ENPM673 - Perception for Autonomous Robots

Project 3

Submitted by:

Raj Shinde (raj0407)	116852104
Shubham Sonawane (shubhams)	116808996
Prasheel Renkuntla (prasheel)	116925570

Contents

1 Problem Statement

The aim of this project is to implement color segmentation using Gaussian Mixture Models and Expectation Maximization. The conditions under which the video has been recorded makes it difficult to hardcode the color thresholds for detecting the buoys. Therefore, we train our model to detect the colors using a dataset, rather than hard-coding it.

2 Data Preparation

A total of 200 frames were extracted from the video. Using a 70:30 split, the 140 frames were randomly chosen and stored in the training set, and the remaining were stored in the test set. In order to extract the buoys from the frame, we use the event LBUTTON_DOWN. For every buoy, present in the frame, we click once in the center, and once on the periphery. This gives us 2 points, using which we construct a circle.

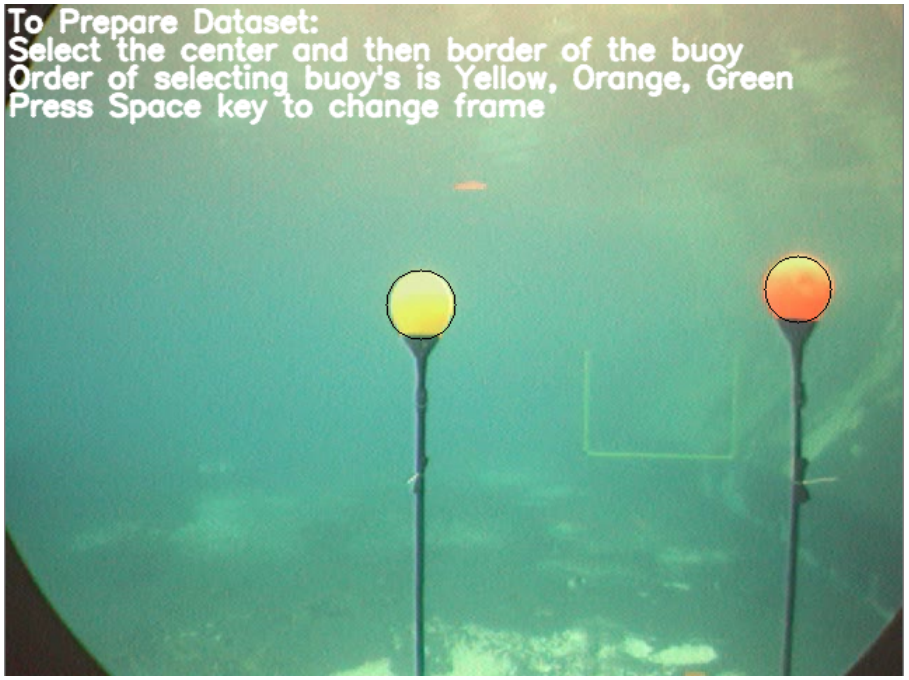


Figure 1: Circles enclosing the Buoy's

We then crop a square that encloses the previously constructed circle that contains the buoy. After that, we perform a bitwise-and operation of the square cropped image with a mask, to replace the background with black colour. The mask is

selected such that the circle enclosing buoy is covered during bitwise-and operation.

(a) Cropped Square (b) Mask (c) Bitwise-and (d) Black to white

Figure 2: Operations

The order in which the buoy's are selected is Yellow, Orange and Green. Once the operation is performed on all the buoys, we use any key to switch to new frame.

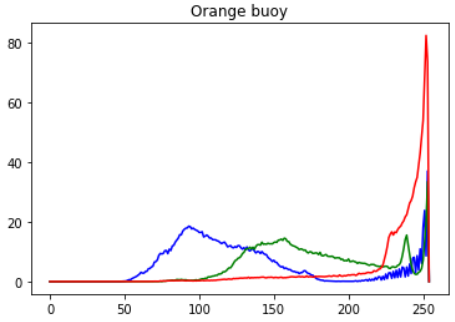
Following are the examples of the buoys cropped from the frames:

(a) Yellow (b) Orange (c) Green

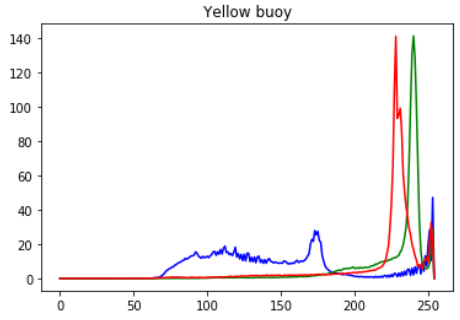
Figure 3: Final images for Dataset

3 Average Histogram

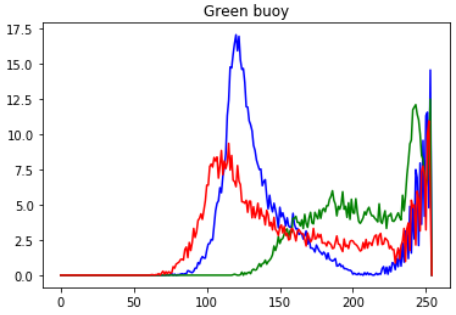
The average histograms are useful in determining the number of clusters required for the further steps of GMM and EM. We develop a function that plots the average histogram for the 3 channels (RGB) of the 3 buoys. The histograms for each image of a buoy is column appended to the previous histogram. The average of each row then forms the final histogram. Following are the results obtained after plotting the average histograms for each buoy:



(a) Average histogram for Orange buoy



(b) Average histogram for Yellow buoy



(c) Average histogram for Green buoy

We observe from the histograms that :

1. **Green Buoy**: The green and blue color have high intensities, whereas the red color has low intensity. The intensity for green color varies between 150-255.

