Color Segmentation using Gaussian Mixture Models and Identification of a Specific Object and its Position

Daniel Pak

***Keywords*— Gaussian Mixture Models (GMM), color segmentation, object identification, distance estimation**

I. INTRODUCTION

Color segmentation is an important low-level processing step for computer vision systems. Gaussian Mixture Model (GMM) is one method of doing this, by predicting the distribution of pixels using a linear combination of multiple Gaussian distributions. There are many different ways to utilize the GMM to solve this problem, but the approach in this project was to use a separate GMM for each color of interest in order to compensate for the various lighting conditions of the images. When done with color segmentation, some systems must also recognize objects. I will be using image processing techniques primarily using OpenCV and skimage libraries to post-process segmented images and identify the barrel using some known properties of the object. Then, I will estimate the distance using a regression method.

II. PROBLEM FORMULATION

The main problem is to identify one or multiple identical red barrels in various images. The variables for this particular project included lighting conditions, orientation of the barrel(s), distance of the barrel(s) from the camera, and obstructions covering the barrel(s).

A. Color Segmentation

The first main sub-problem is to accurately classify pixels that are the exact shade of the barrel red. Due to the inconsistent lighting conditions, the red color of the barrel was unpredictable and could not be represented with a single general distribution. Also, many images included red-like objects such as a red cone, vending machine, floor, buildings, etc. Since all later-stage operations use the results of color segmentation, it was crucial that this step was robust.

B. Identification of the Barrel and its Position

In an ideal world, the color segmentation step would identify only and all of the pixels of the red barrel, and the user would simply identify the pixels' locations to identify the barrel. However, the reality was that some pixels in other objects were classified as barrel red, and some parts of the barrel were not. This made it necessary to include a series of post-processing steps to accurately identify the barrel. Also, a challenging component was that the barrel was very far from the camera in certain instances, and was considered a set of noise pixels. This made it difficult to come up with a general algorithm to differentiate the barrel from the unwanted pixels.

III. TECHNICAL APPROACH

A. Acquiring pixels

In order to train the color segmentation model, I needed pixels for specific objects. In this project, I simply hand-chose those pixels using a library called "roipoly". Due to time constraints, I was only able to label barrel red and non-barrel-red (i.e. red objects that are not the barrel). However, since those are the most important color classes of interest, it did not significantly affect the performance of color segmentation.

B. Gaussiasn Mixture Model (GMM) Overview

GMM was used primarily to account for the varying lighting conditions. Before training anything, I converted the color space from RGB to HSV based on visualization of pixel distribution. For each class, I chose the number of clusters to be 3 based on visualization of cluster means, as well as the time it takes to train the models. The covariance structure of the clusters was also fixed to be a diagonal matrix, mainly to speed up the training process and to account for the fact that the data might not be large enough to calculate the full covariance matrix. A more robust testing would have been to adjust those parameters (number of clusters and covariance structure) based on cross-validation error.

I made two separate GMMs for the barrel red and the non-barrel-red classes. The reasoning is that those two classes have the most potential to overlap in pixel distributions, and the objective is to simply identify the barrel, not perfectly identify every color class in the images. After training, GMMs are able to provide values for $P(x|k) =$ probability of pixel given cluster, and ultimately $P(x|w) =$ probability of pixel given color (linear combination of probability of pixel given cluster). With this value, the Bayes's rule is used to calculate $P(w|x) =$ probability of color given pixel. I assumed uniform distribution of colors, which meant $P(w|x) \propto P(x|w)$. Based on these values, the membership of each pixel to each color class was compared, which then turned images into binary masks that indicate the positions of pixels with highest membership probability to barrel red class. From this point on, these masks are assumed to be the barrel red pixels, and are used for further analysis.

C. GMM Specifics

For learning purposes, everything for the GMM was coded using linear algebra libraries and the EM algorithm.

The objective function, which we are trying to maximize for each pixel, was:

$$\lambda(X, \theta) := \sum_{x \in D_a} \log \sum_{k=1}^{K} a_k \phi(x; \mu_k, \sigma_k)$$

where x = pixel, D_a = set containing all training pixels, k = the cluster index, a_k = the weighting for each Gaussian cluster, ϕ = Gaussian distribution, μ_k = mean of Gaussian, and σ_k = covariance of Gaussian.

Since the partial derivative of the objective function is not solvable analytically, the EM algorithm is used to approximate the local minimum. The EM algorithm works in two-fold: calculating the lower bound of objective function, and choosing parameters of the lower bound to maximize it. Jensen's inequality is used to derive an expression for the first part, and KL divergence and differential entropy are used for the second.

