David Hernandez
ESE 650 Project 1
Color Segmentation

## Introduction

In this project I implemented a segmentation algorithm that isolates red barrels in images and predicts their distance from the camera. Segmentation algorithms are useful for many applications such as identifying objects of interest in images and tracking them over multiple frames. There are many approaches to segmenting objects in an image, including decision trees, forms of regression and modern techniques such as deep learning with convolutional neural networks. In this project I implement a simple color classifier with basic image processing to identify barrels in an image.

## Approach

### Overview

The red barrel segmentation algorithm was based on three main parts. First, a gaussian mixture model was trained to classify red pixels in an image using a labeled dataset. Second, image processing was used to detect red barrels among the classified pixels and create a bounding box around them. Finally, linear least squares regression was run to predict the distance of each barrel from the camera based on features of the bounding box.

### EM Algorithm

A gaussian mixture model was used to probabilistically classify red pixels in new images. After training, the model reads in pixel values from a new image and calculates the probability that they are red based on a weighted linear combination of probabilities from multiple gaussian distributions. Because there is no closed-form solution for optimization of a multivariate gaussian model, the EM algorithm was used to to converge on the optimal parameters for each gaussian. This algorithm switches off between the E-step and the M-step until a local maximum of the overall log likelihood is reached. The E-step calculates the amount each pixel belongs to each member gaussian of the mixture using the current parameters. The M-step then finds the parameters that maximize the log likelihood given these membership probabilities. The data and log likelihood equations which were maximized by the EM algorithm as well as the multivariate gaussian equation used to calculate the probabilities of each data point are shown in figure 1a. In this implementation, the EM algorithm optimized the mean, full covariance and weight of each gaussian mixture using the equations shown in figure 1b. A cutoff value of the log likelihood was used to classify pixels as red or not red, generating binary masks of new images. This cutoff was determined through cross-validation of labeled training images.

**Figure 1**

**a)**

Data Likelihood

$$\Lambda(X,\theta) := \prod_{x \in D_\alpha} \sum_{k=1}^{K} a_k \phi(x; \mu_k, \Sigma_k)$$

Log Likelihood

$$\lambda(X,\theta) := \sum_{x \in D_\alpha} \log \sum_{k=1}^{K} a_k \phi\left(x; \mu_k, \Sigma_k\right)$$

Multivariate Gaussian Distribution

$$\phi\left(x; \mu_k, \Sigma_k\right) x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}}$$

**b)**

(E step)
$$r^{(i)}(k \mid x) = \frac{a_k^{(i)} \phi(x; \mu_k^{(i)}, \Sigma_k^{(i)})}{\sum_{m=1}^{K} a_m^{(i)} \phi(x; \mu_m^{(i)}, \Sigma_m^{(i)})}$$

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

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

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

There were a few image preprocessing steps that I used to train and improve the accuracy of the EM algorithm. The EM algorithm was trained using the original color images and binary masks which represented red pixels with a “1” and non-red pixels with a “0”. These binary masks were generated manually using the Matlab image segmenter application. They were then loaded into python using OpenCV. Color images and their binary images were matched by using the same name for corresponding images in separate folders. After being loaded into python, images were converted into the HSV color space which use Hue, Saturation and Value quantities to represent the color of pixels, rather than standard RGB. This color space uses value to represent the brightness of the color, saturation to represent the intensity of the color and hue to represent color as an angle on a color wheel. This change in color space made the red color class more easily separable in 3D space using a gaussian mixture.

In general, cross-validation was used to determine reasonable values for initialization of gaussian parameters, size and height/width cutoffs and other variables used when training the model. This verified that the parameters would reliably converge to local maxima of the log likelihood function. Overall, the effectiveness of the model was mostly determined visually. For example, I would look for situations where barrels were classified mostly correctly and which did not have too many false detections. It was a bad sign only if a few pixels in a barrel were detected or if a large portion of another object was detected. I also calculated the average squared error of predicted vs actual distance measurements to ensure that the linear least squares regression model was predicting reasonable distances. By using 4-fold cross-validation, I was ensuring that the model worked well on never before seen images of barrels and was not overfitting the training dataset.

Some examples of gaussian mixture model parameters I had to tune include the initialization of the mu, sigma and weight values. I found a reasonable initialization for each element and used slightly different numbers for each component’s initialization. I initialized weight values to simply be uniform with equal weighting for all the clusters. I initialized the covariance matrix, sigma, by setting only the diagonal elements with positive numbers to ensure that it was positive semi-definite. I set the mean, mu, to be close to the average rgb pixel value [122, 122, 122] and found that the initialization worked well even after switching to the HSV color space. In order to terminate the training of the gaussian mixture model I set a threshold “alpha” on the change of log likelihood between iterations. When the log likelihood increased by less than alpha = 5 I ended the training. This proved to be a good threshold for accurate pixel classification through cross-validation.

