

UNIVERSITY OF MARYLAND

PERCEPTION FOR AUTONOMOUS ROBOTS

PROJECT - 3

Buoy Detection using Gaussian Mixture Models

Pruthvi Sanghavi UID: 116951005

Naman Gupta UID: 116949330

Amoghavarsha Prasanna UID: 116952910

6 April 2020



Contents

1	Data Preparation	3
2	Average Histogram for each colored buoys	4
3	Segmentation using 1D Gaussian and Color thresholding	5
3.1	Gaussian Distributions	5
4	Gaussian Mixture Models and Maximum Likelihood Algorithm	9
4.1	Learning Color Models	9
4.2	Buoy Detection	10
5	References	12

List of Figures

1	A single frame from the given video	3
2	cropped images from training data	3
3	Histogram for Red buoy	4
4	Histogram for yellow buoy	4
5	Histogram for green buoy	5
6	General representation of the normal distribution	5
7	Generated Gaussian: Red Channel	6
8	Generated Gaussian: Yellow Channel	6
9	Generated Gaussian: Green Channel	7
10	Segmented red buoy	7
11	Segmented yellow buoy	8
12	Segmented green buoy	8

1 Data Preparation

Following steps are performed on the given video data feed:

Step 1 - Converting video to images

To convert the given video data feed into images, a count parameter was defined and was iterated for each frame. Each frame is saved using the `cv2.imwrite()` function.

Step 2 - Cropping the images and preparing the data set

For each image in the prepared image data set, we constructed a region of interest rectangle around the buoy using the `event handling` function. The event handling (mouse-Click) allows a number of click around the region of interest, once the sufficient points are clicked, the image gets cropped and the next image automatically turn up and the process is repeated. There are three different colored buoys in the images. So the three buoys were cropped and three different data samples for each color (Red, Yellow and Green) was prepared. The cropped image obtained was masked with black color to prevent noise

The obtained data samples for each colored image was divided randomly into training and testing data set keeping the 70 - 30 ratio in mind.

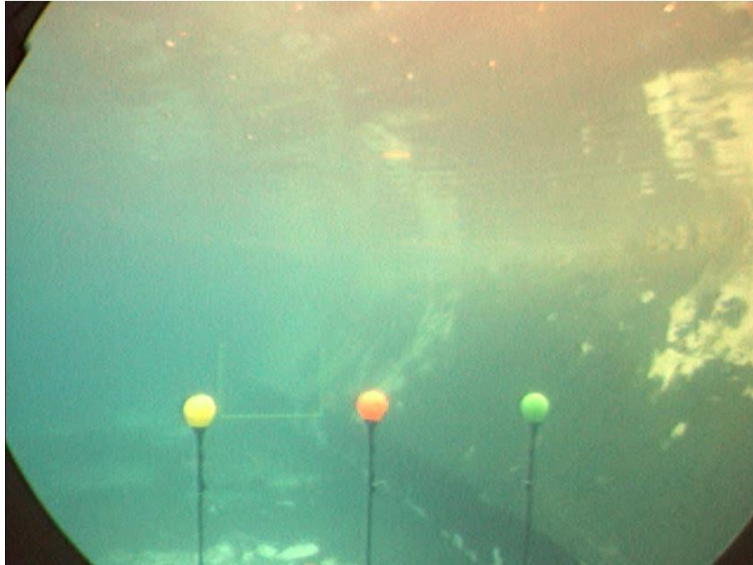


Figure 1: A single frame from the given video



Figure 2: cropped images from training data

2 Average Histogram for each colored buoys

Next, we averaged the color histogram for each channel from the sampled images using the openCV inbuilt `cv.calcHist()` function. Shown below are the histograms for the red, yellow and the green cropped images respectively. Histograms are plotted using `matplotlib` library.