2. **Yellow Buoy**: Histogram of yellow buoy clearly indicates the dominance of green and red color. The blue color has less intensity.

3. **Red Buoy**: The intensity for red color varies from 180 to 255. The remaining colors have low intensity.

Ideally, there should not be any other color present in the histogram fro the particular buoy (exception for yellow buoy). However, the other colors are displayed in the histogram due to the color not being completely uniform. Also, in some images, the

background (predominantly blue and green color) has also been cropped in the buoy area. However, the color corresponding to the buoy color is at the peak intensity in the histograms.

4 1-D Gaussian for detecting buoys

The gaussian equation is given by

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{1}$$

We attempt to fit the Gaussian equation across the color distribution for a particular buoy. Using the Expectation Maximization algorithm, we calculate the mean and standard deviations for the 3 channels of the buoy. Running the algorithm for 70 iterations converges the solution. Since we are dealing with 1D gaussians, we consider the peak channels for each buoy. That means for orange buoy, we consider the red channel, the green channel for green buoy and the green and red channels for the yellow buoy.

After applying the threshold on the image using the Gaussian distribution obtained, we are able to see faint outlines of the buoys being detected. Using the dilation technique preceded by canny edge detection, the buoys can be detected clearly. We have used an elliptical kernel for dilation, rather than using a square or rectangular kernel. This gave slightly better results. As these are clearly defined shapes, we apply contour detection. Finally, a circle is placed around the contour which also encloses the buoy. The center of the circle is also obtained from the minimum enclosing circle function of OpenCV.

As can be seen in the video outputs (link at the end of document), the detection of buoy using 1D gaussian does not yield good results. Often, the surroundings are detected as the buoy.

The results obtained from the 1D detection of the buoys is shown below:

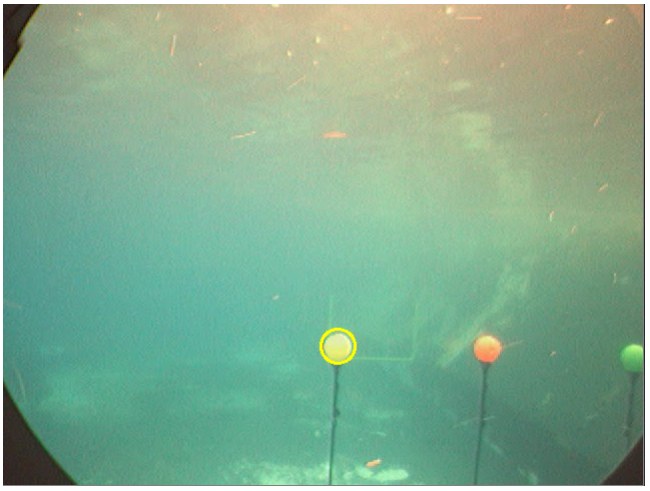


Figure 5: Detection of yellow buoy

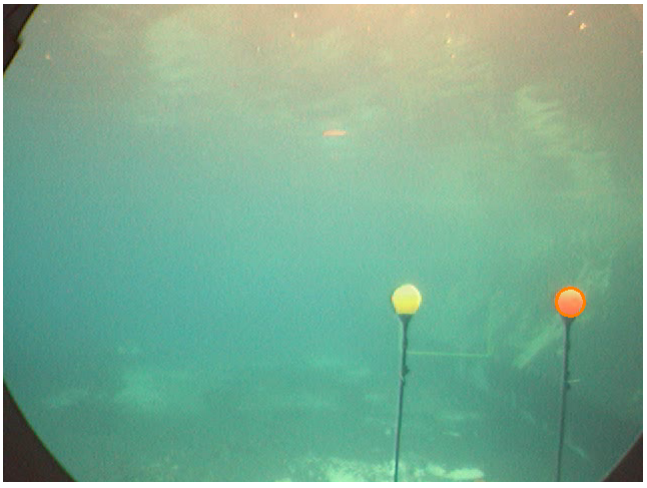


Figure 6: Detection of orange buoy

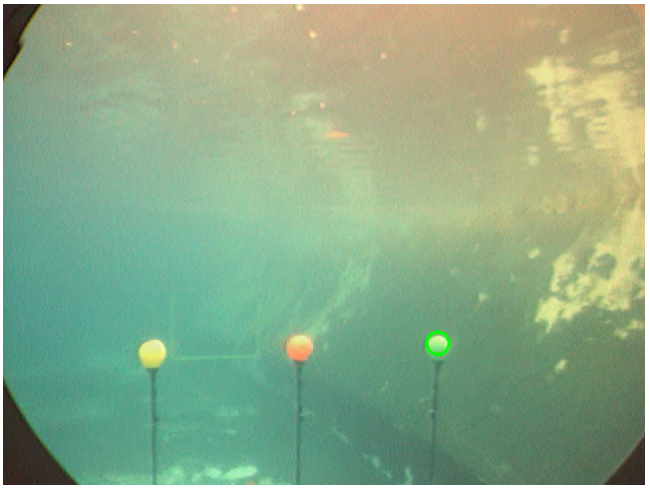


Figure 7: Detection of green buoy

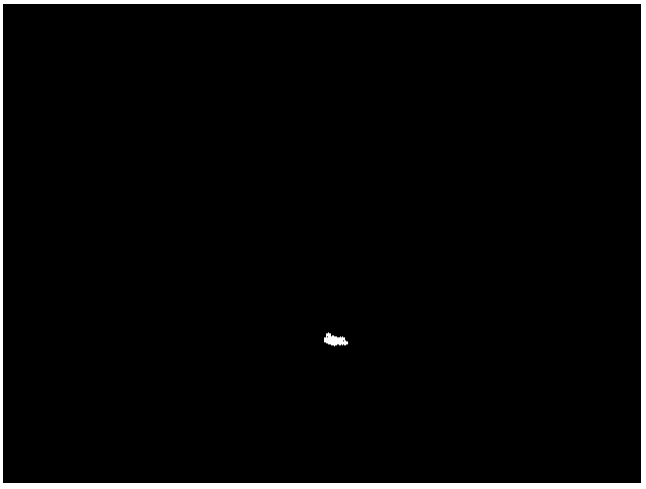


Figure 8: Green buoy after thresholding and dilation

Figure 9: Orange buoy after thresholding and dilation

Problems faced: The process of deciding the thresholds for different buoys was difficult. Due to the lighting conditions, the algorithm was detecting elements other than the buoys. This was solved partially by imposing bounds on the area of the contours.

5 Learning Color Models

5.1 Analysis of Average Histogram

From the previously explained Average histogram procedure, we analyze histograms for each channel, i.e, for blue, green and red channel for each colored buoy. We observe the peaks obtained in each channel of the buoy and estimate a cluster(K) value for each buoy. The following figure is for yellow buoy.

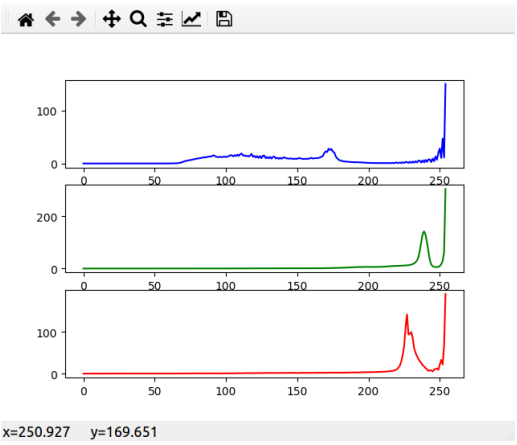


Figure 10: Average Histogram for Yellow Buoy, K = 7

The blue channel has gaussian peaks of upto 4 (changing curves in the graph), green channel has 1 and red channel has 2. So we initialise K with 7 to train the model on the yellow buoys train dataset.

Similarly, for Orange buoy, we estimate a K = 4

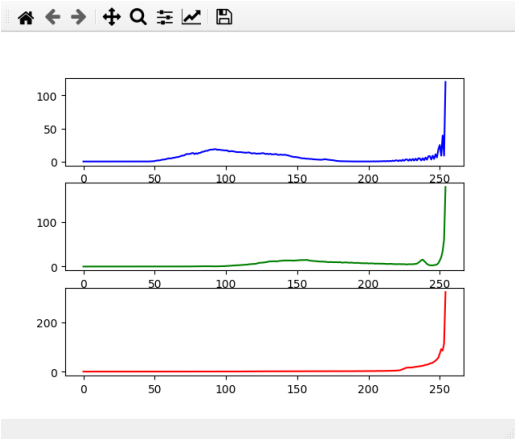


Figure 11: Average Histogram for Orange Buoy, K = 4

And for Green buoy, K = 5

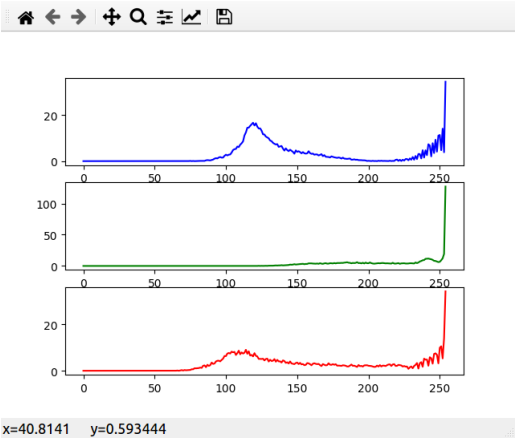


Figure 12: Average Histogram for Green Buoy, K = 5

Notes:

Though, all the colored buoys should have values strictly in their respective channels like- for yellow buoy: only red and green, for orange buoy: only red, for green buoy: only green. But, the awesome dataset for underwater buoy have all channels fused to represent shades of the color in real world scenario.

The N, D values are obtained from the train dataset, i.e N, D are the shape of the xtrain array stored from collection of all the train dataset for each colored buoy. For example, the green dataset has about 29 images which when stored into an array,

we obtain the dimensions of 20074, 3 which make the N, D respectively. The EM algorithm uses these values for N and D.

EM Algorithm Approach:

For a 3D gaussian (since D=3), the EM algorithm takes into account the three channels for each pixel value of all the images stored in the xtrain. Steps are-

- Initialize means, covariances, weights
- Expectation step
- Maximization step
- Iterate until converged on the bounds or max iterations

Initialization-
We initialise the three model parameters- means, covariances and weights as follows-
Mean: Compute mean of randomly selected points in the xtrain to output an np array of size N, K to take into account the clusters K.
Covariance Matrix: We generate an identity I^{DxD} for K times. Then we randomize this matrix to get a varied covariance matrix that also takes into account cluster K.
Weights: The weights (π_K) are initialised with equal probabilities based on K, i.e for K = 4, we initialise π_K as 0.25. The constraint on these probabilities make sure that they sum up to 1.
We also initialise a $P_{Cluster|x}$ and an empty array for Log Likelihood which is to be computed for each gaussian.

