ENPM 673: Perception for Autonomous Robots **Released: Mar-23.**

Report

Name: Arpit Aggarwal and Shantam Bajpai

1 Data Preparation

The first part of the project was the data preparation phase. We used the code reference provided in the report to get the buoy image. To ensure we got the best possible polygon we selected the points within the buoy and used the "cv2.minEnclosingCircle()" function to get the buoy image. The example of segmented yellow, orange and green buoys is shown below:

Figure 1: Segmented Yellow Buoy

Figure 2: Segmented Orange Buoy

Figure 3: Segmented Green Buoy

To decide the number of Gaussians for each buoy, we plotted the average color histograms for each channel of the buoy. The number of Gaussians for the buoy were estimated according to the number of peaks in the average histogram for each channel of the buoy.

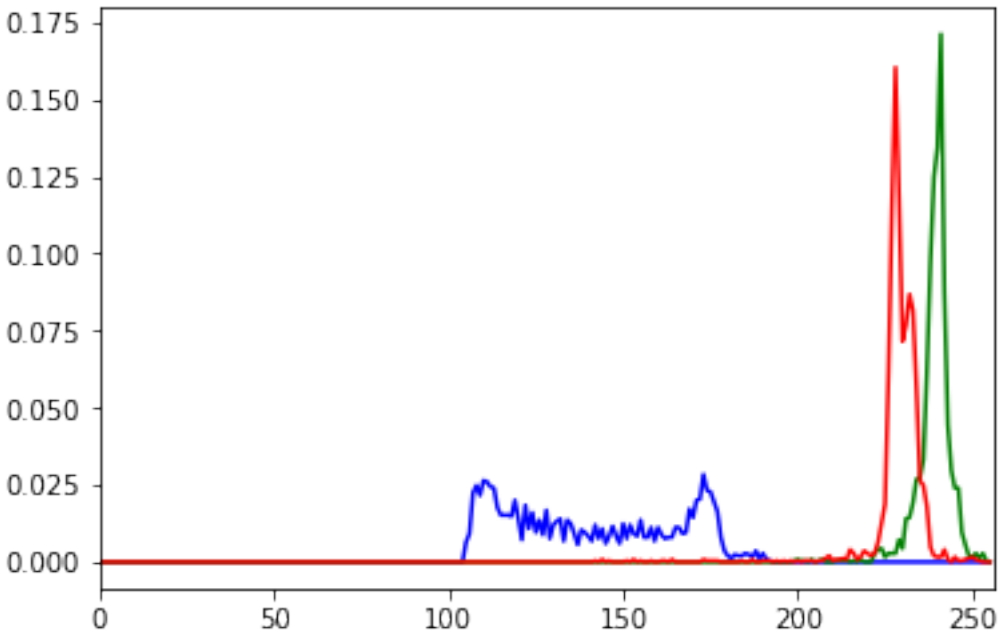

Figure 4: Average color histogram of the Segmented Yellow Buoy

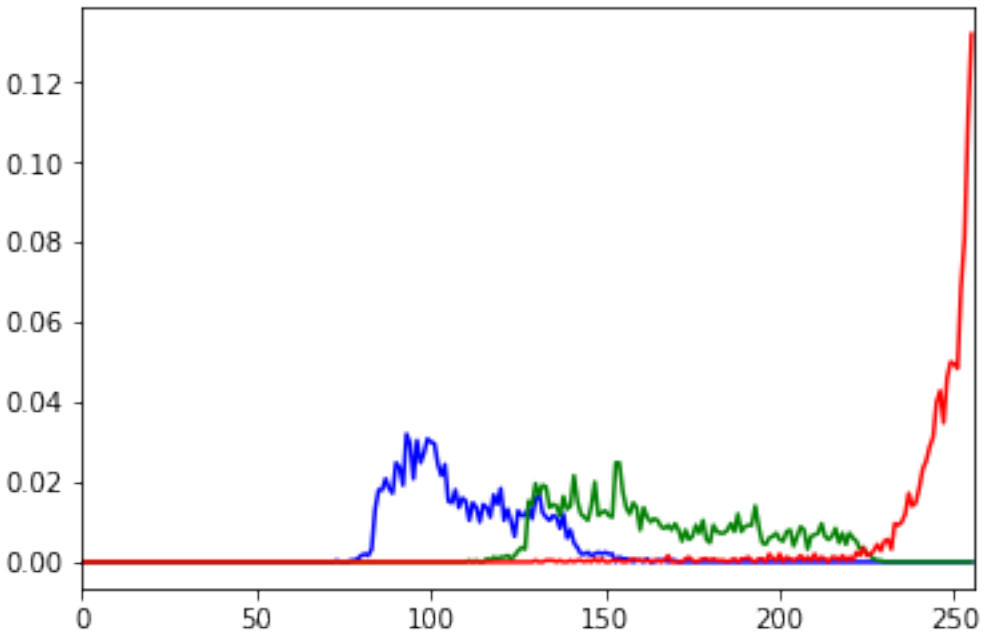

Figure 5: Average color histogram of the Segmented Orange Buoy

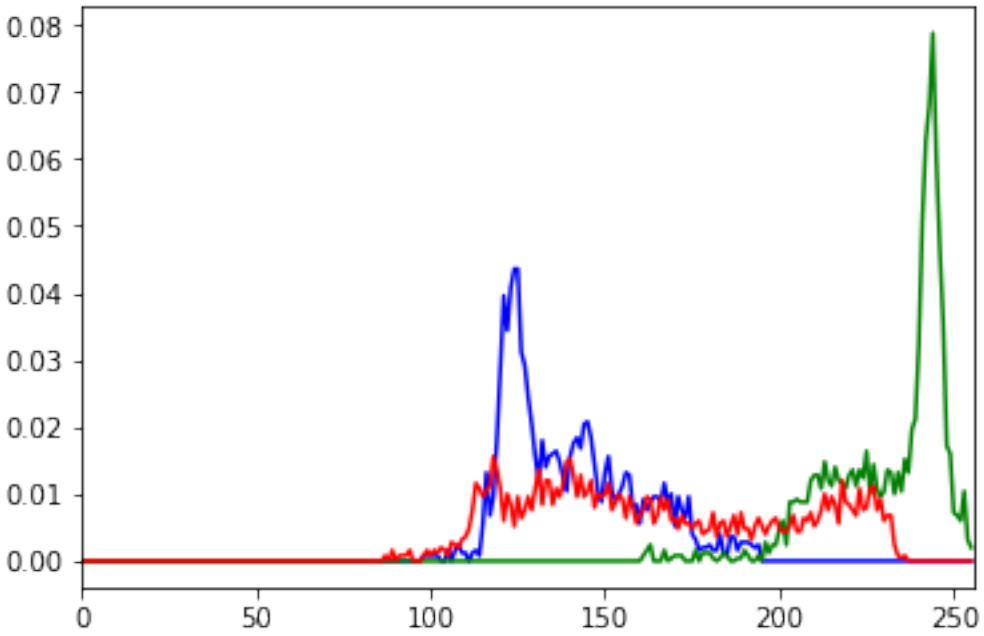

Figure 6: Average color histogram of the Segmented Green Buoy

2 Modelling Color Distribution of buoys using 1-D Gaussian

