Project 3 (ENPM673)

Author Name
Aalap Rana (116421393)
Koyal Bhartia (116350990)
Harsh Bharat Kakashaniya (116311236)

Color Segmentation using Gaussian Mixture Models and Expectation Maximization techniques

Date : 27/2/2019

Contents

List of Figures

1 Introduction

Buoys are underwater floating object used as a marker or as a mooring.The data set given for this project is a video of three distinctly colored underwater buoys. The goal of the project is to segment and detect thee buoys in the video/data set. The segmentation of these buoys can be done based on the color,shape and other geometrical properties; however, for the scope of this project we have segmented the buoys based on the color. The conventional segmentation approach cannot be applied here as the environment is having a large amount of noise and the light intensities are also changing continuously,and this results in ineffective thresholding. In such cases we can implement the concept of color segmentation using Gaussian Mixture Models and Expectation Maximization Techniques. In the above method we do unsupervised learning and teach the model different color distribution and use the learned model to do color segmentation. The process of color segmentation can be divided into the following parts:

1. Data generation
2. Development of Gaussian Mixture Model and Maximum Likelihood Algorithm
3. Learning the color model
4. Detecting the buoys

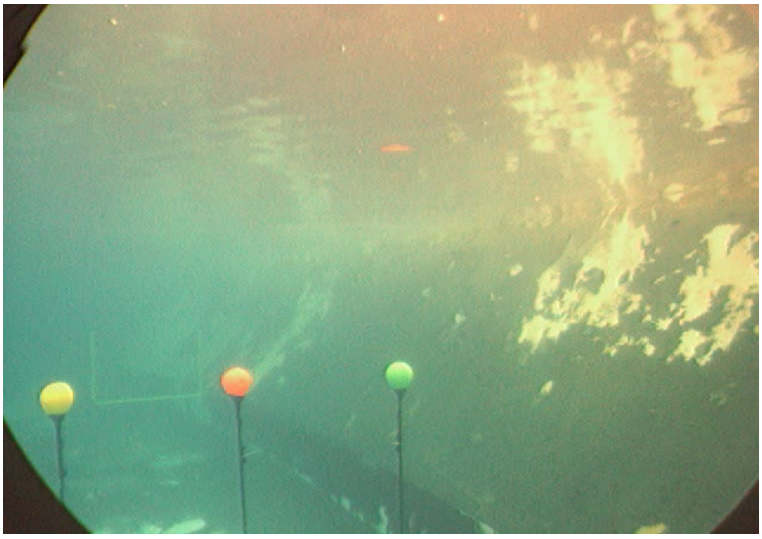

Figure 1: Sample image frame

2 Data Preparation

2.1 Creating Training Data

Folowing are the steps followed to create the training data:

- The given video 'detectbuoy. avi' is first read into individual image frames - total of 200.
- The frame generated in the previous step is adjusted to improve the contrast to be used for training; thus the frame is converted to LAB color space and by applying Contrast Limited Adaptive Histogram Equalization on the L-channel the enhance frame is obtained.

- The cv2 functions used for the same are: **cv2.createCLAHE()**, **cv2.cvtColor(img, cv2.COLORBGR2LAB)** and **clahe.apply(l)**.
- These LAB images converted to GRB space are used to crop out the training data from the given frames.
- The cropped images of each buoy is taken and bitwise anded with the LAB image to get the final cropped images for training.
- The rest of the images which is not used for training is used for testing purpose.

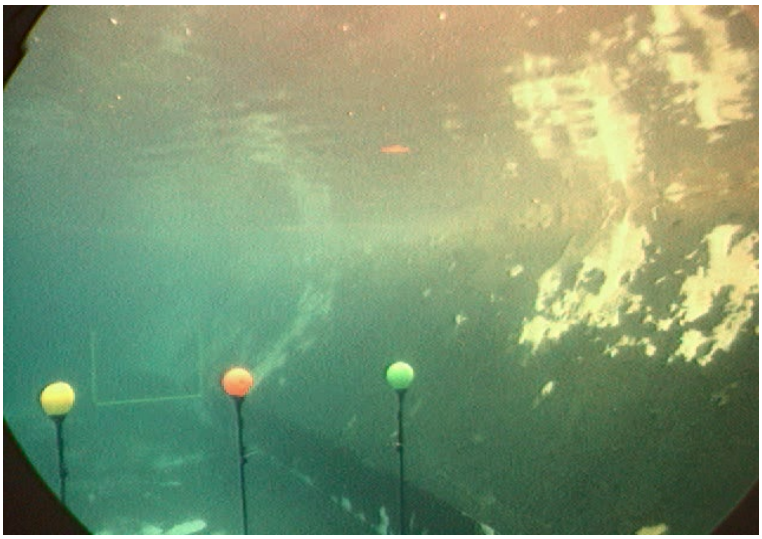

Figure 2: Sample LAB image

2.1.1 Matplotlib event handling

- Matplotlib's event handling function **implot.figure.canvas.mpl**$_c onnectisusedtocreatethe"onClick"event.Theclicksar$
- To crop the buoys, we select two points by mouse click. The first click is at the center of the buoy. The second click is on the outer periphery.
- The coordinates of both the mouse clicks is stored, and the radius is calculated using the distance fromula between the two points.
- Now that we have the center and radius of the buoy, a white circle with these parameters is plotted on a black image. This is donw using the half plane equation of a circle i.e $(x-a)^2 + (y-b)^2 - r^2 = 0$. Where a,b is teh center and r is the radius. All points inside this equation is coloured white.
- This binary image is bitwise anded with the coresponding frame which gives an illusion of successfull cropping. This is stored in the Train folder,
- This is repeated for all the buoys on its respective image frames.

