Project 1: Color Segmentation

Alexander Khoury
A11563298

January 26, 2018

1 Introduction

The problem of fast object detection using vision algorithms is very useful in the field of robotics, whether it is used for applications such as obstacle avoidance, or even localization. In this project, we look to robustly detect a red barrel in an image in a highly varying environment in minimum computation time.

2 Problem Formulation

Let $X^{m \times n}$ be an RGB image which contains one or multiple red barrels, and let $x_{i,j} \, \epsilon \, \mathbb{R}^3$ be the pixel corresponding to the i, j component of the image. Letting k be the number of color classes, estimate a multi-dimensional Gaussian distribution for each color class. Classify each pixel, $x_{i,j}$, in the image under the single Gaussian assumption. After the image is segmented into k colors, use statistical methods to analyze the red regions to locate the barrels, and estimate the distance to the barrel using regression methods.

3 Technical Approach

3.1 Labeling

Given a training set of 50 images each containing one or more barrels, each image needed to be annotated with examples from each of the k color classes. This requires an intelligent selection of classes, as well as an efficient labeling method.

3.1.1 Color Class Selection

Because the most important class is the color of the barrel, optimal color classes should fill a majority of the color space, as well as include colors closer to barrel color, barrel-red, such that the algorithm can differentiate between barrel red and other near-red colors. Thus the $k = 6$ classes intuitively defined are as follows: *barrel-red, not-barrel-red, brown, light gray,*

dark gray, and green, where the *not-barrel-red* class consists of colors that are close to barrel red, such as different shades of red, orange, and violet.

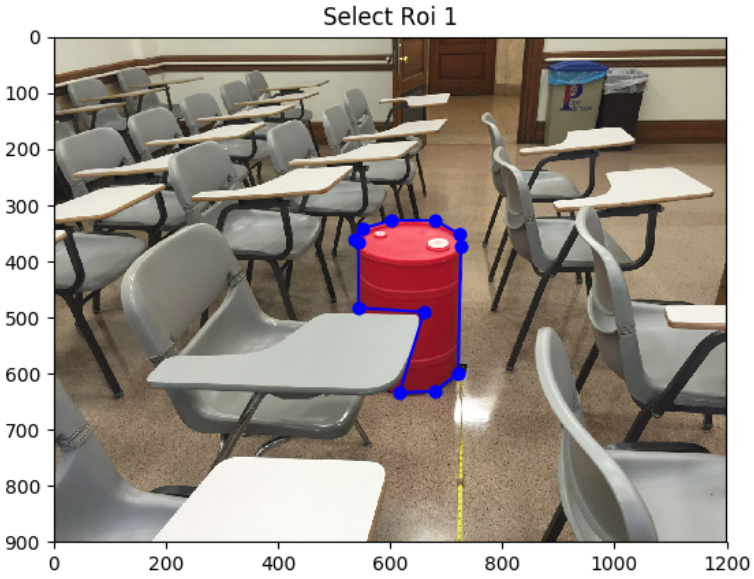

Figure 1: Using roipoly to annotate the barrel-red color class with a

3.1.2 Annotation

Using the roipoly package, color classes could be labeled by interactively drawing a polygon onto a window containing the image. Once a polygon is defined, all the pixel values and other features within the polygon can be easily obtained. Moving from one class at a time, each image was labeled with as many examples of the class as possible.

In addition to classifying the pixels into the k color classes, distance to the barrel needed to be estimated from the geometric properties of the barrel. The distances to the training barrels were given for every image, thus by annotating a tight rectangle around the barrel, the barrel's geometry was captured. All the examples were serialized and saved to file allowing for later use.

3.2 Training

Using the information obtained by labeling the training images, we are able to estimate the parameters of our Gaussian model, as well as fit a regression model to the barrel's geometric data to compute distance to the barrel.

3.2.1 Preprocessing

For each color class, pixels from annotated regions were extracted and separated, to achieve large amounts of pixel examples for each color class.

