Colour Segmentation using Gaussian Mixture Models

Rachit Bhargava and Nikolay A. Atanasov

***Abstract*— This work addresses the task of segmenting an object using Gaussian Mixture Models and finding its distance from the camera. In particular, the task focused on estimating the bounding box for a red barrel in an image and estimating its distance in the world frame. Using training data, a Gaussian Mixture Model (GMMs) is trained for each color. These models are used to classify each pixel in the image and identify those pixels whose color intensities closely match that of the red barrel. This results in a binary image. The bounding box around the red barrel is attained by applying certain image processing techniques and shape filters on the binary image data. Using the height and width of the barrel in both the image and world and the intrinsic parameters (attained from training data), the distance of the barrel to the camera is estimated. The effectiveness of the proposed framework is exhibited by implementation on both the training and testing data.**

I. INTRODUCTION

In image processing, segmentation is a difficult problem to solve. An important aspect of this problem is picking a threshold that allows separation between the intensities of the pixels of the object of interest and the unwanted pixels. A trivial solution to this problem is to estimate some linear separation between pixel intensity data. This involves discarding pixels which lie above (or below) a certain intensity in each of the three channels (RGB or HSV) as unwanted pixels. A linear separation is difficult to find and may not even exist if the data is too complex. Instead of manually searching for a linear separator it would be favorable to learn the model of a particular class of pixels. An approach to this would be to model each class as a Gaussian. However, such an assumption could be erroneous since in most cases data is not uni modal. If such is the case, it is better to model the data for each class as a mixture of Gaussians with a membership weight associated to each Gaussian. A pixel is categorized as a particular color if it gives the maximum probability for the GMM for that color in comparison to the other colors. The following section describes the process of training the GMM for each class (color). It also discusses how each pixel is tested for classification. This is followed by an explanation of the image processing techniques implemented to extract the red barrel from the image and calculate its distance from the camera. The results from implementation on both training and testing data are accumulated in the Results section. The conclusions and future work are summarized in the final section.

II. TRAINING AND TESTING FOR CLASSIFICATION

This section discusses the procedure for extracting the training data and training the Gaussian Mixture Models for each class. It also discusses the procedure to test each pixel using the trained models to segment the barrel in the image.

*This work is a project as part of ESE 650 Learning in Robotics at the University of Pennsylvania

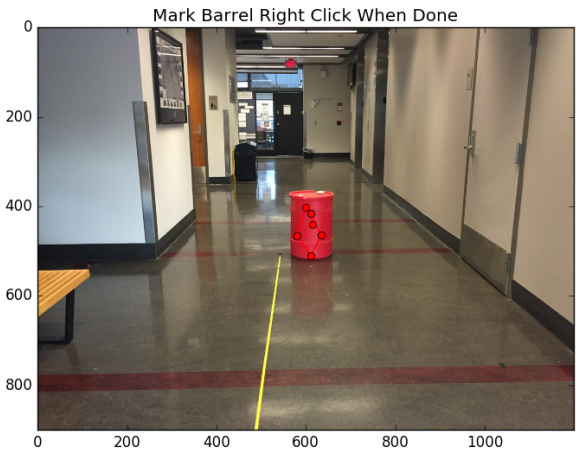


Fig. 1: Marking ROIs in the barrel to extract pixel intensity data

A. Extracting data for training

The first task was to decide the number of classes of colors for which the GMM needs to be trained. The first class, Barrel Red is the class of pixels that need to be segmented i.e. the desired pixels. To encompass all the undesired pixels intensity data, four other classes namely Brown, Black, Yellow and Other Red were chosen. These were chosen owing to the similarity of the pixel intensities of these class of colors with the barrel red intensities. It was assumed that the classifier would work well if it was able to distinguish the barrel red from these class of colors.

The next step involved extracting the pixel intensity data from the training images. This was done using the methods in the `roipoly` class suggested in the project handout. Using the `roipoly()` method, a polygon region of interest (ROI) was drawn in each image for each class as shown in Fig. 1. The intensities of the pixels within each ROI was stored using a binary mask generated using the `getMask()` method. In such a manner, pixel intensities for each class in each of the training images was extracted and stored for training. Upon plotting the RGB data from each class, it was observed that data could be separated using RGB data. By observation, picking 3 clusters for each class appeared to be an ideal choice. This choice proved to be valid since it gave acceptable accuracy during the testing phase.