Based on those expressions, it just so happens that a lower bound for the objective function with a given set of parameters is the membership probability of each point to each Gaussian in the model. For multiple Gaussians, the membership probability is calculated as follows:

$$r^{(i)}(k|x) = \frac{a_k^{(i)} \phi(x; \mu_k^{(i)}, \sigma_k^{(i)})}{\sum_{m=1}^{K} a_m^{(i)} \phi(x; \mu_m^{(i)}, \sigma_m^{(i)})}$$

The initial parameters of each Gaussian was estimated using an established k-means algorithm from the scikit toolbox. When given the number of clusters, a function calculates the mean of clusters and returns points that belong to each cluster, which can then be used to calculate the covariance using a function from the numpy library. The weighting coefficients are initialized to equal values that sum to 1.

Then, to calculate the parameters that maximize the lower bound, the following equations were used:

$$\mu_k^{(i+1)} = \frac{\sum_x r^{(i)}(k \mid x) x}{r^{(i)}(k \mid x)}$$

$$\sigma_k^{(i+1)} = \frac{\sum_x r^{(i)}(k \mid x) diag(x - \mu_k^{(i+1)}) diag(x - \mu_k^{(i+1)})}{\sum_x r^{(i)}(k \mid x)}$$

$$a_k^{(i+1)} = \frac{1}{|D_a|} \sum_x r^{(i)}(k \mid x)$$

Every operation was pretty self-explanatory using matrix multiplication operations with the numpy library. The only additional tweak was that the membership probabilities for both training and testing were calculated without using the PDF evaluator. Using the expression for a Gaussian distribution, the membership probability to each cluster was evaluated by the following equation:

$$p(I|k) \propto \exp\big[-\frac{1}{2} * g\big]$$

where

$$g = \big[(I - \mu_k) * L\big] \odot \big[(I - \mu_k) * L\big]$$

and I = (width*height) x 3 matrix of image pixels, μ_k = mean of cluster, and L = lower left triangular matrix of Cholesky decomposition done on inverse of covariance matrix of cluster.

D. Post-processing of Binary Masks

Going back to the broader picture, all of the math was used to generate a good GMM for each color class for identifying the true barrel red color. GMMs turn colored images into binary masks as mentioned, which can initially look splotchy and noisy. In order to get a more desirable set of masks, high-level functions from OpenCV were used for post-processing. The main functions were morphological transformations such as dilations, erosions, and specific combinations of those such as opening and closing. These operations either remove small noise pixels or combine them to a larger piece. To account for the cases where the barrel was completely split due to obstructions, I used a higher number of dilation iterations to simply combine the pixels together so make future identification steps easier. The drawback was that I had to hand-tune the order of these operations and the parameters (i.e. iteration number) to get the desired effects.

E. Bounding Box

For obtaining all bounding boxes of various pixel clusters in the post-processed binary masks, I used the function minAreaRect from the OpenCV library. This was the only method I could find that allowed bounding boxes of different orientations. The output of this function is an array with the coordinates of the four corners of the bounding box. In order to identify the correct bounding box coordinates for just the barrel, there are some hard constraints that I had to set. By trial and error, I found a sensible range of aspect ratio ($\frac{height}{width}$ of the bounding box) and extent ($\frac{object\ \ area}{bounding\ \ box\ \ area}$). That range was $0.5 \leq$ aspect ratio ≤ 2 (taking into that the barrel can be of different orientations) and extent ≥ 0.65. I used these ranges as the hard cut-off, meaning that I completely ignored any bounding boxes not meeting these criteria. With more time permitted, I would have made a scoring system where each parameter would give a certain percentage of confidence that the chosen bounding box is the barrel, and made a threshold based on that more encompassing value. I also would have investigated more bounding box parameters.

F. Distance of Barrel to Camera

The distance was calculated using a regression method. The area of the bounding box was calculated and recorded for each training image (making sure that the bounding box was the correct one). The true distance was given with each training image. I plotted the log(area) vs. log(distance), and obtained a linear equation. Future distances were calculated using this equation. This relationship was discovered by trial and error, by trying different forms of regression and using the one with the highest R^2 value.

IV. RESULTS

Using the methods described in the Technical Approach section, three of the four plots in Fig. 1 were generated. First is a binary mask that is the result of applying GMMs to an HSV converted image. Second is a binary mask post-processed with OpenCV functions. Third is the original image overlayed with the bounding box. The fourth image is one that was not discussed in this report - a result of applying Sobel edge detection algorithm to the original RGB image. Due to limited space, only an example for one testing image was shown here. All other images are included in the appendix, along with the bounding box coordinates and the estimated distance of the barrel.

