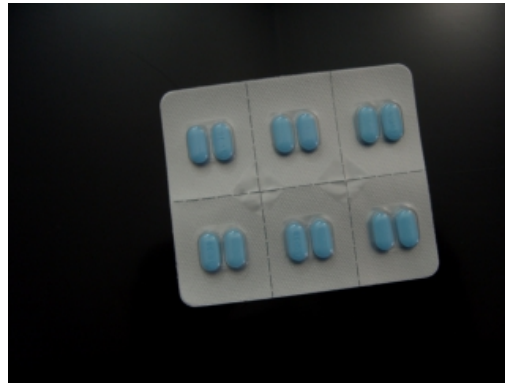


Note the lack of the vertical highlight on the tablets. These highlights cause the color of the tablet to be overpowered by the white light.

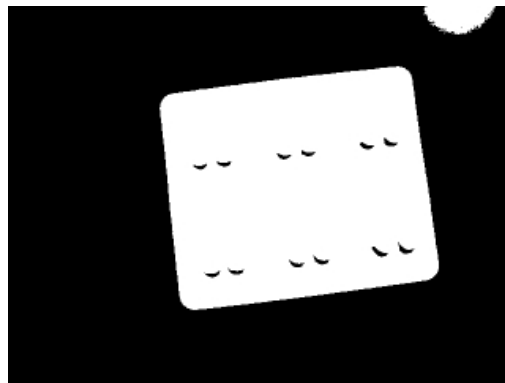
Position

Now that lighting has been resolved, we can experiment with how the object will appear in the camera image with respect to position. The more rigid the object placement the easier the analysis becomes. For example, if we can always guarantee that the object is in exactly the same position in the image without any rotation in any axis nor a scale difference then the analysis gets a lot easier. The reason it gets a lot easier is that we can place probes exactly at the know location to check for any irregularities.

In reality this perfect placement is unlikely. In our current case, we will assume none of these are true in that the blister pack will move around the image and even have some scale change. To get started we chose a sample image:



First we threshold the image to segment the blister pack from the background. This is why we ideally have a high contrast background which makes this process much easier. We do this using the [Auto_Threshold](#) module.



As we have some holes in the object due to some shadows we use the [Fill](#) module to remove them.

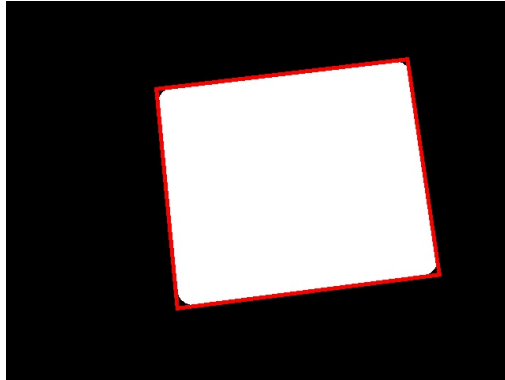


Since our lighting wasn't perfect we have a extra object in the upper right side of the image which received more direct light than the rest of the image. We can remove that object by selecting only the largest object in the image using the [Blob_Size](#) module.





We then take the remaining blob (our blister pack) and replace it with a rectangle using the [Replace Blob](#) module. It uses the single large blob to determine how best to fit a rectangle into the blob. The reason we need to do this is to ensure that we know the bounding box of our blister pack despite it having round edges. Fitting a rectangle into this blob ensures that we have those 4 corner coordinates needed for the next steps.



Now that we know the 4 coordinates, we can then use those in the [Affine](#) module to transform the image into just the blister pack. This removes translation, rotation and scale differences from the image in one module. The result of which is a known image size (300x240) with just the blister pack.



The image is now ready to check inspected using the [Blob_Inspection](#) module. But before we can use that module we must create a mask that tells the [Blob_Inspection](#) module what parts of the image we want to analyze. Since we are not interested in the actual surrounding blister pack we generate a mask that contains mostly the pills. This can be done manually (the blister pack will always be transformed to the same dimension so only one mask is needed) or using RoboRealm to threshold, fill, and smooth an existing image. The result of which is

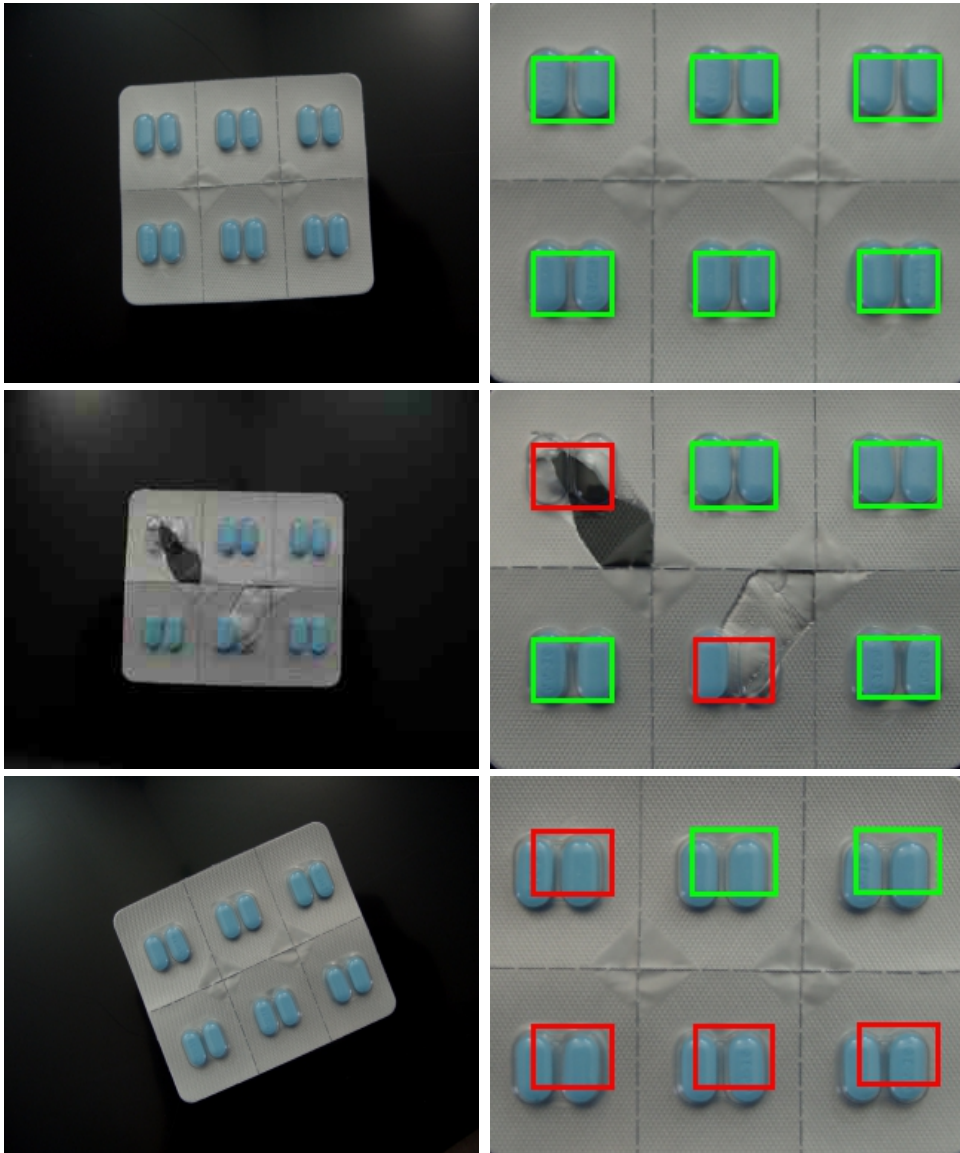


Note that the mask does NOT need to be perfect and can include some parts of the coating. The mask helps to focus the attention on what you want to analyze and some background can always be included.

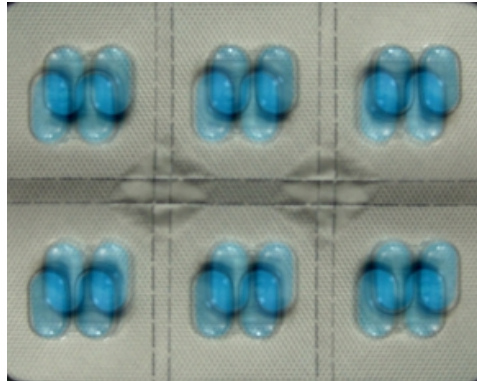
We now add in the [Blob Inspection](#) module. We add a blob to the training database by clicking on Add Blob (ensure you can also save the training file) and selecting a Color Profile of 85. Since, in our case, the tablets are somewhat blue we can just use color to validate that the tablets are present. In other cases this may not be the case which is why the [Blob_Inspection](#) module has several attributes that it can check.



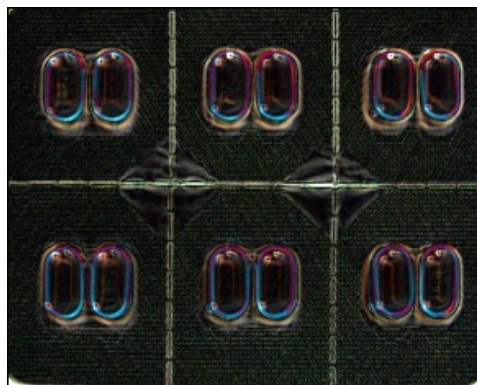
The Color Profile appears to work, we can now try on some other images



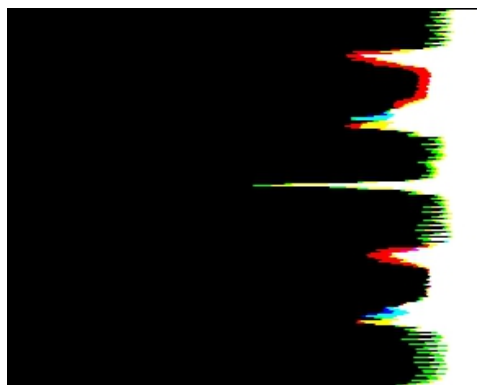
Opps, what happen with Test3? Turns out an assumption that we made, i.e. the blister pack is symmetrical, is not true. Taking one of the images, rotating it 180 and merging those two together we notice that one half of the pack is not the same height as the other.



The solution is to know which way round the pack got detected. As we don't know if there will be any pills in the blister pack or how many we cannot use the pills to decide which orientation we are in. Instead, we look to the perforation to tell us which way around the image is. The first step is to detect these perforations using the [Sobel Edge](#) module.

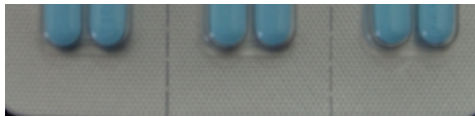


Then we use the [Collapse](#) module to add up all the edges horizontally. We know that the blister pack is aligned correctly so we only need to detect the strongest horizontal line in order to determine where the horizontal perforation is.

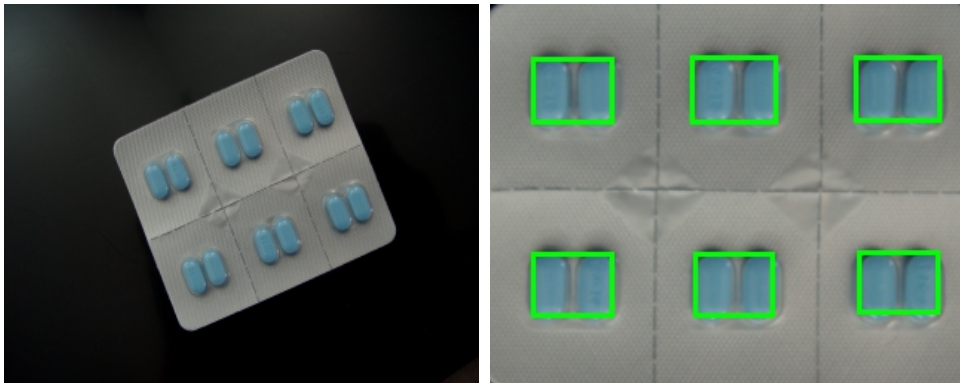


It becomes quickly evident where the middle perforation is. In order to locate the left most point we can use the [Point Location](#) module to determine the Y coordinate of the horizontal perforation.





Using this Y coordinate we can know if we need to rotate the image prior to analysis since the correct orientation that we used to create the Blob_Inspection mask has a perforation above the center Y, whilst the wrong orientation is above. Adding this logic into the pipeline the Test3 becomes



For any solution it is very important to test enough images to validate correctness. Testing on only a couple images is not sufficient.

Download the [Blister Inspection robofile](#) to see the steps and more about the modules used. You can download the original images used in this tutorial [here](#) and the Mask image used [here](#).

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Comparing Objects



Comparing a sample object to a known object is an effective way of determining if a particular object conforms to a known standard. This tutorial shows how to compare an object to a known standard with subtle changes that indicate a problem. The object is assumed to be planar and moved into view on a conveyor belt system. In our case we are using a printed graphic of an Owl that may be seen on packaging or used as product logo on labels. There are small differences in the graphic that are used to simulate printing faults.

First we start with the ideal template image that we want to compare with successive test images.

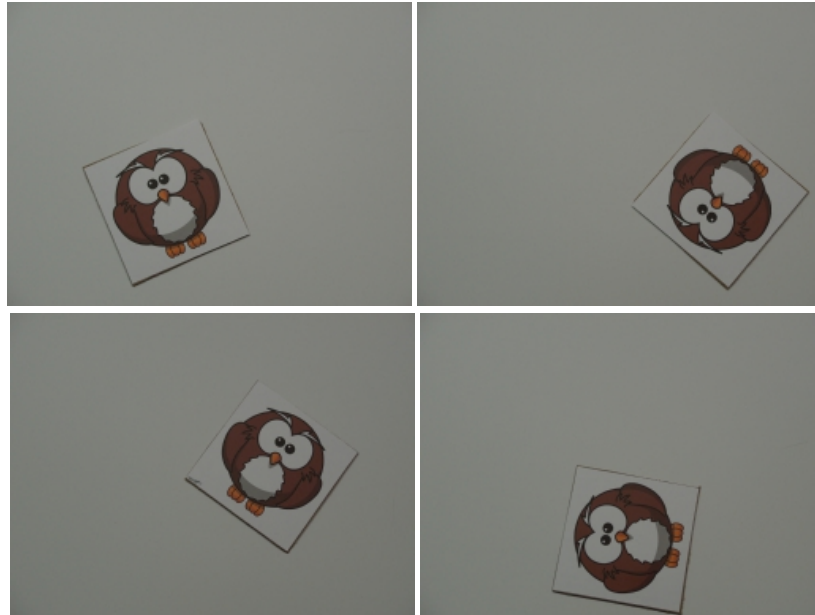


Once we have a good clean copy of the template, we need some example images to test against that are as close as possible to the final production environment where the quality check would be performed. These images are taken with:

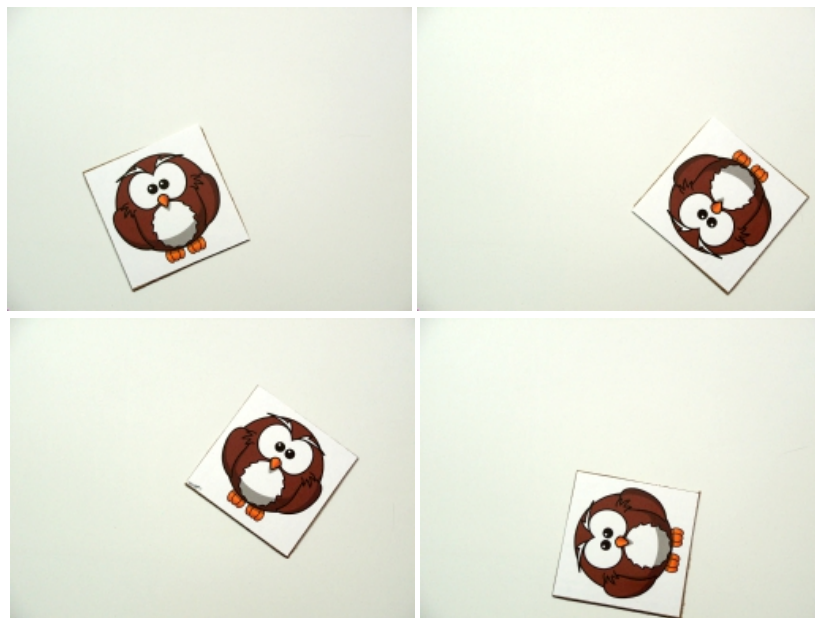
- **With enough light** - Too little light will cause the camera exposure to be long/high. This can cause motion blur which will soften object edges and can even remove smaller parts of the object entirely. Too much light will remove color differences or flatten intensity ranges that are important when comparing objects for intensity changes over the image. Ideal lighting will result in the image being in the middle of the intensity range of 128 (0-255 for most images) which is the case in the following images. Having the mean at 128 ensures that enough high

and low pixel values are captured without causing the CCD sensor to clip intensity values.

- **A stable camera mount** - This is relevant since you want to be sure that image distortion is the SAME for each image that you take. Any camera vibrations should also be eliminated since you want to ensure that the object edges are sharp.
- **A uniform lighting solution** - Shadows, highlights, etc. are minimized to eliminate any image artifacts during comparison. While the images themselves may not appear very bright the lighting is even throughout the image.



We can now use the [Normalize](#) module to ensure that the image pixels occupy the full intensity range and fix our overall intensity range. Some cameras (such as the consumer camera used for this tutorial) will auto-correct lighting variance (high to low range) to a less than ideal range. We can correct for this to some extent in software without sacrificing image quality.

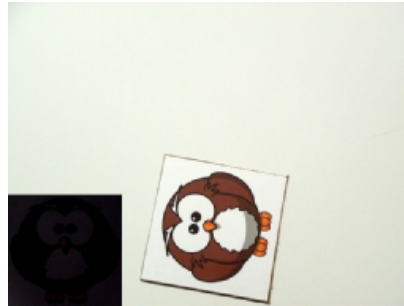


We used 3 small variations in our graphic (eyes, nose and feet). By flipping through all the variations we can see the changes done in the graphic. Those are the variations we are looking for in comparing images.



Comparing objects can be done in many ways. The key is to compare two images with each other in a pixel to pixel comparison. **Before** we can do this we need to ensure that the current image and template image are as close as possible to each other in terms of placement. Our template image was just the pattern to be tested but our example images include a lot of white space and are not necessarily aligned (i.e. rotated) nor the right scale as the template. Thus subtracting the two images from each other to reveal differences without alignment would result in something like

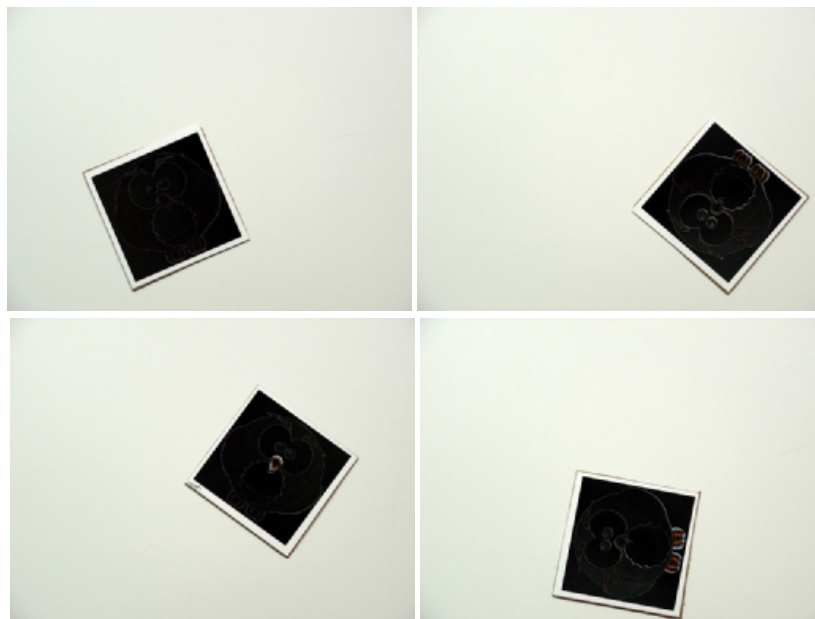
right scale as the template. Thus subtracting the two images from each other to reveal differences without alignment would result in something like:



This bad match shows the template being subtracted from the current image without regard to translation or rotation of the template prior to subtraction. Clearly this isn't what we need.

Instead, the next step is to use the [Align Image](#) module to align the template into the current image. The module allows for quite a few options in order to do this which you can experiment with in order to get the right results.

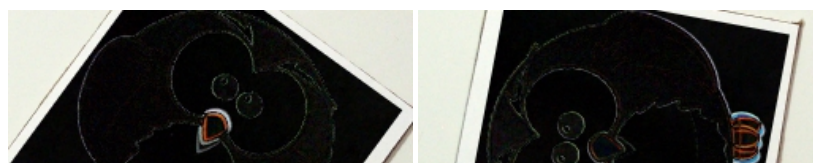
Using this module to rotate, scale and subtract the template from the current image we get a much better result.

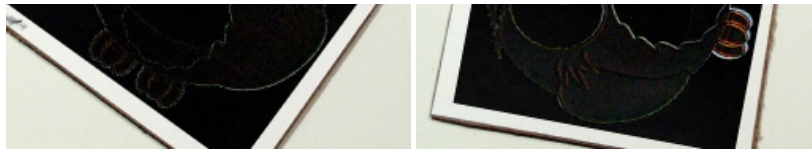


These images show the result of subtracting the template image from the current image **after** aligning the template with the current image. This will resolve translation, scale and in plane rotation differences between the template and current image. Note, if the current image is taken in such a way as to be very different from the template the "match score" result produced by the Align Image module will be very low. In the images above, the alignment score is about 90% which indicates a good template match was made.

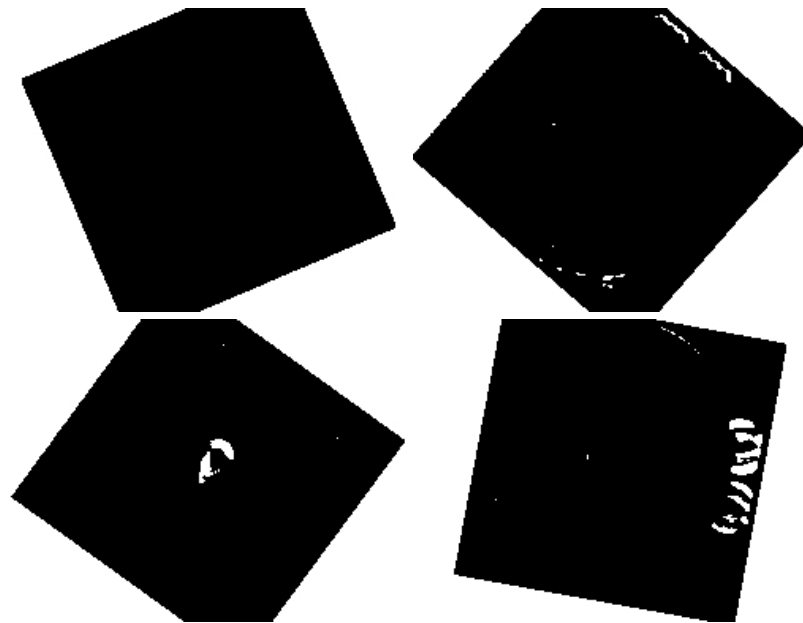
There will always be **some** noise as taking images from two different camera shots will always differ slightly due to acquisition noise.

You can already see some residue from the subtractions. Ex #3 and #4 are already indicate larger differences between the template and image. If we get a closeup image of those two you will more clearly see the issues.

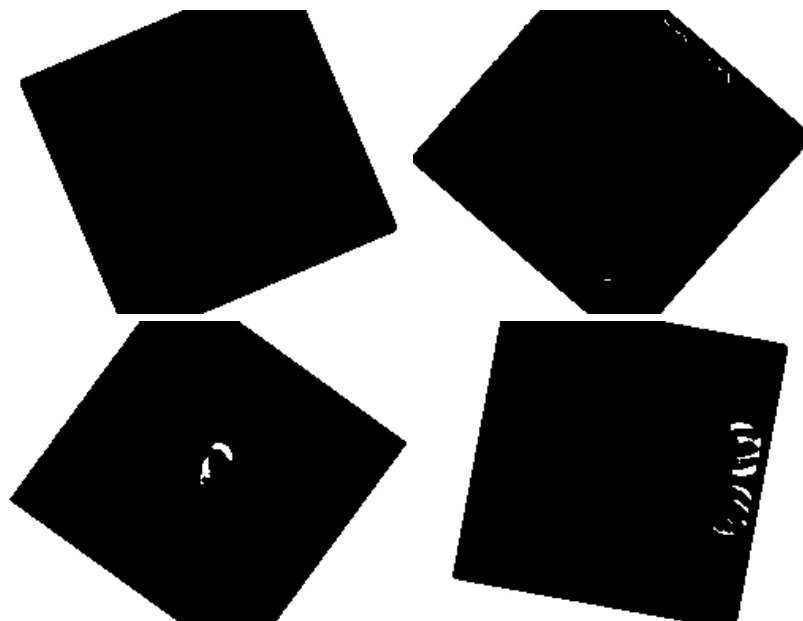




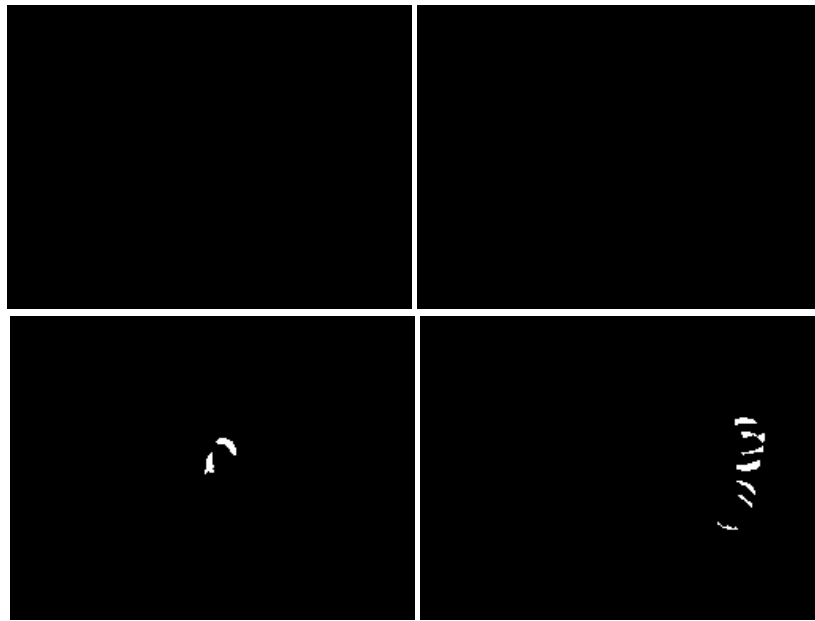
In order to quantify the larger differences (again, some noise will always be present) we run the results through a [Grayscale](#) module to eliminate color (note the orange nose) since we are interested only in the magnitude of the image differences. Then a [Mean](#) filter module will help to merge areas of large differences together followed by a [Threshold](#) module to highlight the difference areas. Using these 3 modules we end up with (zoomed in):



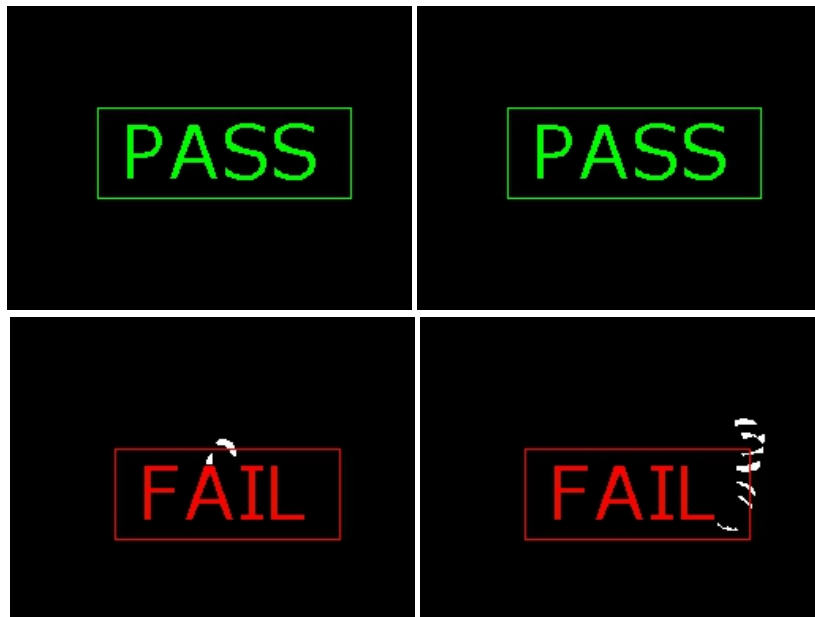
While we can now more clearly understand where the differences are, there is still some fine line differences in Ex #4. These are also present in Ex2. To reduce these edge errors we use the [Erode](#) module to eliminate smaller errors.



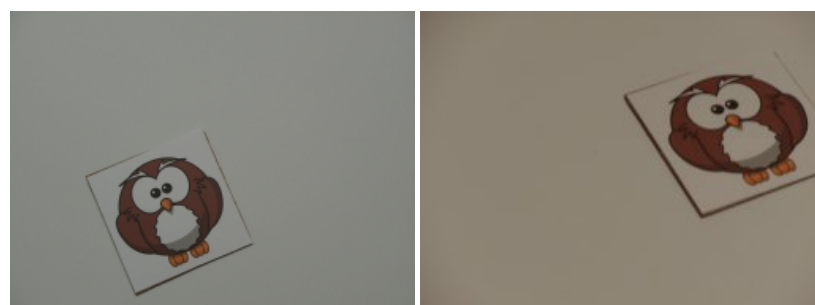
To remove the background (i.e. white area) we can remove very large objects (remember that RoboRealm sees white blobs as objects). Using the [Blob Size](#) module we end up with just the errors in the images.



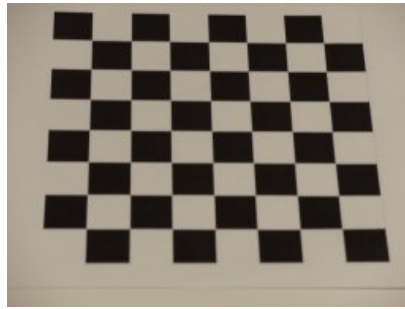
The final images show the differences in Ex3 and Ex4 and can be quantified quite easily by any module that counts the number of on pixels, like the [Geometric Statistics](#). Once measured you can additionally decide how many pixels (i.e. what area) would constitute an error.



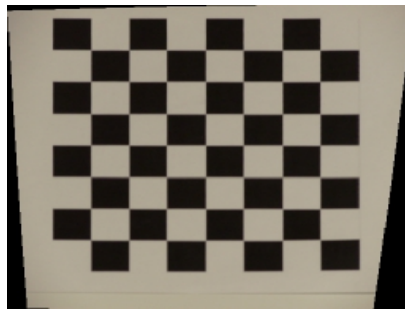
What if less than ideal conditions exist. For example, suppose the camera **cannot** be positioned in a perfectly perpendicular orientation to the object plane as in the above images? For example suppose the following Ex #5 image is more likely to be the image in use. Note the addition of perspective and color distortion in comparison to the above Ex #1 image.



If we run Ex #5 through the current process we end up with a **zero** confidence as the Align Image module cannot work with perspective transform (i.e. things further away appear smaller). Instead we need a calibration grid to help transform the image into a top down view. We start by placing a printed [grid](#) in the same camera view as Ex #5:

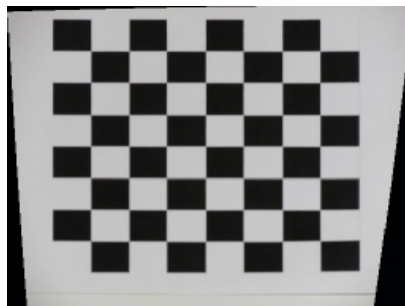


that covers the majority of the work area. We then use the [Auto Image Calibration](#) module to transform that image into a nice calibrated sample. One could also use the [Perspective](#) module to do a similar transform but the Auto module will do the job for us if a nice grid is in view. Using this module we now get:

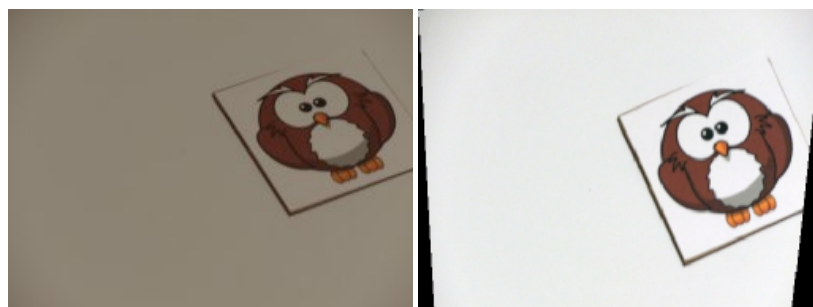


Note that the squares in the bottom of the image are now the same size as those in the top. The black wedges on the side of the image are unknown areas that are created by squeezing the image to fix the aspect ratio of the content of the image.

To fix the color differences, we can use the [Color Balance](#) module in an automatic mode to remove the yellow haze from the image.



Thus, our object #5 after the [Normalize](#) module becomes:



The Align Image module is now able to align the image to 87% accuracy. The resulting threshold image shows no indication of differences ... which is correct. Another example of this correction yielding an 89% match confidence after the calibration process.



You can clearly see how much similar the corrected images are to what we used when the camera was perpendicular to the object plane.

While a lot of error correction can be done in software you should try to create the best image possible and then apply corrective modules to get the desired result. The better the input image, the better the results will be.

Now its your turn! Download the [Comparing Objects robofile](#) to see all the parts described above. You can also [download our database of example images](#) to see more examples of this process in action. And, finally, don't forget the [OWL template](#) file.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

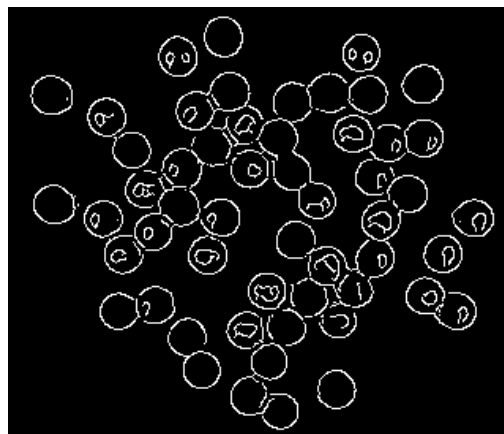
Counting Objects



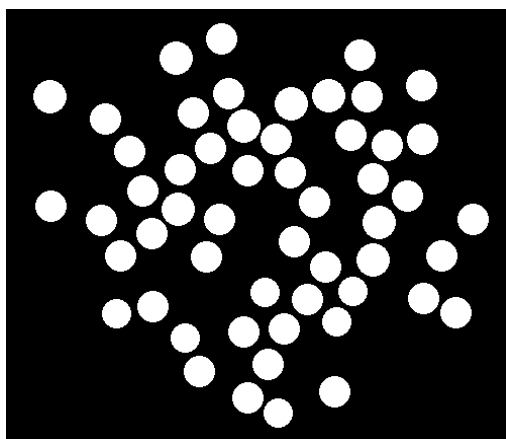
Many objects can be segmented from the background just using color. Working with multiple colored objects can be tricky to keep track of them and to create a summary of the detected objects. This tutorial shows two possible solutions for segmented M&M candy from the background and provides a summary count of what is found in the image.