Iterate on Expectation and Maximization-
We iterate over the Expectation and Maximization to get the maximum likelihood of a point that belongs to a cluster and thus segment them accordingly. The loop exits on two conditions, either the iterations reach 1500 (some large value) or when the difference between the previous value and the current value in log likelihood array is less than a maximum bound of 0.0001. (works most of the time)

Expectation: Expectation step here refers to computing the expectation of the latent variable (γ_K) based on the model parameters μ, σ, π_K. This latent variable denotes the responsibilities for each point, i.e the gaussian function.

$$\gamma_j(x) = \frac{\pi_k \mathcal{N}(x|\mu_k, \sigma_k)}{\sum_{j=1}^{k} \pi_j \mathcal{N}(x|\mu_j, \sigma_j)}$$

Maximization: This step is used to maximize the likelihood of the point to belong to a cluster based on the current responsibilities. We compute a new mean,

covariance matrix and weights, which are iterated over in loop

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

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

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

Evaluate Log Likelihood: Using the below equation, we compute the log likelihood and store it in an array to check for bounds ≤ 0.0001.

$$ln\ p(X|\mu, \sigma, \pi) = \sum_{n=1}^{N}\ ln \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \sigma_k)$$

Finally, we obtain a set of new mean, covariance, weights (μ, σ, π_K) which are the computed model parameters from the train data set. They denote the shade of a color (μ, σ, π_K) that needs to be checked on the test dataset to segment it from the background/other colors.

Model mean values computed for yellow =

$$\begin{bmatrix} 91.60793607 & 229.50384663 & 234.93116247 \\ 141.59637832 & 239.8989564 & 228.77922135 \\ 254.62096519 & 254.6277707 & 254.81850969 \\ 156.4345089 & 238.31995078 & 227.60192473 \\ 244.10161343 & 254.38369265 & 253.67580227 \\ 246.26777204 & 254.67885427 & 247.06887836 \\ 122.57090459 & 195.18739885 & 174.96399183 \end{bmatrix}$$

Model mean values computed for orange =

$$\begin{bmatrix} 252.25151201 & 254.30689618 & 254.19916193 \\ 123.07389523 & 161.74081702 & 196.10454026 \\ 195.85168932 & 244.4096307 & 242.02641534 \\ 99.06984715 & 166.68919841 & 247.4310493 \end{bmatrix}$$

Model mean values computed for Green =

$$\begin{bmatrix} 247.65648988 & 254.53694057 & 247.48295337 \\ 151.17880838 & 244.11637761 & 190.22482776 \\ 121.6729636 & 210.29082351 & 123.21181298 \\ 117.57839442 & 167.21475627 & 108.13253016 \\ 154.17586555 & 219.96334106 & 171.06102072 \end{bmatrix}$$

Model covariance values computed for Yellow =

$$\begin{bmatrix} [1.42985313e+02 & 7.88673818e+01 & -1.83422381e+01] \\ [7.88673818e+01 & 1.13915854e+02 & -1.94146032e+01] \\ [-1.83422381e+01 & -1.94146032e+01 & 1.16430216e+01] \end{bmatrix}$$

$$\begin{bmatrix} [7.79003526e+02 & -3.65756786e+01 & -3.66187692e+01] \\ [-3.65756786e+01 & 4.46645010e+00 & 2.65565636e+00] \\ [-3.66187692e+01 & 2.65565636e+00 & 6.15431919e+00] \end{bmatrix}$$

$$\begin{bmatrix} [5.90366161e-01 & -8.26297050e-02 & -1.56098352e-02] \\ [-8.26297050e-02 & 3.45555700e-01 & -3.14549327e-02] \\ [-1.56098352e-02 & -3.14549327e-02 & 1.48579147e-01] \end{bmatrix}$$

$$\begin{bmatrix} [9.56295668e+02 & 1.32932911e+02 & 6.28086969e+01] \\ [1.32932911e+02 & 9.15172628e+01 & 7.28061349e+01] \\ [6.28086969e+01 & 7.28061349e+01 & 9.82463369e+01] \end{bmatrix}$$

$$\begin{bmatrix} [1.67190617e+02 & -4.52765482e+00 & -2.98545911e+00] \\ [-4.52765482e+00 & 1.50338013e+00 & 2.72385275e-01] \\ [-2.98545911e+00 & 2.72385275e-01 & 2.07830223e+00] \end{bmatrix}$$

$$\begin{bmatrix} [5.20341445e+01 & -1.45944244e+00 & 8.10961534e+00] \\ [-1.45944244e+00 & 6.91657181e-01 & -1.27182933e+00 \\ [8.10961534e+00 & -1.27182933e+00 & 2.29408014e+01] \end{bmatrix}$$

$$\begin{bmatrix} [8.46174612e+02 & 3.53439409e+02 & 2.37399798e+01] \\ [3.53439409e+02 & 9.17742658e+02 & 1.13292960e+03] \\ [2.37399798e+01 & 1.13292960e+03 & 1.86703418e+03] \end{bmatrix}$$