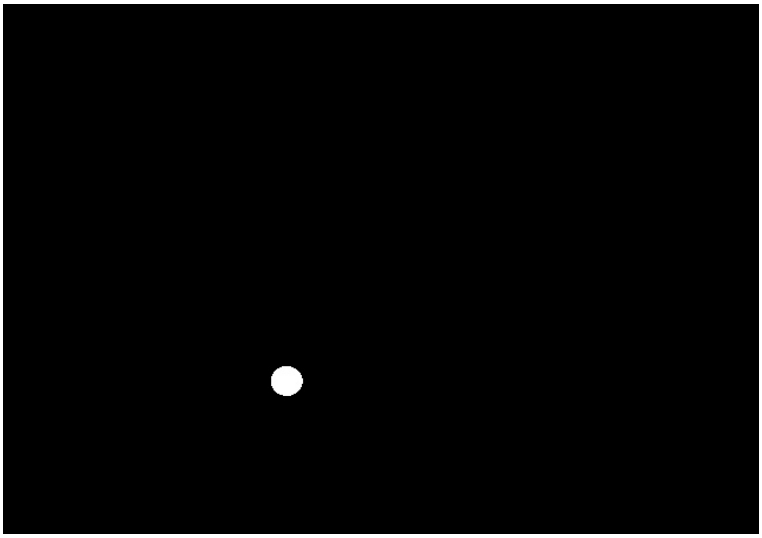

Figure 3: Sample of binary crop

Figure 4: Sample of final crop - Yellow buoy

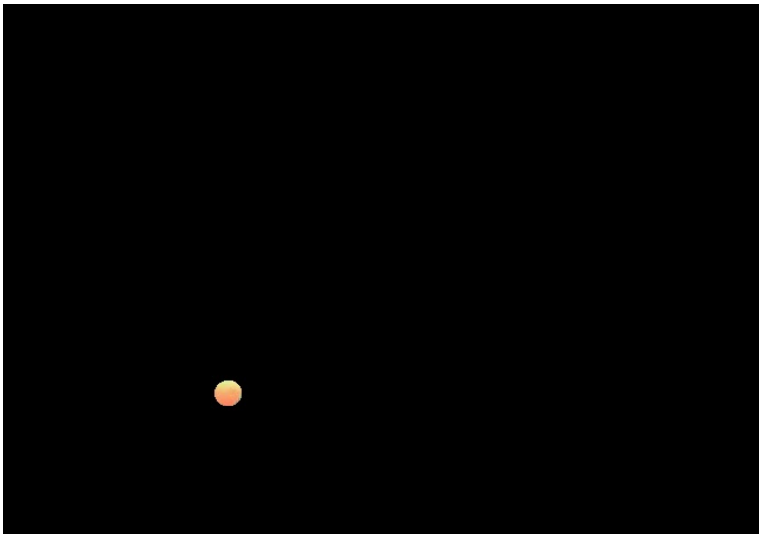

Figure 5: Sample of final crop - Red buoy

Figure 6: Sample of final crop - Green buoy

3 Average Colour Histogram

Following are the Average Histogram graphs of all the channels - Red, Green and Blue of all the three buoys. The histogram is obtained using the **cv2.calcHist()** function.

