Project 3 – Buoy Detection

Submitted in partial fulfilment of the requirements for the course of

ENPM673 – Perception for Autonomous Systems

By

Adheesh Chatterjee

Andre Ferreira

Pablo Sanhueza

Course Faculty
Prof. Cornelia Fermuller

A. James Clark
SCHOOL OF ENGINEERING

INTRODUCTION

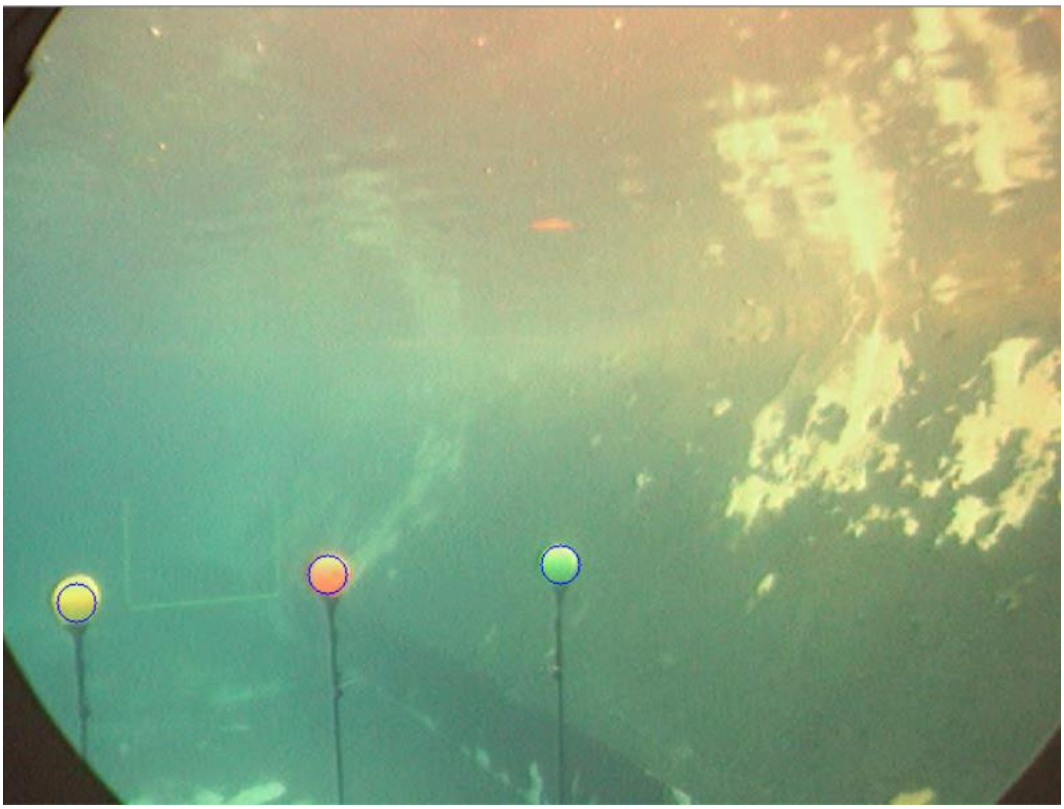
The aim of this project is to apply color segmentation using Gaussian Mixture Models and Expectation Maximization techniques. The concepts learned in class will allow us to write a script that solves the problem of buoy detection. Since the buoys have been captured underwater, change in lighting intensities and noise make it difficult to employ the use of conventional segmentation techniques involving color thresholding. Hence, this is accomplished with different methods explained and provided in the pipeline of the project guidelines. Below you will see explanations of what we did to accomplish such an arduous task.

DATA PREPARATION

1) Creating the Dataset

cropping.py is used to crop out the buoys. We read all the frames of the video one by one, plot circles on all 3 buoys and crop out the circles and save them in individual folders of red, green and yellow. We then split these images into our training and testing set. The pipeline is explained below.

We first define a click event which detects when the left mouse button (LMB) is pressed down using cv2.EVENT_LBUTTONDOWN and when the left mouse button is released using cv2.EVENT_LBUTTONUP. When the LMB is pressed, we plot a circle of radius 12 at that point. We find the center of the circle plotted and crop out a region between (x+8, y+8), (x+8, y-8), (x-8, y-8) and (x-8, y+8). This ensure that we only get the pixels of the buoy and minimize the effect of water in the background.



