ENPM 673, Robotics Perception
Project 2: Where's my BUOY?! There it is...
Due on: 11:59:59PM on Monday – Mar 06, 2017

Prof. Yiannis Aloimonos

1 Color Segmentation

In this project,we will be estimating a Gaussian model for Buoy and developing a Buoy detection function based on the color model.

A buoy is a distinctively shaped and colored floating device, anchored to the bottom, for designating moorings, navigable channels, or obstructions in a water body.

As mentioned, it is sometimes used as underwater markings for navigation. The dataset for the project is a camera view of an underwater vehicle with three distinctly colored buoyed in surrounding. The goal is to segment and detect the buoys in the video sequence.

The figure 1 depicts an example of an input image, a probability map based on the color model, a segmented binary image after filtering, and contours for a red buoy.

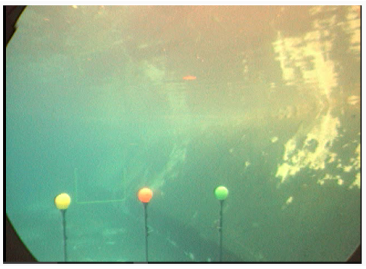
Since, the buoys are distinctly colored and shaped, one approach to segment them is to use color segmentation. But the noise and changing light intensities (due to properties of environment) make it difficult to segment the buoys, using vanilla color threshold based techniques. Hence, we stress on using Gaussian Models to learn the color distribution, and use the model to segment and detect the buoys.

The aim of the project is to get a tight contour over the segmented buoys and also the color of the contour should match the detected buoy. That is, a red buoy, should have a red contour etc.

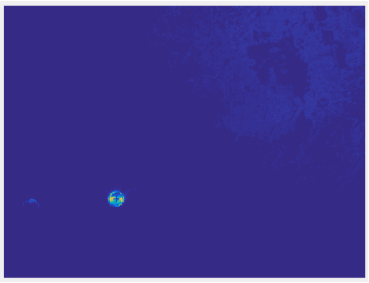
2 Instructions

2.1 Data preparation (10 Pts)

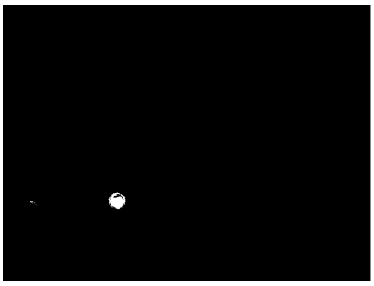
In order to fit a color model of a buoy, we need to have sample data. We will divide the video sequence (dataset) into training and testing frames - where the training frames are



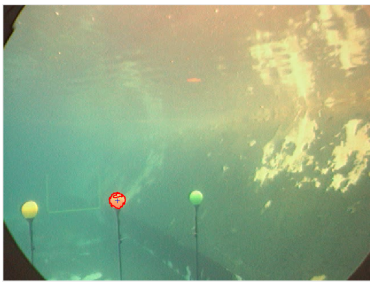
(a) Under Water Buoys



(b) probability map



(c) thresholded binary image



(d) Output

Figure 1: Buoy Detection

used to generate the sample data and fit a color model and the testing frames will be used to evaluate its performance.

1. Extract and Save in memory, the samples of Buoy images from the video sequence provided in the dataset. Extraction of the samples can be done by using roipoly function in MATLAB.

   (Again, It is advised to divide the video sequence into appropriate number of training frames and testing frames, such that the training frames are used to extract model parameters and testing frames are used to evaluate its performance. You DONOT want to over-fit your model by training on all frames).

2. For each colored buoy, gather and visualize color distribution for each channel of the sampled images (RGB image). You might use average color histogram (averaged over each channel) for visualization, but remember to fit the model on all samples of the color distribution. You may also experiment by using other color spaces (*HSV, Lab etc.*) for modeling color distribution of buoy. For this you can use functions such as rgb2hsv, rgb2lab, etc. Fig.2 shows the color distribution of a red buoy.

   Observe the color distributions - eventually we will be modeling these distributions to tell whether a pixel belongs to a buoy or not!

