ECE 5242
Intelligent Autonomous Systems

Project 1

By: Samuel Choi, sgc87

Date: February 11, 2020

Intro

The purpose of this project is to create a model that detects red barrels in real-world images and estimates their distances based on a training model. Using Gaussian models classified with color segmentation, we are able to detect regions of pixels that lie within the Gaussian probabilistic space of the color of the pixels from the training data. Using Bayes rule, used the Gaussian probability of a pixel belonging to each class with the Gaussian formula being

$$p(\mathbf{x}) = \frac{1}{2\pi \| \mathbf{\Sigma} \|^{1/2}} \exp\left(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

and the Bayesian formula being,

$$P(y|x) = \frac{P(y)P(x|y)}{\Sigma P(y)P(x|y_i)}$$

Problem Statement

One can identify the regions of red barrels with training images and approximate pixels in test images that lie within that same color range. Using Gaussian mixture models, one can create several classes and create a Bayesian probabilistic model to classify each pixel of the test images. However, the problem is that the variation in lighting condition and presence of objects with similar colors may interfere with that classification. These issues may be resolved by using a different color space, changing the number of GMM or number of classes, filtering, or a combination.

Approach

Number of GMM:

The number of GMMs used was one. There were two different classes for the model: barrel and not barrel. Although more GMM classes could have been created, just using the two classes was enough to distinguish the red regions and highlight the red barrels in all images except for one, where the lighting conditions were poor. As a result, a two class model was the one implemented for this object detection algorithm. However, if the region detection was poor with the one GMM, one could relabel the training images with more than two classes to see if that helps. This project only calls for the detection of red barrels, so the separation of classification for non-barrel objects would have been redundant as this model only looks at color and not the shape of the objects. An EM algorithm could have been used, but it was not deemed necessary for this case.

Color Space:

The color space used was RGB. Initially, my barrel detection classification performed very poorly, even detecting green turf as red barrel pixels. However, this was fixed once I relabeled my training data pixels more carefully. Initially, I traced the barrels very roughly, leading to a very poor classification model. However, after reselecting my training image pixels, my classification algorithm performed very well. If it still performed poorly after relabeling, perhaps it would have been a good idea to change the color space so that red regions were more distinguished from the rest. If this were the case, I would use Photoshop or some photo editing tool with a histogram to see what sort of adjustments to the color space or rgb saturation levels would help distinguish the red pixels. Since my classification model distinguished the regions well, I stuck with the RGB color space.

Filtering:

After detecting regions of pixels detected as red barrel pixels from the classification images outputted as red in the output images from the algorithm in bayes.ipynb using the regionprops tool from the skimage library, several steps were used to filter images.

- First, there were many small regions of pixels that were classified as an area of interest. To filter those out, I took the parameter of `minor_axis_length` from each region and deleted the regions with a `minor_axis_length` less than 5.
- Next, I determined the minimum value of the region attribute `bbox_area` by taking the `bbox_area` of the smallest detected barrel as a starting point and then reduced it until it started detecting the next smallest red item that wasn't a barrel. I set my threshold between these values at 1250.
- Next, I used the area of each region and got the percent of pixels that were classified as red barrel. This step was useful in getting rid of arbitrary detected areas with sparse detection area coverage. The threshold of this was set so that if under 50% of the area was filled, the region would be eliminated using `filled_area / bbox_area` as the calculation.
- Next, I used the ratio between the height and width of each region detected. Taking into account regions that may be cut off, the range

was set to keep the regions with a ratio lying in between 1 and 3. The calculation for this was `major_axis_length`/`minor_axis_length`.

- Last, the ratio of the remaining regions to the largest region was used as the last filtering mechanism. This was done as some areas cut off from the barrels were still identifying as barrels. However, we want to not eliminate all barrels smaller than the largest one, because there could be multiple barrels. As a result, all boxes with `bbox_area` that are at 30% the size of the largest barrel were kept.

Distance:

In the file, get_distance.ipynb, the estimated height and width of the barrels of each classified training image were determined with regionprops attribute, `bbox`, and the width and height were multiplied with the given distance from the camera in the first number from the file name. Then I calculated the mean for both the height*distance and the width* distance

The calculated means were then divided by the respective estimated height and width in the identify.ipynb file and divided by 2 to get their average. This was then rounded and determined as the estimated distance.