Technique #1

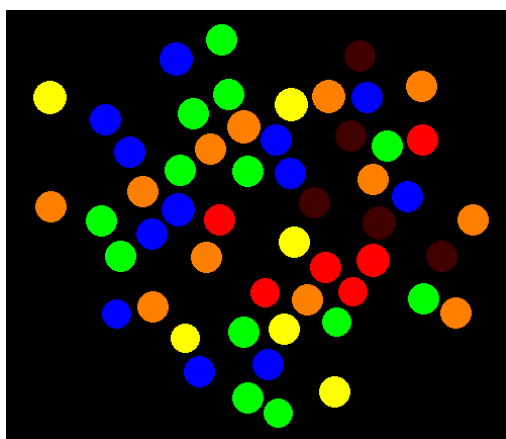
The first technique is to use the circular nature of the M&Ms to first segment them from the background. We do this by first running the [Canny](#) edge detector.



We then use the [Circle](#) module to identify circle. We annotate the location of each circle using a white disk 5 pixels smaller than the originally found circle. This ensure that circles are NOT overlapping. This ensures successive analysis will consider each M&M individually.



Now that we have a mask for each M&M we replace each disk by the color nearest to the mean color of all the pixels that exist in the same location in the original image. The [Blob_Colorize](#) module allows us to do this. Instead of simply replacing each blob with the mean average color of the pixels the white discs are over, it checks first checks within its list of colors for the nearest color and replaces the blob/disc with that color. This performs a crucial step. When the replacement happens, the module also generates an array of the X,Y location of the blob AND includes a label that we create along with those coordinates. The label is important since each green M&M can have very different colors. In order to count all the 'green' M&M's we need to force green colored M&M's into a single label. The nearest color search does this. In other words a blob with mean of 0,245,33 and 4,246,2 which are both green in color will be labeled with 'green'.



Because we now have an array of blobs (BLOB_COLORIZE_LABELS) and their labels we can use a basic VBScript to loop through all the elements in the array in order to count the occurrence of each color.

```
green=0
blue=0
red=0
yellow=0
orange=0
black=0

list = GetArrayVariable("BLOB_COLORIZE_LABELS")
if isArray(list) then
  if ubound(list)>0 then
    for i = 0 to ubound(list) step 6
      if list(i+2) = "green" then green=green+1
      if list(i+2) = "blue" then blue=blue+1
      if list(i+2) = "red" then red=red+1
      if list(i+2) = "yellow" then yellow=yellow+1
      if list(i+2) = "orange" then orange=orange+1
```

```

    if list(i+2) = "black" then black=black+1
    next
end if
end if
end if

SetVariable "Orange", orange
SetVariable "Blue", blue
SetVariable "Yellow", yellow
SetVariable "Red", red
SetVariable "Green", green
SetVariable "Black", black

```

The main part of the VBScript code is

```

if list(i+2) = "green" then green=green+1

```

for each color which simply increments a variable (with the same name as the color we are counting) each time that label is seen in the array. By processing the full array we can get a count for each color. These are then set into RR variables so that we can use the [Display_Variables](#) module to show the results. By using the [Math](#) module we can merge the full color results back into the original image to verify that we didn't miss anything.



Technique #2

The first technique works well by utilizing the circle module to find the initial M&M's. This works well for circular objects but suppose we had other objects that could not be so easily discovered by their shape. In the second technique we assume the objects are not circular but instead JUST rely on color.

The first step is to reduce the actual number of colors in the image. Any image made up by 1000s of pixels will have very subtle color changes which are not exactly the same. By using the [Reduce_Colors](#) module we effectively reduce the image into blobs of similar colors. These blobs will never be perfect in terms of circular shapes but we have to start somewhere.



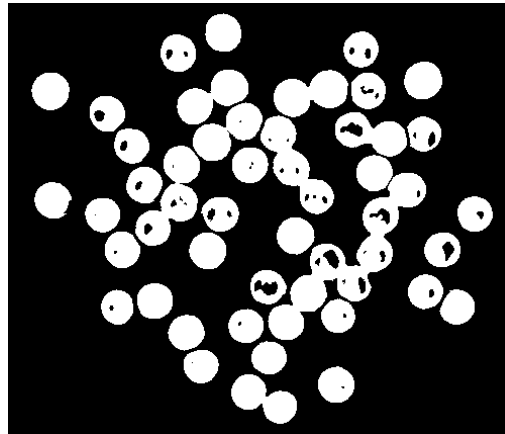
Now that the image has been reduced to a few colors, we want to remove those parts of the image that are not relevant to our analysis. This includes the tan background (originally the wooden table) and parts of the image that make up the shadows of the M&M's. The Circle detection

module in Technique #1 avoided this problem.

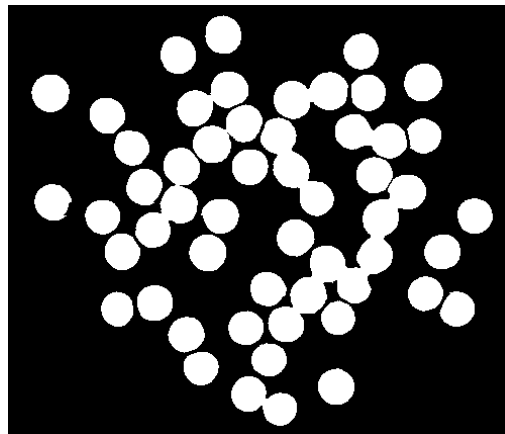
The best module for removing unwanted blobs is the [Blob_Filter](#) module. This module is perhaps the largest module within RoboRealm because it encompasses so many characteristics that can be filtered. Keep in mind that the blob filter will see each different colored group of pixels as an individual blob. The module reports that 202 blobs are found before the module runs. As we only want around 60 actual blobs we have to do a bit of work to remove the 140 that we don't want.

At this point we could also go back and use a circular shape as an attribute to remove all non-circular blobs, but we will stick with color and size to show more generic filtering. In our case we remove objects smaller than 215 pixels since our M&Ms are larger than that, remove objects larger than 5000 pixels (which eliminates the background table blob) and then remove any blob that doesn't have a strong color. Objects such as shadows will not have a strong hue and thus are removed. The problem, however, is that same filter ALSO removes the black M&M's ... since after all black is not a very strong color!

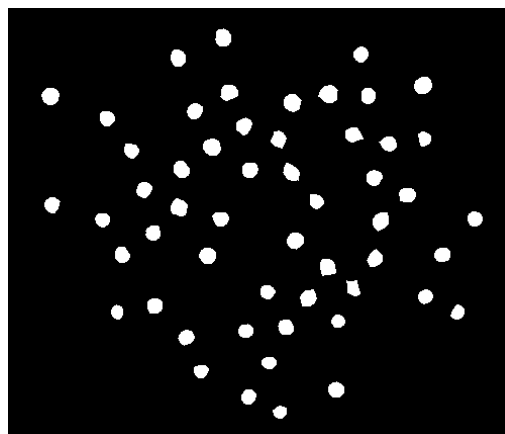
To remedy this issue we have a second filter within the [Blob_Filter](#) module that selects blobs that are much darker than most in the image. This combined with a size filter successfully selects JUST the dark M&M's. This result is then merged into the primary or Main filter to create the final image which includes both colored AND dark blobs.



The problem now is that many of the blobs are still touching each other. In order to count them correctly we need to split them into individual blobs. Before we start splitting we need to fill in the empty holes in the blobs. This is done using the [Fill](#) module.

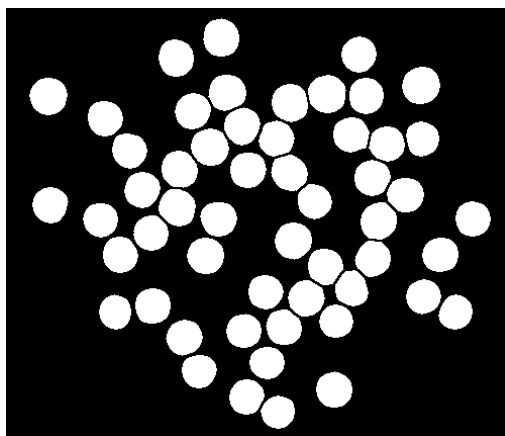


To split the blobs we separate them using the [Erode](#) module. By eroding the blobs by a 10 pixel perimeter which will break them into individual blobs.

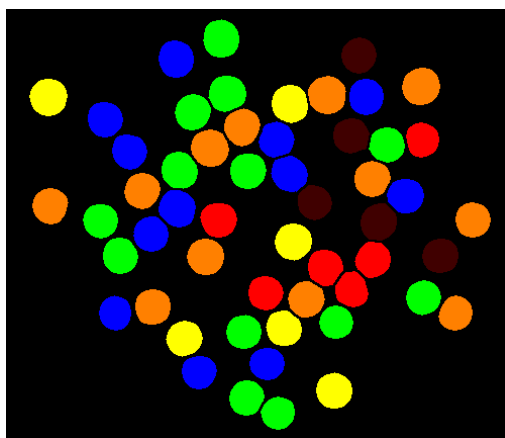


Because we want to preserve the approximate size of each M&M, we use the [Dilate](#) module to dilate each of the remaining blobs by 10 pixels.

Because we want to preserve the approximate size of each M&M we then use the [Dilate](#) module to dilate each of the remaining blobs by 10 pixels BUT we ensure that no blobs will reconnect. This will then result in each blob being separated from each other.



We can now follow a similar process as Technique #1 by utilizing the [Blob Colorize](#) to color in each of the remaining blobs with their source mean color.



and then use the same VBScript to sum up the results.



While the results are the same for both techniques, the pixels that determines a final M&M is a little different. You can see that the color technique does use more of the pixels to define an M&M (i.e. the final area is a bit larger) but as that was not a final measurement we were interested in either technique will work.

In terms of performance, the color technique #2 (while including more modules) is 40% faster than the circle detection technique #1 just for this specific image.

Download the [Count using Circles robofile](#) to see Technique #1 and [Count using Color robofile](#) to see Technique #2.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Object Segmentation



One of the most basic operations in machine vision is to segment objects from the background in order to run specific tests against only those areas of the image that are of interest. When lighting is not ideal, it becomes problematic to deal with shading and highlights especially from metallic parts. This tutorial shows one possible technique for segmenting various tools from a background image with a dominant light in the lower right corner.



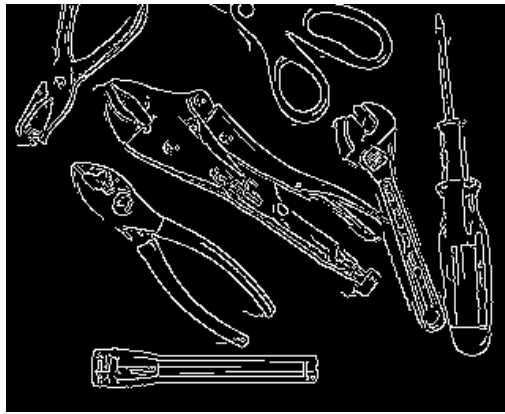
The above image shows the example we will be working with. On initial thought, it may seem that the tools can be extracted from the background using a simple [threshold](#). If we try that at threshold value 168 (manually tweaked) to get the best results, we can see that most of the tool parts are extracted from the mostly white background but as the lighting source comes from the right side a lot of shadows are generated to the left of each object.

Also note that as the threshold removes more of the shadow parts of the metallic tools are also thresholded out. This is not ideal as we want the tools fully extracted from the background.

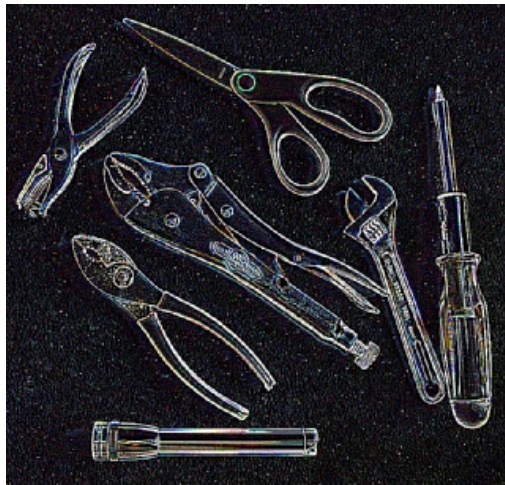


The next thought would be to use an edge detector like the [Canny](#) edge detector. Changing the values for the best results Theta at 0.1, Low Threshold 20, High Threshold 40) we get reasonably good results but still are missing large parts of the tools as the lines are not completely connected. Also again due to the shadows some of the lines wander off due to the low threshold value. As the results are promising, there is definitely a need for edge detection as part of the solution. For example, the flashlight (lowest object) is detected well enough for extraction.





Instead of the [Canny](#) edge detector, we instead use the Outline setting of the [Convolution](#) module. The Convolution module is a generic module that allows us to perform matrix like operations on a image in order to achieve a desired effect. In this case the Outline setting (with a 0.6 divisor to make the results brighter) is used like a high pass filter. I.e. a filter that will highlight noise or edges in an image. This results in a much more grayscale image than what one gets from the Canny module. One can also notice that the shadows are not very prevalent due to the slow intensity changes that constitute shadows.



One might be tempted to then use a threshold module on the resulting image to extract just the outline of the tools. This almost works but we found that the ideal threshold level includes a lot of noise (unwanted particles) that cause the resulting outline not to be very well defined.

Instead, we subtract this result from the original image to better define the edges within the original image.



This has the visual effect of improving sharpness of the image. On closer look we can see that the edges of most of the tools has now improved. Combining those edges with the original image allows us to effectively combine more than one edge detection technique in order to detect all edges. The next 'edge' detection technique is a simple [threshold](#) at level 63. If we compare this thresholding without the subtraction of the detected outline it becomes clear that the subtraction helps considerably.





We then remove a lot of the extra noise created by the original outline detection by removing those objects smaller than 70 pixels in area.



From here, we connect any dangling lines that are close to each other (<15 pixels away) and use the [Close](#) module to connect close parts of the outline.



Because we now have a well connected outline, we can fill in the remaining open areas within the objects in order to create solid blobs. The [Fill](#) module does this quite easily.





To see how well we did, we use the [Border](#) module to create a border from these objects, color that border and overlay it on the original image. This creates a red outline that indicates the boundary that was found using the above technique.



The final results are usable. While not perfect it does a good job on segmenting out the objects from the background. Note that the result assumed that objects do NOT contain inner open areas such as the holes in the scissor handles. In some cases this is ok, in others it is not. Further work would be required to extract out these inner holes or to split the objects that are close together and touching.

Download the [Object Segmentation robofile](#) to see the steps and more about the modules used.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Bottle Cap Inspection



Vision is often used for product verification. This tutorial shows how to locate a bottle cap within an image and determine if it is correctly fitted on a bottle. This would be a typical process to ensure that a bottle is correctly capped prior to packing.

Lighting

The most important aspect of any vision project is lighting. Bad lighting leads to bad results. In our case of bottle cap inspection, the best lighting is one BEHIND the object or called backlighting. The reason for this can be seen in the two following images. One is has lighting from the front and the other from the back. To us this may not seem like a major difference but when we begin the process of identifying the shape or silhouette of the object as the first step in inspection the difference becomes clear.



Using a very simple image threshold the best segmentation of background versus foreground becomes very difficult with the frontal lighting. This is because when the lighting is from the front specular reflection appears in the object which can cause the intensity values of the pixels that represent the object to appear the same as the background. Backlighting solves this issue by ensure that the background is always much lighter than the actual object itself.

If you are unable to place a light directly behind the object use a white background with the light offset from the object. This will reflect enough light from the background to achieve the same result.



Position

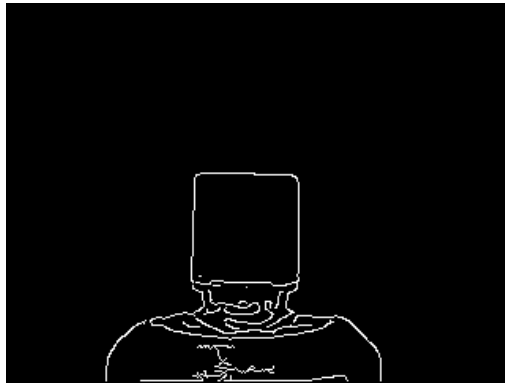
Now that lighting has been resolved, we can experiment with how the object will appear in the camera image with respect to position. The more rigid the object placement the easier the analysis becomes. For example, if we can always guarantee that the object is in exactly the same position in the image without any rotation in any axis nor a scale difference then the analysis gets a lot easier. The reason it gets a lot easier is that we can place probes exactly at the know location to check for any irregularities.

In reality this perfect placement is VERY unlikely. In our current case, we will assume none of these are true in that the bottle top will move around the image, have some out of plane rotation and some scale change. To get started we chose a sample image:

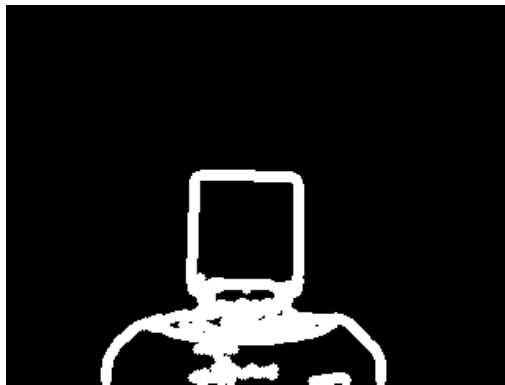




First we identify all the edges within the image using the [Canny](#) module.



Then, to ensure that all edges are well represented we use the [Dilate](#) module.



We then use the [Side Fill](#) module to highlight everything outside of the bottle. By filling from the top in a waterfall style and stopping when a non-black pixel is encountered we achieve an top down outline of the bottle with cap.

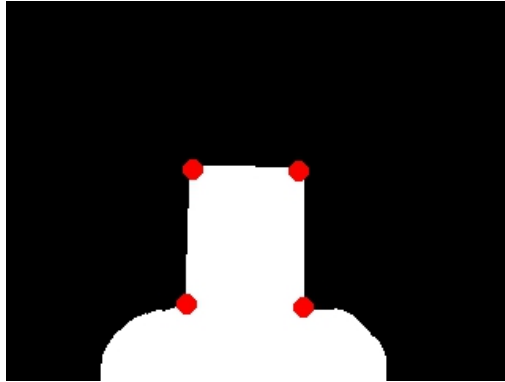


As RoboRealm likes to work with white objects we swap the image to its [Negative](#).





To identify the cap we want to identify the four corners that constitute the cap plus the neck. The reason for including the neck is to ensure that the cap is a certain distance from the bottle shoulders to ensure that the cap is solid on the neck. This is done by identifying the corners of the cap plus neck using the [Sample Curve](#) module.

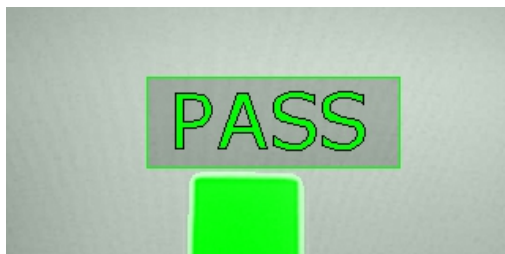


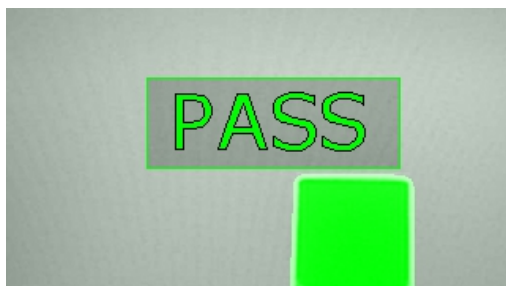
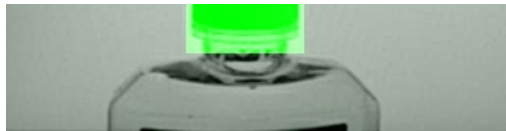
Because we are not interested in the bottle shoulders we draw a black rectangle below the neck to eliminate that from consideration. We setup the Y variable using the [Set_Variable](#) module and the known Curve points (4x).

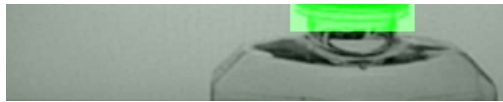


To determine if the shape is correct we then use the [Geometric_Statistics](#) to get the Aspect Ratio of the final shape. The aspect ratio is chosen because it is scale invariant and provides an accurate enough measurement to determine if the resulting rectangle has the correct dimensions. If the rectangle is too high caused by the cap not being tight on the bottle neck the ratio will be off. If the cap is missing or tilted the aspect ratio will also be incorrect. In addition, if the previous 4 curve points were not found we can also declare a FAILED part since something in initial edge shape was incorrect.

As with any machine vision project you need to test with many images in order to verify a correct solution. The PASS or FAIL are determined and combined back with the original image to indicate what the system found during its analysis. Note result 4 didn't even generate the 4 curve points correctly.







Download the [Bottle Cap Inspection robofile](#) to see the steps and more about the modules used. You can download the original images used in this tutorial [here](#).

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!



Fixing Photos

We've been doing a lot of ebay sales lately to reduce our stock of robotic components. While we're not photographers we do know images. This tutorial goes over a couple of the techniques that we've used to correct our bad product photos into something much more appealing ... as we all know, good photos in ebay make the difference between a good and bad sale.

Following is a couple examples on how to fix bad photos using modules within RoboRealm.



GPS device original photo. Camera switched to bad lighting during photo shoot. Hard to read text and really see what the device is.

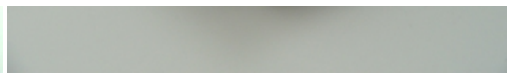
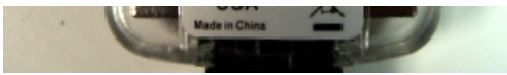


Using the [Normalize module](#) set to ignore 40% of the tailing values the image lighting is significantly improved.

Using the [Rotate module](#) we can rotate the image slightly (by 1.2 degrees) to straighten the image out.

FitPC device original photo. Background is not quite white.





Using the [Normalize module](#) helps to make it more white. But now the details of the device are washed out by the increased in contrast.



Adding in the [Raise Histogram module](#) will crunch the intensity values in a non-linear way to bring out lower intensity values better. This module will create a more grayish image but the normalize module undoes this. For objects that are dark colored this module really helps bring out the details and has been invaluable in improving electronic/circuit details.

Download the [fitpc_image_robotfile](#) to see the steps and more about the

modules used.



RFDuino (Arduino + Bluetooth) device original photo. Overall image is colored incorrectly. Bit too much perspective to get an idea on the size.

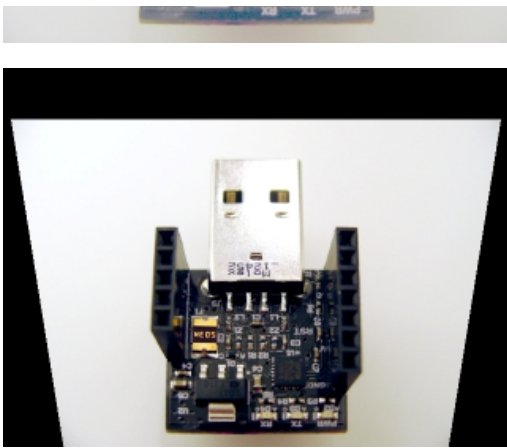


Using the [Color Balance module](#) we can easily fix the incorrect coloring (default settings)

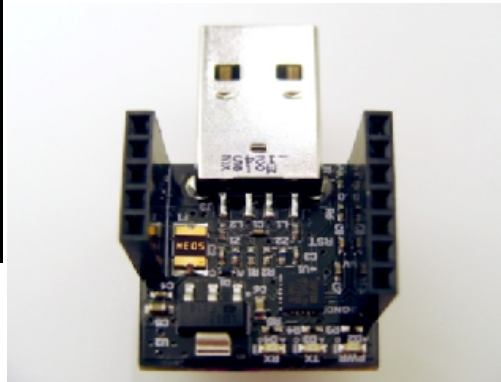
In perpetration of bringing out the details of the chip we use the [Raise Histogram module](#). This creates a not-so-nice looking image but increases the darker areas to be more visible.

Using the [Normalize module](#) fixes the visual issues that the raise histogram module created but still keeps the details we are looking for.





Using the [Perspective module](#) we can help reduce the perspective distortion (the board is square and not triangular) to make the image appear taken more from above than from the side.



Finally using the [Crop module](#) we can remove unwanted parts and scale to the final ideal size.

Download the [RFduino_image robofile](#) to see the steps and more about the modules used.



Lego RCX original image. Slightly out of focus and can't quite see the running man in the LCD interface.



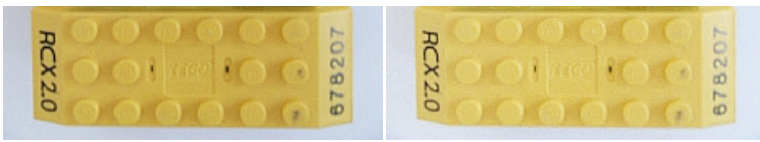
Use the [Rotate module](#) to get the right orientation.

Multiple use of the [Convolution module](#) with the Enhanced Focus setting to improve the clarity of the image. Note, there are limits to the usage of this module. Slightly out of focus can be corrected but the image will become very grainy after multiple uses.

Use of the [Raise Histogram module](#) to make the contents of the LCD image more legible.

Download the [RCX_image robofile](#) to see the steps and more about the modules used.





Laser Leveller original image. Background is dark but we want to keep the laser light.



Using the [Color Balance](#) fixes the color distortion that would be more apparent when Normalized. The overall image appears less yellow.



Using the [Normalize module](#) fixes the lighting but loses the laser light since the red color gets removed from the normalization process.



Instead we first use the [Hue module](#) that increases the laser color. Laser light typically overpowers a camera's pixels and produces more of a white light which is why the normalization process removes the line. Using the Hue module will increase that color. Note how much more red the laser light appears.



After the hue module we can rerun the [Normalize module](#) which now produces a good background with the red laser light. Also note how the red part of the leveller is also more pronounced.

Download the [Laser_image_robotfile](#) to see the steps and more about the modules used.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Recognizing Hand Gestures

The following tutorial describes one way to use a vision system to recognize hand gestures from an overhead webcam looking down at hand gestures over a solid background. The setup allows for various gestures (the digits 1 to 10) from one hand to be recognized by the vision system.

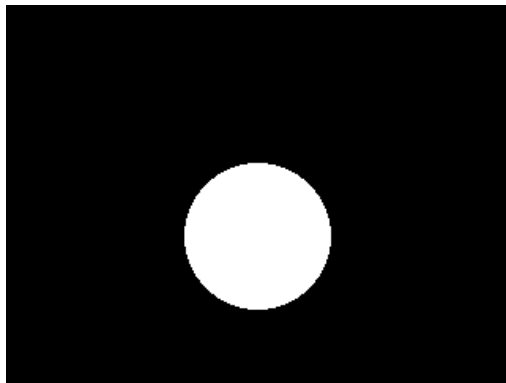
The basic principal to this recognition is to compare the shape of the hand once extracted from the background to a database of existing images that can be used to determine what digit is being indicated. Note that the digits 1 to 10 can be shown on a single hand with the hand in frontal view. We first start out this recognition process with a example image of what the camera sees.



The hand can easily be extracted from the background using a [AutoThreshold](#) module.



The problem now is that the length of the wrist/arm can vary depending on how far in the image the hand appears. This is enough to throw a shape detection system off from the real shape of the hand. To remove the wrist from the shape we replace the hand blob with the largest inscribed circle.



By dilating and subtracting this from the blob image we can separate out the individual parts of the hand.



The result can then be processed to remove any blobs that are touching the border of the image. Since the wrist should be the only part touching the

border of the image it can easily be removed using the [Blob Filter](#) to result with just the fingers.



By merging the remaining fingers with the part of the blob that was removed by the inscribed circle we remain with just the palm and the fingers of the hand. As the inscribed circle will be relative to the size and location of the hand this process is quite repeatable as long as the hand is frontal.



After smoothing this final shape to remove any irregularities the [Shape Match](#) module which is trained on the following images. Note that this database is trained on images previously saved from this point in the pipeline to the file system.



This module will generate a SHAPE_LABEL variable which can be used to indicate what shape graphic the module matched against.

The following [robofile](#) can be used to replicate these results along with the following [video](#). The database to train the Shape Module on can be downloaded as a [Zip File](#).

Note that the gesture for 10 is very inconsistent due to the failure of the inscribed circle method. Once we move the hand to a profile view the inscribed circle will move on each frame since the hand profile does not have a flat inner palm area that dictates the placement of the inscribed circle. Once this moves sporadically the rest of the algorithm will not function well.

Video

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Automated Dice Counting

The following tutorial describes one way to use a vision system to count the values of dice after they are rolled into view of a camera.

Our first example starts with an image of some colored dice.



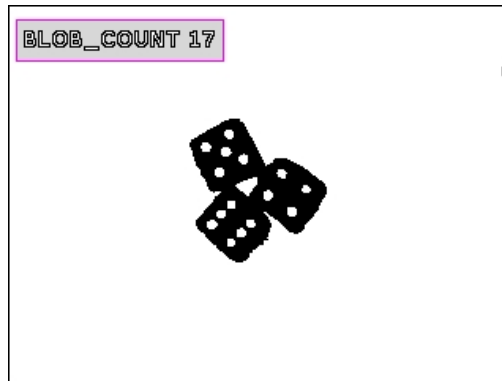
Our first impulse is to extract out the white dots by simply detecting the red color of our dice and then removing that from the image. But first we add in [Mean Module](#) with value of 2 to help smooth the color within the image. You will not notice much of a change to the unmagnified eye but this filter helps to soften the color. Then we apply the [RGB Filter](#) module to extract out just the red within the image. The Mean with Red extraction then looks like



We then select the Black Mask result of the RGB Filter to produce

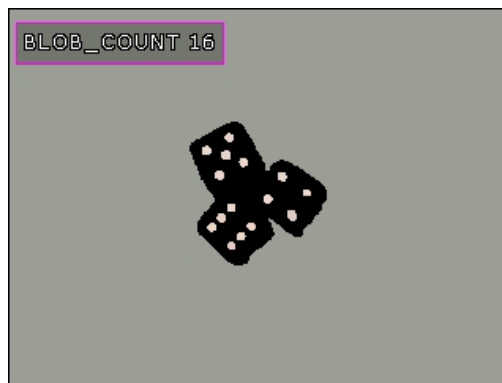


which shows the white dots plus the background white. We can now count the number of blobs using the [Blob Count](#) module. Finally using the [Display Variables](#) module we can display that count into the image. The result appears like



which we can clearly see is not the right count. We would expect the count to be +1 given the background white blob but 17 is incorrect (+2). The reason for this is in the middle of the dice a white blob is created due to the dice touching each other. We could remove this incorrect blob using size but that would restrict the camera movement to the same distance from the dice otherwise the actual spots may be removed if they become larger with camera movement.

Instead, we can use the [Blob Colorize](#) followed by the [Blob Filter](#) to remove all blobs whose original average intensity is below a certain amount. As the center blob is darker than the white spots we can use intensity to remove this erroneous blob.



You can see the effect of the [Blob Colorize](#) module in that the background blob approximates the color of the surface we are using (plastic table). Now the count is correct (minus 1 for the background).

When viewed on a live camera this count can flicker at times since the spots may momentarily become connected. This is normal camera noise which we can reduce by using an [Average](#) module which will average together several images (10 in our case) to produce a much more stable image. You do get some blurring when motion happens but not enough to cause us problems if the number of frames used is kept low.

We can then try a couple more combinations.



which all seem to work fine. The following [robofile](#) can be used to replicate these results.

The problem now is that regular black and white dice will not work! Given the image



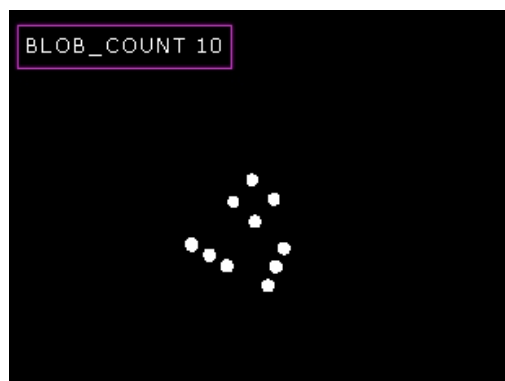
there is no red color and thus no dice are extracted. In this case we need to use a thresholding technique to isolate the spots (now black instead of white) against the rest of the image. We can do this using the [Auto_Threshold](#) module and the Two Peaks setting. The thresholded image set again to Black Mask appears as



but now our previous trick of removing the inner part of a connected set of dice will fail since the spots are now also dark!



In this case instead of using intensity to remove the erroneous inner blob we use the [Blob Filter](#) to remove blobs that are not very circular in shape. A setting of 0.60 will remove the inner blob and the outer background blob (its not a circle either) to leave us with just our circles that we wish to count.



The final count is then displayed correctly for the Black and White dice. The following [robofile](#) can be used to replicate these results.

However, when viewed with a camera the thresholding technique used (Two Peaks) can cause the image to momentarily jump into an incorrect state. As pixel values will contain noise any global thresholding technique can cause flickering in the image. A more stable technique to use is the [Adaptive Threshold](#) module which determines threshold values more locally and is more stable overall. Substituting this module results in final stable

dice counter script. You can download that [robofile](#) that is used in the movie below.

Video

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Lego PC Bot

The following tutorial shows the implementation of a PC robot constructed from a Lego NXT kit in order to create an onboard vision processing system. This platform is then capable of roaming around the home environment using the Floor Finder obstacle avoidance technique described in the [Obstacle Avoidance](#) tutorial.

The parts used in this construction are:

1. Lego NXT kit with USB cable ~ \$250.00
2. [Logitech camera](#) ~ \$100.00
3. [Asus Eee](#) Ultra Low Cost PC ~ \$350.00

The total cost of the platform comes to \$700.00. With some price shopping and camera substitutions that cost can be reduced by about \$100.00.



click on image for larger view

Construction

The Lego NXT kit while very flexible is not the strongest construction material to use when creating a robot to hold significant weight. Thus the choice of the Asus Eee is largely based on weight. There are some other models of PC based robots constructed using the NXT kit that can be seen in at the following sites:



[Mindstorm NXT + Eee PC = Good Times](#) - A robot using the NXT as only an interface from the motor's/sensors to the Eee PC using RoboRealm.



BB Vision Platform V1 Mounting a laptop ontop of a Lego robot using a USB camera and RoboRealm!