B. Training a Gaussian Mixture Model

The next task involved training a Gaussian Mixture Model for each of the classes using the pixel intensity data for each class attained in the previous step. It involved estimating the membership weights π_i, mean μ_i and the associated covariance matrix Σ_i for each cluster i for each class of data using the Expectation Maximization algorithm. As mentioned before, each class was modeled using 3 clusters. For generalization assume number of clusters is n.

The first challenge was choosing a method to initialize the data. The mean μ_i for all the n clusters was initialized by uniformly picking n random points from the training data. The covariance matrix Σ_i for all the clusters was assumed to be equal and equivalent to a diagonal matrix where the diagonal entries contained the variance in the red, green and blue intensities for the entire training data. Finally, it was assumed, for initialization, a point is equally probable to belong in any cluster. Hence $\pi_i = 1/n$ for each cluster. Additionally, the initial log likelihood cost function was initialized to infinity . After the initialization the EM loop begins.

1) Expectation Step (E-step): The E-Step involves estimating the membership probabilities $r(z_i|x_j)$ for each of the clusters i for each $j \in D$, where D is the set the set of all training data points. $r(z_i|x_j)$ gives the probability of belonging to a particular cluster given the parameters of the Gaussian (cluster) and the membership weights of each cluster. It is calculated as:

$$r(z_i|x_j) = \frac{\pi_i \phi(x_j; \mu_i, \Sigma_i)}{\sum_i \pi_i \phi(x_j; \mu_i, \Sigma_i)}$$

for cluster i and point x_j in the training data. Here the function ϕ is the Gaussian function:

$$\phi(x_j; \mu_i, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^3 |\Sigma_i|}} e^{(x_j - \mu_i)^T \Sigma_i^{-1} (x_j - \mu_i)}$$

These membership probabilities are used to update the parameters in the M-step.

2) Maximization Step (M-step): This step involves estimating the updated values of the parameters. This is done by the following equations:

$$\pi_i^{t+1} = \frac{\sum_j r^t(z_i|x_j)}{|D|}$$

$$\mu_i^{t+1} = \frac{\sum_j r^t(z_i|x_j) x_j}{\sum_j r^t(z_i|x_j)}$$

$$\Sigma_i^{t+1} = \frac{\sum_j r^t(z_i|x_j)(x_j - \mu_i)(x_j - \mu_i)^T}{\sum_j r^t(z_i|x_j)}$$

Since there was enough data available for training it was decided that, a full covariance matrix was calculated .

In such a manner the parameters are updated in each loop. After each update the cumulative log likelihood function J is calculated using the following equation:

$$J = \sum_j \log(\sum_i \pi_i \phi(x_j; \mu_i, \Sigma_i))$$

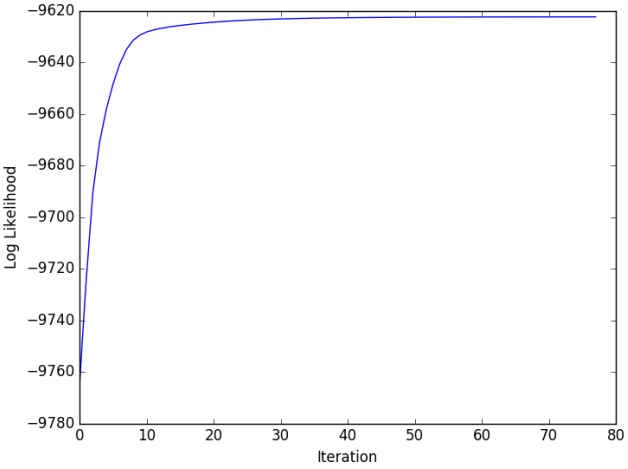


Fig. 2: Plot of log likelihood vs number of iterations

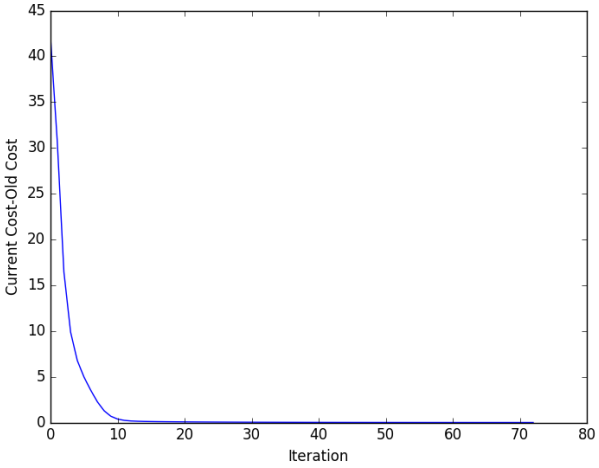


Fig. 3: Plot showing convergence of the EM algorithm

Note that the EM algorithm maximizes this function J as shown in Fig 2. The EM loop terminates when the above function converges within a threshold of $\epsilon = 0.001$ as shown in Fig 3. This entire process is repeated for each of the classes.

C. Testing and Classification for Color Segmentation

Once the models for all the classes are estimated, the testing phase is initiated. Each pixel of the testing image is tested for each class. For each pixel the probability of belonging to a particular class is estimated as

$$P(Class = K|x_j) \propto P(x_j|Class = K)) = \sum_i \pi_i \phi(x_j; \mu_i, \Sigma_i)$$

where i is the cluster in the model for class K. The pixel is assigned the class for which it has the maximum probability $P(x_j|Class = K)$. Note that, it was assumed that the prior for each class is equal and that the probability of each data point is the same.

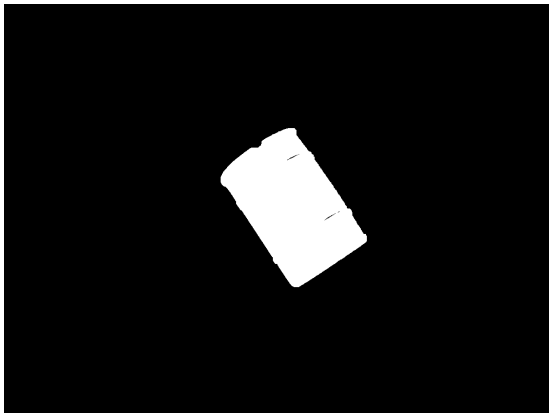


Fig. 4: Binary image attained after semgentation with GMMs

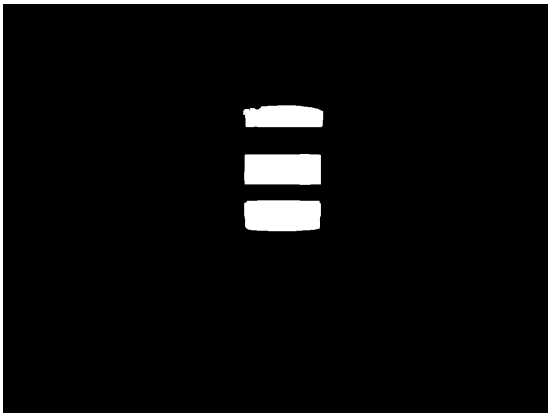


Fig. 5: Disconnected Blobs due to partial occlusion

An important point to note about this technique is that an assumption of conditional independence (Naive Bayes assumption) was assumed between the various classes. The training of one class is unaffected by the training data from other classes. This technique is also know as GMMS with Bayesian inference.

Using the class information for each pixel, a binary image is generated where those pixels which belong to the red barrel class are assigned the intensity 255 (white) and the rest are assigned the intensity 0 (black) as shown in Fig. 4. Once the binary image has been obtained, it undergoes image processing to identify the barrel in the image.

III. Image Processing for Segmentation

This section discusses the image processing techniques implemented to find the barrel in the image using the binary image attained after classifying each pixel. In this project, image processing was done with the focus on identifying multiple barrels in the image while dealing with partial occlusion.