3.2.2 Calculating MLE parameters of Gaussian model

The conditional distribution for a pixel $x_{i,j} \epsilon \mathbb{R}^3$ in an image given the color class $Y = k$ can be modeled under a Gaussian assumption as:

$$P_{X|Y}(x_{i,j}|Y = k; \mu_k, \Sigma_k) = \frac{1}{\sqrt{(2\pi)^3|\mathbf{\Sigma}_k|}} \exp\left(-\frac{1}{2}(x_{i,j} - \mu_k)^T \mathbf{\Sigma}_k{}^{-1}(x_{i,j} - \mu_k)\right) \tag{1}$$

The MLE for parameters $\omega = [\mu_k, \Sigma_k]$ can then be obtained, where X is a $N \times 3$ vector of pixels from class k.

$$w_{MLE} = \underset{\mu_k, \Sigma_k}{\operatorname{argmax}}\ P_{X|Y}(X|Y = k; \mu_k, \Sigma_k) \tag{2}$$

which when evaluated corresponds to the sample mean and sample covariance of the observed data X,

$$\begin{aligned} \mu_k &= \frac{1}{N}\sum_{i=1}^{N} x_i \\ \Sigma_k &= \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_k) \cdot (x_i - \mu_k)^T \end{aligned} \tag{3}$$

3.2.3 Exponential fit for distance estimation

In order to estimate the distance from the barrel to the camera, we needed to fit a regression model correlating our distance and geometric data from the barrels. Looking at the plot of width of the barrel versus distance (below), we can see there is a exponential relationship between the two.

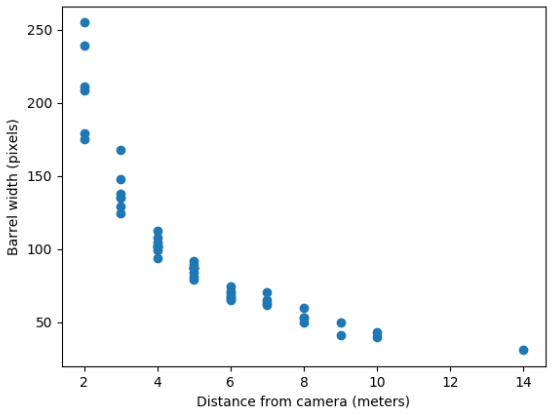

Figure 2: An negative-exponential relationship can be seen relating distance to barrel and the width of the barrel in pixels

Thus an exponential regression model was used, taking the form,

$$y = e^{a+b \cdot x} \tag{4}$$

letting y be the distance to the barrel, and x being the width or height of the barrel. Note that after taking the natural log of both sides,

$$ln(y) = a + b \cdot x_i \tag{5}$$

we can define our error as,

$$e = ln(y) - a - b \cdot x \tag{6}$$

with which we can define a weighted loss function,

$$\text{minimize} \quad \sum_{i=1}^{N} y_i \cdot (ln(y) - a - b \cdot x)^2 \tag{7}$$

which after taking the partial derivative which respect to each parameter and solving, using the notation $< x >= \frac{1}{N} \sum_{i=1}^{N} x_i$

$$\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} < y > & < xy > \\ < xy > & < x^2 y > \end{bmatrix}^{-1} \cdot \begin{bmatrix} < y \cdot ln(y) > \\ < xy \cdot ln(y) > \end{bmatrix} \tag{8}$$

which can be used to calculate distance from the width or height in pixels of the barrel. Thus, to improve accuracy, an exponential fit was preformed for both height and width, and upon computation of the estimate with test data, the final estimate is the average of the two estimates.