[Lego Mindstorm carrying Acer Aspire](#) An assembled structure to use as testbed for researching work.

Which are equally as good and can be used as additional inspiration during your construction.

Some of our requirements while building this robot were:

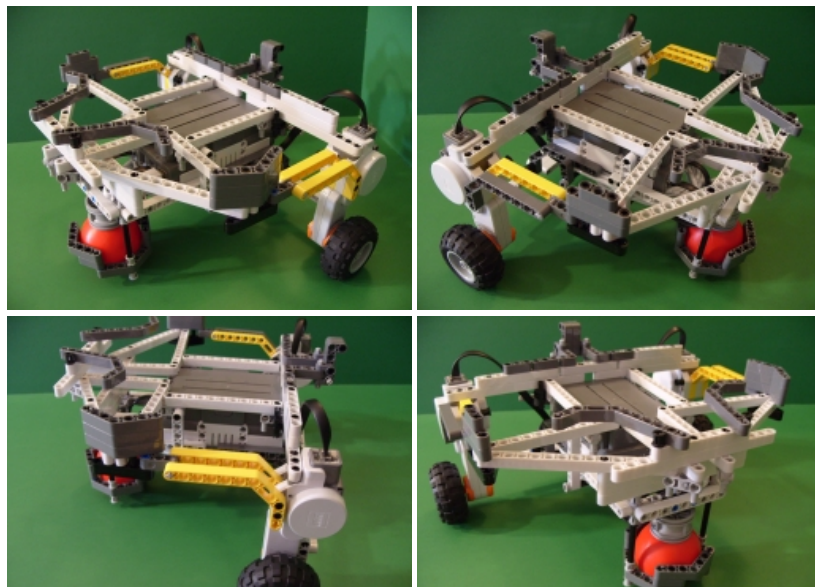
1. The robot must be built with only one kit's components.
2. No additional parts outside of the kit are to be used in platform construction (besides the Asus and Camera).
3. The laptop screen should be open and visible when in operation.
4. The keyboard should be able to still be used when in operation.
5. The laptop should be able to receive wall power without requiring removal of the laptop.
6. Switching batteries for the NXT should not be an involved process.
7. The camera should be attached to the front of the robot looking towards the ground.

In addition we also wanted to not introduce too much weight so that the robot would still allowed the robot to run over apartment thick carpet to ensure home use.

The most difficult process during construction was to ensure stability of the NXT system. Since a wide base (relative to the NXT) was require in order to hold the laptop the robot construction is inherently unstable and has a lot of "give" within the structure. After many hours of stabilizing and suring of the construction the system runs reasonably well but will tend to disintegrate in various places if the robot hits an unintended obstacle. Thus, while the platform is a great experimentation platform, it is by no means capable of running for extended periods without requiring some human "tweaks" to secure the less reliable parts of the construction.

Bot Base

To get an idea of the construction let's have a look at the PC Bot without the laptop or camera attached. Below you can see the scaffolding required in order to support the laptop. Note that the laptop simply drops into the construction and is secure enough for operation but will not be held in if the robot is tilted on its side or carried in any other orientation but upright.





click on image for larger view

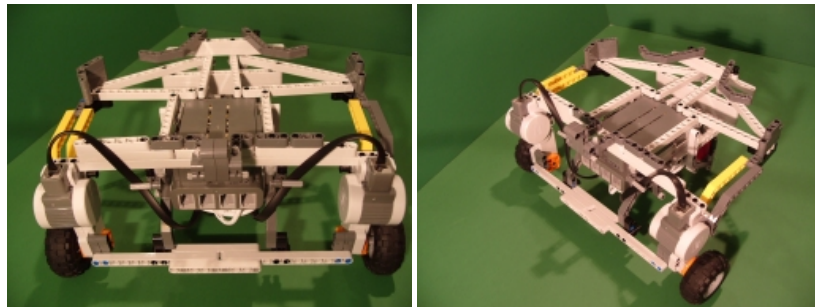
Laptop

The laptop fits snugly into this frame being supported by the web created above the NXT. It is supported in the front with the small wall created between the camera mount and the NXT based. You will also notice the asymmetrical size holders (the gray wall connected to the yellow pieces). The asymmetry was due to a lack of small straight pieces that were exhausted during the construction of this robot. This was not problematic as the laptop's width is slightly smaller than the span between these two side pieces which allowed for one side to use a more curved lego piece. The curved pieces exert more squeezing pressure and allow the laptop to better fit on top of the base.

Motors

We chose to position the motors in a vertical manner as opposed to angled as it made the planning easier to work with. It also allowed the NXT brick to be mounted underneath the laptop platform with enough ground clearance to avoid contact.

This orientation of the motors, while useful for this specific construction, do add to the instability of the system. When the wheels begin turning they cause the motors to experience a twisting torque between the wheels and the momentum of the laptop. Thus the robot has a tendency to "fall flat on its face" if the wheels encounter too much resistance.



click on image for larger view

You can notice the excessive connections to the motor in the front view. Note also the curved bar connection between the NXT and the front white span between the motors. This addition helped tremendously to stabilize the robot and prevent the "face falling" that would happen.

Camera

The logitech camera is built to connect to the top of a laptop. This might imply that a straight vertical attachment would be all that is necessary to clip the camera onto the robot. However, we found that the better attachment as a small 90 angle which would allow the camera to better clip onto the robot in such a way that it would not move vertically during operation. Note that this required a tricky 90 degree connection to connect a horizontal plan (the laptop base) with a vertically oriented piece.

Handles

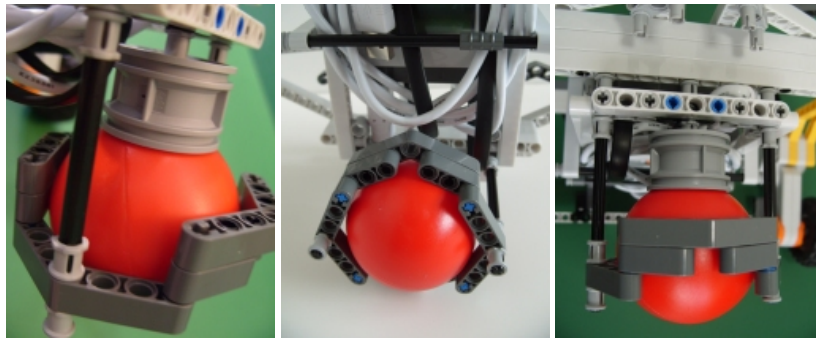
Every robot needs to have a handle which can be used to quickly grab the robot as it zips out of control towards the wall ... or worse off the tabletop! While we were not able to create a design that could be used to carry the robot with one hand we did settle on creating some grasp bars that do allow for easier pickup. You can see these yellow pickup bars connected from the motor to the outmost left and right sides of the laptop platform. These also created more stability in the system by connecting the motor and laptop platform but also serve as grasp handles using two hands to pickup the robot. Prior to these handles the robot would be grasped from the underside using the NXT as a base. This had the unfortunate effect of accidentally turning off or on the NXT base as the orange button on the NXT is slightly raised with respect to the NXT base.

Castor

As you will have already noticed the red ball provided with the Lego NXT kit was used as the laptop castor (or third wheel). We decided not to use the third motor provided in the kit as the rubber wheel would cause too much friction when being forced to the left or right. As the robot is meant for home use the plastic on the ball tends to provide much less friction on a carpet or floor than the third wheel. We also tended away from the other castor implementations seen in other Lego NXT constructions as they are too thin to operate reliably on a thick carpet and tended to cause jerkiness as the robot moved due to the castor wheel slipping into the current orientation.

We also determined that based on the laptop base a lot of weight from the laptop would be resting on the castor which would cause movement issues unless that weight was compensated for by the castor. Thus the red (or blue) ball would work well for distributing the laptop weight and allow the robot to slip over a thick carpet. While the ball mainly slips over the carpet when it does move (over hardwood) it provides a crude omnidirectional wheel.



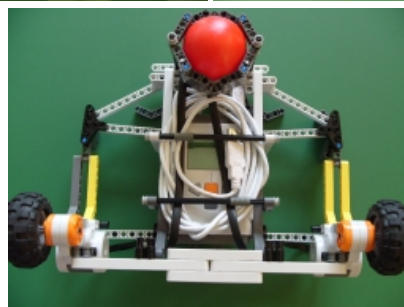
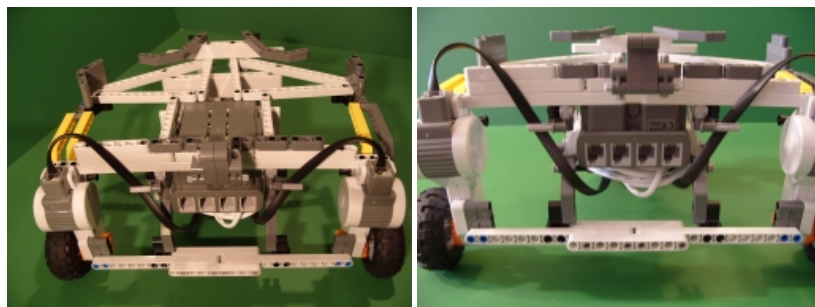
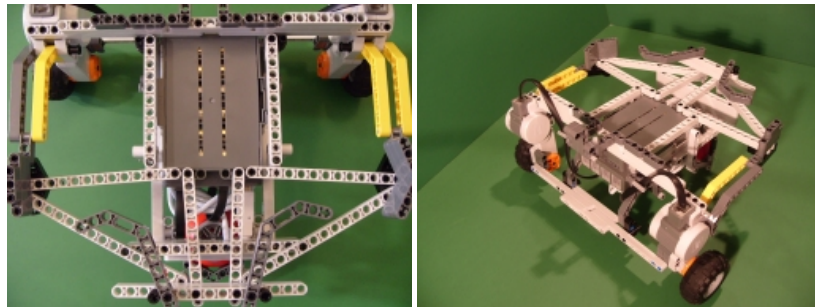


click on image for larger view

The trick behind the castor is to use the third wheel hub as the main vertical support. This allows the weight of the laptop to be distributed on the wheel and provides a good hold on the ball. The supporting lego pieces seen around the ball are used to hold the ball in position and prevent the ball from simply dropping below the wheel hub holder when in operation or when simply picked up. The left and right black bar supports of that structure allow for fine tune adjustment to get the holder into the correct position. With this tuning the ball would either slip out or would require too tight a hold which would prevent the wheel from turning at all.

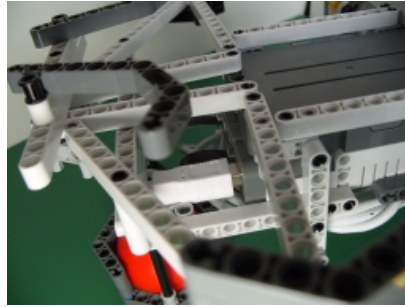
NXT

The NXT brick is at the center of the construction. It is used to start the web that becomes the laptop base. The NXT is mounted upside down to provide quick access to the on/off buttons that are now accessible from the bottom of the robot. The battery back can still be removed without requiring the robot to be deconstructed and would only require the laptop to be removed in order to change batteries. Normally we run the system with the batteries exposed (i.e. with the battery back) as they are held in place and are easier to switch out without having to remove the battery back (while removable it tends to stick to the sides when being removed).



click on image for larger view

Also note the packing of the USB cable underneath the NXT. We chose not to cut the USB cable down to size as this would require careful reassembly of the cable. You could probably purchase a shorter usb cable to reduce this packing requirement. We chose to use USB as apposed to bluetooth as the NXT connected via a USB cable is much more reliable than using bluetooth especially when in an unknown environment. USB also starts up quicker than the bluetooth connection and offers a audible beep when the USB becomes connected or disconnected from the NXT.

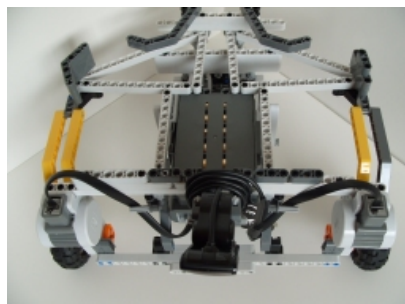
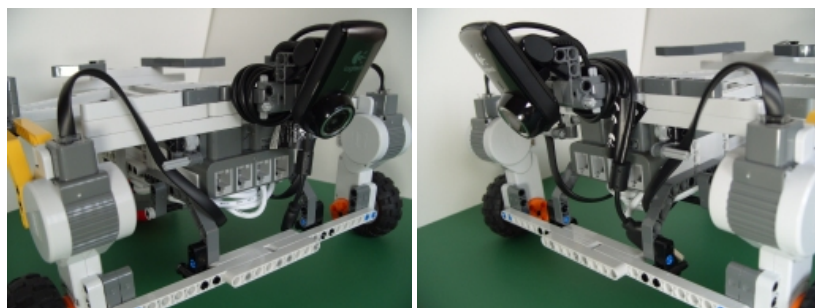
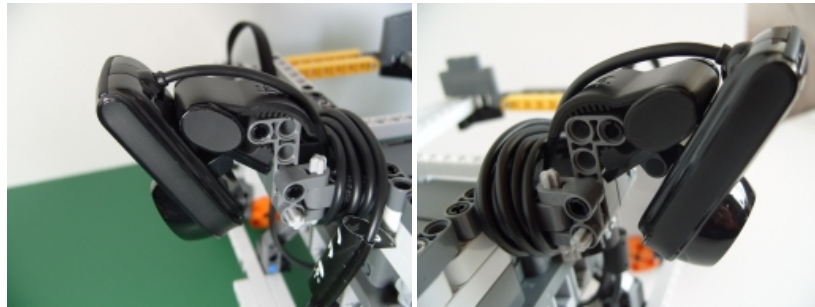


click on image for larger view

The USB connects to the NXT in the rear just above the castor framework. The other end is connected to one of the 3 USB connections on the Asus Eee.

Camera

As previously seen the Logitech camera mounts in the front of the robot. Using a 90 degree lego piece the camera gripper attaches quite securely to the front of the robot at an appropriate angle towards the ground.



click on image for larger view

Once the camera is clipped onto the holder the wire is simply wrapped around behind the camera to ensure that any tugging on the USB cable will

not change the camera orientation. This USB cable is plugged into the side of the Asus Eee and thus is somewhat exposed to obstacles on the side of the robot which might cause some tugging on the wire.

Computer

The Asus is now slipped into the NXT base and the two USB connections (one for the camera and the other for the NXT) are plugged in.



click on image for larger view

Now that the Bot is ready to go we need the software!

Software

We would like to use the Floor Finder obstacle avoidance technique described in the [Obstacle Avoidance](#) tutorial to move the robot around the room avoiding obstacles that are not the same color as the floor plane.

We load in the following [robofile](#) which performs this task specifically using the Lego NXT robot.

Once the video has started and verified we can turn on the NXT. The final task is to press the green Start button in the RoboRealm popup button interface to actually start the robot. This provides a quick way to start and stop the robot and is even possible to click while the robot is moving (tap the mouse pad to click the button).

Videos

Notes:

1. The biggest issue (after Stability of the platform) is the viewable area of the camera. As the camera is pointed towards the ground and the robot base is quite wide the camera cannot see a wide enough view in front of the robot in order to avoid all obstacles. It is possible for an obstacle to slip by the camera view and collide with a wheel. This issue can be solved by either using more than one camera (one for each wheel) or a camera with a wide field of view.
2. There are a couple of issues to solve to make this a reliable platform for continuous use. Improving stability by using a different design or more parts from a second NXT kit would help. In addition the Asus Eee will only last about an hour in terms of power. As the system does not have a charging solution continuous operation is not possible at this point. Note that there are two power systems in operation with this platform, the laptop being one and the other being the NXT.
3. Note that none of the other sensors available in the Lego NXT are NOT being used. One could improve the reliability of this implementation by adding in the other sensors. The sensor attachment side of the NXT is kept clear to allow for these additions. Again, all these sensors could work in conjunction with the onboard vision system either within the NXT or combined with the vision logic within RoboRealm.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

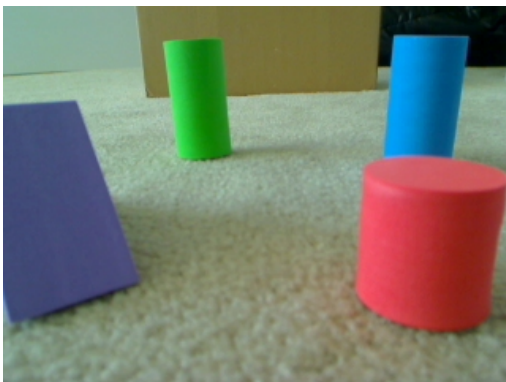
Have a nice day!

Obstacle Avoidance

Obstacle avoidance is one of the most important aspects of mobile robotics. Without it robot movement would be very restrictive and fragile. This tutorial explains several ways to accomplish the task of obstacle avoidance within the home environment. Given your own robots you can experiment with the provided techniques to see which one works best.

There are many techniques that can be used for obstacle avoidance. The best technique for you will depend on your specific environment and what equipment you have available. We will first start with simpler techniques that are easy to get running and can be experimented on to improve their quality based on your environment.

Let's get started by first looking at an indoor scene that a mobile robot may encounter.

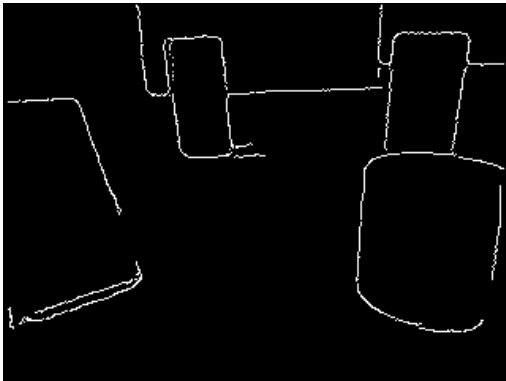


Here the robot is placed on the carpet and faced with a couple obstacles. The following algorithms will refer to aspects of this images and exploit attributes that are common in obstacle avoidance scenarios. For example, the ground plane assumption states that the robot is placed on a relatively flat ground (i.e. no offroading for these robots!) and that the camera is placed looking relatively straight ahead or slightly down (but not up towards the ceiling).

By looking at this image we can see that the carpet is more or less a single color with the obstacles being different in many ways than the ground plane (or carpet).

Edge Based Technique

The first technique that exploits these differences uses an edge detector like [Canny](#) to produce an edge only version of the previous image. Using this module we get an image that looks like:



You can see that the obstacles are somewhat outlined by the edge detection routine. This helps to identify the objects but still does not give us a correct bearing on what direction to go in order to avoid the obstacles.

The next step is to understand which obstacles would be hit first if the robot moved forward. To start this process we use the [Side Fill](#) module to fill in the empty space at the bottom of the image as long as an edge is not encountered. This works by starting at the bottom of the image and proceeding vertically pixel by pixel filling each empty black pixel until a non-black pixel is seen. The filling then stops that vertical column and proceeds with the next.



You will quickly notice the single width vertical lines that appear in the image. These are caused by holes where the edge detection routine failed. As they specify potential paths that are too thin for most any robot we want to remove them as possible candidates for available robot paths. We do this by using the [Erode](#) module and just eroding or shrinking the current image horizontally by an amount large enough such that the resulting white areas would be large enough for the robot to pass without hitting any obstacle. We chose a horizontal value of 20.

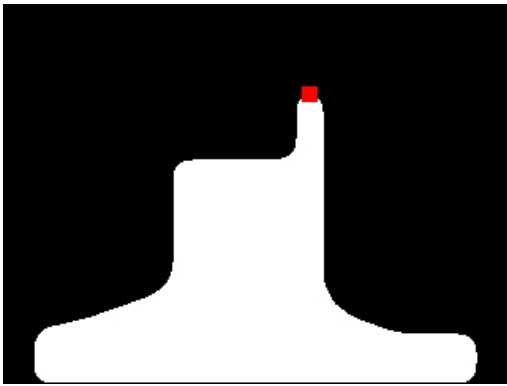


Now that we have all potential paths we smooth the entire structure to ensure that any point picked as the goal direction is in the middle of a potential path. This is based on the assumption that it is easier to understand the highest part or peak of a mountain as compared to a flat plateau. Using the [Smooth Hull](#) module we can round out flat plateaus to give us better peaks.

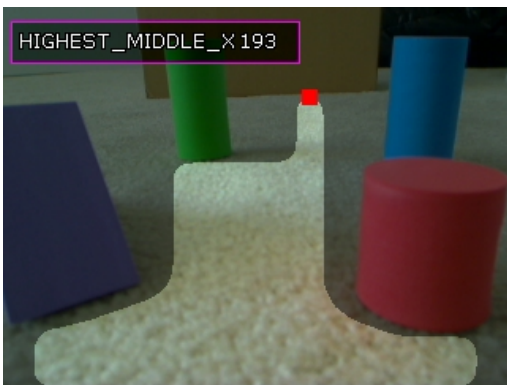




Once this is done we now need to identify the highest point in this structure which represents the most distant goal that the robot could head towards without hitting an obstacle. Based on the X location of this point with respect to the center of the screen you would then decide if your robot should move left, straight, or right to reach that goal point. To identify that location we use the [Point Location](#) module and request the Highest point which is identified by a red square.



Finally just for viewing purposes we merge the current point back into the original image to help us gauge if that location appears to be a reasonable result.



Given this point's X location at 193 and the middle of the image at 160 (the camera is set to 320x240) we will probably move the robot straight. If the X value were > 220 or < 100 we would probably steer the robot to the right or left instead.

Some other results using this technique.

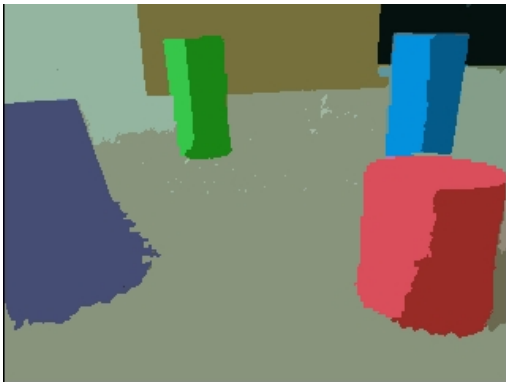




This works reasonable well as long as the floor is a single color. But this is not the only way to recognize the floor plane ...

Blob Based Technique

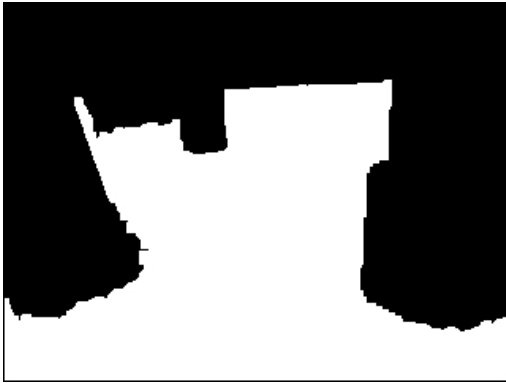
The second technique exploits the fact that the floor is a single large object. Thus starting with the original image we can segment the image into a smaller number of colors in order to connect pixels into blobs that we can process. This grouping can use either the [Flood Fill](#) module or the [Segment Colors](#) module. Using the flood fill module the image becomes



The next step is to isolate the largest blob in the image which is assumed to be the floor. This is done using the [Blob Size](#) module which is set to just return the single largest blob in the image.



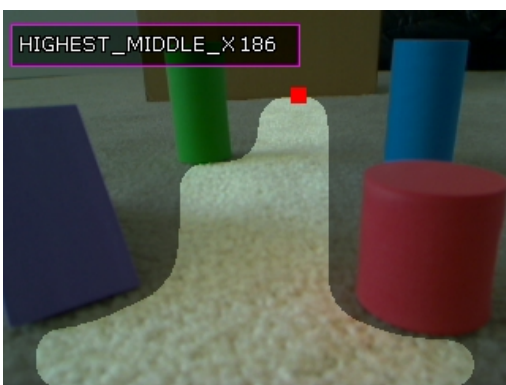
We then dilate this image by 2 pixels using the [Dilate](#) to close all the small holes in the floor blob.



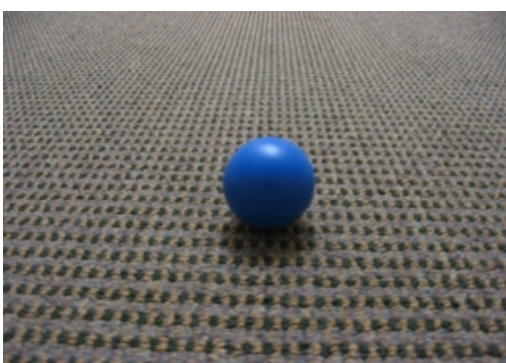
Then we negate this image and use the same [Side Fill](#) module as before to determine the possible vertical routes the robot could take. We need to negate the image prior to this module as the Side_Fill module only fills black pixels. In the above image the object to be filled is white and thus when negated will become black.

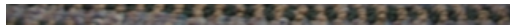


From here on the stages are the same as the previous technique. Namely [Erode](#) to remove small pathways, [smooth](#) the resulting object and identify the [top most point](#). The final image looks similar to the previous technique.

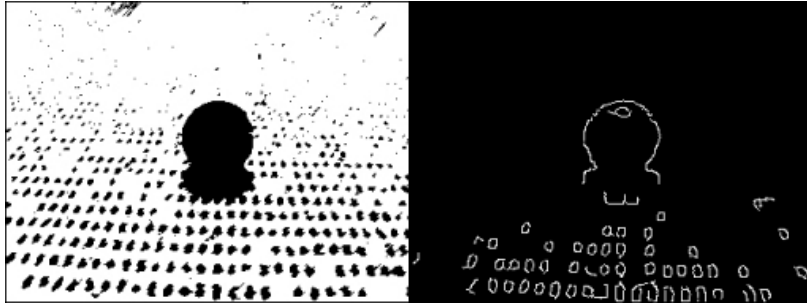


The results are very similar but the first technique exploited edges whereas this one exploited connected pixels of similar color. But the issue of the similar colored floor plane still remains. What happens if you do not have the same colored carpet? For example, suppose that you have a high frequency pattern in a carpet.





The resulting edge and blob based techniques will not work as the blob and edge detection will pick up on the small patterns of the carpet and incorrectly see them as obstacles.



You can notice the failure of both these techniques in the above images which if fully processed would only see non-obstacle space in the lower 10 pixels of the image. This is clearly incorrect!

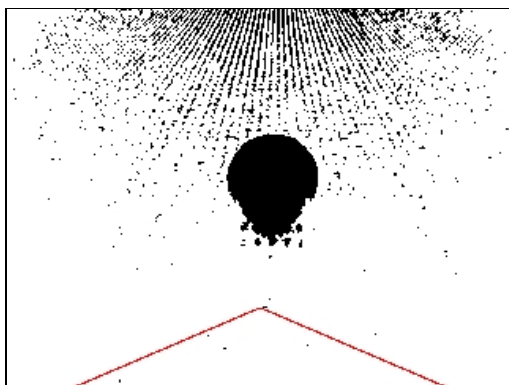
Floor Finder Technique

To improve on our last image we will need to understand that the carpet or floor plane contains more than one pixel color. While you could detect both colors and perhaps merge them in some way an easier approach is to make an assumption that the immediate foreground of the robot is obstacle free. If we were to sample the colors in the lowest part of the image which is the immediate space in front of the robot we could use these color samples and find them in the rest of the image. By searching for all pixels who share the same or similar color to those pixels in this sample space we can theorize that those pixels are also part of the floor plane.

This process can be accomplished using the [Floor Finder](#) module. Given our test image

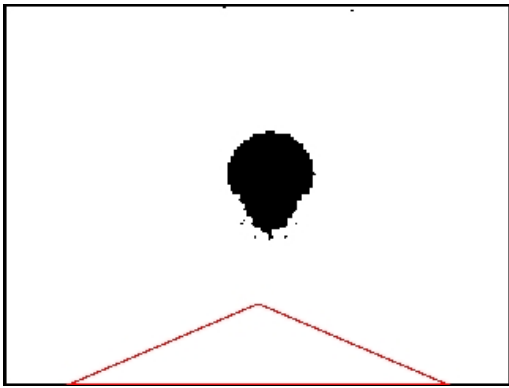


we can run the floor finder module to procedure a theoretical mask of what the floor might be.

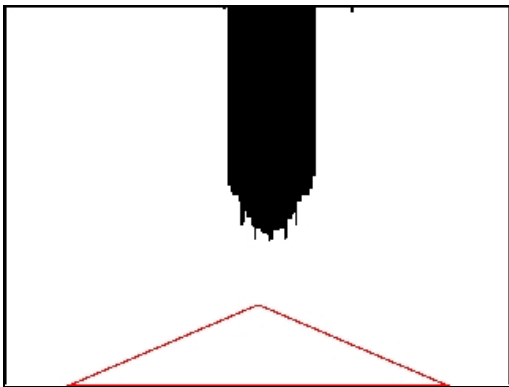


We can see the red triangle that represents the sample area that is used to understand what pixel colors are likely to be floor pixels. The white pixels in the above image now represent all pixels in the image that are similar to those found in the triangle. We can see that this works quite nicely to

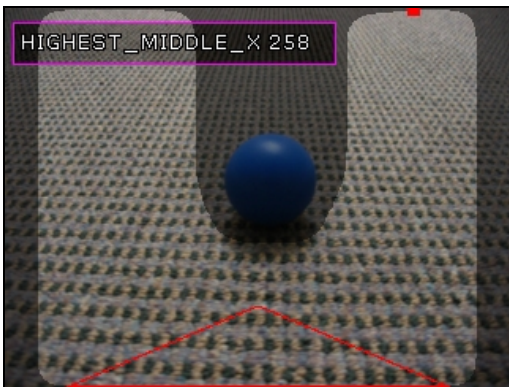
In the above image now represent all pixels in the image that are similar to those found in the triangle. We can see that this works quite nicely to segment out the floor plane. We then dilate the image to remove small holes in the floor plane.



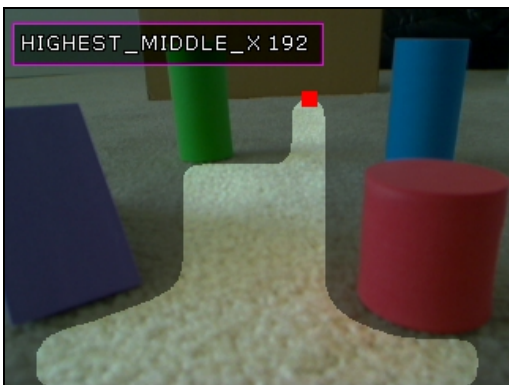
Then we negate this image and use the same [Side Fill](#) module as before to determine the possible vertical routes the robot could take. We need to negate the image prior to this module as the Side_Fill module fills black pixels. In the above image the object to be filled is white and thus negated it will become black.



Following a similar sequence as before we [Erode](#) to remove small pathways, [smooth](#) the resulting object and identify the [top most point](#). The final image is used to verify the recommended goal point.



We then run the same technique over the previous image to check if that still works.



Which it apparently does! So the floor finder module has allowed us to use a single technique for both similar colored carpet to one that has a lot of small internal patterns in it.

Obstacle Avoidance

Notes:

1. Three techniques were discussed with the final technique being the most stable given the test images we used. Your experience will be different. It is worth testing each technique on your own images to get a sense of how stable they are and which one will work best for you.
2. Keep in mind when moving the code into a robotic control that the horizontal erode is used as a gauge to the robot's width. You will have to experiment with your setup to determine which erosion width is best for your robot.
3. The final variable displayed has the X value of the goal point. That X value should be then used by your application or within the [VBScript](#) module to create a value that will control your servos. It could be as simple as

```
x = GetVariable("HIGHEST_MIDDLE_X")
midx = GetVariable("IMAGE_WIDTH") / 2
leftThreshold = midx - 50
rightThreshold = midx + 50
```

```
if x <= leftThreshold then
  SetVariable "left_motor", 0
  SetVariable "right_motor", 255
else
  if x >= rightThreshold then
    SetVariable "left_motor", 255
    SetVariable "right_motor", 0
  else
    SetVariable "left_motor", 255
    SetVariable "right_motor", 255
  end if
end if
```

which will rotate the left and right motors towards the goal or go straight if in the middle. Keep in mind that the above only creates two variables (left_motor and right_motor) that need to be selected in the appropriate hardware control module. You will have to adjust the values for your robot based on what servos, motors, etc. you are using.

Your turn ...

Want to try this yourself? Following is the robofile that should launch RoboRealm with the original image we used in this tutorial. You can click on each of the steps in the pipeline to review what the processing looks like up to that point.

 [Download Edge robofile](#)

 [Download Blob robofile](#)

 [Download Floor Finder robofile](#)

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Marble Maze

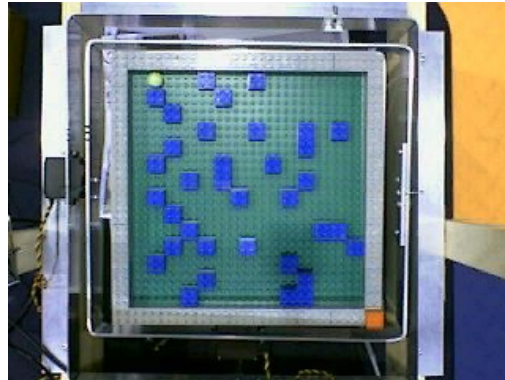
This Marble Maze tutorial shows how to use image processing and custom path planning to solve the maze by rolling a white marble from any starting point to a defined end point within the maze.

In this tutorial we use the setup from a partner site [Tele Toyland](#) called the [Marble Maze](#). The goal of the Marble Maze is to command the platform from over the web to pan or tilt in order to solve the maze by moving the white marble from the start point in the maze to the end point. The trick with this installation is that you need to take into account that the ball will not stop unless there is a Lego brick in its path and that the ball does not always roll in a straight line nor stop exactly where desired. Thus while the maze does not appear to have many "walls" in the maze it is nevertheless

a maze and requires some strategic thought to solve.

The task of this tutorial was to take the manual web interface and create a RoboRealm application that could solve the maze and move the ball from the start to the end. This required no additional hardware than is currently functioning. There is already an overhead camera and a servo board controller (in this case a Parallax USB servo controller) so the hardware build is already done.

The main task of automatically solving this maze is to process the image in order to create a idealized model of the maze and then solve the path planning issue using that model. Using the overhead camera we can acquire an image from the maze to process before each move is made. But first let's see what the maze looks like from the overhead camera's point of view.

