Report #2: Object Detector and Performance
Diya Agarwal and Ritali Jain
November 16, 2024

1 An Introduction/Overview.

The objective of this detector is to detect a blue ball within a certain color range, display its binary image, and draw a contour around its shape in a BGR (blue green red) image. To do this, we first altered the camera's brightness and exposure settings to see a clearer image. We then created a crosshair on the BGR image, placed the blue ball in the center of the frame, and detected the center pixel's HSV values that correspond to the blue color we desired. When using this detector, the user should expect to see two images: a BGR image and a binary image. In the BGR image, green contours are drawn around any blue object over a certain area in the HSV values we previously specified, while red contours are drawn for smaller objects. In the binary image, the blue ball is depicted in white and the surroundings are black. The color and contour detection works well, but it fails when there are other blue objects in the surroundings that get detected. The contours also fail when we disrupt a continuous blue object, since multiple contours are drawn in this case instead of one singular contour.

2 General Structure and Flowchart.

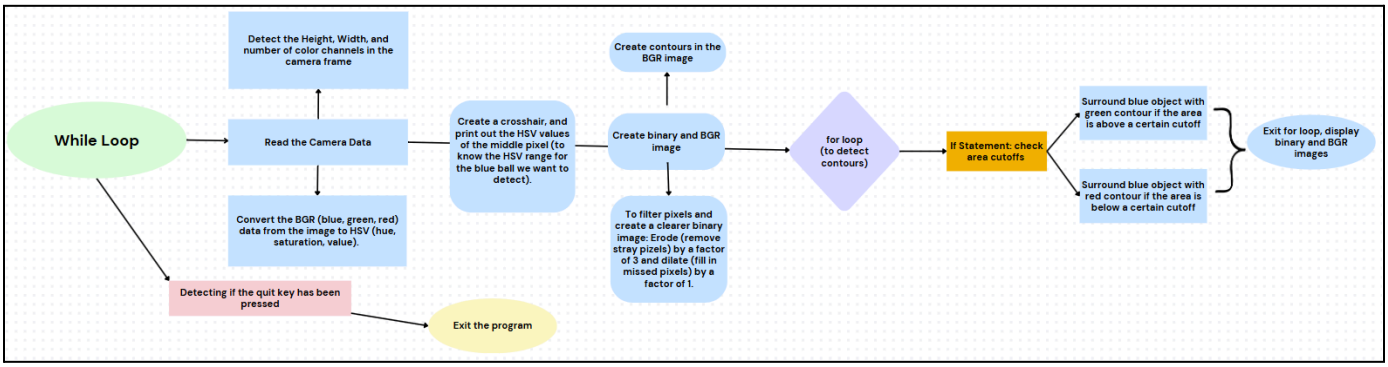

Figure 1: Flowchart of the While Loop for the detector code. The loop detects the blue object, creates a binary image and a BGR image with contours around the object.

3 Detailed Descriptions and Reasons for Choices

3.1 Camera Settings.

Our final camera settings were as follows:

Contrast: 128
Brightness: 154
Saturation: 210
WBalance: 3273
Exposure: 235
Focus: 0

We chose these parameters as the image with the default camera settings was washed out, and in order to improve detection, we wanted the colors to have stronger contrast and pop out. As a result, we increased saturation, exposure, and white balance until the blue color of the ball was distinct from the remainder of the background scene. Because we were using blue, we used a WBalance value on the warmer side (3200 is a warmer value on a scale from 0-5000) because the lighting of the room was warmer and we wanted to neutralize the image and balance out the lighting. We also increased brightness due to the dim lighting of the room to make the foreground more distinct from objects in the background.

Choosing lower exposure would have potentially resulted in shadows being detected since the hue would have been similar due to low contrast.

3.2 Color Isolation.

We were detecting a blue object, and we obtained the following HSV codes for that:

H: 75-115
S: 115-230
V: 50-190

False objects that we were unable to reject were all blue (very similar hue to the ball itself), including extraneous objects in a cabinet in the background, as well as the sleeve of someone wearing blue. We are unable to remove these objects by changing saturation and values because we need our detector to be robust to different conditions on the ball itself. If we chose a different HSV range, we may have gotten different shades of blue or potentially different colors altogether if the H range was sufficiently wider.

3.3 Pixel-Level Filtering

We filtered the image by first eroding by 3 iterations and then dilating by 1 iteration. We determined these operations through trial-and-error; we first tried eroding the image down to remove extraneous pixels (i.e. objects being detected in the distant background) followed by a dilation to fill in the pixels eroded inside the ball. Below is the noisy image, followed by a filtered version:

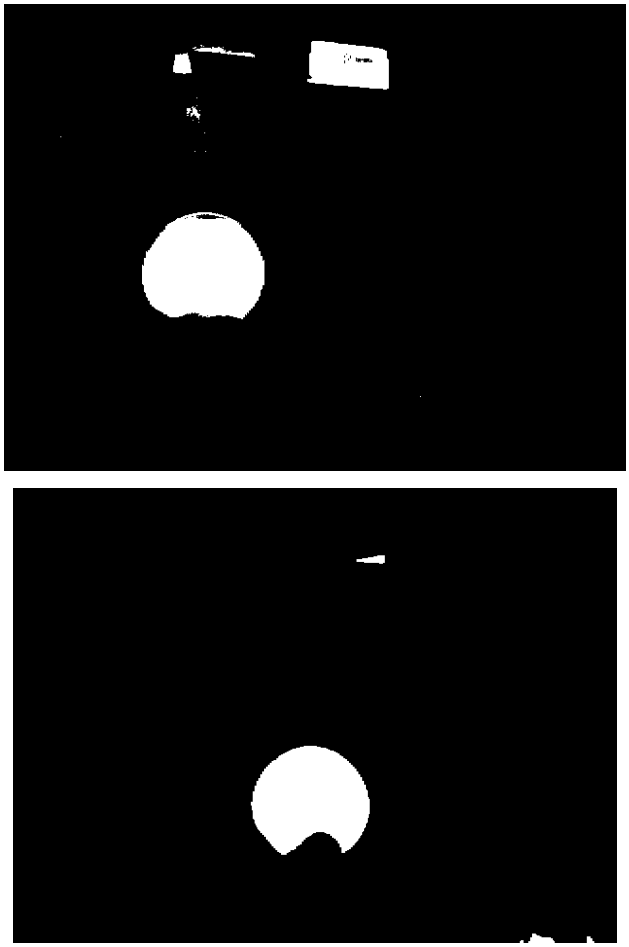

Figure 2: Before and After Pixel Filtering. The top image is the binary image without filtering. The bottom image is after filtering, showing a much cleaner circle (representing the ball).

3.4 Object-Level Down-Selection and Centroid.

We limited the valid contours using a minimum cutoff threshold. That is, the area of the contour had to be at least 1000 pixels in order for the contour to be selected.

We tested two different methods to process the contours into objects and determined their center coordinate locations (i.e. x/y locations). One method was to compute the contour centroid, whereas the other method was to fit an ellipse to the contour and determine the center of the ellipse. Due to our object being round, we noticed that the ellipse method worked

better as it would detect the entire ball even if partially obstructed at an edge. This performance is of course dependent on the assumption that the object is round.

4 User Display

We colored the accepted contours in green (accepted contours are above a certain area cutoff) and we selected contours within the HSV ranges but not meeting the area cutoff as red. We also colored the ellipse-fitting in yellow to differentiate it from the contour selection in green.

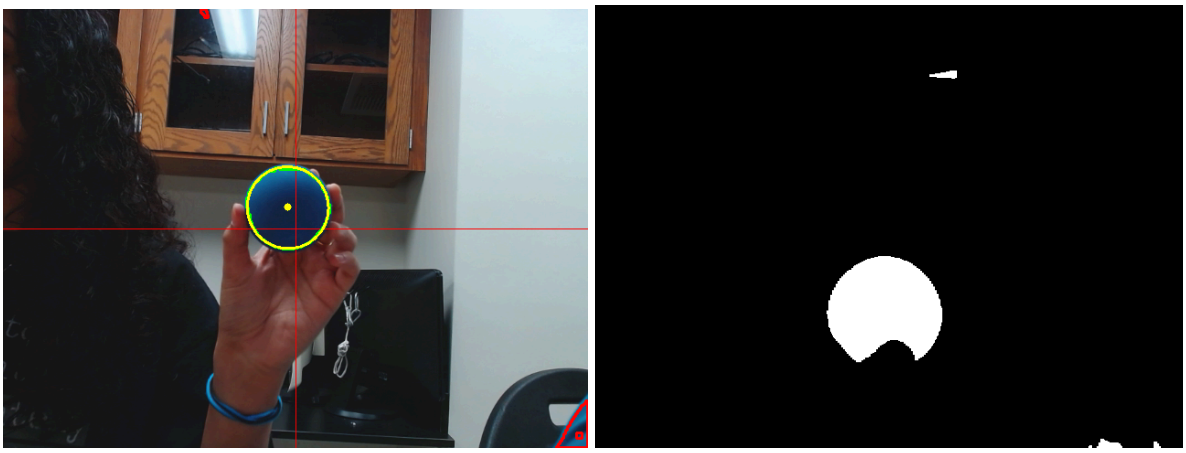

Figure 3: The BGR Image (left) displayed with the ellipse fitting in yellow, the centroid in yellow, and the contour fitting in green. The binary image (right), with the blue ball displayed as a white circle.

5 Performance, Limitations, and Possible Improvements.

We tested different conditions to understand the limitations of our detector. We had someone wear blue in front of the camera and different parts of their clothing were selected as contours, indicating false positives.

We also shined light at the center of the ball, which caused the rim of the ball to be detected by the contour, but not the entirety of the ball (i.e. the ball was not obstructed, but changing lighting conditions on part of the ball caused a visible part of the ball to be ignored). We also got false negatives when we did indeed obstruct the ball with a finger, as the contour of the ball was split into two individual contours which do not meet the maximum area cutoff. Hence, no parts of the ball were selected.

We could have improved the rejection of the contours. Currently, we used a maximum cutoff area to classify objects, but this primarily cuts off items that are too far and negligible to detect. Since we are detecting a round ball, it would make sense to look at the proportions of the contour itself to narrow down the selected contours. One way to do this is by determining the proportion of the contour that lies within a circle of the same dimensions as the contour itself. If over 80% (arbitrarily chosen but could be tuned further) of the contour lies within the circle then we could accept the contour and reject all others.

It may also be possible to interpret contours that are close together and to combine them so that the ball can be detected even when obstructed. We could do this by measuring the distance between separate contours and if they are sufficiently close and approximately fill a circle when combined together, then we could mark them as a single contour.

Another improvement could be to utilize erode-dilate and dilate-erode cycles in order to make the selected contours sharper and respect the amount of chosen pixels as a more invariant property. That way, contour selection/rejection can be separate and modularized from the “cleaning up” or denoising of the selected contours.