Andrew Gates

Professor Nikolay Atanasov

ECE 276A

25 October 2017

Project 1: Color Segmentation

Introduction:

This problem is important on multiple levels. The first being that it is a good way to apply what we have learned about Probability Theory and Supervised Learning to a real world problem, image detection. My approach to this problem was to process every image to get pixel data for mean, variance and covariance. Using these values I calculated a BDR for a single Gaussian which decided whether the color was Red, Brown or Gray. I did this pixel by pixel until I had a mask which I could use to see everything that was the color Red. With this mask I then calculated a bounding box around the area of interest (barrel) which I then drew back onto the image. Using the area of this box I also calculated how far away the barrel was.

Problem Formulation:

In order to solve this problem we needed to calculate the following quantities –

Mean of Pixel Data:

$$\overline{X} = \frac{\sum X}{N}$$

Variance of Pixel Data:

$$\sigma^2 = \frac{\sum (X-\mu)^2}{N}$$

Covariance of Pixel Data:

$$\Sigma = \mathbf{E}\left[(\mathbf{X} - \mathbf{E}[\mathbf{X}])\,(\mathbf{X} - \mathbf{E}[\mathbf{X}])^{\mathrm{T}}\right]$$

BDR to Determine Pixel Class:

$$p(\mathbf{x}\,|\,\theta_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu_i)^T\Sigma_i^{-1}(\mathbf{x}-\mu_i)\right]$$

Described by a d-dimensional mean μ_i
and a $d \times d$ covariance matrix Σ_i

Linear Regression for Barrel Distance:

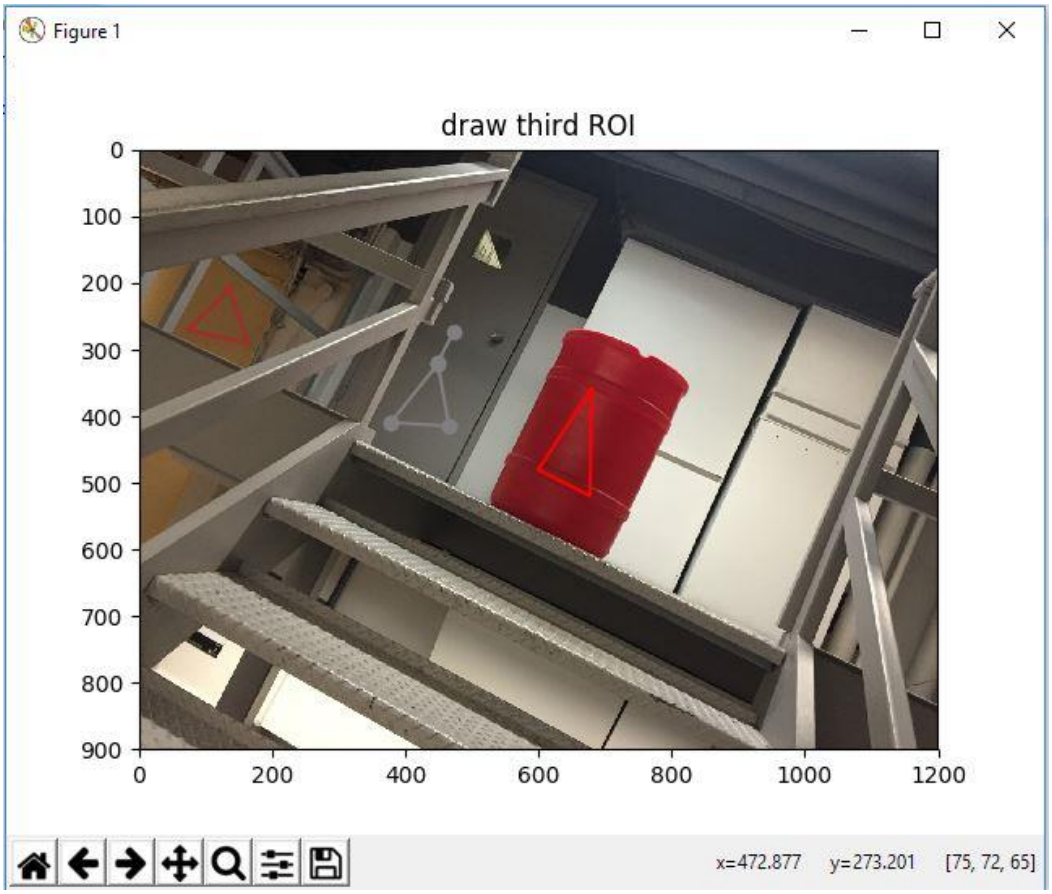
$$Slope\ (b) = \frac{n \times \Sigma xy - (\Sigma x) \times (\Sigma y)}{n \times \Sigma x^2 - (\Sigma x)^2}$$

$$Intercept\ (a) = \frac{\Sigma y - b(\Sigma x)}{n}$$

$$Regression\ (y) = a + bx$$

Technical Approach:

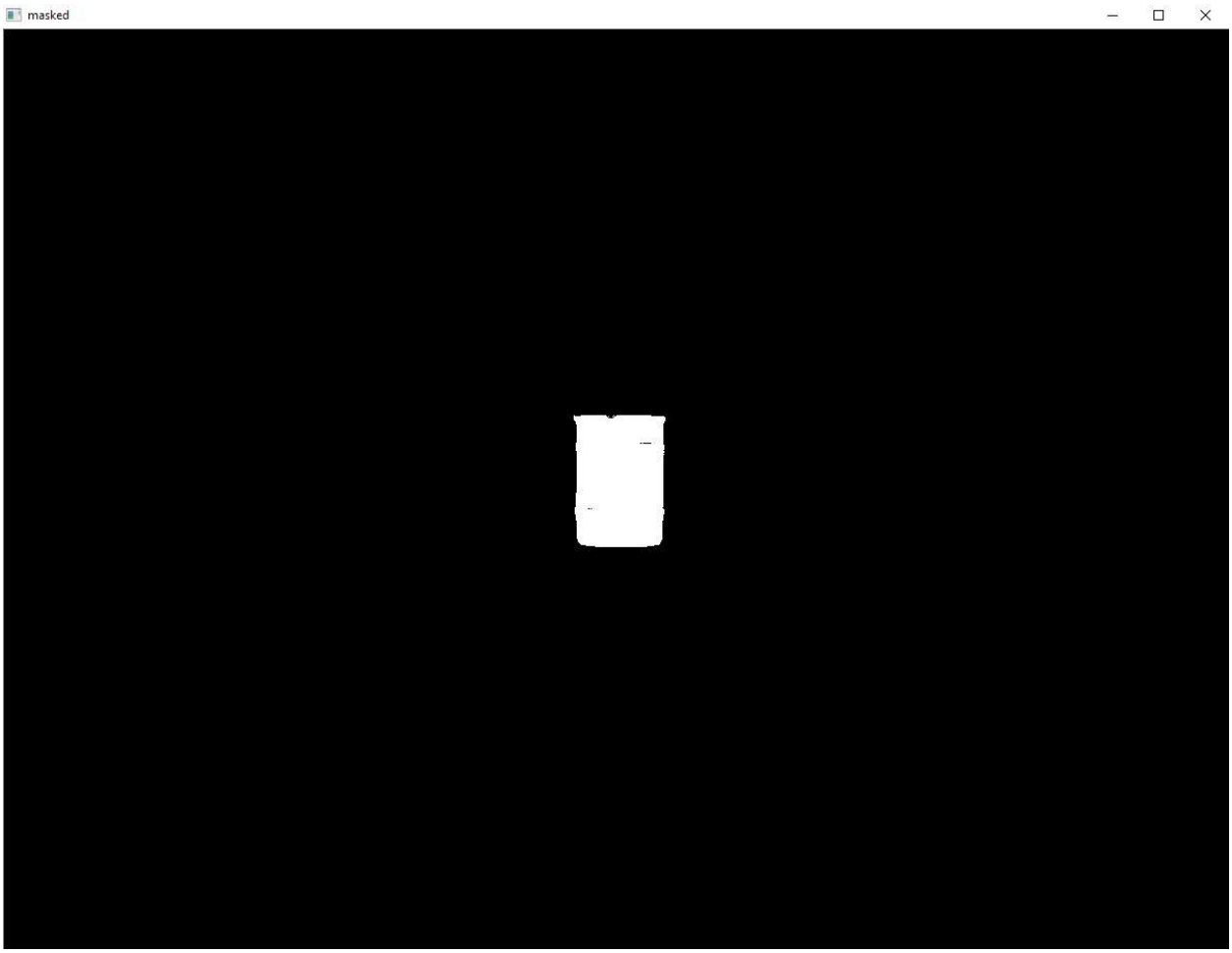
In order to get my pixel color data I used roipoly package to go through each image and hand label ROI for Red, Brown and Gray, like below –



Using this pixel data I then went through and calculated the corresponding mean, variance and covariance using the formulas listed above. Once I had these values I was ready to implement my BDR equation. I implemented the BDR equation as follows –

```
x = 1/np.dot(((2*3.14) ** (3/2)),(redPixelCovarianceDeterminant))
xa = (-1/2)*np.transpose(np.subtract(samplePixel,redPixelMean))
xb = redPixelCovarianceInverse
xc = np.subtract(samplePixel,redPixelMean)
xd = np.dot(np.dot(xa,xb),xc)
probabilityRed = x * xd
```

I did this once for each color, Red, Brown and Gray. Once I had the value for each color I chose the value that was the smallest, if it was Red I returned a 1, and if it was Brown or Gray I returned a 0. I went through each pixel in the image I was testing and sent that pixel to my BDR function, to return the corresponding color associated with that pixel. Once I was done checking every pixel in the image I created a mask out of the corresponding matrix of 0's and 1's. Using my test image below, I got the corresponding mask matrix –



From this mask matrix I used the OpenCV function findContours to create a list of all of the contours associated with this mask. If I had a list of the contours I could thus find where the barrel belongs in this list. By finding the contour with the highest “barrelness” score, I was then able to make a bounding box around this contour to get the coordinates of each edge, top left, top right, bottom left and bottom right. With these coordinates I could then draw this contour back onto the image and calculate the area of the bounding box.

