ME 598 A
Introduction to Robotics

Fall 2022

[ME598 A Lab 3]

[Group 8]
[11/28/2022]

Graduate students:

"I pledge that I have abided by the Graduate Student Code of Academic Integrity."

This report has been prepared by:

1. Rakesh Gummapu
2. Shiv Bhat
3. Sharvil Singh

Abstract:

In this Lab 3 Image processing for Vision-Based Tasks, we used the MATLAB Color Thresholder App for recognizing the orange cone through various filters like RGB, HSV, YCbCr and L*a*b* the then using the filtered Image. We determine the size, shape and color of the cone, also the displacement distance of the cone from the center of the image and the distance between the cone and camera are also taken into consideration. Finally, we approximate the distance between the camera and the object. We create training models for generalizing that task for cones at different orientation and distance. Then we use the MATLAB Image Batch Processor App for processing a group of images through the function that we created in Part 2. Finally, we get the Centroid view, Mobile Robot Overhead View for each cone types to better generalize it for multiple cones.

Table of Contents

1 Introduction

Robot algorithm optimization and self-learning and analysis capabilities are becoming more and more important to our daily lives as robotics advances. Intelligent sweeping robots have an unique use in daily life since they can self-evaluate barriers, create a thorough cleaning route, and automatically scan the entire cleaning area. This brings them closer to the cleaning decisions that people make. People can, however, avoid impediments through vision in a reasonable manner, and the sweeping robot can only adequately plan the route by replacing human vision through the use of visual sensors.

This Lab 3 experiment attempts to identify important location coordinates using color. By identifying items with particular colors (as key positioning coordinates), we can separate from other portions using MATLAB's Image Processing Tool. Through the item's size, its displacement from the image's center, and the distance separating it from the center of the image Calculate the object's distance from the camera.

We can achieve a function akin to the intelligent detection of impediments by a sweeping robot if we do a more in-depth analysis of the total Lab 2 experiment. Even some moving barriers can be detected. As a result, we have developed some fundamental algorithms for autonomous navigation in this lab, primarily concentrating on color recognition and image processing.

2 Theory and Experimental Procedure

We must now carry out the rest of our experiment by taking the subsequent actions.

Selecting several orange cone patterns from a variety of photos is the first stage. This is the experiment's fundamental data.

The range of these orange cones' hue, saturation, and value (HSV) attributes must be identified in order to use them in the second phase for color detection in various test photos. Additionally, a calibrated color detection code can precisely identify cones across the whole spectrum.

<Image>

The third step is carrying out the required image processing operations on the color of the image that was detected in step 2's image. Set the color detector such that it can locate the centroid of each training image for one or more cones. The relative direction and distance to the cone of the capture camera can then be mapped using the recovered cone image data.

The code is tested on the fresh image in the fourth stage, and all configuration Blob analysis and target recognition are once again performed using the previously created color threshold function.

The final step is to test every aspect of the function. To do this, turn the phone's screen to its brightest setting and fully maintain the image FOV at a specific point in the computer webcam (but without completely filling the FOV), then take a snapshot of the simulation cone, save it in MATLAB, select Generate, and use the computer.

3 Part 1: Camera Setup

After having learnt from the video lesson and the instruction provided in the stub file, we achieved the following:-

1. Installed and enabled the webcam within the MATLAB

```
mycam = webcam;
mycam.Resolution='640x480';
```

2. Grabbed the image in the MATLAB

```
img = snapshot(mycam);
figure(1)
imshow(img)
```

3. Saving the image locally on our system for further processing

```
figure(1)
saveas(gcf,'Image1','jpg')
```

4 Part 2: Configure Color Detector

In this section we have analyzed the cone and manually tried to process the image using filters like RGB, HSV, YCbCr and L*a*b*. Our motive in this part is to extract the cone out of the whole image using the above filters.

Below are the images they we processed to extract out the cone from the actual images:-

(Note :- Figure 2(a) - 2(b) are for single cone & Figure 2(f) - 2(i) are for two cones)

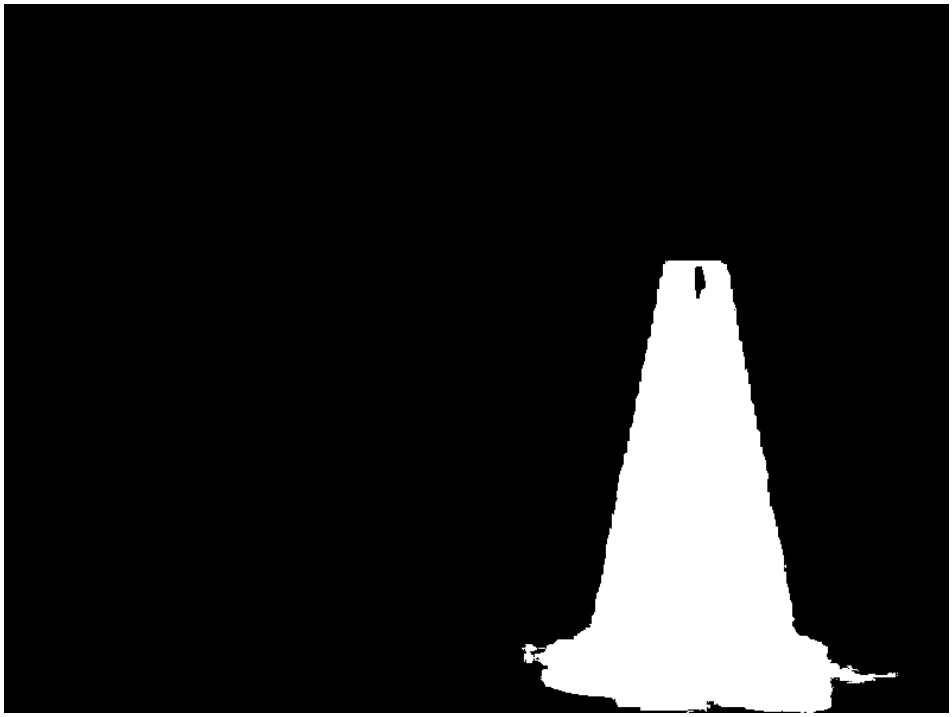

Figure 2(a)

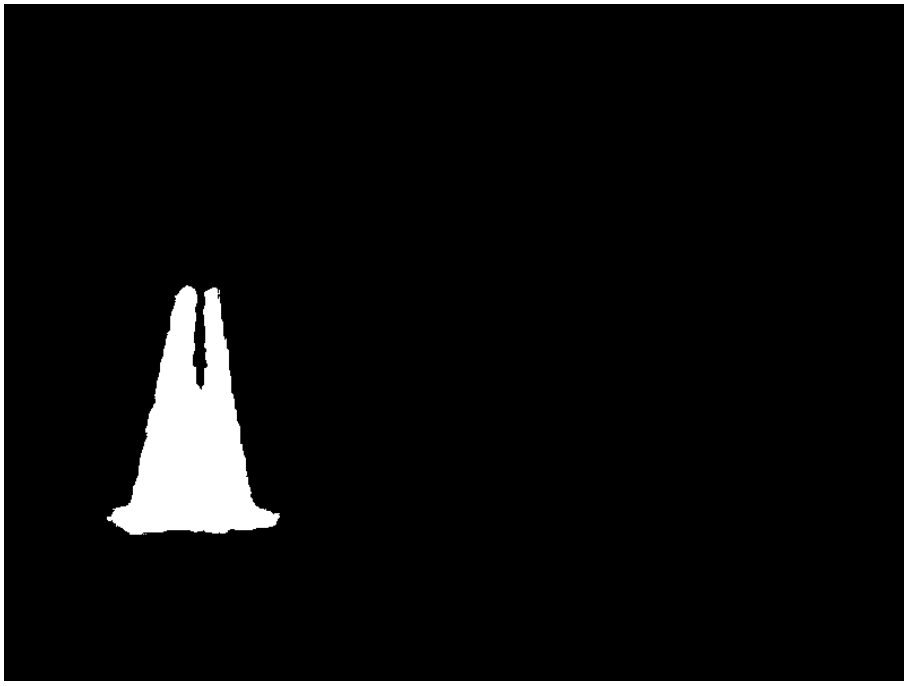

Figure 2(b)

Figure 2(c)

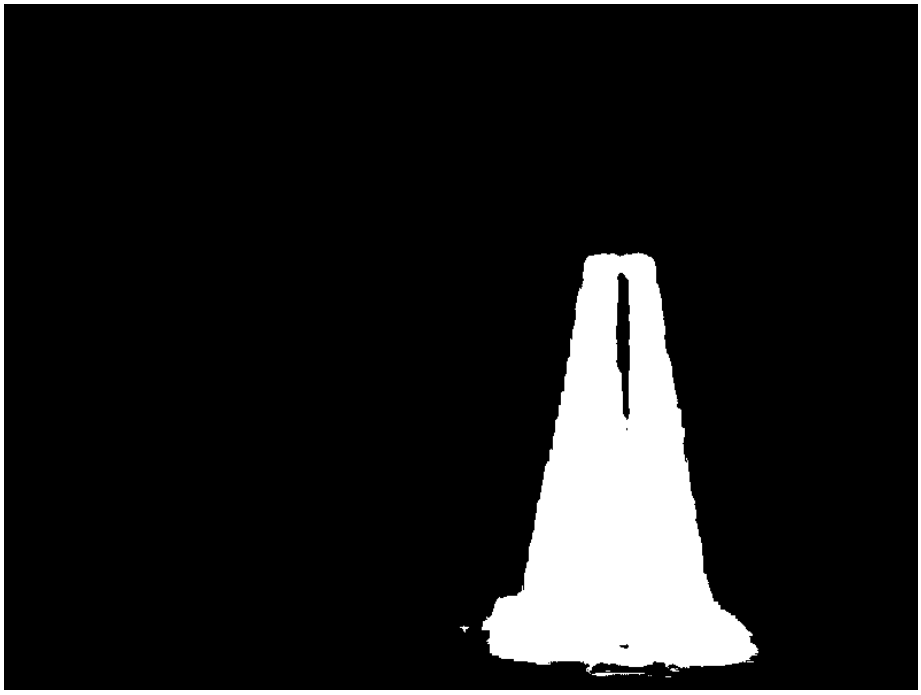

Figure 2(d)

Figure 2(e)

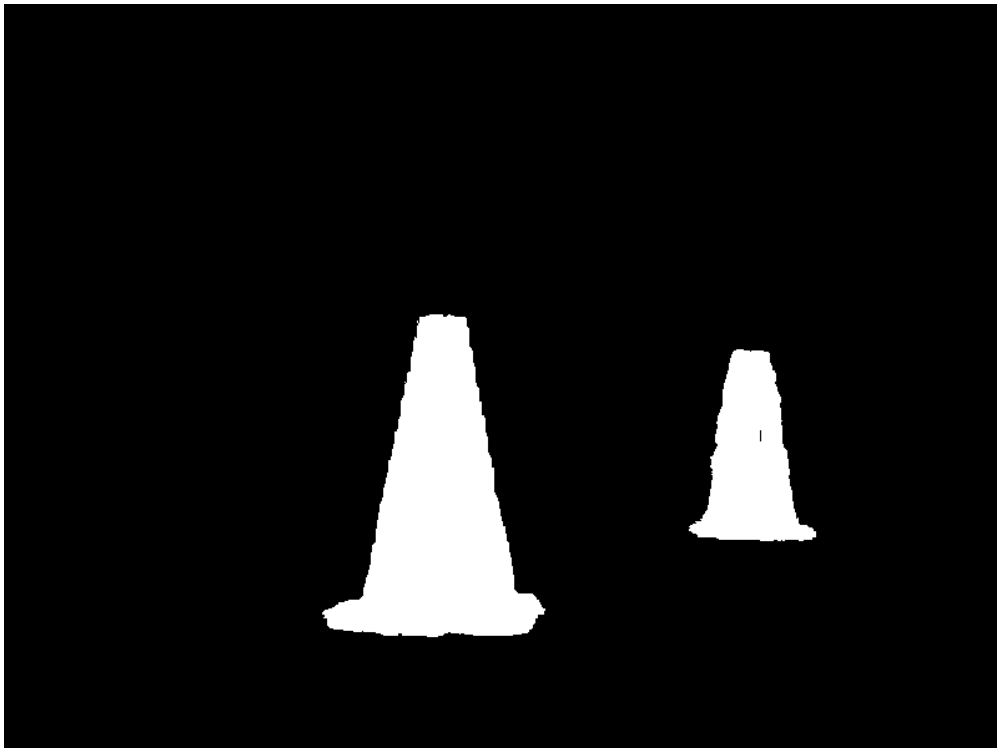

Figure 2(f)

