Vision Processing

Last Updated: 05-11-2016

Table of Contents

Setup

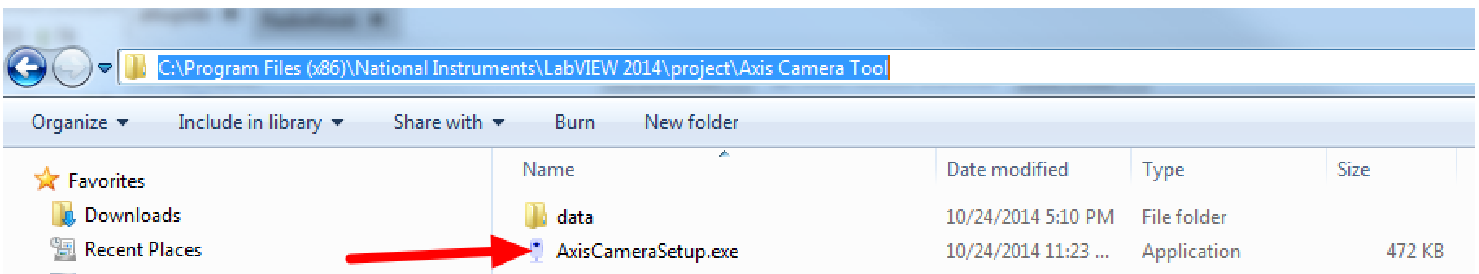
Configuring an Axis Camera

Three different Axis camera models are supported by the FRC software, the Axis 206, Axis M1011 and Axis M1013. This document provides instructions on how to configure one of these cameras for FRC use. To follow the instructions in this document, you must have installed the NI 2015 FRC Update Suite and Configured your radio

Connect the camera

Connect the Axis camera to the DAP-1522 radio using an Ethernet cable. Connect your computer to the radio using an ethernet cable or via a wireless connection.

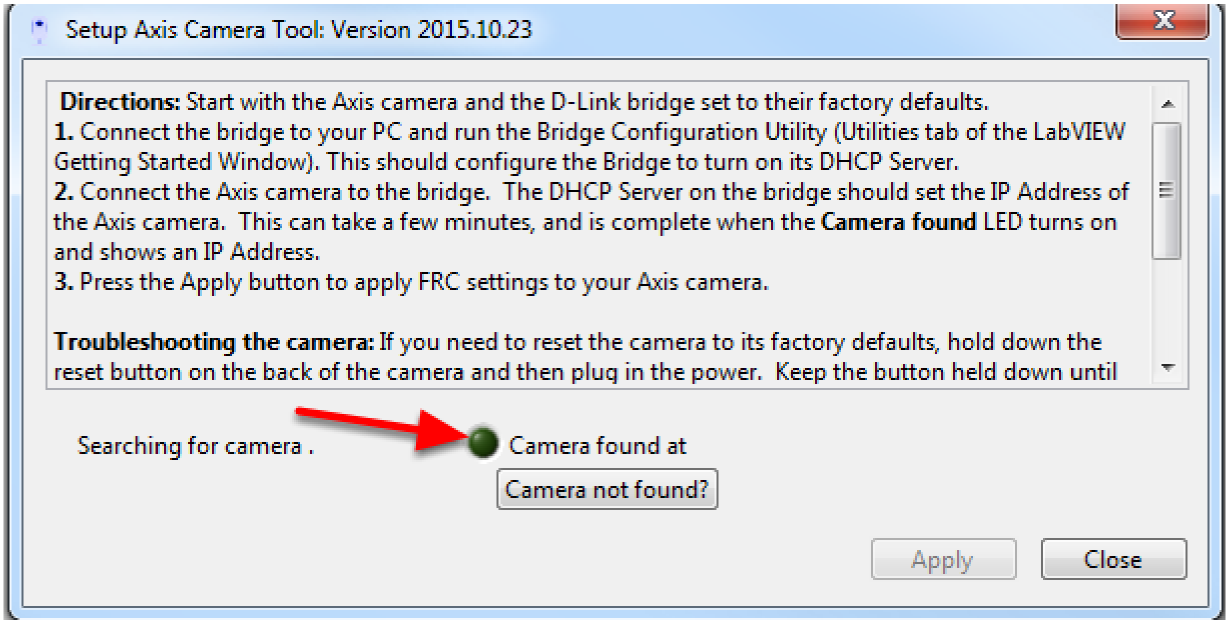
Axis Camera Setup Tool



Browse to **C:\Program Files (x86)\National Instruments\LabVIEW 2014\project\Axis Camera Tool** and **double-click** on **AxisCameraSetup.exe** to start the Axis Camera Setup Tool.

Vision Processing

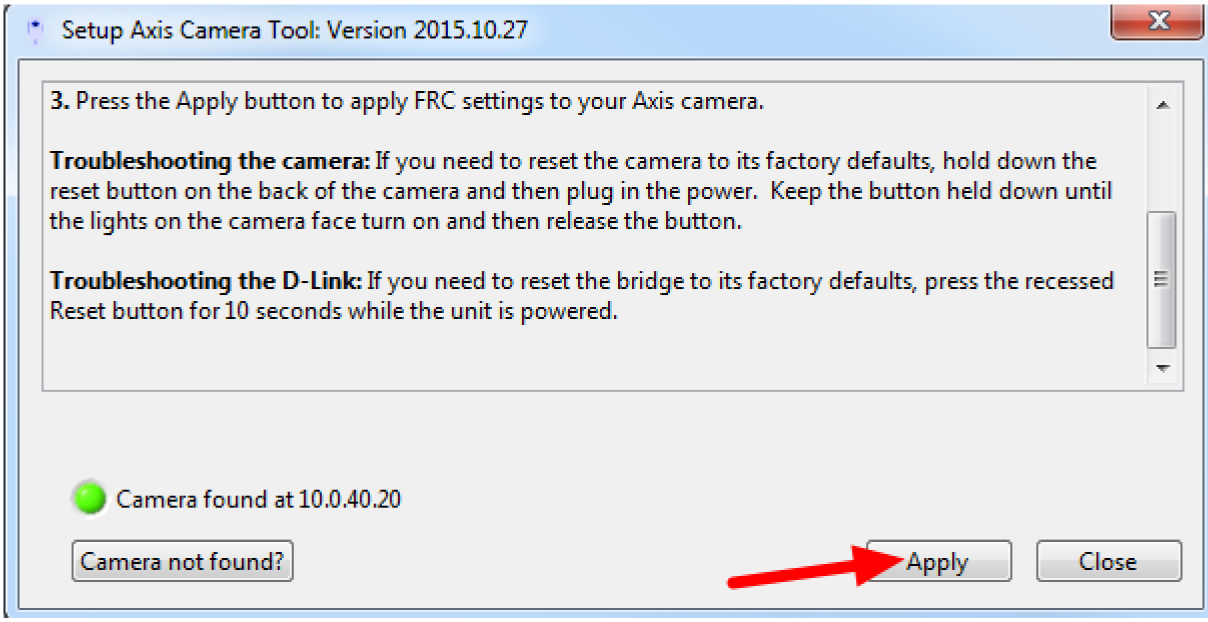
Tool Overview



The camera should be automatically detected and the green indicator light should be lit. If it is not, make sure the camera is powered on (the ring on the camera face should be green) and connected to your computer. If the indicator remains off follow the instructions in the tool textbox next to **Troubleshooting the camera** to reset the camera. You can also use the **Camera not found?** button to check the IP address of your computer, one of the addresses listed should be of the form 10.TE.AM.XX where TEAM is your 4 digit team number.

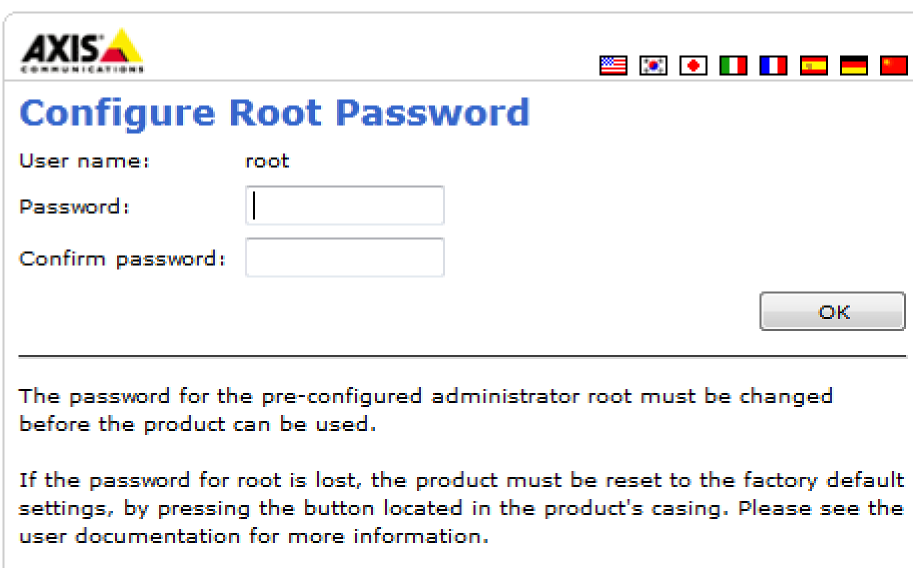
Vision Processing

Setup the Camera



To configure the camera, press Apply. This will configure many of the necessary/recommended settings for using the camera for FRC. Currently the tool does not properly configure the DNS name of the camera in many cases.

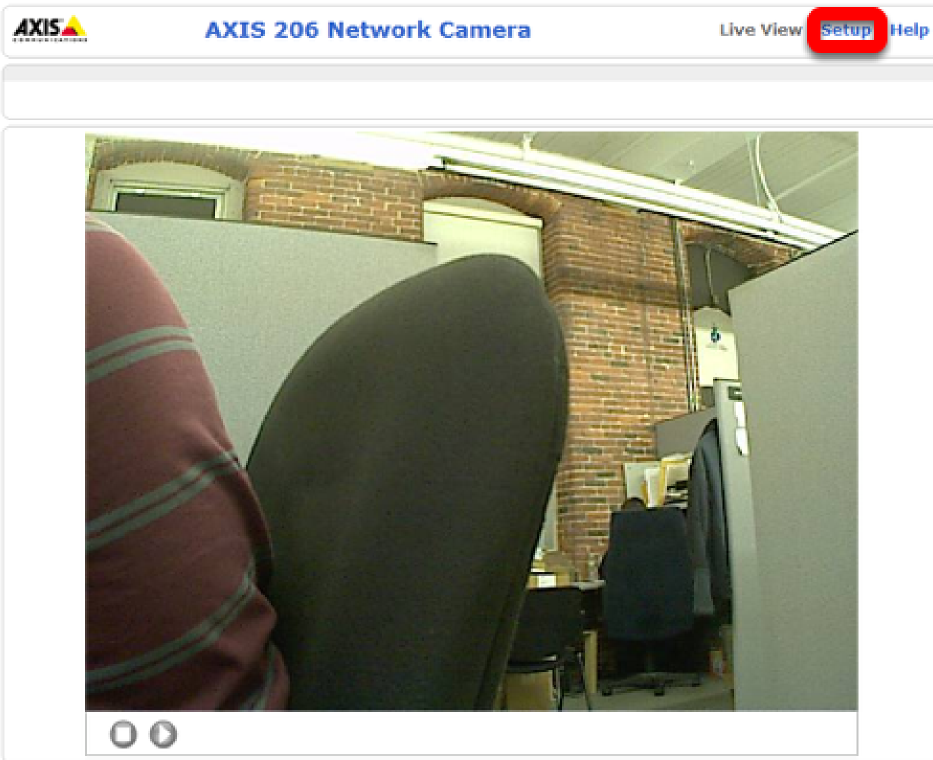
Camera Webpage



Vision Processing

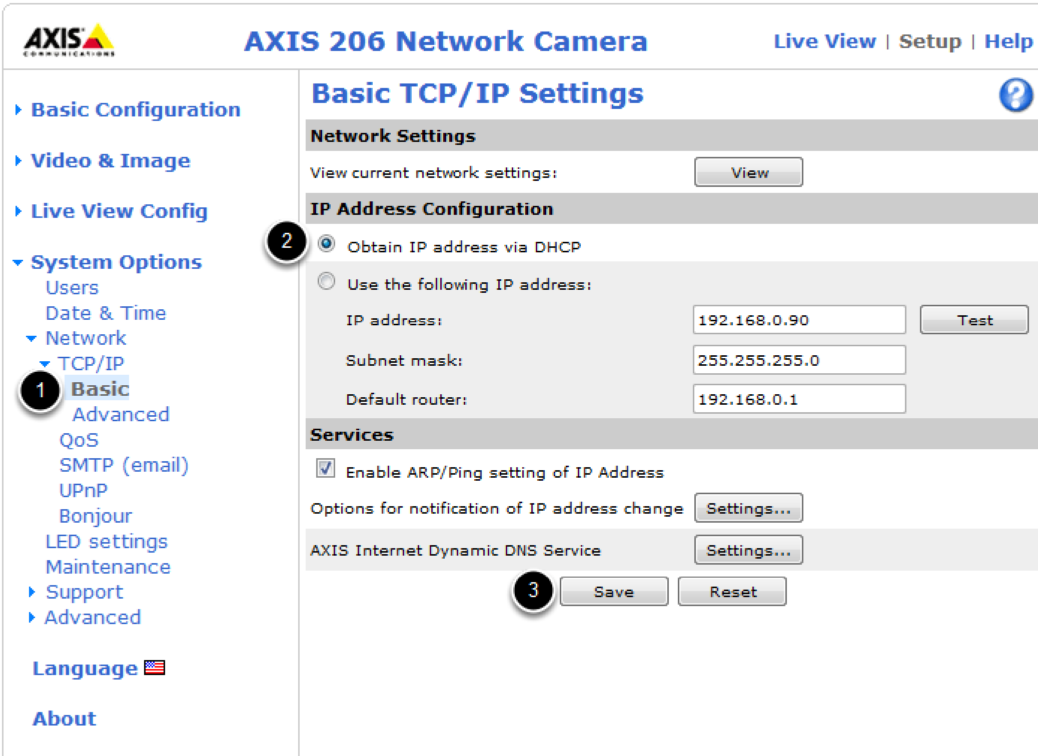
To set the network settings, open a web browser and enter the address shown next to **Camera found at** in the tool (in the example above this is 10.0.40.20) in the address bar and press enter. You should see a Configure Root Password page, set this password to whatever you would like, but **admin** is recommended.