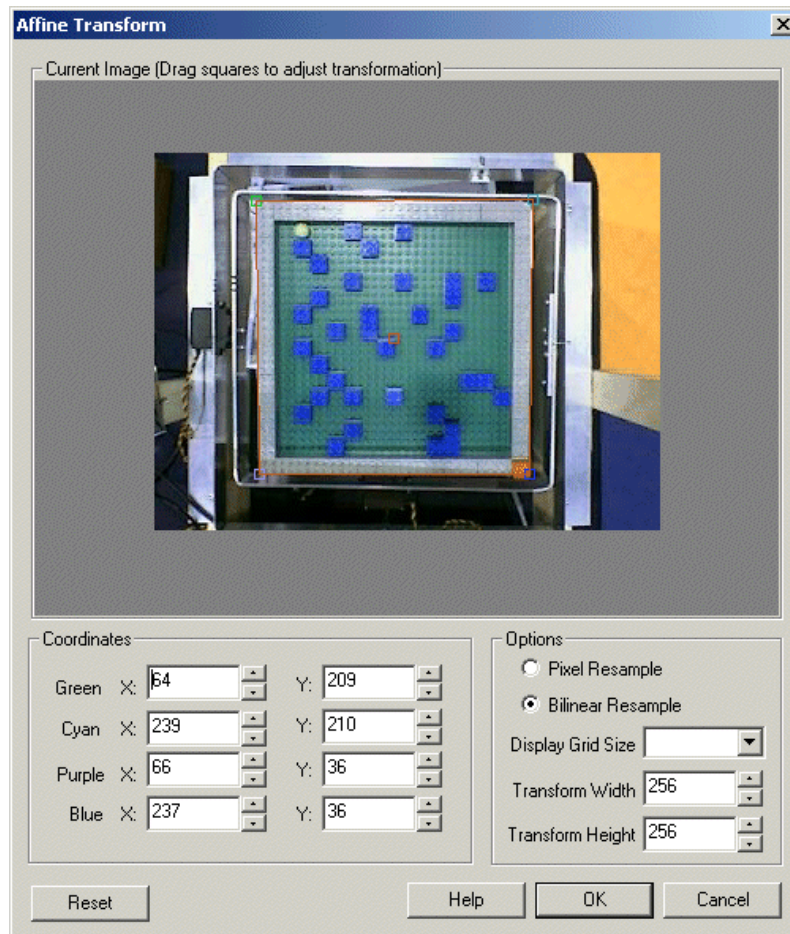


The image above shows the view from the overhead camera looking down at the maze. The main part of the maze is the green background which is the "floor" of the maze. The gray Lego blocks outline the border of the maze. The inner blue blocks are the maze walls. The white ball can be seen in the upper left corner with the destination goal in the lower left specified by the red square.

On first view you can notice that the camera is not perfectly centered above the square maze. Our ideal model of the maze would be a perfect square. In order to align the image better we have to start the image processing part of the tutorial.

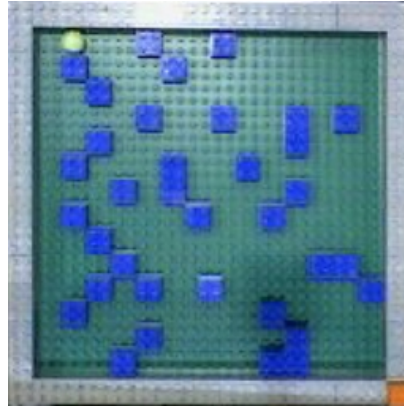
Image Alignment

In order to square off the maze we use the [Affine module](#) and outline the gray square using the provided GUI interface of the Affine module. The interface will look as follows:



which when processing the previous maze image will align as a nice square. The resulting image is scaled to a 256x256 image (the affine module is

also used to scale the image to a specified size).



Why is this needed? ..

Maze Model

To solve the maze we will need to know 3 things:

1. What is open (green) and what is blocked (blue)?
2. Where is the ball?
3. Where is the destination?

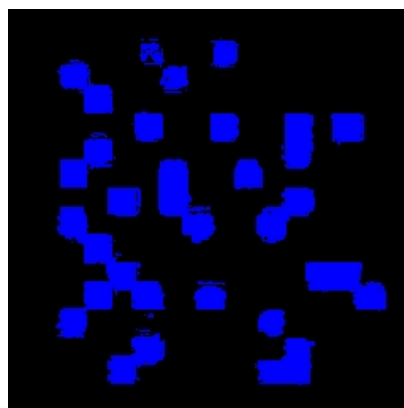
#1 is best represented (or modeled) as a two dimensional array with each array bucket containing a number that represents if the spot in the array is open (green) or blocked (blue). This can be thought of as a bitmap image with each "pixel" representing if the spot is open or closed. Note that you cannot use the image bitmap directly as there are too many pixels that represent an "open" or "closed" spot. The current image needs to be simplified to an ideal model.

#2 and #3 would then just be two x,y coordinates within that two dimensional array.

Thus the ideal maze model is a two dimensional square array. Since the maze is created using Lego blocks there are only so many blocks that can be placed in the maze. In this case the maze can contain a maximum of 14 2x2 blocks in either the horizontal and vertical direction. The array can then be a 14x14 array which is nice and small and can be quickly processed. Therefore, the first part of transforming the image into a small 14x14 array is to align the image into a square. With this alignment the image is now a scale factor representation of the array.

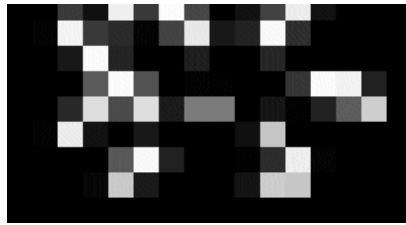
In order to continue the transformation we cannot simply scale the image into a 14x14 bitmap as the resulting array would contain variations of blue, green and gray pixels. This would still be hard to process as it would require a range of pixel colors to be interpreted in order to know that a current position is free or an obstacle.

Instead, the next step in our simplification process is to filter the image for the blue blocks using the [RGB Filter](#) module. We then remove small blobs as they are mainly due to noise and are not part of the blue blocks.

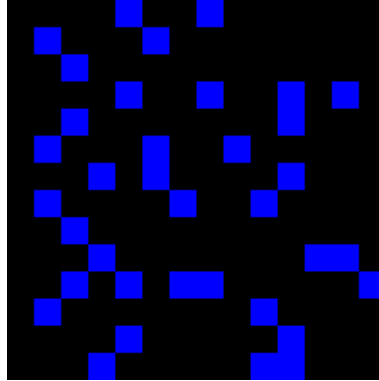


As we can see this certainly helps to isolate the blue blocks. We then scale the image down using the [Scale](#) module to a 16x16 image. For viewing purposes the following is that image scaled back up to a larger size so that we can see each individual pixel.



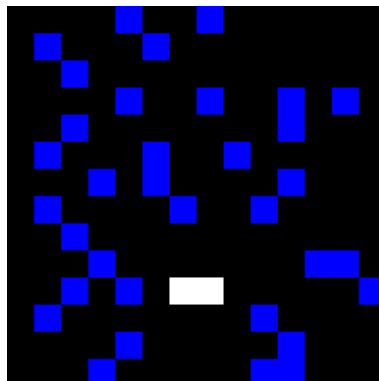


We can see that the blocks are better defined (in terms of accuracy) but we still need a blocked/unblocked signal instead of a varying gray value. We can do this by [thresholding](#) the image. We also crop the image as the border pixels are not needed. This creates a 14x14 sized image.



With the final threshold we can see the blocks nicely defined within a grid. This image can now be accessed as a 14x14 bitmap image which will have an RGB value of blue (0,0,255) in those positions where a blue Lego block occurs otherwise each pixel will be zero (black color).

If you notice closely you will see a single blue block in the lower part of the image being represented as two blocks instead of one. This is due to a single block being offset outside of the 2x2 grid. This breaks our 2x2 alignment requirement but is included to illustrate what happens when this requirement is not met.



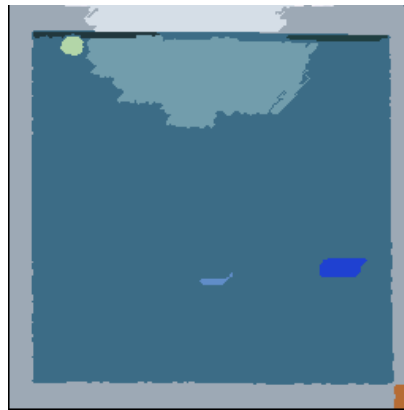
Now that we have the idealized grid model let's find out where the ball is!

Where's the Ball

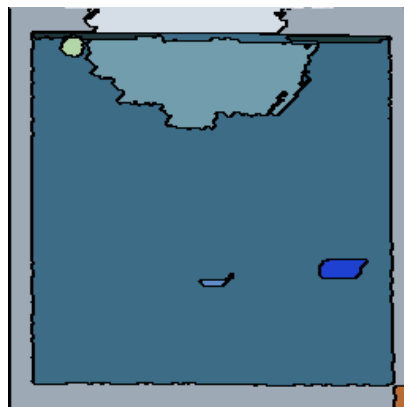
Given our original image we now need to process the same image for the white ball. We can revert RoboRealm to the original image by using the [Marker module](#) to save the current image and revert the viewable image to the aligned source image.



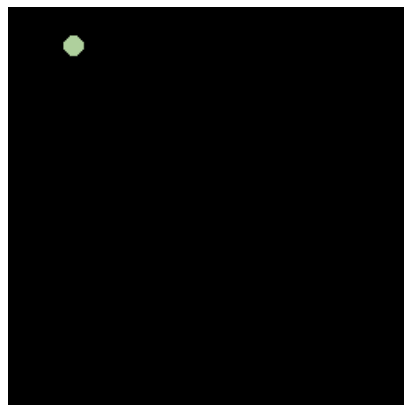
We'd like to separate the white ball from the rest of the maze. The first step is to collect all the white pixels into a blob that can be processed at a higher level in order to determine which one is the white ball. We start this by using the [Flood Fill](#) module.



This helps to group pixels into like colors. Note that as the white ball is considerably higher in intensity from the rest of the image we can use a very large threshold value. While this causes the green and blue colors to merge it does keep the ball separate from the background. As we have already processed for the blue blocks we are not concerned about this green blue color merging. Now we can separate each of the blobs using the [Blob Separate](#) module so that they can be processed individually.



we can process the image for the strongest circles using the [Circles](#) module. Note that using the flood fill module to first group like pixels as apposed to just detecting edges straight from the source image as is normally used as a precursor to the Circles module helps reduce the amount of false circles detected. False circles might be detected due to lighting changes that momentarily cause a circle to be seen.



Once the circles are detected each circle is filled with the mean color of the pixels that makeup that circle using the [Blob Colorize](#) module and checked to see that this value is high (i.e. towards white). This blob removal processing is done by using the [blob filter module](#). What we are left with is a single circle that best represents where the ball is (i.e. it is a white circle). The center of gravity of this circle is then considered to be the X and Y coordinate of the ball. Note that this coordinate is also divided by 14 in order to ensure that the values are within the range of the previously

take the coordinate of the ball. Now that this coordinate is also divided by 14 in order to ensure that the values are within the range of the previously created 2x2 dimensional array that represents the maze.

Next we move to the destination square. This is easily determined by filtering the image for red once again using the [RGB Filter](#) module and then filtered for the largest red square using the [Blob Filter](#) module as seen in the image below. The resulting center of gravity coordinates of the red blob is then also scaled into the 14x14 array.



Now that we have all the information we need we have to determine how to calculate the route from where the ball is to the destination red square using a custom ball rolling path planning algorithm.

Ball Rolling

The trick to understanding how the ball moves is to assume the ball will roll in more or less a straight line and stop only when colliding with a blue block or the gray border. This action can be simulated in our model by using a search along a specific horizontal or vertical line for as long as the value of our model or array is zero.

As the ball may not stop or roll in the manner that we expect we only need to make a single move, restore the maze to horizontal and then take another picture to see what has actually happened. Despite the need for only the very next move we still need to determine the entire path from the white ball to the destination as the next move can only be calculated by knowing the entire route.

We calculate this path by starting at the balls current location and start searching in all the 4 possible directions for the next stop. As we only assume north, south, west and east ball movements we only need to check the possibility of the ball rolling in those directions. This process then repeats in each of those 4 new locations for the next 4 and so on. By checking if any of those new locations is the destination we can search the entire maze for the quickest way to get to the destination, i.e. the red square destination coordinates terminate the search.



The above image show the first four search levels. Note that at every stop point all those directions (other than the one just came from) are explored if there is no obstacle present in that spot in the array. This expansion can increase the total number of search paths quite quickly, however, as we have a relatively small maze this is not currently an issue and no search pruning is required.

The actual algorithm is known as a breadth first search as it will guarantee that the shortest solution (the one with the least ball rolls as apposed to actual distance) will be found first. This algorithm is done in VBScript and implements a queue structure for the search. As each new stop location is found it is added to the queue. The queue then grows with new search locations being added to the end with the top of the queue being the point of new exploration.

See the commented VBScript code contained in the VBScript module for further details.

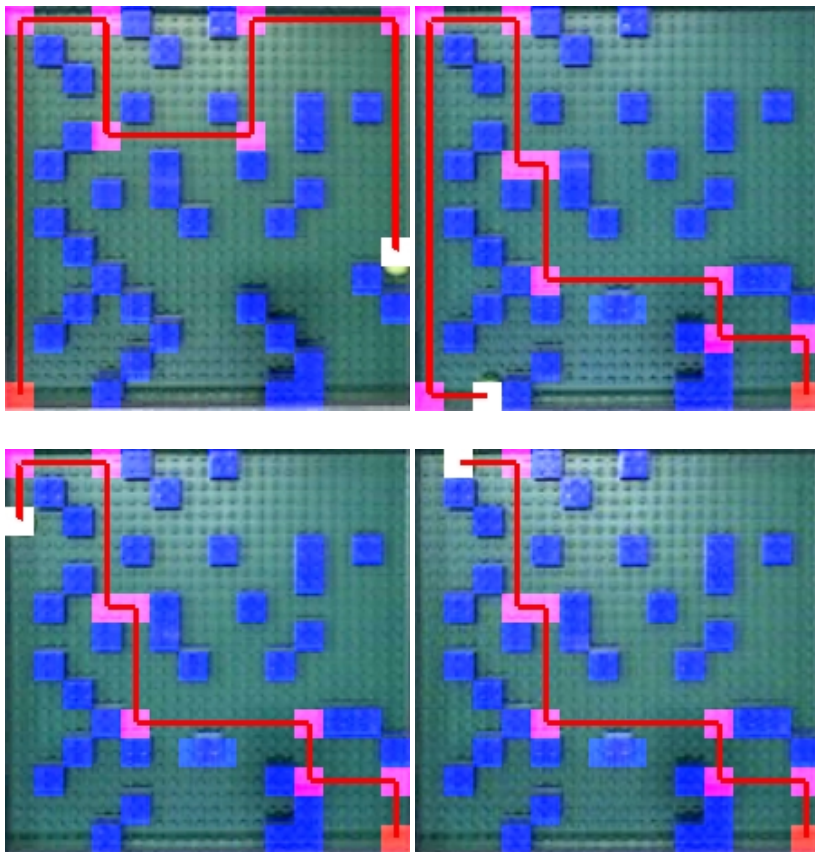
Marble Maze

Video

Video of the marble maze being autonomously solved even with a large roll error.

Results

The following are image captures of the final route as determined by the image processing and route planning. The white square is on top of the ball, the red square is the location closest to the destination red square and the purple squares are the direction changes due to the blue stops in the maze. The path is shown in red that proceeds from the white square to the red square. Note that this planning is performed after every move in case the ball rolls in an irregular direction or bounces off from the blue/gray blocks due to roll momentum.



Issues

Alignment

The alignment of the square maze is key to the correct processing of the maze. With minor misalignment some of the blue squares may bleed into neighboring areas and cause a false obstacle to be detected. This may or may not cause an issue with the maze depending on if the found route depends on that obstacle. If the solution found requires that the blue block exists then it may cause the system to oscillate around a nonexistent solution. In addition, the false detection of the blue block may cause the maze to be defined as unsolvable (i.e. the image will be set to all white).

Lighting

Other issues are due to lighting. Lego blocks and surfaces are actually very reflective and can cause substantial lighting halos that cause the detected color to become over saturated and not detected. This is an issue for all detected objects (blue blocks, white ball, and red destination object). To counteract this issue the maze is tilted slightly i.e. is not perfectly horizontal when the overhead camera takes the maze picture. This slight tilt helps to reduce the glare typically produced by overhead lighting.

Robustness

Despite many of the inaccuracies that can develop in the system (such as the calibration going out of phase) the system is quite robust in that after each move of the ball the entire course is recalculated in order to know the next move to make. Due to these errors that happen due to lighting or even the ball not rolling correctly (as seen in the above video) the end goal is still accomplished as any move will change the dynamics of the system. As long as the system performs more correct than wrong analysis the goal should be achieved.

Failsafe

In some cases the ball may accidentally get stuck in a strange location between the Lego blocks. In this case the system detects this situation to see if the ball coordinate is different than the last time. If the ball has not moved the VBScript module will command both the pan and tilt servos to move with a larger than normal value in an attempt to "free" the stuck ball. Note that when this happens the goal direction is ignored until the ball is found to be free and moving again. This process, while rare, helps to ensure the system has failsafe.

Pipeline

The following is the processing pipeline as seen within RoboRealm to accomplish the marble maze solution.

```
if status = 0 then
  ..Affine
  ..Marker [Corrected]
  ..Flood Fill 131
  ..Blob_Separate
  ..Circles
  ..Blob_Colorize
  ..Blob Filter
  ..Set ball_x = [blobs:0], ball_y = [blobs:1]
  ..Marker [Restore]
  ..RGBFilter Red
  ..Blob Filter
  ..Set goal_x = [blobs:0], goal_y = [blobs:1]
  ..Marker [Goal]
  ..RGBFilter Blue
  ..Blob Size 0 threshold, 30 area, max 1024 objects
  ..Scale 16 16
  ..Threshold 95-255
  ..Colorize
  ..Crop 1,1 - 15,15
end if
VBScript Program
Scale 224 224
if new_image = 1 then
  ..Marker [Map]
end if
Parallax Control
Marker [restore]
Crop 16,16 - 240,240
Math Current Map
Display_Line
```

Notes:

1. You can enable the Parallax controller module if you have the Parallax servo controller. (The robofile has that module disabled on load) If not you will need to replace that module with your appropriate controller. This configured servo controller module assumes that you have two servos attached to the controller to move the tilt and pan of a platform like the Marble Maze.
2. The speed of the system is currently constrained by the time it takes for the marble ball to roll from one side of the maze to the other. Currently the algorithm waits a set amount of time regardless of how long the ball actually needs to roll. An improvement would be to monitor the image and right the maze back to horizontal once no more movement is detected, i.e. the ball has stopped rolling ... but we will leave that improvement to you!
3. The core of the path finding algorithm is in the VBScript module. The VBScript essentially processes the image as a 14x14 pixel array accessed directly in VBScript. Within this code there is a lot of additional code used to generate the final graphic seen above. Keep in mind while looking at this code that a lot of it is not necessary for the actual functioning of the algorithm. Only the very next move needs to be store and not the entire path. The entire path is recorded, however, in order to show graphically what route is currently being attempted.

Your turn ...

Want to try this yourself? Following is the robofile that should launch RoboRealm with the original image we used in this tutorial. You can click on each of the steps in the pipeline to review what the processing looks like up to that point. Note that if you disable the first conditional you can highlight each successive modules to see how the image processing is performed. The first conditional in the pipeline is to ensure that the system only updates the ball path when needed (i.e. the servos have completed their move) such that the system can easily run on a slower platform.

 [Download robofile](#)

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

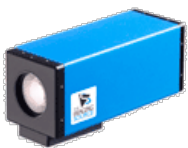
If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Shape Matching



The following tutorial describes one way to use a vision system to identify parts as they move along a conveyor belt. The purpose of the vision system is to identify the part and determine its location, orientation and size relative to a training model. The setup uses a [DBX 21BF04-Z](#) camera from [The Imaging Source](#) mounted vertically looking down at the belt. With the built in zoom the camera placement can be much higher than normal to allow the installation to view a larger or smaller field of view simply by changing the zoom over software.



The camera delivers 640x480 sized images at a maximum rate of 60fps using firewire. This is much faster and bigger than is required by this tutorial. Thus the image is scaled down to 320x240 for videos and images in this tutorial to better accommodate Internet bandwidth constraints. Note that while the DBX 21BF04-Z provides high contrast images and a motorized zoom (which are often needed in industrial settings) you can test the techniques described in this tutorial with a simple webcam.

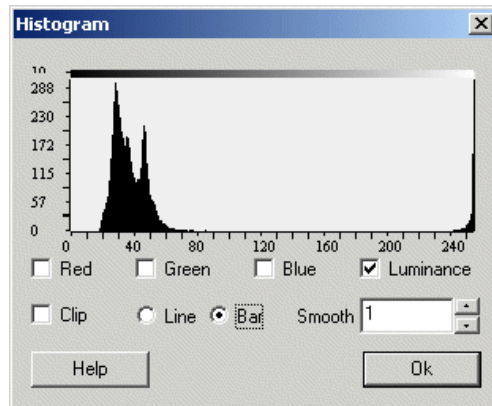
Let's first have a look at all the parts together as they appear to the camera.

Shape Matching



As you can see the camera has almost produced a binary or black and white image. This is desired as the purpose of this tutorial is to identify a part based on its shape. Having a black and white image will allow RoboRealm to know what pixels belong to the "part" and what pixels belong to the background (in this case our black conveyor belt).

A histogram shows the pixel intensity distribution of the above image. The graph shows the relative count of pixels of a certain intensity. In this case the intensity range is from 0 - 255 with a maximum pixel count of 320*240 or 76800. If you had a perfectly white image there would be a single line shown at the 255 (furthest right) place with a value of 76800. In most real images this will never happen and instead you will see a distribution of pixels throughout the histogram.

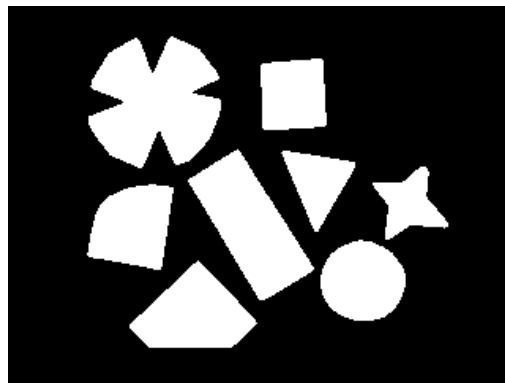


If you use the Histogram module to view the pixel intensity you will notice that the intensity distribution is almost binary but will still require some processing prior to shape matching. Matching shapes requires a perfect two color black and white image. The [DBX 21BF04-Z](#) makes this easy to achieve.

Image Thresholding

In order to create a binary image we use the [Threshold](#) module and set the threshold to a point at 190. Note that as the setting is manual your lighting for your parts should remain somewhat constant. Large changes in lighting will cause this segmentation to fail and parts of the background may be mistaken for foreground objects.

This is just one way of segmenting objects from the background. There are others such as the [Automatic Threshold](#) module or the [Flood Fill](#) module. These modules allow for the overall lighting to change in the image while still creating a reasonable thresholded image.



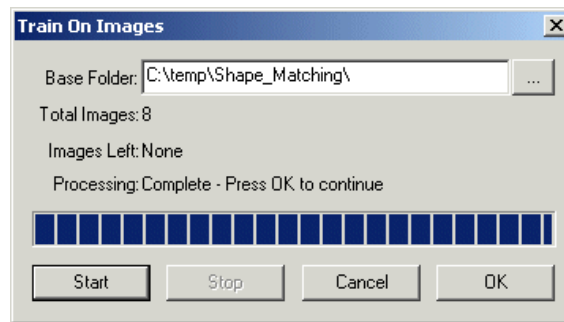
Note the difference in the image? The contrast is now 100% and the edges of the parts appear more jagged due to the thresholding, but this is what we want. The parts are all one color and ready for the shape matching module.

Before we add the Shape Matching module we need to create a database of files that contain the shapes we want to match. This is done by taking the above image and cutting the image using your favorite paint program and saving the individual shapes into each of their own files (.gif, png, ppm, etc.) within a single folder. This folder then contains 8 images one for each part. Below are the individual images used for training.



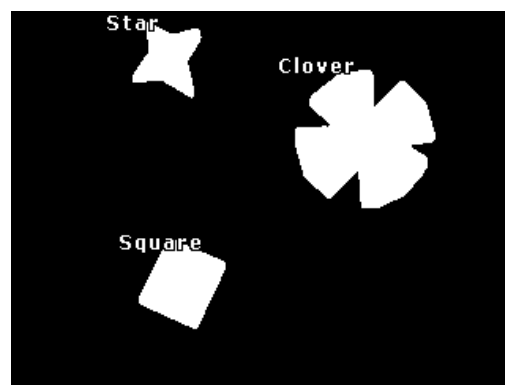
Note that the actual image sizes can be different and that only one object is shown in each image. Be sure to name the files with meaningful names as that will be the label used to identify the part.

Shape Matching



Insert the shape match into the processing pipeline and click on the train button. Specify the folder that you used earlier to save the database of images. Then click on start. RoboRealm will load in and learn the shapes in that folder. Once this is complete press Ok.

In the Shape Matching module click on the "Filename" checkbox. Now as you move your parts past the camera you will notice the filename of the part displayed in the main RoboRealm image preview alongside the object. Note that many variables are created to help you extract out needed information such as the parts Center of Gravity, Relative Size, etc. Refer to the [Shape Matching](#) module for further detail on those variables.



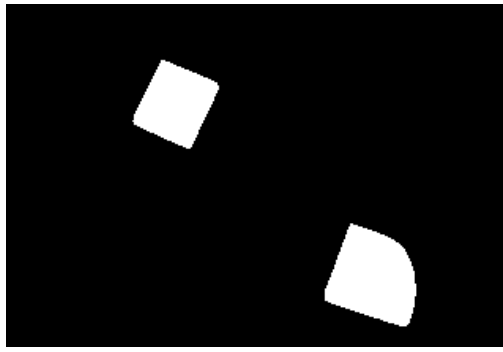
We have made good progress but there are still several issues we need to overcome. In the next pages we will review issues that we noticed while creating this processing loop.

Border Issues

As objects come into and out of the view of the camera their shape will not be correct. During that time the objects may be mistaken for something else.

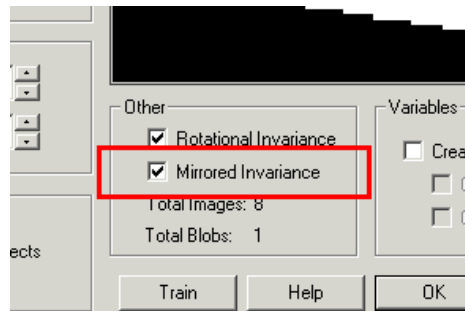


To prevent this from creating false object detections we need to eliminate objects that are touching the border of the image. We can do this using the [Blob Filter](#) module that allows us to only select/preserve objects that avoid the border. The attribute "Location->Avoids Border" which eliminates any objects that are touching the border is added to the Blob Filter criteria. We also use this module to remove any objects smaller than 100 pixels which will help to reduce any noise objects caused by flickering lights or reflections from other moving objects.



Mirrored Parts

We also noticed that the triangle object was matching in the low 90% when moving past the camera. On further investigation we realized that the part we used in the database was a reflection of the actual part. This easily can happen when a flat object is placed upside-down without realizing so. While this would still match correctly we selected the "Mirrored Invariance" checkbox in the Shape Matching GUI dialog to better compensate for this issue.



Helpful Tips

1. What do the parts flicker sometimes?

When the identification fails the part will be deleted from the current image. Thus for moments at a time a part may disappear due to a mismatch.

2. Can parts have holes?

Yes, in fact any holes in an object help by contributing to the matching process. Having consistent holes in an object can make it more unique with regard to other objects.

3. I have a mismatched part. How do I remove it using the Shape Module?

If you are matching a part with very low confidence due to the part not actually being in the database use the Confidence filter in the [Shape Matching](#) module to remove parts that have low confidence. Any part that is below 80% is probably not a good match. The default zero setting allows any confidence level which may result in incorrect matching. This can be corrected by increasing the minimum confidence value allowed.

4. How do I improve recognition?

Try to make the parts as large in the camera view as possible. The more detail in the image of the part the better the identification process can discriminate between similar objects.

If you do consistently get incorrect matches try using the filter options in the shape matching module to reduce the search space. For example, you can turn off rotational invariance which will only match objects based on the orientation of the database template file. Or you can use the size percent to eliminate really large or off-scale matches. Remember, the module is trying to match shapes at any size which can sometimes cause it to create bizarre comparisons. Telling it what to ignore can help reduce this confusion.

Be sure your objects are well segmented. Note that you NEED a black and white image in order for the shape matching to work. Your parts need to be in white and your background in black. If they are not, use the [Negative](#) module to flip black parts into white.

Be sure that your outline of the object is relatively smooth. If it isn't try using the [Smooth Hull](#) module to soften the objects outline.

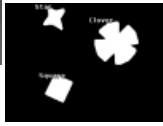
Be sure that your values of the object's relative size are correct by using the [SHOULDN'T](#) module to check the object's values.

Don't expect the orientation to work on symmetrical objects. A perfect circle has no orientation. Objects with one symmetrical axis such as your face can cause the orientation to flip 180 degrees at random. To get the best orientation your part will have to be asymmetrical in shape.

Videos



(862 KB) 22 seconds video of the original images moving over the conveyor.



(534 KB) 33 seconds video of the processed results of the parts (note that the video is slowed for viewing purposes).

Notice how the shapes will suddenly appear and disappear when approaching the border. This is the blob filter removing objects whose shape may be altered by the image borders which may cause mismatches.

Pipeline

The following is the processing pipeline as seen within RoboRealm to accomplish the part processing.

Firewire_Camera	00:000
Threshold 190-255	00:016
Blob Filter	00:000
Shape_Match C:\temp\Shape_Matching	00:031


Notes:

1. You will need to train the Shape_Match module on a folder that contains the database of images to be matched. You can do this by [downloading \(.zip\)](#) all the images into a single folder (like c:\temp\Shape_Match), edit the Shape_Match module, press the train button, change the path to your folder (e.g. c:\temp\Shape_Match) and press start.
2. You can view the final variables (like part orientation or COG) by adding in a Watch_Variables module or selecting the appropriate checkbox in the Shape Matching module.
3. You can add in the firewire camera module or other modules to capture images. Or if you have a webcam connected just press the "Camera" button for a live feed.

Your turn ...

Want to try this yourself? Following is the robofile that should launch RoboRealm with the original image we used in this tutorial. You can click on each of the steps in the pipeline to review what the processing looks like up to that point.

 [Download the Shape Matching](#) robofile.

To use the video downloaded from the above .zip file configure the media reader module in this  [robofile](#) to point to that video. Note that you can slow the video speed down in the media reader to get a slower frame by frame view.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

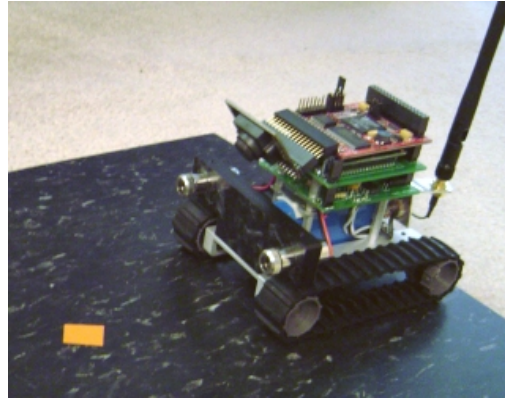
Have a nice day!

Surveyor SRV-1 Trail

The Surveyor SRV-1b is a great platform for experimenting in machine vision. With the wireless communication and built in camera the SRV-1b is a natural choice for machine vision processing using a PC.

In this tutorial we once again experiment with object tracking. In this case we are looking for orange squares that define a trail for the robot to follow. The trail of squares is created using orange electrical tape cut in small pieces and placed approximately 1.5 inches apart. Care needs to be taken to place each square near enough to the previous square otherwise the robot will not be able to see the next square as it loses track of the previous one. Once the robot determines that no additional squares are present it will turn around and proceed back over the course.

The camera angle of the SRV-1b is more or less parallel with the floor. In order for this tutorial to work correctly the camera needs to be tilted down such that more of the floor is in view. This hardware adjustment is necessary otherwise the squares appear more like orange lines due to the large perspective distortion in the default view. The tilting of the camera was accomplished by using a 32 pin connector that fits into the camera socket and bending the pins using a vice to the desired angle. While this is not the most elegant solution it certainly worked for our purposes.



First let us have a look at our sample trail.

Sample Trail



The image above is an overhead shot of the sample trail created using orange electrical tape placed on black tiles. From an overhead view the contrast is very good with the squares being nicely defined against the matt black tiles.



From the robot's point of view the setup looks quite different than from an overhead view. This is largely due to the proximity of the camera to the ground and the lack of adequate lighting to provide enough contrast in the image. The bad lighting of the setup was done on purpose in order to review image processing techniques that can help in these kinds of environments.

The robot view also reveals the pattern in the black tiles that includes white marks. These are less apparent in the overhead view but provide a high level of noise in the robot view.

You may also note that the squares in the image do not appear orange but instead appear more of a yellow color. This is due to color consistency problems which cause colors to appear incorrectly due to camera intrinsics, bad/low lighting and different illumination colors, i.e. colors appear different in sunlight versus florescent lights even if well lit.

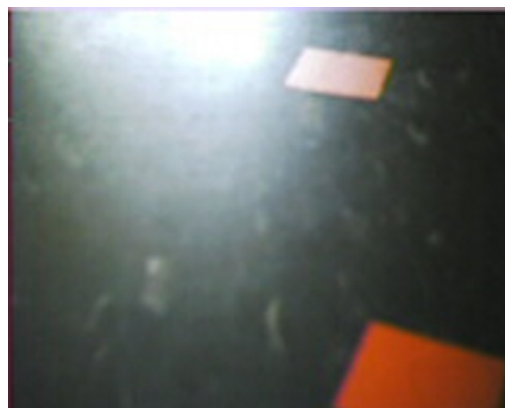
Let's have a look at a couple more images from the robot..

Robot View



It turns out that the previous robot image view was in fact one of the better images! Things just get worse from that! The above image shows a capture from the robot's point of view on its way back over the course. Our setup is placed not far from an outside window. With the strong sunlight (we're in sunny Los Angeles, CA) the glare from the outside causes significant reflection on the black tiles. Despite being black the tiles appear white when in the appropriate reflective angle with respect to the outside light. Again, this bad lighting was done on purpose in order to review possible solutions to this environmental difficulty when you cannot change the environment. If you can change the environment then is always recommended that you do so, i.e. close the curtains!

You can also see from the above image that the "orange" square also suffers the same affliction as the black tiles. The upper square is actually orange but in fact appears completely white in this view due to the reflected light.



The final robot image view is also taken from the reflected sunlight angle but in this case the lower square finally reveals its true orange color. Yet in the very same image the upper square again appears white.

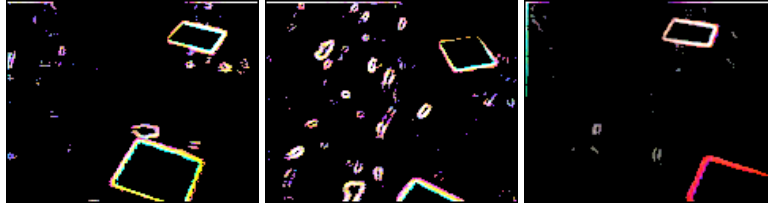
If you are new to image processing and machine vision you will by now start to gain an awareness of the importance of lighting in these scenarios. Lighting is a fundamental issue when working with robotic vision.