Figure 2(g)

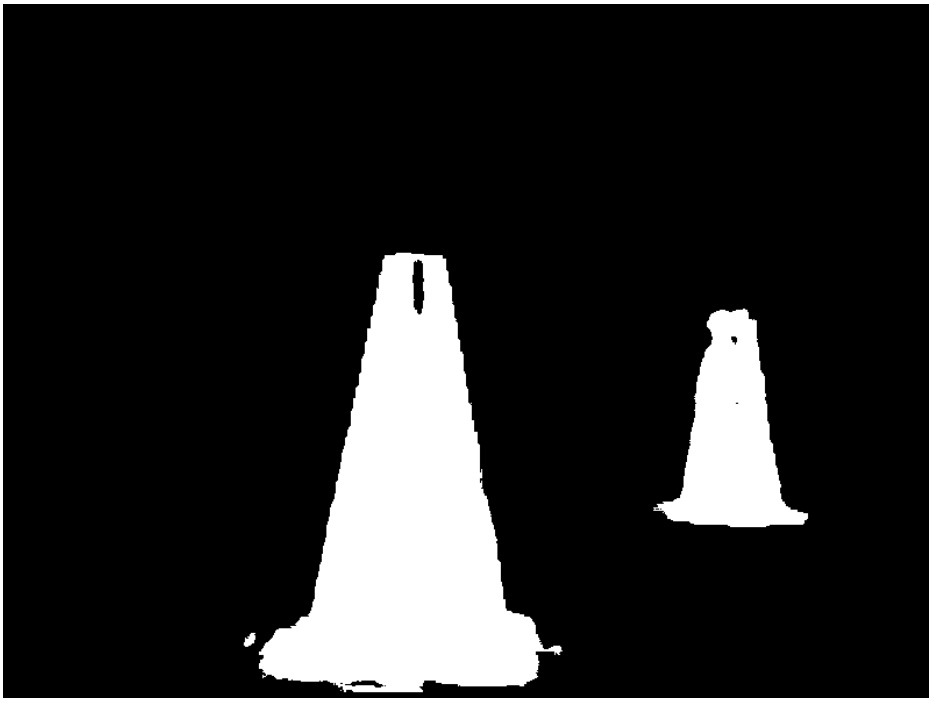

Figure 2(h)

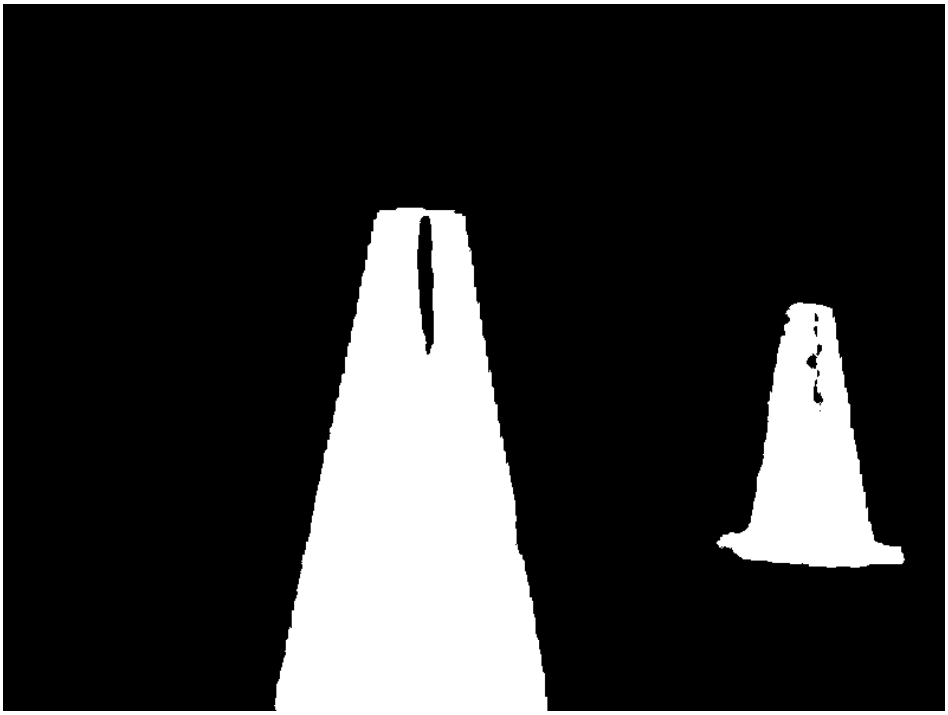

Figure 2(i)

5 Part 3: Configuring Blob Analysis & Target Detection

5.1 Extracting centroid of cone(s) in image

In this step we used necessary Image processing steps on color detected images using the inference we gained from part 2 for determination of the centroid position of the cone or cones in all the training images.

The steps performed in this part are as follows:-

1.) Filtering out connected pixel regions smaller than a user-defined **threshold**
2.) **Filling in holes in the image**
3.) **Labeling the remaining collected components**
4.) **Component** area extraction
5.) Component centroid extraction
6.) Plotting of centroid in the color detected image

Below are the resulting images indicating centroid over one and two cones after the above computation:-

