ENPM 673 - Perception for Autonomous Robots

Color Segmentation using Gaussian Mixture Models and Expectation Maximization techniques

Submitted towards completion of Project-3

Arjun Gupta
Vishnuu AD
Abhishek Banerjee

6th April 2020

1 Introduction

The purpose of this project is to perform an image segmentation task for detecting colored buoys in a noisy video stream. We execute this task by leveraging Gaussian Mixture Models to encode the information in the image, followed by Expectation Maximization to get the best estimate of the location of the colored blobs.

In the first part we use techniques to crop out the differently colored blobs in the original image, to create our dataset. This dataset is used to train the Gaussian Mixture Model by leveraging the Expectation Maximization algorithm. In the second part, we use the "trained" Gaussian Mixture Models to determine the cluster of each relevant color, and thereby get the desired segmented result

2 Data Generation

As stated before, it is difficult to segment images on the basis of color thresholds in noisy environments and the video stream in our current scenario is an example of such a dataset. Therefore we leverage the robustness of Gaussian Mixture Models, hereon referred to as GMMs. The first step in the pipeline is to crop the images of the colored blobs, in order to train our GMM.

For this purpose, we use $cv2.EVENT\_MOUSEBUTTON$ module and it's associated functions to generate the requisite images. For each image, we segment out one specific color of image by making two successive clicks with the left mouse button on a single image. The first click is at the center of a buoy, and the second click is at the periphery of the buoy. We calculate the euclidean distance between the two click points (denoted as D) and crop out an image which has a size of $(2D, 2D)$, centered around the first click. To ensure that noisy data isn't used for training, we mask the cropped image by retaining the color for the buoy and rendering everything else white (intensity of $(255, 255, 255)$ in RGB). A few examples of the cropped images are given herein.

Figure 1: Orange Buoy

Figure 2: Yellow Buoy

Figure 3: Green Buoy

3 Gaussian Mixture Models and Expectation Maximization

The project is concerned with the task of identifying buoys of specific colors from a video sequence, and therefore falls under the class of problems concerned with clustering of points. For GMM's, the classes are grouped together on the basis of how close their statistical properties (in this case, their mean and variance) are. The method is base on the premise that any distribution can be modelled as a set of Gaussians and that any sample is generated from a single Gaussian. The Gaussian distribution is given by:

$$P\left(x_i|z_i=j,c\right)=\frac{1}{\sqrt{\left(2\pi\right)\left|\Sigma_j\right|}}\exp\left(-\frac{1}{2}\left(x-\mu_j\right)^T\Sigma_j^{-1}\left(x-\mu_j\right)\right)$$

Here z is a latent k-dimensional vector, such that the probability of getting a specific gaussian, given a class c is:

$$P\left(z_i=j,c\right)=\phi_j$$

The goal of the project is to estimate the parameters $\phi_j, \mu_j and \Sigma_j$ for each of the colored buoys and this is done using the maximum log-likelihood criteria, which is given by:

$$\begin{aligned}\ell(\phi,\mu,\Sigma)&=\sum_{i=1}\log P\left(x_i;\phi,\mu,\Sigma\right)\\&=\sum_{i=1}^{m}\log\sum_{j=1}^{k}P\left(x_i|z=j;\mu_j,\Sigma_j\right)P\left(z=j;\mu_j,\Sigma_j\right)\end{aligned}$$

For multiple gaussians, the analytical solution of estimating the log-likelihood does not have a closed-form solution. Under such circumstances ,we rely on the Expectation Maximization (EM) formulation. It is an iterative algorithm for using maximum likelihood to estimate the parameters of a statistical model with unobserved variables. It has two main steps. First is the E-step, which stands for expectation. We compute some probability distribution of the model so we can use it for expectations. Second comes the M-step, which stands for maximization. In this step, we maximize the lower bound of the log-likelihood function by generating a new set of parameters with respect to the expectations.