Given these three test images let's begin processing them in order to segment the squares from the rest of the image.

Lighting

We begin the orange square segmentation problem by working on the lighting issue. From the latter two robot images it is clear that the glare needs to be reduced.

A quick technique to resolve lighting issues is to reduce the image to edges. Most edge detection techniques use the local neighborhood of pixels in order to generate the edge strength. This local neighborhood has the advantage of reducing global lighting issues. This technique was used in one of our very first tutorials on [line following](#). The technique requires edge detection and thresholding followed by detection of the Center of Gravity to determine the robot direction.



We can clearly see that while this technique does have some promise the edges detected also include edges from the white spots embedded in the black tiles. If you are using a surface that does not have these noise elements then edge detection is a possible way to go.

In the case of our trail following robot there is a problem at the end of the course when the robot turns around. During the turn procedure the robot will momentarily see the tile edge against a lighter carpet. This boundary appears as a very strong edge which will cause the Center of Gravity measure to veer the robot off course.

Instead, we will try another light adjusting technique.

Lighting #2

We now proceed with a common lighting technique that will level the intensity of all pixels within an image. We will use one of the images to illustrate the process.



First, we convert the image to grayscale using the [Grayscale](#) module. This essentially focuses the image into its luminance or lighting channel. This is the channel that we want to even out within the image.



Next we really REALLY blur the grayscale image using the [Mean](#) module.





From this blurred image we subtract the original color image using the [Math](#) module.

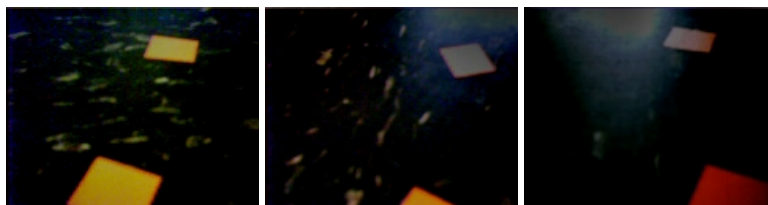


Which results in a much more even intensity across the entire image. This technique works because the grayscale conversion focuses on the intensity channel, the blurring relaxes the edges caused by lighting to effect large areas and the subtraction removes the global lighting changes to leave just the localized intensity changes which are typically associated with edges. This is in effect a form of an edge detector but one that better preserves the image colors than most edge detection techniques.

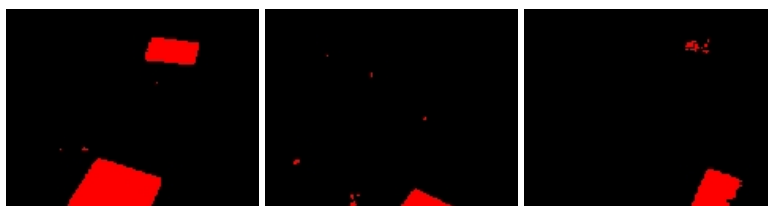
Now let's try detecting colors.

Colors

In order to detect the squares we need to identify them from within the image. Lets try using colors to detect them. Let us review what the three test images currently look like now with more even lighting.



Using the [RGB_Filter](#) module to search for red (yellow and orange both contain red) reveals some interesting but incorrect results.



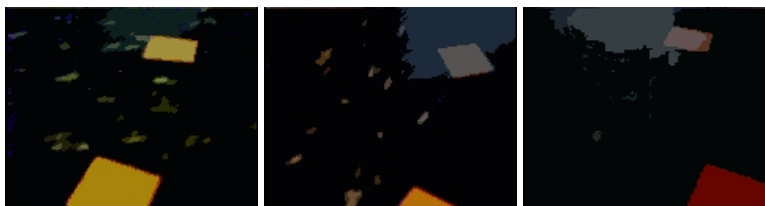
The first image seems fine but the last two mostly miss the upper square. If this happens during the course the robot will start turning around prematurely as it will think that the course has come to an end.

As in apparent from the above images using color to detect a gray object is not going to be very fruitful. In this case we now abandon color as an

identification feature and use it more as a natural pixel grouping feature.

Flood Fill

As we can no longer rely on color as a tracking feature we now turn to shape. To extract objects based on shape we first need to ensure that the object can be analyzed as an object. This means we need to group similar colored pixels with each other to define an object or a blob as known in machine vision terms.



Using the [Flood Fill](#) module (set to 65) we apply a color flattening technique to merge pixels into more meaningful groups. Flood filling is similar to the flood fill that is present in most paint programs. If a pixel is of similar color to its immediate neighbor the two pixels are replaced with the mean color of the two.

This works well to define objects but we still have a problem with image #3. If you look closely the upper square is almost a single color but has a large part of it on the right hand side in another color. This is caused by the tolerance of the flood fill not being high enough to include that part of the blob as a single object. However, increasing the tolerance causes other undesirable effects such as merging the square into the background glare.



Instead we use another feature of the Flood Fill to analyze neighboring blobs and how intense their borders are. The original edge detection tests showed that the outline of the squares were the only dominant edge in each square. The border between the parts of the upper square blob in image #3 above is then not very significant. Thus we use the Merge Borders interface in the flood fill to help merge blobs whose borders are not very strong.



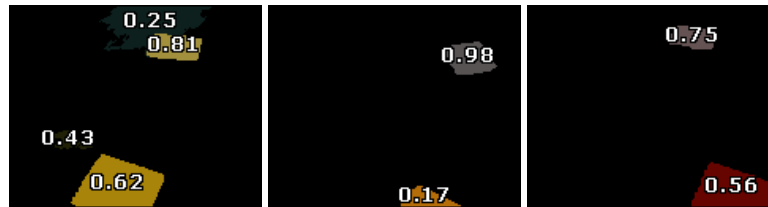
This also has the added benefit of merging the glare objects into a single background blob further decreasing the image noise. Now that we have all of our squares a single color we can start analyzing the shapes of the remaining blobs.

Blob Filter

To extract only the square shapes from the image we utilize the blob filter and the Quadrilateral Area to quantify the resemblance of the blob to a square. The Quadrilateral Area attribute analyses the sides and area of each of the blobs to calculate how close the blob area is to a quadrilateral defined by the 4 major sides of the blob. I.e. how well does the blob fit into the concept of squareness.

The blob filter allows you to add in descriptive attributes that rank each of the individual blobs according to the feature they represent. Thus, for a

perfect square we would expect a rank of 1.0 whereas for blobs that are less square we would expect a <1.0 weight.



Reviewing our test images shows that a cutoff value of around 0.65 would segment all our actual squares from the rest of the detected blobs. The other detected blobs are artifacts from the tile's white spots and the sunlight glare that we reduced earlier on.

It is worth noting that the squares on the sides of the images may or may not be detected by the square shape as part of the square is cutoff from the image. This is not an issue as we also use the blob filter to remove any blobs that touch the border. The reasoning is that we know all squares will fit into the image view and anything on the sides will either come into view or if just going out of view. We also remove border objects as the robot when moving may cause previously seen blobs to once again be detected. This detection can cause the robot to oscillate back and forth as the object moves in and out of partial view of the camera.

Using a 0.65 threshold and adding the COG module ends our image processing process as we now have an approximate point (the COG) for the robot to follow. The red circle identifies the COG point.



Note that if more than one square is visible on the screen the COG will return an average point between the two squares which is desirable as it evens out the transition from one square to the next.

Now that we have the point to move towards we need to use that information to control the SRV-1b appropriately.

VBScript

The point we want the robot to move towards is contained in the COG_X and COG_Y variables. These variables are created by the Center of Gravity module.

To control the motors of the SRV-1b to move towards that point we need to just use the X coordinate and steer left if the point is to the left of the screen or right if the point is towards the right of the screen. Basically we want to control the robot to keep the X COG in the middle of the screen. This will then cause the robot to move along the defined trail and also attempts to keep as much of the significant part of the image (i.e. the squares) in view.

We use the [VBScript module](#) to create two motor variables that are then fed to the [Surveyor SRV-1b](#) module.

```
' determine the center of the screen
midx = GetVariable("IMAGE_WIDTH") / 2

' the amount of turning force or motor
' value difference that is used to
' turn the robot.
factor = 2.5

' robot speed
speed = 155

' even out the motor values as one side can
' be more powerful than the other
left_bias = 5
right_bias = 0

' get the COG_X that the COG module
' calculated
cogx = GetVariable("COG_X")

' if we see something then change direction
' accordingly
```

```

if GetVariable("COG_BOX_SIZE") > 10 then
    ' how far off the center screen is the
    ' robot and how forceful does that
    ' turn need to be
    spread = CInt((midx - cogx) / factor)

    ' set the motor values to be used in
    ' the robot control module
    SetVariable "left_motor", speed - spread
    SetVariable "right_motor", speed + spread
    SetVariable "turn_status", 0

else

    ' if we don't see anything worth following
    ' we need to think about starting to turn.

    ' We first continue straight for a little bit
    ' which ensures that the robot passes
    ' over the current square, we then turn
    ' to the right for a short bit to see if we
    ' missed a square outside the field of
    ' view and then we start doing the full
    ' turn. The variable turn_status keeps
    ' track of where we are during this
    ' sequence.

if GetVariable("turn_status") = "0" then
    ' have a little faith and keep going straight for a short time
    SetVariable "right_motor", speed
    SetVariable "left_motor", speed
    ' in 500 ms the turn_status will jump to the
    ' next step
    SetTimedVariable "turn_status", "2", 500
    SetVariable "turn_status", 1
elseif GetVariable("turn_status") = "2" then
    ' no blobs - turn one way for a little bit and then
    ' turn the other
    SetVariable "right_motor", speed+15 + right_bias
    SetVariable "left_motor", 255 - speed - 15 - left_bias
    ' in 1000 ms the turn_status will jump to the
    ' next step
    SetTimedVariable "turn_status", "4", 1000
    SetVariable "turn_status", 3
elseif GetVariable("turn_status") = "4" then
    SetVariable "left_motor", speed + 15 + left_bias
    SetVariable "right_motor", 255 - speed - 15 - right_bias
end if

end if

```

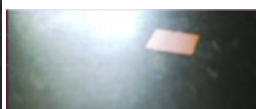
Surveyor SRV-1 Trail Video



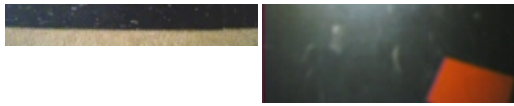
(1.5 MB) Video of the SRV-1b as it makes its way along the trail. You can see each of the major steps along the way as the image is processed for lighting, flood fill and blob filtering.



(1.7 MB) Overhead video of the robot moving along the trail.



(1.3 MB) Standalone video of what the robot sees while moving along the trail for your own testing purposes.




Problems

It is worth pointing out some of the issues and mistakes that we experienced during the creating of this tutorial to help you solve similar issues.

1. **Lighting** - To illustrate different lighting techniques we purposely made the setup environment non-optimal to run the robot. In reality one would probably attempt as much as possible to ensure that the lighting requirements are better met and no sunlight glare is present. This could also have been better mitigated by using different tiles that reflect less light.
2. **Blob-Filter** - Removing blobs based on attribute values is somewhat of a black art. As with any image processing applications noise is ever present in the image and can cause some of the squares to become non-square's momentarily. This causes them to disappear from the robots tracking. Better flood filling and blob filtering might be able to reduce this loss of tracking seen as object flicker in the final videos.
3. **Motion blur** - Capturing images while the robot is moving causes motion blur in the captured image. This motion blur tends to cause the squares to become more irregular in shape and therefore not match the blob filter criteria. To reduce the effects of motion blur the robot is pulsed in-between image captures to ensure that the robot is not moving while an image is being captured. This does slow down the robot movement considerably but is a necessary requirement for stable operation.
4. **Frame rate** - Despite the SRV-1b being capable of more than 10fps when in 160x128 mode the low/bad lighting further reduces the processed frame rate to less than 4fps. This reduction also requires that the robot move slower in order to capture enough frames to react quickly enough to steer the robot. The pulsing added to reduce motion blur provided enough speed reduction to ensure that the robot could react to the visual scene fast enough.
5. **Oscillation** - There are still times where the robot has successfully passed a square but during the turn (due to the robot being a tracked vehicle) the robot may back up enough to again see the most recently passed square. This causes the turn to terminate and the robot to reengage tracking. Once passed, the robot returns to turning mode where the process can repeat itself. This issue can be reduced by further increasing the time after which the robot determines it is no longer seeing a square and when to start a turn execution. Addition filters can also be added to eliminate detection of any square in the lower part of the image while turning with the assumption that any square that should now be tracked would appear somewhat distant from the robot.

Your turn ...

Want to try this yourself?  [Download the Surveyor SRV1b Trail](#) robofile that you can run yourself. Clicking on any of the modules will bring up that interface and allow you to customize it for yourself.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

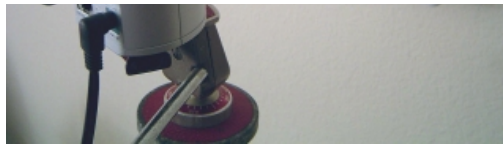
Have a nice day!

Digital Reader

This Digital Reader tutorial shows how to use a camera to read a digital device that may not have a computer interface. This is especially useful for older devices that do not provide an interface to modern computers but do have a human visual readout.

In this tutorial we use a regular air-condition sensor that one may find in any office or home. In the setup we use a DLink DCS900 camera mainly because it has an attachment that allows a large tripod to be connected to the camera and can transmit the image wirelessly to the base computer that is interpreting the image viewed.





First let us have a look at what the camera is seeing in order to begin our analysis ...

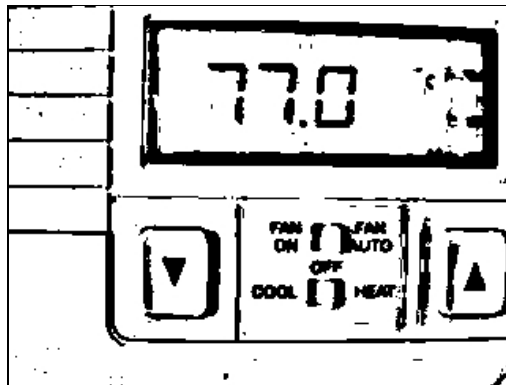
Camera View



On viewing the image it becomes immediately apparent that the lighting is not ideal in this situation. Even for humans the digital numbers (in this case 77.0) are hard to read.

The first task is to segment the digits from the rest of the scene. Note that there are many ways to accomplish this task and the specific modules you may use may differ from this approach.

To begin our segmentation we need to remove the lighting issue. To do this we use the [Adaptive_Threshold](#) module. Adding this module converts the above image to:



The parameters of the adaptive thresholding should be set to approximately the width of one of the digits in order to extract out the digit. We found 20 to be an appropriate window size.

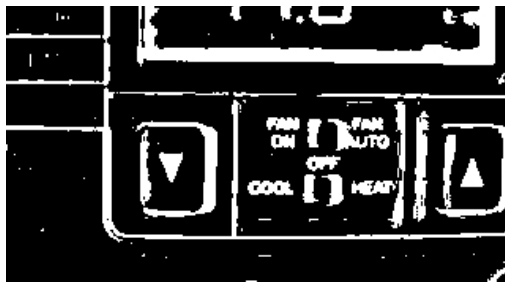
One can easily see that the adaptive threshold does a great job of eliminating any lighting issues and highlights the digits from the background.

But you can notice we've still got some cleanup to do

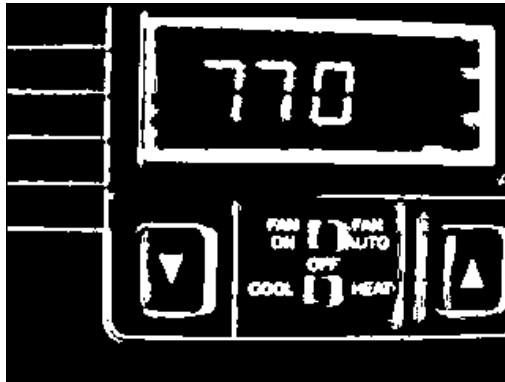
Image Cleanup

First off you will notice that within the image the digits are black. In order to process them correctly going forward we need to be manipulating white objects as RoboRealm considers white pixels as foreground and black ones as background. All the following modules use this assumption as the basis for their processing. To remedy this situation we simply invert the image. Using the [negative](#) module.

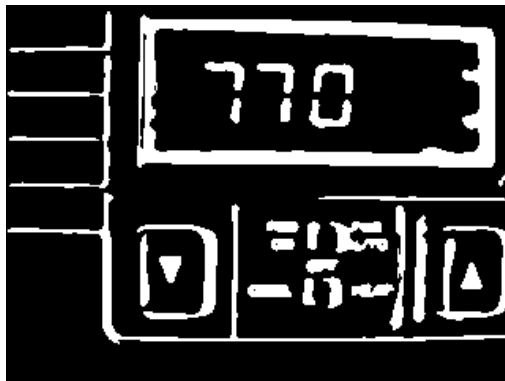




Now that the digits are in the correct color we can continue by removing all blobs (connected pixel groups) that are smaller than 70 pixels. The choice of 70 is largely arbitrary but it should be a number where most of the smaller insignificant objects are removed. The actual size will naturally depend on your situation.



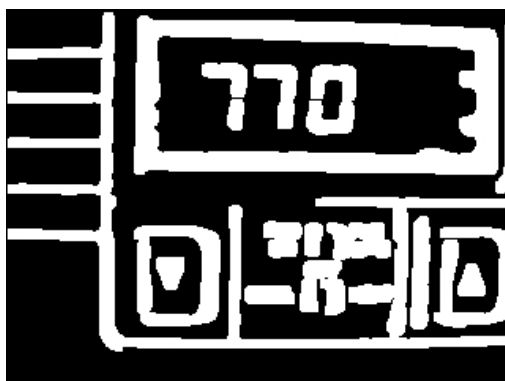
The digits are now appearing more clearly but due to the original low lighting that the image was taken in the border of the digits is not very smooth. To smooth things out a little we use the [Smooth Hull](#) module to round out the digits' shapes.



But you will notice that we still have a rather big issue. The digits themselves are broken into two blobs. In order to best match blobs we need to combine the digits into a single blob ...

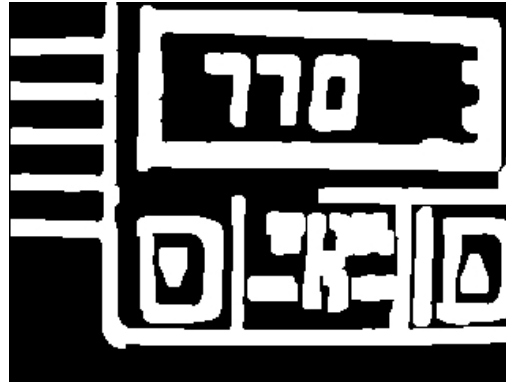
Merging Blobs

To merge the blobs we need to expand all white parts of the image until they touch with nearby parts. An easy way to do this is with a combination of [erode](#) and [dilate](#) modules. We first perform a 2 pixel dilation in all directions.

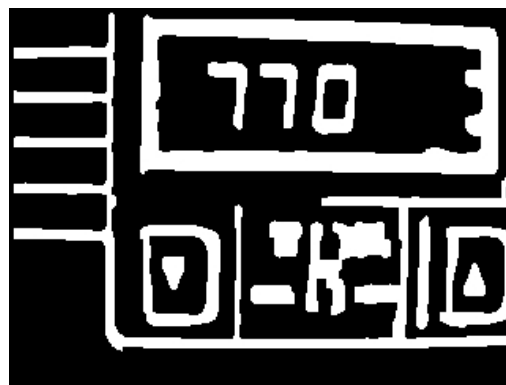


This helps to connect the digits but in some situations a thin line still separates the digits. If we continue to use the dilate module we risk the chance of digits being merged into each other. So instead of using the dilate module to dilate in all directions we simply specify the expansion to only

happen in the vertical direction.



Now that the digits have been merged we reduce them closer to their original size by using the erode module with a 2 pixel erosion.



This reduces the digits down to a more visually appealing size.

Once the digits have been segmented from the original image we now need to understand the digit properties as represented by blobs to help eliminate other objects in the image that are not digits. This helps to reduce the clutter in the final stages and provides a more stable result.

You can notice that the digits appear within a square rectangle. This rectangle was initially formed by the white casing in contrast to the lower light LCD part of the panel. We can use this characteristic to better focus on just the digits. To do this filtering we use the [Blob Filter](#) module with a single 'inner' parameter.



You will notice that while helpful there are still two triangles left in the image. We could use the blob filter to further filter based on triangles and invert that selection to remove them but instead we will use the final step to eliminate those objects.

Shape Matching

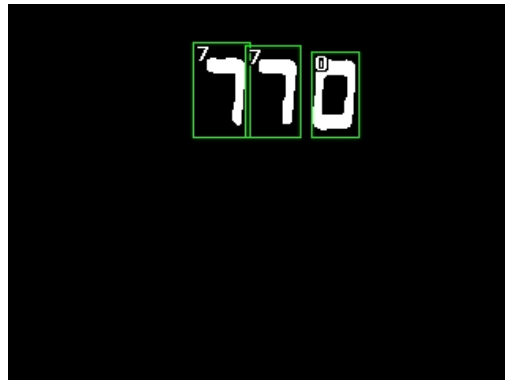
Shape Matching

To recognize which digit is which, we have to compare the resulting objects with a known database of objects. This known database is simply a folder that contains images of each of the separate shapes to be recognized. The folder we used contains the following images:



This database of images was generated by cutting the final image generated up to this point and manually separating each digit into its own file. This was done by pressing CTRL-C inside RoboRealm when the appropriate image was seen and pasting the image (CTRL-V) into our favorite paint program to then crop and save each image into its own file. Note that the image filenames were specified as 0.gif, 1.gif, 2.gif, etc. in order to make the image to numeric translation easy.

This database was then trained by the [Shape Matching](#) module which matches each resulting object to a single file in the database. The confidence and size filters within the Shape_Match module were then used to trim out any bad matches such as those against the remaining triangles.



Note that the digits displayed at the top of each green box is actually the filename of the image matched. Since we named the database images with the digit they represent it is easy to translate the image name into the actual number. In order to do the final grouping we need to use a little VBScript to combine the separate objects together to create the final number.

VBScript

Each recognized shape now represents a single digit. We need to use the shape information to create a single "temperature" variable that contains the final number. We do this by using the [VBScript module](#) to combine the information produced by the Shape_Match module in the SHAPE array.

```
' get a reference to the shapes array information
' that was created by the shape_match module
shapes = GetArrayVariable("SHAPES")
names = GetStrVariable("SHAPES_PATH")
```

```
' temp will hold the final temperature number
temp = ""
```

```
if isArray(shapes) then
' be sure we have found enough digits
if ubound(shapes) >= 26 then
' step through all objects and merge
' the number they represent
for i=0 to ubound(shapes)-1 step 9
```

```
    nstart1 = shapes(i+7)
    temp = temp & mid(names, nstart1+1, 1)
' we artificially put back in the period
' as we know where it should be
    if i = 9 then
        temp = temp & "."
    end if
```

```
next
```

```
' and finally create a variable that represents
' the text that we see. Note that this only
```

' gets set when enough digits are recognized

```
setvariable "temperature", temp
```

```
end if
```

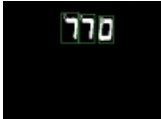
```
end if
```

On executing this script a new variable "temperature" is produced that contains the recognized temperature digits.

Videos



(362 KB) 10 second video of the original temperature images.



(688 KB) 10 second video of the processed results of the temperature images.

Notice how the shapes wiggle in the processed video but seem stable in the original? This is due to low lighting noise that causes the adaptive threshold (first module) to make different decisions as to what is part of a digit and what is not. As this decision affects the final shapes of the digits they will appear to wiggle when viewed as live video. You will also notice that at times the digits will not match or accidentally become merged with other digits that cause the matches to fail. Again this is due to lighting creating noise in the final results.

Pipeline

The following is the processing pipeline as seen within RoboRealm to accomplish the digital reading.


Adaptive_Threshold 20,3	00:0000
Negative	00:0000
Blob Size 0 threshold, 70 area, max 1024 objects	00:0004
Smooth Hull	00:0001
Dilate 2	00:0000
Dilate 2	00:0000
Erode 2	00:0000
Blob Filter	00:0007
Shape_Match c:\temp\digits\	00:0000
VBScript Program	00:0000

Notes:

1. You will need to train the Shape_Match module on a folder that contains the database of images to be matched. You can do this by copying all the images into a single folder, edit the Shape_Match module, press the train button, change the path to your folder and press start.
2. You can view the final "temperature" variable by adding in a Watch_Variables module or by editing the VBScript module and focusing on the second variable column.

Your turn ...

Want to try this yourself? Following is the robofile that should launch RoboRealm with the original image we used in this tutorial. You can click on each of the steps in the pipeline to review what the processing looks like up to that point.

 [Download the Digital Reader](#) robofile. Don't forget that you may need the folder of digits if you want to replicate exactly what we did. You can download that [here](#).

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

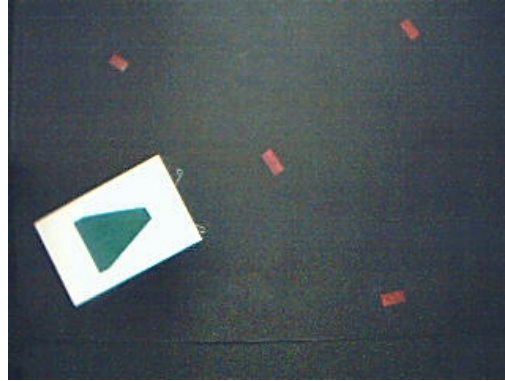
If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

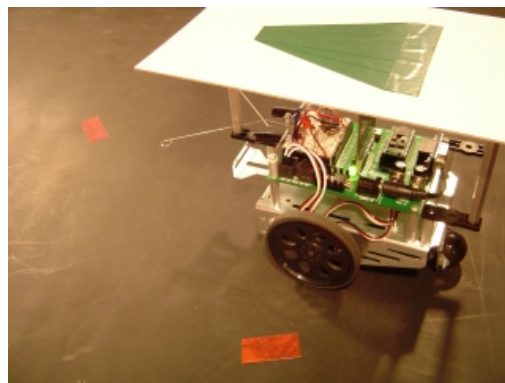
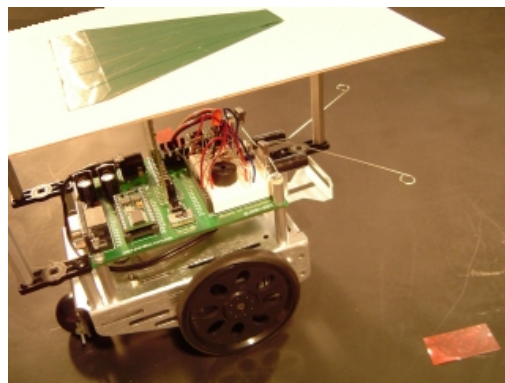
Path Planning

In this tutorial we are introducing the possibility of controlling a [Parallax Boe-Bot](#) robot using an overhead camera (possibly one mounted on the ceiling) to control the robot to move around an arena. It is assumed that the camera is stable, looking down onto the robot arena and does not move during the robot execution.

The following is what such a potential arena looks like from an overhead camera.



The small red marks are electrical tape stuck to a black metallic surface. The green arrow is a small piece of cardboard with green electrical tape attached in the form of an arrow. This cardboard is placed above a Parallax Boe-Bot in order to easily identify the robot from overhead.



The first step is to identify all the parts in the arena. This includes the waypoints and the robot plus its orientation. First we will start with detecting the Boe-Bot.

Detecting the Robot

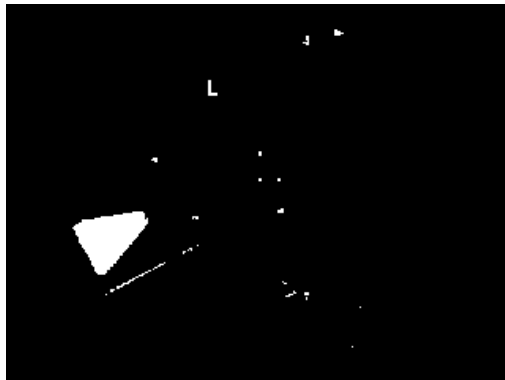




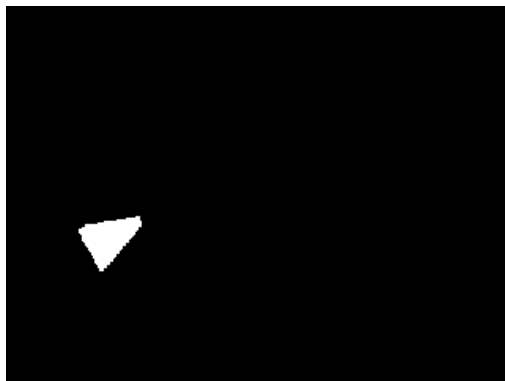
The above image shows the robot under the green triangle. To detect the triangle we use the RGBFilter module set to filter on green.



We can see that the triangle is there but it has some small holes in it. The Fill module with a parameter of zero will fill in any remaining holes.



To smooth out the triangle border and remove some of the additional noise we add the Erode module to produce a cleaner image. We also add in the Blob Filter and set it to eliminate all but the largest blob just in case some other green artifact remains after the erosion. This also provides us with a COG_X and COG_Y array with a single entry that contains the X, Y center of the triangle and is used to determine where the robot's position is.



That completes the green triangle (robot) extraction. Now we need to find out which direction the triangle is pointing so that we know which direction to turn the robot.

Geometric Analysis

Once the green triangle has been segmented from the background we add the Geometric module to run an analysis against the triangle. The value in particular that we are looking for is the ANGLE parameter which gives an indication of orientation of the blob.

Name	Value
AP_RATIO	5.0885
ANGLE	24.4440
AREA	748
BOTTOM_LEFT_X	59
BOTTOM_LEFT_Y	72
BOTTOM_RIGHT_X	61
BOTTOM_RIGHT_Y	72
BOX_ASPECT_RATIO	1.1765
BREATH	30.7084
CIRCULARITY	0.4350
CIRCULARITY_2	0.8398
CIRCULARITY_3	0.3750
COMPACTNESS	20.0001

Entire Image
 Individual Blobs

When viewing this parameter you will notice that it flickers ± 10 degrees or more. This can be reduced by adding in other blob smoothing modules such as Smooth_Hull but this additional precision is not needed for our purposes. Note that the angle is sensitive to the outline or border pixels of the blob. If you make the green triangle ends sharper the stability will also increase.

Once this number has been calculated we save the generated numbers (ANGLE, COG_X, COG_Y) to other variables so they don't get overwritten. This is accomplished using the Statements->Set Variable module. Once this module executes the variables are now called robot_orientation, robot_x and robot_y.

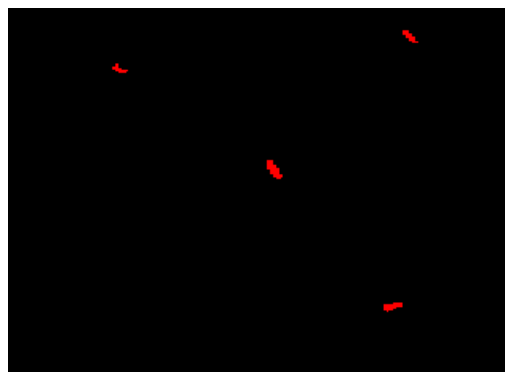
Using the Other->Marker module we revert the image back to the source in preparation for extracting the waypoints.

Waypoint Extraction

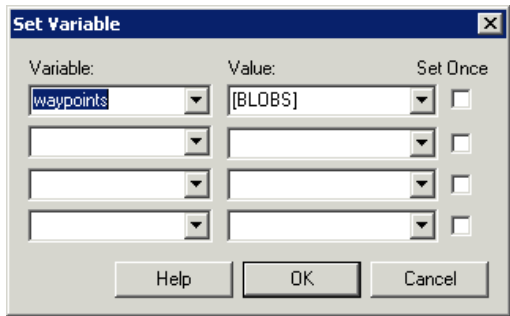
Using the Marker module the current image has now reverted back to the original source. We can once again apply the RGBFilter but this time for red.



To again remove the noise we add in an Erode and Blob Filter to reduce the number of blobs to the 4 largest.



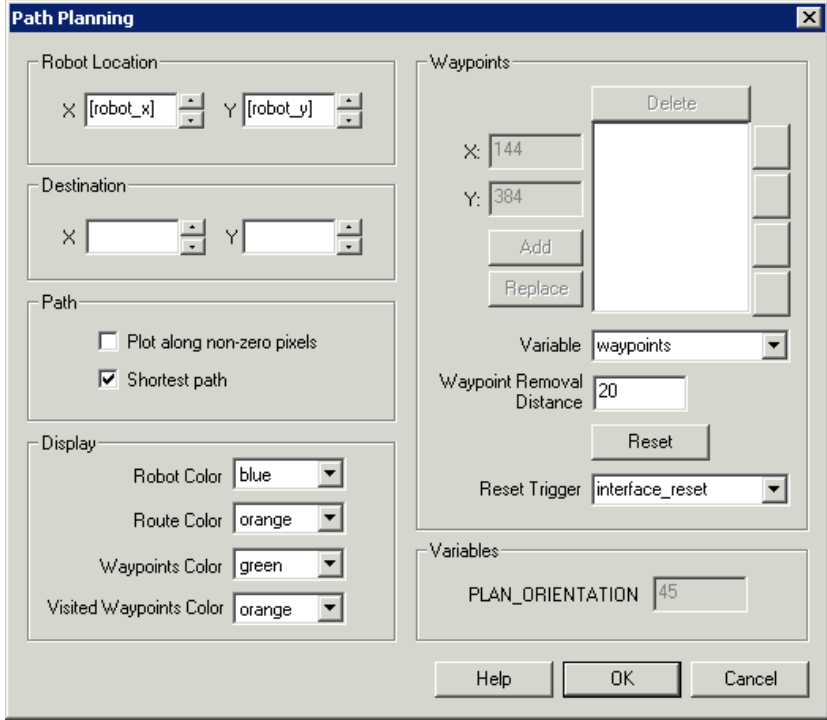
The Blob filter module again leaves us with an array of BLOBS with 4 sets of x, y coordinates. To keep these points from getting overwritten by other modules we use the Set Variable module to make a copy of them as "waypoints".



Now it is time to put all the parts together.

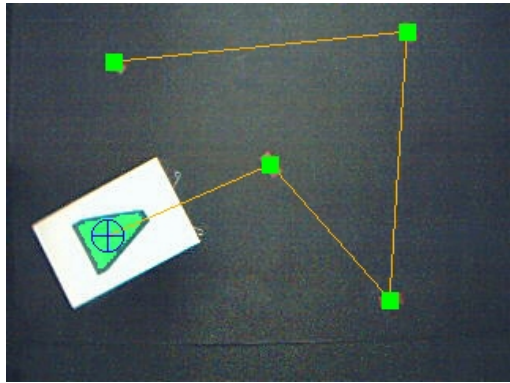
Path Planning

The Path Planning module will use the waypoints and the current robot position to plan a path through all the waypoints.