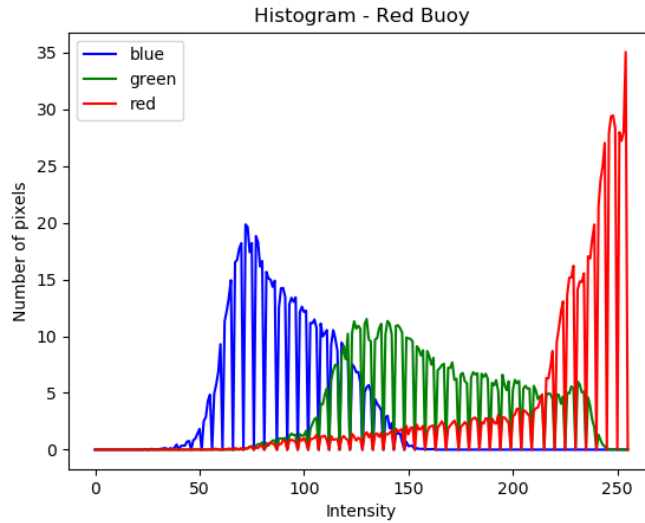


Figure 3: Histogram for Red buoy

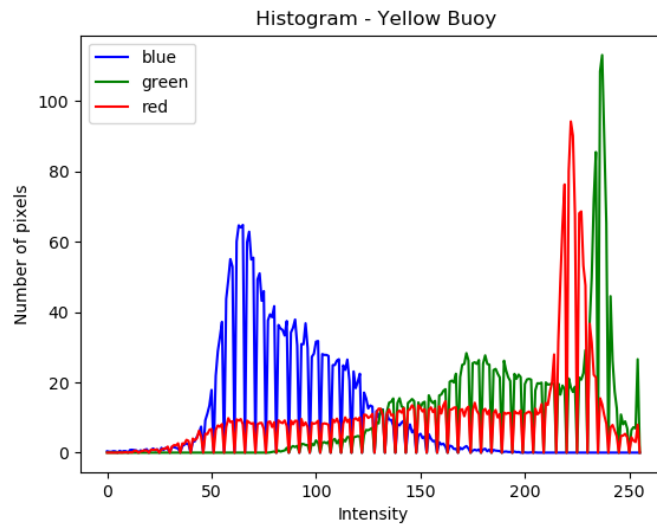


Figure 4: Histogram for yellow buoy

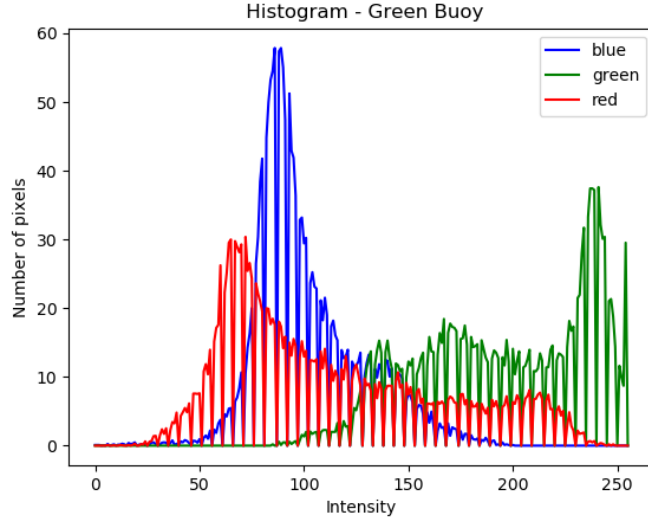


Figure 5: Histogram for green buoy

3 Segmentation using 1D Gaussian and Color thresholding

3.1 Gaussian Distributions

The following probability distribution function is used to determine the 1D Gaussian for each of the channels.

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

Here, μ is the mean, σ is the standard deviation, σ^2 is the variance

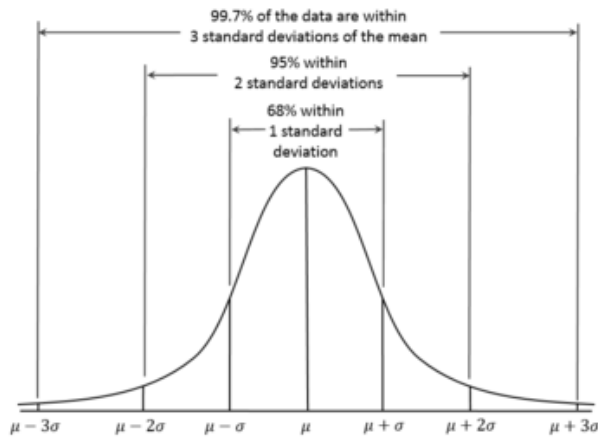


Figure 6: General representation of the normal distribution

Now, 1D Gaussian for each of the color channels are determined from the color histograms for the buoys, using the mean, variance and the standard deviation from the images in the data set. The results respective Gaussian can be seen below.

