ENPM673 Project3
Perception of Autonomous Robot
Underwater Buoy Detection using Gaussian Mixture models

By (Group 2)
B. Sai Praveen(DirID: spraveen)
Preyash Parikh(DirID: pparikh)
Ajinkya Parwekar(DirID: ajinkyap)

April 6, 2020

INTRODUCTION

The objective of this project is to obtain a tight segmentation of three buoys which shows three different colors, namely, yellow, orange and green, by applying a tight contour around each buoy using Gaussian Mixture Models and Expectation Maximization techniques.

1 CREATING DATASET

From multiple frames, training data is cropped out. Our objective is to have different shades of buoy so that the model has these clusters of pixels.

In this project, we have implemented mouse click event to obtain the pixel location and later we used this to crop an image and save it. To have a proper dataset, we have taken different regions of buoy.

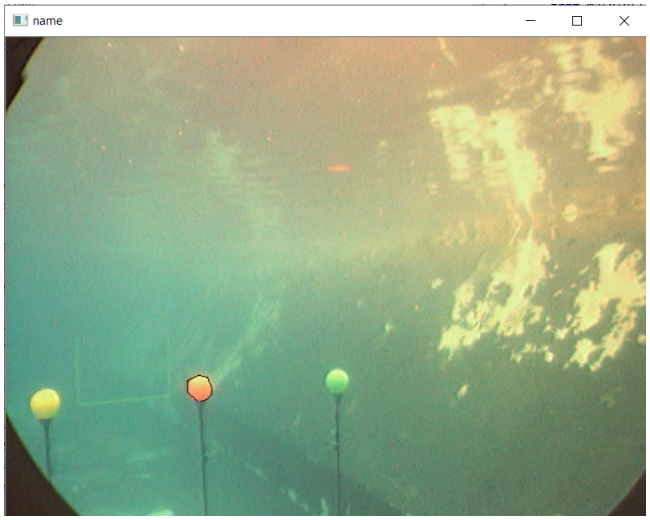

Figure 1: Sample Image of buoys

Figure 2: Orange buoy1

1D Gaussian formula is given by,

$$\mathcal{N}(X|\mu,\, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

Here, μ = Mean, $sigma$ = Variance

3D Gaussian formula is given by,

$$\mathcal{N}(X|\mu, \sum) = \frac{1}{(2\pi|\sum|)^{1/2}} exp\{\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)\}$$

Here, μ = Mean, $\sum$ = Covariance

In complete dataset, when we have a different number of clusters which describes one feature, it is more prudent to take a multivariate gaussian to model the data across the three sets. To get better understanding about how the total dataset is spread we use multivariate Guassian.

This is what we have implemented in our project. In order to detect the probabilities of a pixel lying in a cluster and to estimate the means and co-variances, we use the Expectation Maximization algorithm.

2 EXPECTATION MAXIMIZATION ALGORITHM

An elegant and powerful method for finding maximum likelihood solutions for models with latent variables is called the Expectation Maximization algorithm.

1) STEP 1
Initialize the means and covariances with some random values. The covariance matrix is of size d^2 where 'd' is the number of dimensions of the Gaussian. In general, we can take any random value but it is a good practice to overall mean of the entire dataset or individual mean of clusters.

2) E STEP

To combine Gaussian, the formula is given by,

$$P(x) = \sum_{k=1}^{k} \pi_k [\mathcal{N}(x|\mu_k, \sum_k)]$$

Here, k = number of gaussians and π = mixing coefficient weightage for each gaussian distribution.

$$0 < \pi_k < 1$$

,

$$\sum_{k=1}^{k} \pi_k = 1$$

To Evaluate the responsibilities using the current parameter values, we use Bayes rule, which is given as follows,

$$\gamma_k(x) = \frac{\pi_k \mathcal{N}(x|\mu_k, \sum_k}{\sum_{j=1}^{k} \pi_j \mathcal{N}(x|\mu_j, \sum_j}$$

Here, γ = latent variable.

3) M STEP

Re-estimating the parameter using the current responsibilities. i.e,. calculating the new updated means and covariances.