By using this area I could then take it back to my Linear Regression function to calculate the barrel's distance –

```
global m
global b
area = [row[0] for row in barrelDistance]
distance = [row[1] for row in barrelDistance]
covariance = 0

meanArea = sum(area) / float(len(area))
meanDistance = sum(distance) / float(len(distance))
varianceArea = sum([(i - meanArea)**2 for i in area])
varianceDistance = sum([(i - meanDistance)**2 for i in distance])
for i in range(len(area)):
    covariance += (area[i] - meanArea) * (distance[i] - meanDistance)

m = covariance/varianceArea
b = meanDistance - m * meanArea
```

Plugging my area into y = m * area + b I could then retrieve the distance of the barrel. I repeated this process for every image in the test set to generate the 10 masks and bounding box images below.

Results:

Image 001 –

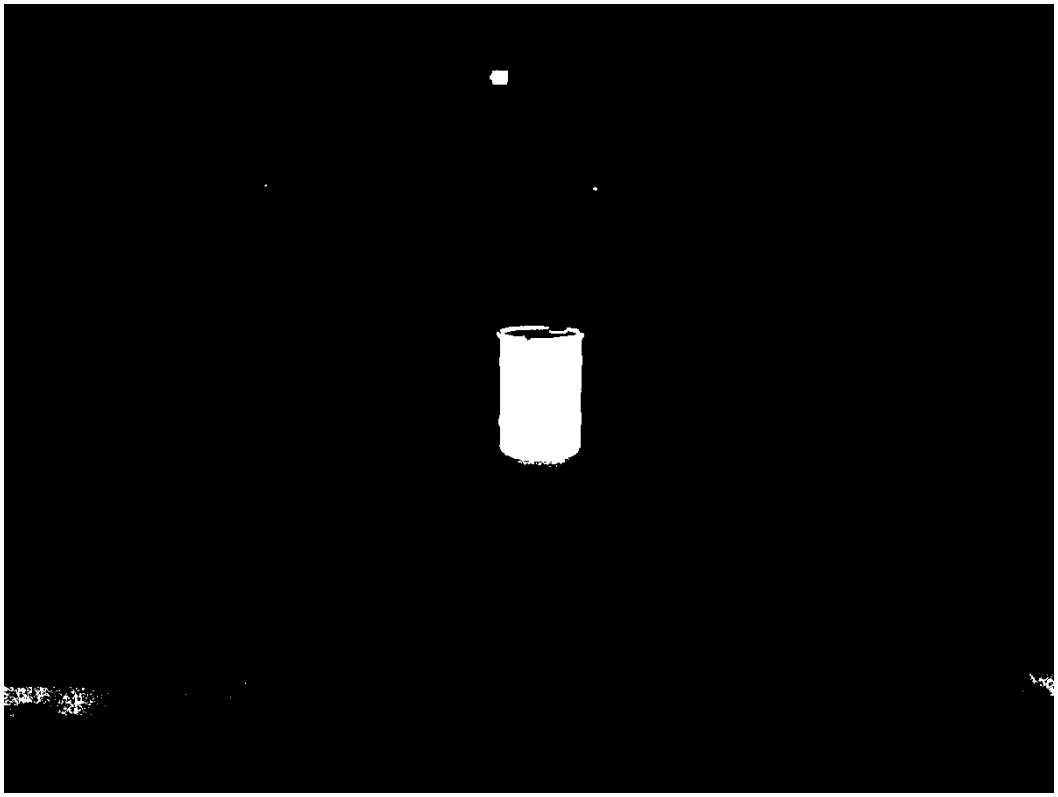
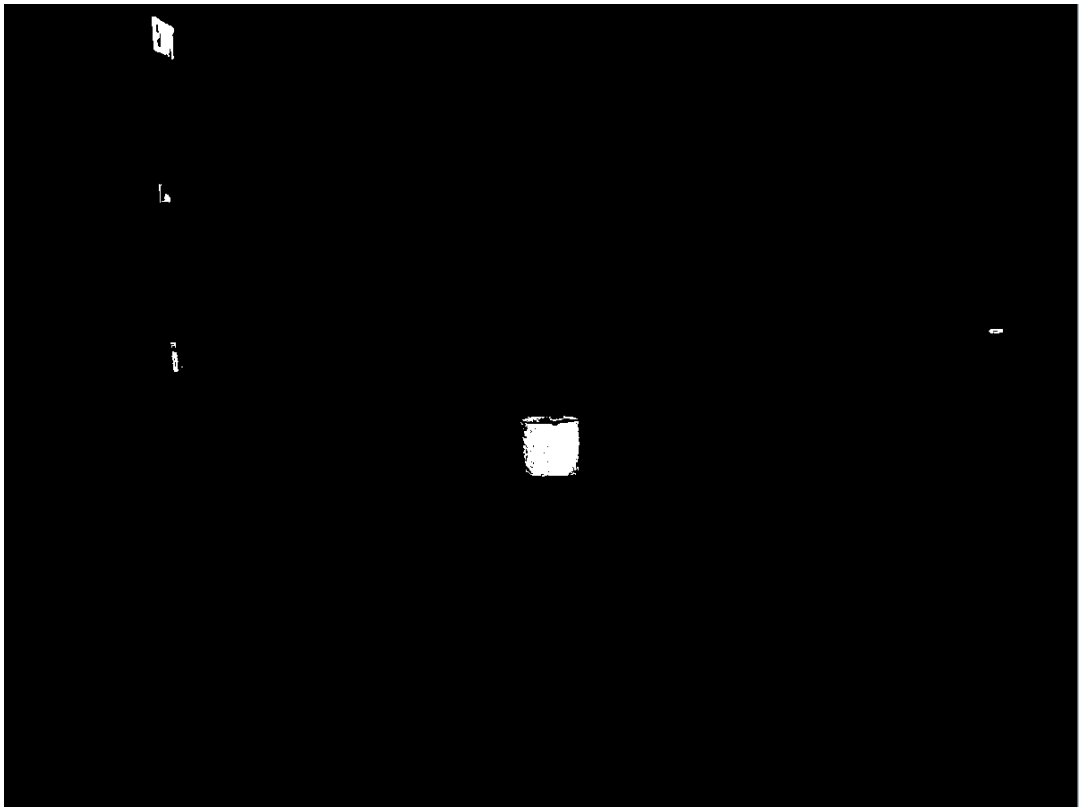