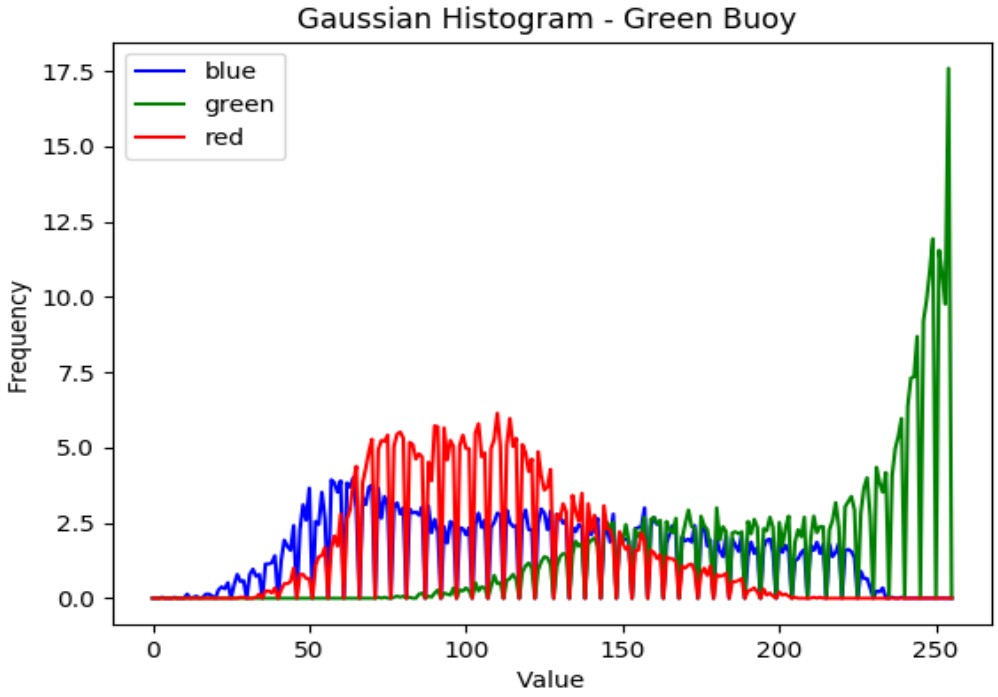

Figure 7: Plot of Average histogram - Green buoy

It can be noticed that the green buoy has a peak of the green channel at around value of more than 200. The red and the blue channels in the green buoy has low values of less than 150. Similar observations can be made for each of the other buoys. When the channel is same as the buoy color, the intensity is high near the value 255. Also notice that for yellow buoy both red and green channels have high intensities at near 255.

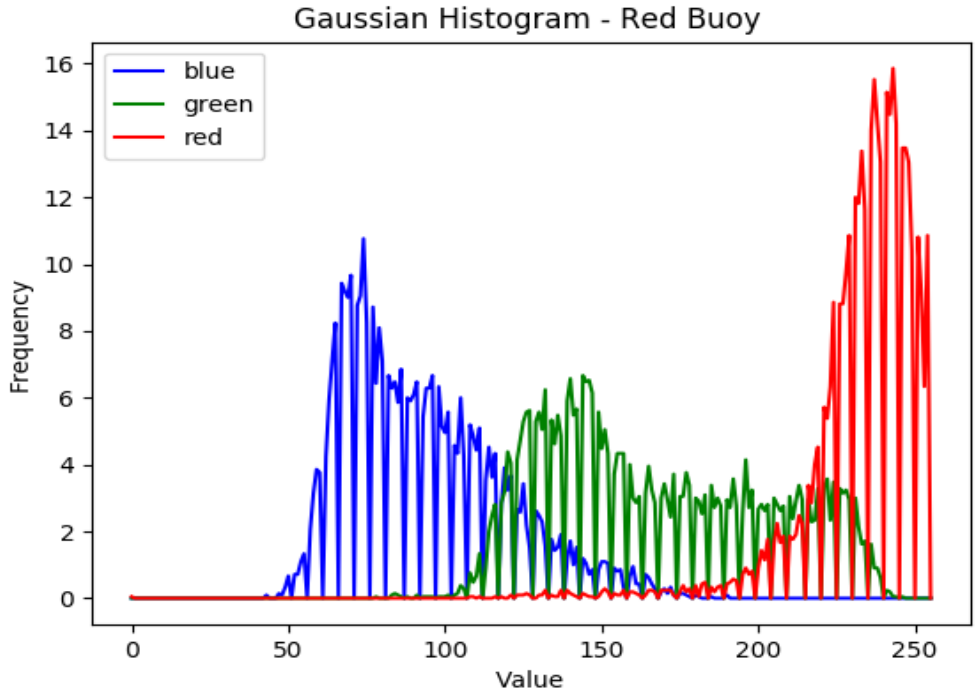

Figure 8: Plot of Average histogram - Red buoy

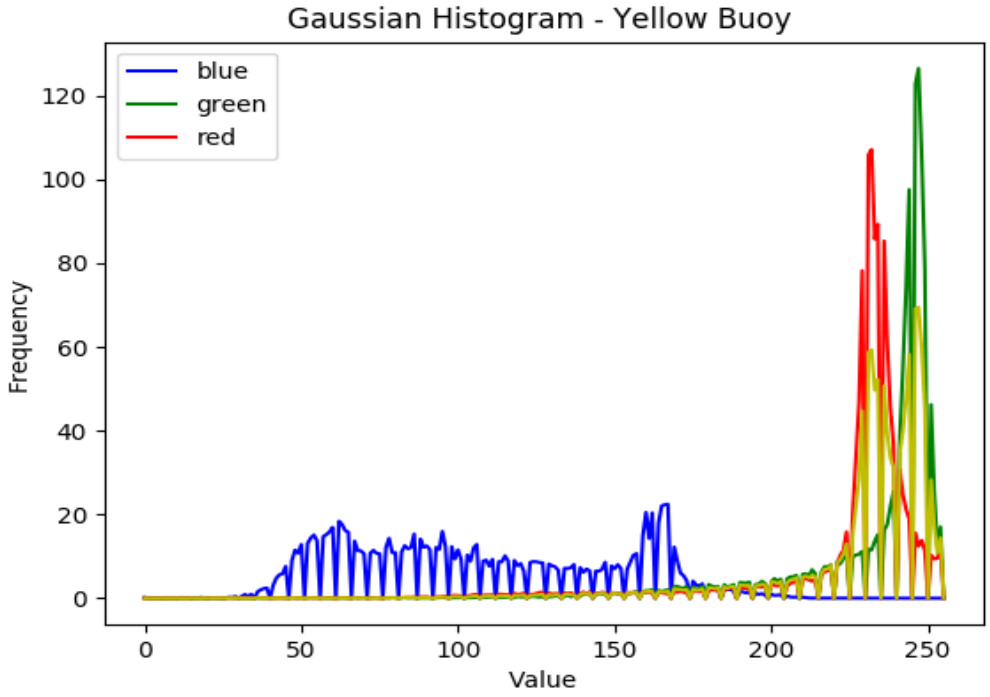

Figure 9: Plot of Average histogram - Yellow buoy

4 Colour Segmentation using 1-D Gaussian

The concept of Gaussian distribution used in this section is the following probability districation function.