1)

$$\mu_j = \frac{\sum_{n=1}^{N} \gamma_j(x_n) X_n}{\sum_{n=1}^{N} \gamma_j(X_n)}$$

2)

$$\sum_j = \frac{\sum_{n=1}^{N} \gamma_j(X_n)(X_n - \mu_j)(x_n - \mu_j)^T}{\sum_{n=1} N\gamma_j(X_n)}$$

3)

$$\pi_j = \frac{1}{N} \sum_{n=1} N\gamma_j(X_n)$$

4) EVALUATE LOG LIKELIHOOD

$$\ln P(X|\mu, \sum, \pi) = \sum_{n=1}^{N} \ln \{\sum_{k=1}^{K} \pi_k N(X_n|\mu_k, \sum_k)\}$$

The log likelihood will iterate through certain iterations until it satisfies certain criteria. The final values are the estimated means (μ) and variances (σ) of the given data set.

3 DATA GENERATION AND EVALUATION OF MEAN AND STANDARD DEVIATION USING GMM AND EM

The given ground truth values are as follows:
Means: 0, 3, 6
Standard deviations: 2, 0.5, 3

The EM algorithm is executed for K = 3. The estimated means and covariance are shown here,

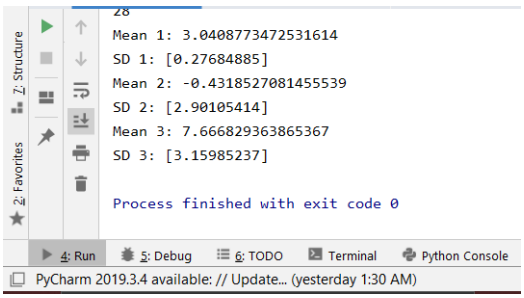

Figure 3: Means and Covariances

The estimated values are near to ground truth values. This shows that ground truth values are covered.

4 BUOY DETECTION - 3D GAUSSIAN

Pipeline is divided into testing and training. To train any given model, we have implemented 3D Gaussian and used the EM algorithm to find the points lying in the clusters.

We begin with taking images from training data set and then cropping them in order to have actual crop of the buoy. After this, we plot histogram of each channel using inbuilt function, cv2.calcHist(). From the histogram, we can add up the number of clusters from each channel's histogram and decide the total number of clusters (K), for the buoy. This denotes the number of Gaussians we need to use to detect this buoy in frame. Since this is the 3D case, we send all the three channels. The next step is to convert this each channel into a column of its constituents. Using random values of proper dimension the parameters are then initialized and the EM algorithm is evaluated. The algorithm runs until it reaches the maximum number of iterations specified (50).

5 CONTOUR DETECTION - 3D GAUSSIAN

After performing thresholding on an image, we applied dilation operation using an inbuilt function to have thick regions around the buoy. After finding the contours, we give these set of points as input to the inbuilt-function "minEnclosingCircle()" function to get the approximate center and radius of a circle. We then plot the circle around the detected buoy using cv2.Circle() function.

The output images are shown here and video is written to file as well.