Submission Details:

1. Save the Training Frames in `Images/TrainingSet/Frames/<FrameNo>.jpg` and `Images/TestSet/Frames/<FrameNo>.jpg`.

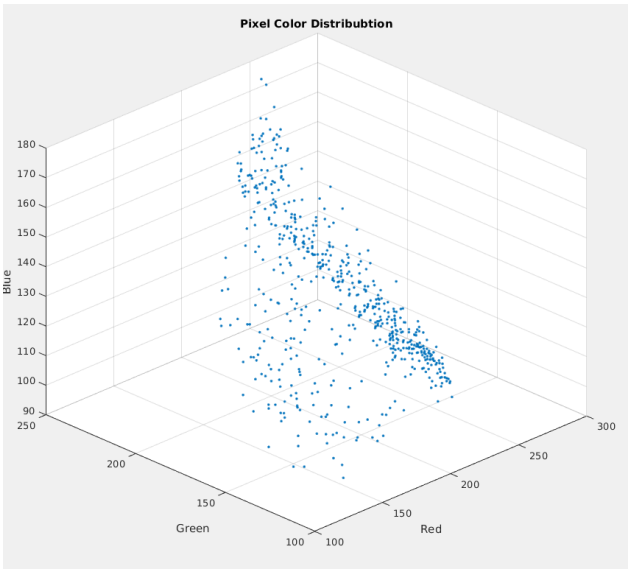


Figure 2: Color Distribution

2. Save the cropped Buoy images in `Images/TrainingSet/CroppedBuoys/<Color_FrameNo>.jpg`. e.g., (`Images/TrainingSet/CroppedBuoys/R_001.jpg`).

3. Submit the script `Scripts/Part0/colorDistribution.m` which reads the images from `Images/TrainingSet/CroppedBuoys/*` and saves the plot in `Output/Part0/<Color>_hist.jpg` (e.g., `Output/R_hist.jpg` for red). The script should **return** an array of samples - color distribution **ColorSamples**.

2.2 Color Model Learning - using single 1-D Gaussian (10 Pts)

1. Write a function to estimate your model parameters, that is, the mean and variance, from the sample data. You will use a single 1-D gaussian to model the color distribution (**Color Distribution is nothing but the *set of samples* and you should use all the pixels from cropped buoy as your samples**)

   Note: While fitting 1-D Gaussian – For each pixel, color sample indicates a *single value*. And depending on which buoy you are detecting, for each pixel you should use appropriate channel as the color sample value - say for red buoy you can directly use R channel value whereas for yellow buoy, you might want to use R+G as the color sample.

2. Save the parameters for later use during buoy detection.

Submission Details

1. Submit the script `Scripts/Part0/estimate.m` which takes the sampled data (Color Distribution - **ColorSamples**) as **input** and **returns** the mean and variance (**ModelParams**) of the model that fits it.

2.3 Buoy Detection - using single 1-D Gaussian (15 Pts)

1. Implement a Buoy detection function which will take a frame id (from the video sequence) and the computed model parameters (from previous part, Section 2.2) as input and generate probability map i.e., a 2D matrix of the same size as that of input image where the entries of probability map indicate the probability of the corresponding pixel being from buoy.

2. In order to segment out the buoy from an input image, you will need to threshold the probability map to generate a binary image.

3. In order to correctly locate the buoy, you can consider applying some filtering on the binary image and get the contours of the *buoy blob*. Useful MATLAB Image Processing Toolbox functions include bwconncomp , regionprops

4. Draw contours on all the buoys detected in the binary image. Use the color of buoy as the color of the contour. **Remember you have to repeat this for each buoy and the final output image should have all the buoys detected (i.e., corresponding colored contours around each buoy)**.

5. Finally write a function to generate a video sequence by reading images from `Output/Part0/Frames/` and save it in `Output/Part0/Video/`..

Submission Details

1. Submit the script `Scripts/Part0/detectBuoy.m` which takes the input frame (**FrameID**) and model parameters (**ModelParams**) as **inputs** and **outputs images with colored contours around each buoy**.

2. Save the binary images (**it should have white blobs for each of the buoys**) in `Output/Part0/Frames/binary_<FrameNo>.jpg`.

3. Save the segmented images (**input image with each buoy highlighted by corresponding color contours**) in `Output/Part0/Frames/output_<FrameNo>.jpg`.

4. Save the plot of single 1-D Gaussian Model (for each of the buoy) in `Output/Part0/<Color>_1gaussD.jpg` (Color code - R (Red), G (Green) and Y (Yellow) buoy).

2.4 Color Model Learning - using single 3-D Gaussian (5 Pts)

Once you model 1-D Gaussian extending to 3-D Gaussian is relatively easy.