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} exp - \frac{(x-\mu)^2}{2\sigma^2} \tag{1}$$

Here x: variable, μ: Mean σ^2: Variance and σ: Standard deviation

The general graph of the above probabily districation is as follows:

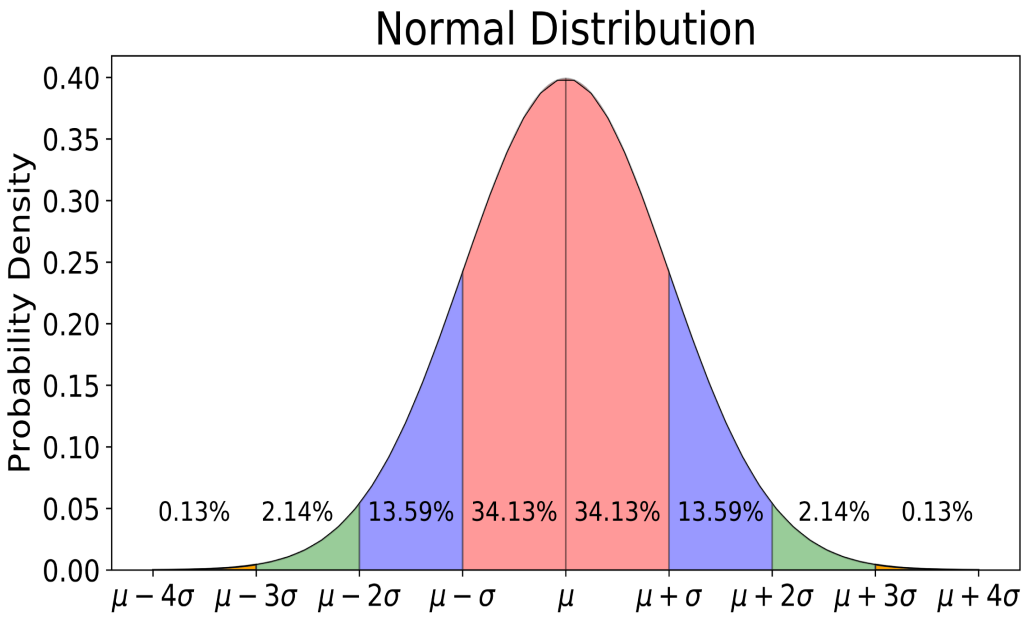

Figure 10: Gaussian Normal Distribution

This is drawn in the code using the function: **stats.norm.pdf(x, mean, sigma)**

4.1 Expected Gaussian obtained using the Histograms above

Approximate Gaussians are plotted using the histograms shown above. Following are the respective gaussian parameter values obtained.