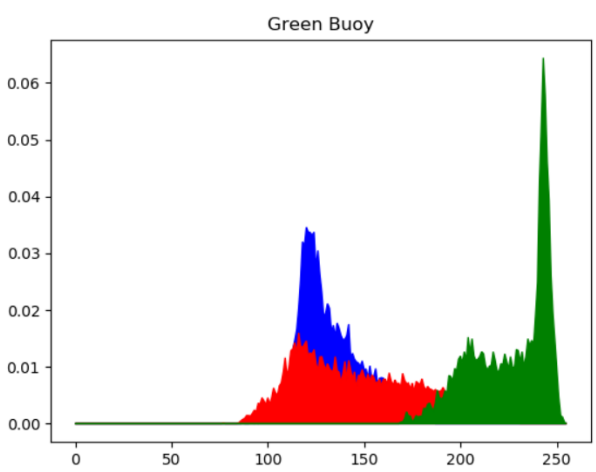
Green Buoy

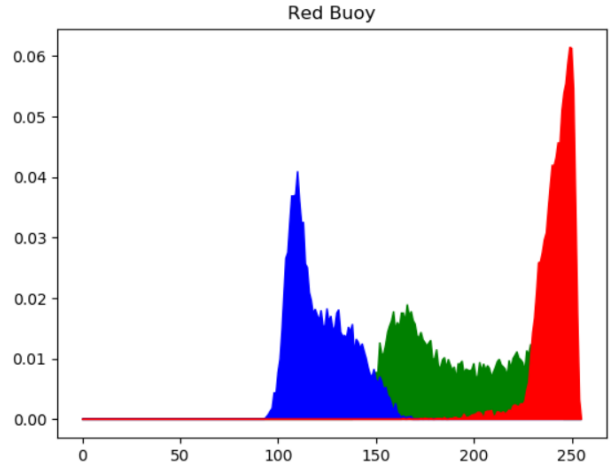
Red Buoy

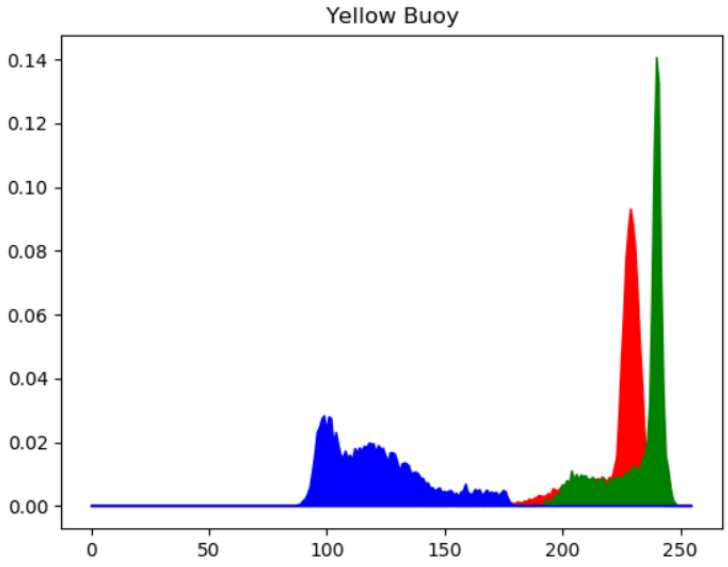
Yellow Buoy

2) Visualizing Histograms

We now compute the individual histograms for all 3 buoys by plotting the R, G, B channels individually. We then isolate individual channels for each buoy i.e. green channel for the green buoy, red channel for the red buoy and the combination of red and green channels for the yellow







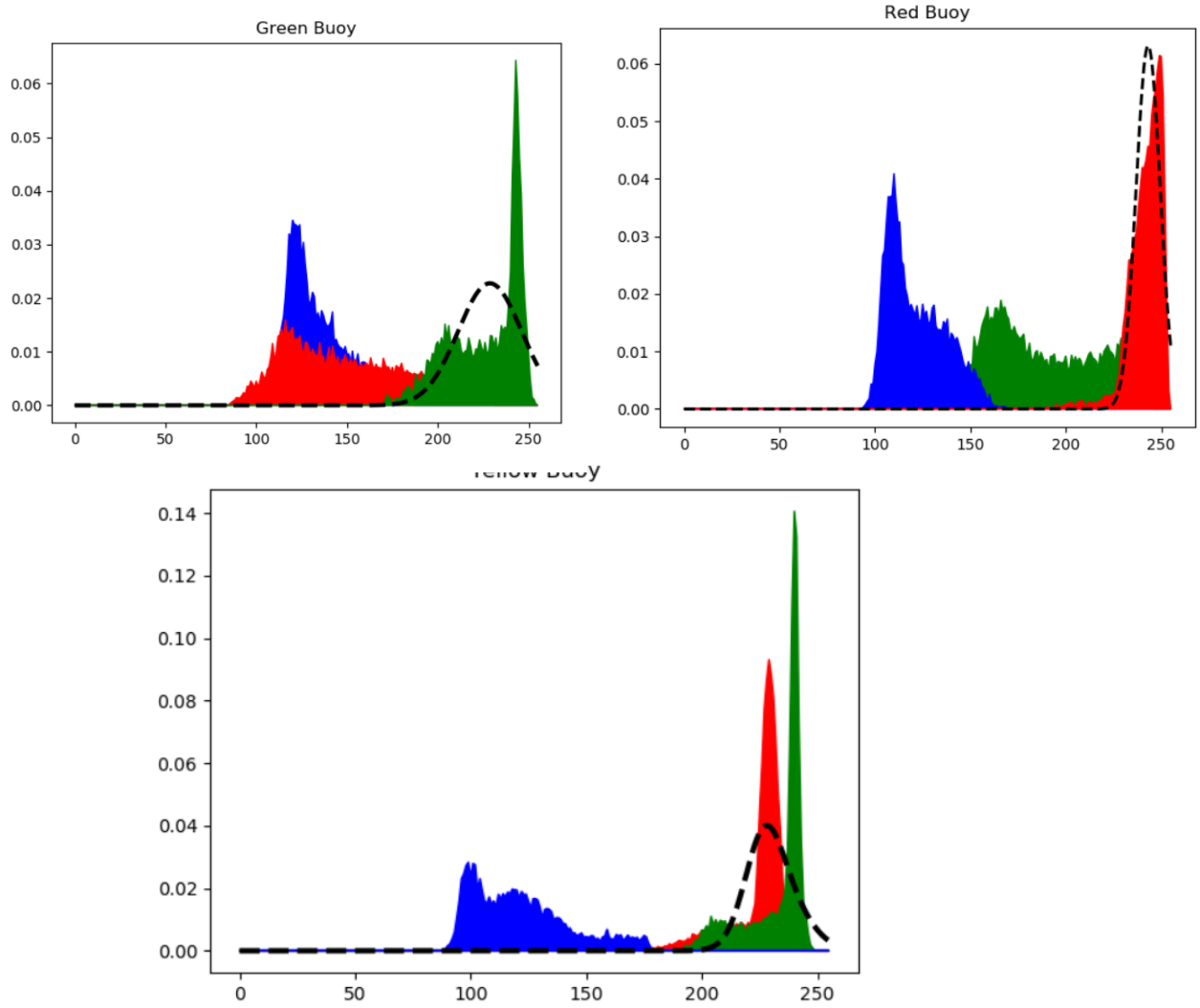
We observe from the above histograms that –

- **Green Buoy** – The green colour intensities for the green buoy vary from around 175 – 255. This indicates higher presence of green colour. The red and blue colours have lesser colour intensities
- **Red Buoy** – The red colour intensities for the red buoy vary from around 180 – 255. This indicates higher presence of red colour. The green and blue colours have lesser colour intensities
- **Yellow Buoy** – Since yellow is mixture of red and green, we notice that both red and green are higher on the histogram scale as compared to blue.

The presence of non-dominant colours is because of the noise during cropping and because the color of the buoy is not entirely pure. However, the main color associated with the buoys have the maximum counts with maximum intensities, which is the required outcome.