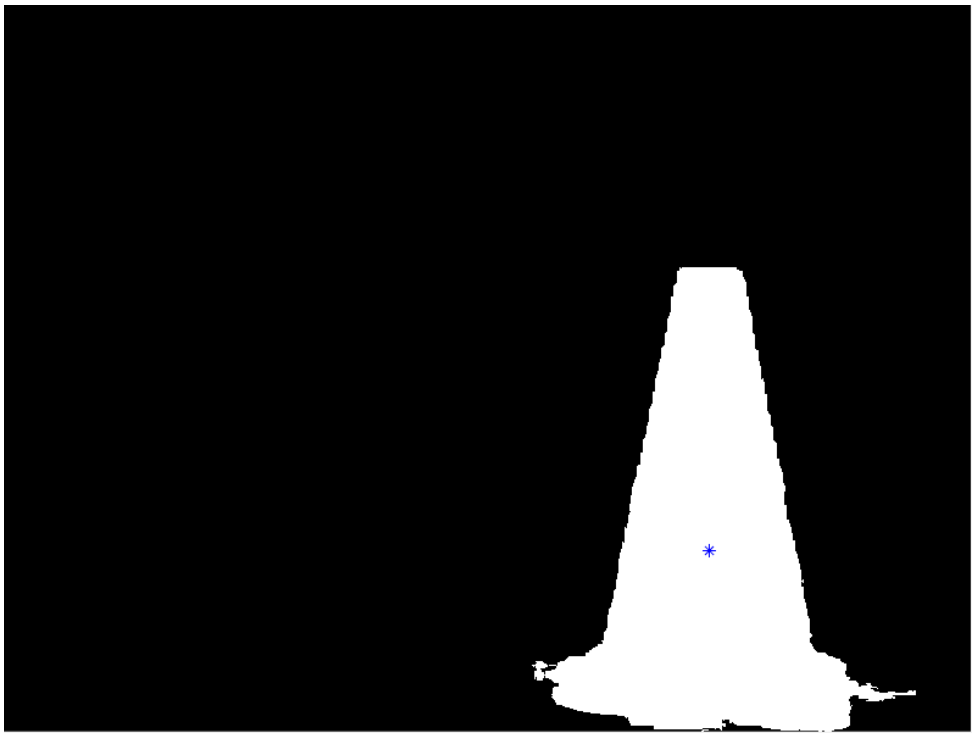

Figure 5.1 1

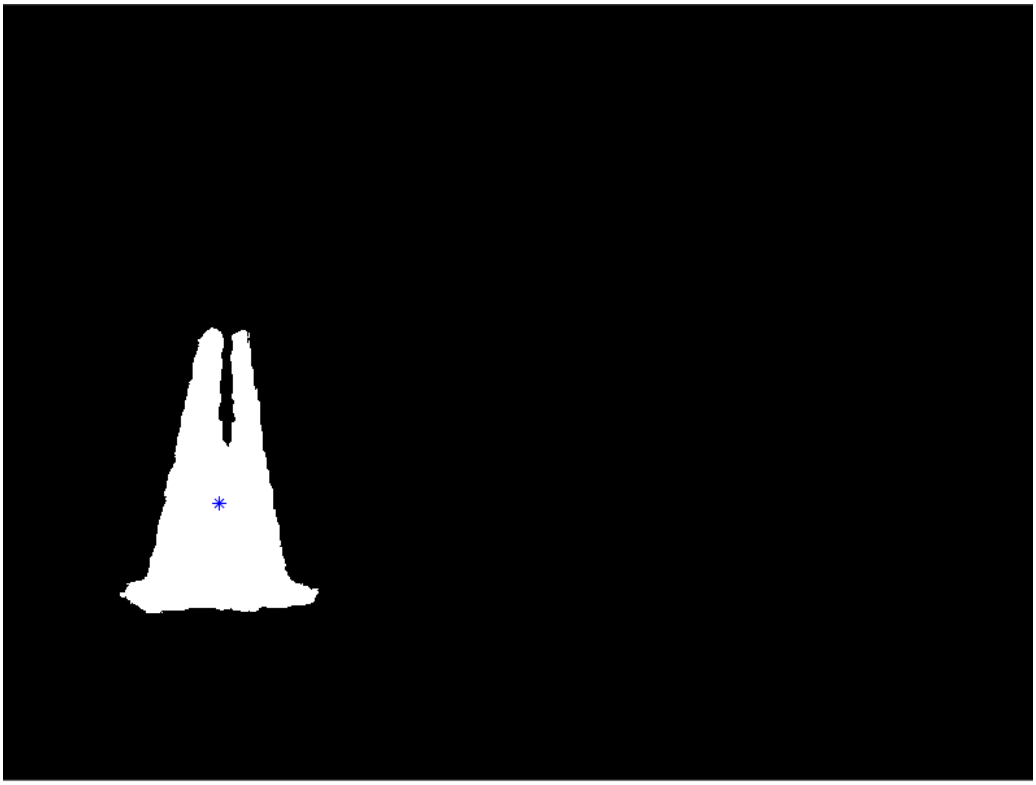

Figure 5.1 2

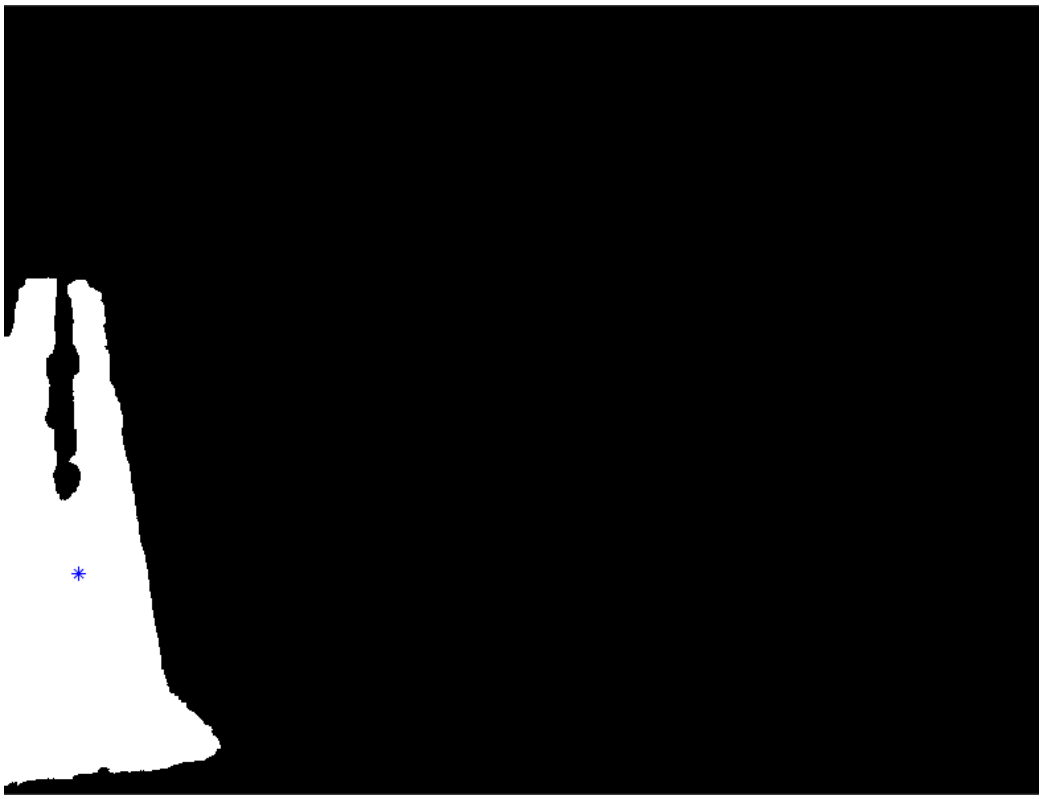

Figure 5.1 3

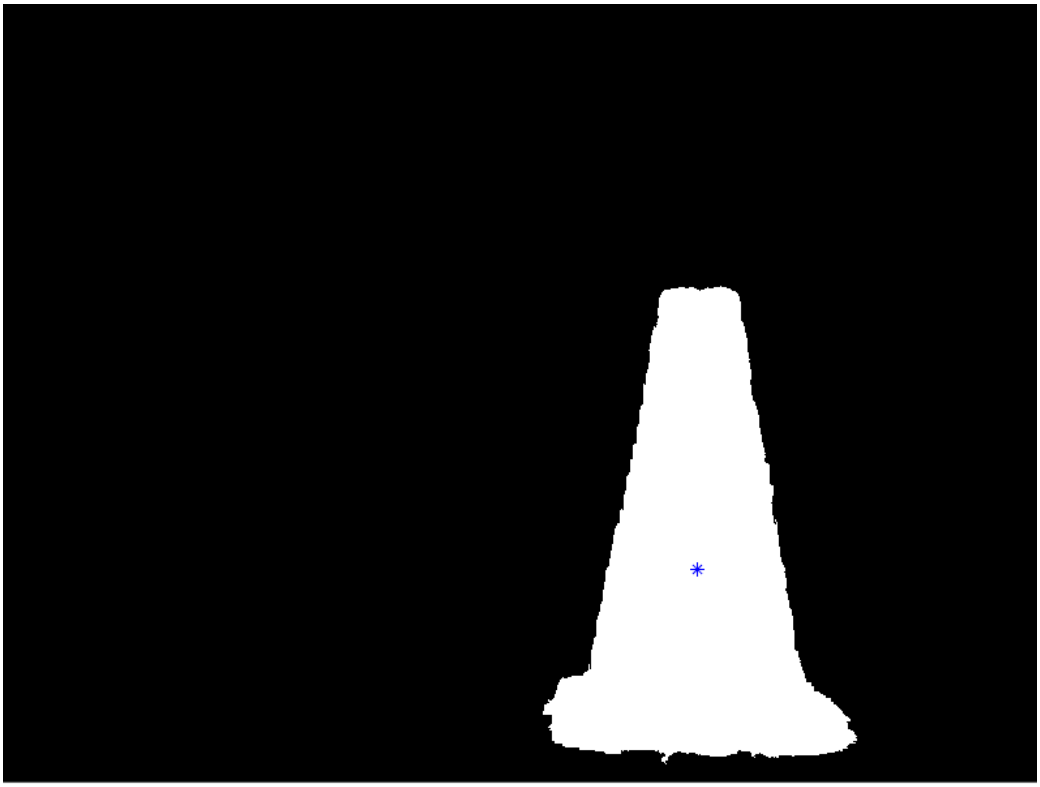

Figure 5.1 4

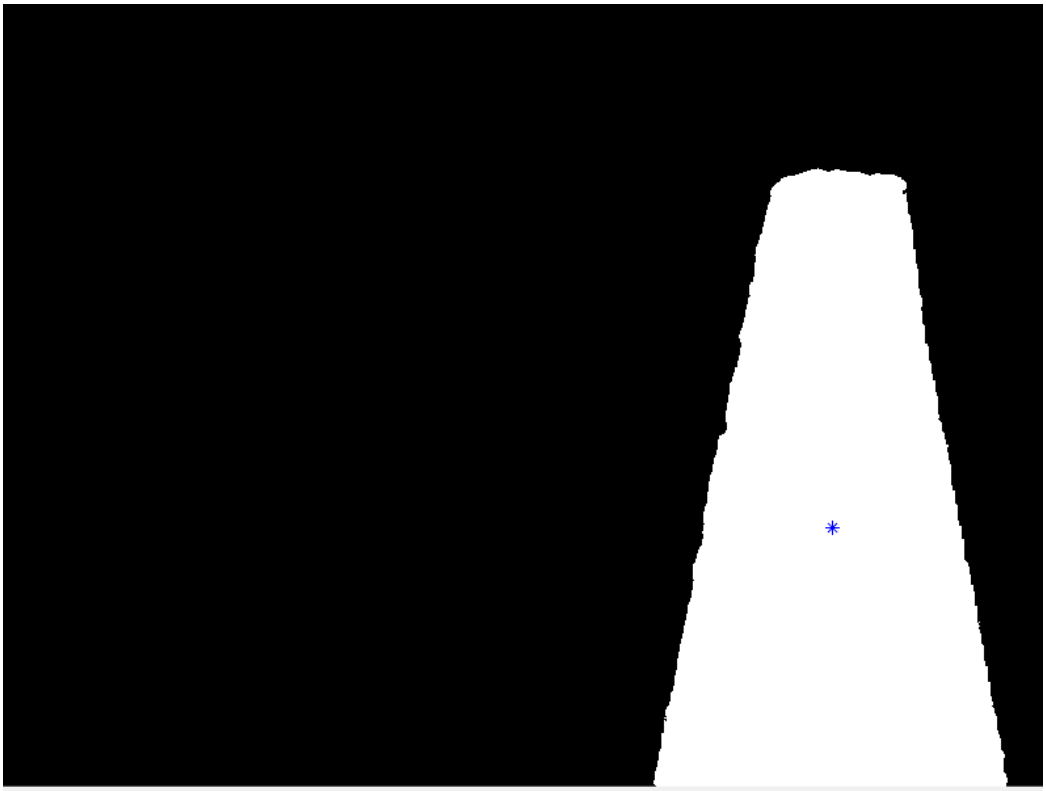

Figure 5.1 5

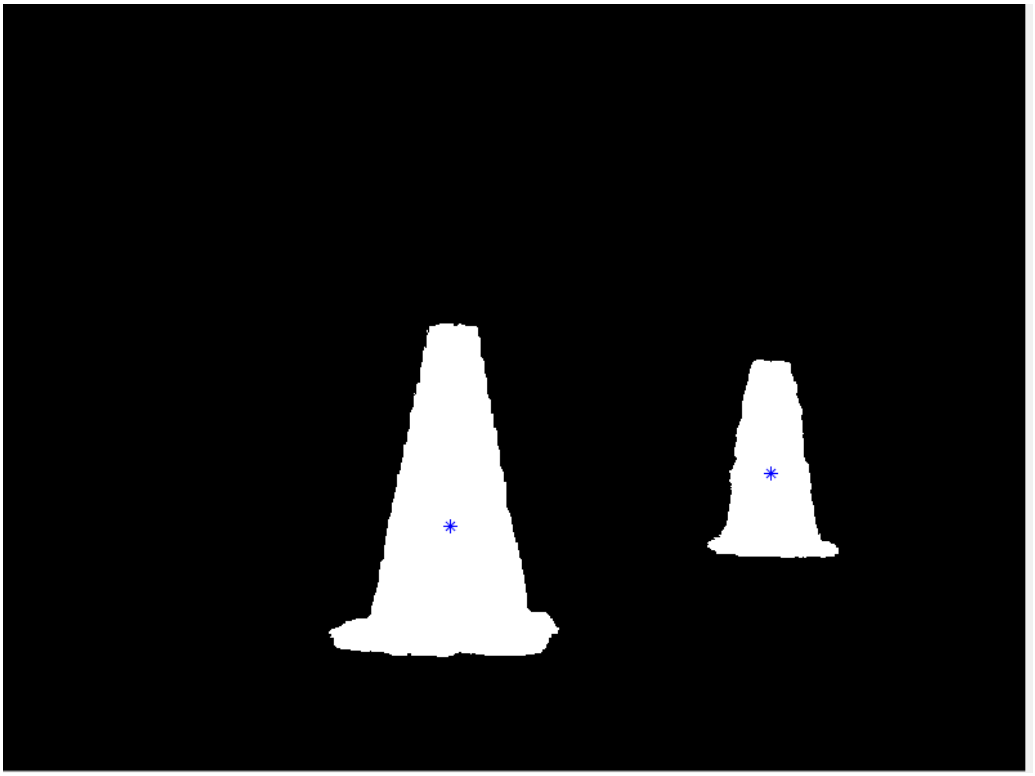

Figure 5.1 6

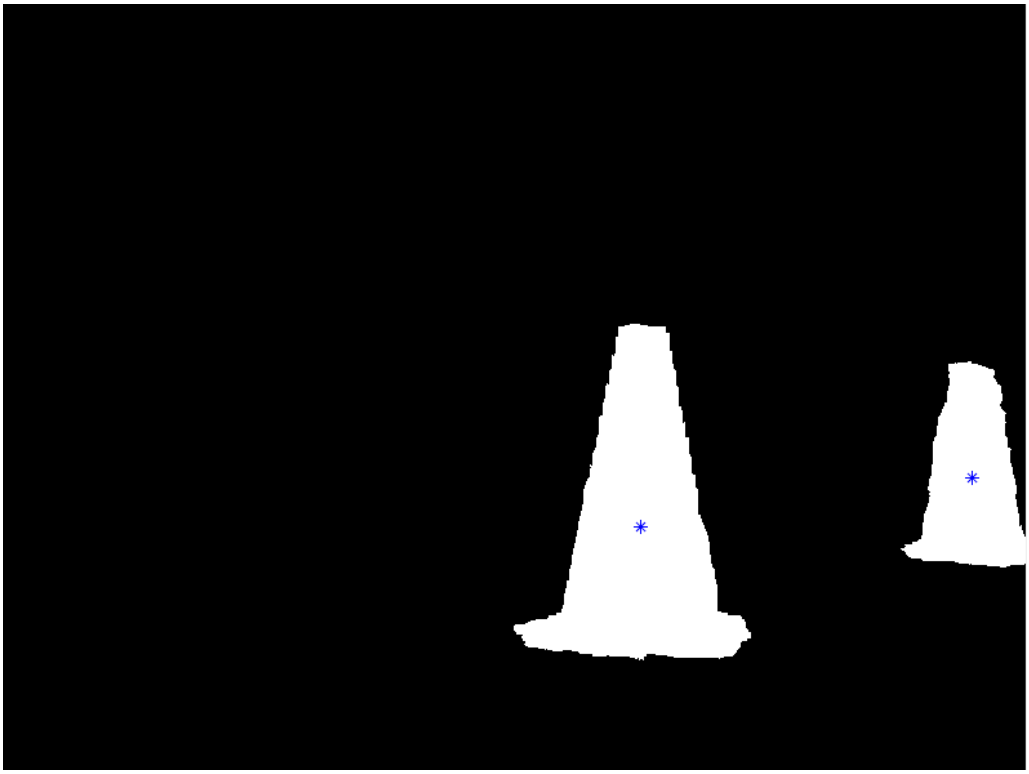

Figure 5.1 7

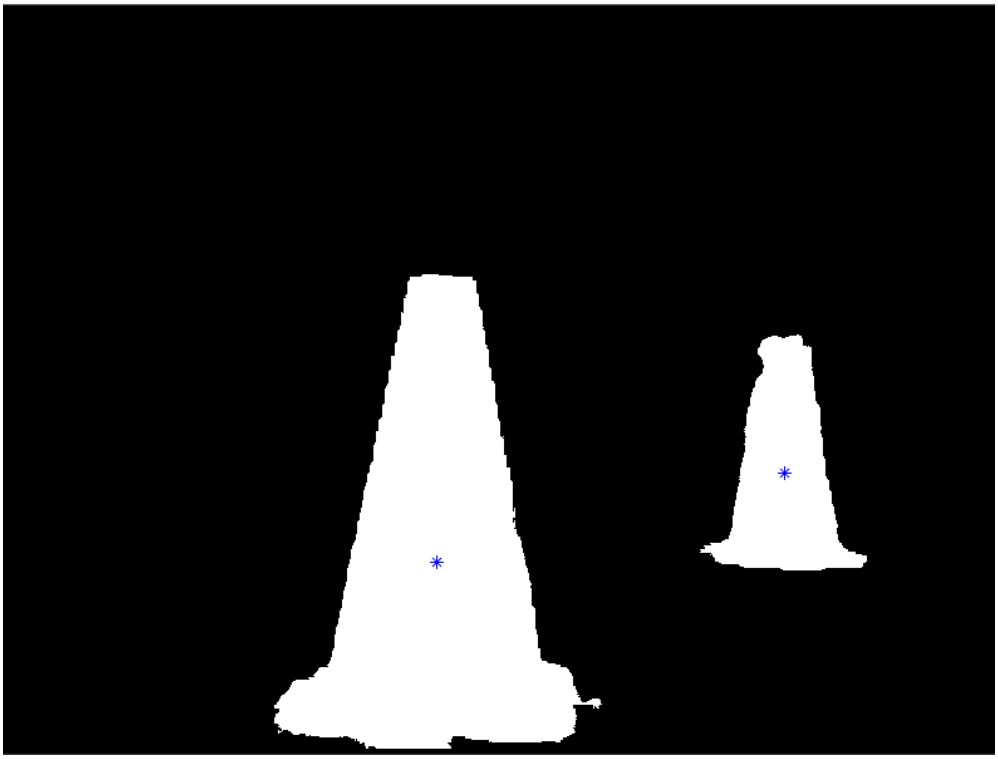

Figure 5.1 8

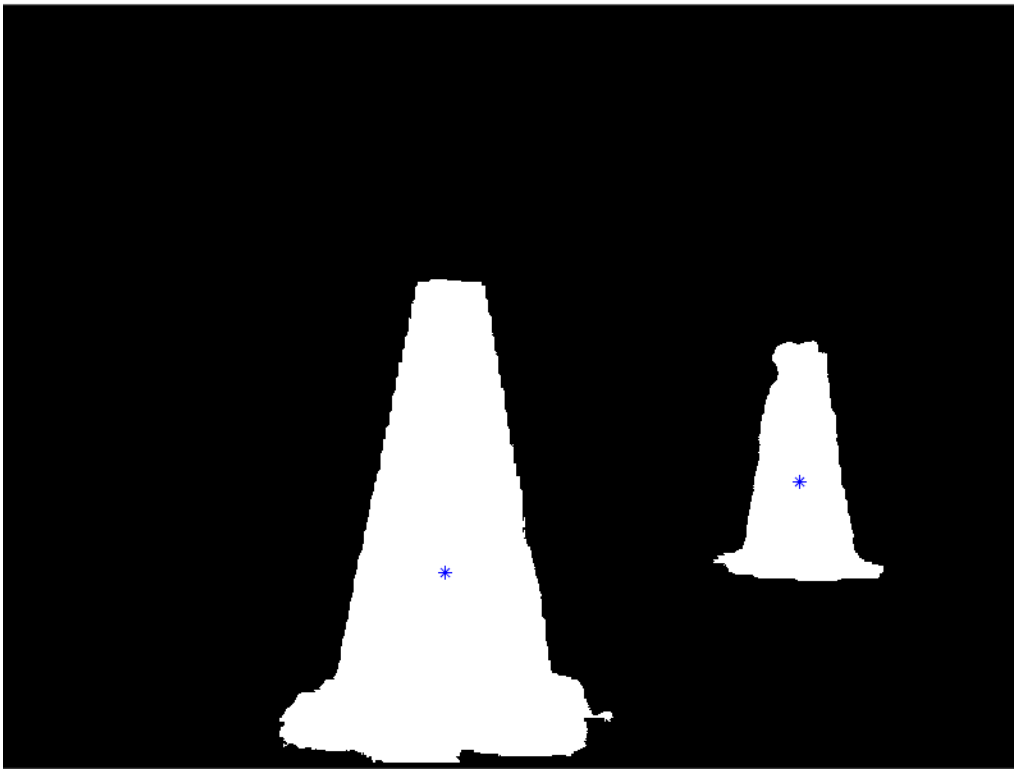

Figure 5.1 9

5.2 Landmark localization to detect targets

In this part we used the extracted cone image data from part 3(a) to ,ap the relative orientation and distance of cones with respect to capturing camera. For achieving this we compare the horizontal centroid of a detected blob to the dimensions of the processed image.

Below are the commands we used in this part:

- To find the size of a matrix in MATLAB

```
[nr nc] = size(I);
```

- Where nr = # of rows in matrix and nc = # of columns. To find the center coordinates for the image we use:

```
ImageCenterX = nc/2;

ImageCenterY = nr/2;
```

- To Compare the ImageCenterX value against the X coordinate of the cone's centroid, CentroidX :

```
Xdiff = CentroidX - ImageCenterX;
```

Below are the images indicating the Mobile Overhead View of the image from part 3(a).

(Note Figure 3(b) a to Figure 3(b) d are for one cone and Figure 3(b) e to Figure 3(b) h are for two cones)