Green buoy

Table 1: Gaussian Parameters using Histogram - Green Buoy

Gaussian	Mean	Variance
1	245.44996712	17.24670195
2	180.43888128	41.50338724

Red buoy

Table 2: Gaussian Parameters using Histogram - Red Buoy

Gaussian	Mean	Variance
1	244.12202519	17.86451239

Yellow buoy

Table 3: Gaussian Parameters using Histogram - Yellow Buoy

Gaussian	Mean	Variance
1	239.71629791	43.15174498
2	194.11582437	40.90080332

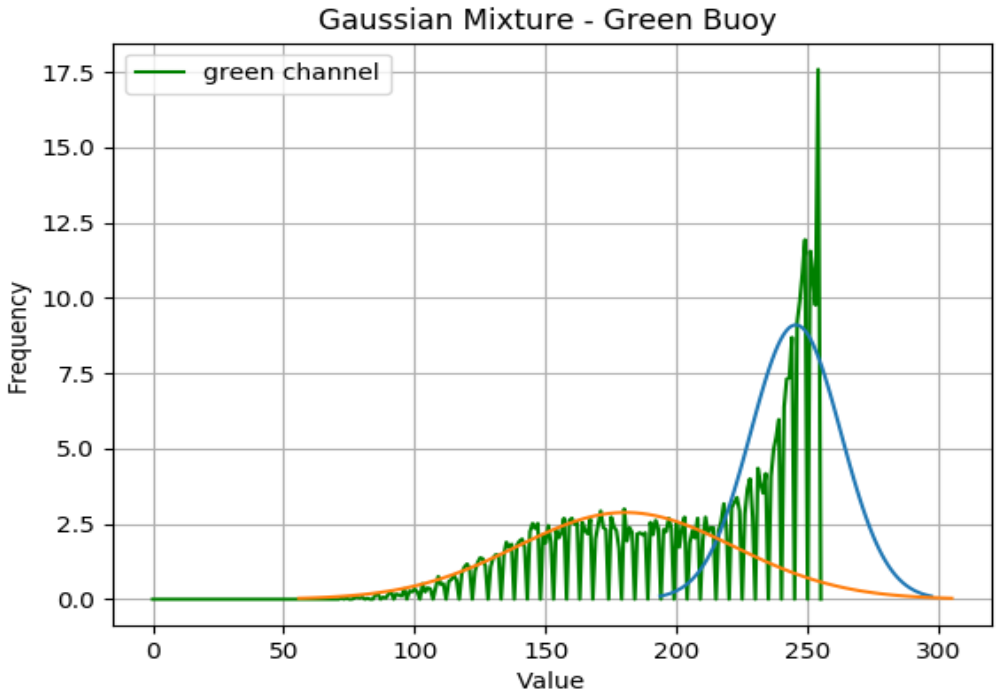

Figure 11: Plot of Gaussian using Histogram - Green buoy

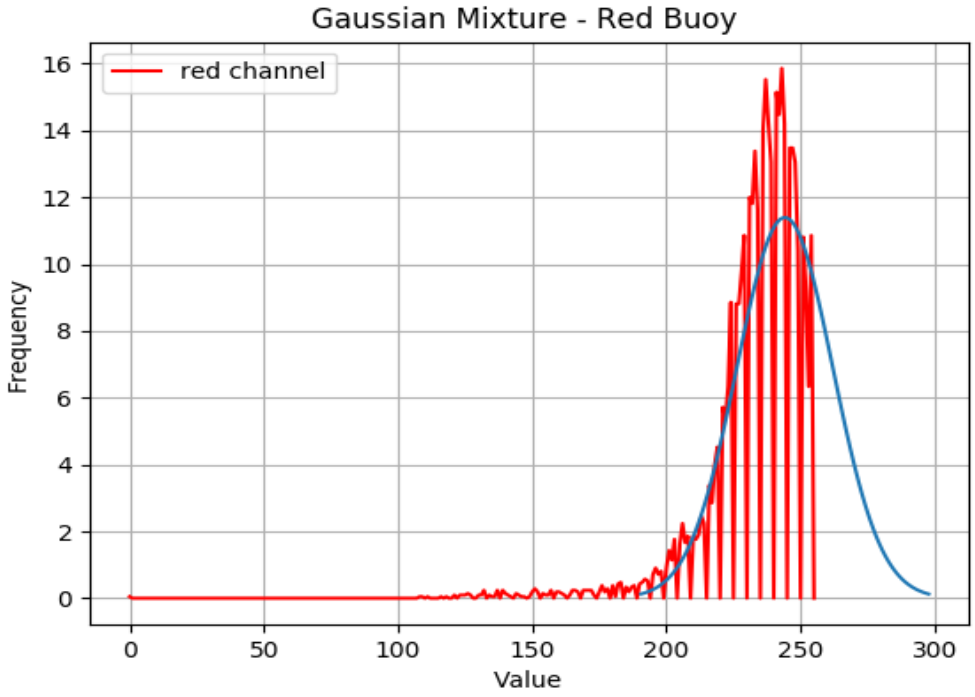

Figure 12: Plot of Gaussian using Histogram - Red buoy

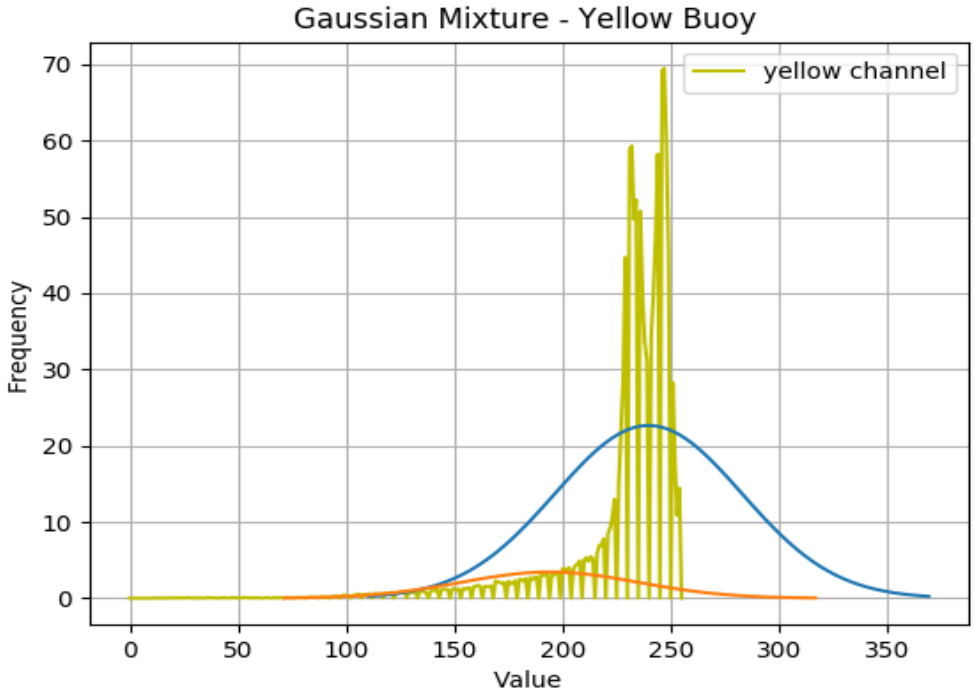

Figure 13: Plot of Gaussian using Histogram - Yellow buoy

4.2 Actual Gaussians obtained using the EM algorithm

The EM algorithm is implemented and following are the values of the weights, means and variances obtained for each of the buoys.