Figure 4: Detected buoys

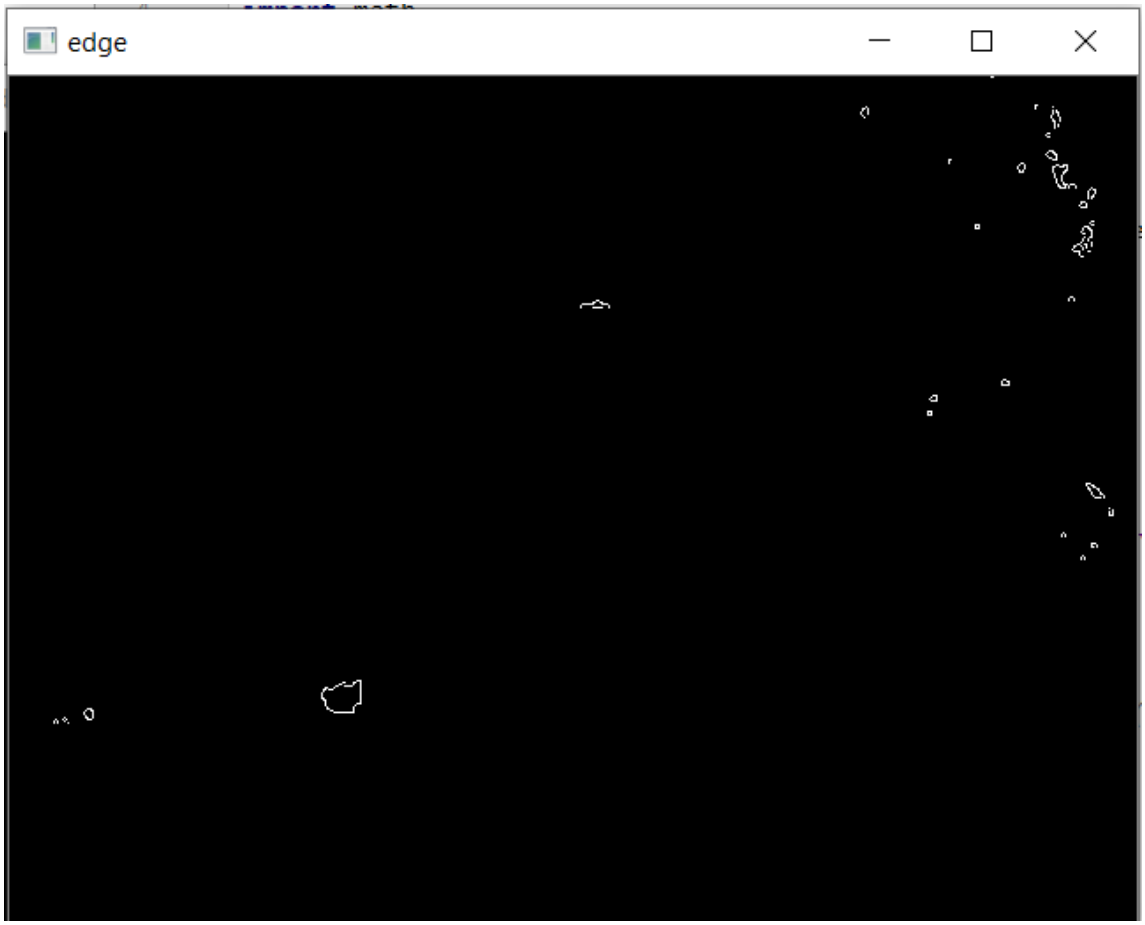

Figure 5: Threshold Image

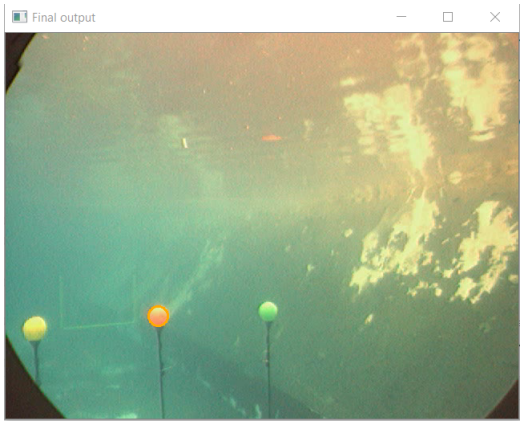

Figure 6: Orange Detected Image

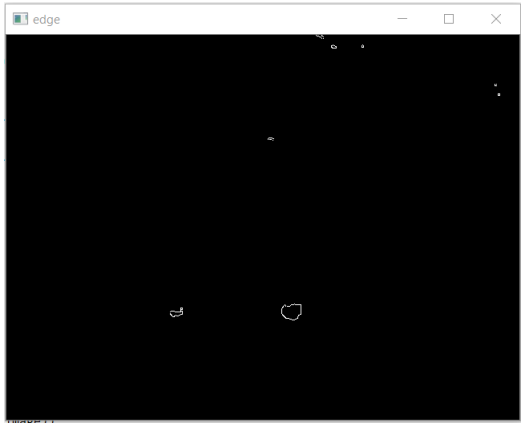

Figure 7: Orange Threshold Image

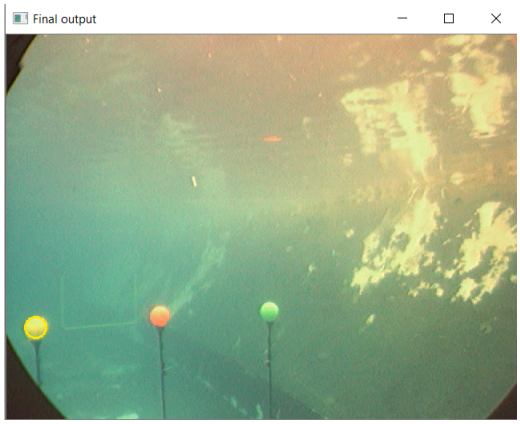

Figure 8: Yellow Detected Image

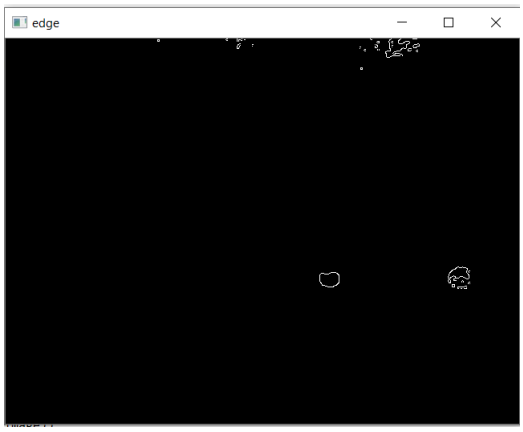

Figure 9: Yellow Threshold Image

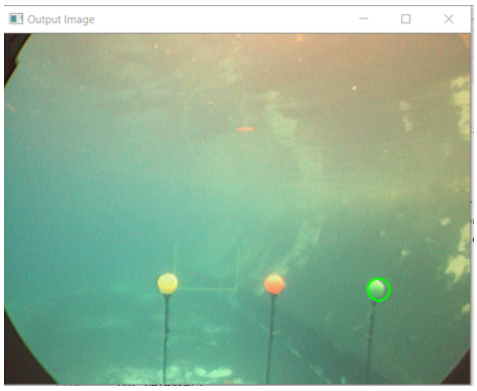

Figure 10: Green Detected Image

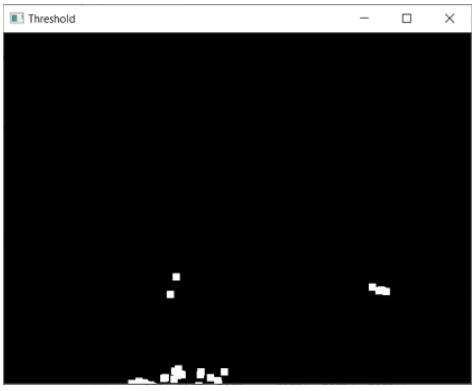

Figure 11: Green Threshold Image

6 BUOY DETECTION - 1D GAUSSIAN

We start by taking in images from the folder containing the training set images. These images are further cropped to get a close crop of the actual buoy itself. We then plot the histograms of each

channel of this image set to get an idea of the number of clusters present. K denotes the number of gaussians we need to use to detect this buoy in frame. Since this is the 1D case, we arrange the extracted channel(s) in a single column. For green buoy, we only use the green channel, for orange we use the red channel and for yellow, we stack the data in both red and green channels to form one column. Here, we have used inbuilt function to calculate the mean and standard deviation of a dataset. Then evaluated univariate Gaussian.