Note that the configuration for the Robot location uses the variables we saved back when detecting the robot position. Also note the use of the variable "waypoints" to automatically provide the waypoints based on a variable as apposed to the manual interface seen directly above.

The graphics created by the path planning module is then overlaid on the current image to view the final planned path.



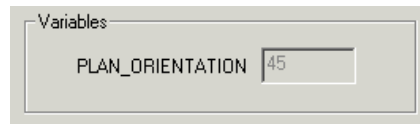
There remains one problem surrounding the waypoint generation. If we generate the waypoints on each new image frame the robot will obscure the waypoints as it nears them. This will cause a chaotic condition as the system will suddenly lose the waypoint and direct the robot to another. Once the robot moves to uncover the last waypoint it will again be directed to that waypoint causing an oscillation.

To remove this issue we only generate the waypoints once and use the reset waypoints trigger in the Path Planning module to regenerate the waypoint list. This resolution can be seen by the usage of the IF statement in the middle of the pipeline.

Now that we have the planned route we need to determine how and when to move the robot.

Robot Control

The Path Planning module generated a PLAN_ORIENTATION variable which tells us what direction the next waypoint is from the current robot position. We need to approximately orient the robot to match that direction.



Since we know the current robot orientation from the ANGLE variable which was renamed to robot_orientation we can compare that variable with the PLAN_ORIENTATION variable to determine how to turn the robot. Once the robot is in that general direction we can just move the robot forward and repeat the checking process. This control logic is performed using a script entered in the VBScript module.

```
' define some constants to use for speed. Note that the boe-bot
' moves forward with left and right being opposite numbers (due
' to the flipping of the servos as part of the robot
' construction)
left_speed = 115
right_speed = 105
rev_left_speed = 105
rev_right_speed = 115
stopped = 110
```

```
' this is the orientation produced by the path_planning and the
' orientation we want the robot to be at
desiredOrientation = GetVariable("plan_orientation")
```

```
' but first check to see if we are running or have completed
' all waypoints.
' When all waypoints have been visited the plan_orientation
' becomes -1.
```

```
if GetVariable("interface_run") <> "1" or _
desiredOrientation = "-1" then
```

```
SetVariable "left_motor", stopped
SetVariable "right_motor", stopped
```

```
else
```

```
' get the current robot orientation
robotOrientation = GetVariable("robot_orientation")
```

```
' reduce the precision of each of the orientations to
' 20 degree increments otherwise the robot (not being
' perfect in its movements) will spend all its time aligning
' to a degree that it cannot achieve.
```

```
robotOrientation = CInt((robotOrientation / 20) ) * 20
desiredOrientation = CInt((desiredOrientation / 20) ) * 20
```

```
' calculate the different between the two angles
diff = abs(desiredOrientation - robotOrientation)
```

```
' if they are the same (within 20 degrees) just
' move forward
if desiredOrientation = robotOrientation then
```

```
SetVariable "left_motor", left_speed
SetVariable "right_motor", right_speed
```

```
' otherwise turn in the appropriate direction. Note the use of
' 180 testing to determine which turn direction would
' be fastest.
```

```
' This allows the robot to turn in the most efficient direction
elseif desiredOrientation > robotOrientation and diff < 180 or _
desiredOrientation < robotOrientation and diff >= 180 then
```

```
SetVariable "left_motor", rev_left_speed
SetVariable "right_motor", right_speed
```

```
else
```

```
' if we don't turn one way then default to the other
SetVariable "left_motor", left_speed
SetVariable "right_motor", rev_right_speed
```

```
end if
```

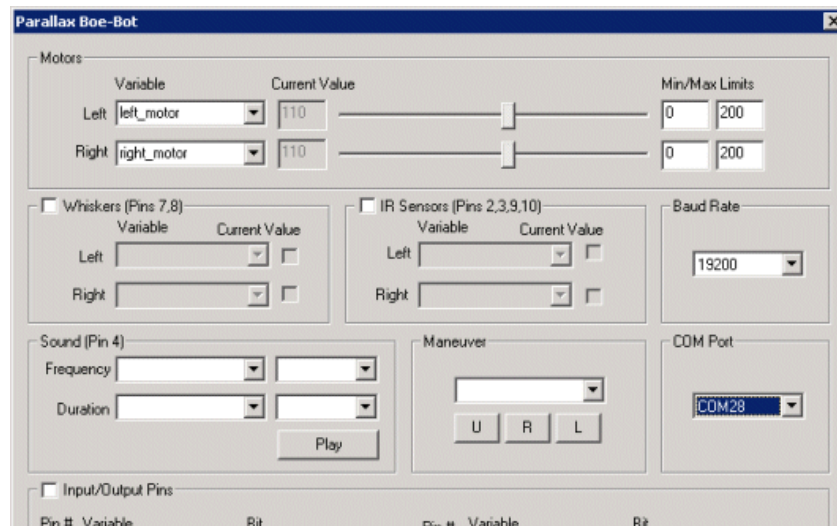
```
end if' interface_pause
```

At the end of this script the variables left_motor and right_motor contain the appropriate values to send to the Parallax Boe-Bot.

Parallax Boe-Bot

To send the left and right motor values to the Boe-Bot we use the Parallax_BoeBot module. Be sure to have downloaded the BS2 code into the Boe-Bot that supports the communication protocol used by this module (note that it is the same protocol as the MSRS interface). See the [Parallax Boe-Bot](#) module for more information.

By specifying the Left and Right motor variables we can now send the motor values to the Boe-Bot.



Now let's review the processing pipeline.

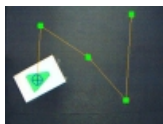
Processing Pipeline

Button_Interface	00:0000
RGBFilter Green	00:0004
Fill 0	00:0010
Erode 2	00:0004
Blob Filter	00:0005
Colorize	00:0000
Geometric_Statistics	00:0003
Set robot_orientation = [angle], robot_x = [blobs:0], robot_y =	00:0000
Marker [reset to source]	00:0001
if INTERFACE_RESET = 1 then	00:0000
..RGBFilter Red	00:0005
..Erode 2	00:0004
..Blob Filter	00:0006
..Set waypoints = [BLOBS]	00:0000
end if	00:0000
Math Current reset to source	00:0008
Path_Planning	00:0002
VBScript Program	00:0008
Parallax_BoeBot	00:0152

Notes:

1. The first module "Button_Interface" provides a simple Start and Stop button interface to easily control the robot and reset waypoints. This interface will popup immediately once this robofile is loaded.
2. The "Set robot_orientation = ..." is used to rename generated variables to other names so that they will not be overwritten by utilizing the same module again.
3. The IF statement controls when the waypoints are regenerated. This is linked to the 2nd Reset button in the Button_Interface module.
4. The Math module is used to merge back the green triangle over the original image. We noticed the triangle would sometimes disappear based on the lighting of the arena. This helped us keep an eye on how that segmentation was working as the robot moves.

Video




(2.3 MB) Video of the BoeBot moving through each of the waypoints as seen from the overhead camera.



(2.1 MB) Different starting location and 3 point light source.

Your turn ...

 [Download the Path Planning](#) robofile. Note that the Parallax module (last module) is disabled as you will need to configure that module for your COM port before enabling (to enable simply unpress the Disable button when that module is selected in the pipeline list). Also note that the red markers are only processed once when the reset button is pressed as it is assumed that the markers remain stationary during the robot operation.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

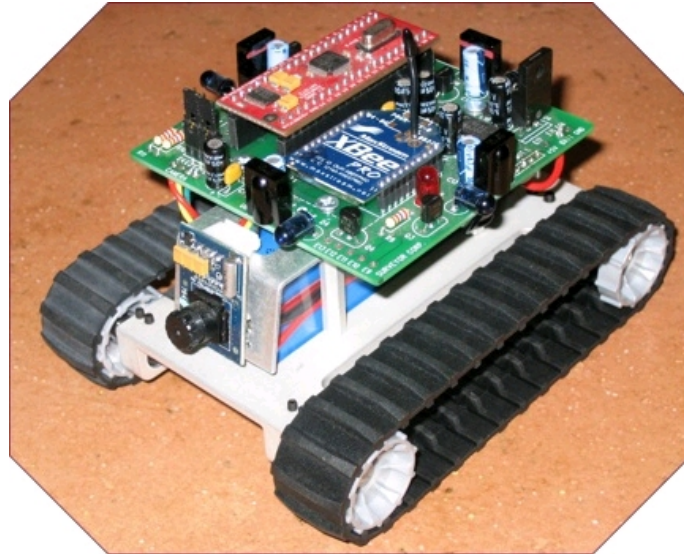
Have a nice day!

Surveyor SRV-1 Maze

The Surveyor SRV-1 is a great platform for experimenting in machine vision. With the wireless communication and built in camera the SRV-1 is a natural choice for machine vision processing using a PC.

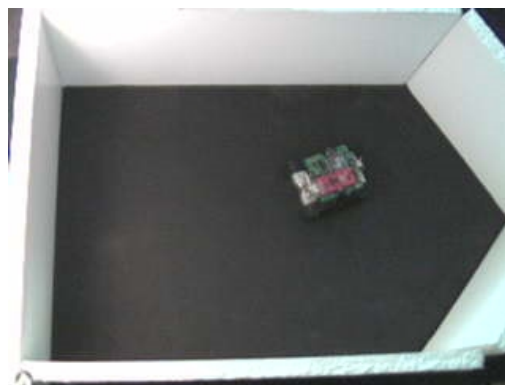
In this tutorial we experiment with the wall avoidance capabilities that can be performed using RoboRealm and the SRV-1. In particular we have a look at how the SRV-1 operates within a confined location surrounded by walls. The environment has been re-purposed from a Trinity FireFighting

Maze to show how the SRV-1 can operate within that environment.



First let us have a look at our simple environment ...

Example Room



Nothing special here. This is basically a pentagon type of room with a sharp corner in the far end of the room. The floor is a painted black metallic sheet while the walls are cut polystyrene insulform with magnets clued to the bottom to provide support. This environment is idea for testing robots as the walls will fall or bend with enough pressure and therefore do not harm uncontrolled robots.

The purpose is to have the SRV-1 roam around within this room while avoiding the walls. We first experimented with the built in "wander mode" of the SRV-1. This mode works fine in a spacious environment but simply keeps rotating on spot within this small room.

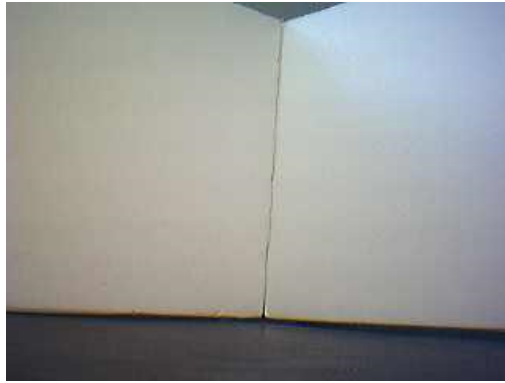
Part of the problem is that the camera on the SRV-1 is angled slightly up and therefore not much of the floor becomes visible. Due to the lack of visible floor we then decided to use the IR capabilities of the SRV-1 instead of vision.

The IR works nicely within its designed space requirements in a living room but are set too bright for our maze room. Turning on the IR and setting the robot within the room would completely saturate the entire room (all walls are bright white) with IR light and cause all readings to be at their maximum. Therefore IR was just not going to work either!

Back to vision ...

Camera View

If the robot is placed in one end of the room facing the other we get an image like

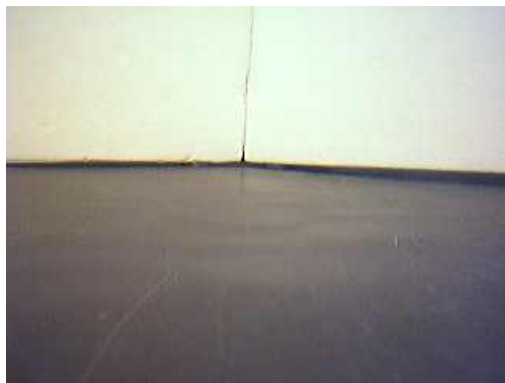


This view is from the robot looking towards the sharp corner as seen in the following overview image.



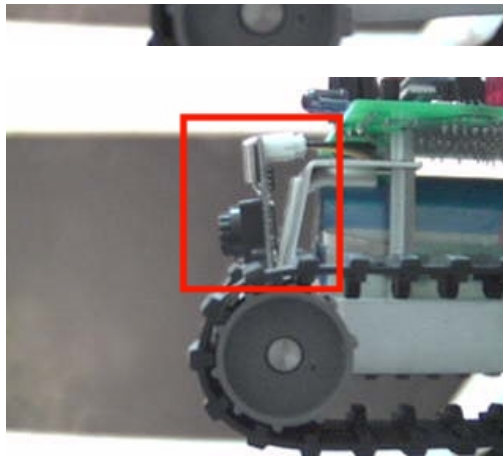
The robot view is a nice image but as the machine approaches the wall the black floor quickly disappears. To remedy this issue we need to tilt the camera down a little to see more floor. The SRV-1 camera is attached to the robot using double sided sticky tape. You can carefully un-stick part of the camera and simply prop it downwards to reveal more of the floor. This simple movement of the camera now allows us to see enough floor to be usable within our maze room.

Another possibility is to bend the aluminum bracket that holds the camera downwards a bit. There is enough room under the bracket to change the angle between 10-15 degrees which should be plenty. From the Surveyor folks themselves: "The only caveat is to make certain that there's no contact between the bracket and the front of the battery."



The robot is in the same position. Compare with first image above. Notice the floor space has substantially increased. This is the modification we used:





A subtle change but now that we can see the floor we can detect the boundary between the floor and the wall using vision.

Onto the Wall Detector ...

Wall Detection

The intensity difference between the floor and the wall is easily determined by the [Wall Finder](#) module. This module will place a red stripe where it seems a maximum change from the bottom of the image towards the top. This is not unlike the built in capabilities of the SRV-1 used in the wander mode.

Running this module on the image reveals



where we can nicely see the red line separating the floor and the wall. When this module runs it produces a couple of variables that help describe the line seen in the image.

```
AVERAGE_WALL_X 159
AVERAGE_WALL_Y 136
HIGHEST_WALL_X 150
HIGHEST_WALL_Y 145
LOWEST_WALL_X 301
LOWEST_WALL_Y 128
WALL_SLOPE -1.0
```

In particular we are interested in the LOWEST_WALL_Y and WALL_SLOPE variables. The LOWEST_WALL_Y indicates the lowest point of the detected boundary and effectively indicates how close the robot is to the wall. The WALL_SLOPE variable provides an estimate of the slope of the detected line and can be used to determine the angle of approach of the robot to the wall.

Using these variables it is now possible to create a script to navigate the robot around our room.

Onto some VBScript scripting ...

VBScripting the SRV-1

Using the variables provide by the [Wall Finder](#) module we can create the following VBScript that will map the wall variables to motor values that are then send to the SRV-1.

We would like to turn the robot away from the wall using the most efficient angle. Always turning in one direction would cause the robot to turn into the wall in many of the wall collision cases. In other words, if the robot approaches a wall like this



the robot should rotate to the left instead of the right. This can be accomplished by checking the sign of the WALL_SLOPE variable (-16 for the above image). If it is negative turn right, otherwise turn left.

However, there is an interesting problem using this technique when approaching a sharp angle corner (like the one at the tip of the room). The problem is that when the robot approaches the corner from one side it will use the WALL_SLOPE to turn the other direction. Once this is done the robot re-samples what it sees and determines that the WALL_SLOPE has now switched signs and therefore will move in the opposite direction to what it just did. Thus the robot is stuck in an oscillation between left and right movements.

To prevent this oscillation from happening we need to bias the rotational movement until we determine that it is safe to resample the WALL_SLOPE for a potentially new rotational direction. We do this by setting a "turn_bias" variable that once set will ensure the robot only rotates in that direction. This variable is then reset when the robot proceeds forward for a least three or more steps. This reset will only occur when the robot has decided that it is safe to move forward and can then resample the slope once again when approaching another wall.

The following VBScript is used within the [VBScript module](#)

```
' check if we are near a wall
if GetVariable("LOWEST_WALL_Y") < 10 then

    ' if we need to turn check the
    ' turn bias to make sure that
    ' if we have already turned that
    ' we continue to turn in the same
    ' direction
    turnBias = GetStrVariable("turn_bias")

    if turnBias = "right" then
        SetVariable "left_motor", 95
        SetVariable "right_motor", 165
    elseif turnBias = "left" then
        SetVariable "left_motor", 165
        SetVariable "right_motor", 95
    ' otherwise check the wall slope to determine
    ' the best direction to turn
    elseif GetVariable("WALL_SLOPE") < 0.0 then
        SetVariable "left_motor", 95
        SetVariable "right_motor", 165
        ' don't forget to bias successive
        ' turns in the same direction
        SetVariable "turn_bias", "right"
    else
        SetVariable "left_motor", 165
        SetVariable "right_motor", 95
        SetVariable "turn_bias", "left"
    end if

else

    ' increment the forward counter
    forwardCount = GetVariable("forward_count")
    SetVariable "forward_count", forwardCount+1
    ' if we have moved forward for 3+ steps reset
    ' the turning bias
```



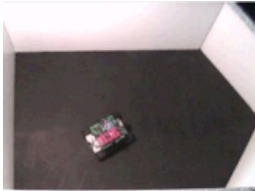
```
if forwardCount > 2 then
  SetVariable "turn_bias", ""
end if

' move the robot forward
SetVariable "left_motor", 150
SetVariable "right_motor", 150

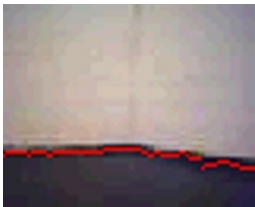
end if
```

Lets see it in action!

Surveyor SRV-1 Room Video



(3.6 MB) Video of the SRV-1 from an overview perspective as it roams around the room.



(338 KB) Video of what the robot sees while navigating around the maze. It is very jumpy since the video stops streaming during the motor moves which are very noticeable when turning.

Problems

It is worth pointing out some of the failures that we experienced during the creating of this tutorial. As one can learn much from mistakes we chose to share some of the issues.

1. The video image size for processing is 80x64. While this is sufficient for wall avoidance a higher frame rate would be desired in order to increase the speed of the robot. We achieved about 4 fps while running the robot within the room. The wireless transmission does add in some delays which can causes the robot "blindspots" during a move. With increased wireless speed the robot would receive more frequent updates from the PC and thus could move in a more reliable and faster way.
2. There are still times that the robot can become stuck within the room. Since we are using a black painted metallic floor (to ensure the wall magnets stick) the robot has a tendency to slip when rotating. This slippage can cause the robot to unintentionally backup into a wall if it is close to it while turning.
3. We did not factor in any noise suppression for the wall detection. Adding a mean or gaussian filter would soften the image and provide a less noisy line detection. This would allow the robot to more confidently approach a wall without turning too soon.

Your turn ...

Want to try this yourself? [Download the Surveyor Maze](#) robofile to run the example yourself. Clicking on any of the modules will bring up that module's interface and allow you to customize it for your needs.

Conclusion

We are extremely happy with the Surveyor SRV-1. It is a nicely built platform that is just fun to work with largely due to its small size. The battery lasts longer than most of the other vision platforms we have experimented with. The small size and weight allows us to experiment with it in confined environments so setup is quicker and less costly. The form factor also allows us to check motor and sensor values while actually holding the robot so a raised testing mount is not needed. Please see the Surveyor Website for more information on how to purchase this little wonder!

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

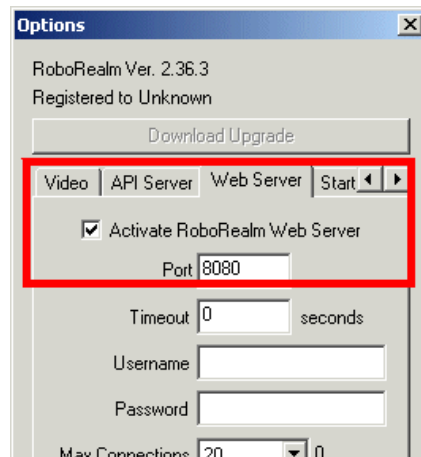
If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Internet Camera

The following tutorial illustrates how to setup RoboRealm to control a moveable camera over the Internet using your web-browser.

The first thing you need to do is to turn on the RoboRealm webservice. You do this by clicking on the options button in the main RoboRealm dialog and selecting the checkbox that appears in that dialog.



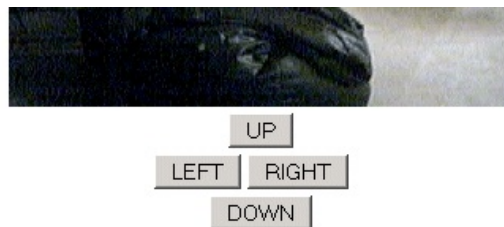
Once you have clicked on that checkbox you can now start streaming video over the Internet. To test your current setup access the following Url in your browser on the machine running RoboRealm.

`http://localhost:8080/index.html`

And you should see the same image in your browser that you see within the RoboRealm interface. Note that you will need Java enabled to see the image. If your IE browser does not support Java try [FireFox](#).

Be SURE that the Max Connections in the interface is larger than 1 otherwise only one person will be able to connect to the webservice at a time. When using the pan/tilt buttons you will need at least 2 connections. One for the video and the other for the move commands so be sure at least 2 connections are allowed.

You will notice a couple of buttons below the image. These buttons can be used to set a variable within RoboRealm. In this case we will use the 'move' variable to signal the type of move we want to accomplish.



The RoboRealm webservice supports getting and setting of variables using the url. The format for setting a variable is as such:

`http://localhost:8080/set_variables?my_var1=test&my_var2=test2`

and can be used to set multiple variables at once. When this url is executed the variables my_var1 and my_var2 will be created in RoboRealm with the appropriate values "test" and "test2". In our case with the buttons we set the variable "move" to 1, 2, 4, etc. based on the [Logitech Orbit](#) camera.

You can modify the Javascript routine contained in the index.html page to set and get any variable within RoboRealm. See the index.html page within the folder where you installed RoboRealm to change that source page.

The problem with setting a variable using the web interface is timing. Once set a variable will stay set and therefore cause the camera to keep moving. What we need is for the variable to become zero after a small amount of time.

This is accomplished using the following VBScript used within the VBScript module.


```
move = GetVariable("move")
if move then
  if GetVariable("LAST_CHECK") = "0" then
    SetVariable "LAST_CHECK", Timer()
  else
    if Timer() - GetVariable("LAST_CHECK") > 2 then
```

```
if timer() - GetVariable( LAST_CHECK ) < 2 then
  SetVariable "move", 0
  SetVariable "LAST_CHECK", 0
end if
end if
```

end if

This script will ensure that the variable move does not contain a non-zero value for more than about a second. In other words, after a second the variable "move" will be set to zero.

Now we lastly need to put all this together to create a robo-file with the appropriate configuration.

To try this out for yourself  [download](#) the robo-file and execute it using RoboRealm. Turn on your webserver and point your browser at the above url. You should now be able to control your Orbit (or any other of the many devices that RoboRealm supports) from within your web-browser.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

USB Missile Launcher

The following tutorial illustrates how to translate values from your PC Joystick to control a USB Missile Launcher and fire at objects that you see from its point of view.

If you are impatient you can skip to the last page of this tutorial and [watch the videos!](#)

The USB Missile Launcher used in this tutorial is sold by [Dream Cheeky](#) and is available from several retailers like [Geeks.com](#) for around \$30.00. We used an inexpensive WebCam that you can get for around \$15.00 for the targeting video view. Held together by elastic bands (yes, very sophisticated construction here!) the system comes together for under \$50.00 with about 5 minutes of construction time ... and hours of fun!



The goal of this project was to use a PC based joystick to control the Launcher whilst looking at the video camera stream on the PC. This platform would allow for distance based firing without needing to be within the vicinity of the Launcher.

In order to connect the software pieces together we used three RoboRealm modules.

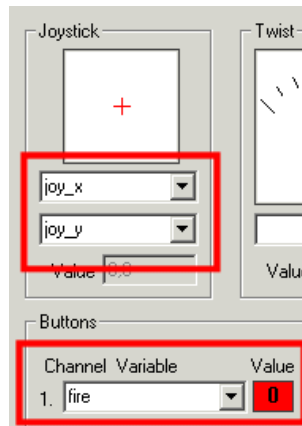
- [Joystick](#) - provides access to the joystick values
- [VBScript](#) - used to scale the joystick values to Launcher control values
- [DC_Missile](#) - provides the interface to the Dream Cheeky USB Missile Launcher

The first step is to map the joystick values to RoboRealm variables ...

Joystick Mapping

Below we see the RoboRealm Joystick mapping interface with the appropriate values mapped to RoboRealm variables. The interface shows all the

elements that can be mapped into RoboRealm but we are only interested in left, right, up, down and fire. If you move your Joystick around and press the fire and other buttons you should see the appropriate values and indicators change in the interface. Note that we are using a Logitech Joystick for this project.



As you can see from the image we have assigned the joystick X coordinates to a variable called "joy_x", Y to "joy_y" and the fire button to the variable "fire".

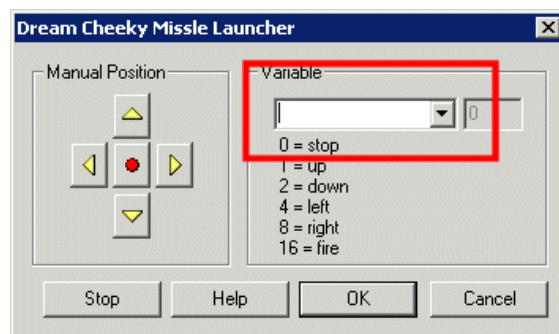
The X values of the joystick vary from -1000 to 1000 with -1000 being extreme left and 1000 being extreme right. Likewise with the Y coordinate but for up and down. The "fire" button will be a 0 (false) unless it is pressed in which case it becomes a 1 (true). The issue now is how to map those values into values that the USB Missile Launcher understands.

The Launcher has uses the following command values that are sent over USB to the device.

- 0 = stop
- 1 = up
- 2 = down
- 4 = left
- 8 = right
- 16 = fire

These are bit values meaning that if you want to move up and to the left you would send a 5 to the Launcher.

Below is the module interface for the Launcher. Note the one variable field that we can use to tell the Launcher module which variable will contain the value to send to the Launcher. For now we don't have such a variable since we have to create one using the values from the joystick. Note that you can immediately use the manual buttons to send commands to the Launcher to test that everything is connected and has power.



To provide this mapping of joystick domain to the launcher domain we use a bit of VBScript ...

VBScript Program

The VBScript module is used to run a small conversion program that creates a single variable from the values received from the PC Joystick. The following is the top part of the VBScript module interface showing part of the code within a text box (you can alternatively use the filename to specify a file on your filesystem).



```

Use Following Text

move=0

if GetVariable("joy_x") < -200 then
  move = 4
elseif GetVariable("joy_x") > 200 then
  move = 8
end if

if GetVariable("joy_y") > 200 then
  move = move OR 1
elseif GetVariable("joy_y") < -200 then
  move = move OR 2
end if

if GetVariable("fire") then
  move = move OR 16
end if

SetVariable "move", move

```

The VBScript program is as follows

```

' set the initial move to nothing
move=0

' if the joystick is to the right assign bit 2
if GetVariable("joy_x") < -200 then
  move = 4
' otherwise if it is to the left assign bit 3
elseif GetVariable("joy_x") > 200 then
  move = 8
end if

' now add in the y coordinate by ORing the
' value with the appropriate bits
if GetVariable("joy_y") > 200 then
  move = move OR 1
elseif GetVariable("joy_y") < -200 then
  move = move OR 2
end if

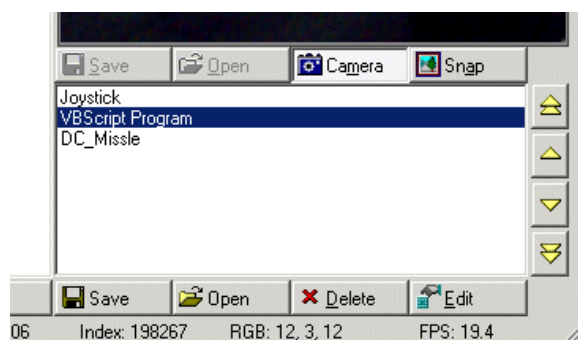
' and finally add in the fire button
' if it is pressed
if GetVariable("fire") then
  move = move OR 16
end if

' the results are in the single
' variable move. Assign that
' back into RoboRealm so we
' can use it in other modules
SetVariable "move", move

```

We finally go back to our DC_Missile module and assign the variable to "move" since that is the variable we created in the VBScript program that contains the correct bits that the Launcher understands.

The RoboRealm program pipeline finally looks like

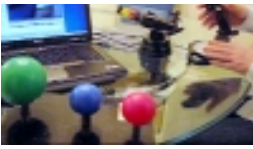


Given all these parts lets see what it looks like in action ...

Vision Based USB Missile Launcher



(3.5 MB) Video of the USB Missile launcher with attached camera being controller by a joystick.




The components of this tutorial were purposely kept cheap which translates into very noisy equipment and imprecise movements and firing ... but the system is nevertheless capable of performing the desired task.

Your turn ...

From this tutorial you should be able to create an automated Sentry gun that instead detects visual based objects and targets those objects instead of using manual control.

Want to try this yourself?  [Download the USB Missile Launcher](#) .robo file to run an example yourself.

If you don't have a Joystick  [try this one](#) which is configured to use your keyboard cursor keys to move the launcher. Press space for fire! Note that you need RoboRealm installed to run this file. Clicking on any of the modules within the RoboRealm pipeline will bring up that interface and allow you to customize it for yourself.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Lego NXT Ball Grabber

The following tutorial illustrates how a modified Lego Mindstorm NXT robot can use machine vision to find a blue ball, pick it up and move it over towards an orange cone.

If you are impatient you can skip to the last page of this tutorial and [watch the videos!](#)

The NXT robot is a modified TriBot with the touch sensor removed and a wireless camera attached in its place. The wireless camera transmits the image to a base station PC that processes the image using RoboRealm. Using bluetooth the appropriate motor commands are then transmitted back to the NXT. This combination of a PC and NXT provides a great degree of processing power and flexibility but does limit the operational range of the NXT to within a couple feet of the base station.



We can build a more powerful TriBot by using a wireless camera and receiver. The wireless camera was attached using lego parts and can be purchased for around \$40.00 from [Computer Geeks.com](#). The camera's receiver requires a NTSC digitizer that can be purchased for around \$50.00 from [Hauppauge](#). The digitizer connects to the PC via a USB connector. For around \$100.00 in COTS (Commercial Off The Shelf) parts you can provide your Lego NXT robot with machine vision capabilities.





Using this equipment the control loop is as follows: camera -> wireless transmission -> wireless receiver -> digitizer -> USB 2.0 -> PC -> RoboRealm -> Bluetooth -> NXT brick -> motor movements and then back to vision. This loop is someone longer than having an on-board vision processor BUT it does have a PC in the middle of the loop which is capable of a lot more processing than most onboard processors can provide.

In this tutorial we introduce a sequence that relies completely on vision without using any other sensors on the NXT (this is possible to do but that is for another tutorial). The sequence that we wish the TriBot to perform are:

1. Spin around in the current position looking for a blue ball
2. Approach the ball until sufficiently close enough to grab it
3. Grab the ball using the TriBot grippers
4. Spin around again looking for an orange cone
5. Approach the cone until sufficiently close
6. Drop the ball close to the cone

While this sequence is quite simplistic there are many problems that one encounters when using a vision system. So let's get started by understanding the camera setup first

Lego NXT Camera

Note that you can click on any of the below images to get a larger view.

As mentioned the camera used is a small wireless NTSC transmitter. The camera requires a 9 Volt battery as a power source. The one used in this tutorial was purchased from [Computer Geeks.com](http://ComputerGeeks.com) and cost around \$40.00.



The wireless camera connected to a 9 volt battery. The camera gets mounted right before the grippers with the 9 volt battery being stowed behind the NXT processor.



As you can see from this picture we used the parts designed to be the ball holder for the TriBot to instead contain the camera. The camera holder requires lego parts to slightly overlap the front and back to ensure that the camera does not slip out. Since the lego parts do not fit the dimensions of the camera exactly the sides of the camera holder are left to slide in along the bars so that the fixture can be compressed to hold the camera tightly. The rubbery parts that the ball holder uses act as a nice buffer between the camera and the lego parts and grip the camera nicely. The white parts in the back of the holder are used to fix it to the TriBot whereas the white parts in the front of the holder slip over the front of the camera but do not obstruct the camera view. (The photo displays the camera holder from the back) The longer gray parts in the back of the holder slip over the back of the camera to hold it in place.

The battery pack is strangely the perfect size for a 9 volt battery for a tight fit. The battery holder is very basic and held on behind the NXT by a single bar slipped through the last remaining hole behind the NXT.



Here we can see the camera being slipped into its fixture from behind. Note the rubber parts grasping the camera and the sides of the surrounding box being adjusted to miss the screws on the sides.



Now the camera is in its fixture and the lower back gray parts have been lowered to hold the camera in place.



This is the final assembly of the camera and battery ready for mounting on the TriBot. You can also see the camera holder from the front with the two small bars holding the camera in place. Note that they are much shorter than the gray bars in the back to ensure that the camera's view is not obstructed.



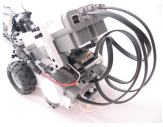
An image of the TriBot with the touch sensor missing. The camera will go right at the end of the shorter inner gray parts which have the small black connectors on the end.



The camera has now been mounted.



A shot from the other side of the camera mount.



The battery goes behind the NXT. Slide the gray holds off from the black bar and thread it behind the NXT through the battery pack and out the other side. Put back on the gray holds to ensure that the bar does not slip out during operation.

As you can see from the picture we used the parts designed to be the ball holder for the Tribot to contain the camera and mounted it inbetween but above the grippers. The position of the camera is very important as the current placement can see straight ahead which provides the furthest field of view. The camera can also still see the surrounding area even when a ball is within the grippers. Mounting the camera lower would allow for easier ball grabbing logic BUT once the ball is within the grippers you would lose any possibility of seeing where you are going!

On the computer side connect the wireless receiver to its power supply. Connect the RCA type connector from the receiver (the yellow plug) into the digitizer. Plug the digitizer USB into your PC, wait for the installation popup and then pop in the purchased Hauppauge installation CD that came with the digitizer.

Note that the camera receiver's tuner is very touch sensitive and care should be taken before any run to adjust the tuner to the best picture quality. Even the slightest bump will change the tuner and cause the image to become unusable.

Now that we have a camera attached we can start looking for a blue ball ...