Model covariance values computed for Orange =

$$\begin{bmatrix} [1.38845885e+01 & 2.81031309e-01 & -2.58525226e-01] \\ [2.81031309e-01 & 1.01148870e+00 & -1.07285015e-01] \\ [-2.58525226e-01 & -1.07285015e-01 & 1.62437413e+00] \\ \\ [6.63161978e+02 & 7.93479669e+02 & 3.10938321e+02] \\ [7.93479669e+02 & 1.14057078e+03 & 7.30277260e+02] \\ [3.10938321e+02 & 7.30277260e+02 & 1.78326324e+03] \\ \\ [1.81353630e+03 & 3.01826094e+02 & 4.53113376e+02] \\ [3.01826094e+02 & 6.45300344e+01 & 7.63711524e+01] \\ [4.53113376e+02 & 7.63711524e+01 & 1.48585028e+02] \\ \\ [4.92479508e+02 & 6.56076445e+02 & -1.07934588e+02] \\ [6.56076445e+02 & 1.06376252e+03 & -1.72701027e+02] \\ [-1.07934588e+02 & -1.72701027e+02 & 4.36175977e+01] \end{bmatrix}$$

Model covariance values computed for Green =

$$\begin{bmatrix} [5.26525323e+01 & -2.62518395e+00 & 4.48788052e+01] \\ [-2.62518395e+00 & 1.05103729e+00 & -2.90190332e+00] \\ [4.48788052e+01 & -2.90190332e+00 & 6.05126758e+01] \\ \\ [1.95035354e+02 & 2.79699033e+01 & 2.73630558e+02] \\ [2.79699033e+01 & 2.37970532e+01 & 2.72345769e+01] \\ [2.73630558e+02 & 2.72345769e+01 & 4.69751301e+02] \\ \\ [6.05420291e+01 & 1.06907518e+02 & 1.06630827e+02] \\ [1.06907518e+02 & 3.31943039e+02 & 2.97472010e+02] \\ [1.06630827e+02 & 2.97472010e+02 & 3.00820366e+02] \\ \\ [1.34088760e+02 & 1.01265363e+02 & 1.52924640e+02] \\ [1.01265363e+02 & 2.58428573e+02 & 1.29594377e+02] \\ [1.52924640e+02 & 1.29594377e+02 & 2.39106346e+02] \\ \\ [7.43704170e+02 & 5.35797585e+02 & 9.28849370e+02] \\ [5.35797585e+02 & 5.73212676e+02 & 7.21392628e+02] \\ [9.28849370e+02 & 7.21392628e+02 & 1.23235434e+03] \end{bmatrix}$$

Model weights values computed for Yellow =

$$\begin{bmatrix} 0.13491365 & 0.35675725 & 0.07805715 & 0.12988553 & 0.09516954 & 0.0424748 & 0.16274208 \end{bmatrix}$$

Model weights values computed for Orange =

$$\begin{bmatrix} 0.15876557 & 0.19870643 & 0.11210311 & 0.53042488 \end{bmatrix}$$

Model weights values computed for Green =
$\begin{bmatrix} 0.22040821 & 0.17246997 & 0.25582982 & 0.23276434 & 0.11852766 \end{bmatrix}$

Note: We have used the mvn.pdf (multivariatenornmal) function from the Scipy library to compute the gaussian as it was faster and had the added advantage of allowing singular matrices. This issue rises when the randomly generated covariance matrix is not singular, due to which the gaussian function cannot be computed. The pdf function takes input as points from data set, mean and covariance matrix.

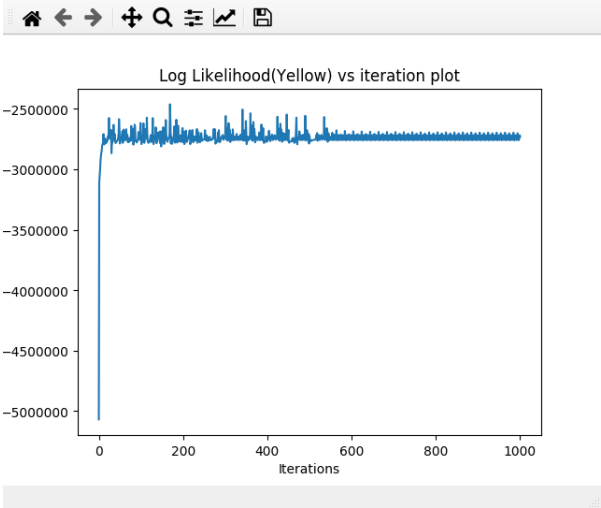


Figure 13: Log likelihood for Yellow Buoy, iterations 1000

Issues faced: Apart from singular matrix issue, there is another bug where the random generator generates huge values either -inf or inf due to which the mvn.pdf function does not work(another advantage). The only workaround is to run the program again.