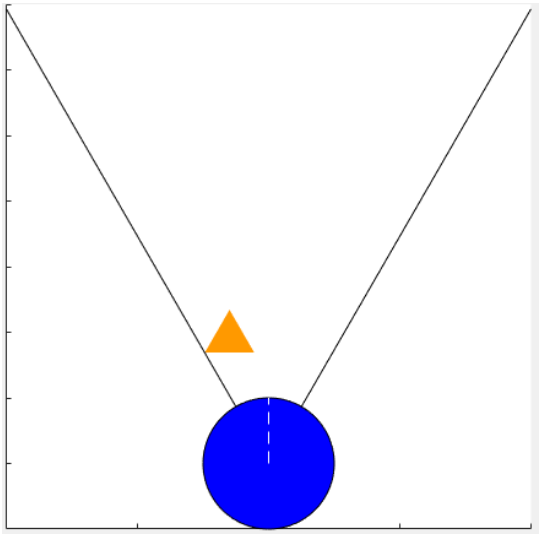

Figure 5.2 1

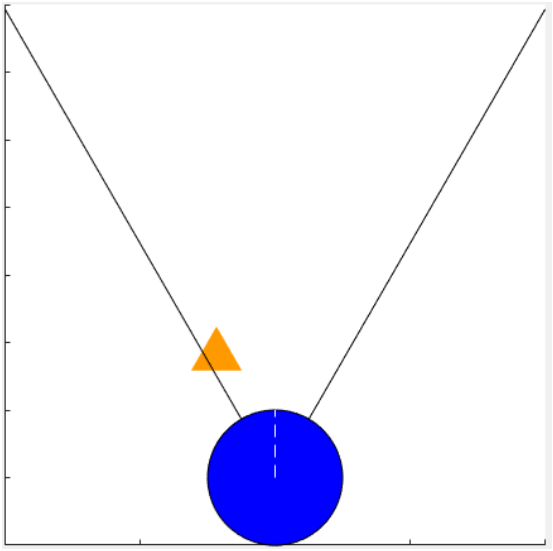

Figure 5.2 2

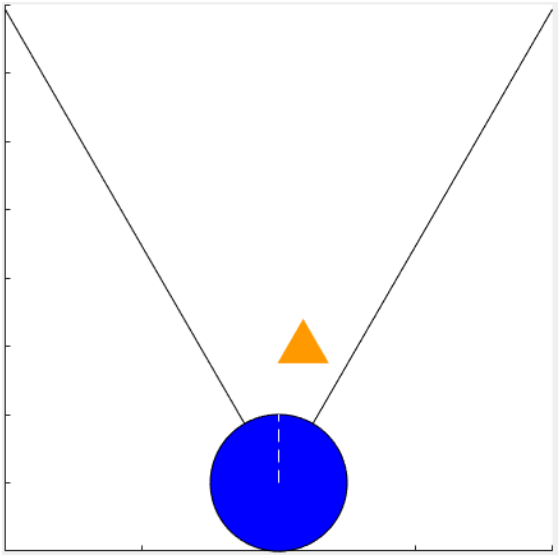

Figure 5.2 3

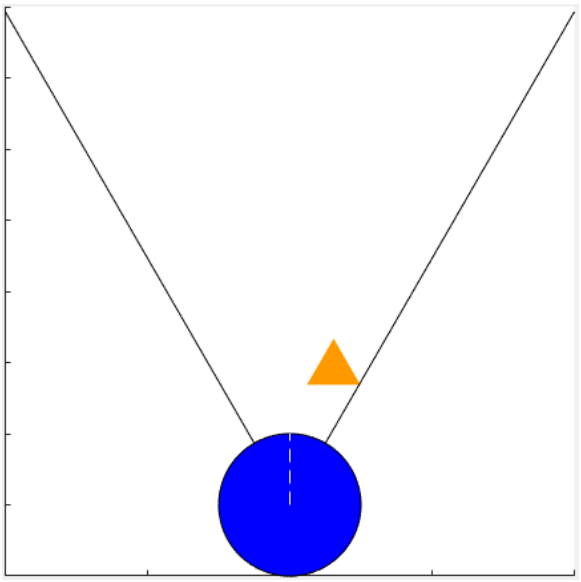

Figure 5.2 4

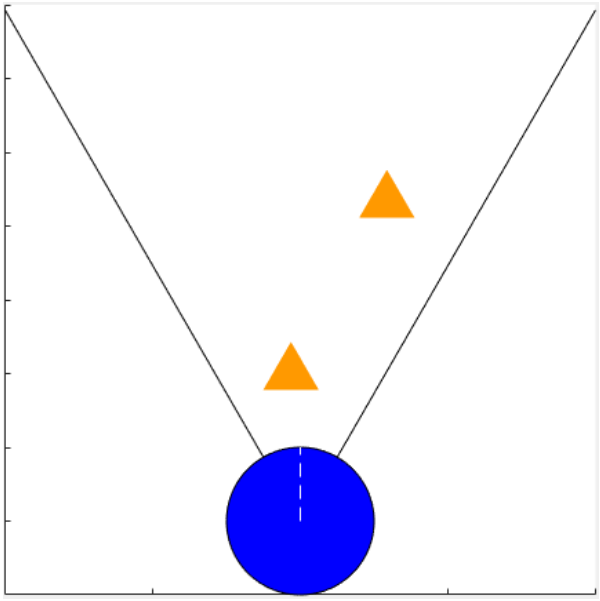

Figure 5.2 5

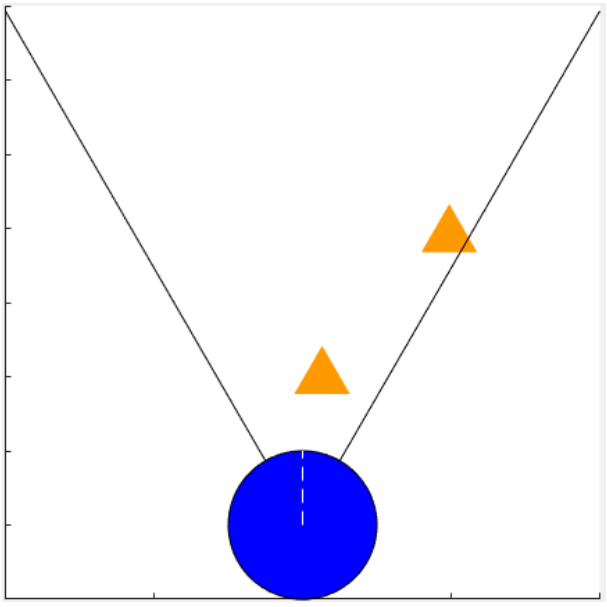

Figure 5.2 6

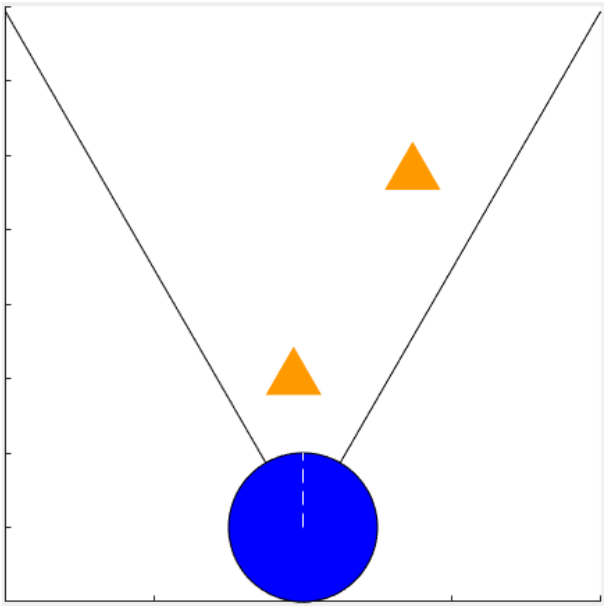

Figure 5.2 7

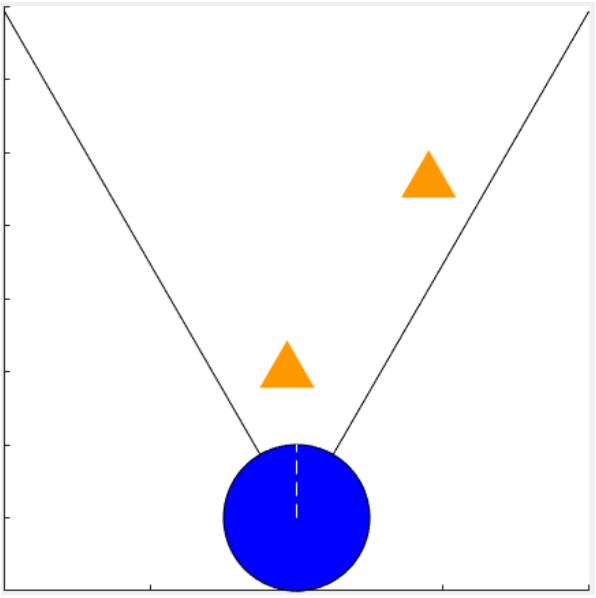

Figure 5.2 8

6 Part 4: Testing Blob Analysis & Target Detection

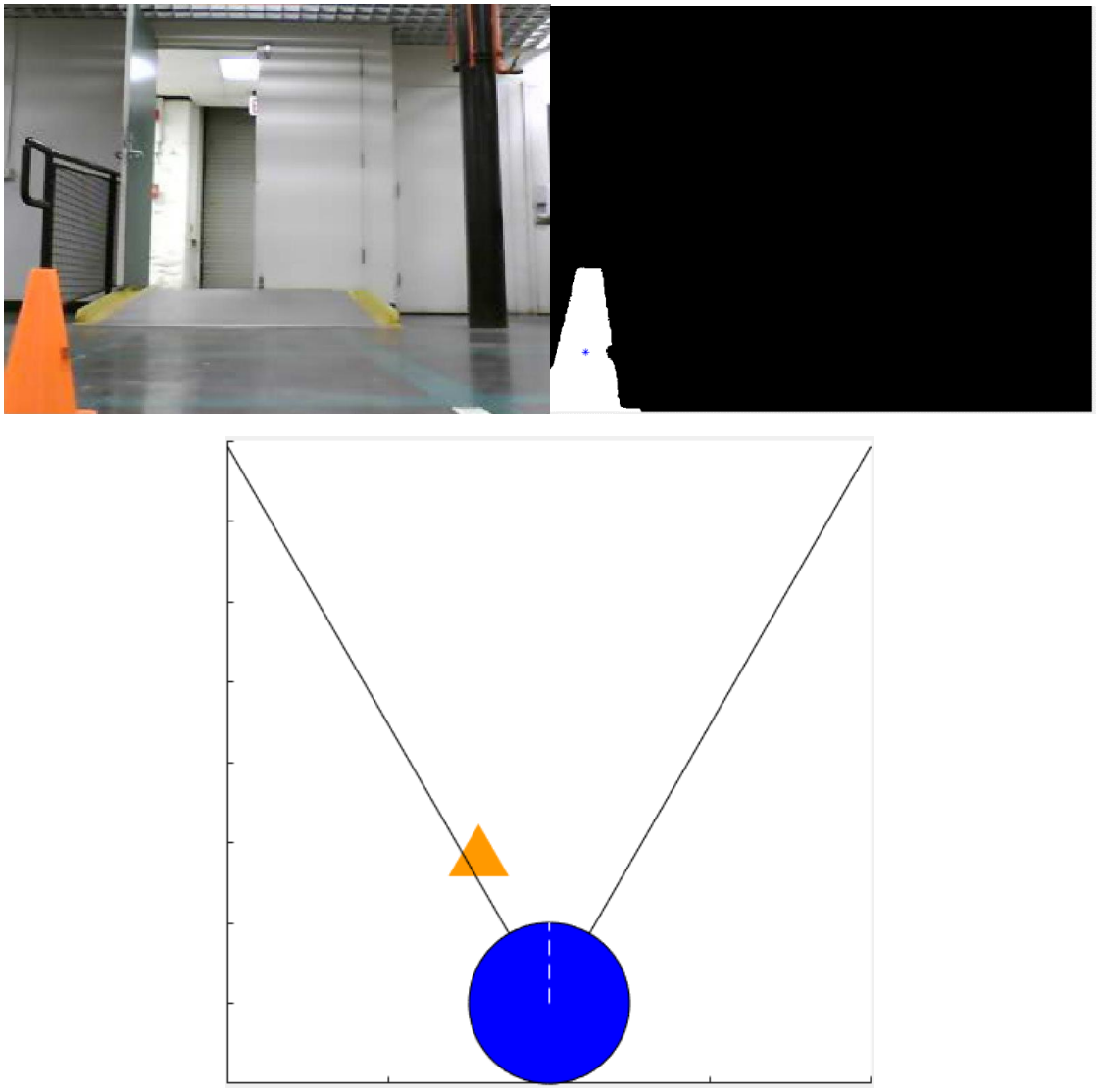

Figure 6 1

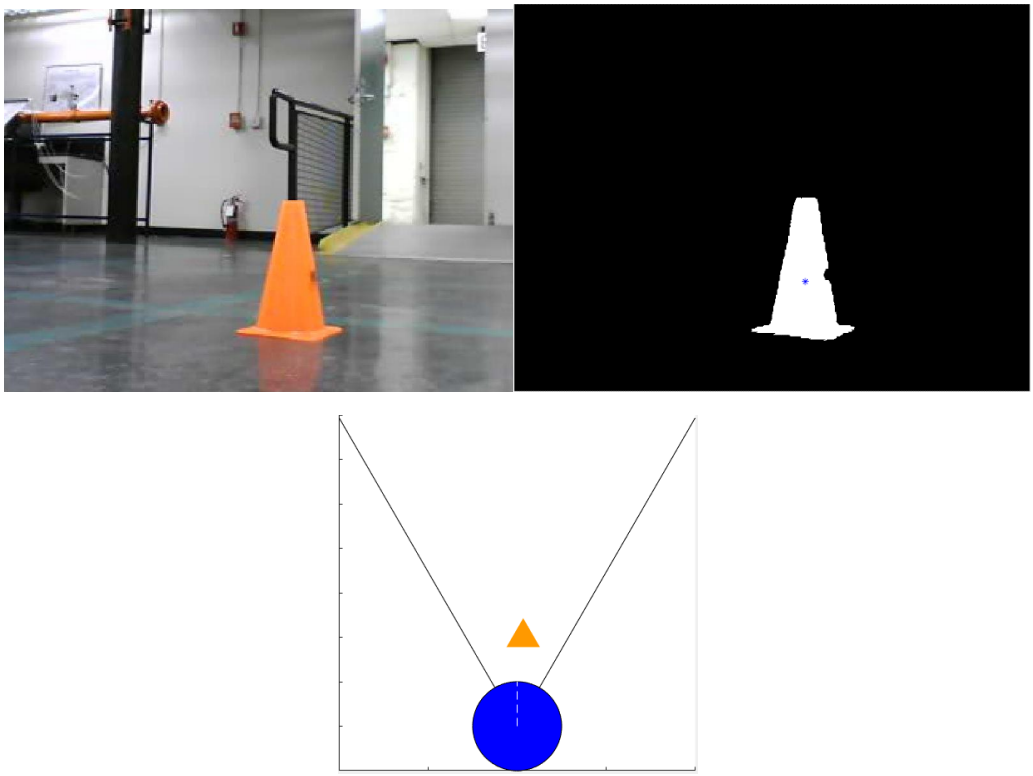

Figure 6 2

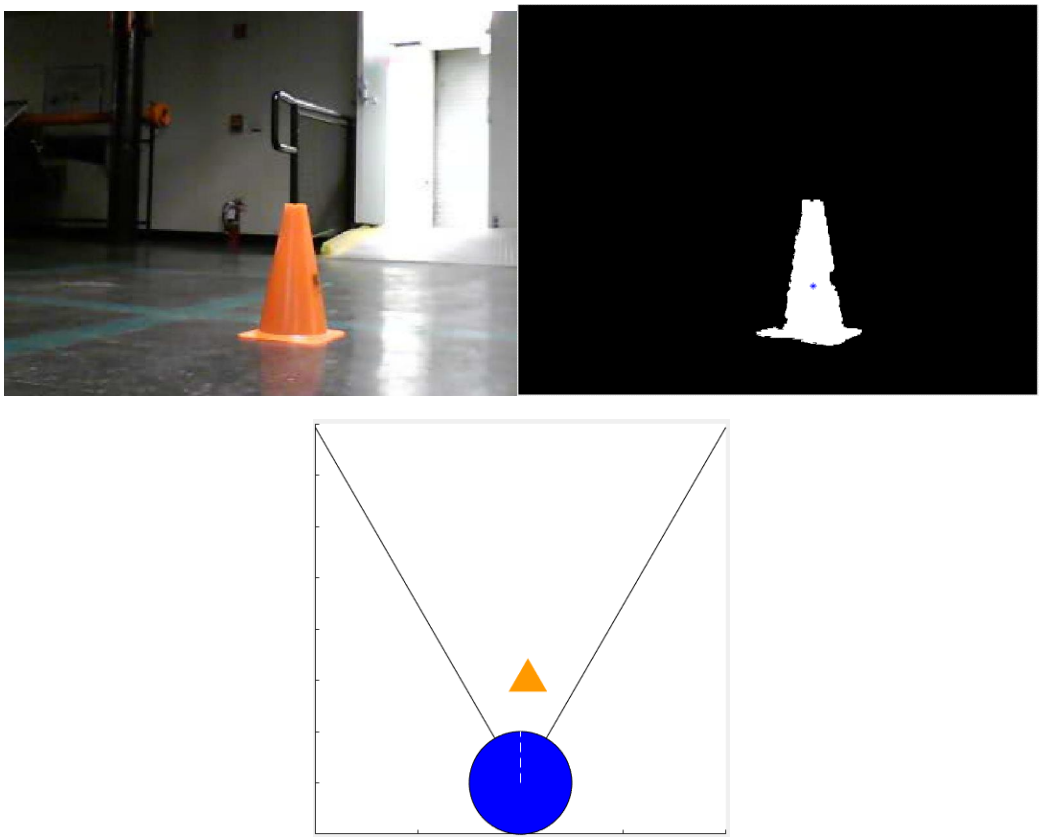

Figure 6 3

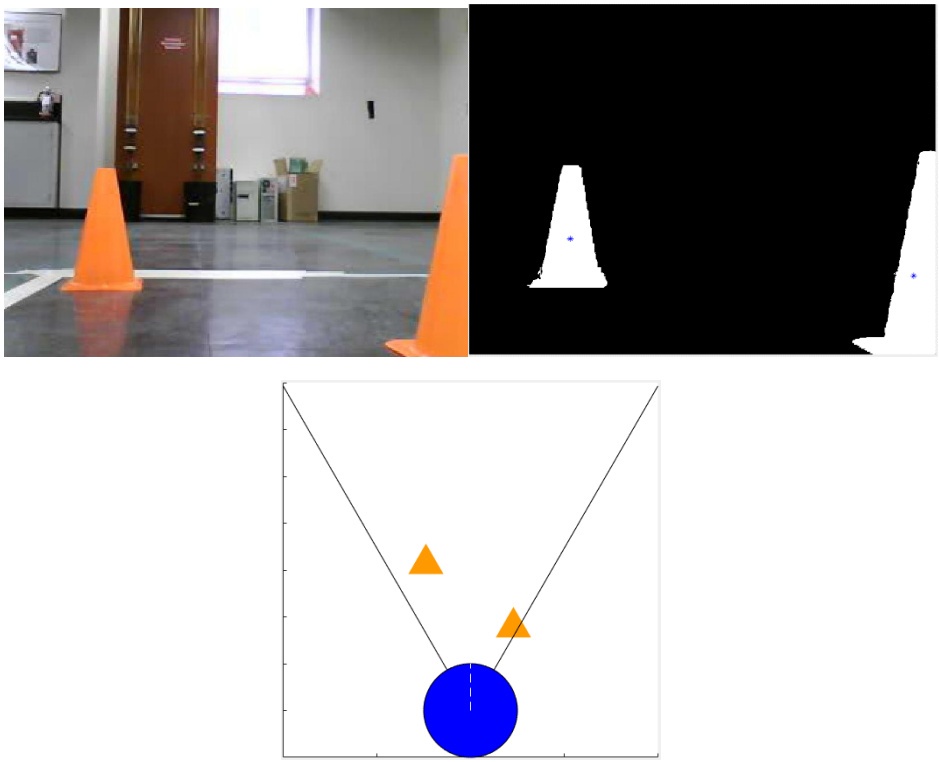

Figure 6 4

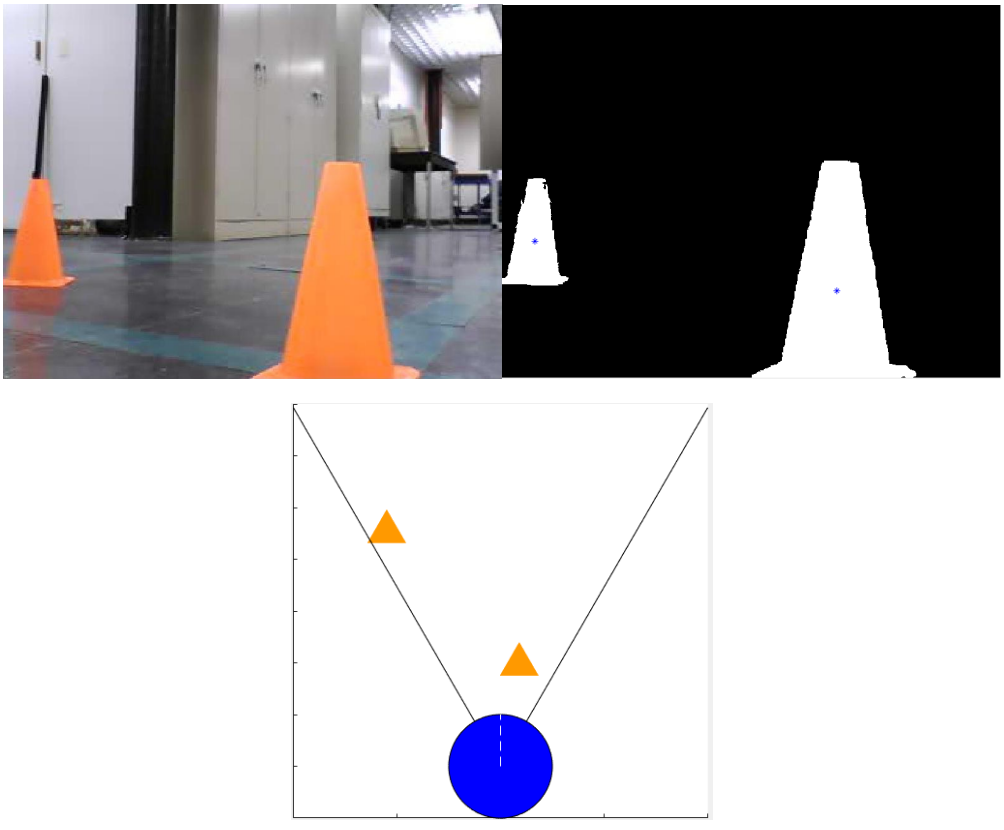

Figure 6 5

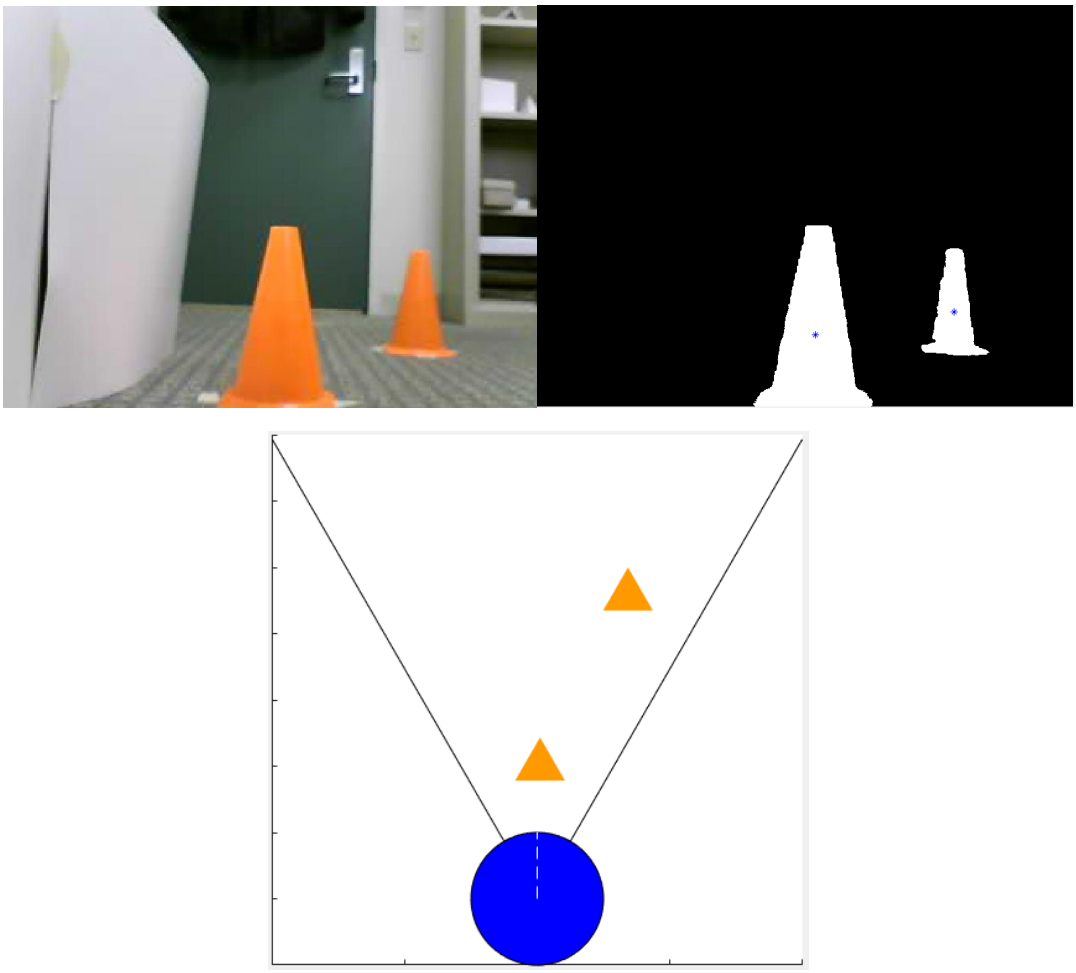

Figure 6 6

7 Part 5: Vision-Based Detection with Newly-Captured Images

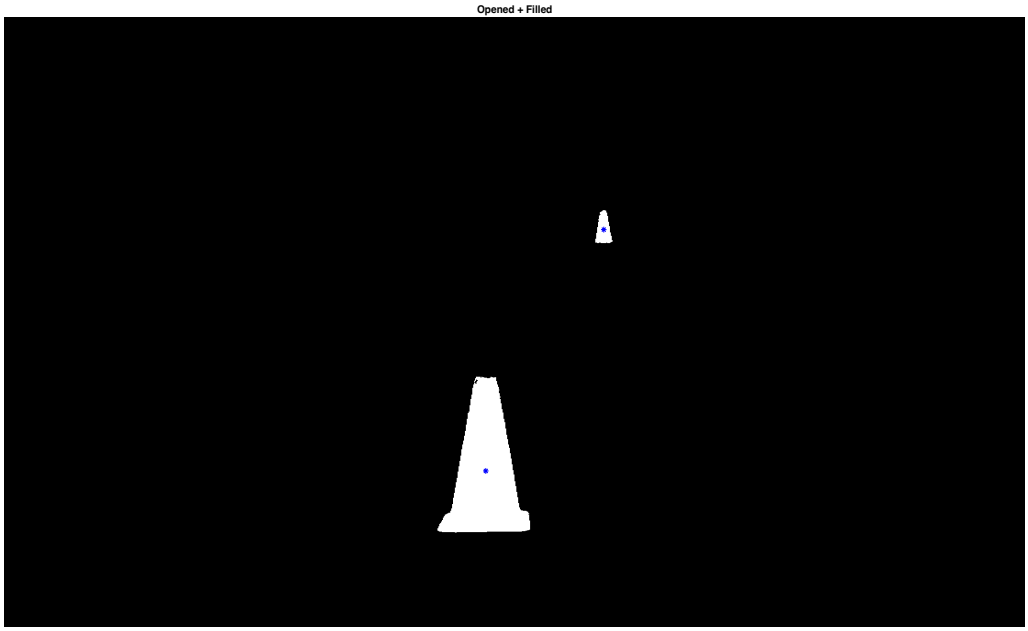

Figure 7 1

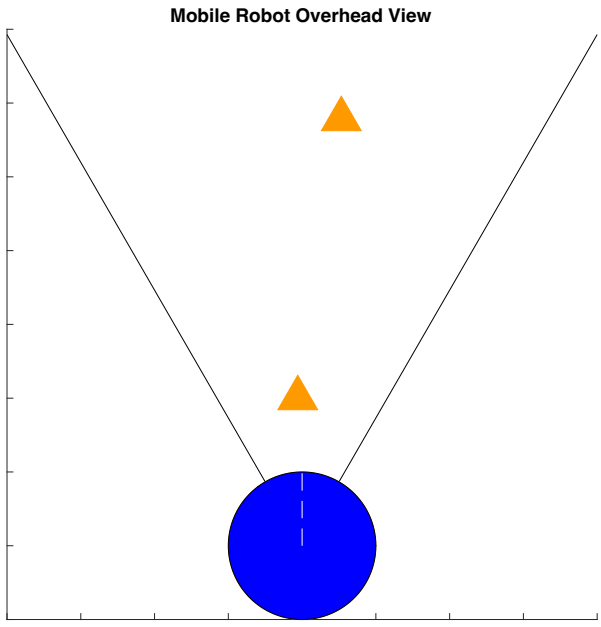

Figure 7 2

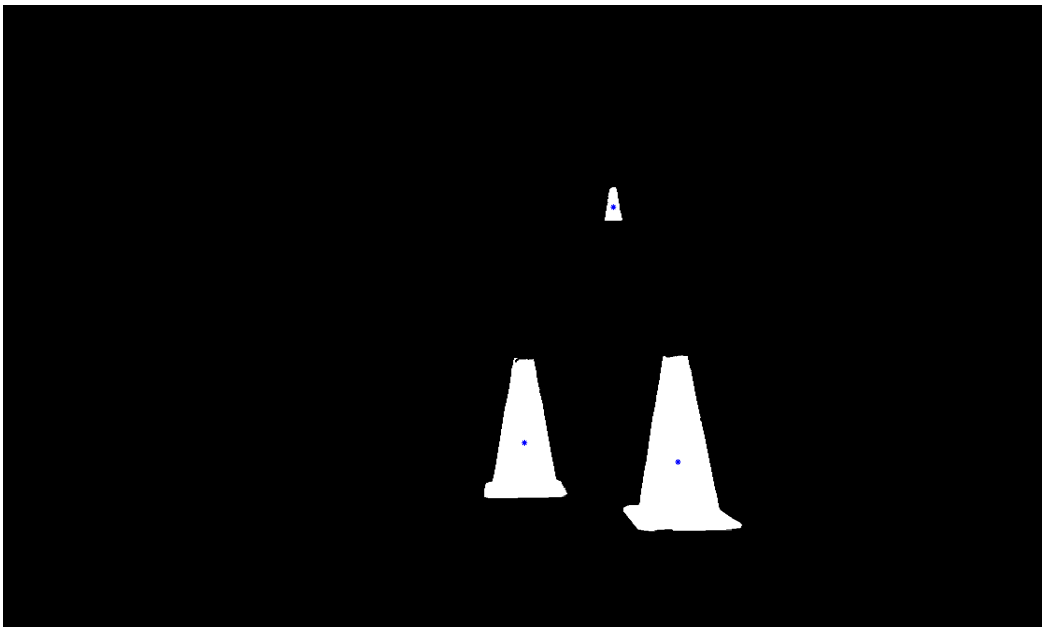

Figure 7 3

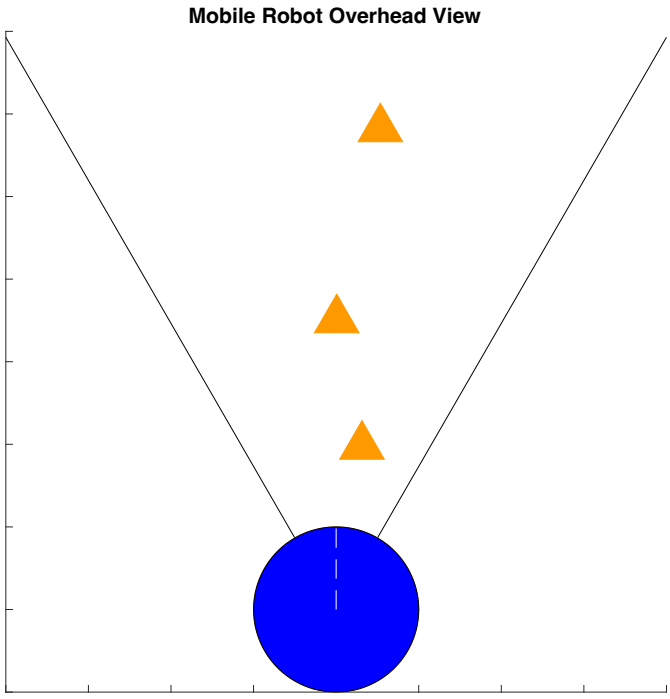

Figure 7 4

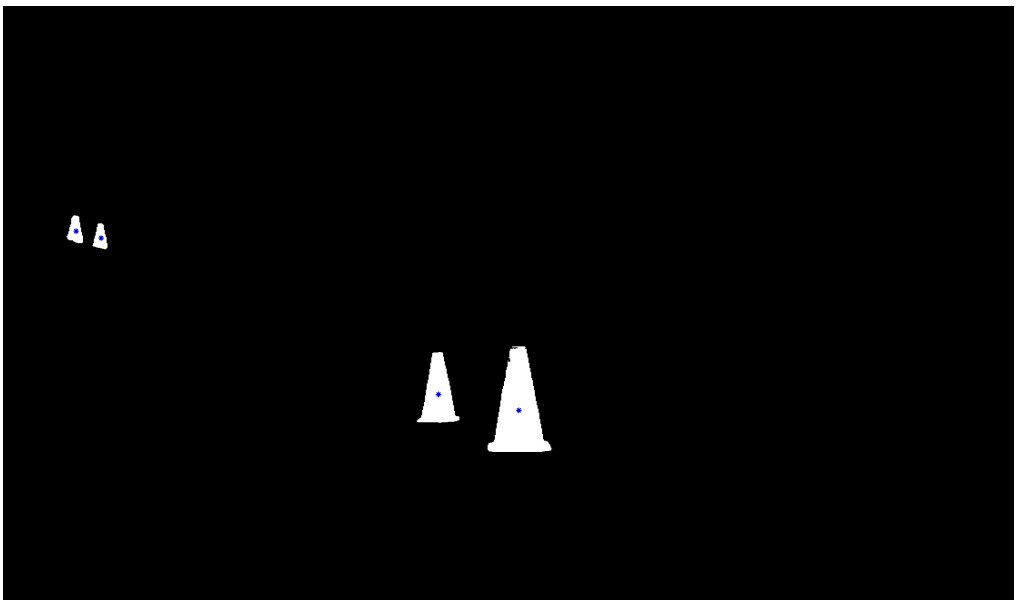

Figure 7 5

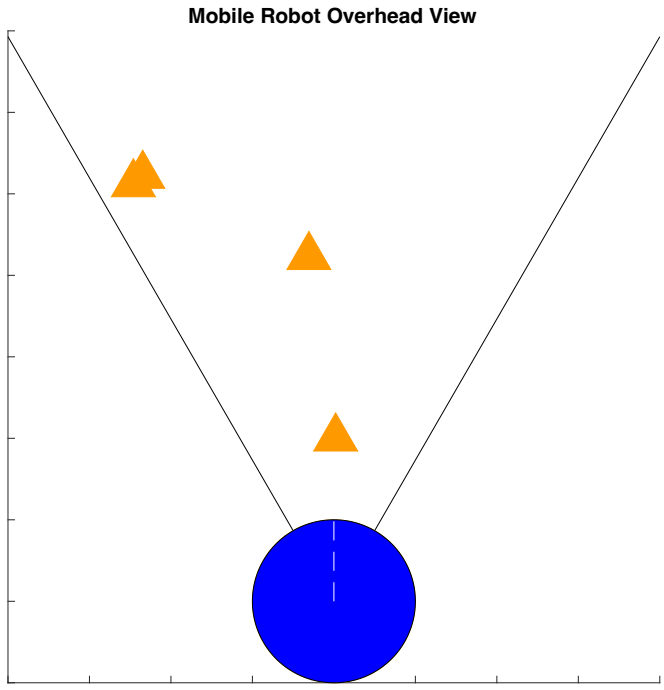

Figure 7 6

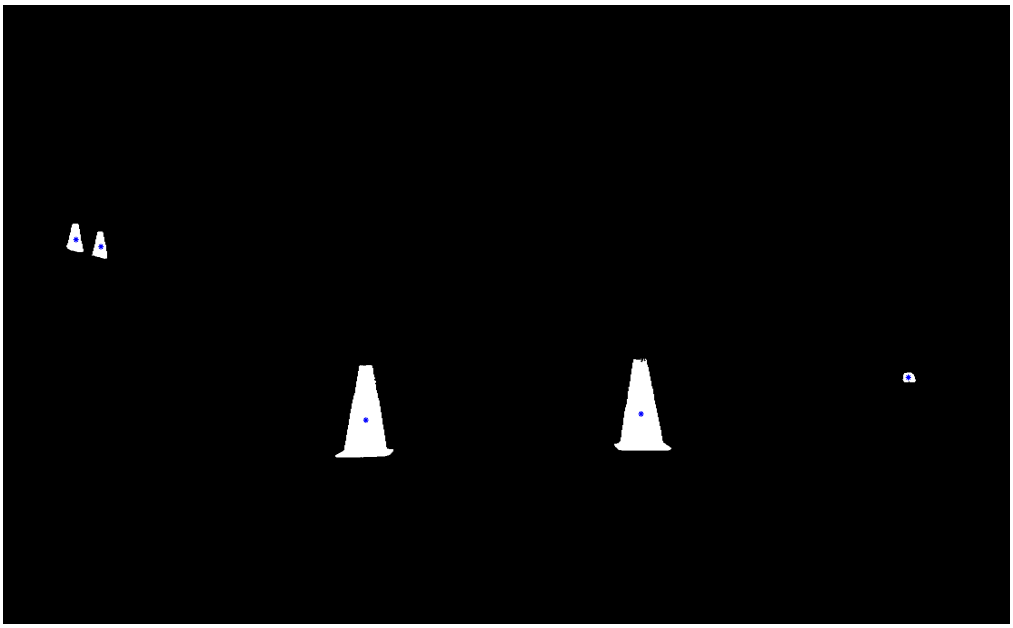

Figure 7 7

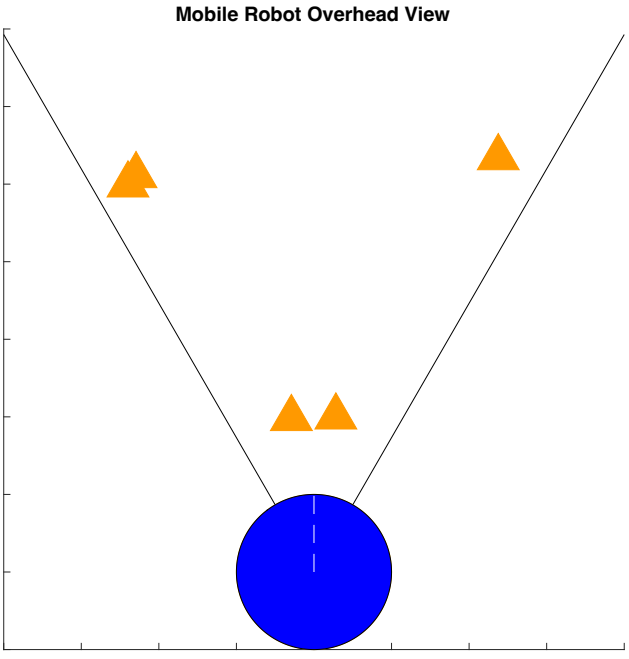

Figure 7 8

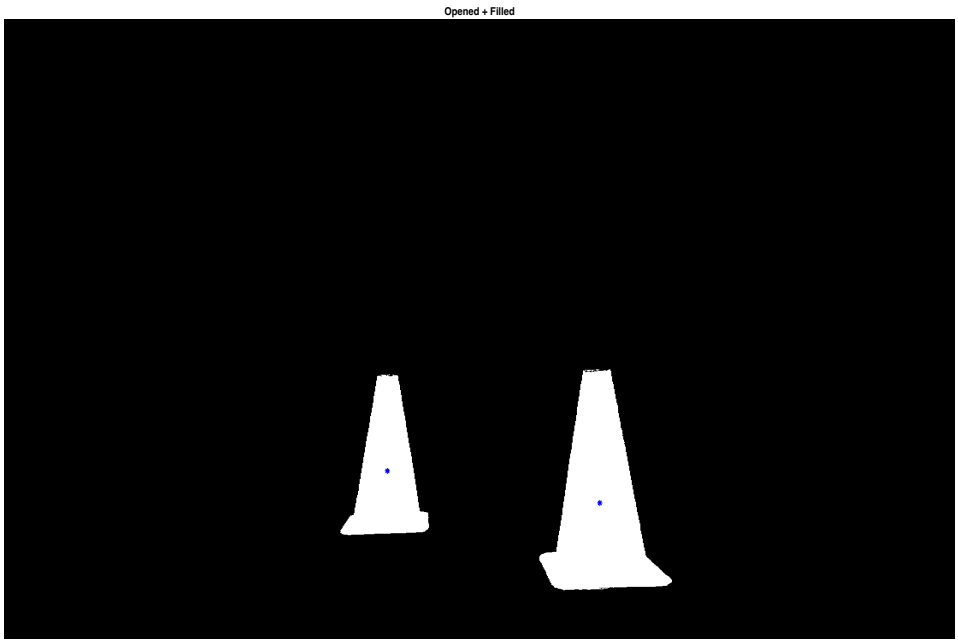

Figure 7 9

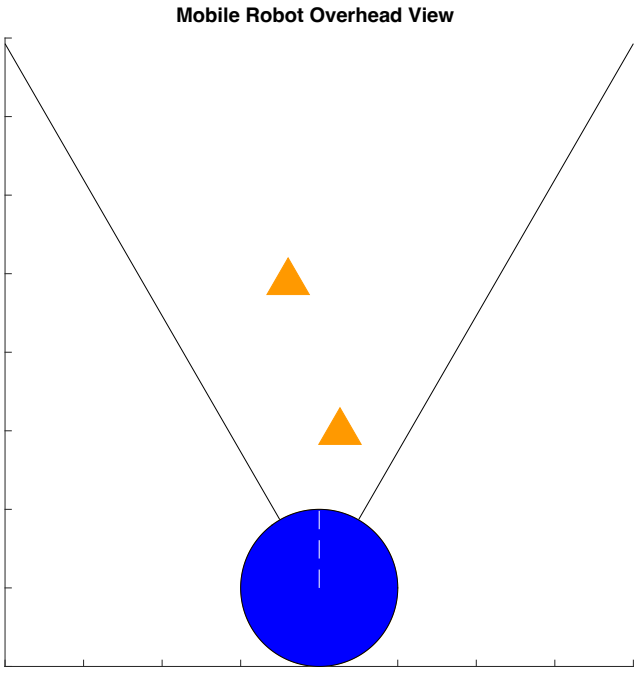

Figure 7 10

8 Part 6: Discussion

Part 1:

we captured the images through the webcam

Part 2:

Secondly, we then go ahead to get the filtered images using different filter to extract the cone out of the whole image

Part 3:

Blob analysis is carried out on the output images of the thresholding process. In this, the boundaries for the desired areas are detected and any black spots are changed to white ones. The unwanted areas are blacked out. The centroids of all desired areas are calculated and marked with an asterisk on the map. The overhead part used the previous information on centroid position and calculated the angle at which it is in the 60-degree field of view(FOV) of the camera. As per the condition required, the code adapts to the image resolution for different camera parameters. The tested code worked efficiently with the training dataset for 1-cone and 2-cone images.

Part 4:

Blob analysis and thresholding are used on the test images provided for the experiment. The code from part 3 to part 4 was not altered as the images were successfully recognized and processed. No changes had to be made to the tolerances or the HSV values being used. No difficulties were encountered in this part.

Part 5:

In this section, we were required to manually click images of different cone configurations. We placed the cones at different distances and angles. The same process was repeated as in the case of part 4 and received satisfactory results. We missed the fact that there were more cones in the surroundings as well. Our code detected them as well.

Difficulties

1. We faced problems coming up with the algorithm to detect multiple desired areas(cones) and centroids for them.

2. We also couldn't get accurate thresholding in part 2 due to some parts of the cone being light orange color due to the room light being reflected from it. This created a black patch in the output. Although this was taken care of in part 3.

9 Part 7: Student Feedback

A). In this Lab-3 (Image Processing for Vision-Based Tasks) we have learnt how to process the images from a camera and allocate things to robot for vision based task that a local robot must perform

1. By detecting the color using the Color Thresholder App. Then we determine the centroid of the cones that are present in the field of view, and we then determine the overhead view of the cones so that mobile robot can sense the cones on the floor.