1. Write a function to estimate your model parameters, that is, the mean and (co)variance, from the sample data. You will use a single 3-D gaussian to model the color distribution (needless to say 3-D gaussian uses (R,G,B) sample points, say (255,0,0) for pure red pixel).

2. Save the parameters for later use during buoy detection.

Submission Details

1. Submit the script `Scripts/Part1/estimate.m` which takes the sampled data (Color Distribution - **ColorSamples**) as **input** and **returns** the mean and (co) variance (**ModelParams**) of the model that fits it.

2.5 Buoy Detection - using single 3-D Gaussian (10 Pts)

1. Implement a Buoy detection function which will take a frame id (from the video sequence) and the computed model parameters (from previous part, Section 2.4) as input and generate probability map i.e., a 2D matrix of the same size as that of input image where the entries of probability map indicate the probability of the corresponding pixel being from buoy.

2. In order to segment out the buoy from an input image, you will need to threshold the probability map to generate a binary image.

3. In order to correctly locate the buoy, you can consider applying some filtering on the binary image and get the contours of the *buoy blob*. Useful MATLAB Image Processing Toolbox functions include bwconncomp , regionprops

4. Draw contours on all the buoys detected in the binary image. Use the color of buoy as the color of the contour. **Remember you have to repeat this for each buoy and the final output image should have all the buoys detected (i.e., corresponding colored contours around each buoy)**.

5. Finally write a function to generate a video sequence by reading images from `Output/Part0/Frames/` and save it in `Output/Part0/Video/`..

Submission Details

1. Submit the script `Scripts/Part1/detectBuoy.m` which takes the input frame (**FrameID**) and model parameters (**ModelParams**) as **inputs** and **outputs images with colored contours around each buoy**.

2. Save the binary images (**it should have white blobs for each of the buoys**) in `Output/Part1/Frames/binary_<FrameNo>.jpg`.

3. Save the segmented images (**input image with each buoy highlighted by corresponding color contours**) in `Output/Part1/Frames/output_<FrameNo>.jpg`.

4. Save the plot of single 3-D Gaussian Model (for each of the buoy) in `Output/Part1/<Color>_1gaussD.jpg` (Color code - R (Red), G (Green) and Y (Yellow) buoy).

3 Gaussian Mixture Models (GMM) and Maximum Likelihood Algorithm

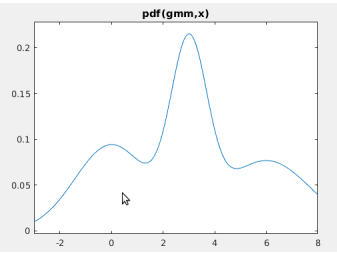
A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.

The main difficulty in learning Gaussian mixture models from unlabeled data is that one usually doesn't know which points came from which latent component (if one has access to this information it gets very easy to fit a separate Gaussian distribution to each set of points).

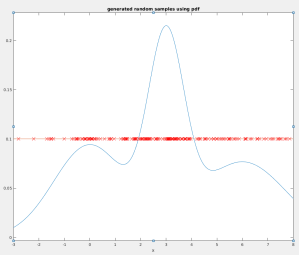
Expectation-maximization is a well-founded statistical algorithm to get around this problem by an iterative process. First one assumes random components (randomly centered on data points or cluster centroids learned from k-means or even just normally distributed around the origin) and computes for each point the probability of being generated by each component of the model. Then, one tweaks the parameters (which include weights of each gaussian, mean & co-variance parameters for each gaussian) to maximize the likelihood of the data given those assignments. Repeating this process is guaranteed to always converge to a local optimum.

3.1 Using Expectation Maximization to extract GMM parameters (20 Pts)

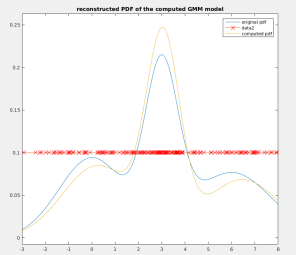
1. Generate data samples from 3 1-D Gaussians, with different means and variances. (You can generate 50 samples from each distribution using mvnrnd function)



(a) Probability Density Function



(b) Randomly generated samples



(c) pdf of extracted GMM parameters

Figure 3: Fitting GMM for a Probability Density Function using Expectation Maximization

E.g.: The Fig.3 plot shows combined *pdf* of 3 Gaussian distributions with means (μ) at 0,3 and 6 and covariance (σ) of 2, 0.5 and 3 respectively.