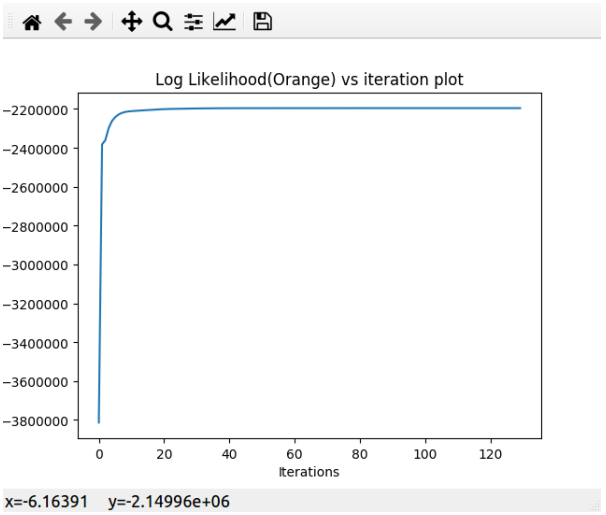


Figure 14: Log likelihood for Orange Buoy, iterations 140

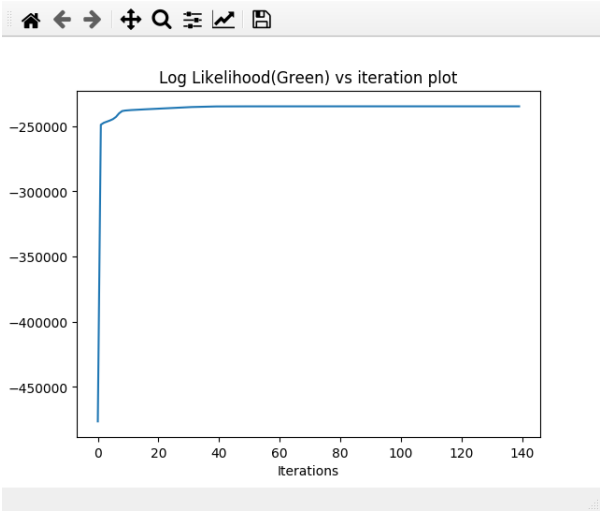


Figure 15: Log likelihood for Green Buoy, iterations 140

6 Buoy Detection

The model is trained first separately on each of the buoys and tested on the video. Then, it is combined with each color, to detect all three buoys at once. Now, from the train data set, we have computed the mean, covariance matrix and weights (μ, σ, π_K) for each colored buoy separately. Now, we validate these parameters on the same video.

Pipeline

- Read the model parameters.
- Resize the frames from video to get the data as an array of N(h*w) x D
- Initialise the probability and likelihood to np zeros array of same dimensions for a given color.
- Compute probability and likelihood using the gaussian function.
- create a mask and dilate the likelihood.
- Detect edges with Canny and find the corresponding contours
- sort these contours (maximum score first) that can be used in Convex hull (generalise the shape of contours to a convex)
- Make a minimum enclosing circle with the hull thus formed, to approximate the center and radius of the circle.
- Draw circle and detect the boi.

Approach

From the pipeline mentioned above, we read the parameters μ, σ, π_K from a binary file which was saved previously post learning with EM. Similar to data preparation in EM, we reshape the image to get the dimension as a 3D array of all points, i.e new dimensions - ((height*width), dimension). Post initialisation, we compute the probabilities and likelihood

Now, we create a mask based on the likelihood thus found. This likelihood represents the maximum score that a particular color is in the image. This mask will take in all the channels of that color. Consider yellow boi, which will take the three channels and give us a mask of the following which upon dilation gives the blob on the right.

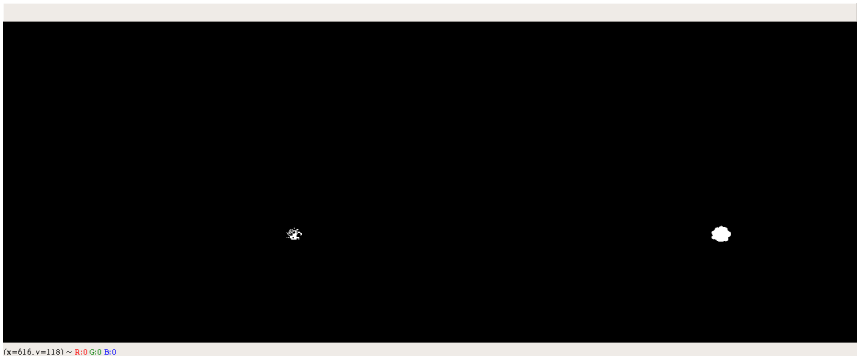


Figure 16: Yellow blob - Mask (left) and its dilated mask (right)

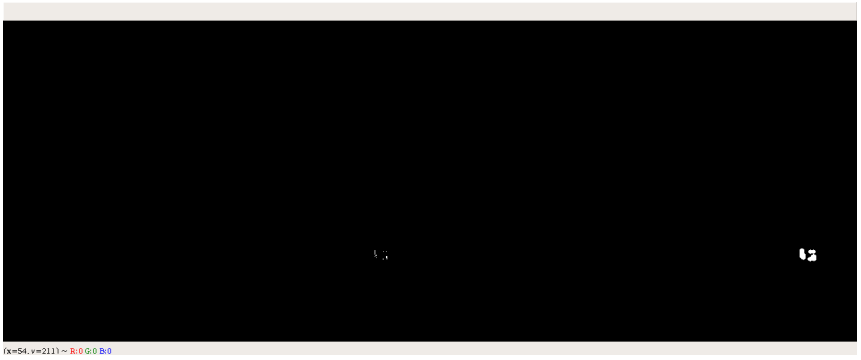


Figure 17: Green blob - Mask (left) and its dilated mask (right)

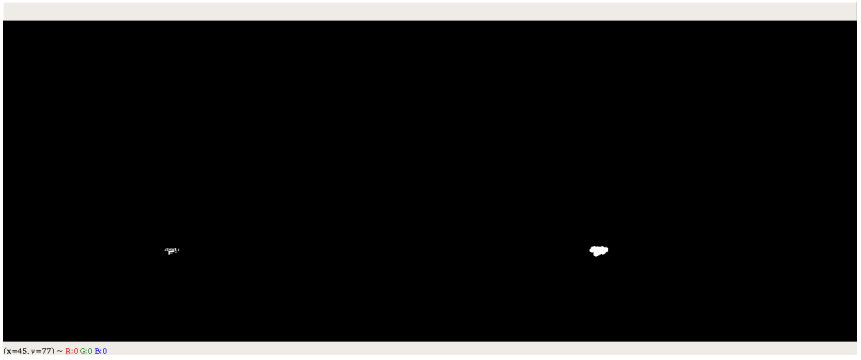


Figure 18: Orange blob - Mask (left) and its dilated mask (right)