A. Contour Extraction and Noise Removal

Images usually contain a lot of noise. Some of the noisy pixels may not be filtered out after testing. One effective method of removing noise in binary images is through morphological operations. A common technique for this task is the closing operation. A kernel of 10x10 was used for this task. First the image was eroded using this kernel and then dilated. After applying this operation, the `cv2.findContours()` method in the OpenCV library was used to extract all possible contours in the image. A number of contours were discarded by applying a threshold on the image moment m_{00}, the number of white pixels in each contour. A threshold of 300 was used in this project.

B. Contour Clustering

An important challenge to be dealt with is that of partial occlusion. The loss of data caused by partial occlusion makes it harder to find the barrel. This is because the blob's shape no longer resembles the barrel shape. It could also happen that the barrel may be segmented as disconnected blobs in the binary image as shown in Fig.5. In an attempt to overcome this issue, the blobs were clustered using the distances between the corner points of contours of different blobs. A `contours near()` method was implemented that judged two contours to be close if there existed at least one corner point in one contour that was within a certain distance of a corner point of the other contour. The threshold chosen in this project was 70 pixels. All the contours that were close to one another were clustered together. A convex hull was then found taking all the points in the cluster of contours as input. Using the `cv2.minAreaRect()` method in the OpenCV library, the minimum rotated rectangular area around the convex hull was found.

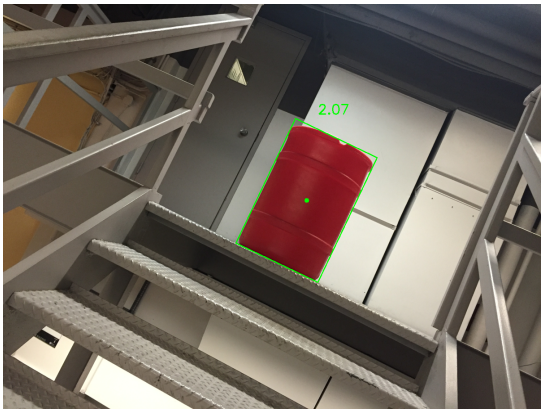
C. Additional shape filters

Once the minimum rectangular area was found, certain filters were applied on the rotated rectangular boxes. An assumption was made that the barrel would not be rotated by a great extent. So any bounding box that was tilted more that 20 degrees was discarded. Additionally, the constraint that the height would be at least 1.2 times greater than the width was applied. Finally a threshold was imposed on the area of the bounding box area to eliminate bounding boxes that were extremely small. Here it was chosen as 2500 pixels squared.

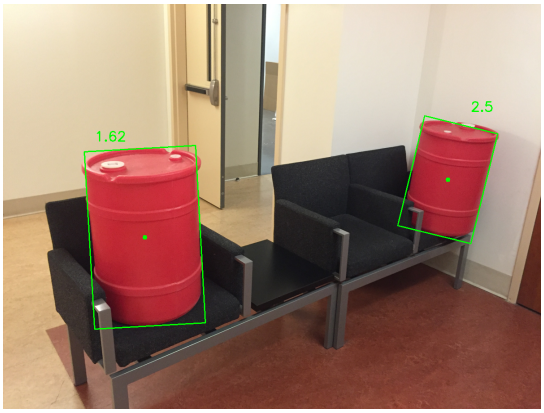
The bounding boxes attained at the end are classified as the barrel. The next step involved getting an estimate of the distance of the barrel from the camera using the intrinsic parameters and the height and width of the barrel in the image.

IV. Depth Estimation

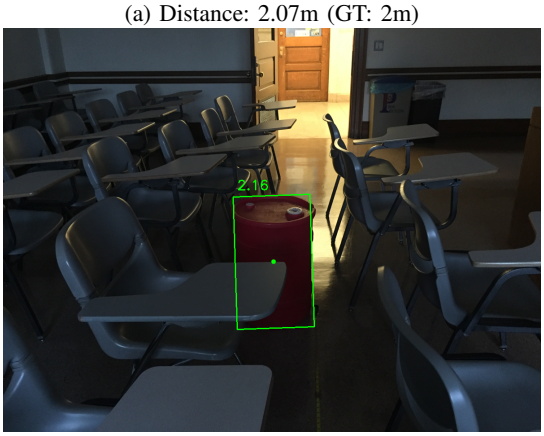
The final task of the project was to estimate the distance of barrel from the camera. This required the height and width of the barrel in world , the height and width of the barrel in the image and the intrinsic parameters, in particular the focal length parameters f_X and f_y in pixels. In the easy data, the intrinsic parameters was unknown but the ground truth of the distance of the barrel from the camera was given for each training image. Given the width w of the barrel in the image, height h of the barrel in the image, width W of the barrel in