Although the detected height and width may not be perfect, if one dimension is skewed, the other is there to compensate for the discrepancy. This model works, because the ratio between the pixels and the actual image has a linear correlation due to the geometry of similar triangles. The similar triangles enable the ratio to remain constant, enabling an average of values from the training data to be used to estimate the distance of the test images. The following was the line used to calculate the estimated distance.

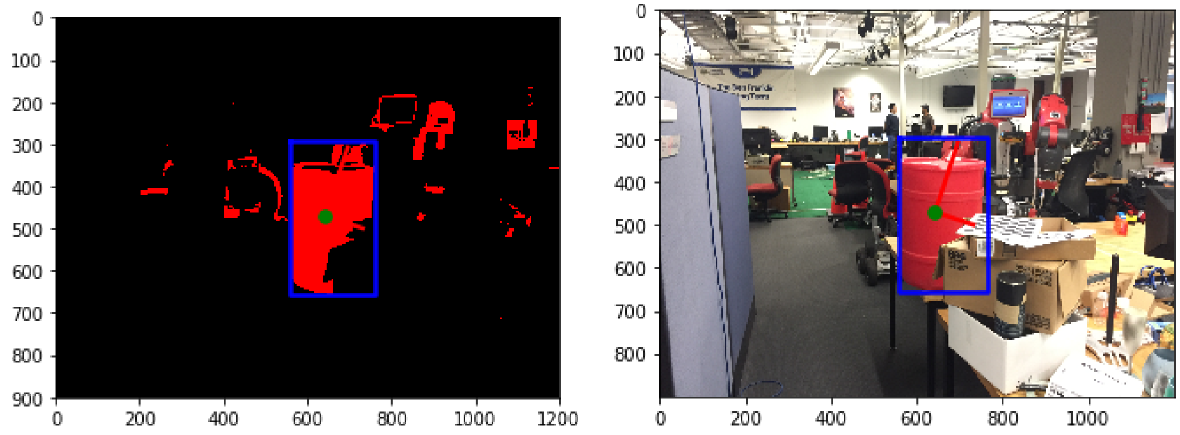
```
np.round((450.3636363636364/w + 648.3863636363636/h)/2)
```

Results

* Note the only changes made after initial code submission was the addition of comments and the addition of the distance calculation. This affected 2 lines of code in identify.ipynb and a separate file, get_distance.ipynb was made.
* Also, a block that was used for testing was eliminated in the identify.ipynb file.

Image 1:

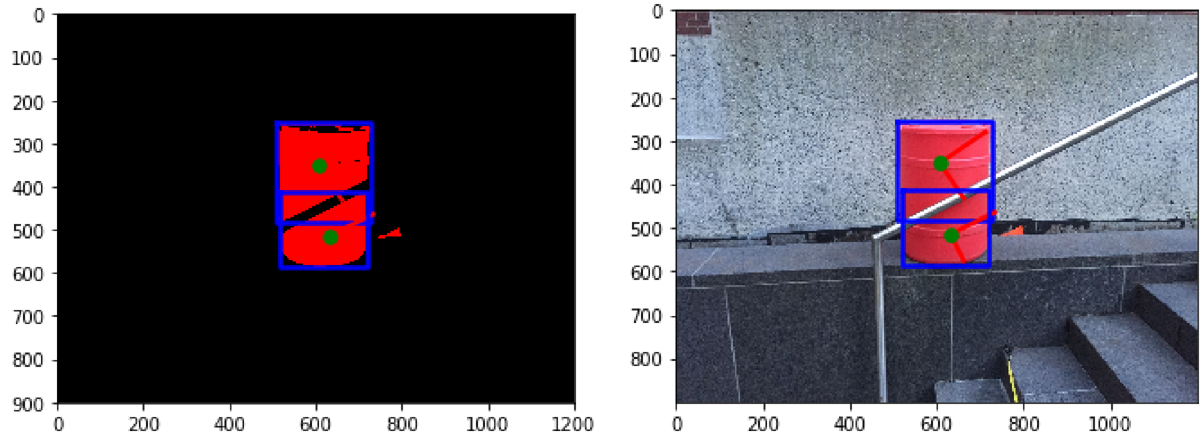
```
ImageNo = [001], CentroidX = 642.6896543703773, CentroidY = 470.3222077499186, Distance = 2.0
```



We can see that the barrel has been correctly identified. It seems that the robot behind may have skewed the height a little bit, but overall, it has identified the region pretty well.