The dilated mask is generated with an ellipse kernel of size (7,7) which is of the type

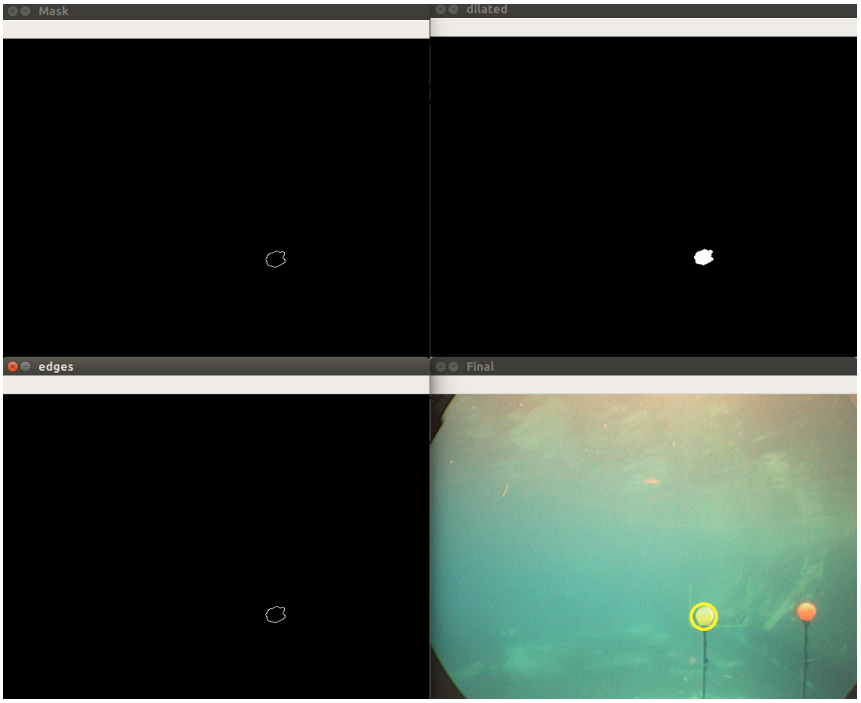


Figure 19: Yellow blob detection Frame - 40

of ellipse.

$$\begin{bmatrix} 0,0,1,1,1,0,0 \\ 0,1,1,1,1,1,0 \\ 1,1,1,1,1,1,1 \\ 1,1,1,1,1,1,1 \\ 1,1,1,1,1,1,1 \\ 0,1,1,1,1,1,0 \\ 0,0,1,1,1,0,0 \end{bmatrix}$$

Now, once we have the mask and its dilated version, we send this to a Canny edge detector to find the edges on the blob. The bottom left section of the image above represents the edges formed by Canny.

Post this, we find contours on this image and sort these contours. The sorting operation results in giving us the contour with maximum likelihood. Once, we have the best score contour we now generate a convex hull encloses the thus formed blobs into a convex shape.

Finally, the minEnclosingCircle will result in coordinates for center and the radius of the convex shape(circle) thus formed from before. This center and radius is used to draw a circle on the image and thus buoys are detected.