Green Buoy

Table 4: Gaussian Parameters using EM - Green Buoy

Gaussian	**Weight**	**Mean**	**Variance**
1	6.13706209e-10	108.91231614	1.06779996
2	5.91448440e-01	241.15779102	12.56758694
3	4.08551560e-01	179.95432209	22.21725008

Red Buoy

Table 5: Gaussian Parameters using EM - Red Buoy

Gaussian	**Weight**	**Mean**	**Variance**
1	1.28667007e-04	137.0000001	1.99999994e-07
2	8.72802231e-01	241.48904147	8.13059751e+00
3	1.27069102e-01	214.27877843	2.20071630e+01

Yellow Buoy

Table 6: Gaussian Parameters using EM - Yellow Buoy

Gaussian	**Weight**	**Mean**	**Variance**
1	6.13706209e-10	108.91231614	1.06779996
2	5.91448440e-01	241.15779102	12.56758694
3	4.08551560e-01	179.95432209	22.21725008

As seen for each of the buoys the weights of the Gaussians sums to a total of 1. Out of the 3 possible gaussians, in each case the weight of 1 gaussian is veryy negligible and hence can be neglected. Hence in each case 2 gaussians are considered whose whose weights are considered. Following are the gaussian graphs obtained using the means and sigmas received using the EM algorithm.

5 Gaussian Mixture Models and Maximum Likelihood Algorithm

The EM-algorithm is an iterative algorithm which begins from some initial mostly random values of means and variances and then iterative updation of the means,variances and weights is done until convergence is reached. The EM algorithm can be divided into two parts mainly:

1. E-Expectation

$$\gamma_j(x) = \frac{\pi_k \mathcal{N}(x \mid \mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j)}$$

2. M-Maximization

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

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

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

All formulas are as used in the link: `http://www.cse.iitm.ac.in/~vplab/courses/DVP/PDF/gmm.pdf`

E-Expectation step: In the E-step the probability of the data belonging to different gaussian is calculated and the number of gaussian are decided by the analysis of the histogram. This is done by the following formula:

case1) 1D gaussian:

case2) 3D gaussian:

M-Maximization step: The Maximization step is the most curtail step.As in this step the new parameters are calculated until the convergence is reached. The M-step outputs the updated mean,sigma and weights.

Implementation in code:

The process of generation of a 1 dimension Gaussian for fitting the data points is implemented using five functions, they are as follows:

- data_generator()
- Initialize_parameter()
- gaussian()

- E_step()
- M_step()

The first step for fitting a 1 Dimensional Gaussian probability distribution to the given data is formation of a structured data set. The function data_generator() is used to form a well formatted data set from the raw data obtained from the image which is desired by the functions in the subsequent steps. The function "data_generator()" takes the input image and splits it into 3 channels namely B(blue),R(red) and G(green) and stores the intensity values in data. The final data set "data" thus formed is transposed.

The next step for fitting a 1 Dimensional Gaussian probability distribution to the given data requires initialization of the matrices that would be used in the process. This is done using the function "Initialize_parameter()". The function forms the following matrices:

- mean(N*1)
- sigma(N*1)
- weight(N*1)

The function "gaussian()" is formed that calculates the probability of the data point belonging to the Gaussian Probability Distribution . The function takes data points,mean, sigma as the input parameter and return the probability corresponding to that point of belonging to the gaussian under consideration.

The function "E_step()" is a user defined function which estimates the probability of all the data point using the function "gaussian()" to belong to a gaussian. A local matrix called classified is formed to store the corresponding normalized probabilities ,where the dimension of classified is(N*3). A variable called "check" is used to monitor the convergence of the EM- algorithm.

The Maximization-step of the EM-algorithm is done by using the function "M_step()". The function takes the matrix classified_data which contains normalized probability for the data points and data. The iterative process of updating the mean,weights and sigma is done and stored in new_mean, new_sigma and new_weight.

The new weight is obtained by division of cumulative sum of normalized probability for a given data point to belong to a gaussian to the total number of data points for that gaussian. In the same manner all weights are calculated.

The new mean is calculated by dividing the cumulative sum of product of data point and the normalized probability from the classified matrix with the sum of the normalized probability for that gaussian distribution. On the similar manner all means are computed and stored in new_weight.

The new sigma is formed by division of cumulative sum of product of normalized probability and square of difference of data and corresponding mean with the sum of the normalized probability for that gaussian distribution.

6 3D - EM

The process of generation of a 3 dimension Gaussian for fitting the data points is implemented using five functions, they are as follows:

- data_generator()
- Initialize_parameter()
- MV_gaussian()
- E_step()
- M_step()

The first step for fitting a 3 Dimensional Gaussian probability distribution to the given data is formation of a structured data set.The function data_generator() is used to form a well formatted data set from the raw data obtained from the image which is desired by the functions in the subsequent steps. The function "data_generator()" takes the input image and splits it into 3 channels namely B(blue),R(red) and G(green) and stores the intensity values in data1, data2 and data3.The final data set "data" is formed by vertical stacking the data1, data2 and data3,and the formed matrix is transposed.