Another important parameter was the number of clusters. I found that adding too many clusters often led to too many misclassifications of other pixels as red in new images, possibly due to overfitting. This could possibly have been addressed by labeling other colors in the training set and creating more than one mixture in the model. I also found that just a single gaussian would both misclassify some pixels and not fully detect the red color in barrels of new images. I settled on two clusters which seemed to produce good classification of barrel pixels in new images and very little noise in the background.

Contour Analysis

After individual pixels were classified as red or not red in a binary image, contour analysis was used to detect barrels and their dimensions. These images were processed and contours were detected using standard image analysis libraries such as matplotlib and opencv. There were multiple steps which filtered out pixels misclassified as red and created a bounding box around every barrel that was detected.

The segmentation outputs of the gaussian mixture models had a small amount of misclassified pixels which appeared as noise in the background of the image. Using a binary erosion filter which removes pixels that are not surrounded by many other pixels all of this background noise was

effectively removed. For the rare cases where a large enough cluster of misclassified pixels existed in the binary image, it remained in the image after the erosion. After this binary erosion, a binary dilation operation was performed which dilated the remaining pixels according to a structuring element. This dilation was used to fill empty gaps within objects in the binary image. For example, some barrels had small objects occluding them but the space which was not classified as red could be filled with this dilation using a large structuring element. This connected parts of a barrel that were near each other, but which would otherwise belong to different connected components and would be detected as separate barrels. Furthermore, many barrels were not fully detected because their color was close to the cutoff probability of the gaussian mixture model, so this dilation operation filled gaps in the barrels and ensured that a more complete barrel binary image was generated.

After the erosion and dilation filters were applied to the binary images, most of them accurately displayed a binary mask of the barrels. In a few images there still remained some other small objects that were incorrectly classified as belonging to a barrel, but usually these objects were small and had an irregular shape. In order to isolate the connected components that belonged to barrels, I applied a size cutoff and aspect ratio test which eliminated nearly all of the misclassified pixels. To start, I used the regionprops function to find all connected components in the image. Then, I found the minimal vertical bounding box. I set a cutoff for the area of each connected component and also set a minimum of 1:1 as the ratio between the height and width of their bounding boxes as requirements for connected components to be considered barrels. I found that with these simple conditions only about one in twenty images had a misclassified barrel through cross-validation.

Linear least squares regression to predict distance

After the bounding box was found for each connected component remaining in the binary images, linear least squares regression was used to used to predict the unknown distances of barrels in the testing dataset given the dimensions of their bounding boxes. This regression model was trained using known distances of the labeled training set to predict distances of new barrel segmentations. The height and width of the labeled dataset was found using a bounding box after applying the same morphological dilation described above. Using the camera model, a linear relationship between the pixel height and width of the bounding box and the inverse of the distance to the barrel was determined using the equations in figure 2a. Then, the known height and width values were placed into a matrix and the distance values were placed into another. The linear least squares solver from the scipy library was then used to solve the system of equations in figure 2b. The output of this equation was the set of parameters α and β which were then be used to predict the unknown distances of barrels in the testing dataset.

**Figure 2**

**a)**

$$\frac{h_p}{f} = \frac{h_r}{d}$$

$$\frac{w_p}{f} = \frac{w_r}{d}$$

$$d = \frac{f * h_r}{h_p} = \frac{\alpha}{h_p}$$

$$d = \frac{f * w_r}{w_p} = \frac{\beta}{w_p}$$

**b)**

$$A = \begin{bmatrix} h_1 & w_1 \\ h_2 & w_2 \\ \vdots & \vdots \\ h_n & w_n \end{bmatrix} \quad x = \begin{bmatrix} \frac{1}{\alpha} \\ \\ \frac{1}{\beta} \end{bmatrix} \quad b = \begin{bmatrix} \frac{1}{d_1} \\ \vdots \\ \frac{1}{d_n} \end{bmatrix}$$