Image 002 –



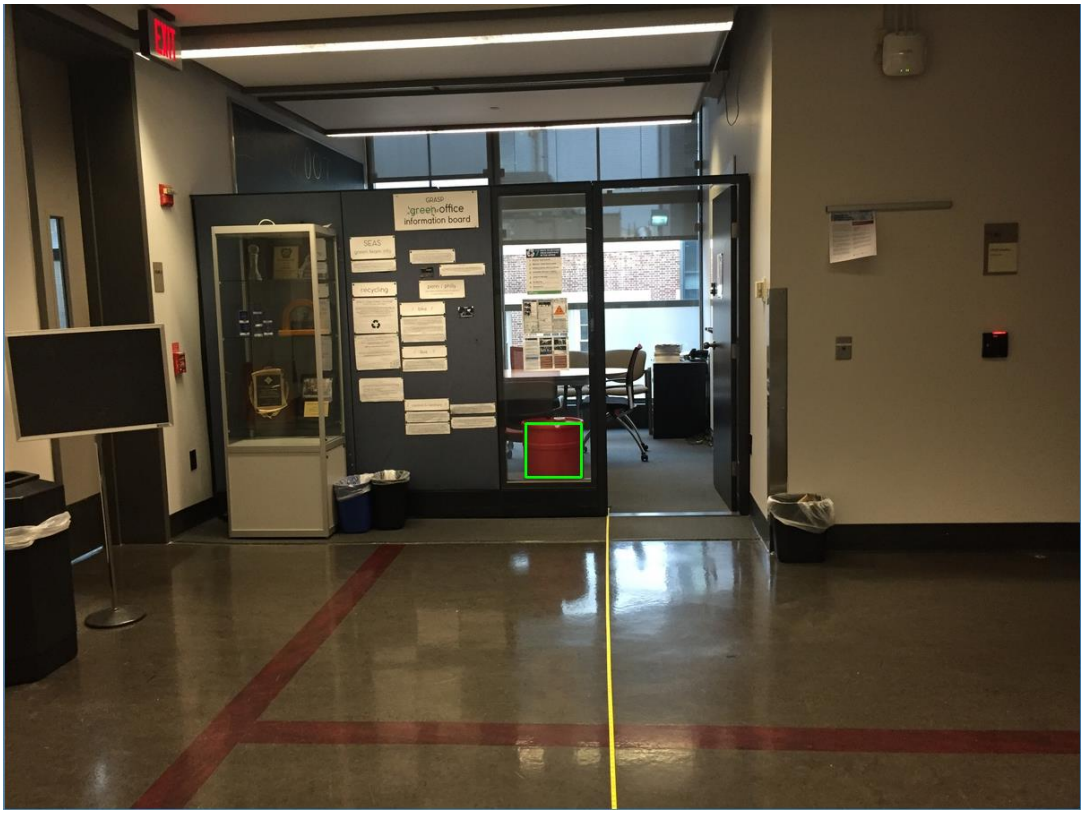


Image 003 –