Image 2:

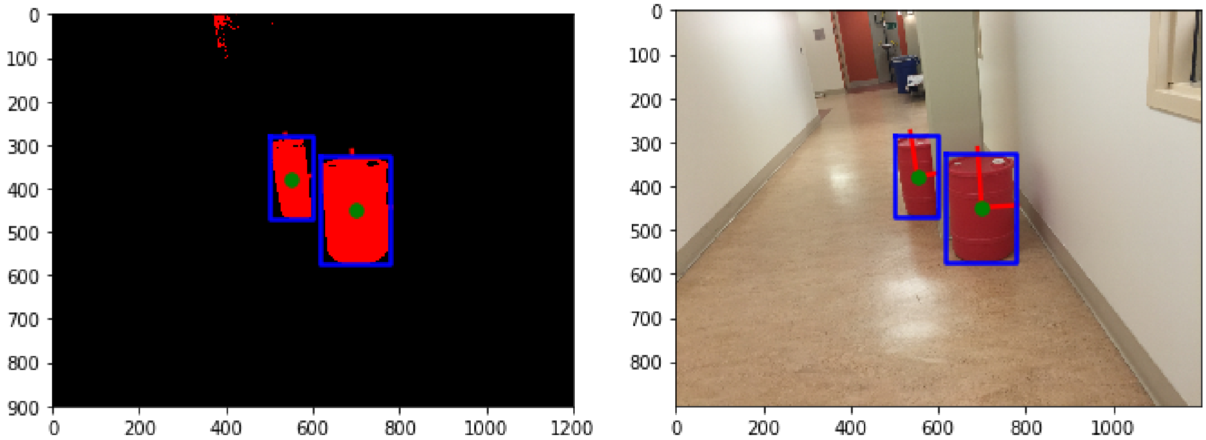
```
ImageNo = [002], CentroidX = 607.2284823165846, CentroidY = 350.5270687879135, Distance = 2.0
ImageNo = [002], CentroidX = 632.1024782608696, CentroidY = 513.8512608695652, Distance = 3.0
```



We can see that the pole going across the red barrel in the image has thrown off the classification mechanism. Since there is a perfect break in the image. Perhaps having an exception for overlapping boxes could fix this.

Image 3:

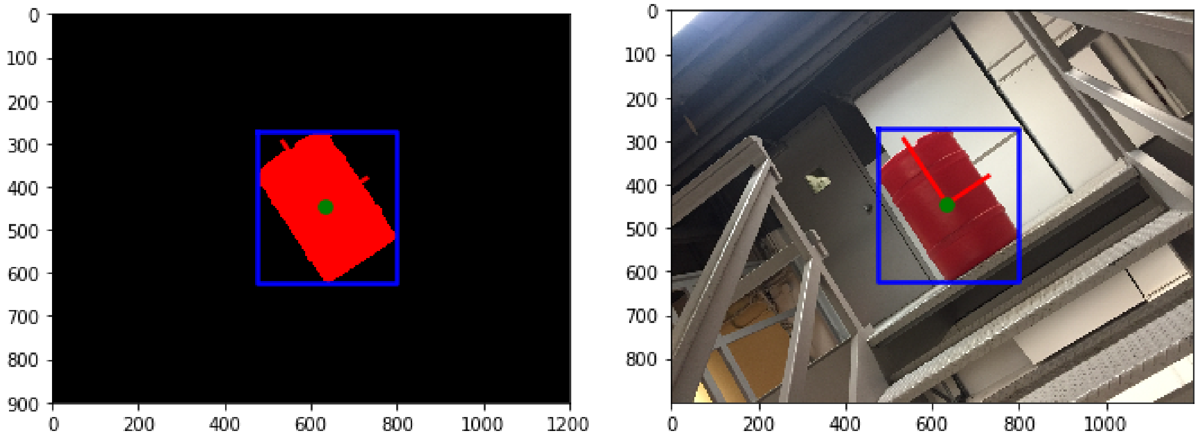
```
ImageNo = [003], CentroidX = 552.0389629851641, CentroidY = 377.5125131125431, Distance = 4.0
ImageNo = [003], CentroidX = 698.9842642890883, CentroidY = 448.53271138403403, Distance = 3.0
```



We can see here that the algorithm has identified two barrels correctly. Also since there aren't many other artifacts interfering nearby, this image seemed to be an easy one to identify red barrels in.