Now , for creating the initial Gaussians from all the color channels we can compute the ground truth values for mean and co-variance by using the Maximum Likelihood Estimate (MLE). The datapoints are generated by the above histograms. In the maximum likelihood estimate, we maximize the log likelihood function so that under the assumed statistical model the data that has been generated is most probable. As we need to compute the model parameters $\theta = \{\mu, \Sigma\}$ we maximize the joint likelihood function. Mathematically,

$$\hat{\theta_{ML}} = \arg\max_{\theta} \; p(x_1 ... x_n|\theta)$$

Now by the optimality criterion we have that the maximum of a function is achieved when its gradient equals zero. Now as the data produced is independent and identically distributed we compute the gradient as

$$\nabla_\theta \prod_{i=1}^{n} p(x_1 ... x_n|\theta) = 0$$

Now when we take the log of this function the equation gets simplified as

$$\nabla_\theta \sum_{i=1}^{n} \ln p(x_1 ... x_n|\theta) = 0$$

Replacing the likelihood function with the gaussian distribution, taking its logarithm and then computing the gradient along the μ and Σ we get the maximum likelihood mean and the covariance as

$$\hat{\mu}_{ml} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

and

$$\hat{\Sigma}_{ml} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu}_{ml})(x_i - \hat{\mu}_{ml})^T$$

After obtaining these equations we can compute the ground truth values for the mean and covariance base upon our datapoints and then compute the gaussian distribution based on the mean and covariances obtained.