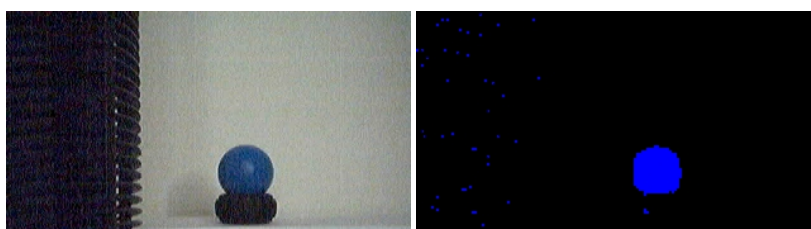
Searching for Blue

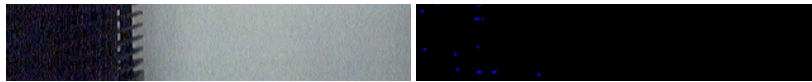
The first step in our tutorial is to spin in place while looking for a blue ball. To do this we added an [RGBFilter](#) for the color blue. This module is similar to the [Color Filter](#) but provides a quick way to detect primary colors. This module works nicely in that it can detect a blue ball.



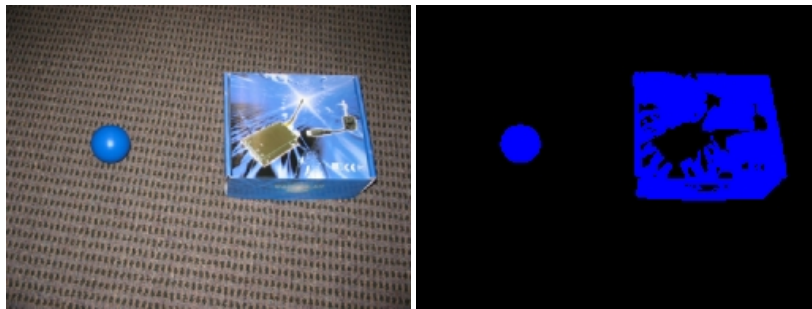
But in certain environments this is actually more problematic than one may expect as the color blue is often detected in shadow or dark areas (especially outside where shadows will contain blue areas due to the reflection of the sky). Blue color detected in shadows have a flickering type quality to them in that they do not reliably detect the color blue in the same pixel per each frame. We used the [Flicker](#) module to remove these areas of blue that are not stable.

For example, notice the blue areas detected in the following image. The [flicker](#) module helps to reduce this noise.





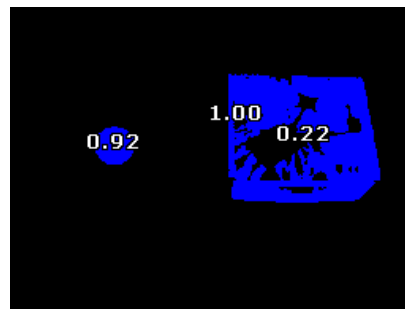
Additional problems arise when there are other blue colors seen while the robot is spinning around looking for the blue ball. For example, if you happened to have the camera box lying around an image like the following could occur.



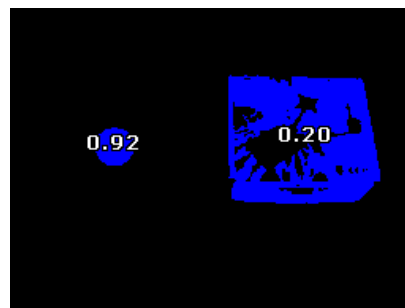
But we can quickly see that the shape of the box is not quite like the shape of the ball. To ensure that the NXT Tribot ignores the box we can use the blob filter to help us out.

Blob Filtering

From the previous slide we can see that the blue ball is very round while the detected camera box is not. By adding in the [Blob Filter](#) module and adding in a circular criteria above 0.65 we can eliminate all but the most circular blobs.



You will notice that 3 blobs have been detected. The 3rd blob is within the camera box and is a small circular blob not connected to the rest of the box. This often happens as objects with pictures will not always connect. To eliminate this false blob we could use the morphological [Fill](#) module to fill in any holes within objects. This would cause the inner blob to merge with the rest of the box and therefore become one object. We could also use the [Convex Hull](#) to create a solid object with the same convex hull as the detected blob. While these techniques are certainly possible we chose to add in another criteria within the current blob filter to ensure that all detected objects are 'outer' objects. This filter criteria specifies that if a blob is contained within another blob it should be eliminated from the current active blobs. Adding this criteria yields:

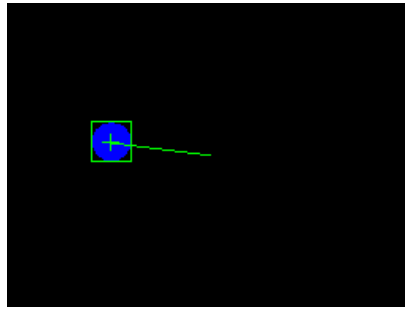


To ensure that we get the closest ball we also add in a 'bottom' criteria to the blob filter. This will ensure that a ball lower down on the screen will be given higher weights. Likewise, we also add in a 'largest' criteria to ensure that we go for the largest blue blob (this assumes that larger is closer). Finally we also add in a 'belowY 120' criteria to ignore anything above the ground plane in case the robot starts to track some guy in a blue shirt! Given all these criteria we begin to narrow the focus of the robot to tracking blue balls that are within a certain size on the floor.

Next we have to go and grab the ball.

Blob Grabbing

We are now able to somewhat reliably identify a blue ball. We can steer towards the blue ball by centering on its Center of Gravity and move in such a way to ensure that the COG of the ball stays in the center of the screen. We already have a tutorial about tracking the center of gravity at [Color Object Tracking](#) so we will not get into detail about that here.



We do, however, have a new problem that once we get really close to the ball we need to know when to grab it. The issue we have is that when we are ready to grab the ball the ball is now out of camera view!

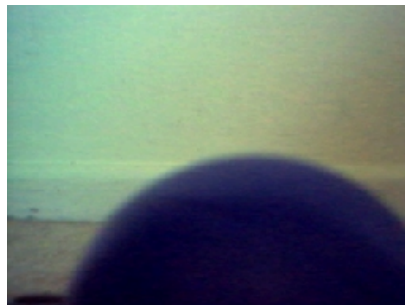
Knowing when to grab the ball requires that the robot 'know' without seeing the ball. We accomplish this by keeping track of the COG of the ball. When the ball becomes quite large within the image we know we are getting closer and begin a different kind of tracking. This tracking will notify us when the ball's COG suddenly disappears out of sight. At this point (assuming the ball was not suddenly picked up by someone) the ball should have just slipped by under the camera view and is now just in range of the grippers. Based on this assumption we can now close the grippers.

Below you can see this from the Robots perspective. The first image is when the ball is outside of the robots grippers. You can see the gripper ends on the left and right side of the image. The second image shows when the ball is within the grippers and almost out of view.



Interestingly enough, when the grippers close, the ball pops back into the camera view! While not done in this tutorial this is a nice effect that can be used to ensure that we actually grabbed the ball, i.e. if the grippers close and no COG of blue is found then we probably missed the ball. Best to backup and try again.

Below we can see what the image looks like when it had grabbed the ball. Notice that the top of the ball is once again in view.

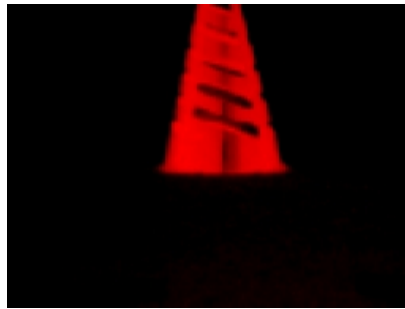


Once we have grabbed the ball we need to find our destination cone ...

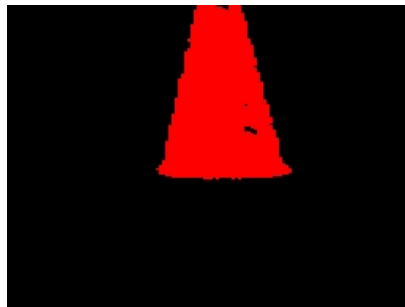
Finding a cone



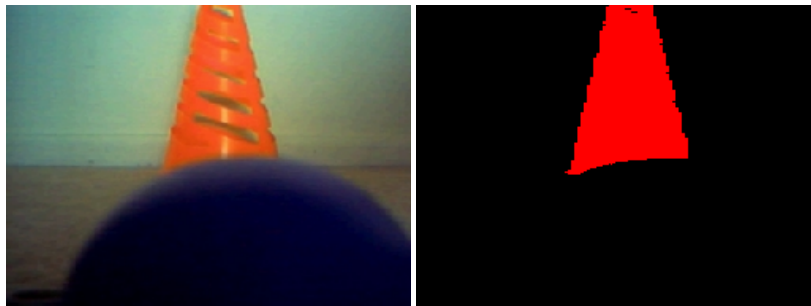
Our destination cone looks somewhat like a real traffic cone except that it is much smaller for the sake of our purposes but its color is still very distinct. We can now use the RGBFilter to look for something reddish as orange has red in it. The images here are taken from the wireless camera from the point of view of the Lego Tribot.



To eliminate other red objects above the ground plane we again use our blob filtering to remove small red objects and anything above the Y coordinate of 120. This results in a nice easy to track red object. We again use the same tracking and movement code as we did before to now move towards the orange cone.



Once enough of the cone is in view we assume that we are close enough and can drop the ball. It is worth noting that the ball is obstructing the view of the cone. In reality the processed image looks as below. Regardless, there is still enough of the cone in view to steer towards it even with the blue ball in the grippers.

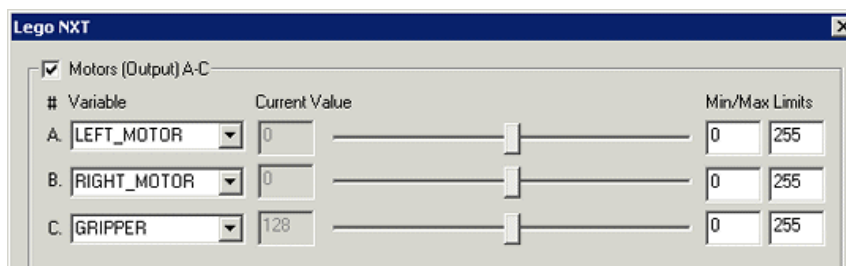


Now we know how to find the initial blue ball, when to grab the ball and are able to track the destination code all using vision. We need to communicate this information back to the Lego Mindstorm NXT to tell it what to do.

Talking to the NXT

To communicate with the NXT brick we use the RoboRealm Lego NXT module under the control category. This module provides us a way to use variable values to control or read aspects of the Lego NXT. The module uses serial communication over bluetooth to communicate with the Lego NXT. See the [documentation](#) for more information about the setup.

We will only be using the motor commands to steer the Lego NXT robot to its destination. While all the other sensors can be used this tutorial only focuses on the motors of the robot.



From the interface image you can see that we have 3 variables that are required in order to move the robot. Depending on the values of

LEFT_MOTOR and RIGHT_MOTOR the appropriate commands will be sent over bluetooth to the NXT in direct mode to command the left and right motor to move. Be sure that your left and right motors are plugged in the right way round otherwise your robot will move left when it should have moved right!

Note that the gripper on the Tribot is also a motor. BUT you don't want to leave that motor on for long otherwise you will stress the structure and potentially pop the grippers AND camera off from the robot (this was done numerous times while developing this tutorial). The gripper motor should only be activated for a short period of time to open the gripper but once the ball has been grabbed you will need to leave the motors on to ensure the ball stays in place.

The last part that we need for this project to work is to connect what the vision system produces with these motor commands. For that we need a bit of programming code ...

NXT Vision VBScript code

To translate the vision properties to motor command variables we need to add the VBScript module. This module will allow us to use VBScript programming to calculate the appropriate motor commands to send to the NXT and also know what mode we are in. There are 7 modes that we need to perform in this tutorial.

1. Open the grippers for 3 seconds. Then switch them off.
2. Spin around in place until you see a blue and big enough ball.
3. Move towards the ball and stop when it goes under the camera's field or view
4. Spin around looking for a reddish cone
5. Move towards the cone and stop when close enough
6. Release the ball by opening the grippers for 3 seconds
7. Cleanup ready for next round.

The steps are quite basic but you need to keep track of what step or state you are in as each entry into the VBScript module needs to reestablish what mode it is in. The VBScript module is not like calling a function which returns when it is finished as the VBScript module is only called once each new image has been processed and reactions to that new information is desired. You can think of this type of programming as a pipeline where you cannot stop the pipeline otherwise everything else will stop functioning. The best you can do is perform a check depending on what state you are in and what the motor and gripper variables should be in that state. Below we see the full annotated VBScript code used to maintain state during our task.

```
' the speed of the robot motors (a constant)
SPEED = 30

' grab what state we are in using the
' getvariable routine
state = GetVariable("state")

' jump to the appropriate state
select case state
case 0:
    SetVariable "Status: ", "Opening grippers"

    ' start the open gripper timer if it is not already
    ' initialized
    if GetVariable("GRABBER_OPEN") = "0" then
        SetVariable "GRABBER_OPEN", Timer()
    end if

    ' only execute this step for < 2 seconds
    if Timer() - GetVariable("GRABBER_OPEN") < 3 then
        ' set the grabber variable to open ... this is
        ' passed to the NXT as a motor command
        SetVariable "GRABBER", 180
    else
        ' set the grabber variable to neutral
        SetVariable "GRABBER", 128
        ' re-zero the time so we can use it again below
        SetVariable "GRABBER_OPEN", 0
        ' transition to the next state
        SetVariable "state", 1
    end if
```

case 1:

```
SetVariable "Status: ", "Searching for ball"
```

```
' grab the COG size that is created by the COG  
' module
```

```
size = GetVariable("COG_BOX_SIZE")
```

```
xw = (GetVariable("IMAGE_WIDTH")/2)
```

```
' if a big enough blob is seen then proceed to
```

```
' the next state
```

```
if size > 20 and GetVariable("COG_X") > xw - 60 _  
and GetVariable("COG_X") < xw + 60 then
```

```
SetVariable "MAX_SIZE", size
```

```
SetVariable "state", 2
```

```
else
```

```
' otherwise turn left until we see something big
```

```
' enough to track the left and right motor
```

```
' variables are passed to the NXT
```

```
SetVariable "LEFT_MOTOR", 128
```

```
SetVariable "RIGHT_MOTOR", 215
```

```
end if
```

case 2:

```
SetVariable "Status: ", "Approaching ball"
```

```
size = GetVariable("COG_BOX_SIZE")
```

```
maxSize = GetVariable("MAX_SIZE")
```

```
if size > maxSize then
```

```
SetVariable "MAX_SIZE", size
```

```
end if
```

```
' if we passed over the ball we are near enough to
```

```
' the ball to grab it this is determined by the COG
```

```
' going blank OR the ball suddenly got much smaller
```

```
' since we probably picked up a bit of blue noise
```

```
' once the ball when out of view
```

```
if (size < 20 or GetVariable("COG_Y") = "") _
```

```
and maxSize > 100 then
```

```
' stop the motors
```

```
SetVariable "LEFT_MOTOR", 128
```

```
SetVariable "RIGHT_MOTOR", 128
```

```
' turn on the grabber
```

```
SetVariable "GRABBER", 40
```

```
' continue to the next state
```

```
SetVariable "state", 3
```

```
else
```

```
' otherwise keep steering towards the ball's COG
```

```
' get the screen width and height (will always
```

```
' be 320 in our case)
```

```
xw = (GetVariable("IMAGE_WIDTH")/2)
```

```
' how far off center is the blob?
```

```
xMove = GetVariable("COG_X") - xw
```

```
xoff = xw - 5
```

```
' if the blob is off to the right then increase
```

```
' the right motor a little depending on how far
```

```
' off right we are
```

```
if xMove > 0 then
```

```
rightMove = 215 - SPEED + (((xoff - xMove - k) / xoff) * SPEED)
```

```
leftMove = 215
```

```

else

' otherwise up the left motor in a similar manner
leftMove = 215-SPEED+(((xoff-(-xMove)-k)/xoff)*SPEED)
rightMove = 215

end if

' set the variables so that the NXT module can use them
SetVariable "LEFT_MOTOR", leftMove
SetVariable "RIGHT_MOTOR", rightMove

end if

case 3:
' we've got a ball now head towards the red cone!!
SetVariable "Status: ", "Searching for cone"

' the processing pipeline has already switched the blob
' tracking to a red object since state >= 3. We just
' need to find it and head towards it.

size = GetVariable("COG_BOX_SIZE")
if size > 30 then
  SetVariable "state", 4
else
  ' turn left until we see something big enough to track
  SetVariable "LEFT_MOTOR", 128
  SetVariable "RIGHT_MOTOR", 215
end if
case 4:
SetVariable "Status: ", "Approaching cone"

' this routine is exactly like the one above
' except for the criteria that the blob cog
' size is 200

size = GetVariable("COG_BOX_SIZE")

' if we are near enough to a cone drop the ball
if size > 200 then
  ' stop the robot
  SetVariable "LEFT_MOTOR", 128
  SetVariable "RIGHT_MOTOR", 128
  SetVariable "state", 5
else

' otherwise keep steering towards the cone's COG
xw = (GetVariable("IMAGE_WIDTH")/2)

xMove=GetVariable("COG_X")-xw

xoff = xw - 5

if xMove > 0 then

  rightMove = 215-SPEED+(((xoff-xMove-k)/xoff)*SPEED)
  leftMove = 215

else

  leftMove = 215-SPEED+(((xoff-(-xMove)-k)/xoff)*SPEED)
  rightMove = 215

end if

```

```

SetVariable "LEFT_MOTOR", leftMove
SetVariable "RIGHT_MOTOR", rightMove

end if
case 5:

SetVariable "Status: ", "Opening grippers"

' initialize timer
if GetVariable("GRABBER_OPEN") = "0" then
  SetVariable "GRABBER_OPEN", Timer()
end if

' open grabber .. but only execute this step
' for < 3 seconds
if Timer() - GetVariable("GRABBER_OPEN") < 3 then
  SetVariable "GRABBER", 180
else
  SetVariable "GRABBER", 128
  SetVariable "GRABBER_OPEN", "0"
  SetVariable "state", 6
end if

case 6:
SetVariable "Status: ", "Mission Complete"

' clean up for next run and stop all robot motors
SetVariable "GRABBER", 128
SetVariable "GRABBER_OPEN", "0"
SetVariable "GRABBER_CLOSE", "0"
SetVariable "LEFT_MOTOR", 128
SetVariable "RIGHT_MOTOR", 128
end select

```

And lets have a look at the specific image processing pipeline ...

Ball Grabber Pipeline

The vision processing pipeline also has some basic logic in it to switch from looking for blue blobs to red blobs. The conditional in the pipeline uses the same state variable that is specified in the VBScript program. You can also see the ordering of the modules which is important since all the processing modules need to run before the VBScript module in order to provide the most current numbers. The Lego NXT control module needs to be after the VBScript program since the VBScript program is where the LEFT_MOTOR and RIGHT_MOTOR values are set. Note that the # comments are not displayed in RoboRealm.

```

# blur the image so we get better color detection

Mean filter 7

# detect blue blobs if we're looking for the blue ball

if state < 3 then

--RGBFilter Blue

  # reduce image flickering esp in darker shadows

--Flicker 3, 50%

  # filter the current blobs to yield the most ball like

--Blob Filter

end if

# check if we should be detecting the red cone

if state >= 3 then

--RGBFilter Red

  # and filter on something large

```

```

--Blob Filter
end if

# analyze the resulting single blob for location and size
Center of Gravity

# use the blob statistics to set motor variables
VBScript Program

# communicate those motor values to the NXT
Lego_NXT

```

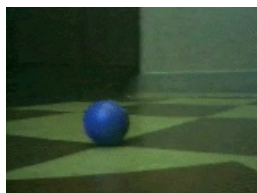
Note that you can change the filters to detect other colored objects or insert new modules to detect other aspects of objects.

Now lets see it in action!

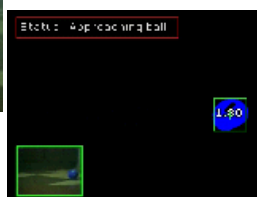
Lego NXT Vision Ball Grabber



(1.2 MB) Video of the Lego NXT from our perspective as it performs its task.



(1.1 MB) Video from the point of view of the Lego NXT as it sees the ball and cone. This is the original unprocessed video.



(1.4 MB) The processed video as seen from the Tribot's perspective. Notice the switch from blue ball to red cone tracking!

Problems

It is worth pointing out some of the failures that we experienced during the creating of this tutorial. As one can learn much from mistakes we chose to share some of the issues this current robot configuration has.

1. Perhaps the most difficult issue is keeping the robot together. Grabbing, bumping, and jerking can reduce the solidity of the Lego robot and it tends to start becoming loose which causes the camera mount to drop or raise slightly. This changes the field of view of the camera. Giving a quick squeeze to the camera mount once in a while ensured that the mounting of the gripper claw was stable and that the camera could receive the best field of view.
2. The wireless camera+receiver can be very noisy. In fact many trials were terminated due to the camera image suddenly becoming violently colorized with jagged lines everywhere (the kind of screen one sees in a highly noisy electrical environment). We will even go as far as creating a module around this issue to stop the robot when it detects this scenario. It only effects a couple frames at a time but often the ball is only seen for a couple frames and thus the robot can easily miss the detection.
3. Camera battery life is an issue. Once the battery becomes less powerful the image tends to become darker and darker. It is bad enough to have to deal with the specular lighting that is produced on shiny surfaces but we also had to deal with the camera light being reduced. Luckily using the flicker reduction and tweaking the threshold values the robot could still see the ball in an image where a human would have great difficulty in seeing it. In this case software was able to compensate.
4. The Lego NXT bluetooth is great for going wireless but we found that the delay in motor control can become too great for near real-time control. From the video you can see how the ball is almost overshoot but then corrects itself. This is in part due to the induced delays inherent in the system of which Bluetooth is a factor. Regardless, reducing speed is the best way to solve this issue.

Your turn ...

Want to try this yourself? [Download the Ball Grabber](#) .robo file to run an example yourself. Note that you need RoboRealm installed to run this file. Clicking on any of the modules will bring up that interface and allow you to customize it for yourself. Try getting the robot to track a green ball instead of a blue one ...

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Lego Mindstorms Orientation

The Lego Mindstorms RCX is a great platform for experimenting in machine vision. While the IR tower used to connect to ver. 2.0 and below is somewhat limiting in distance it is strong enough to provide an interesting platform for experimentation.

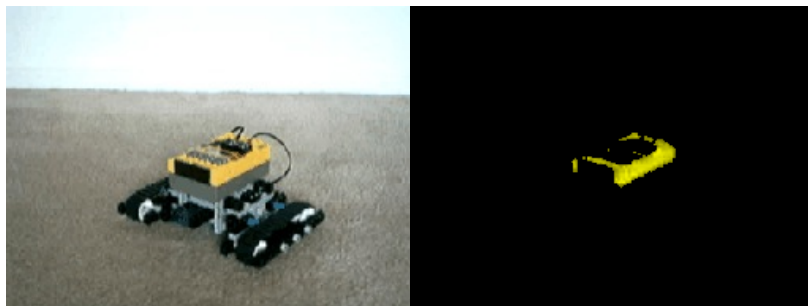
In any mobile robot project the orientation of the robot is important to know as motor commands need to be based on the current robot configuration. As an example of what is possible we set out to create a technique that will determine when the Lego Mindstorm is pointing directly toward the camera. The setup requires an external USB camera that is statically mounted in view of the Lego Mindstorm's RCX brick. We chose the tracked Roverbot as a simple example.



First we need to detect the Lego Roverbot...

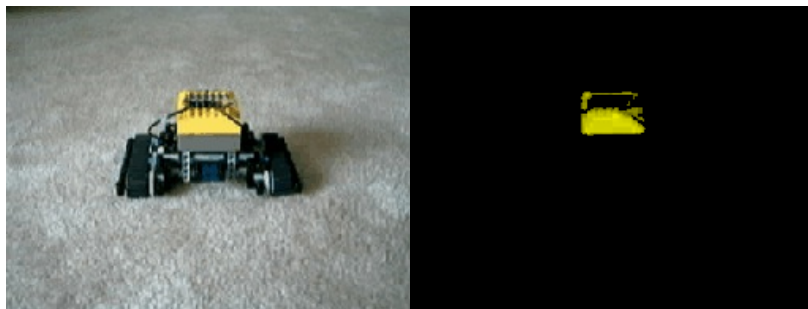
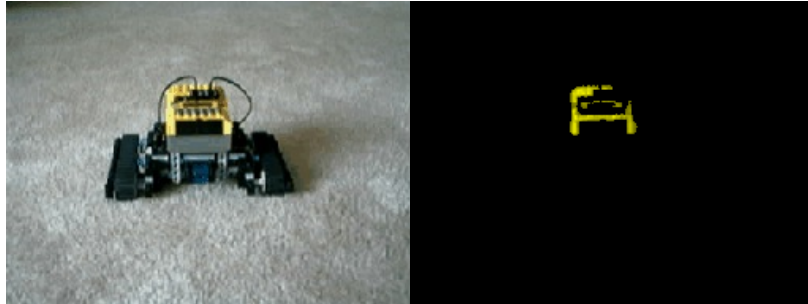
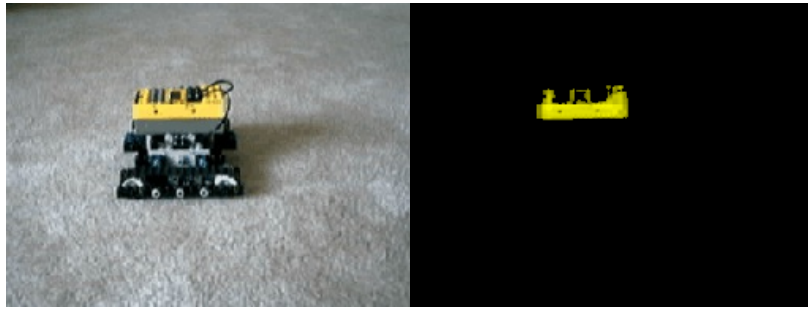
Detecting the Lego Roverbot

The yellow RCX brick of the Lego Roverbot is a nice property to it that makes it quite easy to detect. We can use the yellow [RGBFilter](#) to filter out all other colors but the brick color.



If we move the Roverbot around in terms of its orientation we notice that the yellow area widens and then narrows depending on what side of the RoverBot is visible to the camera. We are assuming the static camera viewing the Roverbot is tilted towards the ground at an angle where the sides of the Roverbot are the predominant yellow areas.

Seen below are a few more images from the Roverbot at different orientations.



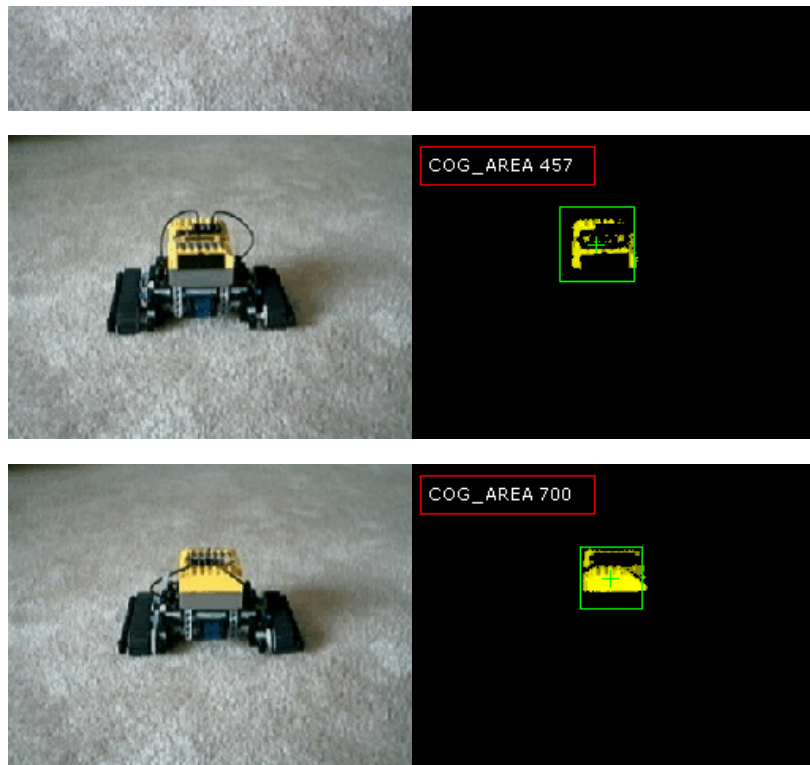
Next we need to detect the size of the yellow area.

Sizing the Lego Roverbot

As we have noticed the yellow area is largest when the Roverbot's sides are visible by the camera and smallest when viewed from the front or back. We can use the [COG](#) module to tell us more about the size of this yellow area. By selecting a bounding box that covers 90% of the detected yellow area we can be sure that we have a good relative approximation of the RCX brick. The COG_AREA will be smaller than when the RCX brick is pointed directly away or towards the camera and larger when the left or right sides are showing.

The following images show the bounding box overlaid on the detected yellow Roverbot.





But how do we know how small the COG_AREA should be when the Roverbot is orientated towards the camera?

Scanning with the Lego Roverbot

Given that the Roverbots position within the camera's field of view is unknown we cannot use absolute pixel size to determine which orientation the Roverbot is relative to the camera. We can only determine the smallest yellow blob size by scanning all sizes of the Roverbot. This scanning technique involves using a [VBScript](#) to spin the Roverbot for at least one revolution by setting the left motor to an opposite value to the right motor. While spinning the camera continues to detect the blob size and records the minimum pixels of that blob size. The minimum area will be the side that is facing us as the Roverbot front is where the IR receiver on the Roverbot is located. Since the back of the Roverbot is all yellow the front will contain fewer yellow pixels than the back.

Once the smallest yellow blob is detected we can continue spinning the Roverbot until we once again see this smallest yellow area (within a specified error margin). Once this is detected we can stop the spinning and the Roverbot will be oriented straight towards the camera.

The following VBScript program performs this scan.

```
' first determine what state we're in
select case GetVariable("state")

case "0"

' if we are still sampling then get
' the sample count
sample = GetVariable("sample")

' set the motors to rotate
SetVariable "LEFT_MOTOR", 3
SetVariable "RIGHT_MOTOR", -3

' get how many pixels exist in the
' current bounding box
size = GetVariable("COG_AREA")
min_size = GetVariable("min_size")
if min_size = 0 then
  SetVariable "min_size", size
else
  ' if it is the smallest we've seen
  ' then remember that
  if size < min_size then
    SetVariable "min_size", size
  end if
end if
```

```

end if

sample = sample + 1

' if we're done sampling
' change the state to the next
' step
if sample > 80 then
  SetVariable "state", "2"
  SetVariable "sample", 0
else
  SetVariable "sample", sample
end if

case "2"

' we're done sampling, now we
' need to keep rotating until
' we see the smallest side
' again
SetVariable "LEFT_MOTOR", 3
SetVariable "RIGHT_MOTOR", -3

size = GetVariable("COG_AREA")
min_size = GetVariable("min_size")

' if the current pixels within the
' bounding box is with 80% of the
' smallest recorded amount then
' assume that we have a match
if (min_size / size) > .8 then

  ' we have a match so stop
  ' the motors and jump to the
  ' next state
  SetVariable "LEFT_MOTOR", 0
  SetVariable "RIGHT_MOTOR", 0
  SetVariable "state", "3"

end if

case "3"

' we're done ... reset the min_size
' in case we run this again.
SetVariable "min_size", 10000

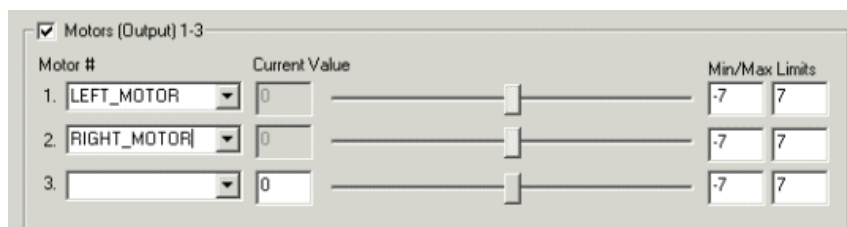
end select

```

Now that we have the script we need to map the left_motor and right_motor values to the Lego Mindstorm's RCX brick through the IR tower.

Lego Roverbot Motor Control

Mapping the RoboRealm variables created by the orientation.vbs script is done by using the [Lego Control](#) module. Specifically we are interested in mapping the two variables to the motor values of the Lego RCX system.



That should do it! Now lets see it in action ...

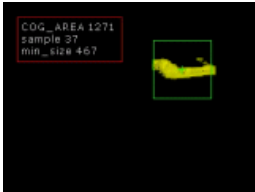
Lego Roverbot Orientation



(2.6 MB) Video of the Roverbot from our perspective as it spins and stops pointing towards the camera. The camera and IR tower are positioned on the right of the image facing towards the Roverbot. Notice as the RCX starts to move the two flashes in the front of the yellow brick. That is the IR transmission being picked up by the overhead camera. Many cameras can see IR light that is not visible to the naked eye.



(3.2 MB) Video of the Roverbot as seen from the tracking camera. This is the original unprocessed video.



(389 KB) The processed video as seen by the static tracking camera (i.e. processed version of video above). The yellow blob is the Roverbot. Notice how the green bounding box changes in size as the Roverbot spins.

Your turn ...

Want to try this yourself? [Download the Lego Orientation](#) .robo file that you can download and run yourself. Note that you need RoboRealm installed to run this file. You will also need the Phantom DLL (no longer available!) installed to communicate with the Lego RCX using the USB or Serial communication. Clicking on any of the modules will bring up that interface and allow you to customize it for yourself.

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

Why track colors?

Tracking objects based on color is one of the quickest and easiest methods for tracking an object from one image frame to the next. The speed of this technique makes it very attractive for near-realtime applications but due to its simplicity many issues exist that can cause the tracking to fail. This tutorial is about how you can use a colored object's size to distance the robot (i.e. our BucketBot robot) from that object. Note that RoboRealm is running ON the robot as it is equipped with Windows 2000 and a NTSC camera with a USB digitizer.

First, let's start by looking at an image which contains an object to be tracked.

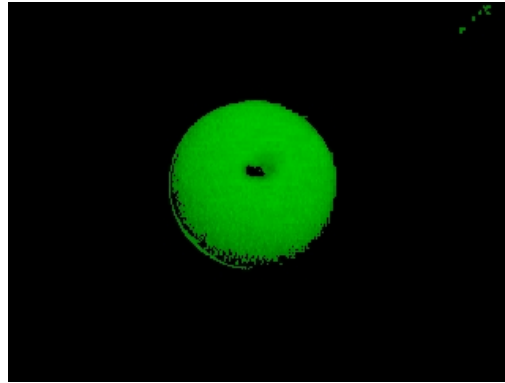


The sample image taken from BucketBot's camera contains a green ball. We'd like to move the robot away from the ball (i.e. backwards) when the ball approaches and move towards the ball (forward) when it moves away.

Detecting the green ball

We use the RGBFilter of RoboRealm to remove all objects in the image except the green ball. The RGBFilter is set to filter out green objects using

an RGB channel subtraction and then normalizes the remaining values. This will highlight green objects nicely.