3.3 Segmentation

Given a pixel, $x_{i,j}$ in X, the color class chosen for that pixel corresponds to finding the color class k that maximizes the log likelihood of our Gaussian model.

$$\underset{k}{\text{argmax}}\ log(P(Y = k|X)) = \underset{k}{\text{argmax}}\ log(P(X|Y = k)) + log(P(Y = k)) \tag{9}$$

and assuming the prior probability of seeing any color is the same, the equation simplifies to,

$$\underset{k}{\text{argmax}}\ log(P(X|Y = k)) \tag{10}$$

which after plugging in our model and simplifying is equal to,

$$\underset{k}{\text{argmax}}\ log(\sqrt{(2\pi)^3} \cdot |\Sigma_k|) + \frac{1}{2}(X - \mu_k)^T {\boldsymbol{\Sigma}_k}^{-1}(X - \mu_k) \tag{11}$$

where we can compute the most likely class for each pixel $x_{i,j}$ and form a color segmented image.

However, computing the second term, the Mahalonobis distance, in the above equation for large vectors of size N is computationally expensive, as a $N \times N$ matrix would be computed, whose diagonal represents the $N \times 1$ vector desired. However, using the fact that, $a^T V^{-1} a = \|V^{-1/2} a\|_2^2$ for a positive definite, symmetric matrix V, calculating the Mahalonobis distance can be calculated by preforming a Cholesky Decomposition on the inverse of the covariance matrix, which is positive definite and symmetric matrix Σ_k^{-1}, where $\Sigma_k^{-1} = S_k^T S_k$, can greatly reduce computation time, as the resulting distance for each pixel in a vectorized image $X^{N \times 1}$ can be calculated as follows:

$$D = \sqrt{(X - \mathbf{1}\mu_k^T) S_k^T \cdot \mathbf{1}} \quad \text{where } \sqrt{\cdot} \text{ is a component wise square root} \tag{12}$$

3.4 Detection

Using this color segmented image, a binary mask can be created for the barrel-red pixels in the image, and using openCV's findContours function, contours can be drawn around all the barrel-red regions. We can analyze each of these contour's properties, such as area, center, height, and width to narrow down the candidates to the contour that contains the barrel.

One situation that can arise is that portions of the barrel may be covered by an object, resulting in a partitioning of the barrel's contour. Such that if any analysis were done on any of the partitions alone, no barrel would be detected. Only the combination of all the partitions would result in a detected barrel. Thus any contours whose centroids were located within a distance threshold from each other were iteratively combined.

From the training data, mean barrel ratio, defined as height/width, was calculated, as well as the standard deviation in this ratio. A barrel is defined as a contour whose bounding box satisfies:

$$\begin{aligned} &\mu_{ratio} - 2 \cdot \sigma_{ratio} < Box\ Ratio < \mu_{ratio} + 2 \cdot \sigma_{ratio} \\ &Contour\ Area > 5\ pixels \end{aligned} \tag{13}$$

Setting a lower limit on the contour area help eliminate contours that are too small to be barrels. Otherwise, by taking advantage the uniform size and shape of the barrels, we are able to use the ratio between height and width to determine barrelness.

Another situation that can occur is the barrel can be in front of or right next to another barrel-red like object, where the difference in color was not discriminated via segmentation. In this case, we may observe a very large contour containing all the objects. Thus the algorithm works on a two-attempt basis, such that if no barrel is detected on the algorithms first attempt, it tries again but investigates further into any large red contours, if they exist, by running a k-means clustering algorithm on the original pixels within the contour. Using this method, the algorithm hopes to separate the barrel's color from any other close-red colors in the contour. The cluster containing a mean closest to the barrel-red sample mean obtained from training is kept, and the associated pixels would form a new contour (seen below). Then, if exists, barrel detection would be tried with this new contour.