Image 4:

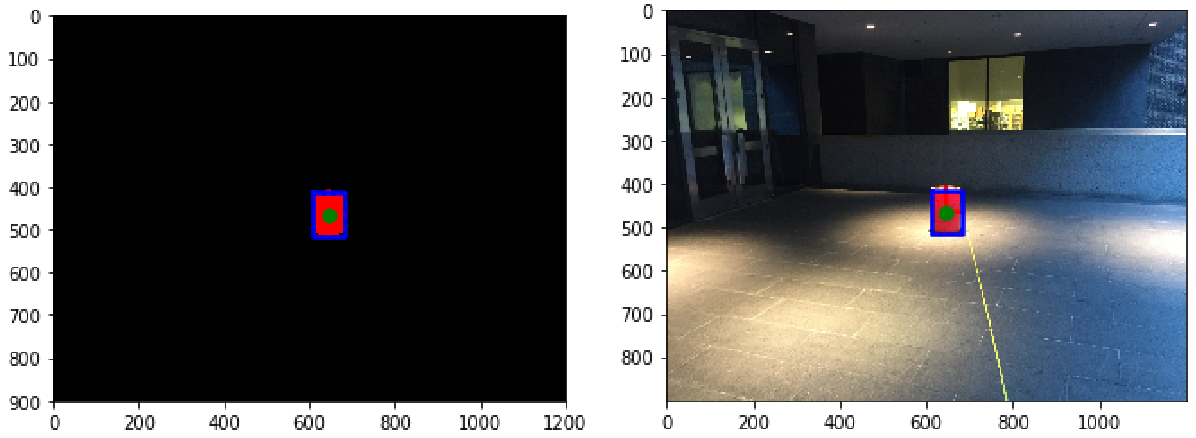
```
ImageNo = [004], CentroidX = 634.4984755351932, CentroidY = 444.33515399540215, Distance = 2.0
```



This image correctly identified the barrel. Although the image was taken at an angle, the red barrel stands out from the rest of the image, making it easy to identify.

Image 5:

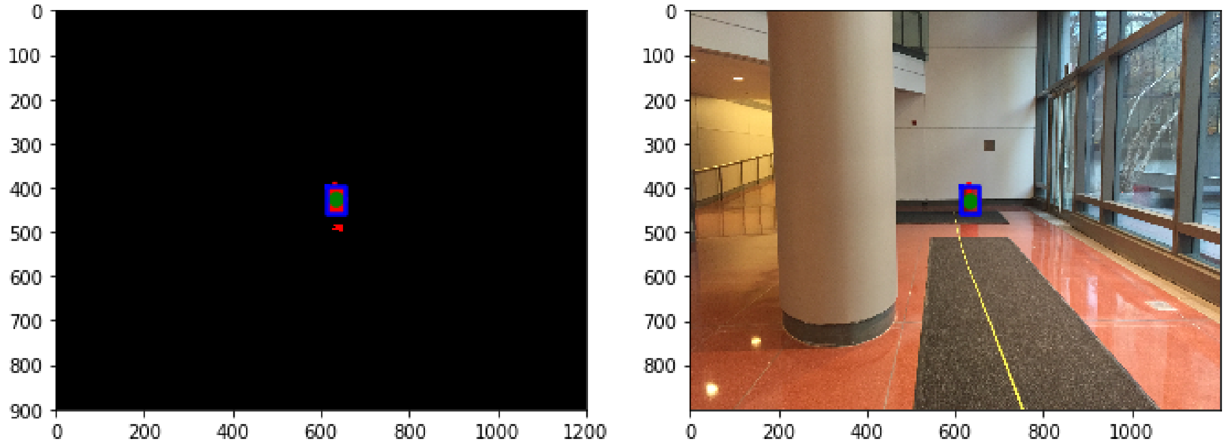
ImageNo = [005], CentroidX = 645.3195034302515, CentroidY = 464.78079059131005, Distance = 6.0



The red barrel has been correctly identified in this image with little issue. The top of the barrel was excluded, but overall the classifier did well in this image.