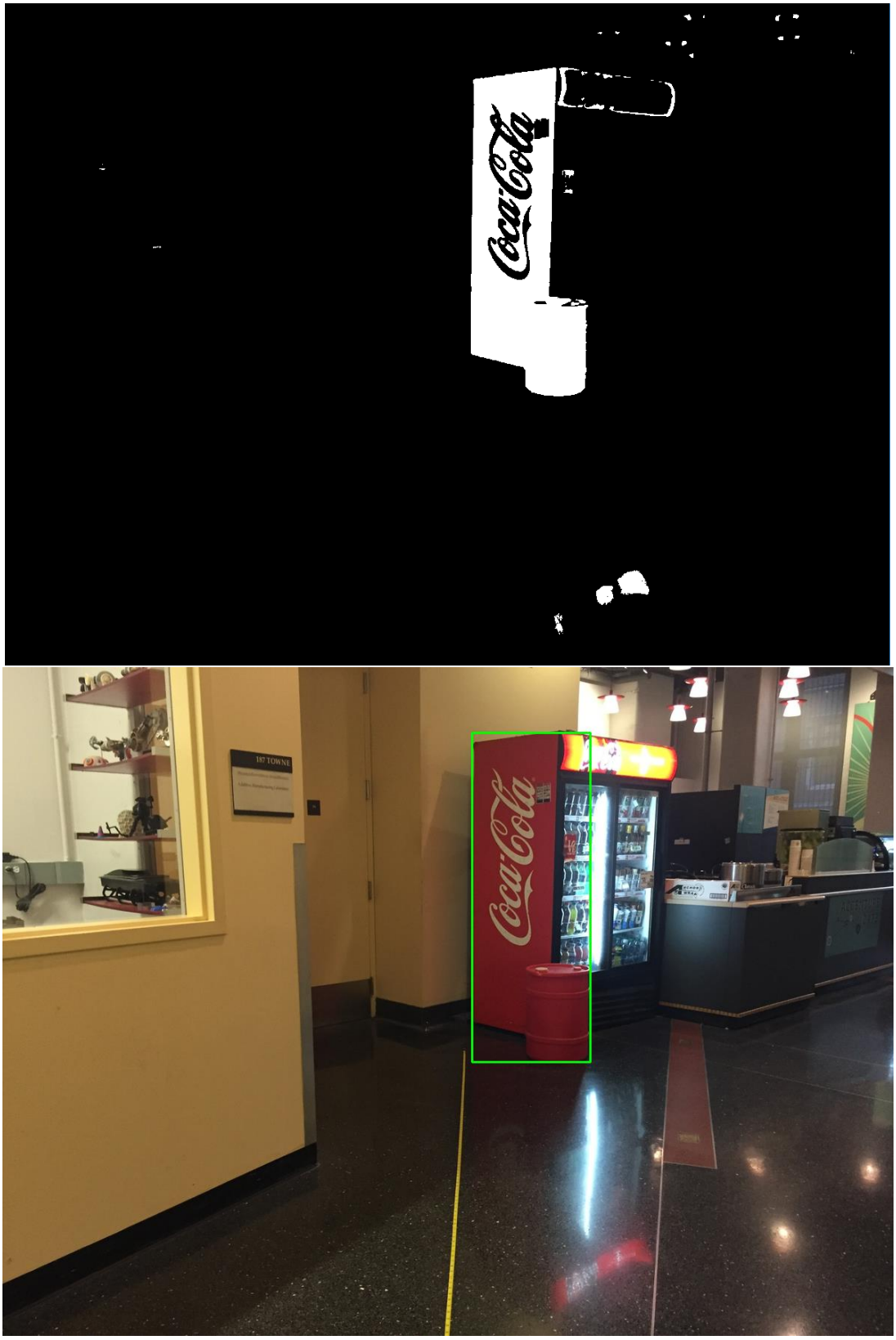
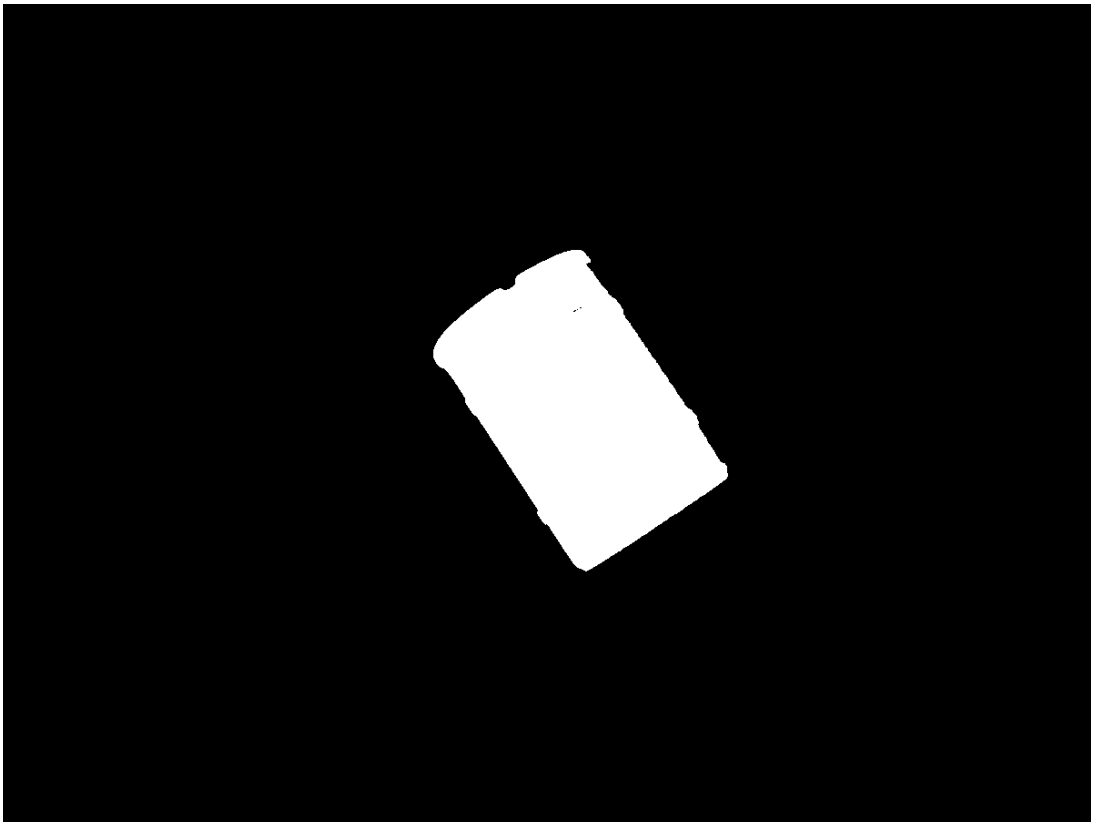