3 Pipeline for Detecting Buoys

The pipeline we followed for detecting the buoys using 1D-Gaussian was as follows:

1. First, we selected the number of Gaussians(k) to be used for segmenting the buoy. For example, for yellow buoy we took k=3, for orange buoy we took k=2 and for green buoy we took k=2. Then, we decided the dimension of the Gaussian, i.e we took d=1 and d=3 for buoy segmentation.

2. Next, we got the training data for the buoy and then ran the expectation-maximization algorithm to get the weights, means and variances of the k-Gaussians in the Gaussian Mixture Model(GMM).

3. Simultaneously we computed the log likelihood of the data with respect to the gaussian parameters π_k, μ_k, Σ_k and if the log likelihood value was greater than the previous iteration we can say with confidence that our loss is converging towards zero and hence we would be able to select the optimal values for modelling our gaussian distribution for our buoys.

4. After finding the parameters for the Gaussian Mixture Model(GMM), next step was running on each frame of the video for the buoy segmentation.

5. For each frame of the test video, we found the probability of each pixel of the frame lying in the k-Gaussians. For Green buoy, we took the green channel values into account, for yellow buoy we took the green channel and red channel values into account and for orange buoy we took the red channel values into account.

6. Next, we used a threshold on the probability values to get a binary image. The "1" in the binary image meant that the pixel had a high probability lying the k-Gaussians of that buoy while the "0" meant the pixel had less probability of lying in the k-Gaussians. Examples of binary image for yellow buoy, orange buoy and green buoy are shown in Figure 7.

7. Lastly, contour detection was used for segmenting the buoy using cv2.findContours() function.

4 Example outputs for fitting 1D-Gaussian for Orange Buoy

For Orange Buoy, we used the pixel values from the red channel as data points for training. The number of gaussians(k) was taken as 2 and the dimension of gaussian(d) was taken as 1. After find-

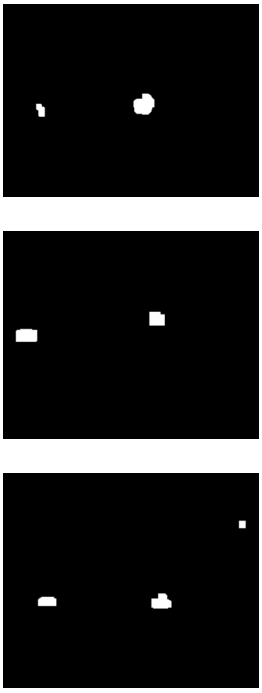

Figure 7: Binary Images obtained using 1D Gaussian

ing the weights, mean and variances of the k-Gaussians in the Gaussian Mixture Model(GMM), the orange buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 8).

5 Example outputs for fitting 1D-Gaussian for Green Buoy

For Green Buoy, we used the pixel values from the green channel as data points for training. The number of gaussians(k) was taken as 2 and the dimension of gaussian(d) was taken as 1. After finding the weights, mean and variances of the k-Gaussians in the Gaussian Mixture Model(GMM), the green buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 9).

6 Example outputs for fitting 1D-Gaussian for Yellow Buoy

For Yellow Buoy, we used the pixel values from the green and red channel as data points for training. The number of Gaussians(k) was taken as 2 and the dimension of Gaussian(d) was taken as 1. After finding the weights, mean and variances of the k-Gaussians in the Gaussian Mixture Model(GMM), the yellow buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 10).