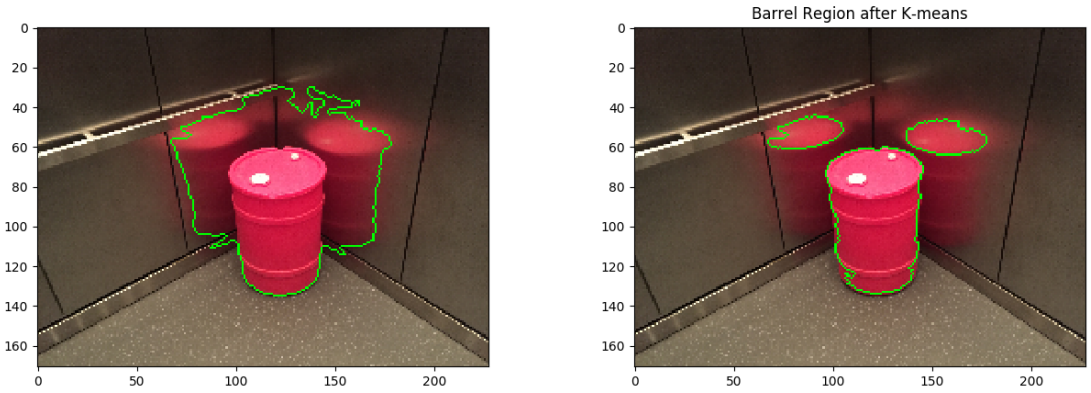

Figure 3: Contour before k-means (left), Contour after k-means (right)

4 Results

4.1 Training Results

Taking a 80/20 split for training/validation, 710304 training pixels and 143327 validation pixels, the barrel-red pixels were detected with an error of 0.063212. This error may be due to human error mislabeling pixels with crudely placed polygons.

4.2 Test Results

This algorithm was able to successfully detect 9/10 barrels in the test images. Images showing the detected barrels and color segmentations are shown below:

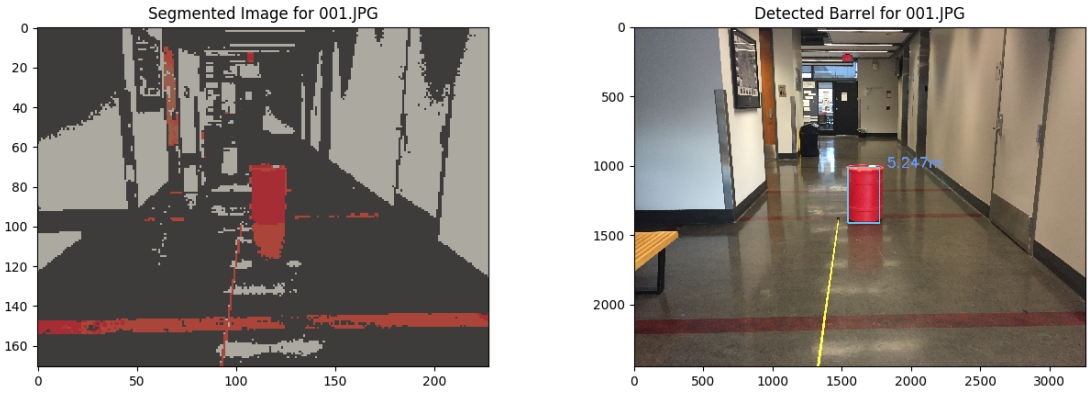

Figure 4: 001.JPG, Distance = 5.247m

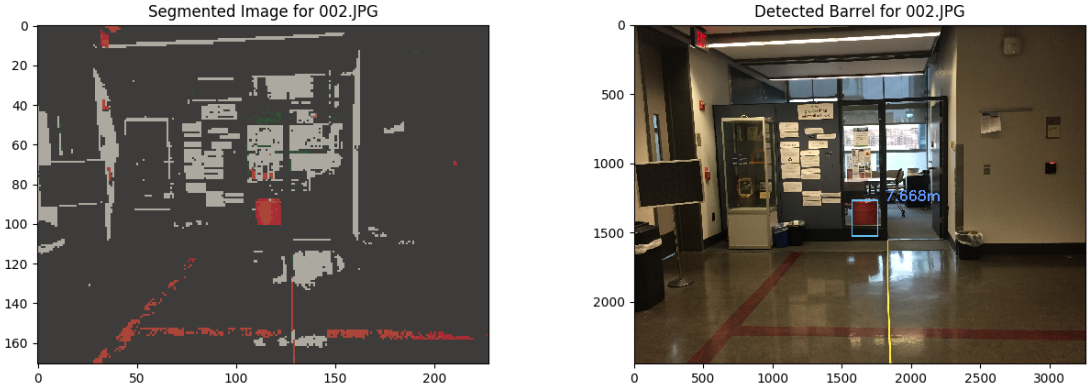

Figure 5: 002.JPG, Distance = 7.668m

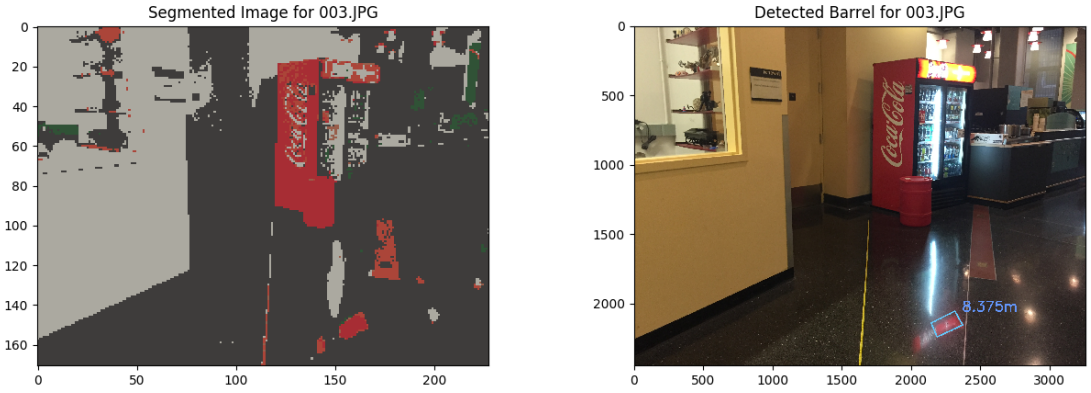

Figure 6: 003.JPG, Distance = 8.375m

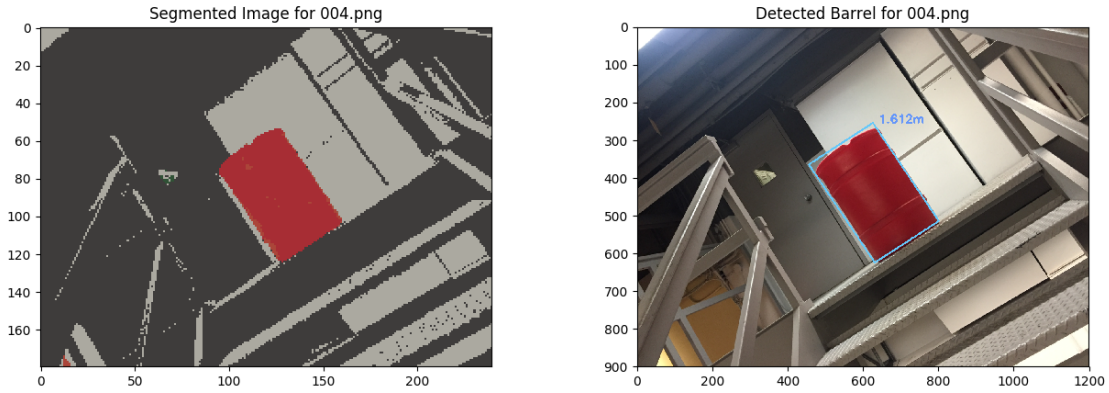

Figure 7: 004.png, Distance = 1.612m

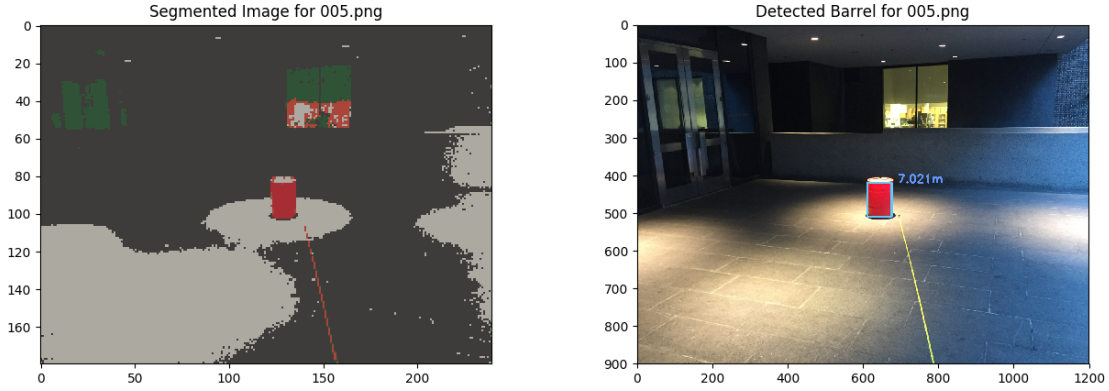

Figure 8: 005.png, Distance = 7.021m

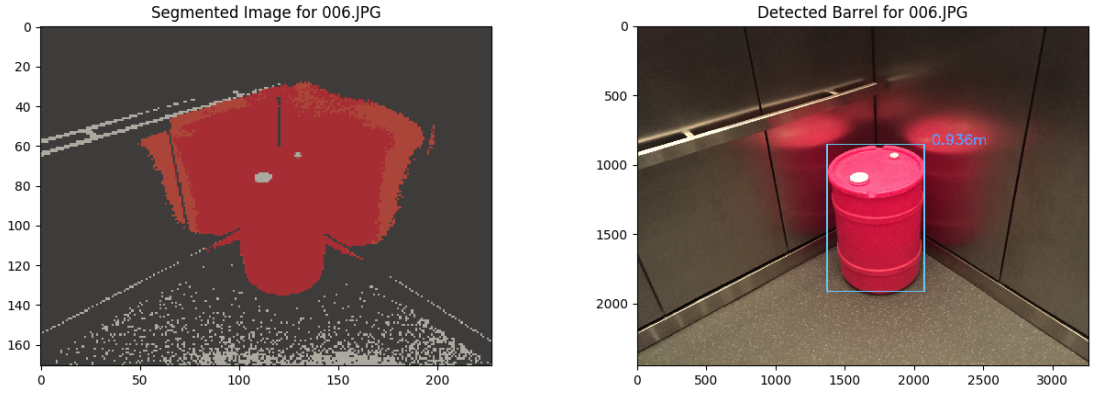

Figure 9: 006.JPG, Distance = 0.936m

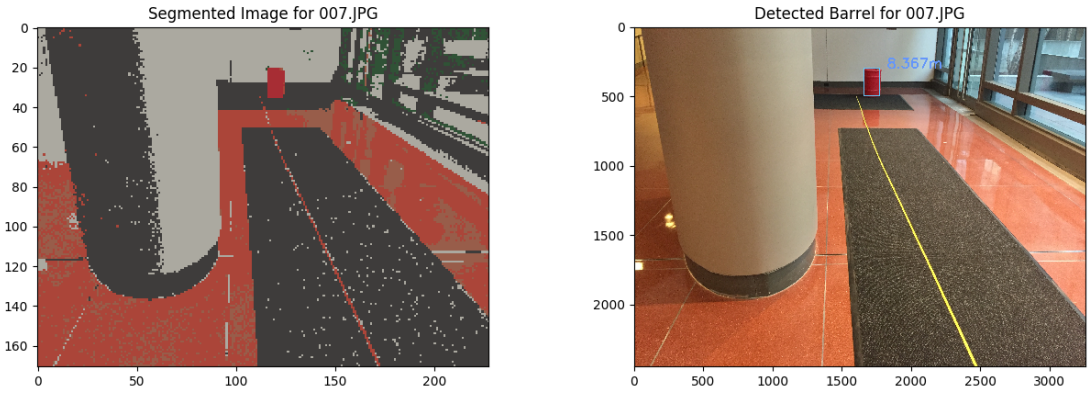

Figure 10: 007.JPG, Distance = 8.367m

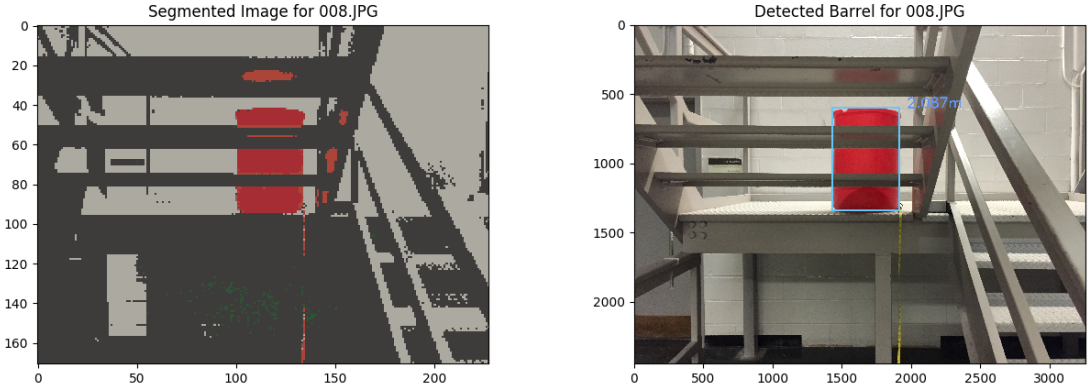

Figure 11: 008.JPG, Distance = 2.087m

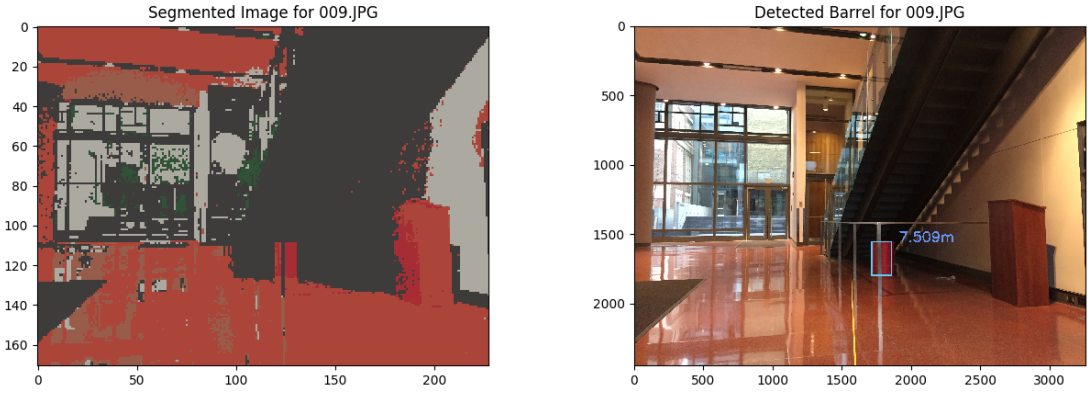

Figure 12: 009.JPG, Distance = 7.509m

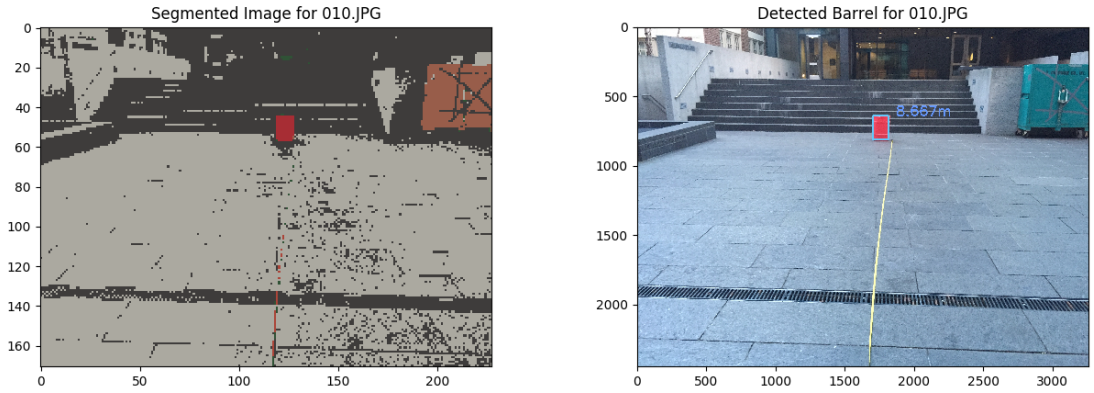

Figure 13: 010.JPG, Distance = 8.667m

```
ImageNo = [01], BottomLeftX = 1543, BottomLeftY = 1414
                TopRightX = 1771, TopRightY = 1014, Distance = 5.2467