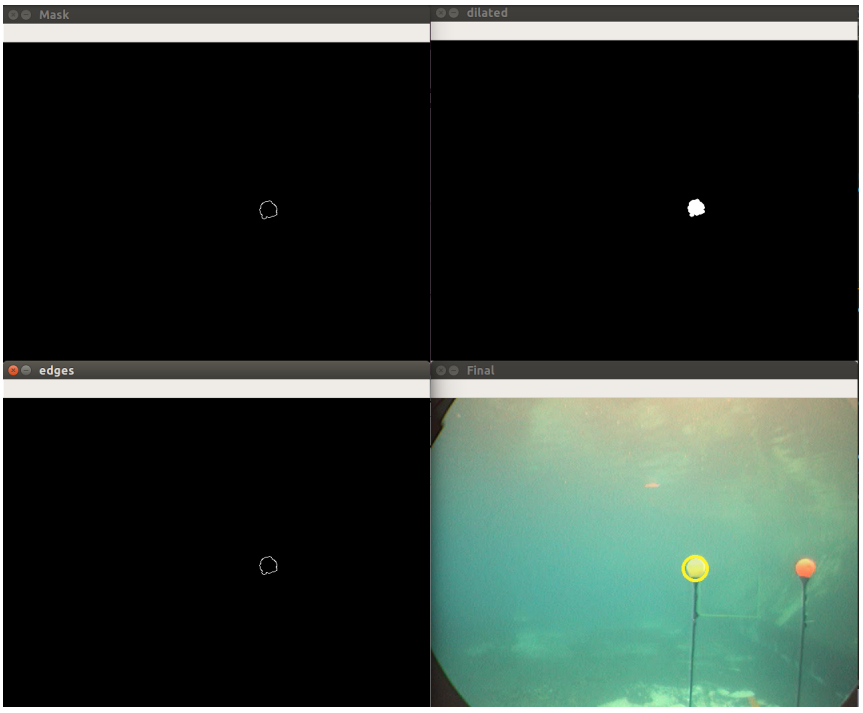


Figure 20: Yellow blob detection Frame - 74

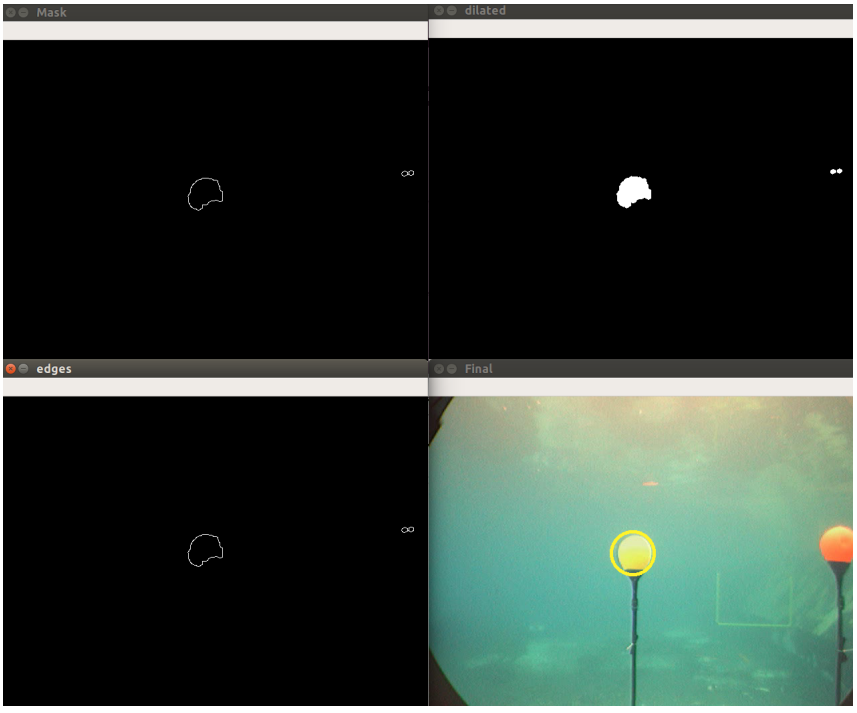


Figure 21: Yellow blob detection Frame - 100

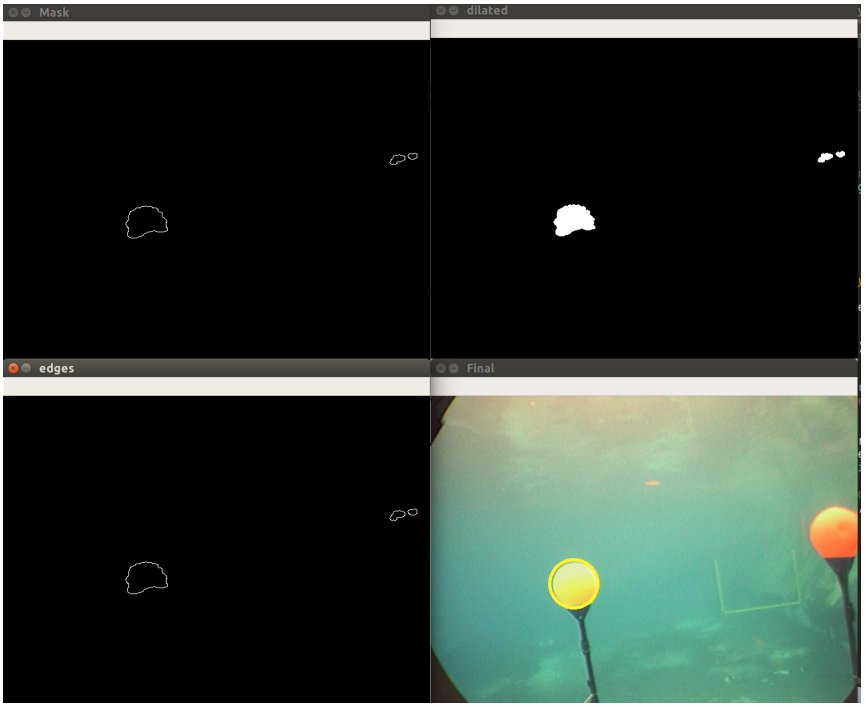


Figure 22: Yellow blob detection Frame - 141

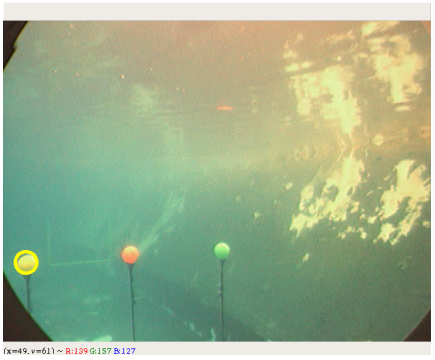


Figure 23: Yellow Blob detection

Figure 24: Yellow and Orange Blob Detection

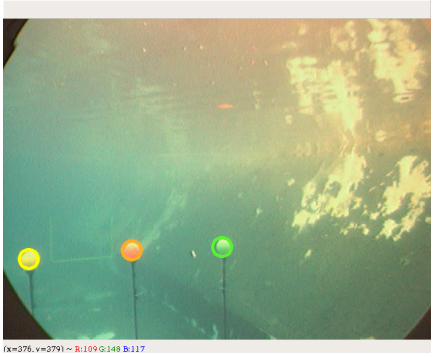


Figure 25: All Buoys detection

Bibliography

[1] Color Classification Article - CMSC426, https://cmsc426.github.io/colorseg/#colorclassification

[2] Gaussian Mixture Model explanation Article, https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

[3] GMM Article, https://towardsdatascience.com/gaussian-mixture-modelling-gmm-833c88587c7f

Click this link for video outputs: Video Link