Therefore, the steps for EM algorithm for GMMs can be summarized as under:

1. Initialize the mean μ_j , covariance matrix $\sum_j$ and ϕ_j , and evaluate the initial value of the log likelihood.

2. Calculate the weights of each gaussian approximation (E-Step) by:

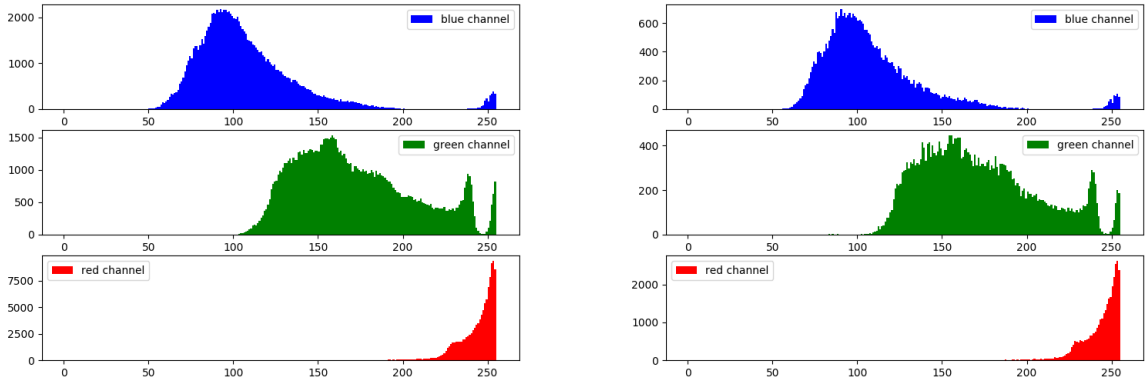
$$
\begin{aligned}
w(j,i) :&= P\left(z_i = j | x_i; \phi, \mu, \Sigma\right) \\
&= \frac{\phi(j) P\left(x_i | z_i = j; \mu_j, \Sigma_j\right)}{\sum_{l=1}^{k} \phi(l) P\left(x_i | z_i = l; \mu_l, \Sigma_l\right)}
\end{aligned}
$$

3. Maximize the log-likelihood to obtain:

$$
\begin{aligned}
\phi(j) &:= \frac{1}{m} \sum_{i=1}^{m} w(j,i) \\
\mu_j &:= \frac{\sum_{i=1}^{m} w(j,i) x_i}{\sum_{i=1}^{m} w(j,i)} \\
\Sigma_j &:= \frac{\sum_{i=1}^{m} w(j,i) \left(x^{(i)} - \mu_j\right) \left(x_i - \mu_j\right)^T}{\sum_{i=1}^{m} w(j,i)}
\end{aligned}
$$

4 Histogram Generation

For each colored buoy, we compute and visualize the average histogram across all three channels, i.e., R,G,B. For this we first flatten an entire image(belonging to one specific color) into one vector, and for each channel we calculate the histogram. The is repeated for all the images specific to that buoy, and these are concatenated for a complete set. Thereafter, we obtain the number of pixels for each intensity across the three channels. The histograms are given herein:

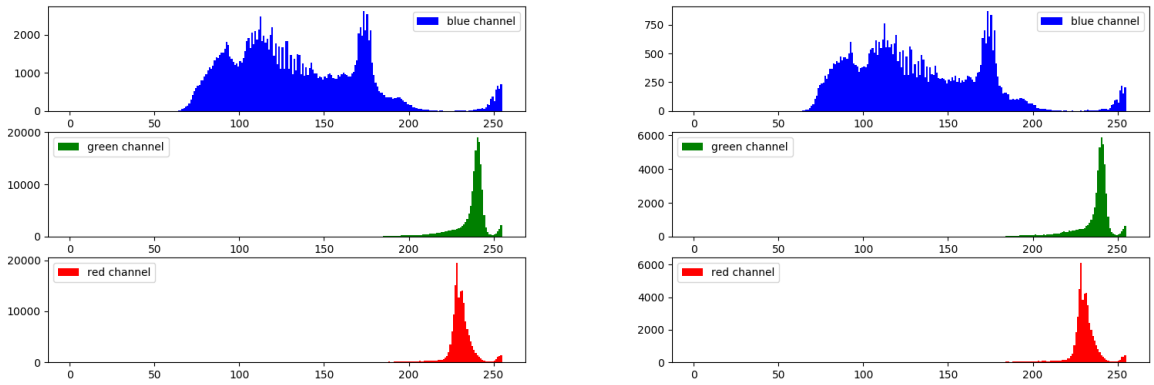


(a) Training Histogram

(b) Testing Histogram

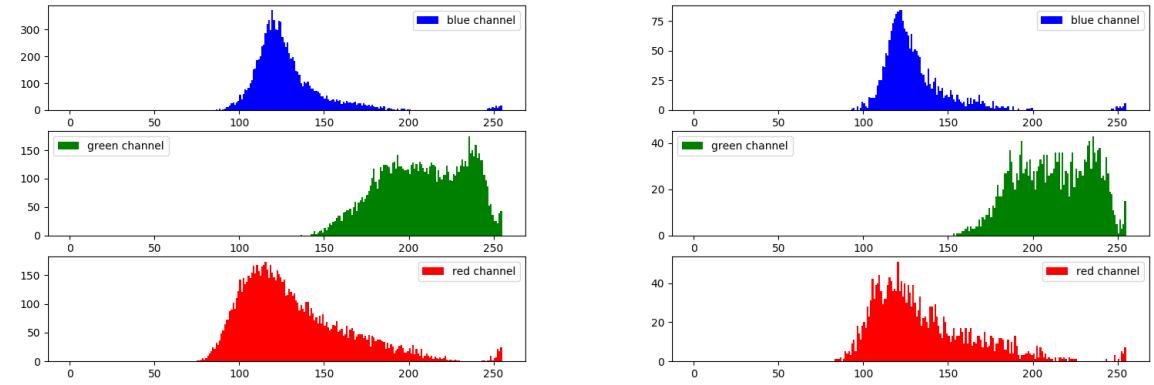
Figure 4: Histogram of Orange Buoy

On observation, we can estimate the number of 3D Gaussians that are necessary to model each colour to be the number of peaks that are obtained in the histogram. This is akin to fitting a 3D gaussian to the "average" of all images. On the basis of this strategy, based on the histograms shown here we take each color to be a mixture of 2, 3D Gaussians. In the adjoining figures, we compare histograms for training as well as test dataset, and the results are very similar, validating our assumption of a 2 Gaussian Mixture.



(a) Training Histogram (b) Testing Histogram

Figure 5: Histogram of Yellow Buoy



(a) Training Histogram (b) Testing Histogram

Figure 6: Histogram of Green Buoy

5 Learning Color Models

The subsequent step, after determining the number of Gaussians to be used to create the mixture model, is to determine the parameters of interest. These are the mean μ_j and covariance matrix $\sum_j$ of each color and this is determined through the EM algorithm detailed before.

Once we have decided the number of 3-D Gaussians to model each color, we initialize a mean value for each Gaussian along all its dimension (3×1). Along with this, we also initialize the associated covariance matrix (3×1). We read in images sequentially($R \times C \times 3$), and flatten them $(R \times C \times 3)$. Thereafter, for each the Gaussian models (2 Gaussians in our case), we perform the Estimation/ Expectation step i.e we find the probabilities of each pixel belonging to either one of the Gaussian's of that color. The probabilities are then normalized in order for the process to be

computationally relevant. For this estimation step, the *multivariate_normal.pdf* function in the *scipy.stats* module is used. Based on the obtained probabilites, we calculate the weights as mentioned in Step 2 which is then used to update the mean and co-variance of the distribution. This process of predict/estimate/expectation and update/maximization is repeated for every training image belonging to a particular color to train a color model. Such a process is repeated for a fixed number of iterations or up until the parameters converge below a certain threshold.