$$Ax = b$$
$$x = A^{-1}b$$

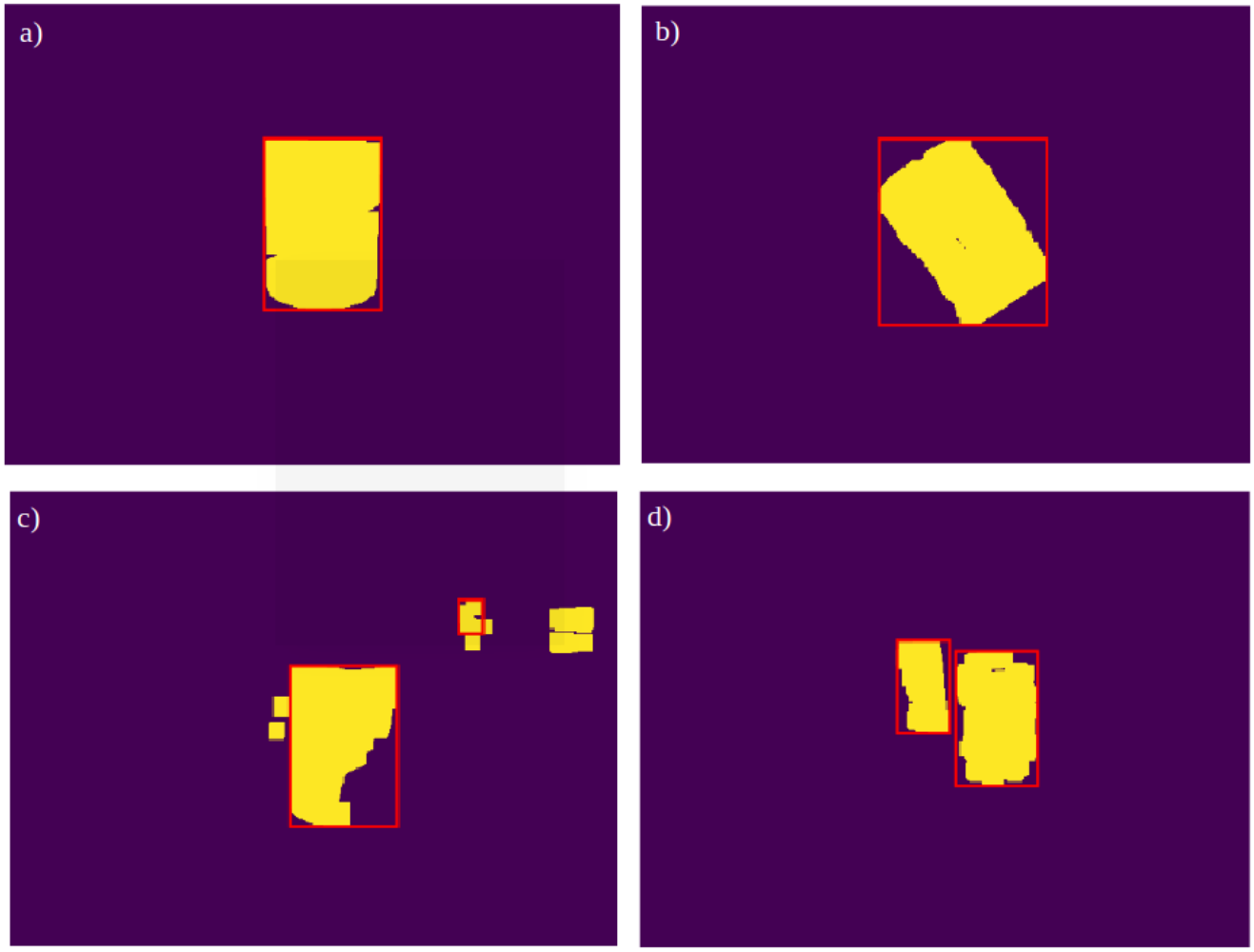
**Results**

Figure 4 shows the outputs of the barrel detection and distance prediction algorithm. For each barrel the bounding box was found using the approach described above and the corresponding centroid, height, width and predicted distances were printed to the console at the end of the program. Figure 5 shows some examples of detection outputs on the testing set and some cases in which the classification may not result in the correct bounding boxes and/or distance predictions. The appendix contains all of the image outputs of the testing set with the visual bounding boxes.

**Figure 4**

| | Centroid | Distance | Height | Width |
|---|---|---|---|---|
| 001.png Barrel #1 | (640.29, 481.91) | 2.35 | 316 | 211 |
| 001.png False Detection | (907.00, 247.73) | 11.19 | 67 | 49 |
| 002.png Barrel #1 | (616.92, 423.48) | 2.23 | 334 | 229 |
| 003.png Barrel #1 | (559.04, 383.79) | 4.05 | 181 | 103 |
| 003.png Barrel #2 | (702.36, 447.33) | 2.80 | 263 | 160 |
| 004.png Barrel #1 | (634.32, 440.93) | 2.09 | 367 | 330 |
| 005.png Barrel #1 | (646.17, 472.26) | 7.46 | 102 | 85 |
| 006.png Barrel #1 | (632.86, 427.07) | 9.25 | 81 | 59 |
| 007.png Barrel #1 | (566.00, 628.33) | 3.12 | 237 | 150 |
| 008.png Barrel #1 | (738.34, 477.36) | 7.74 | 94 | 49 |
| 009.png Barrel #1 | (673.35, 428.08) | 9.73 | 77 | 56 |
| 010.png Barrel #1 | (663.06, 398.28) | 6.73 | 111 | 78 |