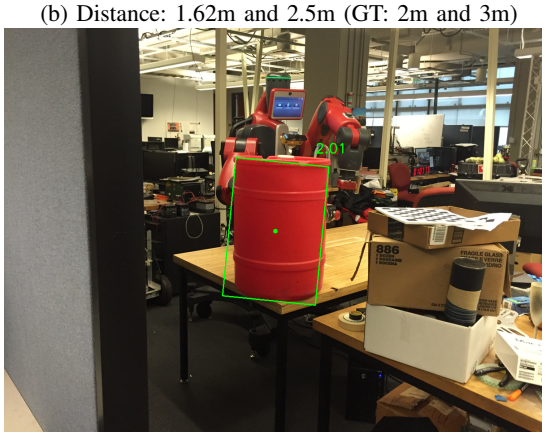
(a) Distance: 2.07m (GT: 2m)



(b) Distance: 1.62m and 2.5m (GT: 2m and 3m)



(c) Distance: 2.16m (GT: 2m)



(d) Distance: 2.01m (GT: 2m)

Fig. 6: Results of implementation of algortihm on training data

the world, height H of the barrel in the world and distance to the barrel in the world Z, the horizontal and vertical focal lengths f_x and f_y can be estimated as:

$$f_X = Z\frac{w}{W}$$

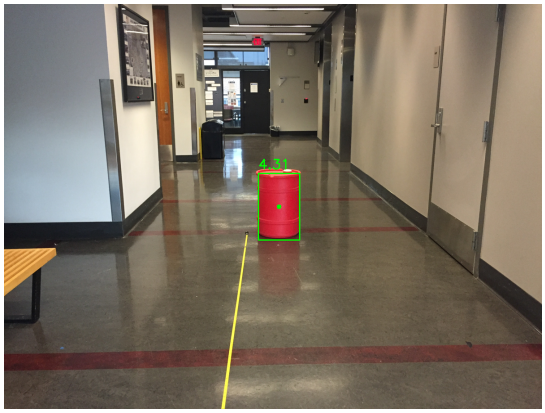
$$f_y = Z\frac{h}{H}$$

To calculate the focal lengths those training images were picked where the barrel was standing straight (there was no rotation) and a precise bounding box was extracted around the barrel. Barrel width and height values were given in the problem statement as 40cm and 57 cm respectively. From each of the images, the values of f_x and f_y were estimated. The final values of f_x and f_y were calculated by averaging over these estimates. Hence when the test image is passed and its bounding box is found, the depth is calculated using the equation $Z = f_x\frac{W}{w}$ and $Z = f_y\frac{H}{h}$. The minimum of these values is taken as the final distance estimate to account for anomalies in the values of h and w due to occlusion or error in bounding box estimation.