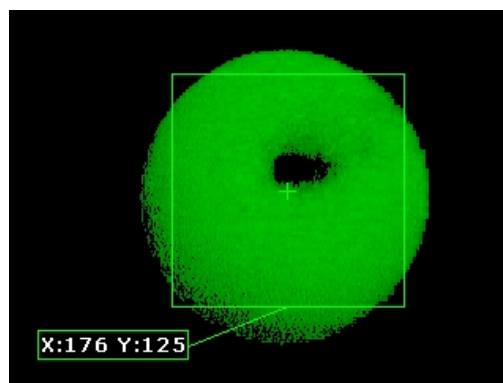
Now that the ball is segmented from the background we need to determine the objects relative size from image to image to determine if we need to move forward, backwards or remain stationary.

Ball Size

We don't really need the exact size of the ball since our movements will be based on any arbitrary measurement that is consistent from one frame to another. When that measurement decreases we should move forward, when it increases we should move backwards.

By adding in the [Center of Gravity](#) module we now have some RoboRealm variables that we can use for this task. We will chose COG_BOX_SIZE and configure the module to show a bounding box around 80% of the pixels. The COG_BOX_SIZE is the width OR height of the bounding box as the bounding box in this case is always ensured to be square.

Below we can see the Center of Gravity graphical overlay for two images. One closer than the other.

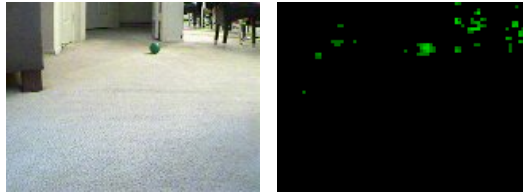


The COG_BOX_SIZE for image #1 is 148 while for image #2 it is 61. It is clear that we can use this number as a movement indicator to the BucketBot.

This works well but we have a problem when images get much further away.

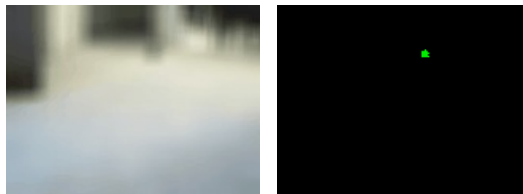
Green Noise

When the ball is far away from the robot the ball's image is very small and only turns on a couple pixels. With so few pixels the RGBFilter becomes more sensitive to background noise. This problem is further exacerbated when the image size is reduced to a small 160x120. Because of this additional noise the COG module starts to have issues centering on the actual ball and may chose not to target the ball at all. This problem can be seen in the following images:



You can see that the processed image using the green RGBFilter introduces a lot of noise. Looking closely at the original image we notice some spurious green pixels that are somewhat isolated in parts of the image that are not actually green but form the transition from one color to another. We think this noise is introduced by the NTSC camera to digitizer process on the BucketBot as we do not see those pixel transition problems on other web cameras.

Regardless, to reduce this error we introduce a [mean filter](#) of size 12 to the processing stream. Huh?? The mean filter causes the image to become blurred. This blurring will reduce the spikiness of the spurious pixels and allow the RGBFilter to better focus on the actual ball! This blurring trick does have some correlation to the human visual system. It is theorized that the [gaussian blur](#) is an active component of our visual system. While not quite a gaussian distribution the mean or box filter provides a similar effect at a much faster frame rate. And you thought you needed glasses to see those precise edges?



Now that we can better track the ball even at distance we now have to understand when and how to move.

Keeping the ball at a distance

Now that we have our COG_BOX_SIZE variable we can arbitrarily chose 85 as the size that BucketBot should try to maintain (this equates to a couple inches from the ball). Doing so will require forward and backwards movement based on the relative size of the green ball or COG_BOX_SIZE. We also need to steer the robot to keep the ball in the center of the screen. This involves rotating the robot from right to left and moving the camera from up to down.

To create these movement commands we'll add the [SSC](#) module (the BucketBot uses electronic speed controls compatible with PWM signals created by the SSC) to move the robot. Note that the SSC module communicates to the SSC board via serial commands.

The SSC module will send two variables to the SSC board that range from 0 to 255 with 0 meaning reverse, 128 neutral and 255 full forward. The BucketBot has two motors that control the left and right wheels independently. (This is nice since it can rotate in one spot.) There is one more servo that controls the tilt of the camera that also ranges from 0 to 255 with 220 meaning the camera is pointed about straight. Our next task is to translate the COG_BOX_SIZE variable into motor commands from 0 to 255 for each motor and for the tilt servo.

Doing this translation will require a bit of logic. So we add in the [VBScript Program](#) module into the current processing pipeline. The VBScript module allows us to insert this program

```
' get the current center of the image
midx = GetVariable("IMAGE_WIDTH") / 2
midy = GetVariable("IMAGE_HEIGHT") / 2

' set the movement neutral range, i.e. if the
' cog is with range of the center of the image
' then consider it ok and move straigh. This
' helps to prevent fine tune movements that
' can cause oscillations.
range = 5

' determine the factor that scales image width
' to a range of 0 to 128 .. increase beyond 128
' if your robot turns too quickly
horizFactor = 200 / midx
```

```

' vertical factor ... we don't have much
' range here so the number is larger than
' horizontal movement
vertFactor = 300 / midy

' ball size when it is too close
ballSize = 40

' initialize our start motor values to neutral
left_base = 128
right_base = 128

' get the size (width or height) of the current
' bounding box
size = GetVariable("COG_BOX_SIZE")

' if it is equal to "" then no object was detected
if size > 10 then

    ' get the horizontal center of gravity found by the COG
    ' module
    cogX = GetVariable("COG_X")
    ' if it is less than 75 the blob is on the left side
    ' of the screen so move that way
    if cogX < (midx - range) then
        left_base = 128 - (midx - cogX) / horizFactor
        right_base = 128
    ' otherwise move to the right if above 85 pixels (and no
    ' movement if 75 < cogX < 85 )
    elseif cogX > (midx + range) then
        left_base = 128
        right_base = 128 - (cogX - midx) / horizFactor
    end if

    ' if the ball is too close then we have to move backwards
    if size > ballSize then
        left_motor = left_base + ((size - ballSize) * 2)
        right_motor = right_base + ((size - ballSize) * 2)
    ' otherwise move forward
    else
        left_motor = left_base - ((ballSize - size) * 2)
        right_motor = right_base - ((ballSize - size) * 2)
    end if

    ' set the final motor values to be picked up by
    ' the SSC module
    SetVariable "LEFT_MOTOR", CInt(left_motor)
    SetVariable "RIGHT_MOTOR", CInt(right_motor)

    ' now lets work on the tilt ... grab the current
    ' tilt value
    tilt = GetVariable("TILT")
    ' if it was not set then default to 220 (in our case
    ' this is horizontal to the floor)
    if tilt = "" then
        tilt = 220
    end if

    ' if the blob is below 55 then tilt down up a bit more
    ' otherwise tilt up a little more
    cogY = GetVariable("COG_Y")
    if cogY < midy - range then
        tilt = tilt - (midy - cogY) / vertFactor
    elseif cogY > midy + range then
        tilt = tilt + (cogY - midy) / vertFactor
    end if

```



```
end if
```

```
' we don't want to look up more than horizontal to the  
' floor as we do not expect the  
' ball to be on the ceiling  
if tilt > 220 then  
  tilt = 220  
end if  
if tilt < 0 then  
  tilt = 0  
end if
```

```
' and set that value for the SSC module too  
SetVariable "TILT", tilt
```

```
else
```

```
SetVariable "LEFT_MOTOR", 128  
SetVariable "RIGHT_MOTOR", 128
```

```
end if
```

The program first grabs our size variable COG_BOX_SIZE and compares it with what size we'd like to maintain in view. If the object is too big we move backwards by a relative amount or if the object is smaller than we'd like we move forwards. However, we also need to change the relative amounts for the right and left motors to ensure that we can rotate towards the ball as appropriately.

The LEFT_MOTOR and RIGHT_MOTOR variables that are set are then mapped to the left and right motors using the [SSC](#) module. Likewise, the TILT variable is mapped to the camera tilt servo. Note that they all range from 0 to 255. If you change the above equation to be more aggressive you need to ensure that the values stay within the 0 to 255 range.

BucketBot keeping a ball at a distance

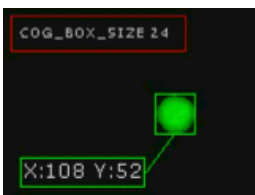


(3.9 MB) Video of the BucketBot from our perspective as it reacts to the green ball.

Please note that the following videos were taken on the bucketbot during processing so they are small.



(1.9 MB) Video of the BucketBot as it sees the ball and adjusts by moving. This is the original unprocessed video.



(1.4 MB) The processed video as performed on the BucketBot. The objective is to keep the green ball in the middle of the screen!

Your turn ...

Want to try this yourself? [Download the Object Tracking](#).robo file to run an example yourself. Note that you need RoboRealm installed to run this file and an SSC servo controller to run the servo motors. Clicking on any of the modules will bring up that interface and allow you to customize it for yourself. Try getting the robot to track red objects ...

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

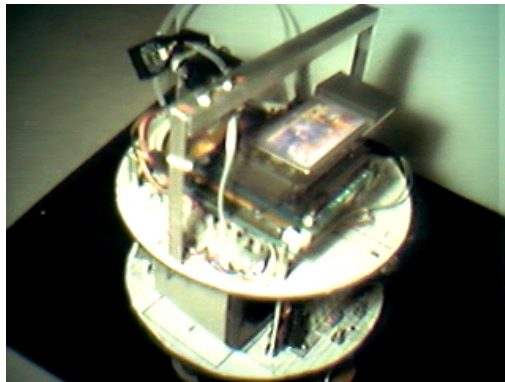
Have a nice day!

Visual Line Following

The following tutorial is one way to use a vision system to identify and follow a line.

The system uses:

- single CCD camera
- USB Digitizer
- Pentium CPU on board
- Left, Right differential motors

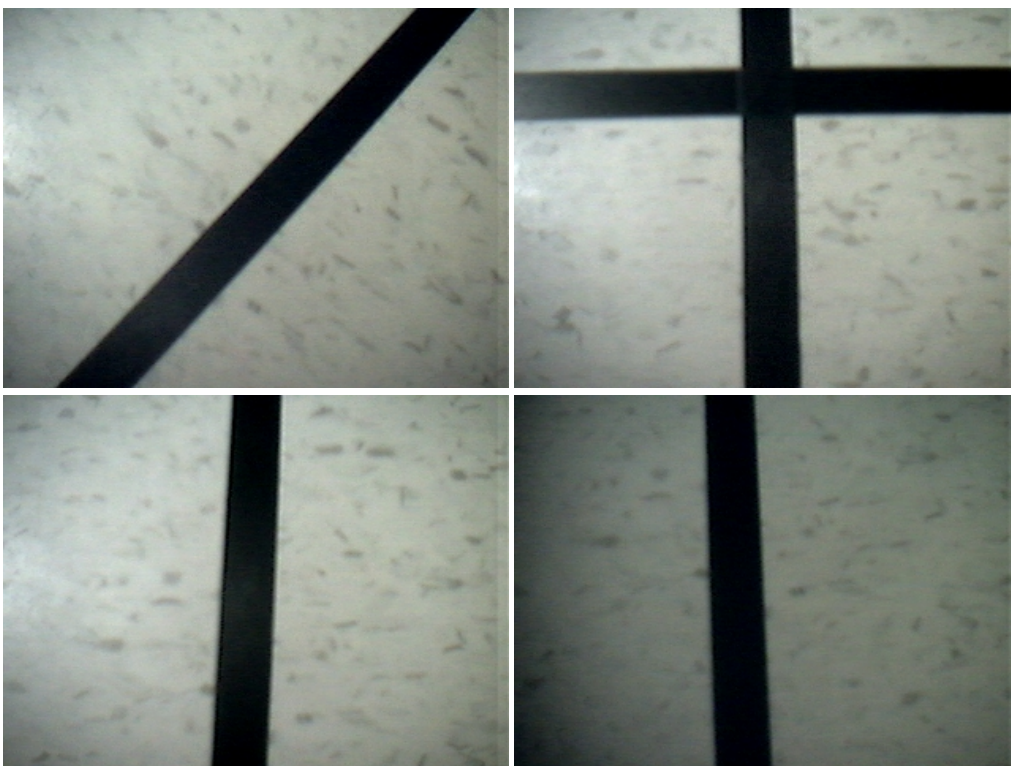


Example Images

To get a better idea of what we're trying to accomplish lets first look at some sample pictures that the BucketBot took. The images are from the CCD camera mounted to the front of the BucketBot angled towards the ground.

Note that we have not done any calibration of lighting adjustments. The images are straight from the camera.

If it worth mentioning that the lines are created by black electrical tape stuck on moveable floor tiles. This allows us to move the tiles around and experiment with shapes quite easily.



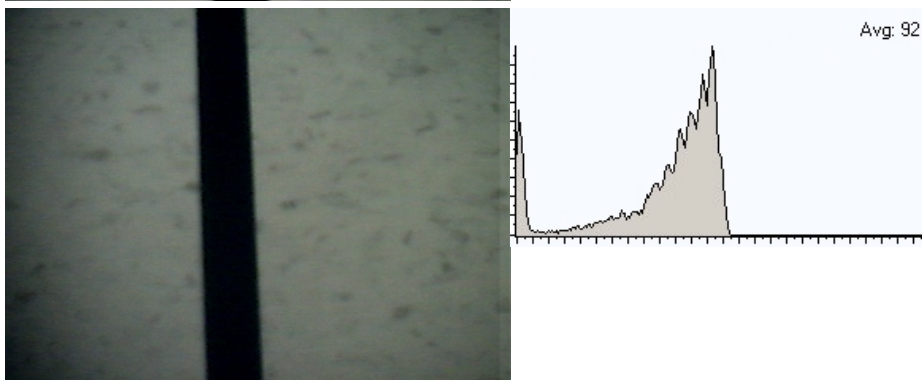
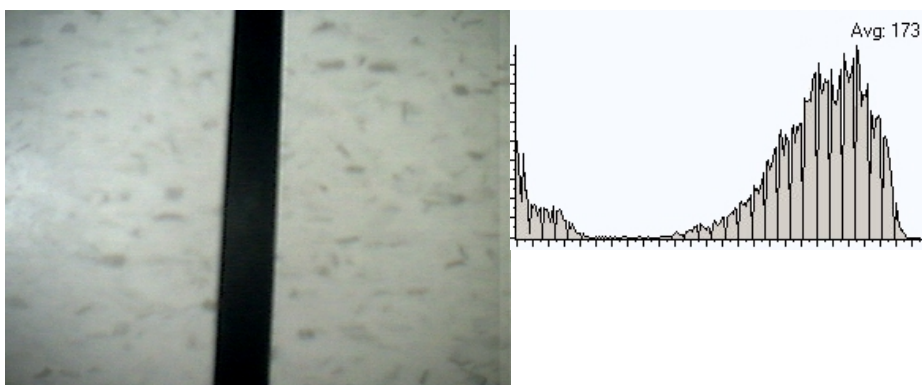


Lighting Issues...

Bad lighting can really cause problems with most image analysis techniques (esp. when thresholding). Imagine if a robot were to suddenly move under a shadow and lose all control! The best vision techniques try to be more robust to lighting changes.

To understand some of those issues lets look at two histograms from a straight line image from the previous slide. Next to each of these images are the images histogram. The histogram of an image is a graphical representation of the count of different pixel intensities in the image.

As you can see, histograms for lighter images slump towards the right (towards the 255 or highest value) whereas darker images have histograms with most pixels closer to zero. This is obvious but using the histogram representation we can better understand how transformations to the image change the underlying pixels.



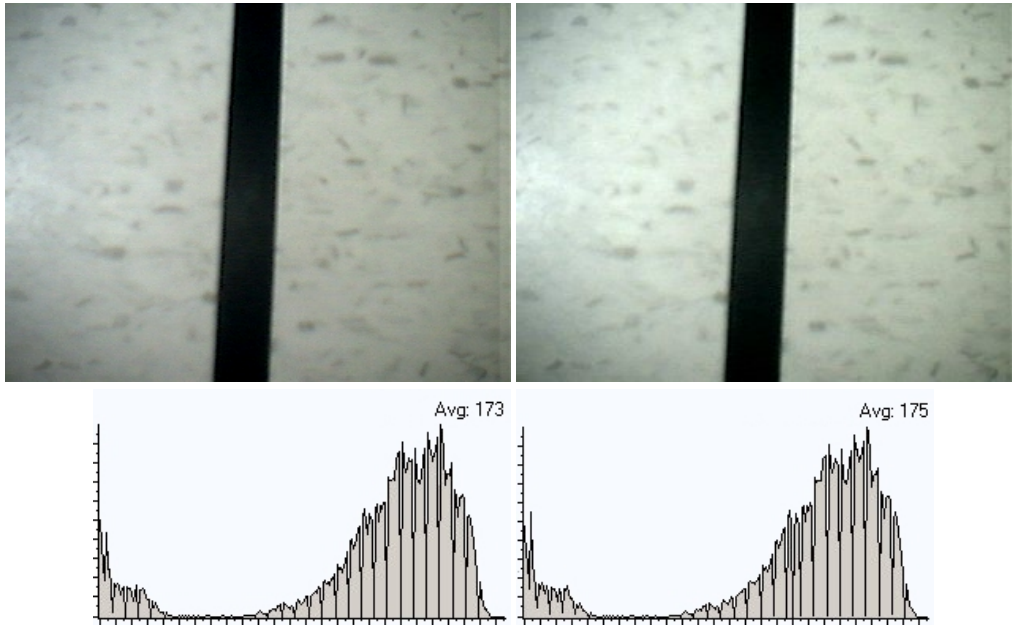
The next step is to see if we can correct these two images so that they look closer to one another. We do that by normalizing the images.

Normalize Intensities

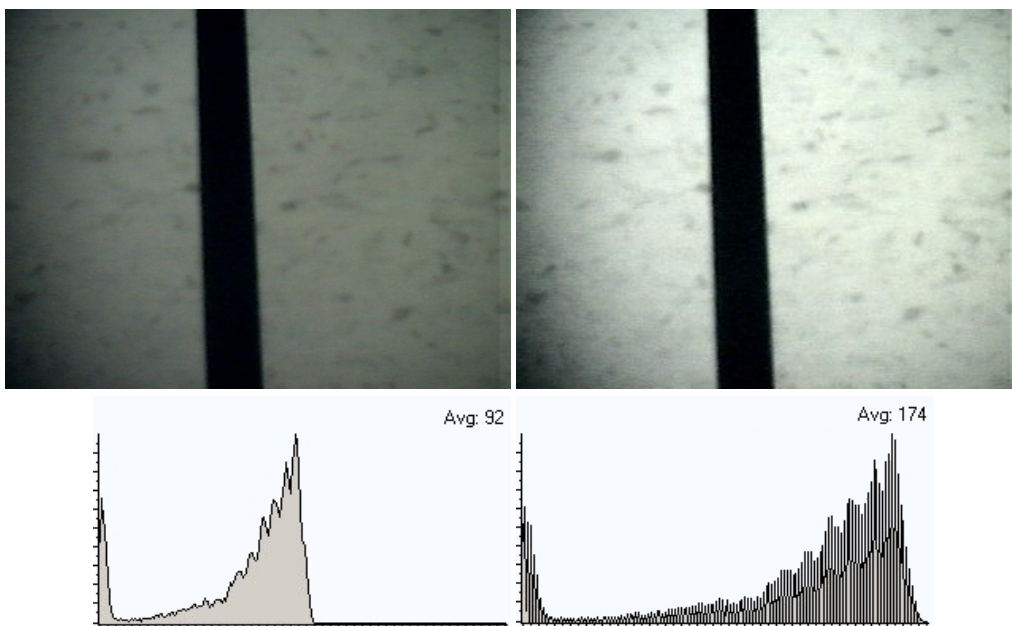
To counter the effects of bad lighting we have to normalize the image. Image normalization attempts to spread the pixel intensities over the entire range of intensities. Thus if you have a very dark image the resulting normalization process will replace many of the dark pixels with lighter pixels while keeping the relative positions the same, i.e. two pixels may be made lighter but the darker of the two will still be darker relative to the second pixel.

By evenly distributing the image intensities other image processing functions like thresholding which are based on a single cutoff pixel intensity become less sensitive to lighting.

For example, the following images from the previous page show what happens during normalization. It is important to note that any image transformation that is meant to improve bad images must also preserve already good ones. In testing image processing functions be sure to always understand the negative side effects that any function can have.



Normalization did not create much change because image was lighted ok to begin with.



The bad image experienced a large amount of change as the image intensities did not cover the entire intensity range due to bad lighting. You can see from the histogram that the image intensities are now more evenly distributed.

Also you can note how the new histogram appears to be not as solid as the original. This is due to how the intensity values are stretched. Since the new image has exactly the same number of pixels as the old image the new image still has many pixels intensity values that do not exist and therefore show up as gaps in the histogram. Adding another filter like a mean blur would cause the histogram to become more solid again as the gaps would

show up as gaps in the histogram. Adding another filter like a mean filter would cause the histogram to become more solid again as the gaps would be filled due to smoothing of the image.

Next we need to start focusing on extracting the actual lines in the images.

Edge Detection

In order to follow the line we need to extract properties from the image that we can use to steer the robot in the right direction. The next step is to identify or highlight the line with respect to the rest of the image. We can do this by detecting the transition from background tile to the line and then from the line to the background tile. This detection routine is known as edge detection.

The way we perform edge detection is to run the image through a convolution filter that is 'focused' on detecting line edges. A convolution filter is a matrix of numbers that specify values that are to be multiplied, added and then divided from the image pixels to create the resulting pixel value.

To learn more about convolution filters have a look at our [help section](#).

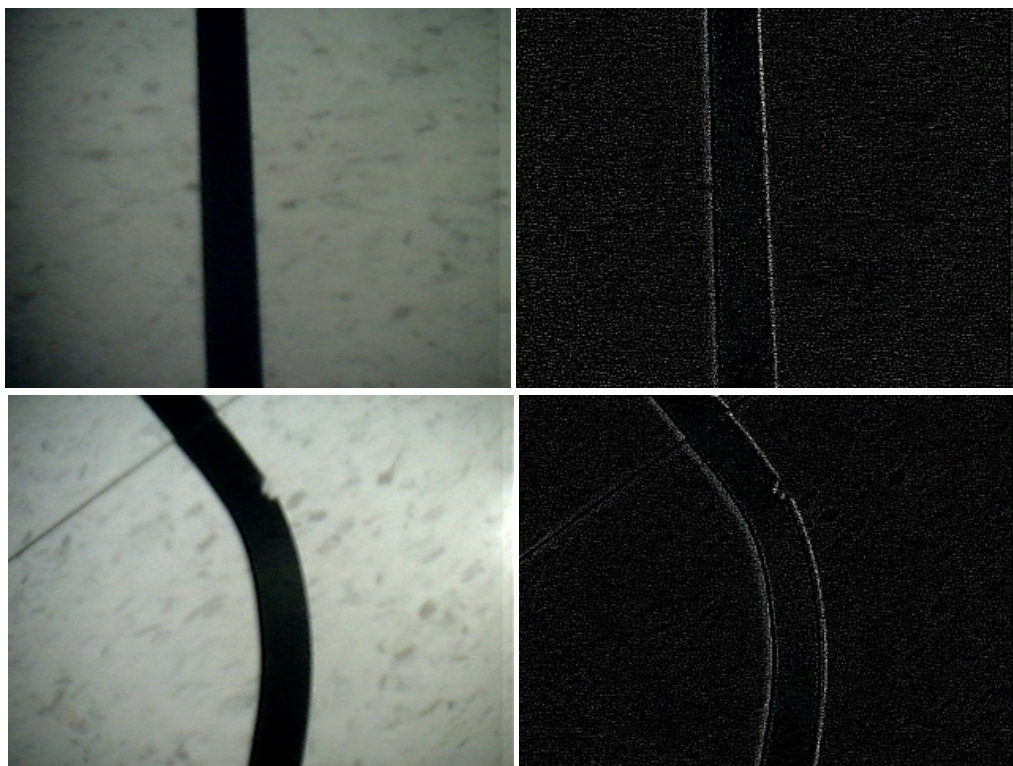
We will start with the following convolution matrix that is geared to detect edges:

-1	-1	-1
-1	8	-1
-1	-1	-1

The next step is to run the images through this edge detector and see what we get ...

Edge Detection

Here are two sample images with the convolution edge detection matrix run after normalization:



It is interesting to note that while the convolution filter does identify the line it also picks up on a lot of speckles that are not really desired as it

causes noise in the resulting image.

Also note that the detected lines are quite faint and sometimes even broken. This is largely due to the small (3x3) neighborhood that the convolution filter looks at. It is easy to see a very large difference between the line and the tile from a global point of view (as how you and I look at the images) but from the image pixel point of view it is not as easy. To get a better result we need to perform some modifications to our current operations.

Modified Line Detection

To better highlight the line we are going to:

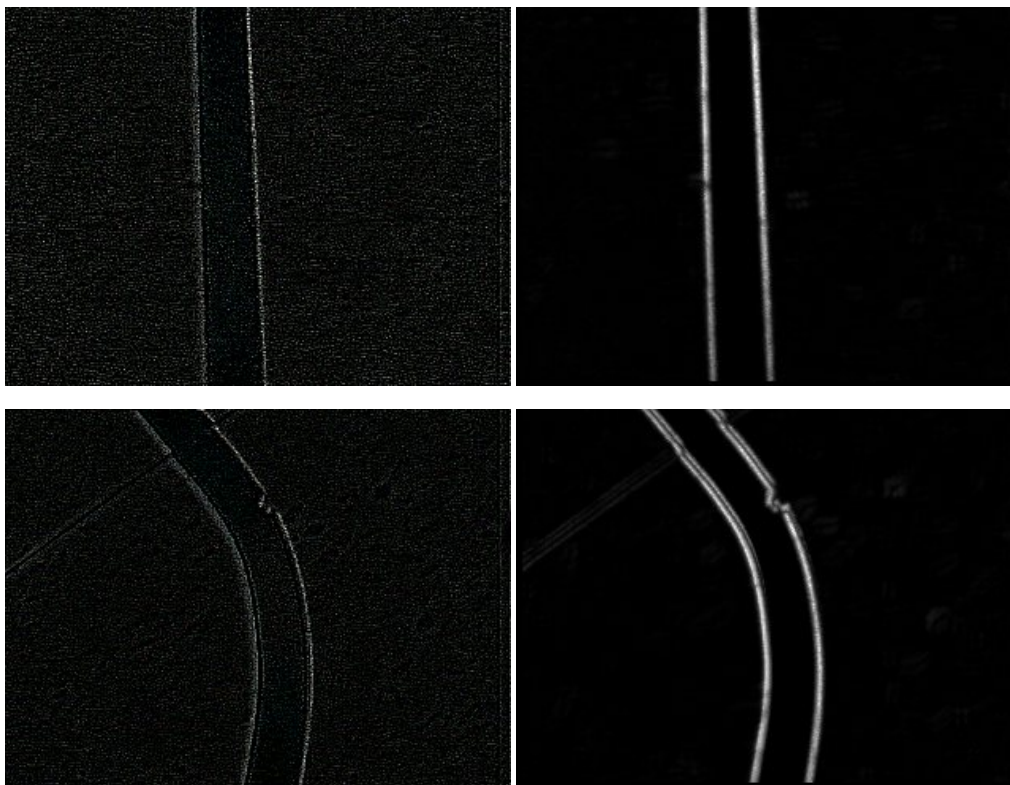
1. Use a larger filter; instead of a 3x3 neighborhood we will use a 5x5. This will result in larger values for edges that are "thicker". We will use the following matrix:

```
-1 -1 -1 -1 -1
-1  0  0  0 -1
-1  0 16  0 -1
-1  0  0  0 -1
-1 -1 -1 -1 -1
```

2. Further reduce the speckling issue we will square the resulting pixel value. This causes larger values to become larger but smaller values to remain small. The result is then normalized to fit into the 0-255 pixel value range.

3. Threshold the final image by removing any pixels lower than a 40 intensity value.

The results of this modified technique:



We now nicely see the line edges and most of the noise speckles are gone. We can now continue to the next step which is how to understand these images in order to map the results to left and right motor pulses.

Center of Gravity

There are many ways we could translate the resulting image intensities into right and left motor movements. A simple way would be to add up all the pixel values of the left side of the image and compare the result to the right side. Based on which is more the robot would move towards that side.

At this point, however, we will use the COG or Center of Gravity of the image to help guide our robot.

The COG of an object is the location where one could balance the object using just one finger. In image terms it is where one would balance all the white pixels at a single spot. The COG is quick and easy to calculate and will change based on the object's shape or position in an image.

To calculate the COG of an image add all the x,y locations of non-black pixels and divide by the number of pixels counted. The resulting two numbers (one for x and the other for y) is the COG location.

... (code for finding the center of the image) ...

The COG location is the red square with a green line from the center of the image to the COG location.

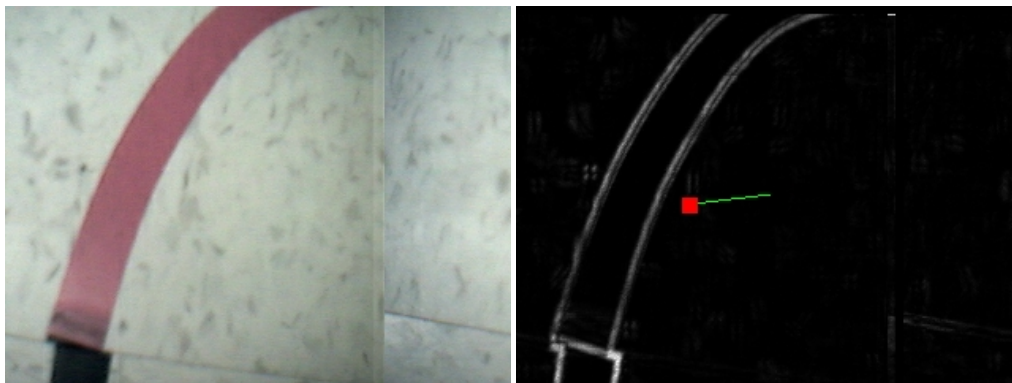


Based on the location of the COG we can now apply the following conditions to the robot motors:

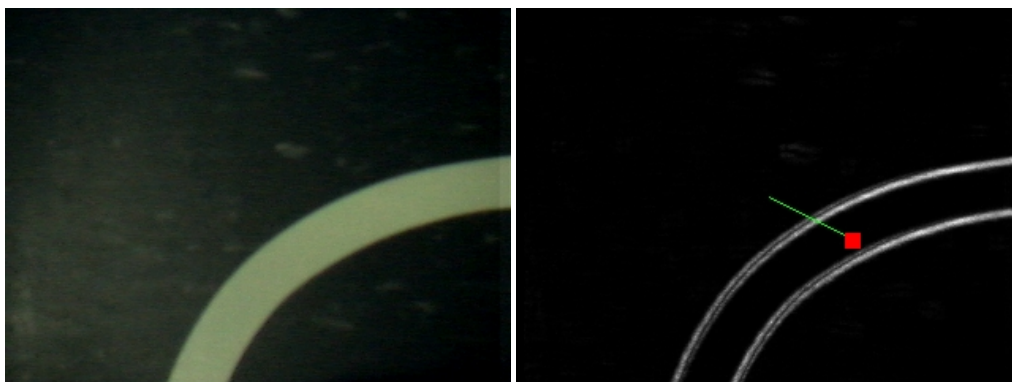
- When the COG is to the right of the center of screen, turn on the left motor for a bit.
- When the COG is on the left, turn on the right motor.
- When the COG is below center, apply reverse power to opposite motor to pivot the robot.

You can also try other conditions such as when the distance of the COG to the center of screen is large really turn up the motor values to try to catch up to the line.

This technique works regardless of the color of the line ... as long as the line can be well differentiated from the background we can detect it!



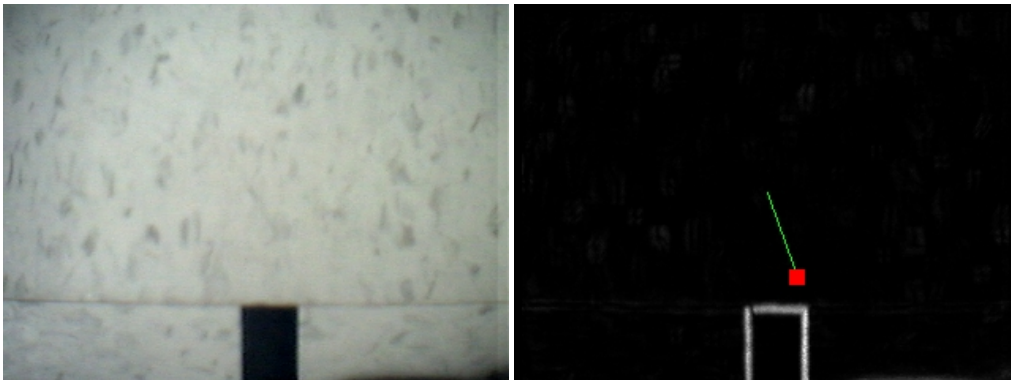
It even works if we reverse the original black line on white tile assumption.



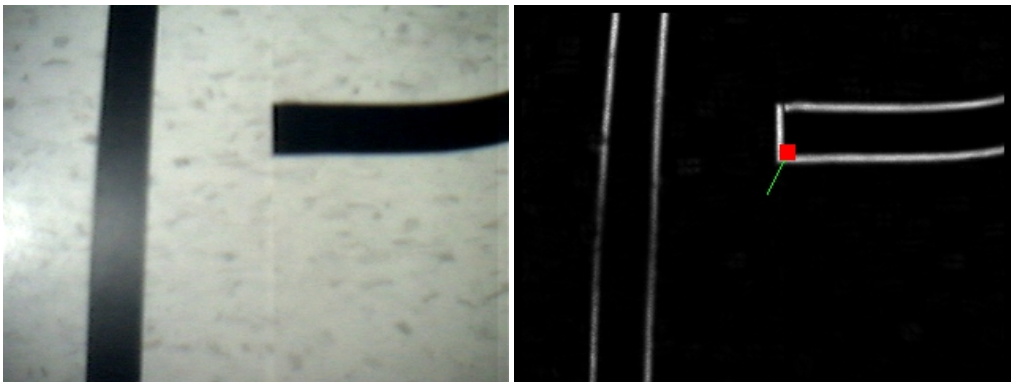
In fact, we don't really even need a connected line!



Here is a case where the end of the line is reached. Notice that the COG has fallen below the center of the screen. Perhaps we can use this condition to stop the robot!



Opps, a mistake! The robot will start following the wrong line ... but there again no technique is perfect.



Your turn ...

Want to try this yourself? Following is the robofile that should launch RoboRealm with the original image we used in this tutorial. You can click on each of the steps in the pipeline to review what the processing looks like up to that point.

[Download Line following .robo file](#)

The End

That's all folks. We hope you've enjoyed this little adventure into an application of machine vision processing and have inspired you to [download a free trial](#) of our software.

If you have any questions or comments about this tutorial please feel free to [contact us](#).

Have a nice day!