The final 3D Gaussian Mixture is depicted in the Fig 7. The model is saved in an npz file and can be reloaded when validation is required.

6 Segmentation on video stream

We arrive at the actual task of segmentation of the three bouys from the entire flattned image frame ($R \times C \times 3$). Using the saved gaussian mixture model, we calculate the probability of a particular pixel belonging to each colour model (Prediction vector shape = $R \times C \times 3$).Among the three classes, we identify the highest probability and thereby classify the image to belong to either "Green Model" or "Orange Model" or "Yellow Model". Then we only retain those pixels which are above a particular threshold for each color model. Visualising these pixels, we find certain outliers, certain empty pixels for which we use Morphological operations to get only the segmentations from the buoys.

In order to perform morphological operations we use the elliptical kernel rather than rectangular as the shape of the buoys is circular. We first use morphological opening to remove small blobs followed by closing opration to fill the gap and then finally dilate the image to enhance the detections. The segmentation we receive after dilation is show in figure 9. Once we have the segmented image, we calculate the expected radius of the buoy and plot it on the original image. Calculating radius is a bit tricky part as the detected blobs are not circular. We thus utilise the inbuilt functions such as $cv2.arcLength$ and $cv2.countourArea$ on the detected contours to determine the circularity of the blobs which is given by:

$$circularity = \frac{4\pi r^2}{perimeter^2}$$

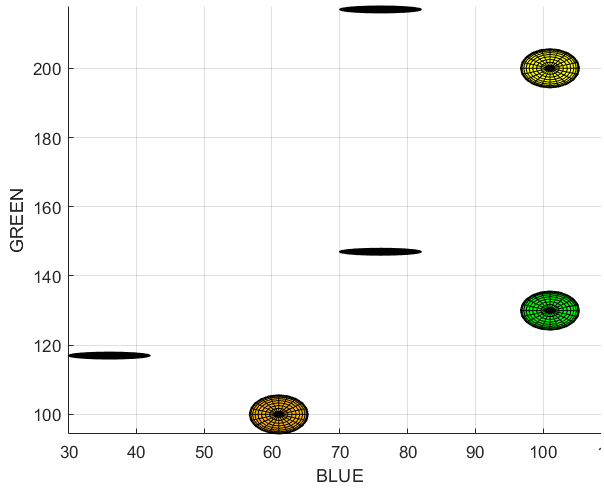
where perimeter is given by the *arcLength* function. We only consider the contours which have the value between 0.2 and 0.9. Then we find the size of the smallest circle that can enclose the segmented blob and then plot the circles. Th final detection is in figure 10.

7 Results

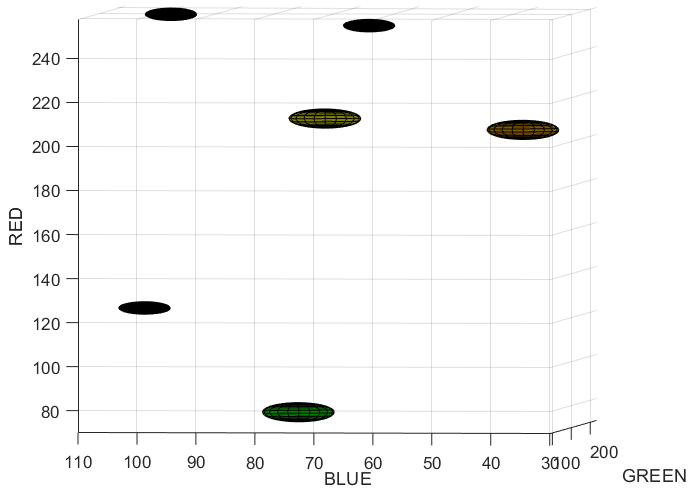
Our results of segmented buoys and video showing coloured circles around the respective buoys can be found here.