3) 1D Gaussian

We then fit a 1D Gaussian to the previously isolated channels to form the pdf by finding the mean and standard deviation of the Gaussian generated. We then use these to test values to test the video file and segment the buoys.



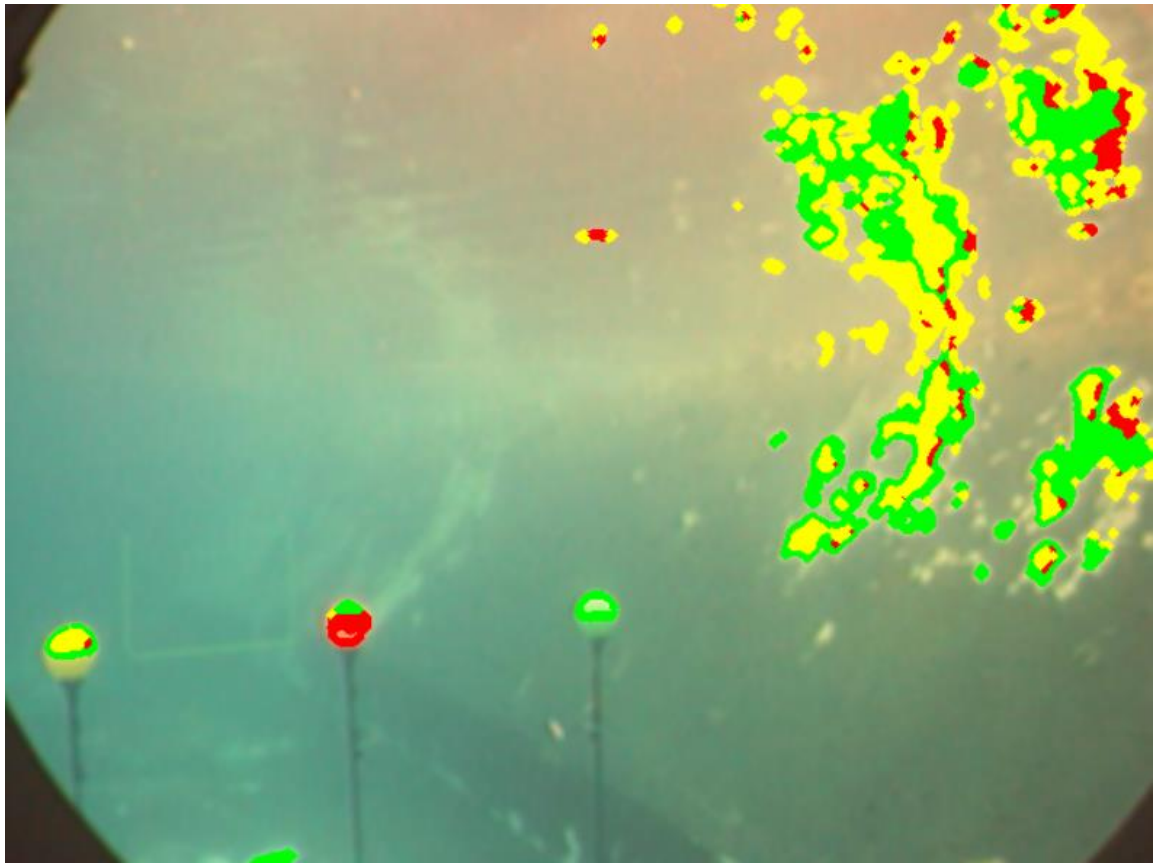
We observe from the above plots that –

- **Green Buoy** – The 1D gaussian doesn't fit the green buoy well. We see that the mean of the gaussian is not near the peak value, hence the pdf calculated may fit random non green points
- **Red Buoy** – The 1D fits the red gaussian sort of well. We see that the mean of the red buoy is very close to the peak value, hence the pdf calculated will fit red points well and very few random non-red points may be detected.
- **Yellow Buoy** – The 1D gaussian doesn't fit the yellow buoy well either. Since the yellow is a combination of the red and green channels,the gaussian may pick up a lot of random non-yellow points.

Since we use only 1 1D gaussian, the gaussian tries to cover the entire range of the isolated colour and hence the peak values are sometimes ignored. This also increases th likelihood of identification of random points in that colour range.

Now we test these Gaussians on the original video. We take every pixel of each frame of the video and run their values in the pdf function for all 3 Gaussians. Whichever pdf function is higher, we assign that pixel to that specific Gaussian and plot them.

Since we were only testing this method to see if it would work, we did not plot specific colored circles for specific buoys. Instead we just plotted every detected point for every detected color.



We could obviously have optimized it to fit it better and plot circles around each buoys, but since training the data would have better fit it, we decided to use this method as a stepping stone for our next step where we would train multiple gaussians onto the data.