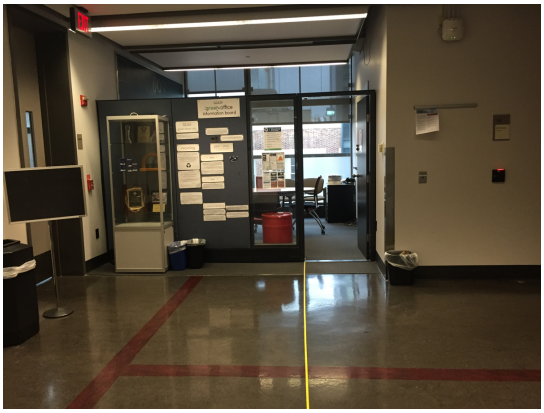
V. Results

The results from implementation of the proposed algorithm are shown in Fig. 6 an 7 . The depth estimates have been included in the captions below the images. Results of implementation of the algorithm on the training data (Cross Validation) are shown in Fig. 6. In Fig. 6a, it can be seen that the algorithm is able to account for rotated barrels and make an accurate bounding box around them. Fig. 6b exhibits the ability of the algorithm to detect multiple barrels in the scene at different distances. Fig. 6c highlights the ability to extract the bounding box accurately even in low lighting and partial occlusion. Fig 6d. exhibits the ability of the algorithm to accurately find the barrel even with the similarly colored Baxter robot in the background. It can also be observed that the depth estimate was relatively accurate.

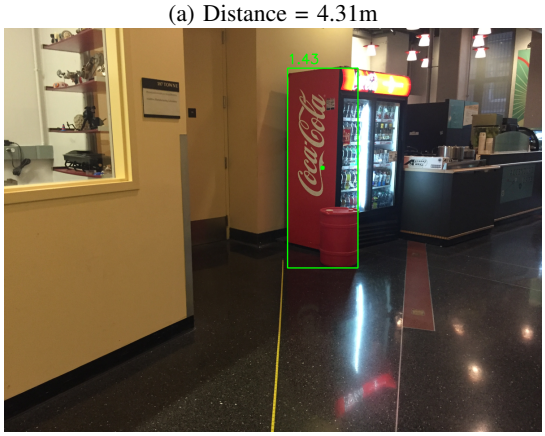
Fig. 7 exhibits the results attained on implementation of the algorithm on the test data. It works particularly well in simple cases like 7a,7c,7g and 7j. It produced a good result for the tilted barrel in fig 7d. It even performed well in partially occluded cases such as 7h and 7i. However it failed in 3 cases. In Fig. 7b the barrel was not detected since the aspect ratio condition was not satisfied by the bounding box around the barrel. Such an aspect ratio was not accounted for in the algorithm. In Fig. 7c the color of the barrel and the Coca Cola board are too similar. They were both classified by the classifier as barrel red. Thus they could not be distinguished in the binary image. Finally in Fig. 7f, the pixels that were part of the reflection of the barrel were also classified as barrel pixels. The contours from the



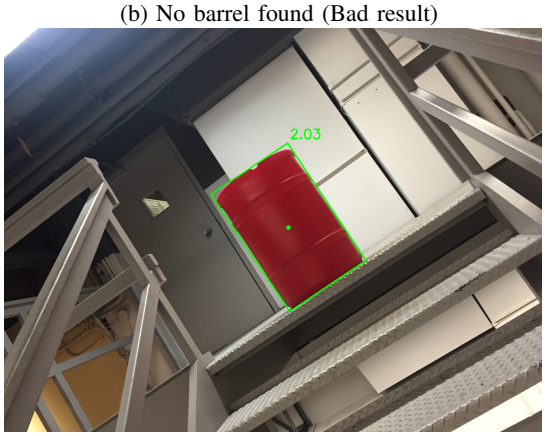
(a) Distance = 4.31m



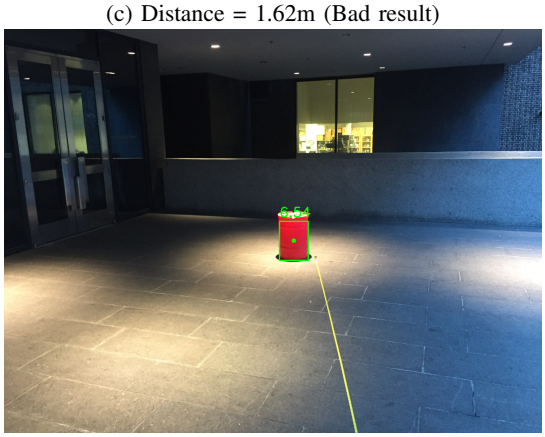
(b) No barrel found (Bad result)