Image 004 –



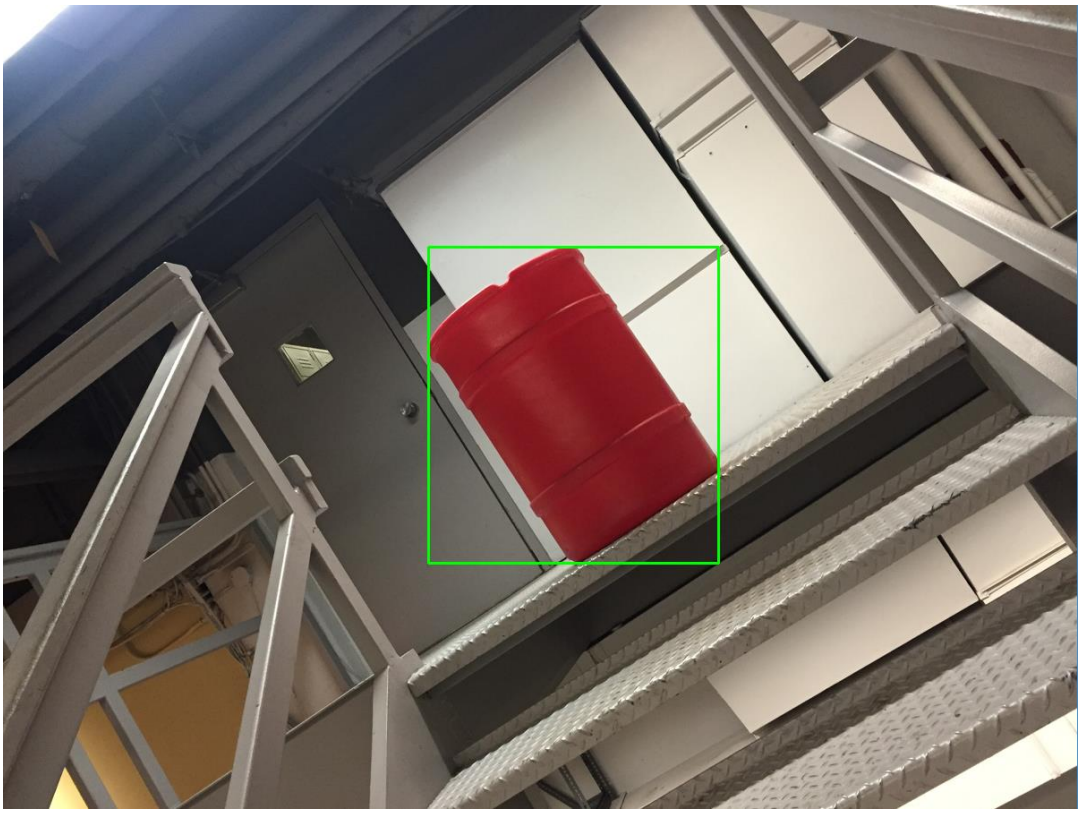


Image 005 –