2. We will train the algorithm to detect the cones using train data a set of cones using the Color Thresholder App, so that it will detect the cones which are orange in color. We can understand how to detect the colors and train the model to detect the specific colors and determine the centroid of the cone, similarly we can detect the overhead view of the robot to determine where the cones are located in the plane. This lab is helpful for us to understand the image processing for vision based tasks.

B). 1. We faced problems coming up with the algorithm to detect multiple desired areas(cones) and centroids for them.

2. We also couldn't get accurate thresholding in part 2 due to some parts of the cone being light orange color due to the room light being reflected from it. This created a black patch in the output. Although this was taken care of in part 3.

C). It took us 2 full days to understand the concept and proceed with the matlab code and to get the results. All thanks to our batchmates we did a great job in finally completing the tasks for the Lab 3.

10 Appendices

5.3 Appendix A: Code

5.3.1 Part 2

5.3.1.1 Cone Threshold Function

```
function [BW,maskedRGBImage] = coneThreshold_R8(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%  [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB using
%  auto-generated code from the colorThresholder app. The colorspace and
%  range for each channel of the colorspace were set within the app. The
%  segmentation mask is returned in BW, and a composite of the mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 21-Nov-2022
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.030;
channel1Max = 0.119;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.193;
channel2Max = 1.000;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.833;
channel3Max = 1.000;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

5.3.2 Part 3(a)