**Figure 5**




**Discussion**

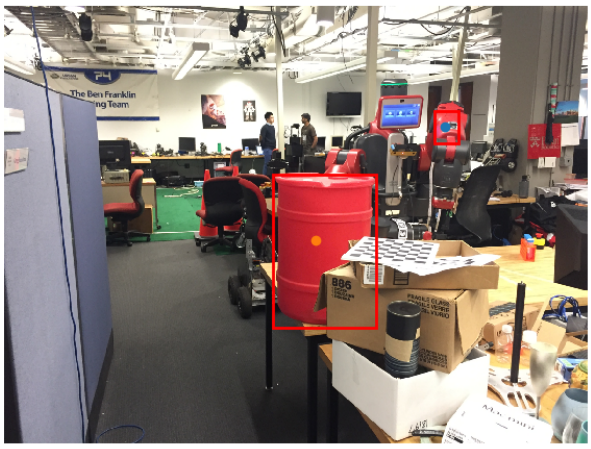
As a whole, the results of running the barrel detection algorithm worked very well on the testing set provided. Most of the bounding boxes appear to fit the barrels accurately and the distance predictions seem reasonable. For example, figure 5a shows the binary detection of a barrel that had a bar partially occluding it and separating it into two components but the binary dilation operation connected the components into an accurate detection of the barrel. There were a few cases which led to false detections of barrels or less accurate predictions of distance. Figure 5b shows a case where the barrel is detected correctly but is tilted, leading to an inaccurate height and width measurements and, as a result, a less accurate distance prediction. To improve on this, the area of the connected component could have been used in the linear regression instead of height and width. Figure 5c shows the only test case where another object was misclassified as a barrel. In this case a robotic arm in the background with similar HSV values to the barrel was large enough to get a bounding box and had a larger height than width. This led to a second barrel detection, with a distance and centroid prediction of its own, in image 001.png. To improve the misclassification of objects detected as barrels, a better implementation would use more advanced features of the objects detected to determine whether or not they are barrels, or would simply have a better color classifier for the “barrel red”. Finally figure 5d shows an example of how sometimes large parts of barrels are completely occluded, leading to bounding boxes which are too tall or wide. In this case, the smaller width most likely caused a less-accurate distance prediction.

This kind of occlusion is difficult to deal with since part of the barrel is not visible at all and can’t be classified by color. Possibly some form of feature identification could recognize it as a barrel and better predict its actual size.

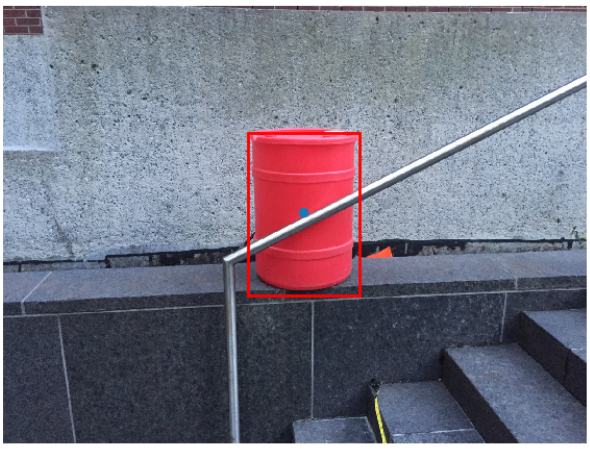
Overall, this method of detecting barrels has many advantages. First of all, it is very fast and mostly accurate, providing segmentation results that can be used in real-time. It is also very adaptable to other objects by simply using different labels and adapting some parts of the post-processing of binary images. It could even play a part of a larger model where a more expensive algorithm is run only on the pixels close to the cutoff to get improved segmentation. Some disadvantages include the inability to differentiate objects which have similar paint to a barrel and the fact that it relies only on simple features of the contour. Other techniques have advantages and disadvantages when compared to the technique presented. For example, convolutional neural networks are better at detecting objects based on their shape and other features, but require much more data and are much slower to train and run. Still, the simplicity of this technique and fast implementation make it a great starting point from which to develop and evaluate more advanced techniques for object detection.