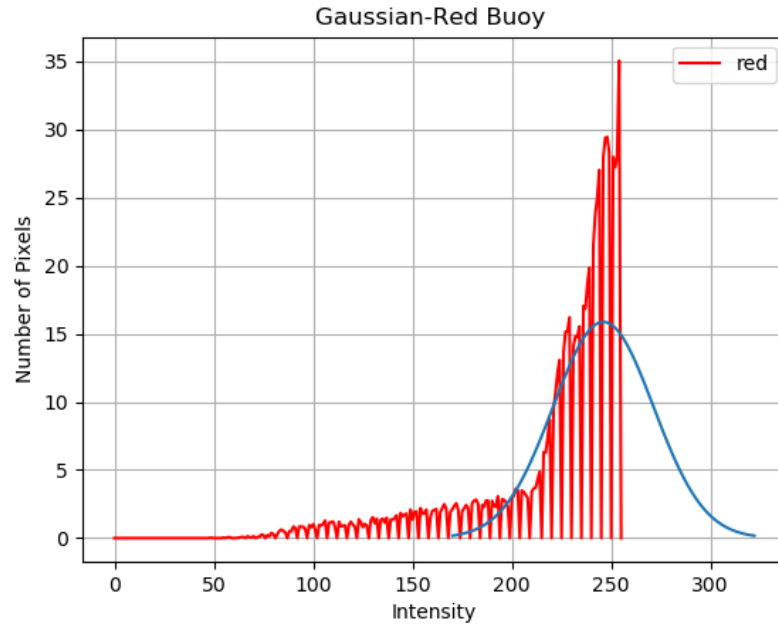


Figure 7: Generated Gaussian: Red Channel

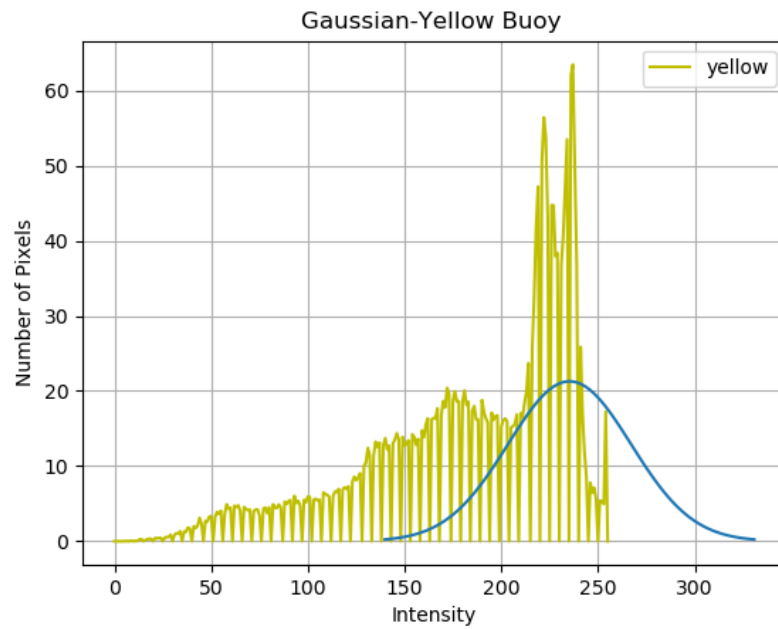


Figure 8: Generated Gaussian: Yellow Channel

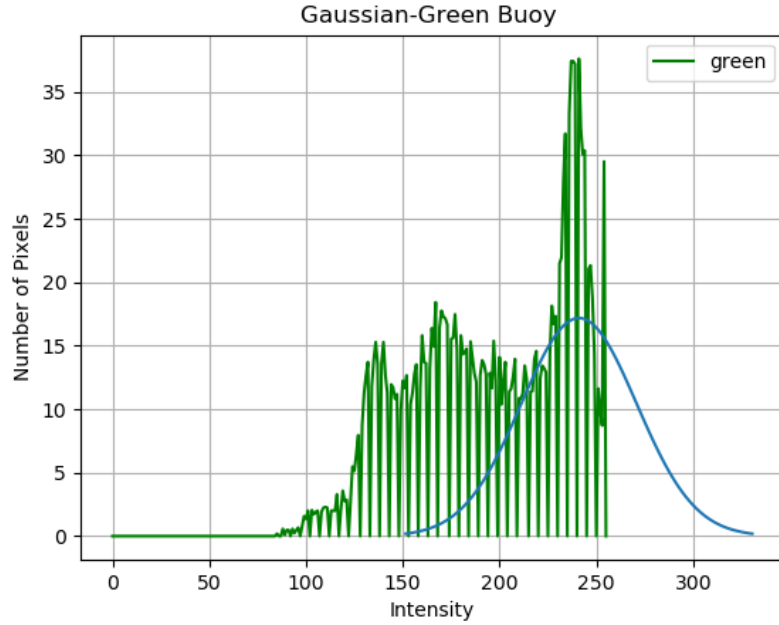


Figure 9: Generated Gaussian: Green Channel

The next step was to segment the colored buoys from the 1D Gaussian and through thresholding. Since the image is colored it contains the RGB components. Therefore, by setting threshold values for the HSV color model, by converting RGB to HSV for easier implementation.

Using the `cv2.bitwise AND` function, masking is achieved and preserving the selected ROI or cropped region from the training data set.

The result of segmentation is shown in the figures below, refer code for full implementation.



Figure 10: Segmented red buoy



Figure 11: Segmented yellow buoy

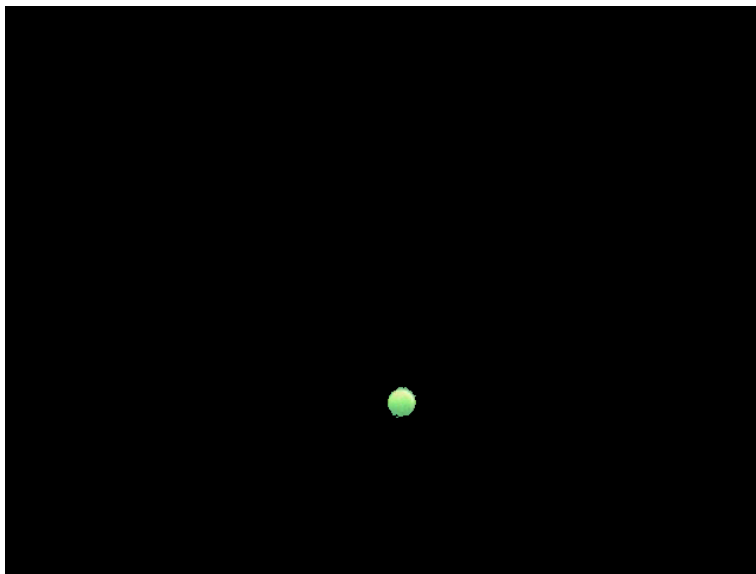


Figure 12: Segmented green buoy

4 Gaussian Mixture Models and Maximum Likelihood Algorithm

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters.

4.1 Learning Color Models


After the data is prepared in the sections of 70 percent training data and 30 percent testing data, the training data is used to learn the color models. The learning model determines the color parameters that are mean, co-variance and weight.

- Feature vectors are extracted that is every image's pixel value. By thresholding the unwanted pixels which has value less than 50, the features are extracted.
- Based on the histograms, we assume the initial parameters of mean ' μ ', co-variance matrix ' Σ ' and weight ' π '.
- Now we calculate posterior probability, also called as responsibilities, for every pixel using Bayes rule which is

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

Here, N is multi-variate Gaussian Distribution of every pixel value which is calculated using

$$\mathcal{N}(\mathbf{x} | \mu, \Sigma) = \frac{1}{(2\pi|\Sigma|)^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \mu)^T \Sigma^{-1}(\mathbf{x} - \mu)\right\}$$



Here,

$$\Sigma = \begin{bmatrix} \sigma_R^2 & \sigma_R \sigma_G & \sigma_R \sigma_B \\ \sigma_R \sigma_G & \sigma_G^2 & \sigma_G \sigma_B \\ \sigma_R \sigma_B & \sigma_G \sigma_B & \sigma_B^2 \end{bmatrix}$$

The diagonal elements denote the variance of individual R, G, B channels and other terms denote correlation of RGB terms of one over other. The posterior probability is expectation step which computes the expected value for given parameter.