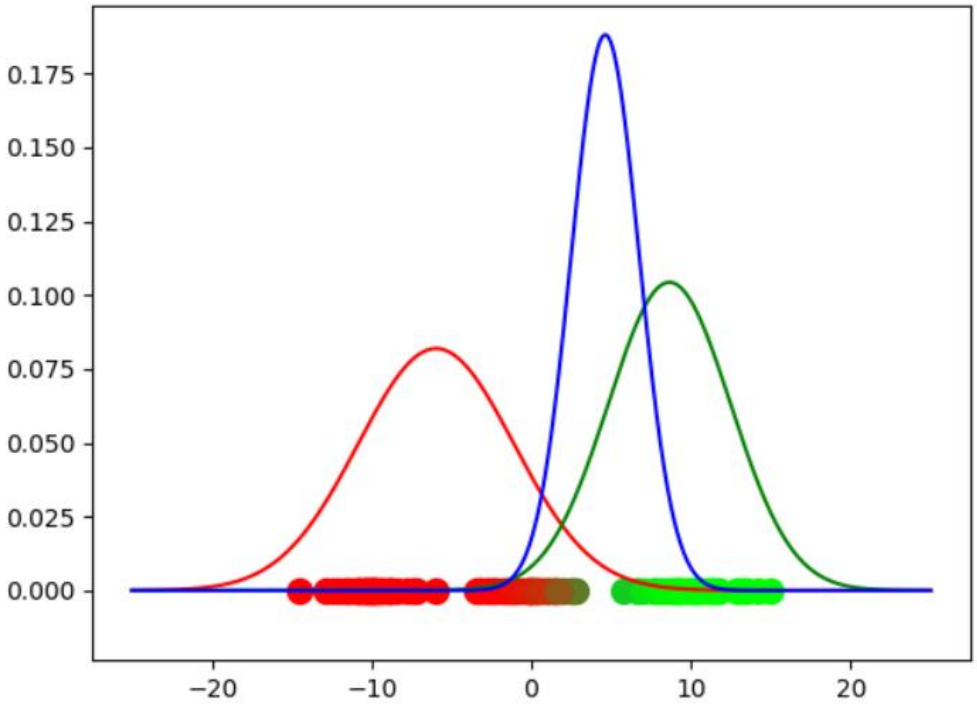
GAUSSIAN MIXTURE MODELS

1) Expectation Maximization

E Step

We create 3 random 1D data samples over a range of (-5, 5). We choose our initial guesses for mean as (-8, 8, 5) and standard deviation as (5, 3, 1). We then create random gaussians using our initial guesses and the pdf function. We also choose an initial probability of each dataset belonging to each gaussian as (0.33, 0.33, 0.33). Hence each dataset has a 0.33 probability to belong each gaussian initially. This is the Expectation step.

We then plot them as shown –



So we have fitted three arbitrarily chosen gaussian models to our dataset. We use a new variable ***r*** in which we store the probability for each point x belonging to each gaussian g. Then, we plot the x points and colour them according to their probabilities for the three clusters. We can see that the points which have a very high probability to belong to one specific gaussian, has the colour of this gaussian while the points which are between two gaussians have a color which is a mixture of the colors of the corresponding gaussians.

M Step

Now for the M step. We iterate over the initial values updating the mean, standard deviation and initial probabilities using the total weight ***m*** (the fraction of points allocated to gaussian).

We use the following formulas to update the values –

$$m_{new} = \sigma \cdot r$$

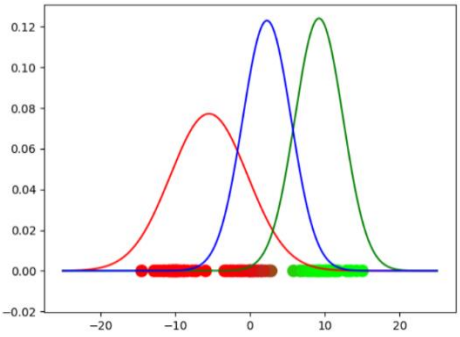
$$\pi_{new} = \frac{m_{new}}{m}$$

$$\mu_{new} = \frac{1}{m_{new}} \cdot (\sigma \cdot r \cdot x)$$

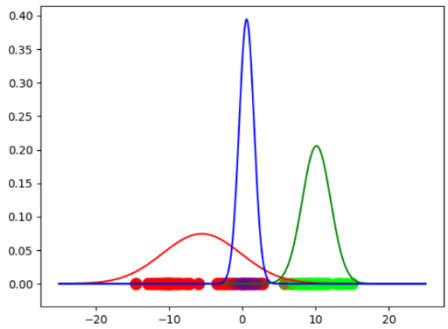
$$\sigma_{new} = \sigma \cdot r \cdot (x - \mu_{new})^T \cdot (x - \mu\_new)$$

Here, μ is the mean, σ is the standard deviation, π is the initial probabilities, m is the weight that it belongs to a particular cluster, and r as defined above in the E Step.

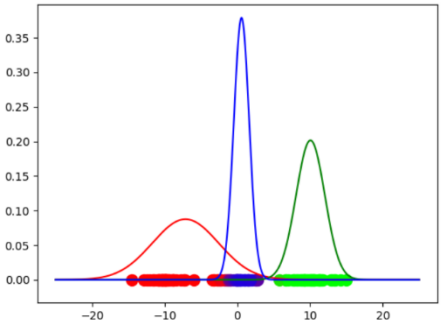
We run the loop for 100 iterations and break the loop when the absolute value of the difference between the means for all 3 gaussians in one frame and the means for all 3 gaussians in the next frame is less than 0.001, as this implies that the gaussians have successfully converged. We plot for each iteration as shown below –



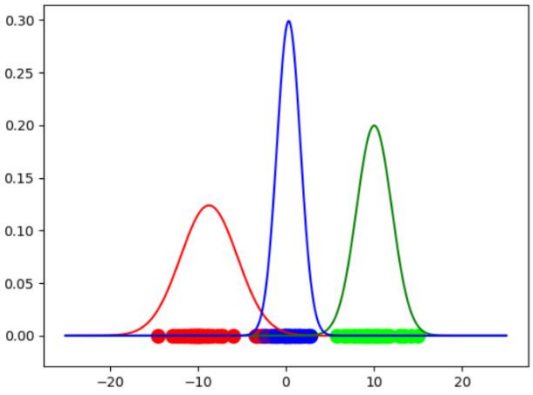
Iteration 3



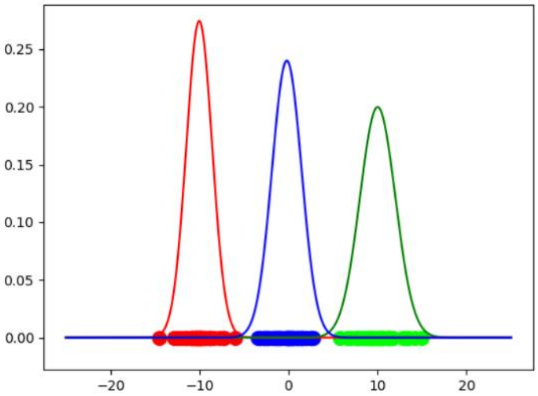
Iteration 7



Iteration 11

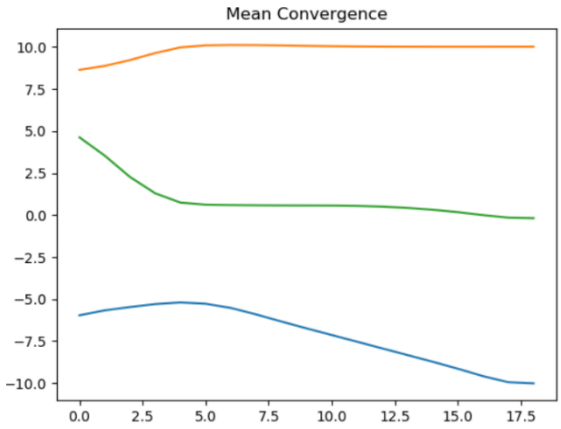


Iteration 15



Iteration 17

We observe that by iteration 17, the loop breaks and the gaussians fit the datasets perfectly.