Image 6:

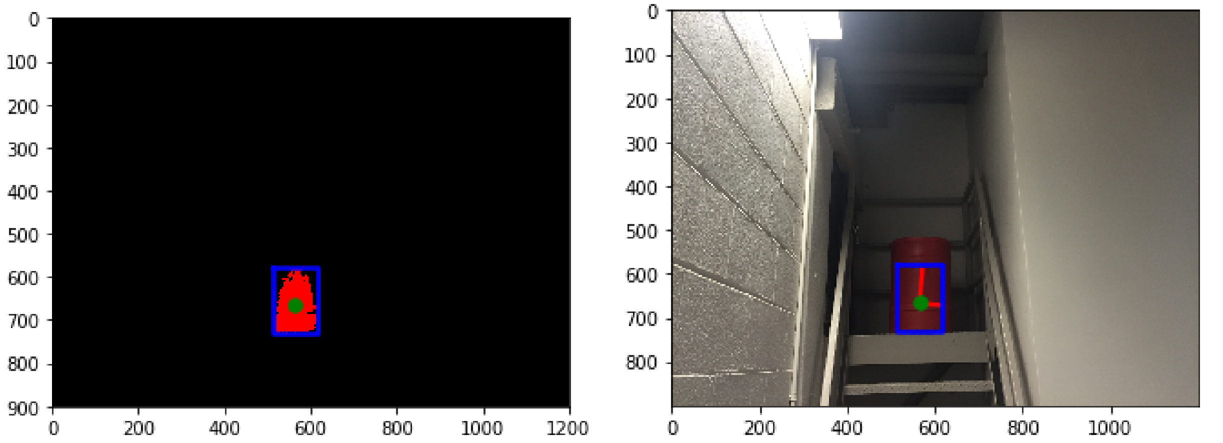
ImageNo = [006], CentroidX = 632.1589656634167, CentroidY = 426.81517592200083, Distance = 11.0



Despite the reflection of the barrel on the floor being classified as a possible red barrel region, the filtering did a good job of eliminating it. Also the floor is red, but not quite in the RGB colorspace region to be identified as a red barrel.

Image 7:

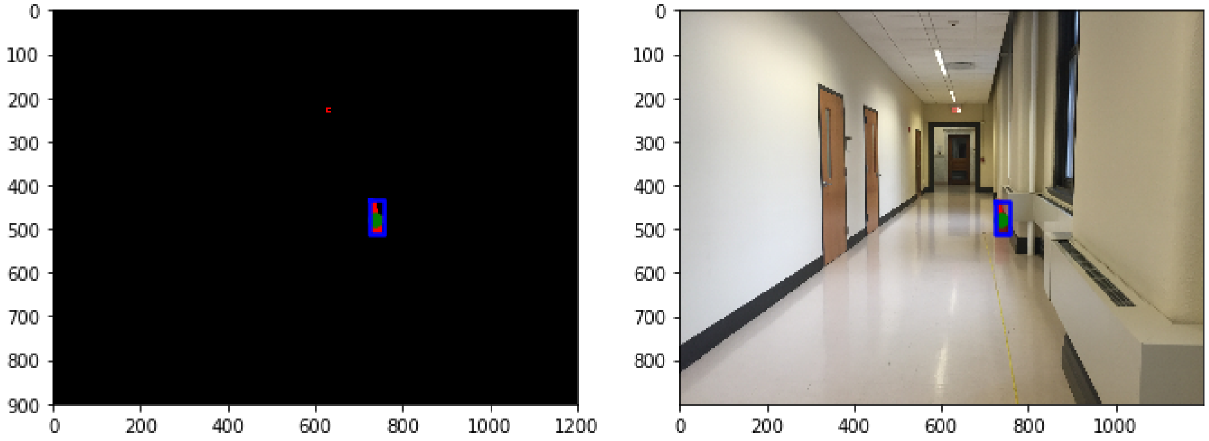
ImageNo = [007], CentroidX = 564.4218326230143, CentroidY = 666.270728399845, Distance = 4.0



The barrel has been correctly identified overall. However, due to the poor lighting, the top portion was a bit cut off. Perhaps adjusting the lighting in this image would have made the outcome better.