Setup Page



Click **Setup** to go to the setup page.

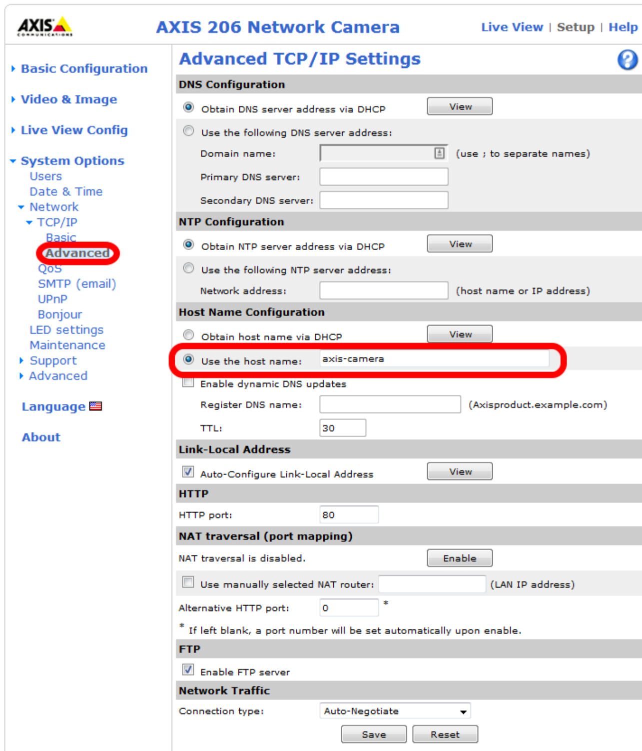
Configure Basic Network Settings



To configure the network settings of the camera, click the arrow to expand the System Options pane, then click the arrow to expand Network, then expand TCP/IP and select Basic. Set the camera to obtain an IP address via DHCP by selecting the bubble. Alternately, you may choose to set a static IP in the range 10.TE.AM.3 to 10.TE.AM.19. This is outside the range handed out by the DAP-1522 radio (home use) or FMS system (event use) so you will avoid any IP conflicts.

Click Save.

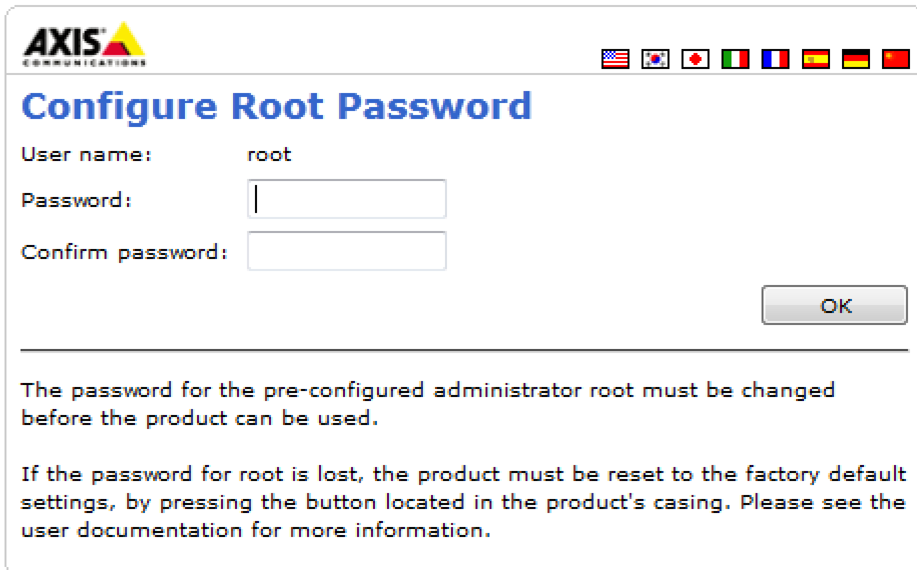
Configure Advanced Network Settings



Next click **Advanced** under **TCP/IP**. Set the **Host Name Configuration** to **"Use the host name:"** and set the value to **"axis-camera"** as shown. If you plan to use multiple cameras on your robot, select a unique host name for each. You will need to modify the dashboard and/or robot code to work with the additional cameras and unique host names.

Click **Save.**

Manual Camera Configuration



It is recommended to use the Setup Axis Camera Tool to configure the Axis Camera. If you need to configure the camera manually, connect the camera directly to the computer, configure your computer to have a static IP of 192.168.0.5, then open a web browser and enter **192.168.0.90** in the address bar and press enter. You should see a Configure Root Password page, set this password to whatever you would like, but **admin** is recommended.

If you do not see the camera webpage come up, you may need to reset the camera to factory defaults. To do this, remove power from the camera, hold the reset button while applying power to the camera and continue holding it until the lights on the camera face turn on, then release the reset button and wait for the lights to turn green. The camera is now reset to factory settings and should be accessible via the 192.168.0.5 address.

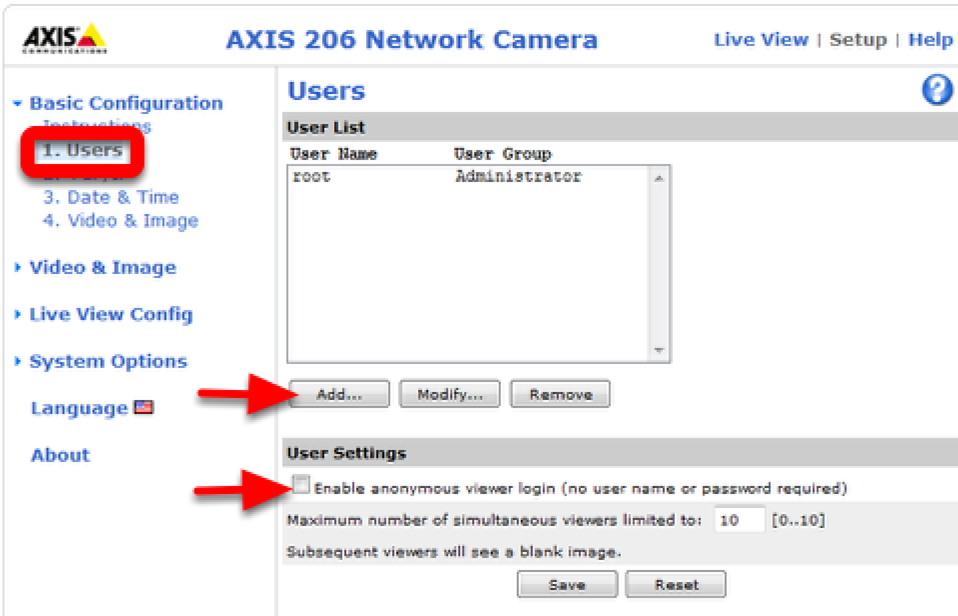
Vision Processing

Setup Page

Click **Setup** to go to the setup page.

Vision Processing

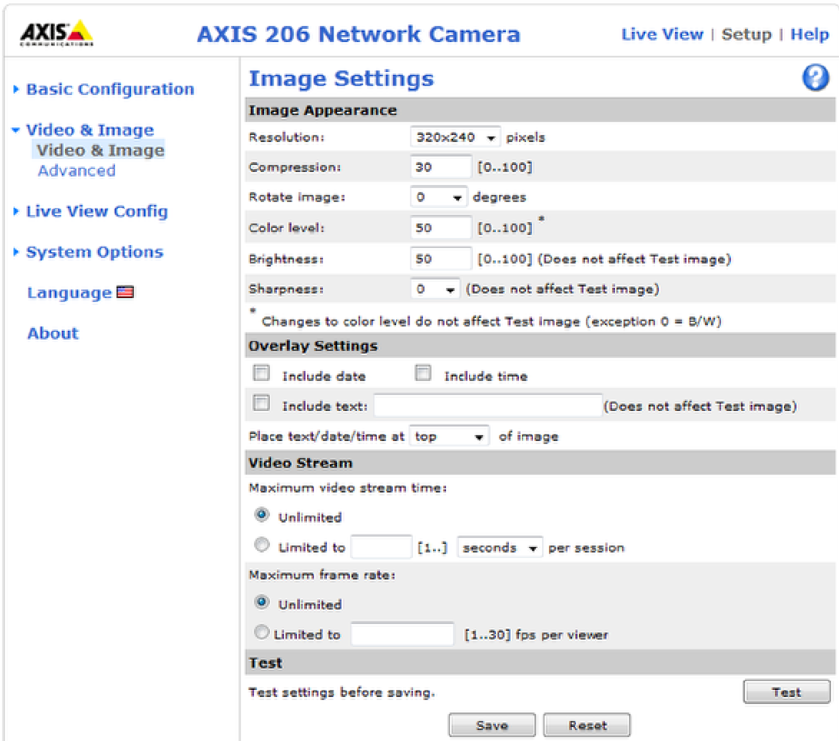
Configure Users



On the left side click **Users** to open the users page. Click **Add** then enter the Username **FRC** Password **FRC** and click the **Administrator** bubble, then click **OK**. If using the SmartDashboard, check the **Enable anonymous viewer login** box. Then click **Save**.

Vision Processing

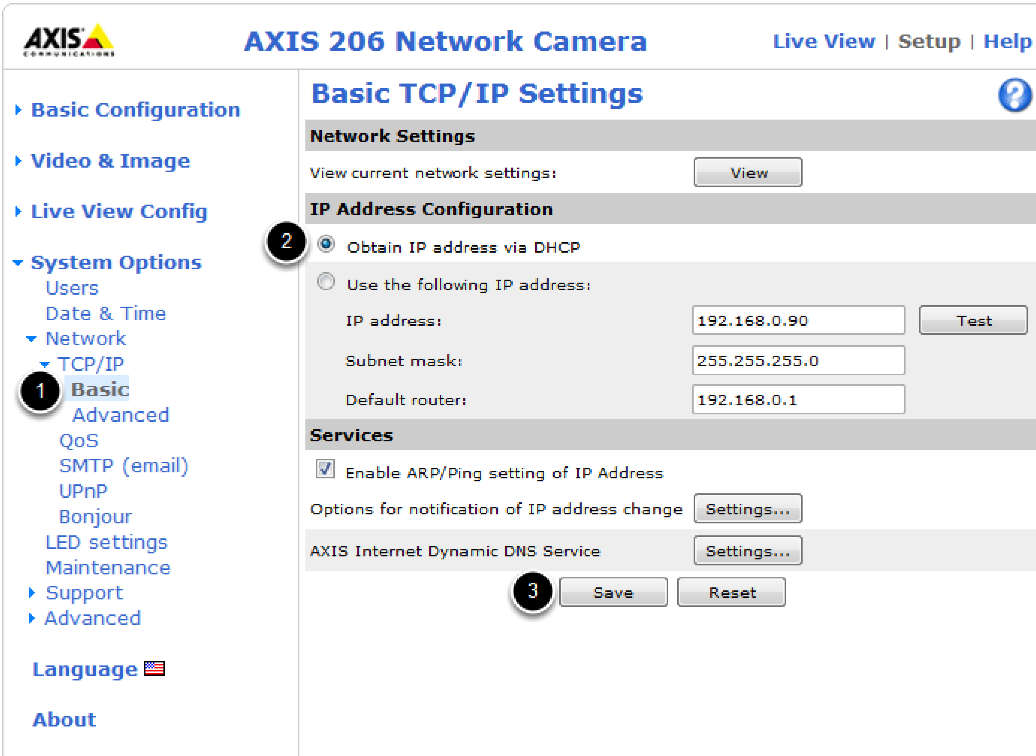
Configure Image Settings



Click **Video & Image** on the left side to open the image settings page. Set the **Resolution** and **Compression** to the desired values (recommended **320x240, 30**). To limit the framerate to under 30 FPS, select the **Limited to** bubble under **Maximum frame rate** and enter the desired rate in the box. Color, Brightness and Sharpness may also be set on this screen if desired. Click **Save** when finished.

Vision Processing

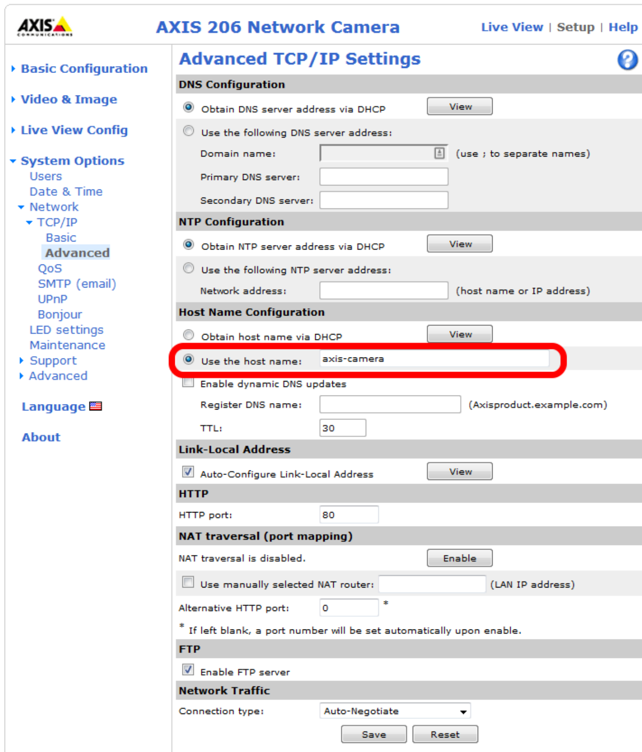
Configure Basic Network Settings



To configure the network settings of the camera, click the arrow to expand the System Options pane, then click the arrow to expand Network, then expand TCP/IP and select Basic. Set the camera to obtain an IP address via DHCP by selecting the bubble. Alternately, you may choose to set a static IP in the range 10.TE.AM.3 to 10.TE.AM.19. This is outside the range handed out by the DAP-1522 radio (home use) or FMS system (event use) so you will avoid any IP conflicts.