- We need to maximize the logarithm of the probability function which is weighted sum of Gaussians. This is maximum likelihood estimation method using every parameter. The log-likelihood is calculated as

$$\ln p(\mathbf{X} | \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^N \ln p(\mathbf{x}_n) = \sum_{n=1}^N \ln \left\{ \sum_{k=1}^K \pi_k N(\mathbf{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

- For Maximization step, the parameters mean, co-variance and weights are updated based on the latent variable calculated using estimation step.

$$\mu_j = \frac{\sum_{n=1}^N \gamma_j(\mathbf{x}_n) \mathbf{x}_n}{\sum_{n=1}^N \gamma_j(\mathbf{x}_n)} \quad \Sigma_j = \frac{\sum_{n=1}^N \gamma_j(\mathbf{x}_n) (\mathbf{x}_n - \mu_j)(\mathbf{x}_n - \mu_j)^T}{\sum_{n=1}^N \gamma_j(\mathbf{x}_n)} \quad \pi_j = \frac{1}{N} \sum_{n=1}^N \gamma_j(\mathbf{x}_n)$$

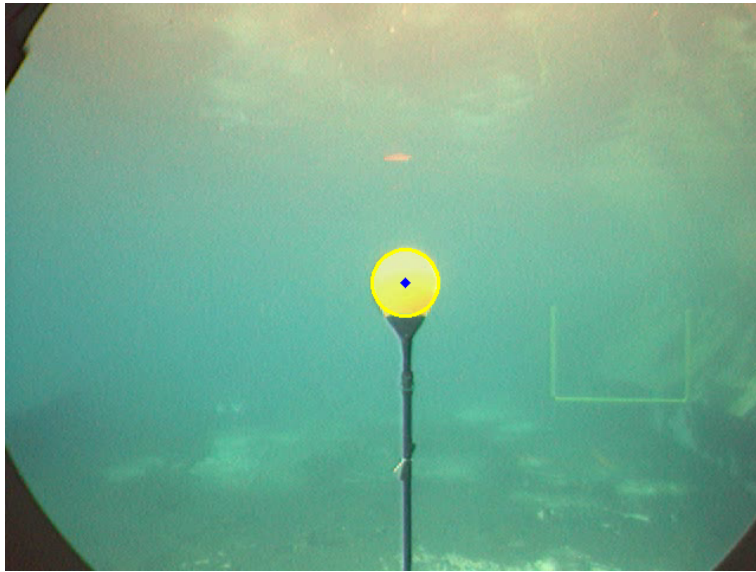
The parameters are now updated and the new parameters are used for the expectation step. This loop will keep on the running until there is no convergence. As convergence arises, this loop will break and the last updated parameters are printed that is the trained parameters to detect our buoys. The entire method is used for all three buoys to get their own trained parameters.

4.2 Buoy Detection

Now, we have the updated parameters saved in the text file. Taking those parameters and substituting into probability distribution function, we get the pixel values which are used to saturate the required color from the actual image.

$$f(x|\theta) = w f_1(x | \mu_1, \sigma_1^2) + (1 - w) f_2(x | \mu_2, \sigma_2^2)$$

Here, w is weight, μ is mean and σ is co-variance. Based on this function, we get pixel values that is used to saturate the original image. From the saturated image using hough circle function, the circle coordinates are detected and drawn on the original image.



Link to Results:

<https://drive.google.com/drive/folders/17eO_HZwCzxNqrd-DXKExQSckOnobPJWr?usp=sharing>

5 References

Event Planning :https://matplotlib.org/users/event_handling.html

Expectation Maximization:

<http://www.cse.iitm.ac.in/~vplab/courses/DVP/PDF/gmm.pdf>

http://www.rmki.kfki.hu/~banmi/elte/bishop_em.pdf