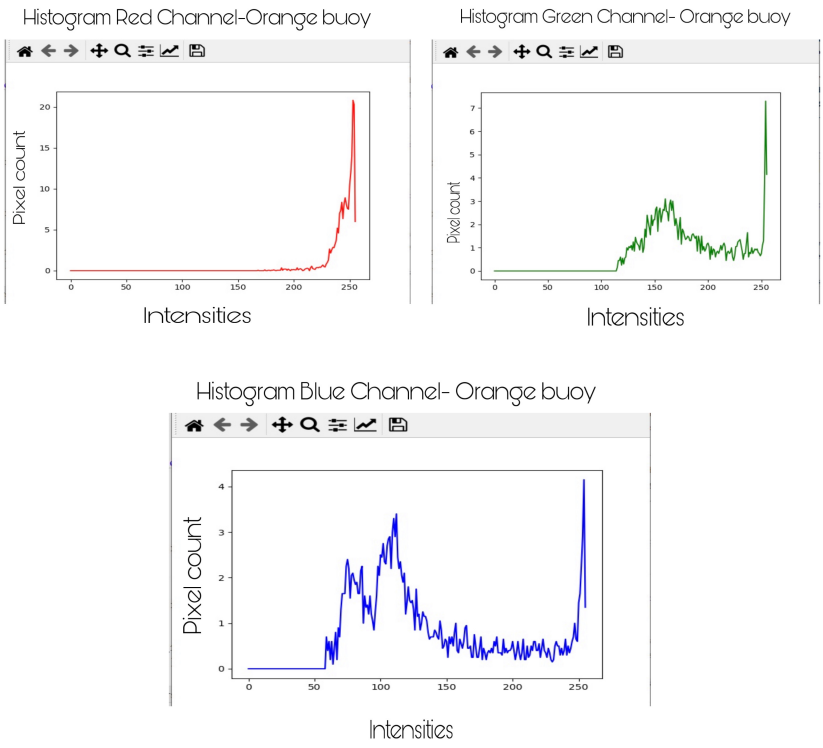

Figure 12: Histogram for each channel for Orange data set

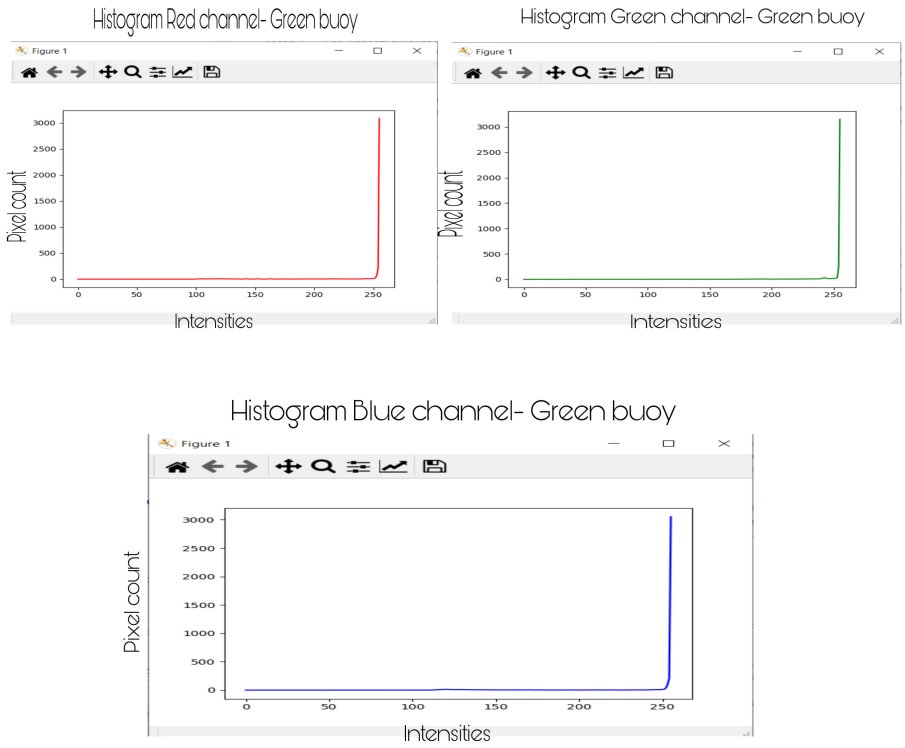

Figure 13: Histogram for each channel for Green data set

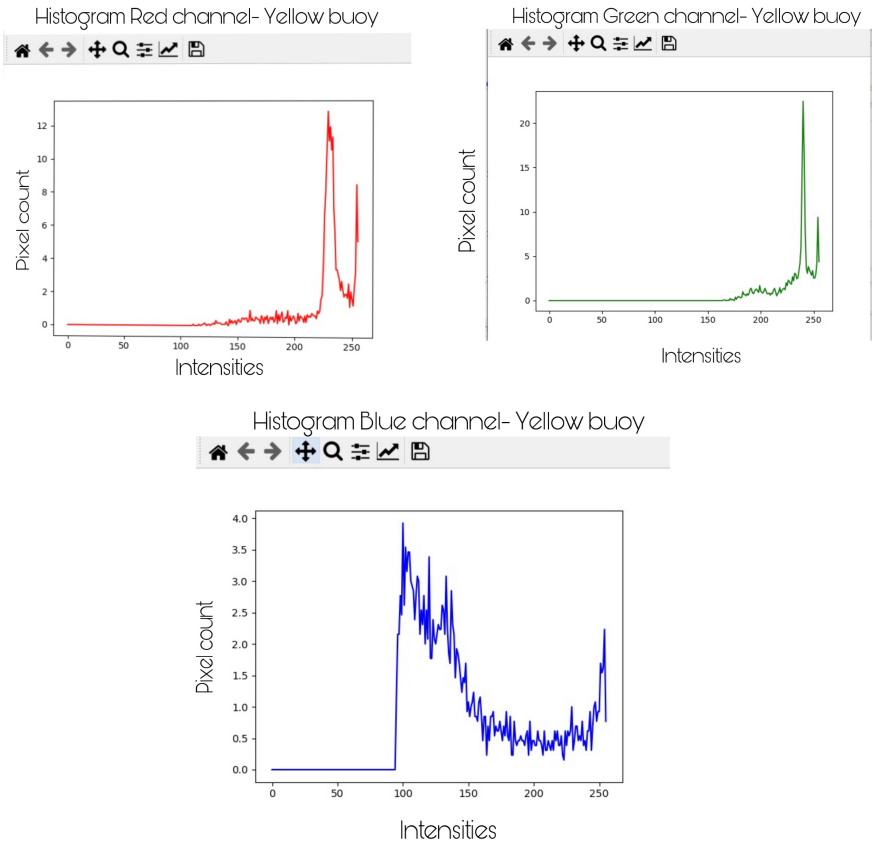

Figure 14: Histogram for each channel for Yellow data set

7 CONTOUR DETECTION - 1D GAUSSIAN

For orange and yellow buoy, we blur the image and detect edges using Canny edge detection. Then, we use findContours() function to do this. Once the contours are detected, we sort them. After that, we give these set of points as input to the minEnclosingCircle() function to get the approximate center and radius of a circle. For green buoy, after performing thresholding on an image, we applied dilation operation using an inbuilt function to have thick regions around the buoy. After finding the contours, we give these set of points as input to the inbuilt-function "minEnclosingCircle()" function to get the approximate center and radius of a circle. We then plot the circle around the detected buoy using cv2.Circle() function.

8 OUTPUTS

The following is the google drive link for the ouput videos.
https://drive.google.com/drive/folders/1Jvgha4cNVgWZVZIpP-lyhBT2ijB2UrGE?usp=sharing

9 CONCLUSION