Click Save.

Configure Advanced Network Settings



Next click **Advanced** under **TCP/IP**. Set the **Host Name Configuration** to **"Use the host name:"** and set the value to **"axis-camera"** as shown. If you plan to use multiple cameras on your robot, select a unique host name for each. You will need to modify the dashboard and/or robot code to work with the additional cameras and unique host names.

Click **Save**.

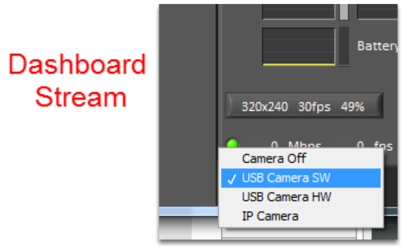
Using the Microsoft Lifecam HD-3000

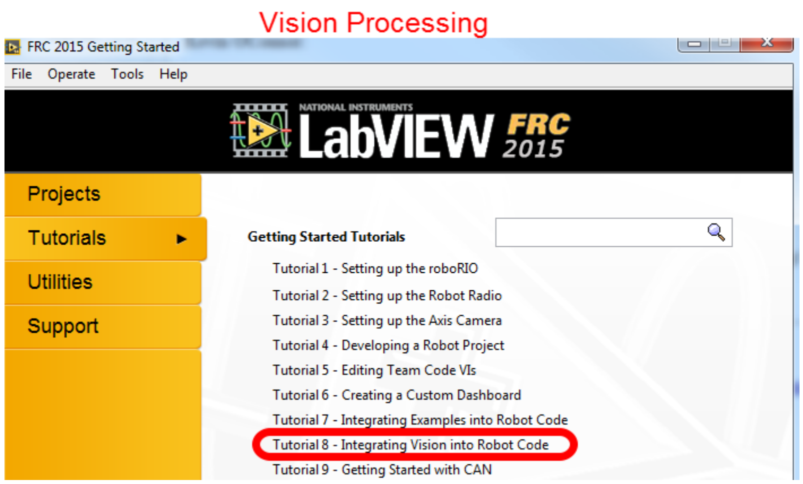
The Microsoft Lifecam HD-3000 is a USB webcam that was tested with the roboRIO as part of the Beta testing and software development effort. While other USB webcams may work with the roboRIO, this camera has been tested to be compatible with the provided software.

Connecting the camera to the roboRIO

The camera can be connected to either of the roboRIO USB ports.

Using the camera - LabVIEW





To stream the camera back to the Dashboard using LabVIEW, no additional code is necessary. Simply select USB HW (image compression done by the camera, fewer options but lower roboRIO CPU usage) or USB SW (image compressed by roboRIO, more options, but higher roboRIO CPU usage) and the image should begin streaming back.

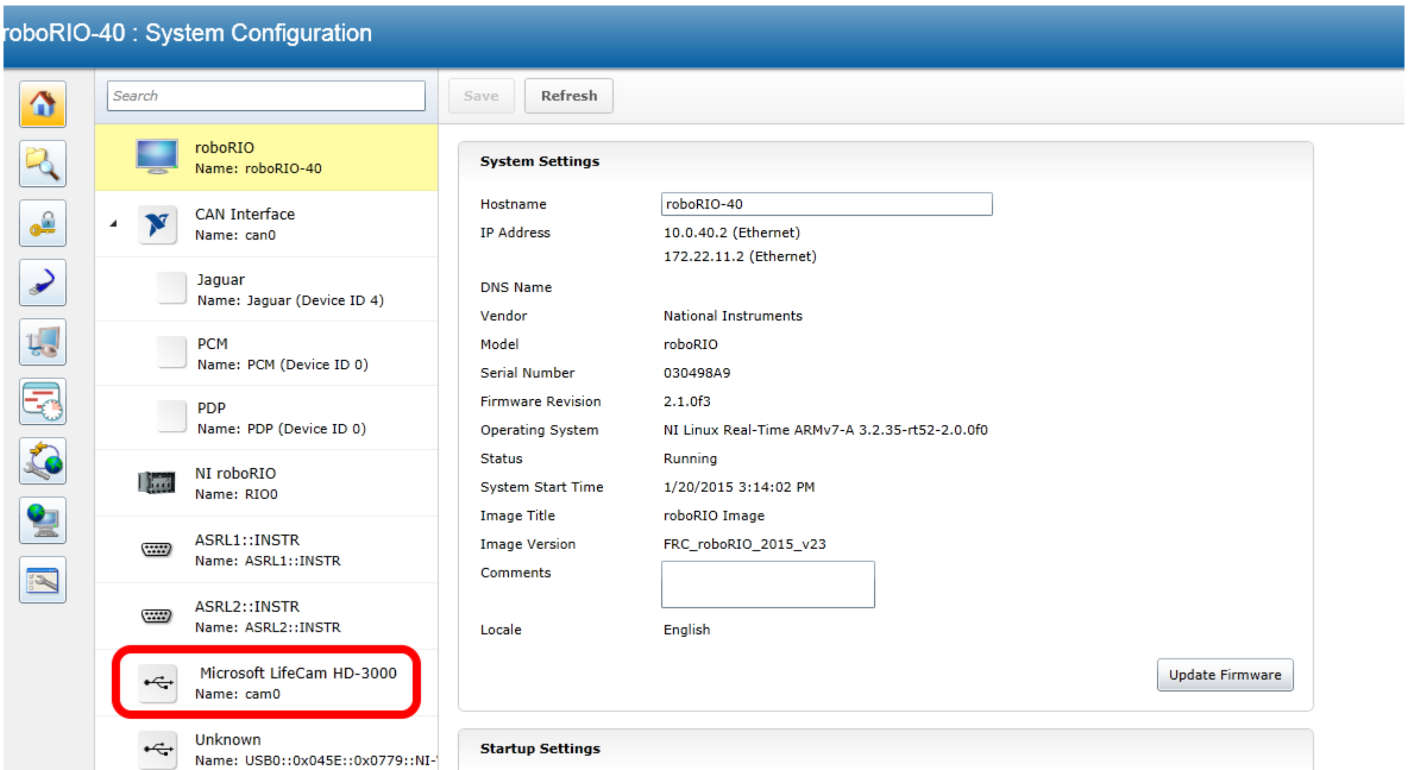
Note: The camera should be plugged in before your LabVIEW code starts running to work properly. If you just plugged in the camera rebooting the roboRIO is a quick way to make sure it is recognized properly.

The default LabVIEW templates and the image processing examples are already set up for the USB camera if you want to do image processing. On the LabVIEW splash screen, click Tutorials, then click Tutorial 8 for more information about integrating Vision processing in your LabVIEW code.

Using the Camera - C++\Java

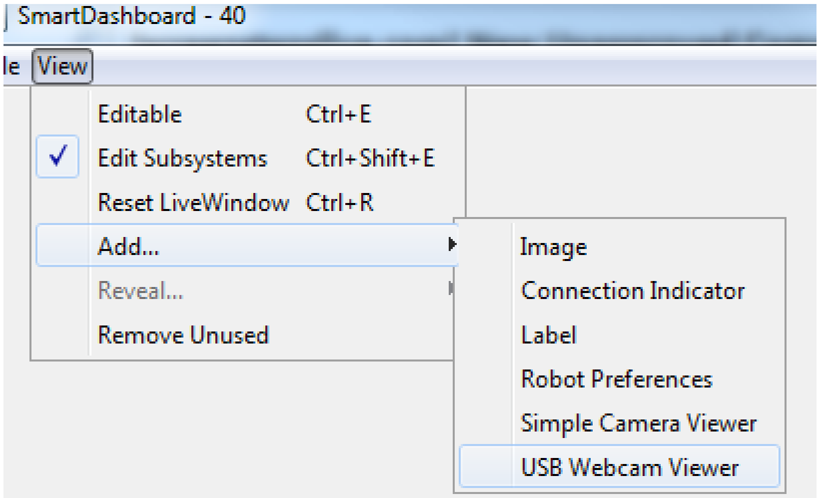
To stream the camera back to the Dashboard using C++ or Java robot code, you will need to add some code to your robot project. Example have been provided in Eclipse to illustrate the use of the CameraServer class to automatically stream images back to the Dashboard (SimpleVision) and send back modified images (IntermediateVision and the 2015 Vision examples). This class will allow images to be streamed back to either the SmartDashboard "USB Webcam Viewer" or the LabVIEW Dashboard (set to USB HW).

Determining the Camera name



Unlike the LabVIEW code which attempts to determine the camera name of the camera you want to use, the C++\Java code requires you to specify the camera name. To determine the name of the desired camera, you will need to use the roboRIO webdashboard. For more information about accessing the roboRIO webdashboard see RoboRIO Webdashboard. Open the roboRIO webdashboard and locate the camera in the left pane and note the Name, this is the string you will pass into the camera server or IMAQDx open (in the image above, the camera name is cam0).

Using the SmartDashboard USB Camera Viewer

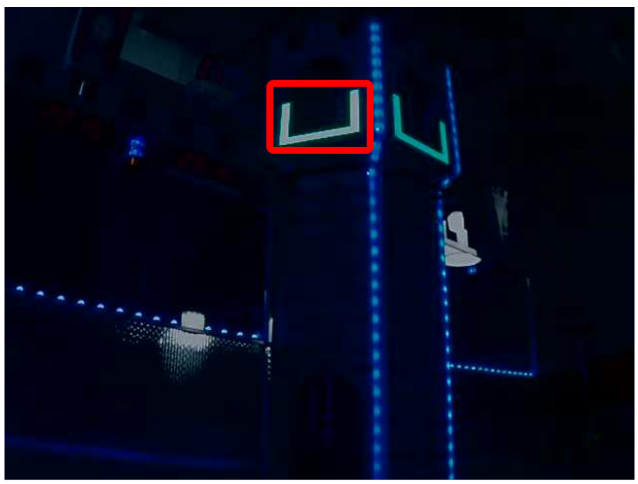


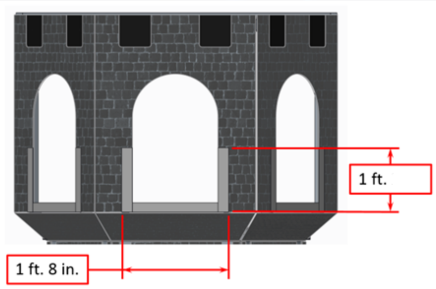
To view the camera stream from the LabVIEW dashboard, set the camera dropdown to USB HW. To view the stream from the SmartDashboard you will need to add a USB Webcam Viewer widget to your layout. For more information on the SmartDashboard and widgets see the SmartDashboard manual. To add the USB Webcam Viewer widget to the SmartDashboard, select View -> Add -> USB Webcam Viewer. To move or resize the viewer widget, make the layout Editable by selecting that option in the View menu (select again to disable).

Target Info and Retroreflection

This document describes the Vision Targets from the 2016 FRC game and the visual properties of the material making up the targets. Note that for official dimensions and drawings of all field components, please see the [Official Field Drawings]

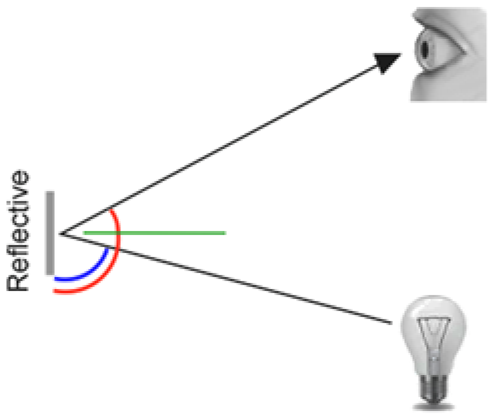
Targets

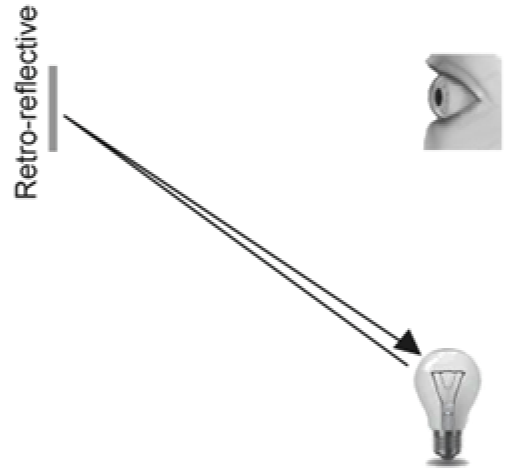




Each Target consists of a 1' 8" wide, 1' tall U-shape made of 2" wide retroreflective material (3M 8830 Silver Marking Film). The targets are located immediately adjacent to the bottom of each high goal. When properly lit, the retroreflective tape produces a bright and/or color-saturated marker.

Retroreflectivity vs. Reflectivity

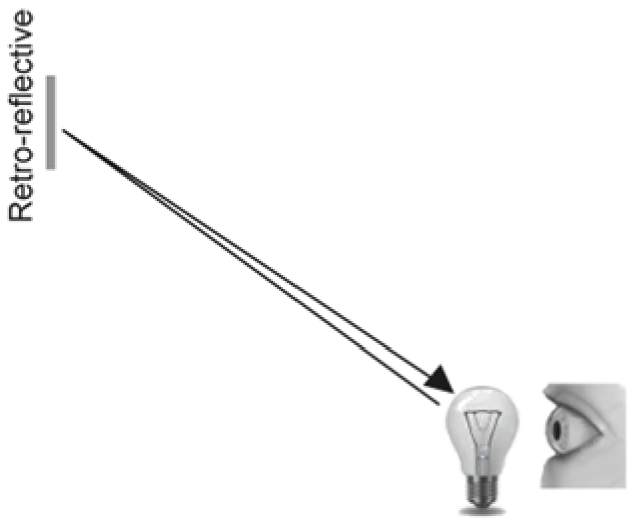




Highly reflective materials are generally mirrored so that light “bounces off” at a supplementary angle. As shown above-left, the blue and red angles sum to 180 degrees. An equivalent explanation is that the light reflects about the surface normal the green line drawn perpendicular to the surface. Notice that a light pointed at the surface will return to the light source only if the blue angle is ~90 degrees.