## Appendix

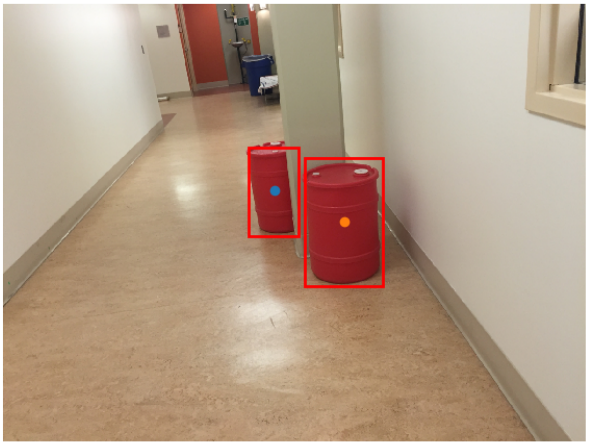
001.png



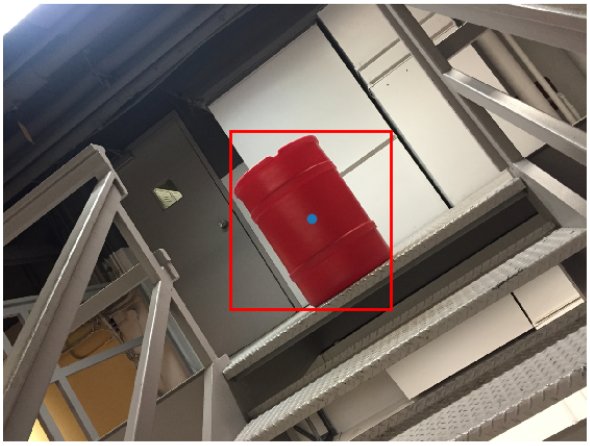
002.png



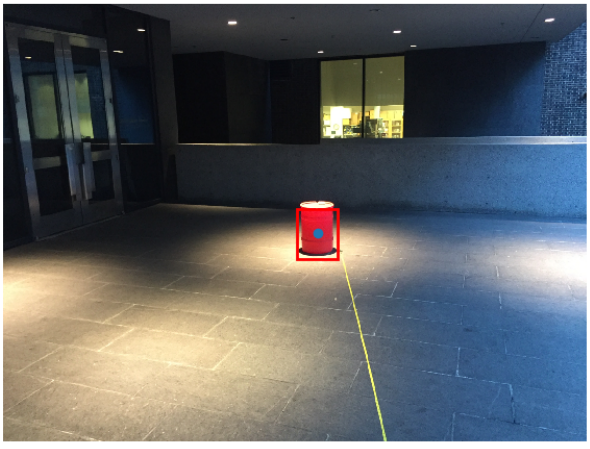
003.png



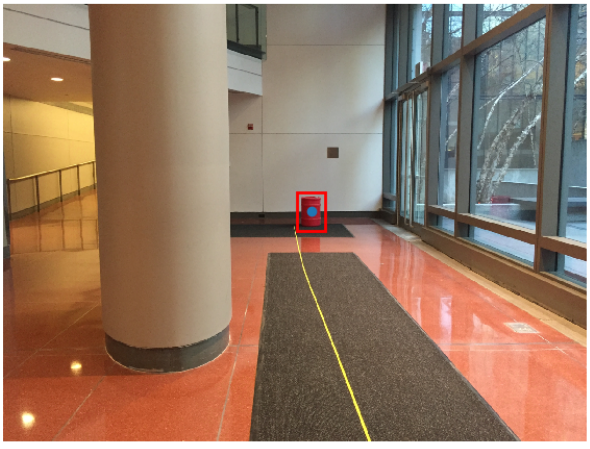
004.png



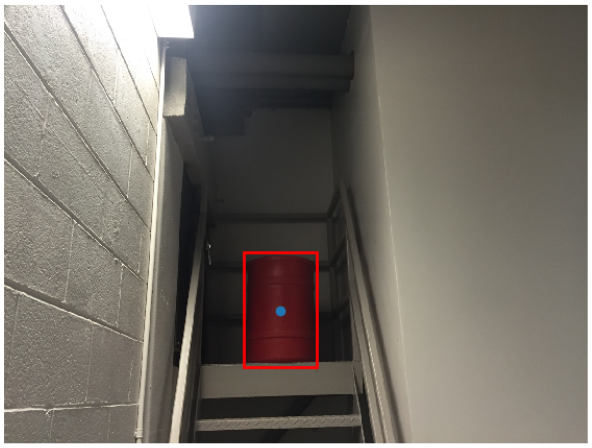
005.png



006.png



007.png



008.png

009.png

010.png