From the output videos, it was observable that for 1D Gaussian, the detection of each buoy was not satisfactory. In case of Green buoy, the green color is predominant in the background as well as in the yellow areas including the yellow buoy. Hence, during contour detection there were times when the contour detects background which matches criteria that of green buoy. This happens because, all channels are not trained together.
In 3D case, the detection is robust because all the channels of the green buoy are trained together for the number of clusters specified. Thus, the likelihood of finding pixels that belong to green buoy increases.
Same results were observed for Orange and Yellow buoy.

10 CHALLENGES FACED

1) Initially the outputs were not satisfactory even after using appropriate number of Gaussians. We fixed it by adjusting various values.
2) For satisfactory results, we had to try various filters to denoise the image before detecting contours.
3) We were not able to understand the concept of channel and how we can accomodate multivariate and unvariate Gaussian into channels.

11 REFERENCES

1) https://matplotlib.org/users/event_handling.html
2) https://github.com/JCardenasRdz/roipoly.py
3) http://www.cse.iitm.ac.in/%7Evplab/courses/DVP/PDF/gmm.pdf
4) http://www.rmki.kfki.hu/%7Ebanmi/elte/bishop_em.pdf
5) https://www.bogotobogo.com/python/OpenCV_Python/python_opencv3_image_histogram_calcHist.php
6) https://cmsc426.github.io/colorseg/#colorclassification