Retro-reflective materials are not mirrored, but it will typically have either shiny facets across the surface, or it will have a pearl-like appearance. Not all faceted or pearl-like materials are retro-reflective, however. Retro-reflective materials return the majority of light back to the light source, and they do this for a wide range of angles between the surface and the light source, not just the 90 degree case. Retro-reflective materials accomplish this using small prisms, such as found on a bicycle or roadside reflector, or by using small spheres with the appropriate index of refraction that accomplish multiple internal reflections. In nature, the eyes of some animals, including house cats, also exhibit the retro-reflective effect typically referred to as night-shine. The Wikipedia articles on retro-reflection go into more detail on how retro-reflection is accomplished.

Examples of Retroreflection



This material should be relatively familiar as it is often used to enhance nighttime visibility of road signs, bicycles, and pedestrians.

Initially, retro-reflection may not seem like a useful property for nighttime safety, but when the light and eye are near one another, as shown below, the reflected light returns to the eye, and the material shines brightly even at large distances. Due to the small angle between the driver's eyes and vehicle headlights, retro-reflective materials can greatly increase visibility of distant objects during nighttime driving.

Demonstration

To further explore retro-reflective material properties:

1. Place a piece of the material on a wall or vertical surface
2. Stand 10-20 feet away, and shine a small flashlight at the material.
3. Start with the light held at your belly button, and raise it slowly until it is between your eyes. As the light nears your eyes, the intensity of the returned light will increase rapidly.
4. Alter the angle by moving to other locations in the room and repeating. The bright reflection should occur over a wide range of viewing angles, but the angle from light source to eye is key and must be quite small.

Experiment with different light sources. The material is hundreds of times more reflective than white paint; so dim light sources will work fine. For example, a red bicycle safety light will demonstrate that the color of the light source determines the color of the reflected light. If possible, position several team members at different locations, each with their own light source. This will show that the effects are largely independent, and the material can simultaneously appear different colors to various team members. This also demonstrates that the material is largely immune to environmental lighting. The light returning to the viewer is almost entirely determined by a light source they control or one directly behind them. Using the flashlight, identify other retro-reflective articles already in your environment … on clothing, backpacks, shoes, etc.

Lighting

We have seen that the retro-reflective tape will not shine unless a light source is directed at it, and the light source must pass very near the camera lens or the observer's eyes. While there are a number of ways to accomplish this, a very useful type of light source to investigate is the ring flash, or ring light, shown above. It places the light source directly on or around the camera lens and provides very even lighting. Because of their bright output and small size, LEDs are particularly useful for constructing this type of device.

As shown above, inexpensive circular arrangements of LEDs are available in a variety of colors and sizes and are easy to attach to the Axis cameras. While not designed for diffuse even lighting, they work quite well for causing retro-reflective tape to shine. A small green LED ring is available through [FIRST Choice.](#) Other similar LED rings are available from the supplier, [SuperBrightLEDs.com](#)

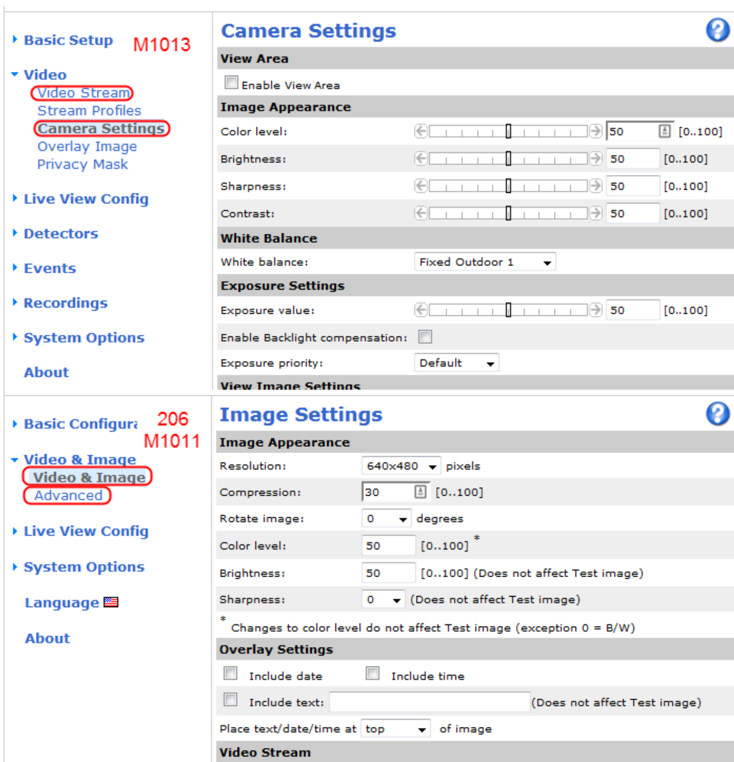
Sample Images

Sample Images are located with the code examples for each language (packaged with LabVIEW, in a separate ZIP in the same location as the C++\Java samples).

Camera Settings

It is very difficult to achieve good image processing results without good images. With a light mounted near the camera lens, you should be able to use the provided examples, the dashboard or SmartDashboard, NI Vision Assistant or a web browser to view camera images and experiment with camera settings.

Changing Camera Settings



To change the camera settings on any of the supported Axis cameras (206, M1011, M1013), browse to the camera's webpage by entering it's address (usually 10.TE.AM.11) in a web browser. Click **Setup** near the top right corner of the page. On the M1013, the settings listed below are split between the **Video Stream** page and the **Camera Settings** page, both listed under the **Video** section.

Focus

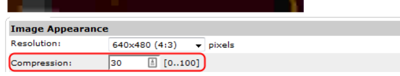
The Axis M1011 has a fixed-focus lens and no adjustment is needed. The Axis 206 camera has a black bezel around the lens that rotates to move the lens in and out and adjust focus. The Axis M103 has a silver and black bezel assembly around the lens to adjust the focus. Ensure that the images you are processing are relatively sharp and focused for the distances needed on your robot.

Compression

320x240 Color Image:
Compression set to 0. Image file size is 20,715 bytes.
High quality, but large in size. May be slower to compress and decompress -- which impacts frame rate.

320x240 Color Image:
Compression set to 30. Image file size is 6,450 bytes. Good quality, relatively small in size. Some image artifacts are present on edges.

320x240 Color Image:
Compression set to 100. Image file size is 2,222 bytes. Poor quality for processing. Notice blocky artifacts and rough edges.



The Axis camera returns images in BMP, JPEG, or MJPG format. BMP images are quite large and take more time to transmit to the cRIO and laptop. Therefore the WPILib implementations typically use MJPG motion JPEG. The compression setting ranges form 0 to 100, with 0 being very high quality images with very little compression, and 100 being very low quality images with very high compression. The camera default is 30, and it is a good compromise, with few artifacts that will degrade image processing. Due to implementation details within the VxWorks memory manager, you may notice a performance benefit if you keep the image sizes consistently below 16 kBytes. Teams are advised to consider how the compression setting on the camera affects bandwidth if performing processing on the Driver Station computer, see the [FMS Whitepaper] for more details.

Resolution

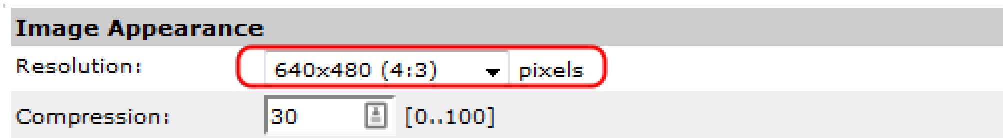


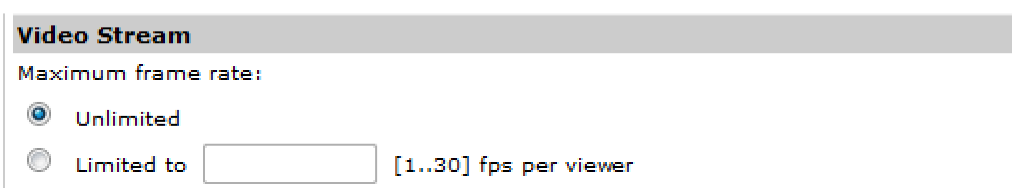
Image sizes shared by the supported cameras are 160x120, 320x240, and 640x480. The M1011 and 1013 have additional sizes, but they aren't built into WPILib. The largest image size has four times as many pixels that are one-fourth the size of the middle size image. The large image has sixteen times as many pixels as the small image.

The tape used on the target is 4 inches wide, and for good processing, you will want that 4 inch feature to be at least two pixels wide. Using the distance equations above, we can see that a medium size image should be fine up to the point where the field of view is around 640 inches, a little over 53 feet, which is nearly double the width of the FRC field. This occurs at around 60 feet away, longer than the length of the field. The small image size should be usable for processing to a distance of about 30 feet or a little over mid-field.

Image size also impacts the time to decode and to process. Smaller images will be roughly four times faster than the next size up. If the robot or target is moving, it is quite important to minimize image processing time since this will add to the delay between the target location and perceived location. If both robot and target are stationary, processing time is typically less important.

Note: When requesting images using LabVIEW (either the Dashboard or Robot Code), the resolution and Frame Rate settings of the camera will be ignored. The LabVIEW code specifies the framerate and resolution as part of the stream request (this does not change the settings stored in the camera, it overrides that setting for the specific stream). The SmartDashboard and robot code in C++ or Java will use the resolution and framerate stored in the camera.

Frame Rate

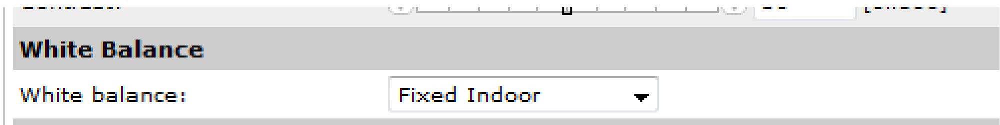


The Axis Cameras have a max framerate of 30 frames per second. If desired, a limit can be set lower to reduce bandwidth consumption.

Color Enable

The Axis cameras typically return color images, but are capable of disabling color and returning a monochrome or grayscale image. The resulting image is a bit smaller in file size, and considerably quicker to decode. If processing is carried out only on the brightness or luminance of the image, and the color of the ring light is not used, this may be a useful technique for increasing the frame rate or lowering the CPU usage.

White Balance

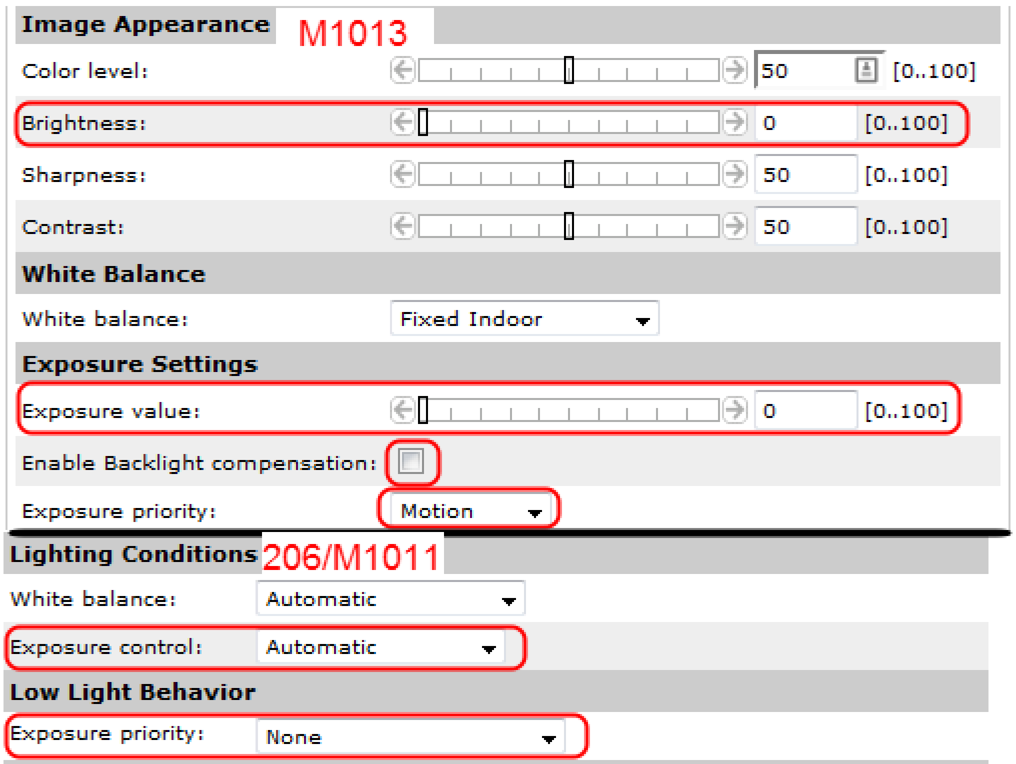


If the color of the light shine is being used to identify the marker, be sure to control the camera settings that affect the image coloring. The most important setting is white balance. It controls how the camera blends the component colors of the sensor in order to produce an image that matches the color processing of the human brain. The camera has five or six named presets, an auto setting that constantly adapts to the environment, and a hold setting -- for custom calibration.

The easiest approach is to use a named preset, one that maintains the saturation of the target and doesn't introduce problems by tinting neutral objects with the color of the light source.

To custom-calibrate the white balance, place a known neutral object in front of the camera. A sheet of white paper is a reasonable object to start with. Set the white balance setting to auto, wait for the camera to update its filters (ten seconds or so), and switch the white balance to hold.