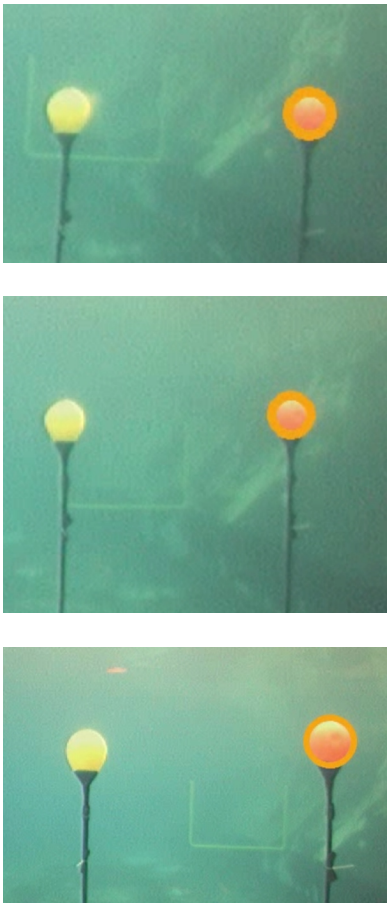

Figure 8: Segmenting Orange Buoy using 1D Gaussian

7 Video outputs for 1D Gaussian Case

The following are the video outputs for 1D Gaussian case:
1. Green Buoy:
https://drive.google.com/file/d/1c7sCpCfFSPFj0L7QAg-KP9DXe7yYJxOn/view?usp=sharing

2. Orange Buoy:
https://drive.google.com/file/d/1L7GVmyw3Y6AdLBBXdIFvcszeAp3yuHTV/view?usp=sharing

3. Yellow Buoy:
https://drive.google.com/file/d/1B4AZc6Ippshhfa6X6WZkgKwpBZXQEiP-/view?usp=sharing

8 Theory of the EM Algorithm and Gaussian Mixture Models

The expectation-maximization algorithm states that given a set of observed data the model parameters and the set of latent variables , it seeks to find out the value of the latent variables which when introduced in the model keeps the marginal distribution unchanged. In general if the latent

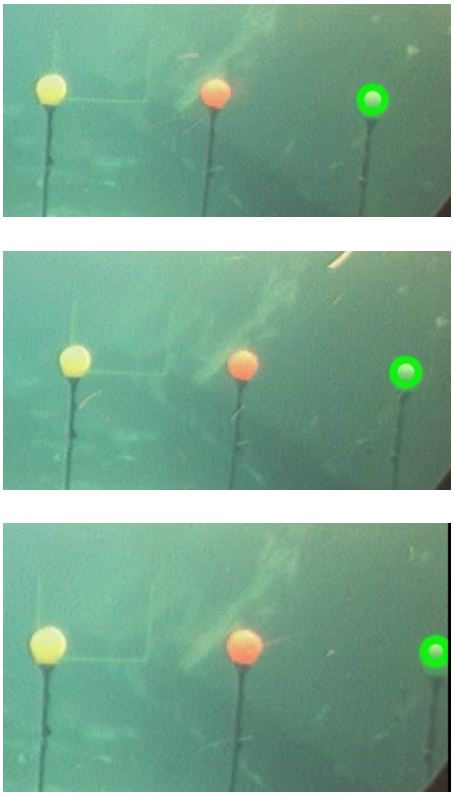

Figure 9: Segmenting Green Buoy using 1D Gaussian

variables were observed(missing data for eg) then the following optimization problem would be easy to solve

$$\mu_{ml}, \Sigma_{ml} = \arg\max_{\mu,\Sigma} \sum_{i=1}^{n} \ln p({x_i}^o, {x_i}^m | \mu, \Sigma)$$

Hence EM algorithm helps us in predicting the values of the missing variables based upon a set of observed variables by optimizing the marginal distribution $p(x|\theta_1)$. We use $p(x, \theta_2|\theta_1)$ for convenience where θ_2 is a prior. The marginal likelihood can be mathematically written as

$$p(x|\theta_1) = \int p(x, \theta_2|\theta_1) d\theta_2$$

Now the EM algorithm seeks to find the Maximum Likelihood or the Maximum A Posteriori estimate and this can be achieved by using the following EM Objective function

$$\ln p(x|\theta_1) = \int q(\theta_2) \ln \frac{p(x, \theta_2|\theta_1)}{q(\theta_2)} + \int q(\theta_2) \ln \frac{q(\theta_2)}{p(\theta_2|x, \theta_1)}$$