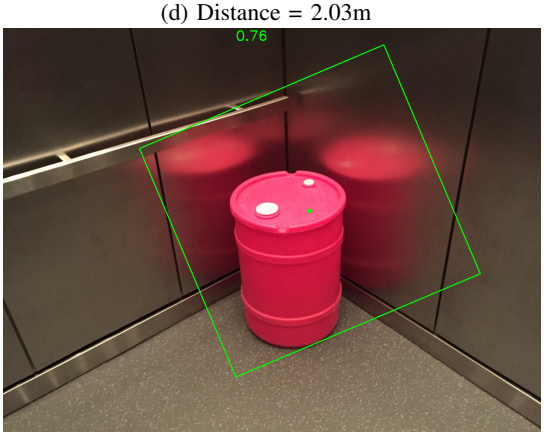
(c) Distance = 1.62m (Bad result)



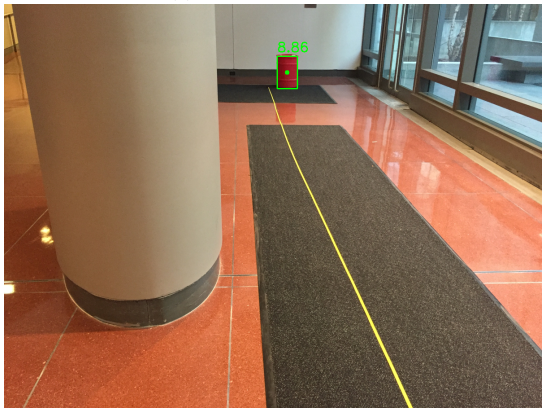
(d) Distance = 2.03m



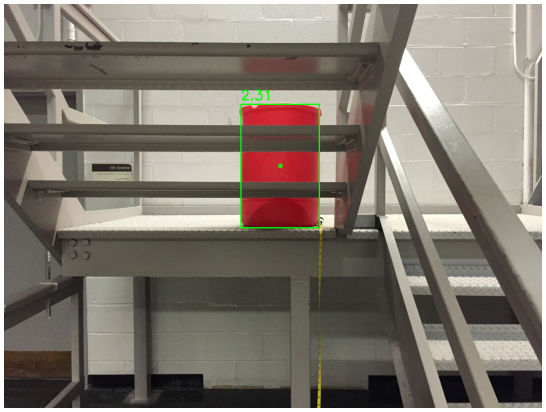
(e) Distance = 6.54m



(f) Distance = 0.76m (Bad result)

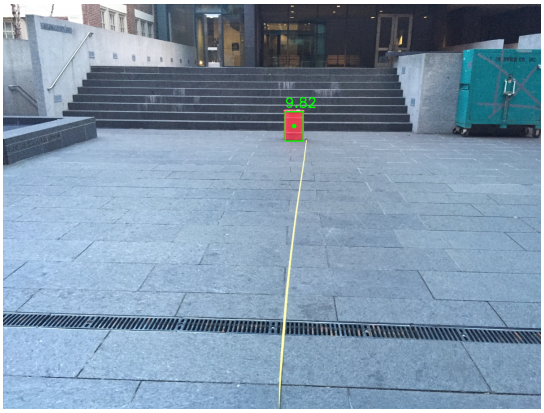


(g) Distance = 8.86m



(h) Distance = 2.31m

(i) Distance = 6.38m



(j) Distance = 9.82m

Fig. 7: Results of implementation of algortihm on training data

2 reflections and the barrel were clustered by the contour clustering algorithm resulting in such a big bounding box around the barrel.

VI. CONCLUSIONS

In this paper, an algorithm to detect a red barrel in an image by color segmentation using Gaussian Mixture Models has been discussed. It also discusses the image processing techniques implemented to extract the bounding box in the segmented image. The algorithm works pretty well for most cases. But, the algorithm since has not been tested for different color spaces and different number of clusters. It might work well by tuning through cross validation. Implementation of shape filters during image processing could also be improved. Work can be done on picking optimal values of parameters of the thresholds. Alternative methods of dealing with partial occlusion can also be explored.

ACKNOWLEDGMENT

I want to thank Dr. Atanasov and the TAs for helping me with my doubts on Piazza.

REFERENCES

[1] Slides for ESE 650 from Dr. Atanasov