Exposure



The brightness or exposure of the image also has an impact on the colors being reported. The issue is that as overall brightness increases, color saturation will start to drop. Lets look at an example to see how this occurs. A saturated red object placed in front of the camera will return an RGB measurement high in red and low in the other two e.g. (220, 20, 30). As overall white lighting increases, the RGB value increases to (240, 40, 50), then (255, 80, 90), then (255, 120, 130), and then (255, 160, 170). Once the red component is maximized, additional light can only increase the blue and green, and acts to dilute the measured color and lower the saturation. If the point is to identify the red object, it is useful to adjust the exposure to avoid diluting your principle color. The desired image will look somewhat dark except for the colored shine.

There are two approaches to control camera exposure times. One is to allow the camera to compute the exposure settings automatically, based on its sensors, and then adjust the camera's brightness setting to a small number to lower the exposure time. The brightness setting acts similar to the exposure compensation setting on SLR cameras. The other approach is to calibrate the camera to use a custom exposure setting. To do this on a 206 or M1011, change the exposure setting to auto, expose the camera to bright lights so that it computes a short exposure, and then change the exposure setting to hold. Both approaches will result in an overall dark image with bright saturated target colors that stand out from the background and are easier to mask.

The M1013 exposure settings look a little different. The Enable Backlight compensation option is similar to the Auto exposure settings of the M1011 and 206 and you will usually want to un-check this box. Adjust the Brightness and Exposure value sliders until your image looks as desired. The Exposure Priority should generally be set to Motion. This will prioritize framerate over image quality. Note that even with these settings the M1013 camera still performs some auto exposure compensation so it is recommended to check calibration frequently to minimize any impact lighting changes may have on image processing. See the article on Calibration for more details.

Identifying and Processing the Targets

Once an image is captured, the next step is to identify Vision Target(s) in the image. This document will walk through one approach to identifying the 2016 targets. Note that the images used in this section were taken with the camera intentionally set to underexpose the images, producing very dark images with the exception of the lit targets, see the section on Camera Settings for details.

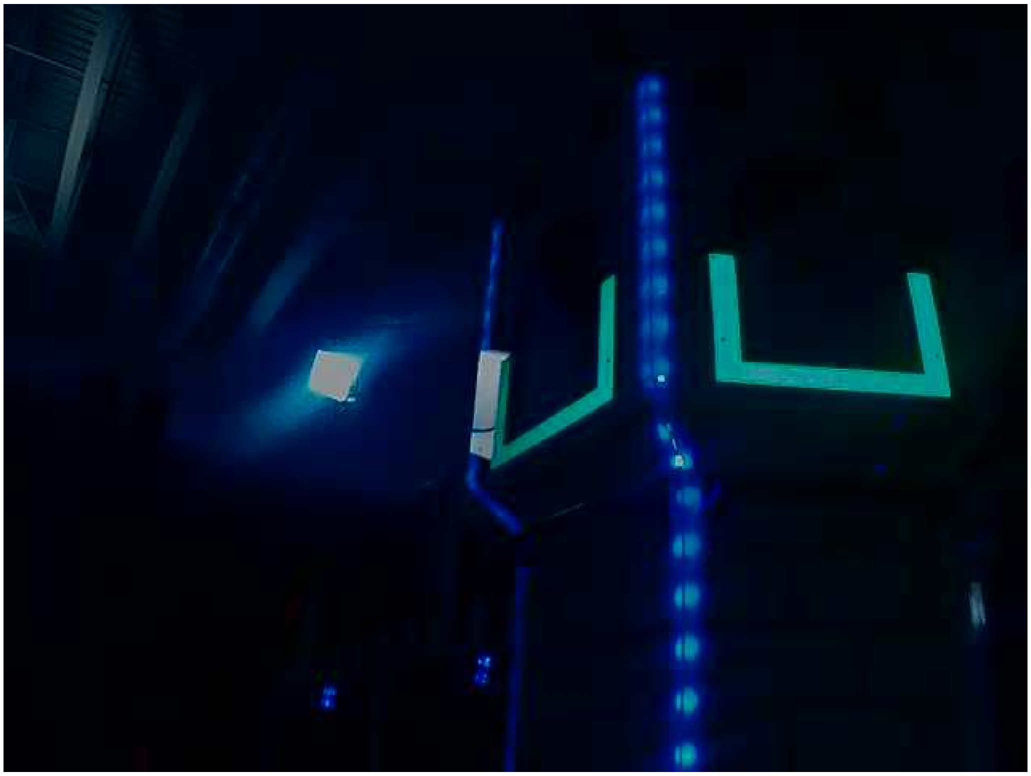
Additional Options

This document walks through the approach used by the example code provided in LabVIEW (for PC or roboRIO), C++ and Java. In addition to these options teams should be aware of the following alternatives that allow for vision processing on the Driver Station PC or an on-board PC:

1. RoboRealm
2. SmartDashboard Camera Extension (programmed in Java, works with any robot language)
3. GRIP

Original Image

The image shown below is the starting image for the example described here. The image was taken using the green ring light available in FIRST Choice. combined with an additional ring light of a different size. Additional sample images are provided with the vision code examples.



What is HSL/HSV?

The Hue or tone of the color is commonly seen on the artist's color wheel and contains the colors of the rainbow Red, Orange, Yellow, Green, Blue, Indigo, and Violet. The hue is specified using a radial angle on the wheel, but in imaging the circle typically contains only 256 units, starting with red at zero, cycling through the rainbow, and wrapping back to red at the upper end. Saturation of a color specifies amount of color, or the ratio of the hue color to a shade of gray. Higher ratio means more colorful, less gray. Zero saturation has no hue and is completely gray. Luminance or Value indicates the shade of gray that the hue is blended with. Black is 0 and white is 255.

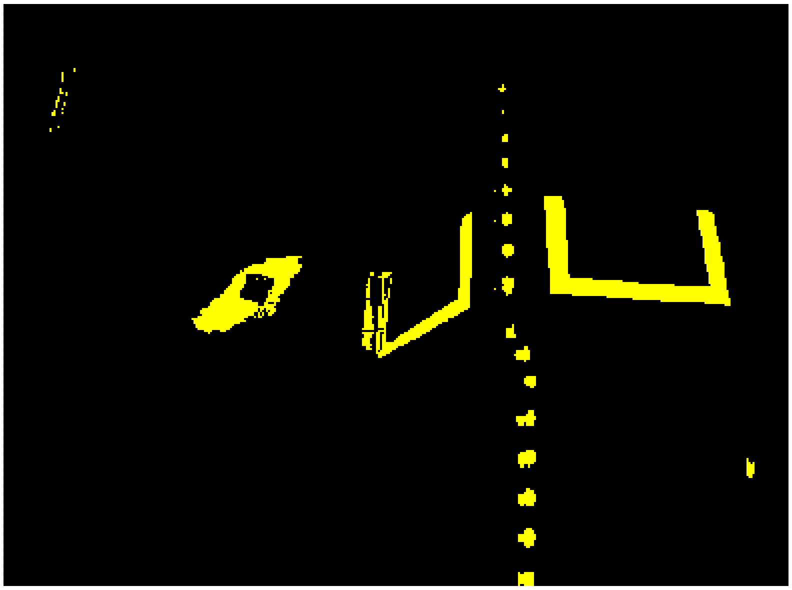
The example code uses the HSV color space to specify the color of the target. The primary reason is that it readily allows for using the brightness of the targets relative to the rest of the image as a filtering criteria by using the Value (HSV) or Luminance (HSL) component. Another reason to use the HSV color system is that the thresholding operation used in the example runs more efficiently on the roboRIO when done in the HSV color space.

Masking

In this initial step, pixel values are compared to constant color or brightness values to create a binary mask shown below in yellow. This single step eliminates most of the pixels that are not part of a target's retro-reflective tape. Color based masking works well provided the color is relatively saturated, bright, and consistent. Color inequalities are generally more accurate when specified using the HSL (Hue, Saturation, and Luminance) or HSV (Hue, Saturation, and Value) color space than the RGB (Red, Green, and Blue) space. This is especially true when the color range is quite large in one or more dimension.

Notice that in addition to the target, other bright parts of the image (overhead light and tower lighting) are also caught by the masking step.

Teams may find it more computationally efficient, though potentially less robust, to filter based on only a single criteria such as Hue or Value/Luminance.



Particle Analysis

After the masking operation, a particle report operation is used to examine the area, bounding rectangle, and equivalent rectangle for the particles. These are used to compute several scored terms to help pick the shapes that are most rectangular. Each test described below generates a score (0-100) which is then compared to pre-defined score limits to decide if the particle is a target or not.

Coverage Area

The Area score is calculated by comparing the area of the particle compared to the area of the bounding box drawn around the particle. The area of the retroreflective strips is 80 square inches.

The area of the rectangle that contains the target is 240 square inches. This means that the ideal ratio between area and bounding box area is 1/3. Area ratios close to 1/3 will produce a score near 100, as the ratio diverges from 1/3 the score will approach 0.

Aspect Ratio

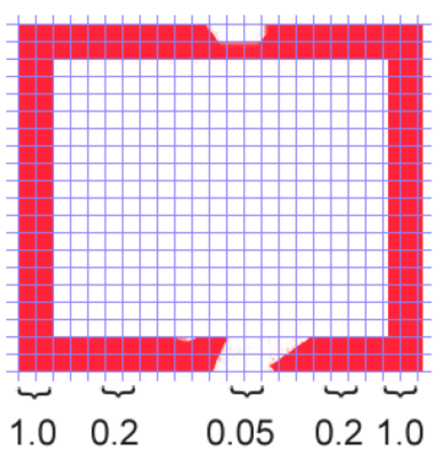
The aspect ratio score is based on (Particle Width / Particle Height). The width and height of the particle are determined using something called the "equivalent rectangle". The equivalent rectangle is the rectangle with side lengths x and y where 2x+2y equals the particle perimeter and x*y equals the particle area. The equivalent rectangle is used for the aspect ratio calculation as it is less affected by skewing of the rectangle than using the bounding box. When using the bounding box rectangle for aspect ratio, as the rectangle is skewed the height increases and the width decreases.

The target is 20" wide by 12" tall, for a ratio of 1.6. The detected aspect ratio is compared to this ideal ratio. The aspect ratio score is normalized to return 100 when the ratio matches the target ratio and drops linearly as the ratio varies below or above.

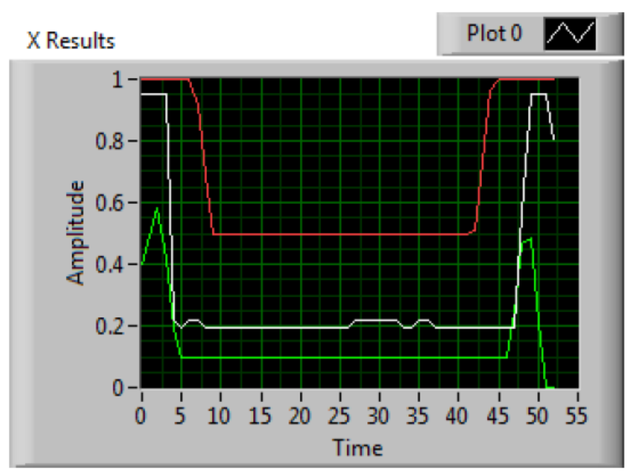
Moment

The moment measurement calculates the particles moment of inertia about it's center of mass. This measurement provides a representation of the pixel distribution in the particle. The ideal score for this test is ~0.28. [Moment of Inertia]

X/Y Profiles



Column averages for a particle rectangle.



White line is the average, red is upper limit, and green is lower limit..

The edge score describes whether the particle matches the appropriate profile in both the X and Y directions. As shown, it is calculated using the row and column averages across the bounding box extracted from the original image and comparing that to a profile mask. The score ranges from 0 to 100 based on the number of values within the row or column averages that are between the upper and lower limit values.

Measurements

If a particle scores well enough to be considered a target, it makes sense to calculate some real-world measurements such as position and distance. The example code includes these basic measurements, so let's look at the math involved to better understand it.

Position

The target position is well described by both the particle and the bounding box, but all coordinates are in pixels with 0,0 being at the top left of the screen and the right and bottom edges determined by the camera resolution. This is a useful system for pixel math, but not nearly as useful for driving a robot; so let's change it to something that may be more useful.

To convert a point from the pixel system to the aiming system, we can use the formula shown below.

The resulting coordinates are close to what you may want, but the Y axis is inverted. This could be corrected by multiplying the point by [1,-1] (Note: this is not done in the sample code). This coordinate system is useful because it has a centered origin and the scale is similar to joystick outputs and RobotDrive inputs.