ImageNo = [02], BottomLeftX = 1571, BottomLeftY = 1529
                TopRightX = 1757, TopRightY = 1271, Distance = 7.6685
ImageNo = [03], BottomLeftX = 2186, BottomLeftY = 2243
                TopRightX = 2314, TopRightY = 2057, Distance = 8.3747
ImageNo = [04], BottomLeftX = 630, BottomLeftY = 625
                TopRightX = 625, TopRightY = 255, Distance = 1.6124
ImageNo = [05], BottomLeftX = 610, BottomLeftY = 510
                TopRightX = 675, TopRightY = 420, Distance = 7.0209
ImageNo = [06], BottomLeftX = 1371, BottomLeftY = 1914
                TopRightX = 2071, TopRightY = 857, Distance = 0.9358
ImageNo = [07], BottomLeftX = 1657, BottomLeftY = 500
                TopRightX = 1771, TopRightY = 300, Distance = 8.3666
ImageNo = [08], BottomLeftX = 1429, BottomLeftY = 1343
                TopRightX = 1914, TopRightY = 600, Distance = 2.0872
ImageNo = [09], BottomLeftX = 1714, BottomLeftY = 1800
                TopRightX = 1857, TopRightY = 1557, Distance = 7.5089
ImageNo = [010], BottomLeftX = 1700, BottomLeftY = 814
                TopRightX = 1814, TopRightY = 643, Distance = 8.667
```

Figure 14: Results

4.2.1 Remarks

With an accuracy of 0.9, only one barrel was not detected correctly. This was the test image, '003.JPG,' and the difficulty can be accreditted to the similarity of color of the coke machine and the barrel, such that not even the implemented k-means was able to differentiate between the two colors. Thus a reflection on the floor was marked instead.

Throughout the course of developing the algorithm, I faced many problems. One of which

was the situation faced by '006.JPG,' where the reflections of the barrel was classified as the same color as the barrel in the first attempt. I was able to solve this using the kmeans method I described.

The other problem I faced was minimizing computation time, as the images are high resolution and looping through the pixels one by one would take to long to compute, and classifying the image as a whole would use too much memory. My first solution to this issue was to break the image up into chunks of 10,000 pixels and compute the chunks and concatenate the result. This was still too slow, as I then decided to resize the image to a much smaller resolution, small enough such that it did not compromise accuracy of detection, but significantly reduce computation time. The resolutions were scaled as follows: 0.07 times smaller for the 2448x3264px images, and 0.2 times smaller for the 900x1200px images. This in combination with the "chunking" method enabled my detection to work in about 15 seconds. Wanting faster, I was able to utilize the Cholesky decomposition to compute my classification in under 0.2 seconds.