The next step for fitting a 3 Dimensional Gaussian probability distribution to the given data requires initialization of the matrices that would be used in the process. This is done using the function "Initialize_parameter()" which takes two input namely K and D where K is the number of Gaussian to be fitted and D is the dimension of data point. The function forms the following matrices:

- mean(K*D)
- sigma(D*D*K)
- weight(K*1)

The function "MV_gaussian()" is formed that calculates the probability of the data point belonging to the Gaussian Probability Distribution .The function takes data points,mean, sigma and dimension as the input parameter and return the probability corresponding to that point.

The function "E_step()" is a user defined function which estimates the probability of all the 3 dimensional data point using the function "MV_gaussian()".A local matrix called classified is formed to store the corresponding probabilities,where the dimension of classified is(N*2).A variable called "check" is used to monitor the convergence of the EM- algorithm.

The Maximization-step of the EM-algorithm is done by using the function "M_step()".The function takes the matrix classified_data which contains normalized probability for the data points, data and dimension. The iterative process of updating the mean,weights and sigma(co-variance) is done using local matrices new_mean, new_sigma and new_weight.

The new weight is obtained by division of sum of column of classified_data divided by sum of the cumulative sum of the columns of classified data and stored in new_weight matrix.

In the case of 3 dimensional gaussian probability distribution the sigma is actually the co-variance matrix.The co-variance matrix is formed using the linear algebra formula for each gaussian. The co-variance matrix consist of 3 variance and 6 cross-variance terms.The variance terms are calculated by the cumulative sum of product of normalized probability of the data point with the difference of data and its mean divided by the sum of normalized probability for that gaussian. The cross variance is calculated by the cumulative sum of product of normalized probability of the data point with the the difference between data point value and mean1 (for instance blue channel) and the product of difference between data point value and mean2(for instance green channel).The process for calculation of variance and cross variance is done for all the channels namely red ,green and blue, which results in formation of the required covariance matrix.

The new mean is calculated by division of cumulative sum of product of data point and normalized probability of that data point by the sum of normalized probability.

7 Learning Color Models

Following are the Gaussians obtained using the learning model of EM. The approach and method followed has been explained above.