Vision Processing

$$A_{x,y} = (P_{x,y} - \frac{resolution_{x,y}}{2}) / \frac{resolution_{x,y}}{2}$$

Pixel Coordinate System – $P_{x,y}$

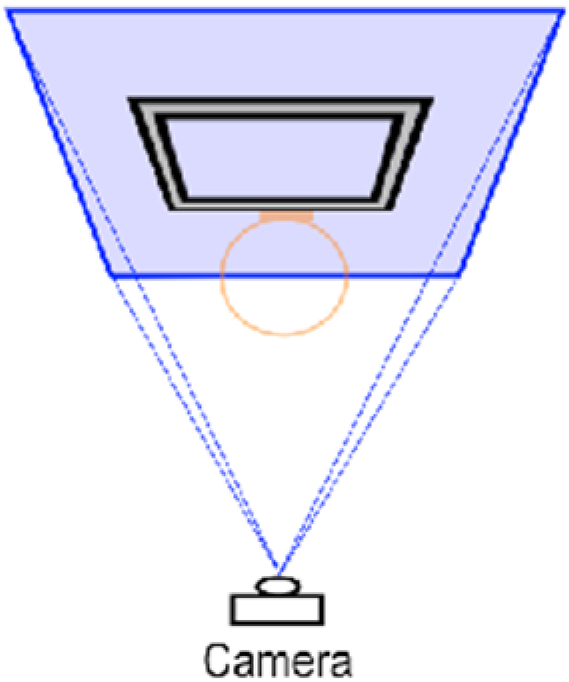
Aiming Coordinate System – $A_{x,y}$

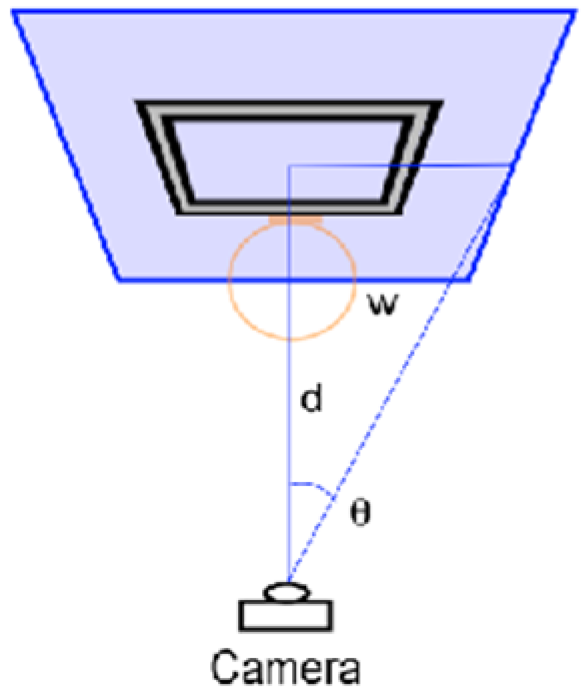
Distance

The target distance is computed with knowledge about the target size and the camera optics. The approach uses information about the camera lens view angle and the width of the camera field of view. Shown below-left, a given camera takes in light within the blue pyramid extending from the focal point of the lens. Unless the lens is modified, the view angle is constant and equal to 2Θ. As shown to the right, the values are related through the trigonometric relationship of ...

tanΘ = w/d

The datasheets for the cameras can be found at the following URLs:[Axis 206,]() [AxisM1011,]() [Axis M1013,]() [Lifecam HD3000](). These give rough horizontal view angles for the lenses. Remember that this is for entire field of view, and is therefore 2Θ. This year's code uses the vertical field-of-view and it is therefore highly recommend to perform calibration (as described in the next article) to determine the appropriate view angle for your camera (empirically determined values for each camera type are included in the code as a reference).





Distance Continued

The next step is to use the information we have about the target to find the width of the field of view the blue rectangle shown above. This is possible because we know the target rectangle size in both pixels and feet, and we know the FOV rectangle width in pixels. We can use the relationships of ...

Tft/Tpixel = FOVft/FOVpixel and FOVft = 2*w = 2*d*tanΘ

to create an equation to solve for d, the distance from the target:

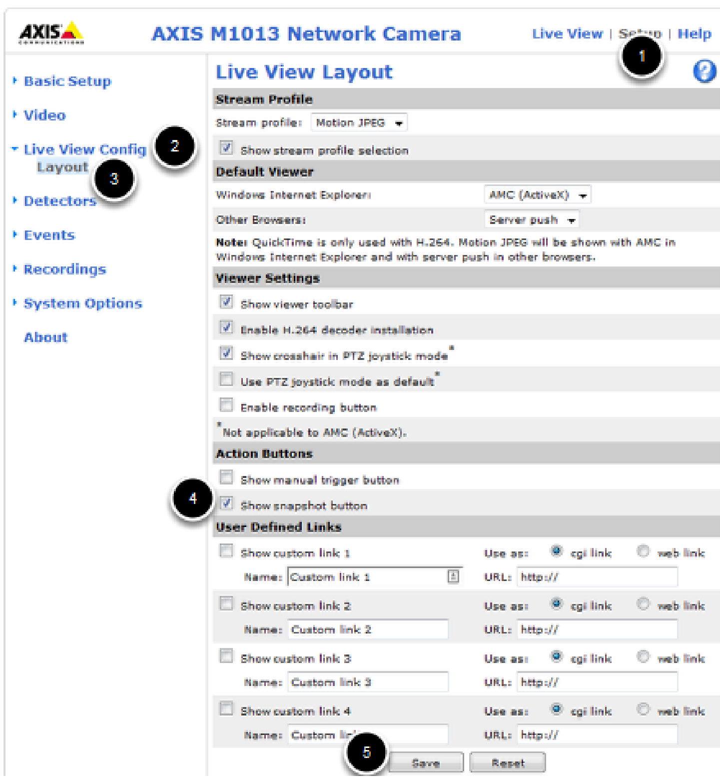
d = Tft*FOVpixel/(2*Tpixel*tanΘ)

Notice that the datasheets give approximate view angle information. When testing, it was found that the calculated distance to the target tended to be a bit short. Using a tape measure to measure the distance and treating the angle as the unknown it was found that view angles of 41.7˚ for the 206, 37.4˚ for the M1011, and 49˚ for the M1013 gave better results. Information on performing your own distance calibration is included in the next article.

Calibration

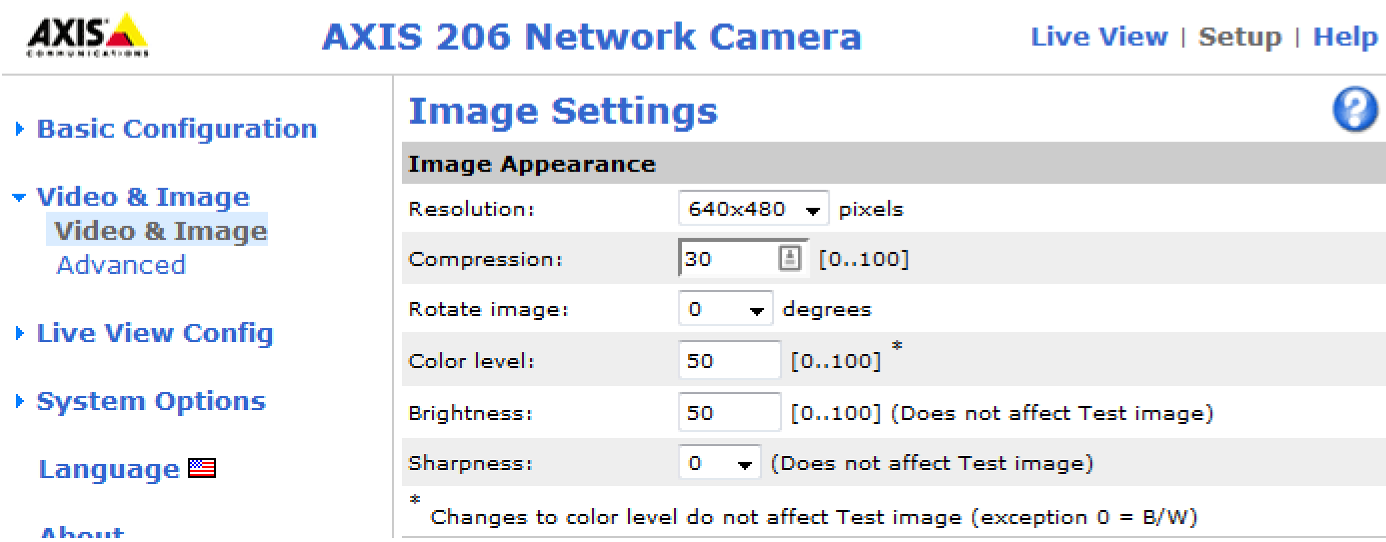
While many of the numbers for the Vision Processing code can be determined theoretically, there are a few parameters that are typically best to measure empirically then enter back into the code (a process typically known as calibration). This article will show how to perform calibration for the Color (masking), and View Angle (distance) using the NI Vision Assistant. If you are using C++ or Java and have not yet installed the NI Vision Assistant, see the article Installing NI Vision Assistant.

Enable Snapshots



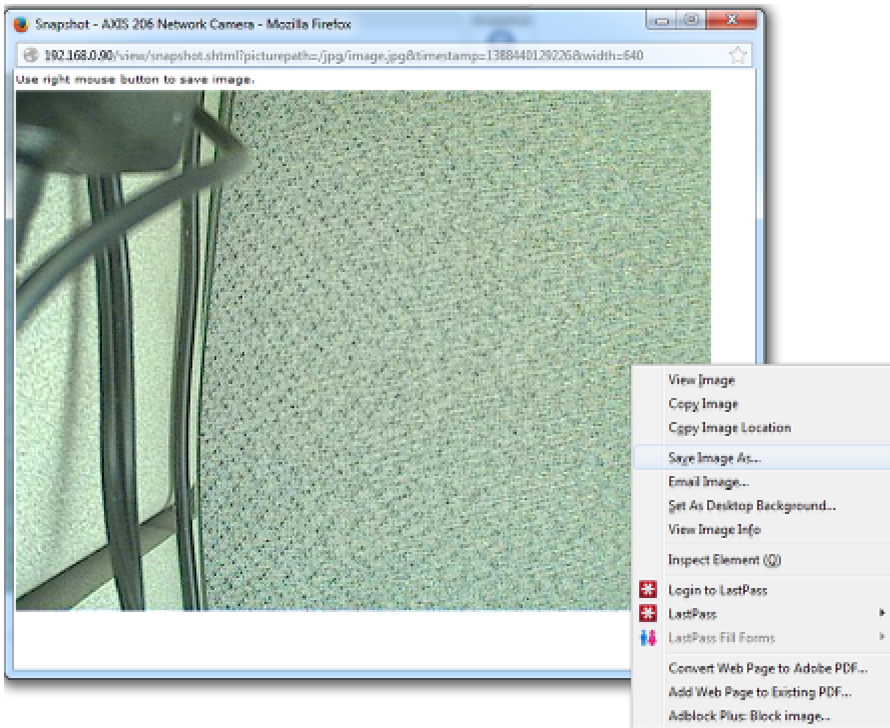
To capture snapshots from the Axis camera, you must first enable the Snapshot button. Open a web-browser and browse to camera's address (10.TE.AM.11), enter the Username/Password combo FRC/FRC if prompted, then click Setup->Live View Config->Layout. Click on the checkbox to Show snapshot button then click Save.

Check Camera Settings



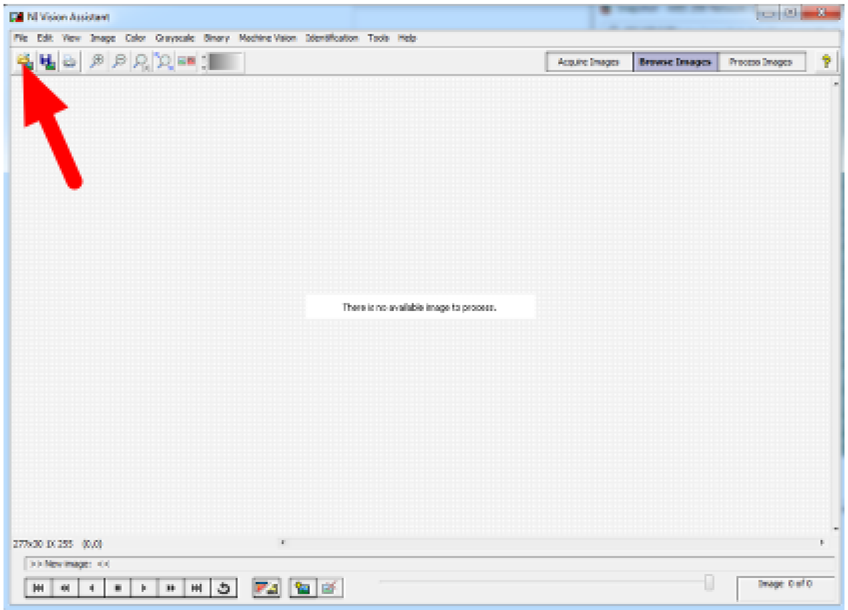
Depending on how you are capturing the image stream in your program, it may be possible to stream a different resolution, framerate and/or compression than what is saved in the camera and used in the Live View. Before performing any calibration it is recommended you verify that the settings in the camera match the settings in your code. To check the settings in the camera, click on the Video and Image header on the left side of the screen, then click Video and Image.

Capture Images



Click the Live View button to return to the Live View page and you should now see a Snapshot button. Clicking this button opens a pop-up window with a static image capture. Right-click on this image, select Save Image as and select your desired location and file name, then save the image.

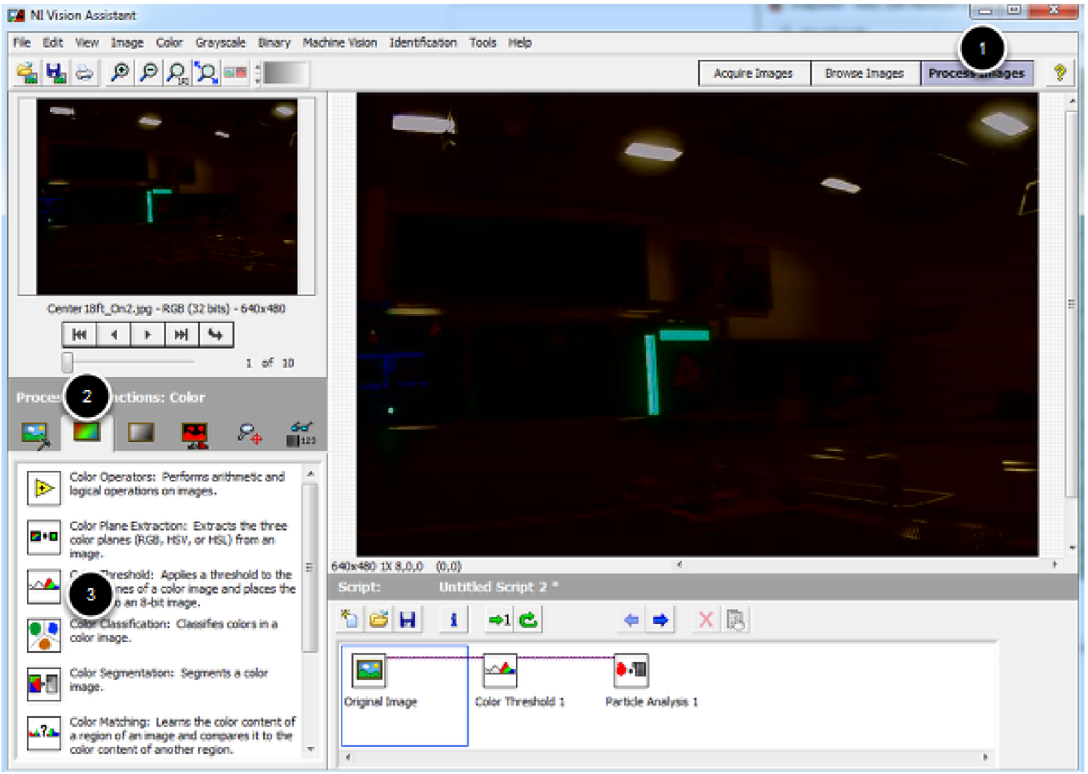
Load Image(s) in Vision Assistant



Open the NI Vision Assistant and select the Browse Images option. Select the Open Images icon in the top left of the Toolbar, then locate your images. Repeat as necessary to load all desired images.