8 Challenges Faced

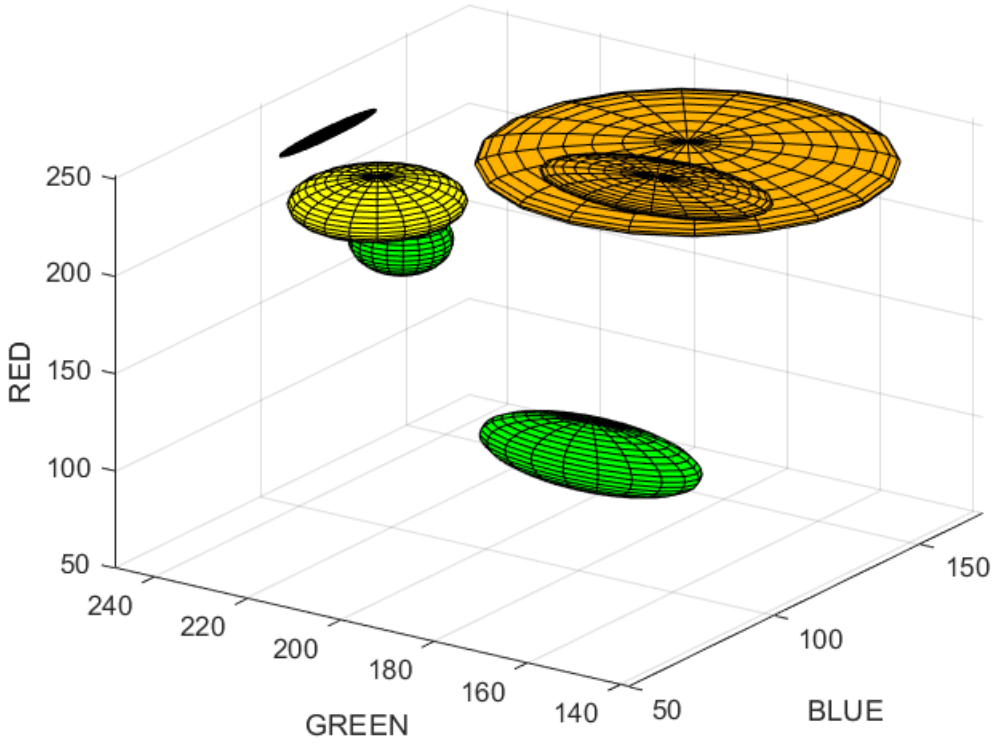
- We faced challenges with initializing the gaussian's mean and covariances. Intializing a very tight gaussian only outputed very small probablities(1e-50 or lesser) and hence later had issues of "divide by zero" or sometimes "NaN or inf"



((a))



((b))



((c))

Figure 7: (a) and (b) are the ellipsoid representation of the multivariate gaussians at the time of initialization. Here we have shown the two images to facilitate the comparison with (c) which is the ellipsoid representation of the gaussians obtained after Expectation Maximization.

Figure 8: Original image of the frame

Figure 9: Segmentation output of the frame

Figure 10: Final output of the pipeline

- We had difficulty training the yellow color model. Specifically,when we train over the entire yellow buoy image, we effectively train a Gaussian which models the light shade of yellow. It turns out that majority of pixels from the water tank also have the same color and hence is being classified as "Yellow" inspite of the high threshold. This was fixed by training the "Yellow" color model on only the bottom 65% of the yellow buoy training images.
- We found it hard to tune the thresholds for each of the color models as this is a iterative process.
- Just like point 2, we find a shade of green occurring at the bottom part of the buoys. Due to time constraints, we couldn't reject those outliers but suspect that it restricts tighter training.

********** END OF DOCUMENT **********