We plot the mean convergence as shown.

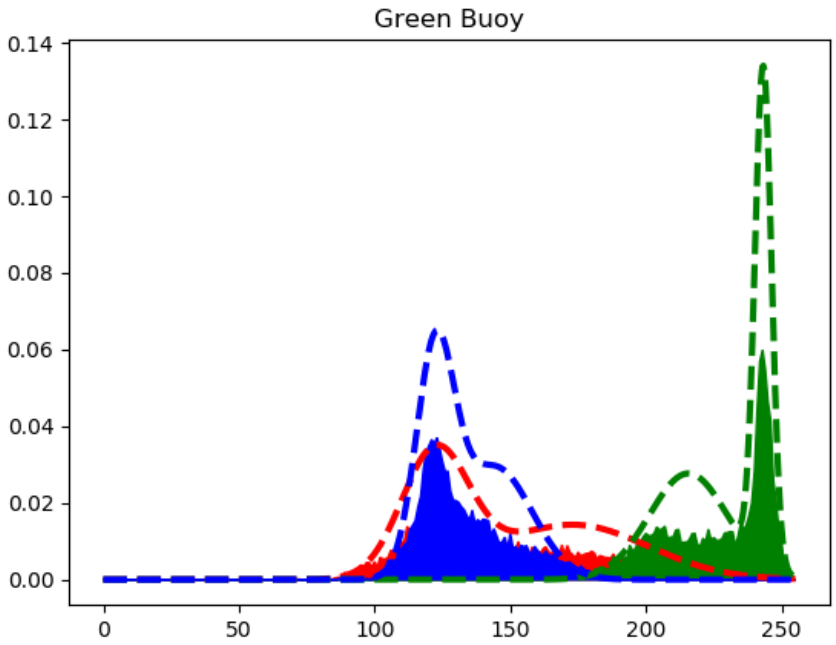
2) Learning Color Models

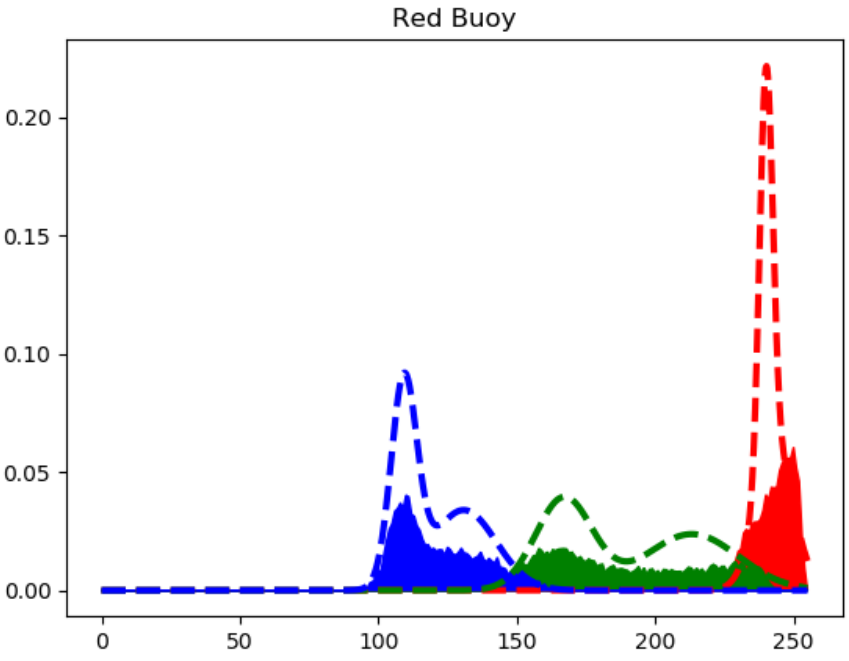
1D EM Implementation for Color Analysis

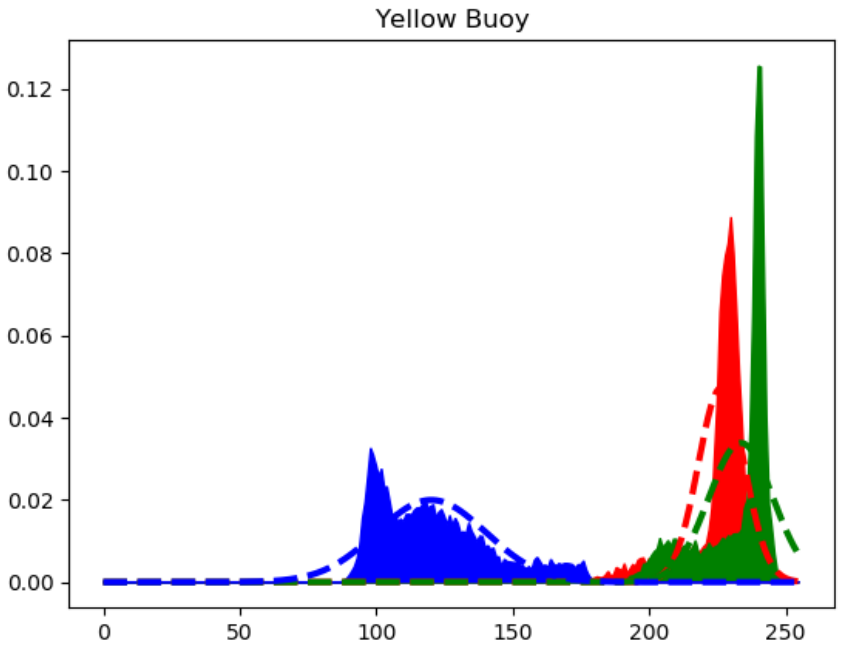
In order to properly identify the three buoys (yellow, red, and green), we utilized the Expectation Maximization (EM) algorithm to find the best fitting model for proper image segmentation. As a result, the group first analyzed each video frame by cropping each buoy and then stored it as training data. From these cropped images, we then found the histograms of each buoy according to each color channel (RGB).

