Finding a goal using the camera

Using the line trackers is a reliable method for navigating to the goals, but it only works to navigate to the goals with lines and it might not be easy to acquire the line during the teleop part of the program. Another technique is to use the camera and take advantage of the retroreflective tape on the goal. The retroreflective tape reflects back in exactly the same direction as the light source that illuminated it. If a set of lights are mounted near the camera lens, then the camera would see the reflected light regardless of the angle of the camera to the target.

This is a sample program that uses the existing Java (or C++) camera and image processing classes and methods to get data that can then be used to drive a robot towards a goal.

To make the code work more consistently we connected the camera directly to a laptop to set and save the exposure but shining a bright light directly into the lens. This reduced the sensitivity of the camera and made it much better at seeing the red light from out LEDs mounted next to the camera. Once the sensitivity is set, then the settings can be saved from the camera web page on the laptop.

We determined the threshold values for the red LEDs using the NI Vision Assistant by taking some pictures with the camera, loading them into the Vision Assistant and applying a Color Threshold operation. By double-clicking on the color threshold icon a set of sliders pop up representing the HSL values contained in the image. You can then adjust sliders to get the upper and lower HSL values (6 floating point numbers) to find settings that only detect the color of your light source. The constants in the program below represent the values for our red LEDs, you need to find your own values for your light source. Once you have the values, they can be used in your program.

```
BinaryImage binImage = colorImage.thresholdHSL(242, 255, 36, 255, 25, 255);
```

In Java it is necessary to release the memory used by the objects because they are actually allocated in C++ code that is part of WPILib. Normally Java would take care of garbage collecting no longer used memory, but for performance, the image classes call C++ code that doesn't automatically free the memory.

```
colorImage.free();
binImage.free();
```

The two image operations used in the code are:

1. A color threshold operation returns a binary image (single bit/pixel) with the bit set in the output for each corresponding pixel that is within the specified range.
2. A particle analysis that takes a binary image and finds groups (blobs) of pixels. These correspond to areas of red light.