Vision Processing

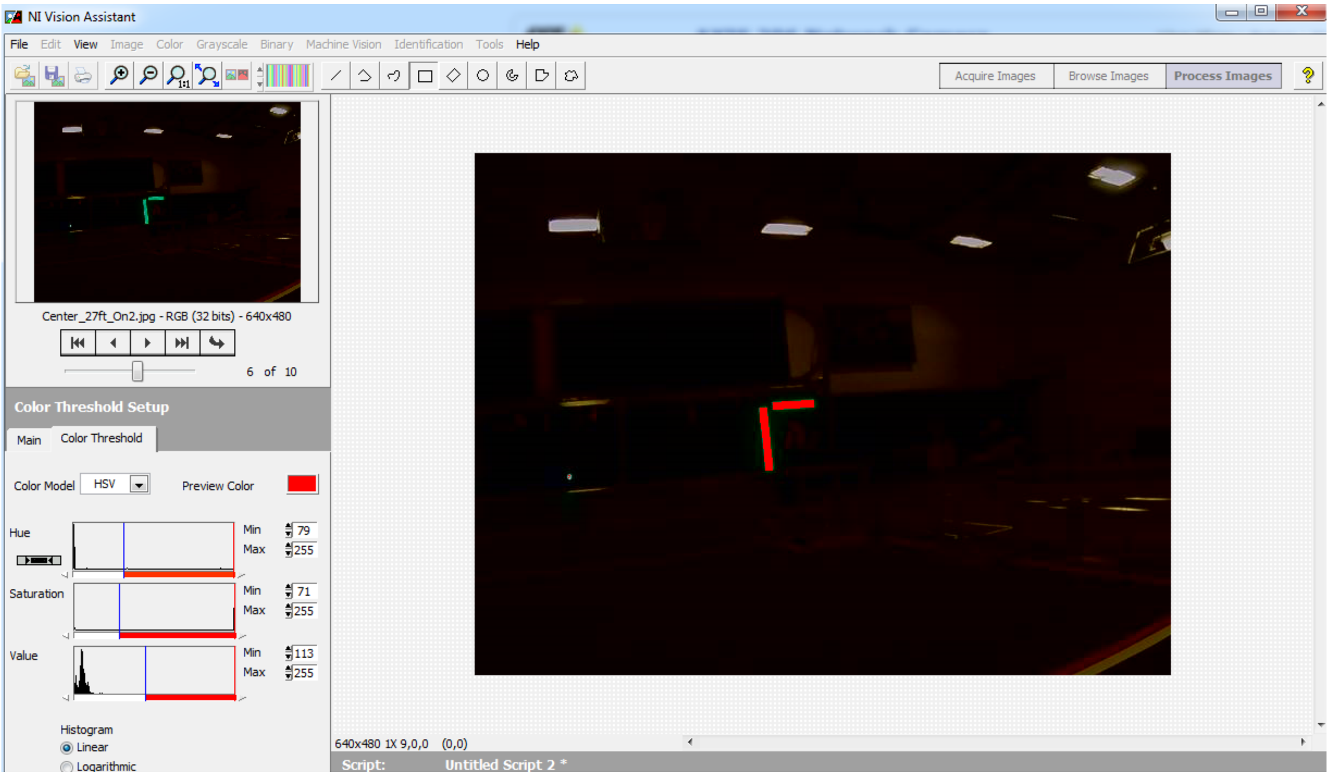
Color Threshold



Click Process Images in the top right, then select the color tab on the bottom right and click the Color Threshold icon.

Vision Processing

HSV Calibration



Change the Color Model dropdown to HSV. Next tune the window on each of the three values to cover as much of the target as possible while filtering everything else. If using a green light, you may want to use the values in the sample code as a starting point. If you have multiple images you can use the controls in the top left to cycle through them. Use the center two arrow controls or the slider to change the preview image in the top left window, then click the right-most arrow to make it the active image. When you are happy with the values you have selected, note down the ranges for the Hue, Saturation and Value. You will need to enter these into the appropriate place in the vision code. Click OK to finish adding the step to the script.

You may wish to take some new sample images using the time for camera calibration at your event to verify or tweak your ranges slightly based on the venue lighting conditions.

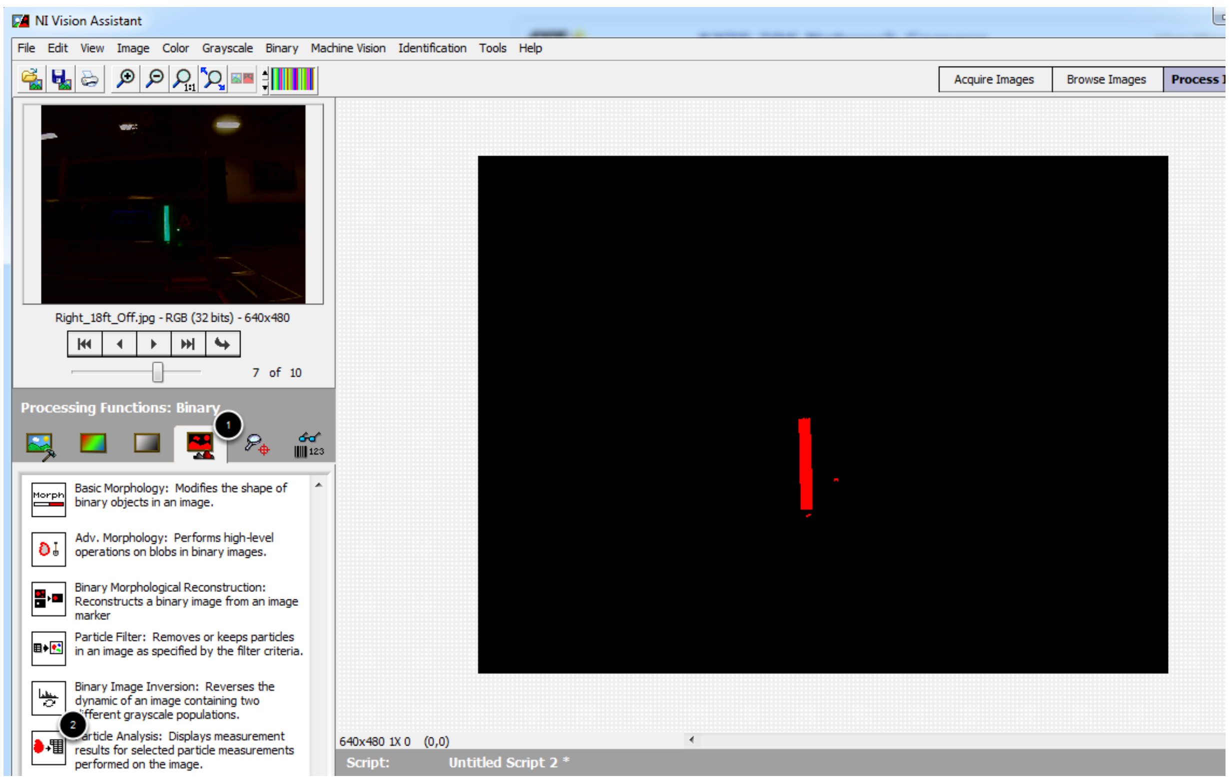
View Angle/Distance Calibration

While a theoretical view angle for each camera model can be found in the datasheet, empirical testing has found that these numbers may be a bit off even for the horizontal view angle. Given that this year's code uses the vertical field-of-view it is best to perform your own calibration for your camera (though empirical values for each camera type are included in the code as a reference). To do this set up an equation where the view angle, Θ, is the only unknown. To do this, utilize a target of known size at a known distance, leaving the view angle as the only unknown. Let's take our equation from the previous article, $d = Tft*FOVpixel/(Tpixel*\tan\Theta)$, and re-arrange it to solve for Θ:

$$\tan\Theta = Tft*FOVpixel/(Tpixel*d)$$

$$\Theta = \arctan(Tft*FOVpixel/(Tpixel*d))$$

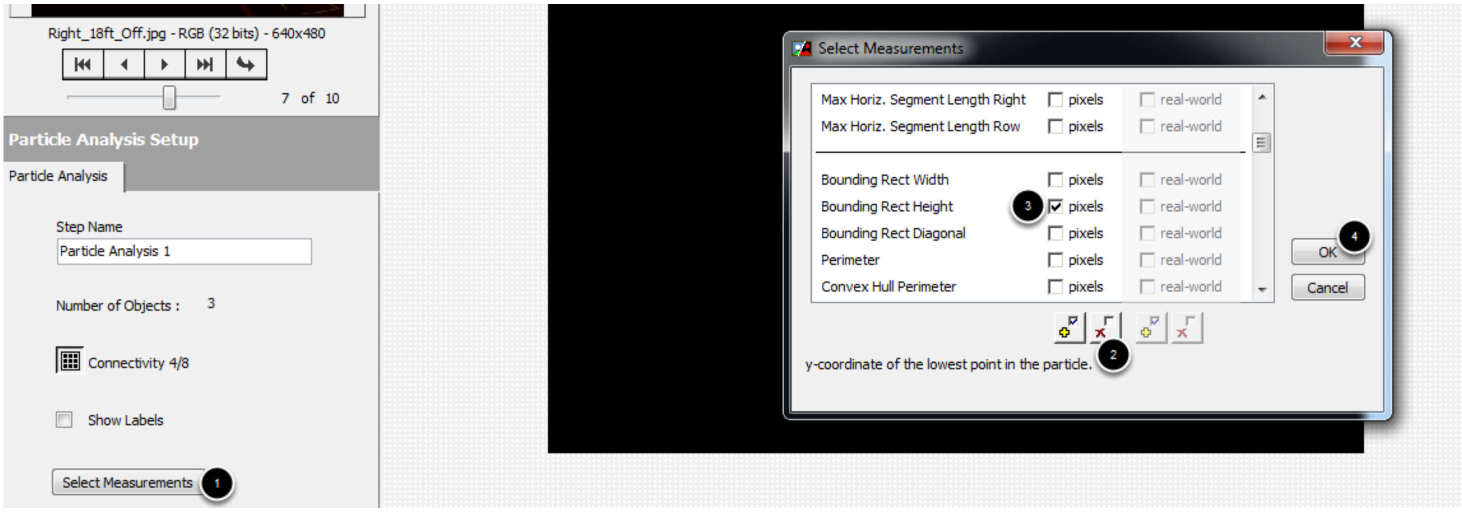
Taking measurements



One way to take the required measurements is to use the same images of the retro-reflective tape that were used for the color calibration above. We can use Vision Assistant to provide the height of the detected blob in pixels. By measuring the real-world distance between the camera and the target, we now have all of the variables to solve our equation for the view angle.

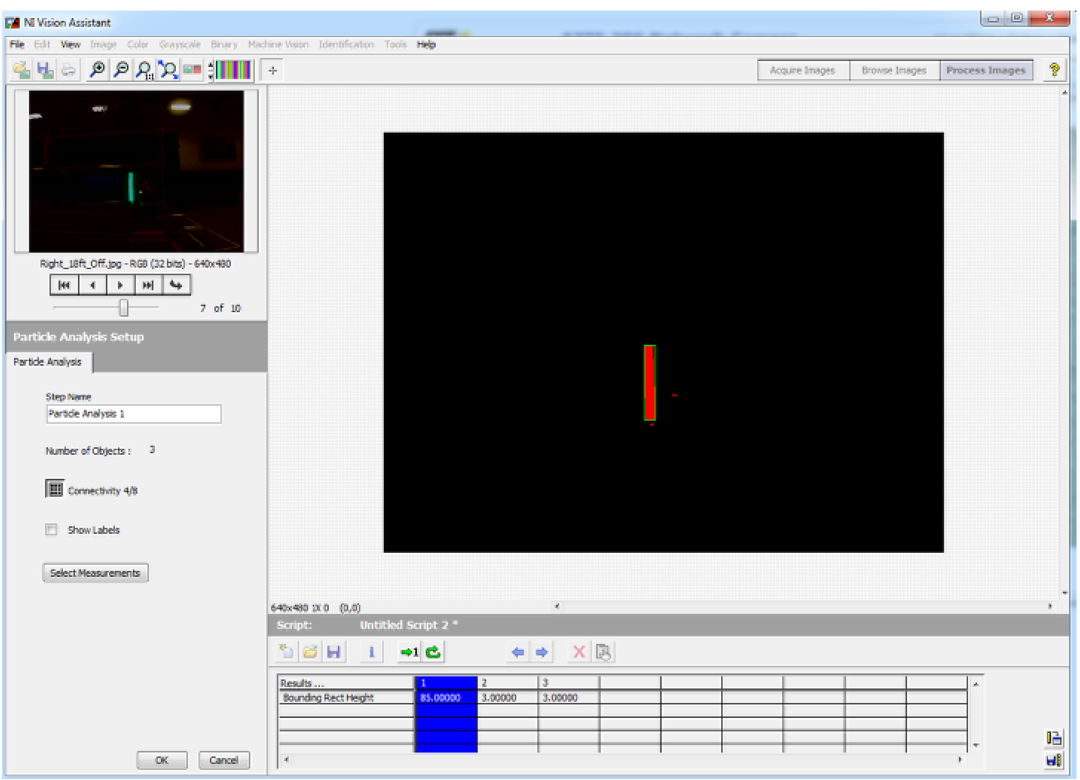
To measure the particles in the image, click the Binary tab, then click the Particle Analysis icon.