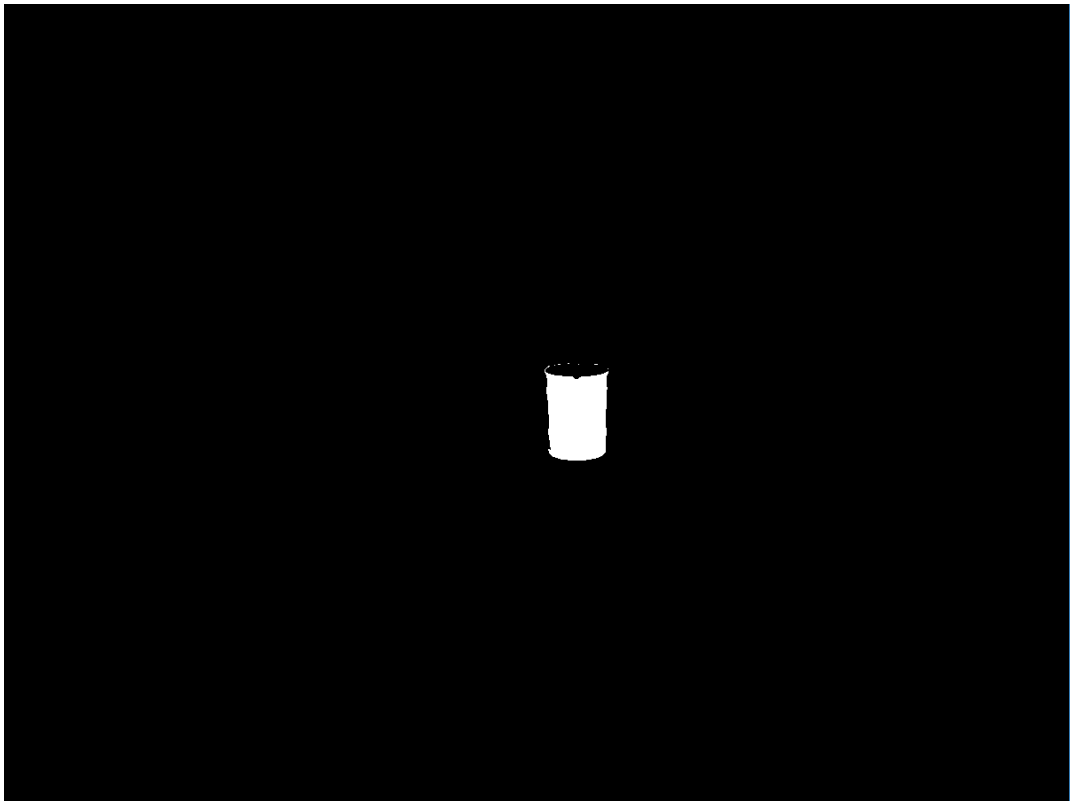
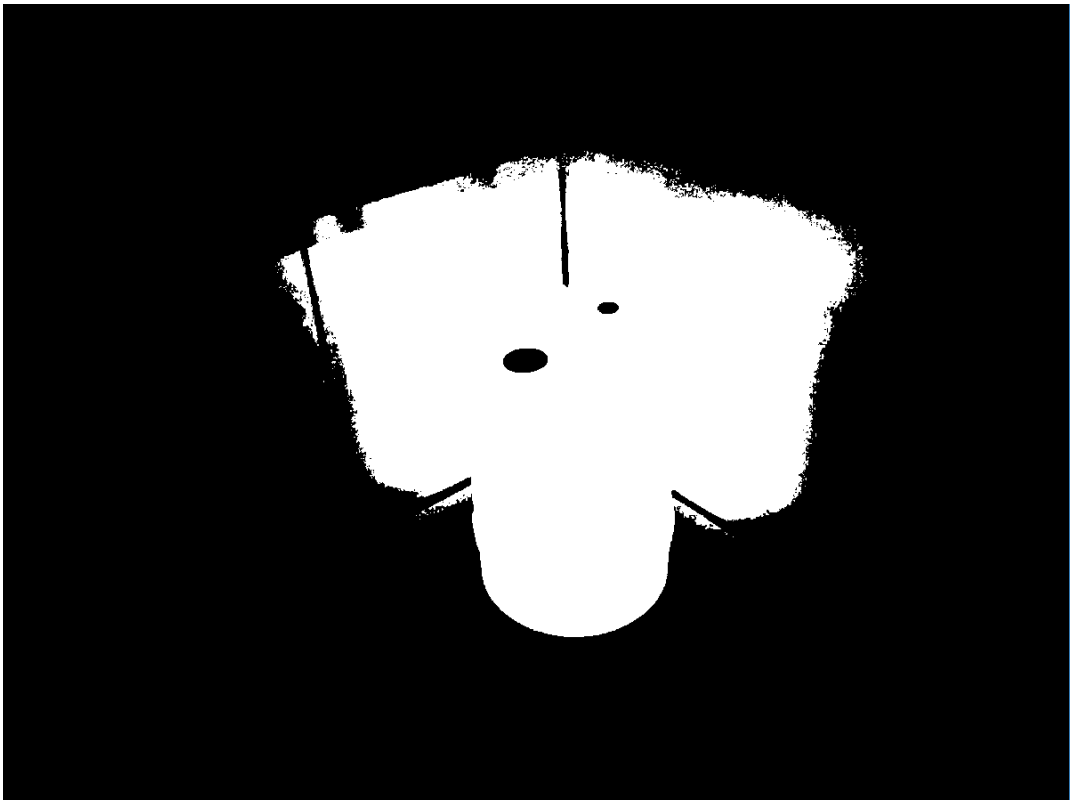