Here $q(\theta_2)$ is any continuous probability distribution.
Through the EM Algorithm we seek to iteratively optimize $\ln p(x|\theta_1)$ with respect to θ_1. Basically we want to generate values for θ_1 such that $\ln p(x|\theta_1^{(t)}) \geq \ln p(x|\theta_1^{(t-1)})$ so that $\theta_1^{(t)}$ converges to a global maximum as logarithm is a monotonically improving function.

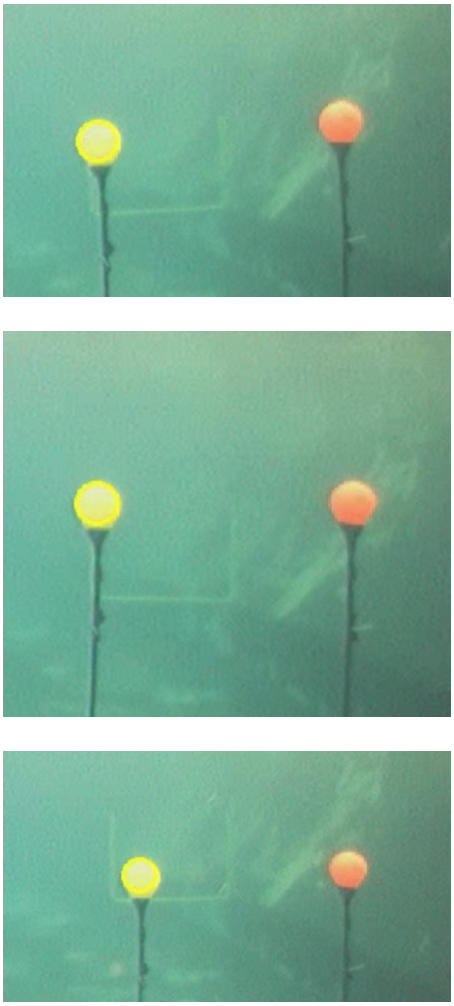

Figure 10: Segmenting Yellow Buoy using 1D Gaussian

8.1 Gaussian Mixture Model

In a gaussian mixture model let π be a K dimensional probability distribution/mixture coefficient with the constraint that $\sum_{k=1}^{K}\pi_k = 1$ and (μ_k, Σ_k) be the mean and covariance of the k^{th} gaussian distribution. The goal is to learn π,μ,Σ.
We employ the Expectation maximization technique to learn the parameters from a gaussian mixture model.
The modified EM objective function for the Gaussian Mixture model with the cluster assignment vector $c_i = (c_1.....c_k)$ which gives us the cluster assignment for the ith datapoint. Now we can mathematically represent the objective function as

$$\sum_{i=1}^{n} \ln p(x_i|\pi,[\mu],\Sigma) = \sum_{i=1}^{n}\sum_{k=1}^{K} q(c_i=k)\ln\frac{p(x_i,c_i=k|\pi,\mu,\Sigma)}{q(c_i=k)} + \sum_{i=1}^{n}\sum_{k=1}^{K} q(c_i=k)\ln\frac{q(c_i=k)}{p(c_i=k|x_i,\pi,\mu,\Sigma)}$$

Now Given the data points $(x_1.....x_n)$

Initialize the Mixture Coefficient, mean and covariance with some value and choose K i.e. the

number of Gaussian Mixtures. The size of the mixture coefficients should be $(K \times 1)$, the mean should have size $(k \times d)$ where d is the dimension of the gaussian and the covariance matrix should have the size $(k \times d \times d)$

E-Step:

For $i = 1.....n$ set

$$\phi_i(k) = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i|\mu_j, \Sigma_j)}$$

This term basically gives the probability of the ith data point coming from the kth gaussian distribution.