Selecting Measurements



Click on the Select Measurements button. In this case, we are only interested in the bounding box height. Click on the button with the X to deselect all measurements, then locate the Bounding Rect Height measurement and check the box. Click OK to save.

Measuring the Particle



The measurements for each particle will now be displayed in the window at the bottom of the screen. If your image has multiple particles, you can click in each box to have Vision Assistant highlight the particle so you can make sure you have the right one. This article will show the calculation using a single image, but you may wish to perform the calculation on multiple images from multiple distances and use a technique such as averaging or least squares fit to determine the appropriate value for the View angle. You can use the same arrow controls described in the color section above to change the active image.

Calculation

As seen in the previous step, the particle representing the 32in tall vertical target in this example measured 85 pixels tall in a 640x480 image. The image shown was taken from (very roughly) 18 ft. away. Plugging these numbers into the equation from above....

$\Theta = \arctan(2.66*480/(2*85*18)) = 22.65$ degrees

Depending on what you use to calculate the arctangent, your answer may be in radians, make sure to convert back to degrees if entering directly into the sample code as the view angle.

Note: The code uses **View Angle** and we just calculated Θ. Make sure to multiply Θ by 2 if replacing the constants in the code.Multiplying our result by 2 yields 45.3 degrees. This image is from a M1013 camera, so our value is a bit off from the previously measured 29.1 but given that the 18ft. was a very rough measurement this shows that we are in the ballpark and likely performed the calculation correctly.

Axis M1013 Camera Compatibility

It has come to our attention that the Axis M1011 camera has been discontinued and superseded by the Axis M1013 camera. This document details any differences or issues we are aware of between the two cameras when used with WPILib and the provided sample vision programs.

Optical Differences

The Axis M1013 camera has a few major optical differences from the M1011 camera:

1. The M1013 is an adjustable focus camera. Make sure to focus your M1013 camera by turning the grey and black lens housing to make sure you have a clear image at your desired viewing distance.
2. The M1013 has a wider view angle (67 degrees) compared to the M1011 (47 degrees). This means that for a feature of a fixed size, the image of that feature will span a smaller number of pixels

Using the M1013 With WPILib

The M1013 camera has been tested with all of the available WPILib parameters and the following performance exceptions were noted:

1. The M1013 does not support the 160x120 resolution. Requesting a stream of this resolution will result in no images being returned or displayed.
2. The M1013 does not appear to work with the Color Enable parameter exposed by WPILib. Regardless of the setting of this parameter a full color image was returned.

All other WPILib camera parameters worked as expected. If any issues not noted here are discovered, please file a bug report on the WPILib tracker (note that you will need to create an account if you do not have one, but you do not need to be a member of the project).

Using the Axis Camera at Single Network Events

The 2015 convention for using the Axis camera uses mDNS with the camera name set to axis-camera.local At home this works fine as there is only one camera on the network. At official events, this works fine as each team is on their own VLAN and therefore doesn't have visibility to another team's camera. At an offseason using a single network, this will cause an issue where all teams will connect to whichever team's camera "wins" and becomes "axis-camera", the other cameras will see that the name is taken and use an alternative name. This article describes how to modify the Dashboard and/or robot code to use a different mDNS name to separate the camera streams.

Changing the camera mDNS name

To change the mDNS name in the camera, follow the instructions in Configuring an Axis Camera but substitute the new name such as axis-cameraTEAM where TEAM is your team number.

Viewing the camera on the DS PC - Browser or Java SmartDashboard

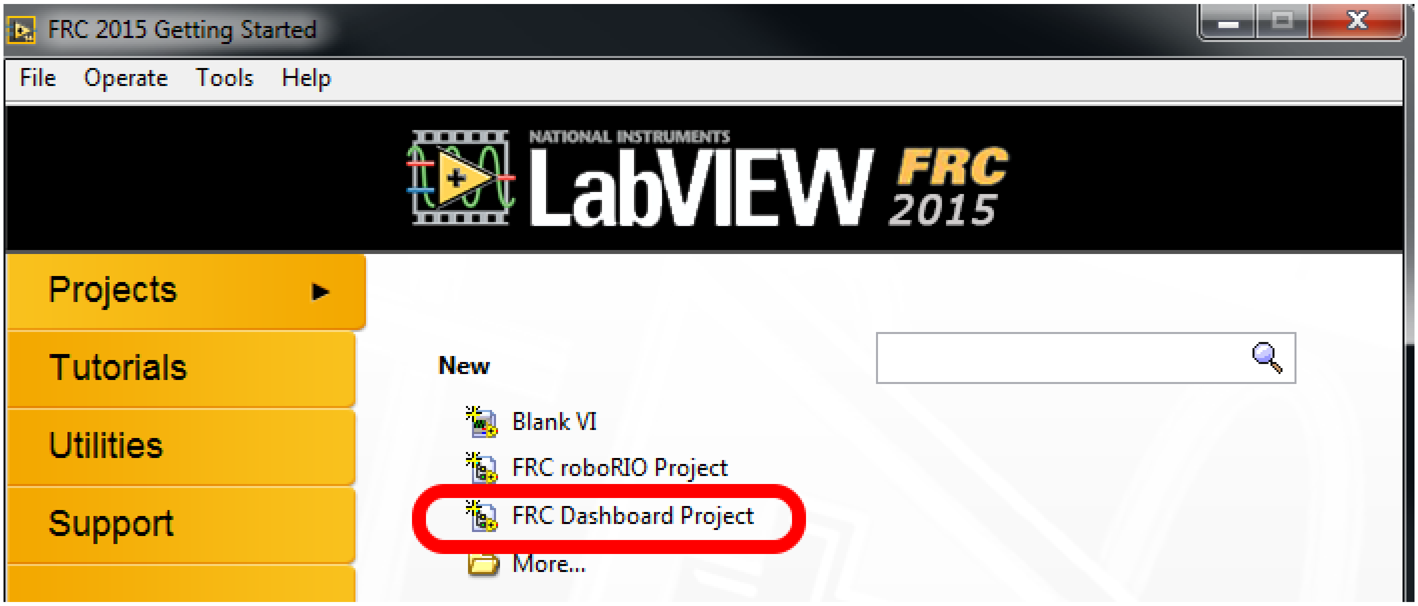
If you are using a web-browser or the updated Java SmartDashboard (which accepts mDNS names for the Simple Camera Viewer widget), updating to use the new mDNS name is simple. Simply change the URL in the browser or the address in the Simple Camera Viewer widget properties to the new mDNS name and you are all set.

Viewing the camera on the DS PC - LabVIEW Dashboard

To view the camera stream in the LabVIEW Dashboard, you will need to build a customized version of the Dashboard. Note that this customized version will only work for the Axis camera and will no longer work for a USB camera, revert to the default Dashboard to use a USB camera.

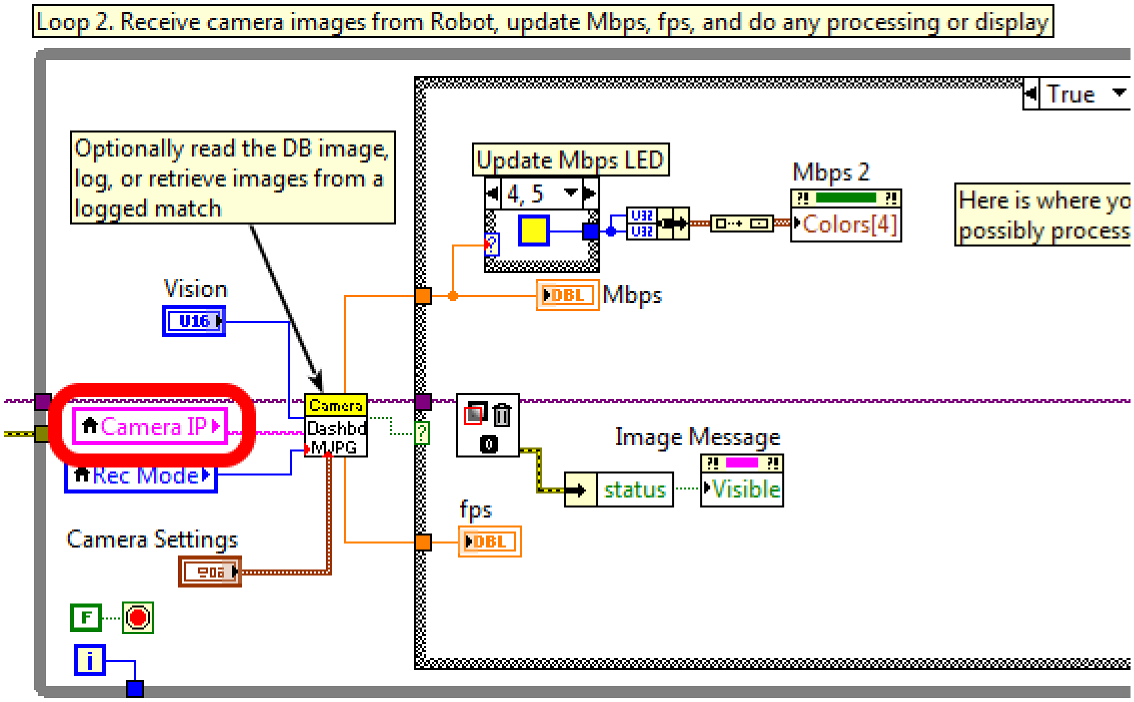
Vision Processing

Creating a Dashboard Project



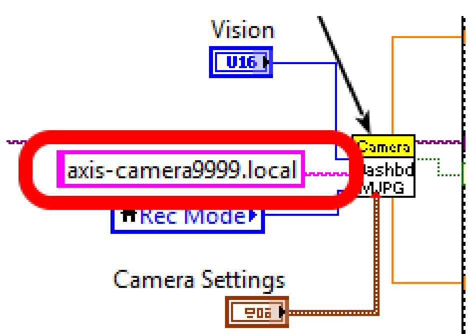
From the LabVIEW Splash screen, select "FRC Dashboard Project". Name the project as desired, then click Finish.

Locating Loop 2 - Camera IP



Double click on Dashboard Main.vi in the project explorer to open it and press Ctrl+e to see the block diagram. Scroll down to the loop with the comment that says Loop 2 and locate the "Camera IP" input.

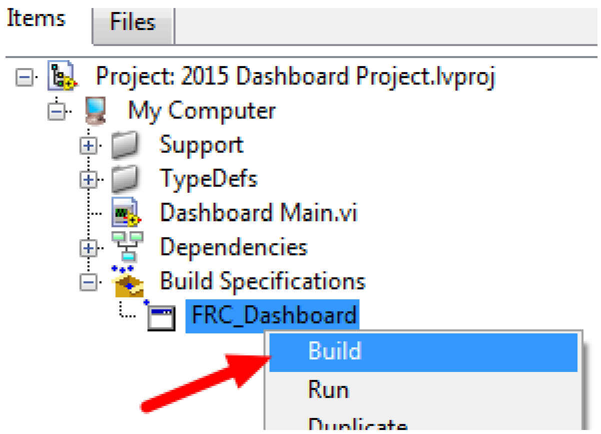
Editing the camera IP



Delete the Camera IP node, right click on the broken wire and click Create Constant (connect the constant to the wire if necessary). In the box, enter the mDNS name of your camera with a ".local" suffix (e.g. "axis-cameraTEAM.local" where TEAM is replaced with your team number). In this example I have used a sample name for team 9999. Then click File->Save or Ctrl+S to save the VI.

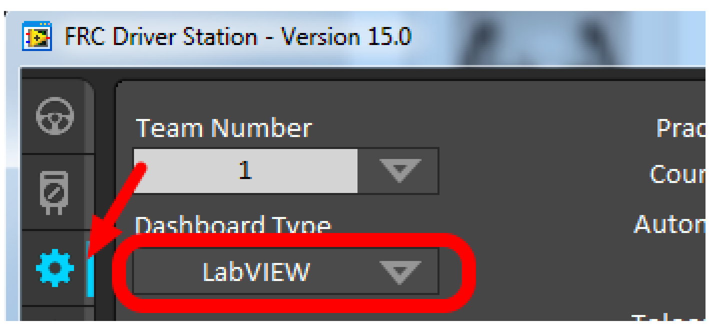
Note: You may also wish to make a minor modification to the Front Panel to verify that you are running the right dashboard later.

Building the Dashboard



To build the new dashboard, expand Build Specifications in the Project Explorer, right click on FRC_Dashboard and select Build.

Setting the Driver Station to launch the modified Dashboard

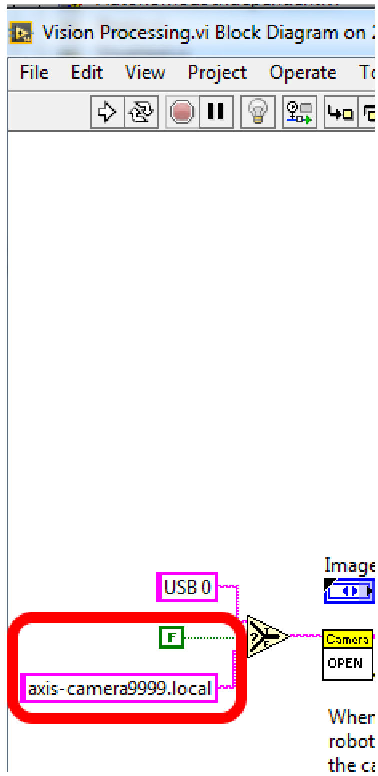


On the Setup tab of the Driver Station, change to dropdown box to LabVIEW to launch your new Dashboard.

Accessing the camera from Robot Code

If you wish to access the renamed camera from your robot code, you will have to modify it as well. In C++ and Java, just change the String used for the camera host name to match the new name. In LabVIEW follow the step below.

Modifying LabVIEW Robot Code



In the Project Explorer, locate Vision Processing.VI and double click to open it. Then press Ctrl+e to open the Block Diagram. Locate the string "axis-camera.local" near the left of the image and replace with "axis-cameraTEAM.local" Also make sure the constant is set to "False" to use the Axis camera instead of USB.