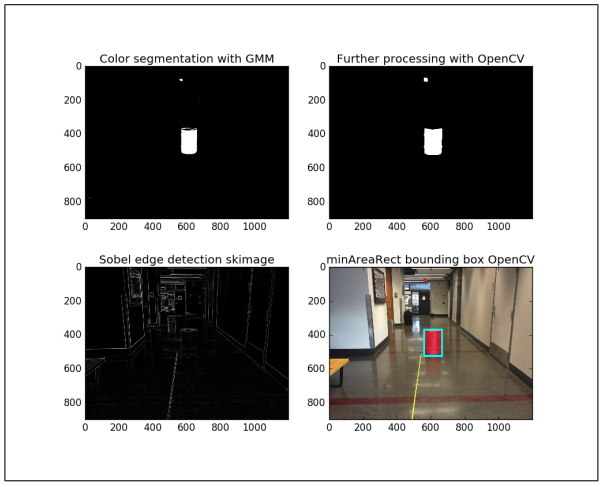

Fig. 1. Sample of resulting images. Results for all test images are included in the appendix.

For the testing set, the success rate of identifying the perfect bounding box for the barrel was 60% (6 out of 10). In the cases of failure, one was due to not being able to recognize the red barrel (lighting condition not included in the training set), one was due to having a red object with the exact same shade of red right beside the barrel, one was due to reflections, and one was due to obstructions. These will further be analyzed in the discussion section. On the other hand, the training set had a success rate of 88% (44 out of 50).

For training the GMMs, I initialized the parameters using the k-means algorithm in the scikit toolbox. Fig. 2 shows the initial values of the mean and the pixel distribution of the training pixels for the red barrel. The training pixels are the ones that were hand-chosen using the "roipoly" file. Fig. 3 shows the change in the location of the means after completing the EM algorithm for training the GMM. Finally, Fig. 4 shows a sampling of points from the means and covariances of the final GMM. The number of sampled points were determined as a ratio of the weighting coefficients. The figures provide evidence that the training was successful, and that the GMMs reflected the true distribution of the training pixels.

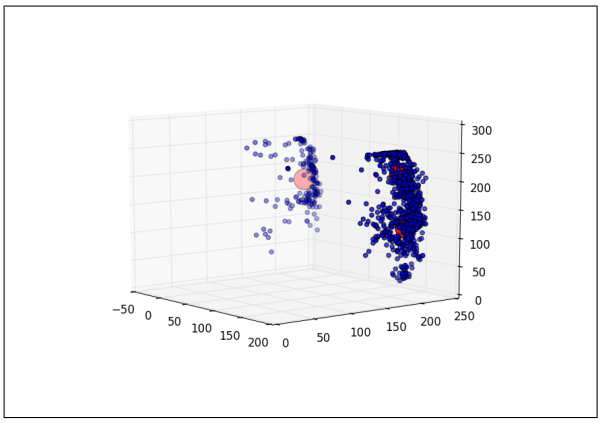

Fig. 2. Blue: randomly chosen pixels from the training set. Red: Initial means derived from k-means algorithm

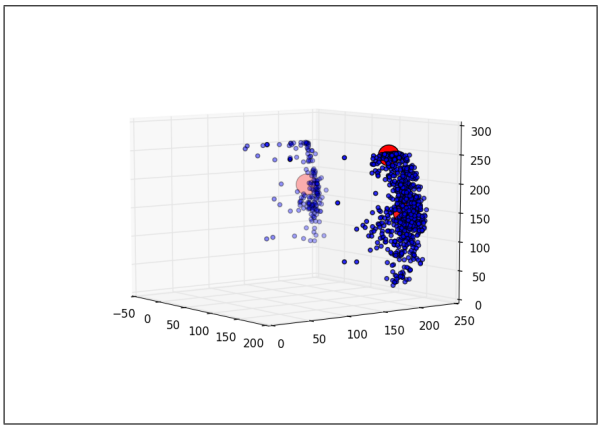

Fig. 3. Blue: randomly chosen pixels from the training set. Red: Final means after applying the EM algorithm for training GMM