M-Step:

For $k = 1.....K$ define $n_k = \sum_{i=1}^{n} \phi_i(k)$ Now the expressions for achieving the mean, mixture coefficients and the covariance for the kth gaussian distribution is

$$\pi_k = \frac{n_k}{n} \quad \phi_k = \frac{\sum_{i=1}^{n} \phi_k x_i}{n_k} \quad \Sigma_k = \frac{\sum_{i=1}^{n} \phi_i(k)(x_i - \mu_k)(x_i - \mu_k)^T}{n_k}$$

After performing the M-Step we can assess the convergence of the function $\ln p(x|\pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}))$ as a function of the number of iterations.

9 Example outputs for fitting 3D-Gaussian for Green Buoy

For Green Buoy, we used the pixel values from the green, red and blue channels as data points for training. The number of gaussians(k) were taken as 4 and the dimension of gaussian(d) was taken as 3. After finding the weights, mean and variances of the k-Gaussians in the Gaussian Mixture Model(GMM), the green buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 11).

10 Example outputs for fitting 3D-Gaussian for Orange Buoy

For Orange Buoy, we used the pixel values from the green, red and blue channels as data points for training. The number of gaussians(k) were taken as 6 and the dimension of gaussian(d) was taken as 3. After finding the weights, mean and variances of the gaussian mixture model, the orange buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 12)

11 Example outputs for fitting 3D-Gaussian for Yellow Buoy

For Yellow Buoy, we used the pixel values from the green, red and blue channels as data points for training. The number of Gaussians(k) was taken as 6 and the dimension of gaussian(d) was taken as 3. After finding the weights, mean and variances of the k-Gaussians in the Gaussian Mixture

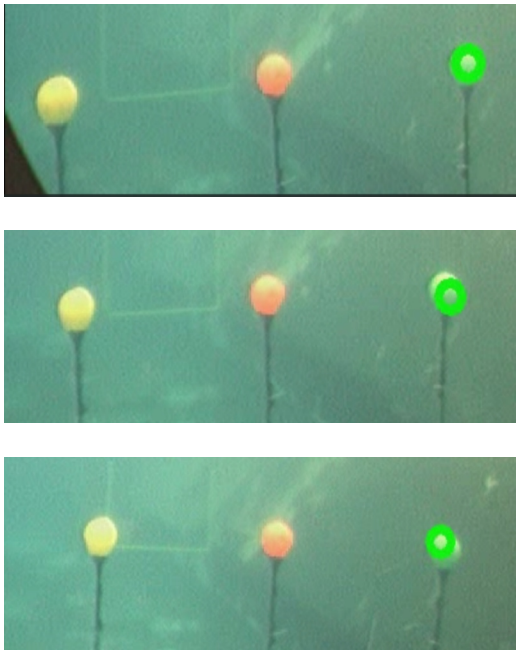

Figure 11: Segmenting Green Buoy using 3D Gaussian

Model(GMM), the yellow buoy was segmented by using the OpenCV functions cv2.GaussianBlur(), cv2.threshold() and cv2.findContours(). The following are the outputs obtained. (Figure 13)

12 Video outputs for 3D Gaussian Case

The following are the video outputs for 3D Gaussian case:
1. Green Buoy:
https://drive.google.com/file/d/1KF7YapHArDj96n0UFk4Yir2W-_WTzNTD/view?usp=sharing

2. Orange Buoy:
https://drive.google.com/file/d/1LX8y2OswzIaLFnf8wHNFR6CwVS_W8PaJ/view?usp=sharing

3. Yellow Buoy:
https://drive.google.com/file/d/1ITfKQs-SYkKZF_eHf9lqcXbSkKSZT4bh/view?usp=sharing

4. All Buoys:
https://drive.google.com/file/d/1tUqjM42VuV4SkqgpoNXoalpR5i0rMLgB/view?usp=sharing

13 References

1. https://towardsdatascience.com/an-intuitive-guide-to-expected-maximation-em-algorithm-e1eb93648ce9
2. https://medium.com/@siddharthvadgama/gaussian-mixture-model-gmm-using-em-algorithm-from-scratch-6b7c764aac9f