2. Try to recover the model parameters used in previous part, using the samples generated. Use Expectation Maximization (EM) algorithm - Refer CM Bishop's Lecture Slides for

the details of the algorithm. YOU CAN USE built-in MATLAB functions to fit a GMM model. Check Matlab gmdistribution class for relevant resources and the details of the algorithm.

3. Try to recover the model parameters for 4 Gaussians (means and variances of 4 Gaussians that fit the data generated by 3 in step 1. Discuss what you observe.

4. Confirm your implementation works for generalized N Gaussians each of dimension D. **Note that the above example is given to explain a simple mixture of 1-D gaussians, but the idea can be extented to multi-dimensional gaussians**

Submission Details

1. Submit the script `Scripts/Part2/EM.m` which takes number of gaussians **N**, **data** and **plot_path** as **inputs** and **outputs** the the array $\boldsymbol{N} \times \boldsymbol{D}$ mean and $\boldsymbol{N} \times \boldsymbol{D} \times \boldsymbol{D}$ covariance model parameters. The function should plot the computed Gaussians and save it to the relative path mentioned in plot_path (if plot_path = "<some_output_directory>", plot and save operation should be avoided).

2. Save the plot of the model for the generated data with N = 3, in `Output/Part2/EM1D3N.jpg` (output of step 2)

3. Save the plot of the model for the generated data in step 1 with N = 3, in `Output/Part2/EM1D4N.jpg` (output of step 3)

3.2 Color Model Learning - using GMM (20 Pts)

Now, we use Expectation Maximization to model Color Distribution of Buoy with the eventual goal of color segmentation.

1. For each colored buoy, compute and visualize the average color histogram for each channel of the sampled images (RGB image). This should provide some intuition to decide number of Gaussians [N] to fit the color histogram. Also, can you determine the dimension of each Gaussian [D] for your model? **You will use a 3-D Gaussian if you are using RGB color space**

2. Use the previous implementation to compute the model parameters (Mean and Covariance parameters of N D-dimension Gaussian) of the color distribution for each buoy.

Submission Details:

1. Save the plot of the model in `Output/Part2/EM_<Color>.jpg` for each colored buoy (Color code - R (Red), G (Green) and Y (Yellow) buoy).

3.3 Buoy Detection - Using Gaussian Mixtures (10 Pts)

- Implement a Buoy Detection function which will take a frame (from the video sequence) and the computed model parameters (from Section 3.1) as input and generate probability map.
- Segment the buoy by thresholding the probability map and generate a binary image for each buoy. Compute the corresponding contours and center of each *buoy blob*.
- Draw the bounding contours around the detected buoy (color of the contour should be the color of buoy detected). **Remember you have to repeat this for each buoy and the final output image should have all the buoys detected (i.e., corresponding colored contours around the buoy)**.
- The assignment will be graded on how tight the bounding contours cover the buoys.

Submission Details:

The Buoy Detection function should be : Scripts/Part3/detectBuoy.m (**inputs - FrameID, ModelParams**). The function should

1. Display and Save the binary image `Output/Part3/Frames/binary_<FrameID>.jpg`.
2. Display and Save the image with contours in `Output/Part3/Frames/out_<FrameID>.jpg`.

Finally write a function to generate a video sequence by reading images from `Output/Part3/Frames/` and save it in `Output/Part3/Video/`.

4 Extra Credit

1. The Color Segmentation scheme used the RGB representation of the images. Are there any other representations of images that would be more effective to the algorithm (Yes/No, defend your reasoning)? – **10 Pts**
2. Can you come up with your own representation that encompasses the color-distribution information which can improve the performance of the algorithm? – **35 Pts**
3. Implement the algorithm using any other color spaces and compare the results. – **35 Pts**

Submission Details:
All the plots and functions for Extra Credit part should be saved to /ExtraCredit/ folder and details are to be mentioned in the report.

5 Submission Guidelines

A detailed report (pdf) of the steps taken, methods used, important plots with analysis, difficulties faced and reasoning of the solution should be submitted in Reports/ folder.

Keep all the helper functions in ../helper/ as subpart of any folder you want.

The function names and definitions are just an outline, feel free to change them as desired but keep them meaningful. However, do not change the folder structure.