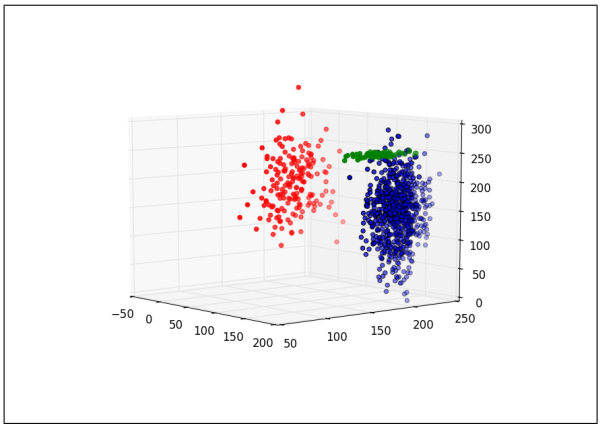

Fig. 4. Sampling of points using the means and covariances of the 3 Gaussians in the GMM. Number of sampled points is proportional to the weighting coefficient

Finally, the distance of the barrel from the camera was calculated using the regression graph shown in Fig. 5. As mentioned before, the log-log plot of the area of the barrel and the distance showed the best linear relationship, and the accuracy was around 80% for the correctly identified bounding boxes. Since the real distance was also given in

whole meter resolution (i.e. 1m, 2m instead of 1.1, 1.95m, etc.), the accuracy of the equation was pretty satisfactory.

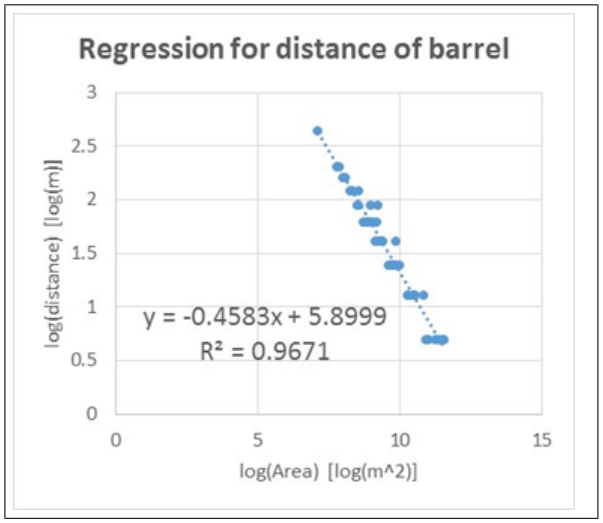

Fig. 5. Sample of resulting images. Results for all test images are included in the appendix.

V. DISCUSSION

There are several things that could have been improved. First is the number of Gaussian clusters in the not-barrel-red model. Since this training set included a whole array of different objects in various lighting conditions, the performance of color segmentation probably could have been improved if the number of clusters was higher. However, the roadblock for this was that the training procedure consumed a lot of time (1hr each run), which made repeated trials unfeasible. The reason for such a long training procedure was that I was unable to find an efficient way to make the diagonal matrix with each row of an array, so I had to loop through every pixel and call the diag function in the numpy library. Without this step, I approximate that this process would have taken 15 minutes. Also, a big fault in choosing to divide the training sets into barrel red vs. not-barrel-red was that the number of pixels for the not-barrel-red set depended very heavily on how many pixels I could safely choose by choosing the desired regions in the training image. For instance, the red floor and red buildings were much easier to choose than the red of a vending machine, small ball, or bike frame, which undoubtedly affected the distribution of pixels in the training set, regardless of how important each object was.

A good way to resolve this issue could have been to make separate GMMs for each object that may conflict with the barrel's color. Although I am unable to do this now, I would like to test this to see the algorithm's change in performance. However, this method could also adversely affect the performance due to overfitting or the fact that it does not have a weighting coefficient that may affect the contribution of a membership probability to the overall likelihood.

On another note, the edge detection algorithm could be the saving grace for some of these problems. As mentioned in the Results section, I had some difficulties in the cases where a red object with the exact same shade of red as the barrel itself is right behind the barrel. Unless the GMMs were fit to the point where it could distinguish these points separately (which I suspect requires a little bit of overfitting), there is no where for me to identify the barrel because the binary mask has the two objects represented as one. No amount of post-processing would help this issue. However, if I run the edge detection algorithm first on the original image and then segment the image, I can consider the edge as a boundary and maybe even distinguish the separate objects. The edge detection in the Appendix section for the third image in the testing set shows a glimpse of possibility for this method (must zoom in a lot to see the edge, since it is a very small change in pixel values).

VI. CONCLUSION

This report was on the topic of color segmentation and identification of a red barrel in real world images. GMM is a good method for doing the segmentation, but there were definitely rooms for improvement at various points of the algorithm. With more research on post-processing, some unsolved mysteries such as identifying a barrel completely split with large gaps at multiple points (i.e. 8th figure in the testing set) may be resolved.

VII. REFERENCES

All information was acquired from Unviersity of Pennsylvania's ESE650 lectures, discussions on Piazza, and official websites of Python and the respective libraries.