5.3.2.1 Blob Function

```
%% ME 598
% Blob Analysis Demo
clear all
close all
clc
% Read image into MATLAB
CD = imread('1Cones3_output.bmp');
figure(1)
imshow(CD)

title('Original BW')
% Open image by removing connected pixel regions less than 1300 pixels in
% size
CDfiltered = bwareaopen(CD, 400);
figure(2)
imshow(CDfiltered)
title('Opened')
%%
% Fill image
CDfilled = imfill(CDfiltered, 'holes');
figure(3)
imshow(CDfilled)
title('Opened + Filled')
%%
% Label connected components
L = bwlabel(CDfilled);
% Calculate region properties for connected components
s = regionprops(L);
% Concatenate an array of all the region's 'area' values
areas = cat(1, s.Area);
% Concatenate an array of all the region's 'centroid' values
centroids = cat(1, s.Centroid);
centroidsX = [];
centroidsY = [];
for i = 1:length(areas)
% Find the index in the 'areas' array corresponding to max_area
idx = find(areas == areas(i));
% Get the centroid value for the region with the largest area
centroidsX = [centroidsX centroids(idx,1)];
centroidsY = [centroidsY centroids(idx,2)];
end
```

```
% Find the index in the 'areas' array corresponding to max_area
idx = find(areas);
% Select the connected component corresponding to this region
BW2 = ismember(L, idx);

% Plot the image of the largest connected region
figure(4)
imshow(BW2)
hold on
% Plot a blue star in centroid of region
for i = 1:length(centroidsX)
plot(centroidsX(i), centroidsY(i), 'b*')
hold on
end
hold off

[nr,nc] = size(CDfilled);
ImageCenterX = nc/2;
ImageCenterY = nr/2;
fov = 60;
angle = [];
r = [];
max_area_cone = areas(find(areas == max(areas)));
for i = 1:length(centroidsX)
    xdiff = ImageCenterX - centroidsX(i);
    angle = [angle xdiff*fov/nc];
    if centroidsX(i) == centroidsX(find(areas == max(areas)))
        r = [r 1];
    else
        comp_area =areas(i)/max_area_cone;
        r = [r 1+(2-comp_area*2)];
    end
end

overheadView(angle,r,5);
```