Having all colored channels, we then began the training aspect of this project. Using all the histograms of the cropped buoy images, we found an average buoy for each colored channel. With this information we created a function that backtracks the pixel intensity counts due to the fact that the average counts of pixel intensities are displayed as decimal values (for example an average count of 7.64 pixels for a pixel intensity of 235). Using this data we implemented the Expectation Maximization algorithm to obtain the best Gaussian fit for each colored channel.

By analyzing the histograms we were able to differentiate the main peaks, and therefore determine the K value for the EM algorithm. This K value determines the number of clusters or Gaussian's that the algorithm need to differentiate. For example, if the green color channel of the green buoy has 3 main peaks, we will then run the EM for three clusters and obtain the means and standard deviations for three Gaussians. To obtain the perfect fit it is possible to add Gaussians and plot them to fit the data given. Shown below are the fit examples of the Gaussian fits for each colored buoy.


Green Buoy
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
50
100
150
200
250


Red Buoy
0.20
0.15
0.10
0.05
0.00
0
50
100
150
200
250



Shown in the figures above, we can see that the Expectation Maximization algorithm fits all channels almost perfectly. The dashed lines are the fitted Gaussians to the training dataset.

3D EM Implementation Color Analysis

After analyzing the 1D EM implementation the group decided to use a multivariate Gaussian distribution to obtain a relationship between all the channels (RGB) for each buoy. As a result, the entire EM function was changed to fit a 3D Gaussian. The PDF calculation for a multivariate Gaussian distribution can be utilized not for a 3D Gaussian, but for almost any dimension. Shown below is the equation followed during the implementation.

$$p(x|\mu,\Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\}$$

$$n - Dimension\ of\ gaussian\ distribution$$

$$\sum - Sigma\ is\ the\ covariance\ matrix\ (3x3\ matrix)$$

$$\mu - Mean\ (1x3\ vector)$$

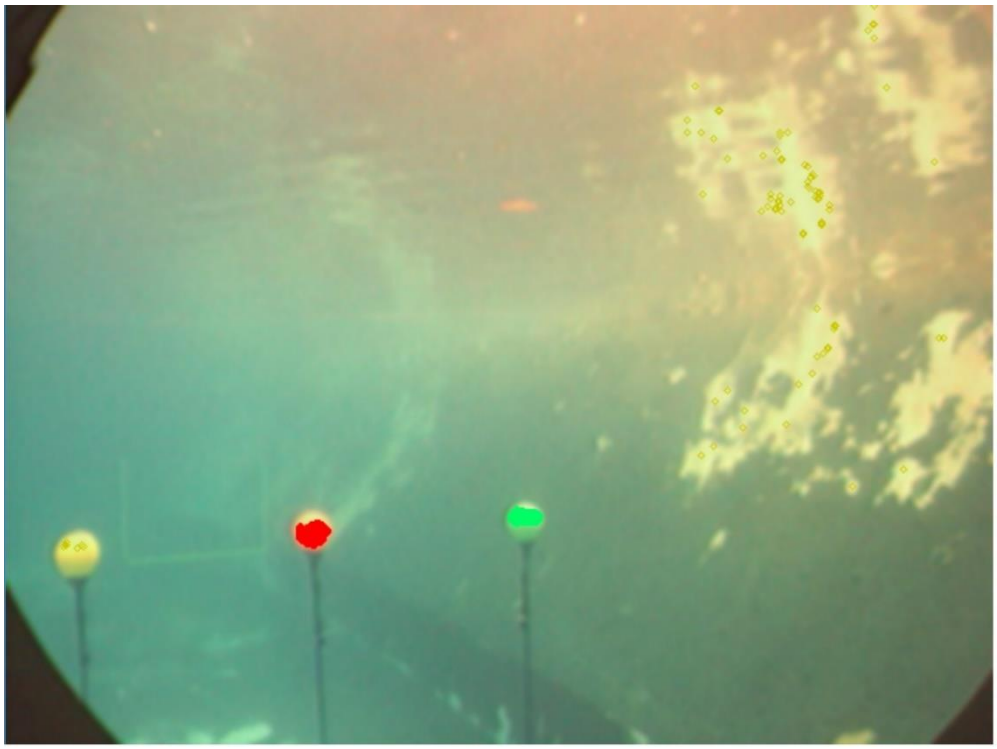
To implement this distribution with the EM algorithm we need to treat each variable as matrices and vectors. For example, μ is a vector with three means, and $\sum$ is the covariance matrix that shows the relationship between datasets. In this case, the covariance matrix is a 3x3 matrix.

The data utilized for the 3D Gaussian implementation was similar to what was done for the 1D section. The slight discrepancy that we thought would achieve better results was by stacking

all training images of each buoy by color together. This provided an exact number of pixels that were found in the training cropped images and were one of the inputs to the multivariate EM algorithm.

Some differences with the 1D implementation is that the multivariate algorithm outputs a vector of three means for 1 3D Gaussian. The 1D implementation always outputs one mean for each cluster, whereas for the 3D implementation it's 3 means for each cluster. After running the algorithm for several iterations, we found the convergence of the means and covariance's and used these to determine if a 3D implementation was feasible or not. Histogram plots are not provided due to the fact that plotting a 3D Gaussian would need a fourth dimension to display the likelihood or probability values.

To determine if this implementation was good for the task, we plotted tiny circles onto the original video frame to determine if the buoys were being detected or not. Shown below is an example of the results of the 3D implementation. It is very noticeable that there were good results for both the green and red buoy, but the yellow buoy was not as distinct as the other two. To achieve these results we applied a small threshold to each buoy separately to get rid of small outliers in the surroundings of the buoys. Even though a threshold was applied, there were still several outliers throughout each frame of the video, especially towards the end.