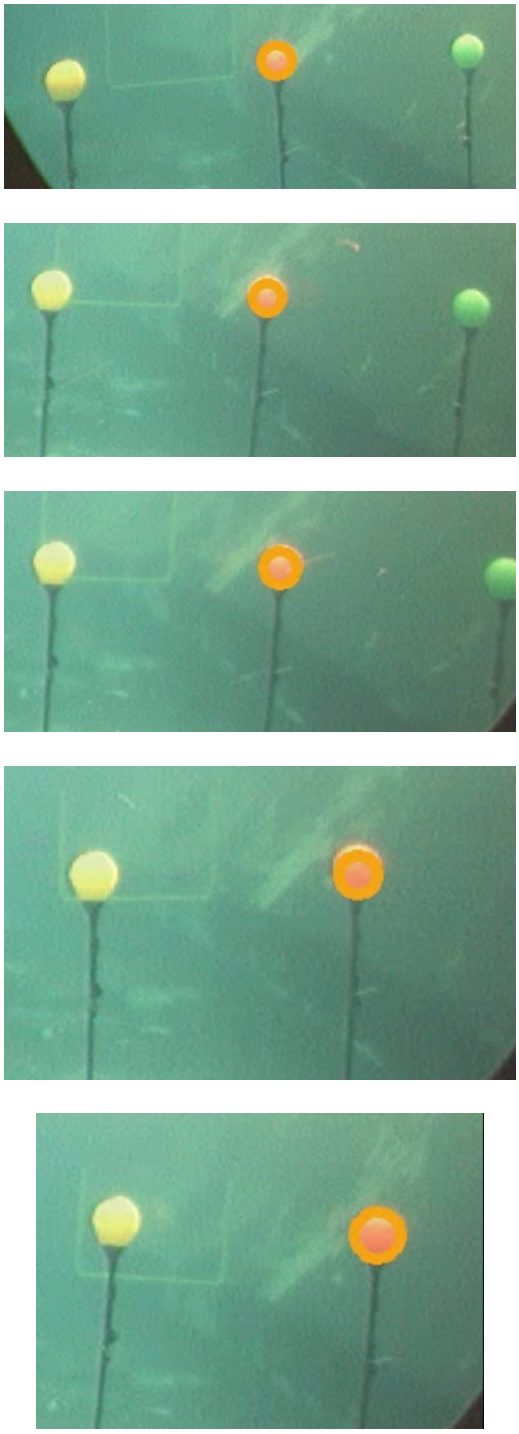

Figure 12: Segmenting Orange Buoy using 3D Gaussian

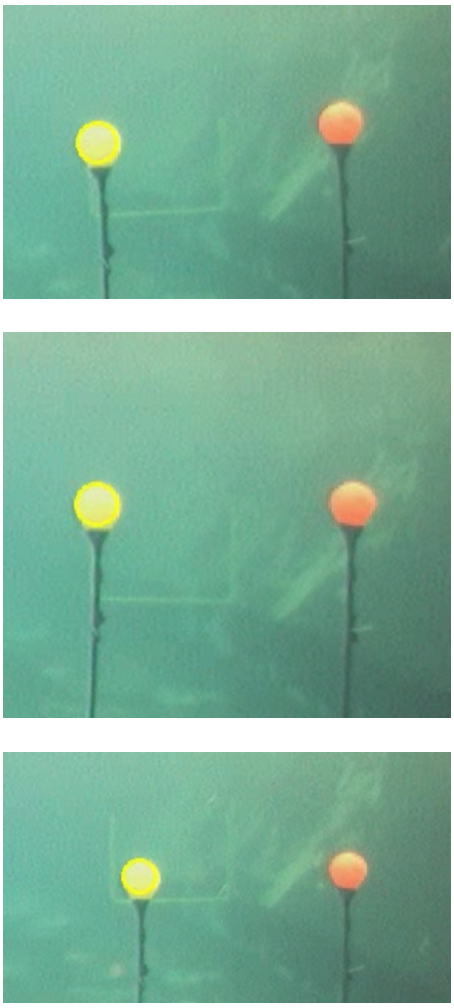

Figure 13: Segmenting Yellow Buoy using 3D Gaussian