Image 8:

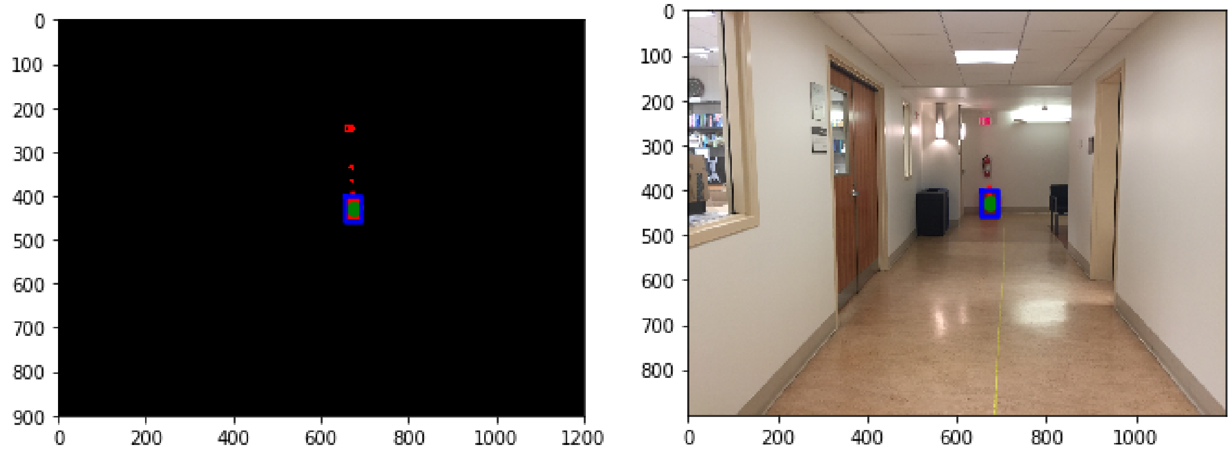
ImageNo = [008], CentroidX = 737.4533258803801, CentroidY = 478.02459474566797, Distance = 11.0



Despite being so far away, the barrel was correctly identified. However, due to being a bit cut off, the distance calculations seem to be a little off.

Image 9:

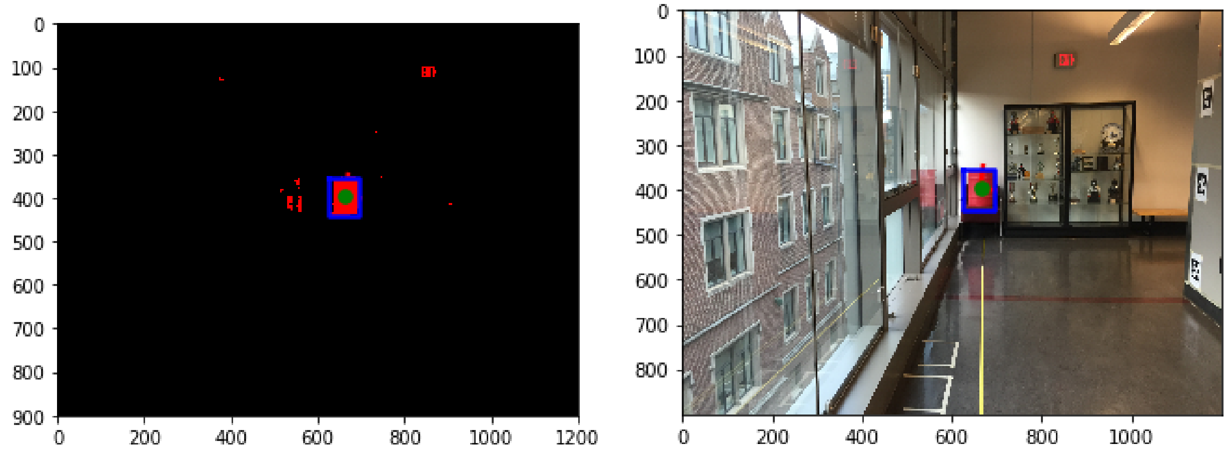
ImageNo = [009], CentroidX = 673.2832221652129, CentroidY = 427.29861467419187, Distance = 12.0



Despite there being several other red object that tricked the classification, the filtering did a good job of eliminating the non-barrel objects. As a result, the detection of this images seems to be accurate.

Image 10:

ImageNo = [010], CentroidX = 663.8138108428967, CentroidY = 397.00811238622873, Distance = 7.0



Although the reflections and red exit sign have interfered, the filtering has once again a good job at correctly identifying the barrel.