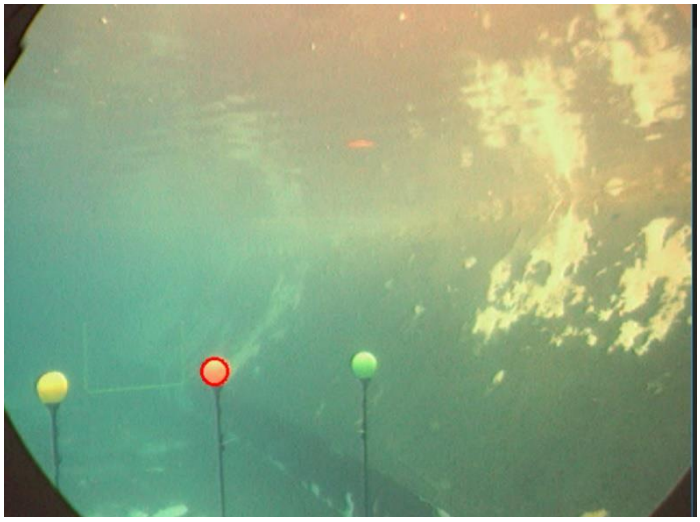
3) Buoy Detection

Now after analyzing both the fit of a 3 1D Gaussians and 1 3D Gaussian, we observe that the binary image of 3D input has a lot of noise near the buoys and therefore it is difficult to segment them. Even after applying thresholding values, the noise near the buoys was hindering the segmentation process.

Compared to 1D input, even though we identify multiple colored buoy, the noise near the buoys is less and therefore it is easier to segment them. Therefore, for the final pipeline of the buoy detection, we have used 1D input with multiple Gaussians.

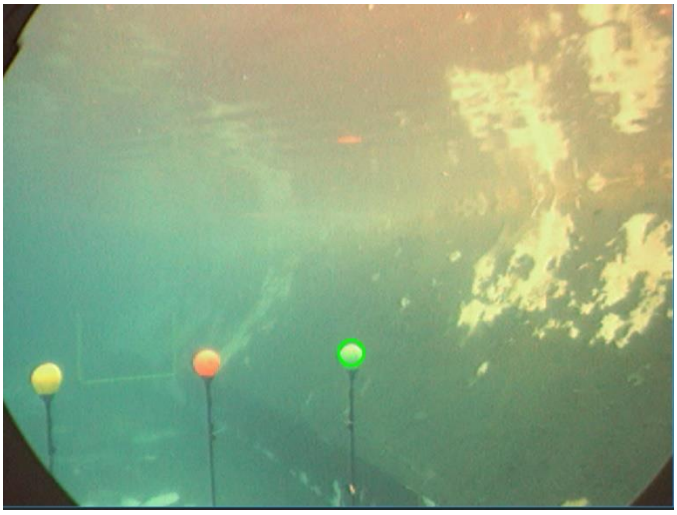
We chose the red channel for the red buoy and fit 3 Gaussians to it. We chose the green channel for the green buoy and fit 3 Gaussians to it as well. We chose both the red and green channels for the yellow buoy and fit 3 Gaussians to each channel.

We created a binary image where we plotted the detection of each individual buoy using its own threshold. After applying morphological operations to clean our image, we used contours on our binary segmented image to detect all the clustered points. We then used cv2.minEnclosingCircle and created a circle using the points detected in our binary image. This circle was then plotted in our original video frame. We essentially processed the color model developed in our binary image and plotted the output in our original video.

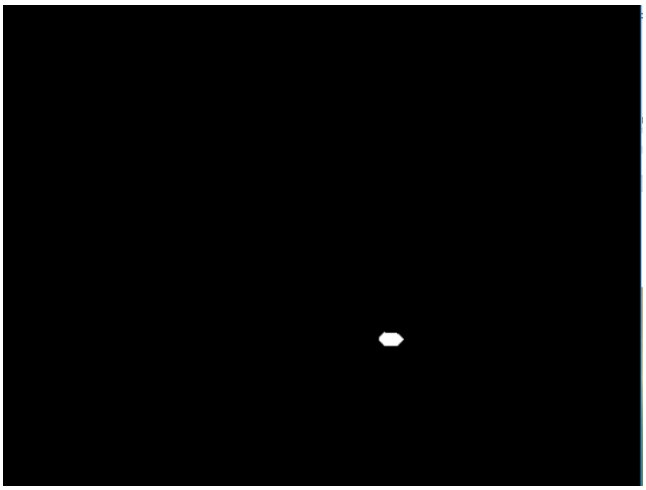


Original Video (Red Buoy)

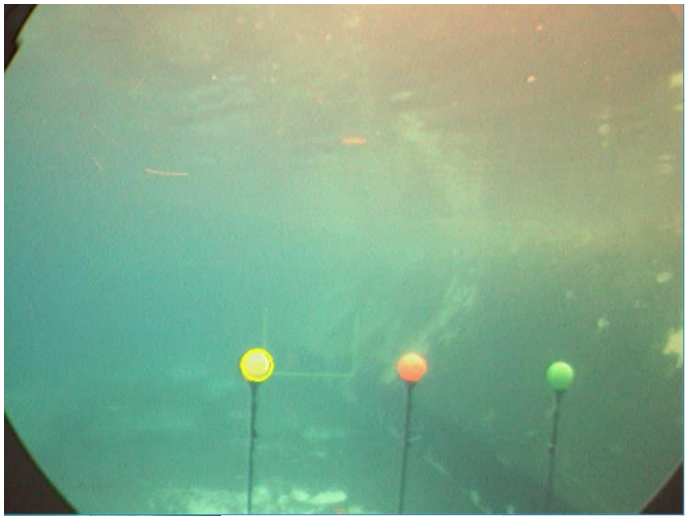
Binary Image (Red)



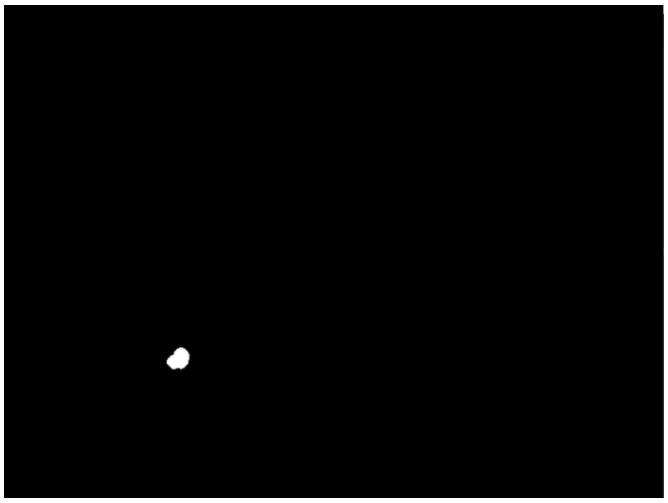
Original Video (Green Buoy)