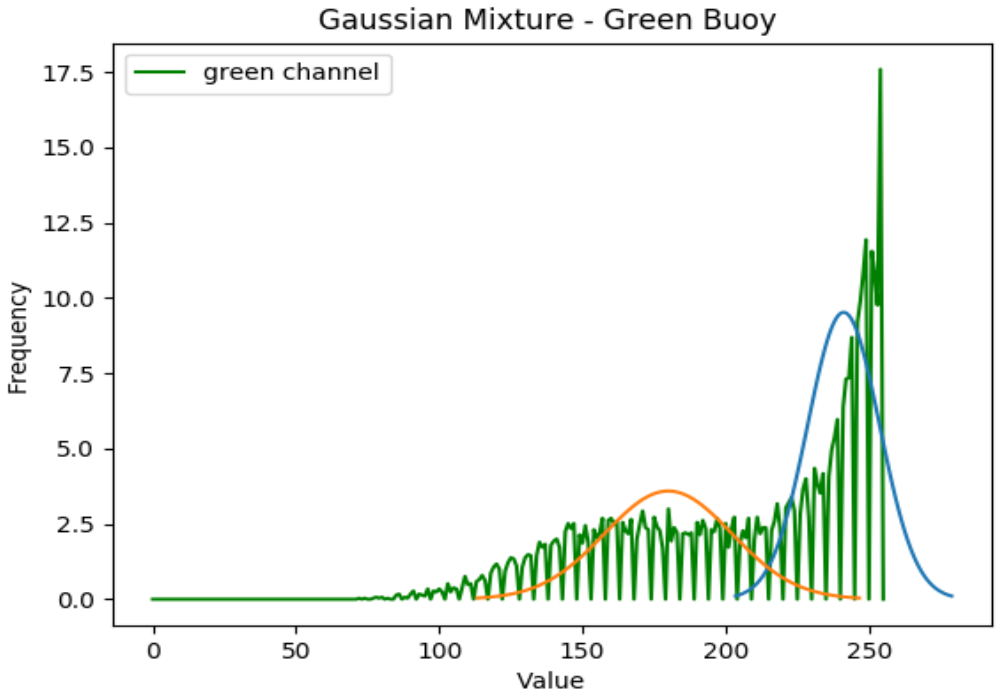

Figure 14: Plot of Gaussian Mixture - Green buoy

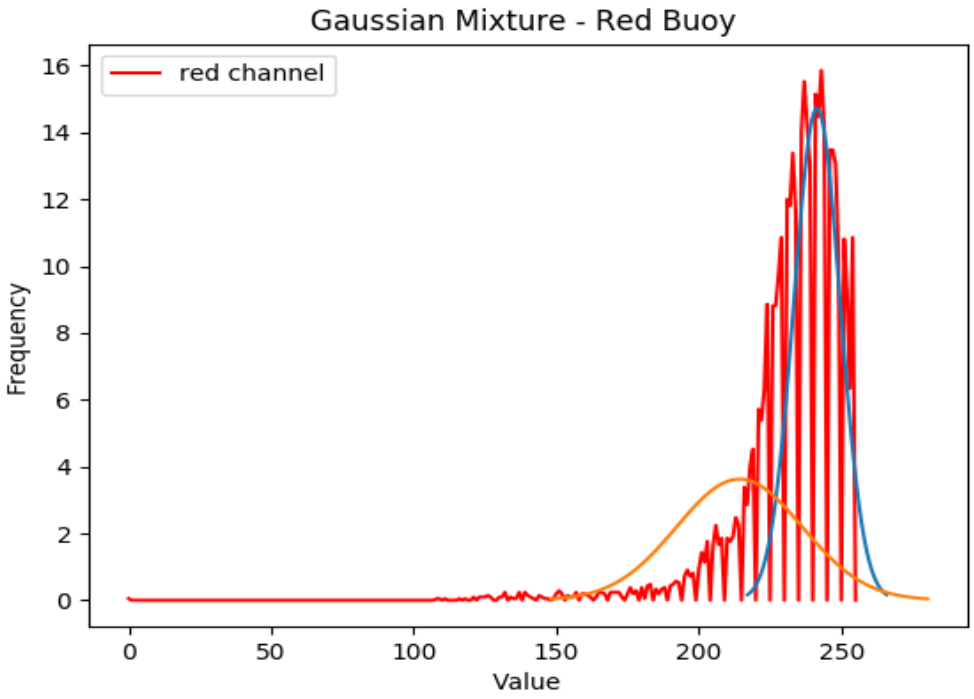

Figure 15: Plot of Gaussian Mixture - Red buoy

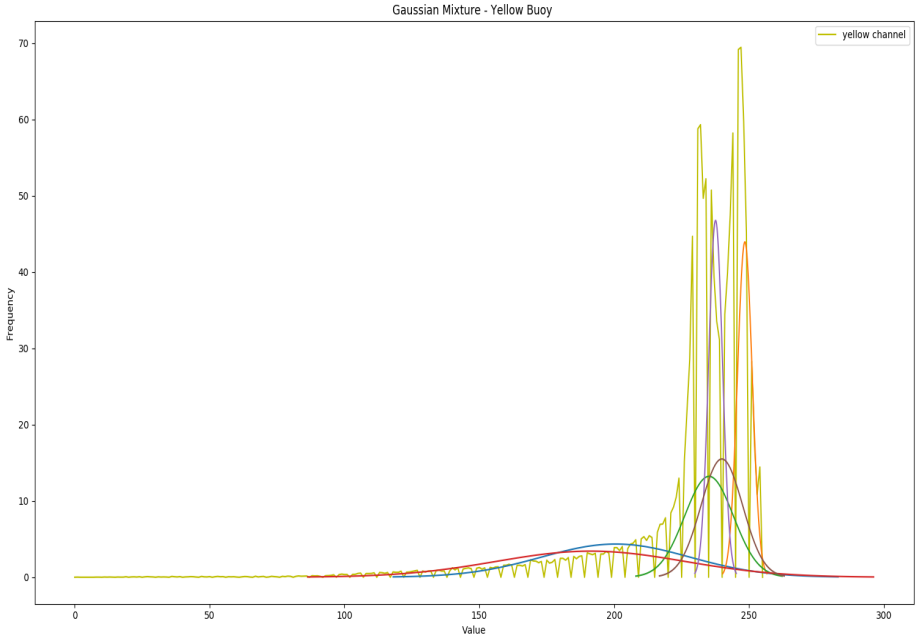

Figure 16: Plot of Gaussian Mixture - Yellow buoy

8 Buoy Detection

Buoy detection has been implemented using Hough Circles. This is used for the detection of of the circles/buoys. This uses the Hough Gradient Method which uses the gradient information of edges.

The function used is:
cv2.HoughCircles(InputArray image, OutputArray circles, int method, double dp, double minDist, double param1,double param2, int minRadius,int maxRadius)

Here the threshold params param1 and param2 are taken as 100 and 10 respectively. The min and max radius are taken as 13 and 30 as the approximate values for the radius of the buoys.

The folder with the 3D Gaussian contains shows the detection of buoys using the EM algorithm implementation of 3D Gaussian.

9 Further Analysis

Yes, if other representations other than RGB is provided, it would be more effective at solving the problem of segmenting buoys underwater. For eg. we have used the LAB colour space in this project as alreay explained above. This helps decode the brightness information of images.

L – Lightness (Intensity). a – color component ranging from Green to Magenta. b – color component ranging from Blue to Yellow. The Lab color space is quite different from the RGB color space. In RGB color space the color information is separated into three channels but the same three channels also encode brightness information. On the other hand, in Lab color space, the L channel is independent of color information and encodes brightness only. The other two channels encode color. Hence in environents like water, where there are various fluctuations in intensity, a colour space like this helps.

Link with output videos:
`https://drive.google.com/drive/u/0/folders/1vEHi9kKYj8Pyllbr7qTmfpCv9TX3xPRd`