ENPM673 - Perception for Autonomous Robots

Project 3

Underwater Buoy Detection

Authors:
Trevian Jenkins (116781381)
Karan Sutradhar (117037272)
Markose Jacob (117000269)

Date: April 5th, 2020

Objective:

Our objective is to develop a detection algorithm that detects the segmented colored buoys underwater. We aim to generate a tight segmentation of each buoy for the entire given video sequence by applying a tight contour around each buoy with respect to color.

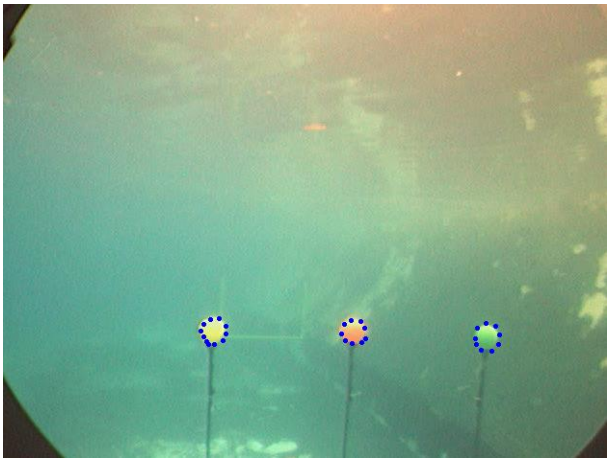
Introduction:

In this project, a Gaussian Mixture Model (Gmm) is used to probabilistically learn and detect colors of buoys. In the K-means algorithm, each data point is first given a hard assignment to a cluster, out of k-number of clusters, in what is known as the "E-step". Next, using the new data point assignment, the parameters for the new clusters are estimated in what is known as the "M-step". The implementation of Gmm is very similar to that of k-means, except that it uses a soft clustering assignment where each point is given a likelihood of belonging to a certain cluster. The result is that each point is effectively partially in every cluster, and these likelihoods are used as weights for the next E-step.

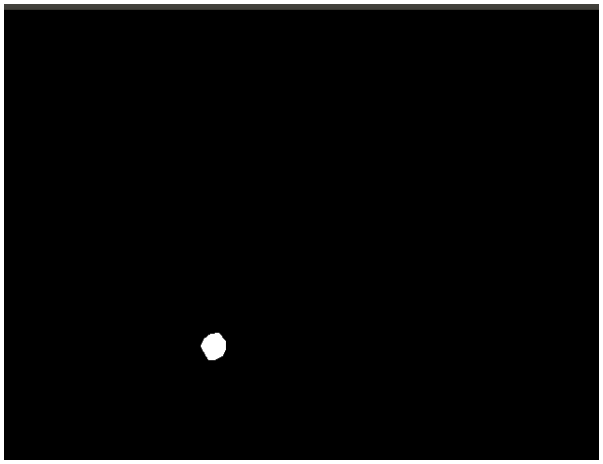
Preparing Data:

We first converted the video into frames using openCV and took 70% of the frames as training data. This program can be found in convertToFrames.py. Please refer to the readme for running instructions.

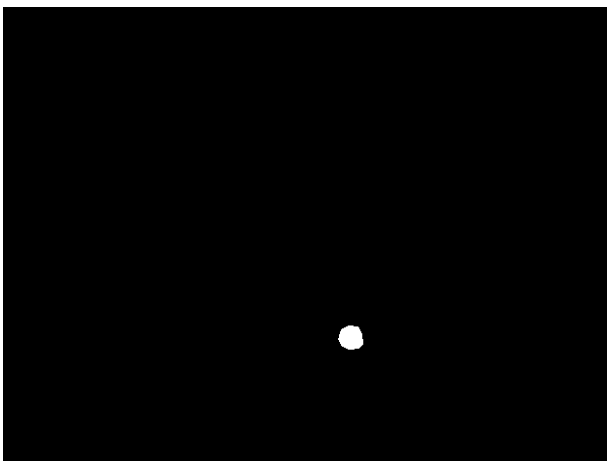
With the help of openCV's mouse setMouseCallBack the user is able to draw a mask around each buoy frame by frame and save these masks which are later used to extract pixel values for yellow, orange, and green. FillPoly was used to fit a polygon using the points clicked by the user. This program can be found in getTrainingMaskedFrames.py. Please refer to the readme file for running instructions.



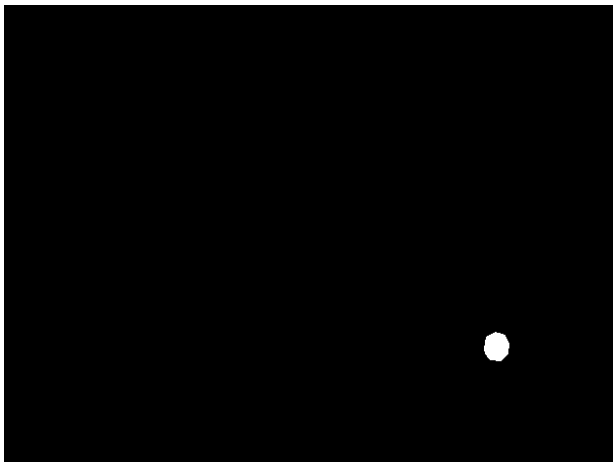
Pixel value extraction process



Yellow buoy mask



Orange buoy mask



Green buoy mask

Theory:

We utilized a 2D Gmm to estimate the appropriate group for the color of each buoy. The two attributes used are the red and green color channels since these two are the only ones necessary to distinguish the different buoys. The data points are gathered from the average value of the green and red color channels for each buoy in each frame of the training data. To implement the Gmm, the parameters of the N clusters (where N simply denotes the number of clusters) are first semi-randomly assigned by evenly spreading them across the data set using the mean and standard deviation of the entire set. The means are initially an equal distance from the mean of the entire data set, and their distance is determined by the standard deviation of the entire set, using the following equation:

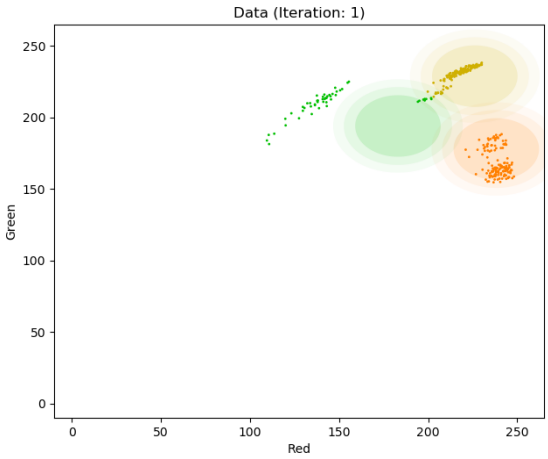
$$\mu_{xk} = \mu_x + \sigma_x cos(\theta + 2\pi \frac{k}{N})$$

$$\mu_{yk} = \mu_y + \sigma_y cos(\theta + 2\pi \frac{k}{N})$$

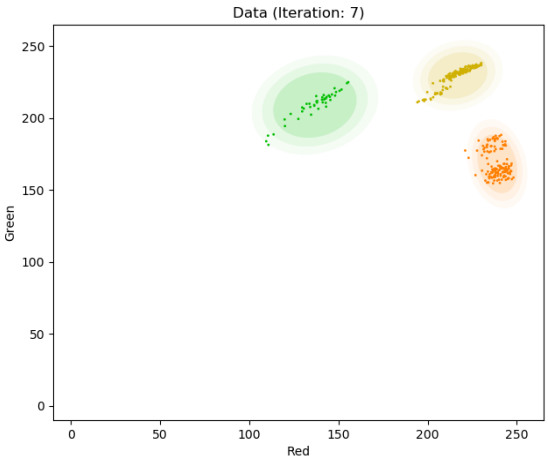
$$\sigma_{xk} = \frac{\sigma_x}{N}$$

$$\sigma_{yk} = \frac{\sigma_y}{N}$$

In the above equation, k denotes the k-th cluster out of N clusters, (μ_{xk}, μ_{yk}) denotes the cluster mean, (σ_{xk}, σ_{yk}) denotes the cluster standard deviation, (μ_x, μ_y) denotes the set mean, and (σ_x, σ_y) denotes the set standard deviation. Theta is randomly initialized, and covariance is set to zero. This initial parameter assignment distribution is shown below.



Initial parameter assignment



Final Gaussian mixtures (showing 90%, 97.5%, and 99.5% confidence levels)

Then, we make use of the expectation-maximization technique as described previously. The probability that a point lies within a certain cluster is the following:

$$P(x_i \mid c) = \frac{1}{\sqrt{2\pi|\Sigma|}^{\gamma}} e^{-\left[\left(\frac{x-\mu_x}{2\sigma_x}\right)^2+\left(\frac{y-\mu_y}{2\sigma_y}\right)^2\right]}$$

In the above calculation, γ is a weight used to prioritize the distance from a point to a cluster over the number of points in the cluster, and Σ is the covariance. Using the above calculation, the posterior probability is the following equation:

$$P(c \mid x_i) = \frac{P(x_i \mid c)P(c)}{\sum_{c'=1}^{N} P(x_i \mid c')P(c')}$$

Where A is the number of data points and the priors, or total contribution of the data set to each cluster, is the following:

$$P(c) = \frac{1}{A}\sum_{i=1}^{A} P(x_i \mid c)$$

The posterior probability is used as the weight for soft clustering. After these weights have been determined, the new cluster parameters are calculated as described below. The contribution of each point to the new cluster is the following:

$$a_i = \frac{P(x_i \mid c)}{nP(c)}$$

The mean, standard deviation, and covariance for each cluster c and attributes j and k are the following:

$$\mu_{cj} = \sum_{i=1}^{A} a_i x_{ij}$$

$$\sigma_{cj} = \sqrt{\sum_{i=1}^{A} a_i (x_{ij} - \mu_{cj})}$$

$$\Sigma_{cjk} = \sum_{i=1}^{A} a_i (x_{ij} - \mu_{cj})(x_{ik} - \mu_{ck})$$

After finding these parameters, the probabilities and weights for each point are recalculated as before, and the Gaussian parameters are recalculated, and this process iterates until convergence, when the change in the location of the means is negligible.

Gaussian Mixture Model and Buoy Identification:

The result of the Gmm is used to categorize each buoy. In the execution test, the buoys are segmented using Canny edge detection and Hough circles to find the edges of the buoy. After finding round contours possibly corresponding to the shape of a buoy (excluding false positives such as the glare in the background), the contour is used to generate a mask for the frame, where average color under the mask is calculated. If this average color falls within a 99.5% confidence level of any of the clusters, the shape is indeed determined to be a buoy and its color is categorized as belonging to that cluster. We drew the outline of the contour based on the mean values of the red and green channels of the cluster as determined by the Gmm.

Result:

The video output of our program can be found at the link below:
https://drive.google.com/open?id=1g7n3CDNtIvtxrhJRqnG_eTM6I_I1znce

Screenshot of the video output

Note:

To run the code you need to follow the instructions provided in the README file.