Binary Image (Green)

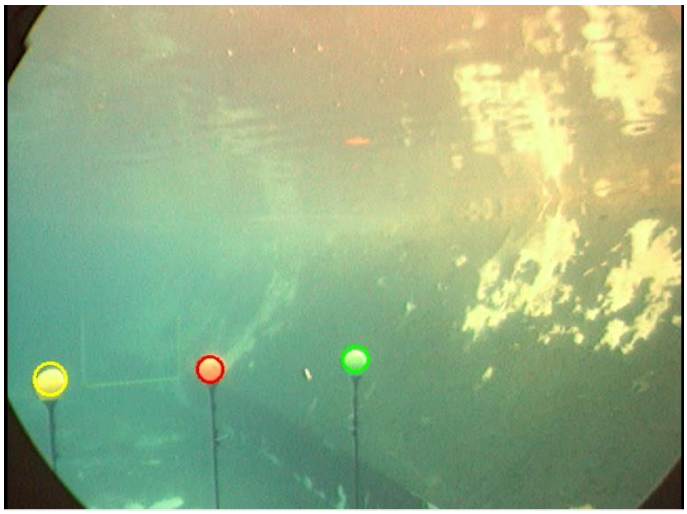


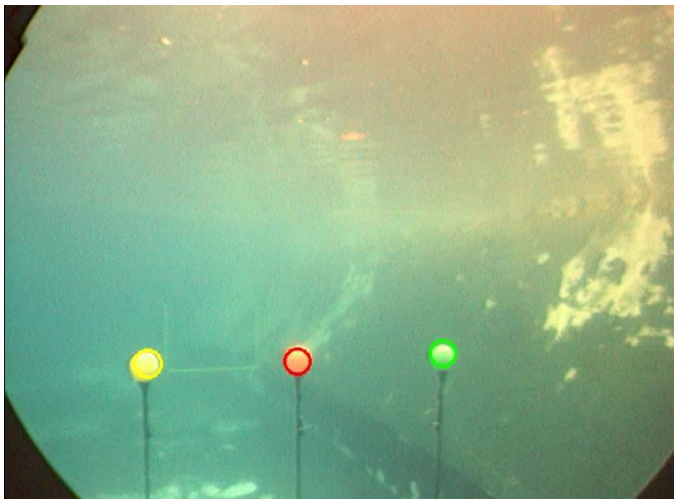
Original Video (Yellow Buoy)

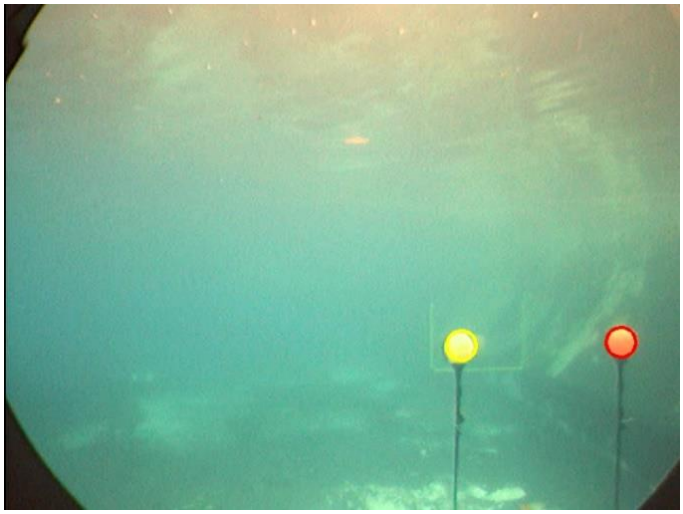


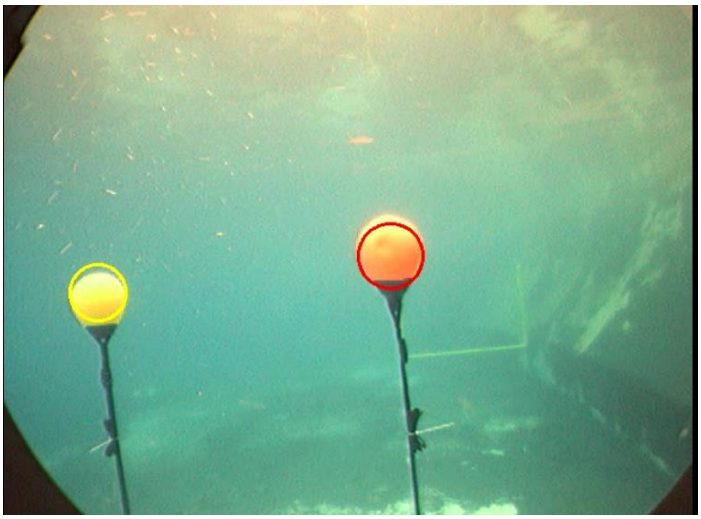
Binary Image (Yellow)

The final buoy detection for all 3 buoys was thus found –









Further Discussion – Color Space

It is accurate to conclude that using RGB channels is one of the main problems faced for color detection, due to the fact that the color of the buoys aren't of pure color. Also, there is a lot of noise because of the background, the lighting color, and the moving of the camera which makes it difficult to segment the buoys accurately. As a result, on the outset, our main issue was to get rid of different intensity levels of the same color hue, and that's when we realized that the HSV color space could solve this issue. The HSV color space has different parameters for the HUE, the SATURATION and the VALUE (intensity) of the pixel and it will be more adaptive to the changes in the images such as lighting, shadow or any such noises. The hue would stay relatively constant for the buoys, the differences in lighting would be represented through the value component of each color.

The LAB color space should technically also work properly, as it will account for Lighting as the name suggests. Also, it has separate planes for the A and B channels, where A is the green/red channel and B is the yellow/blue channel and this should make detection easier.

References

[1] https://www.youtube.com/watch?v=gMTuMa86NzU

[2] https://www.cs.bgu.ac.il/~ben-shahar/Teaching/Computational-Vision/StudentProjects/ICBV121/ICBV-2012-1-OfirNijinsky-AvivPeled/report.pdf