Image 006 –



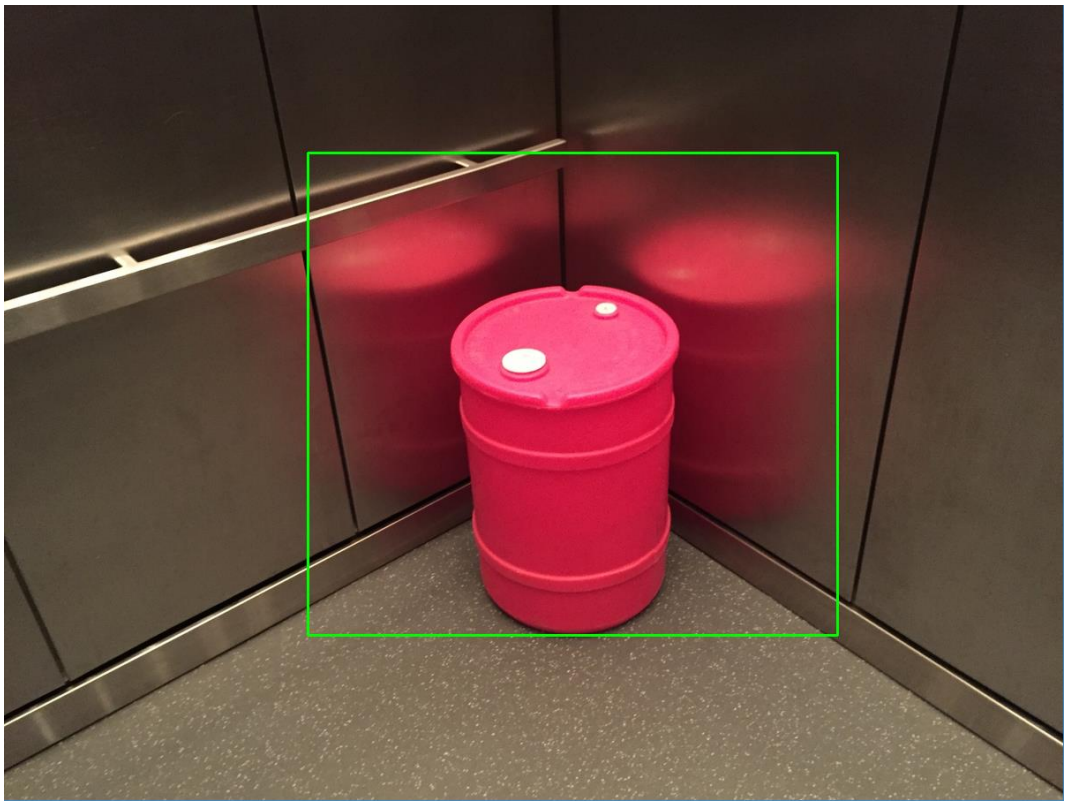
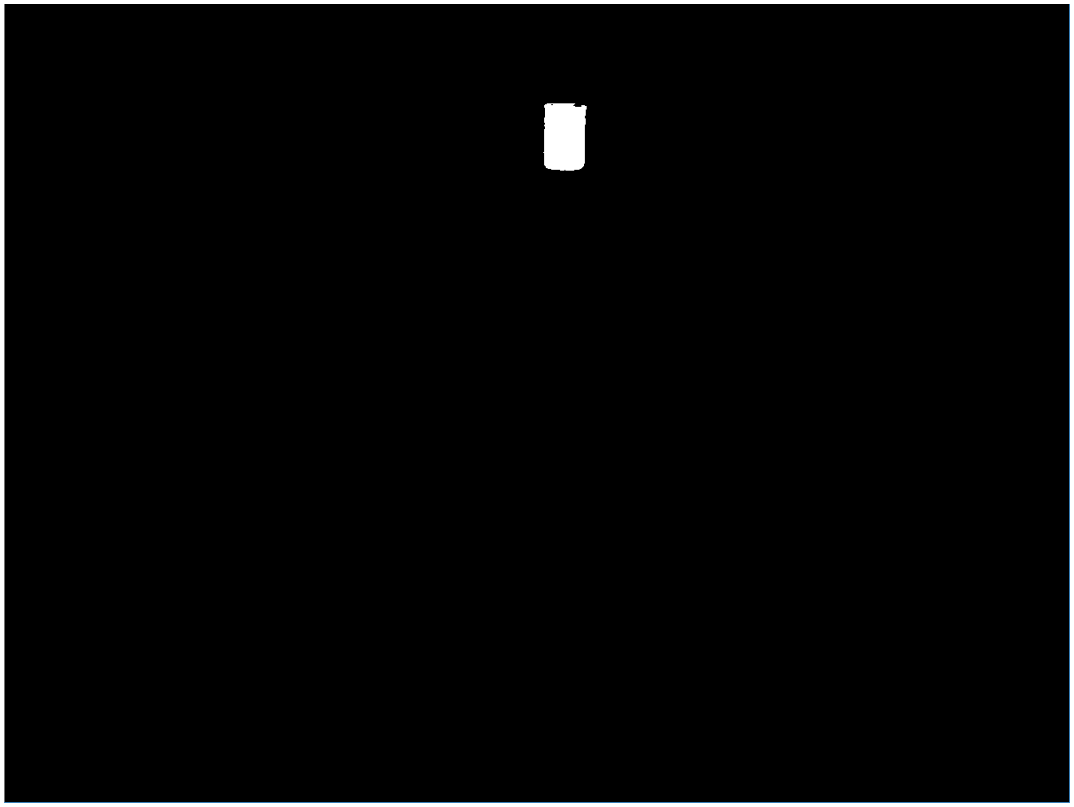


Image 007 –



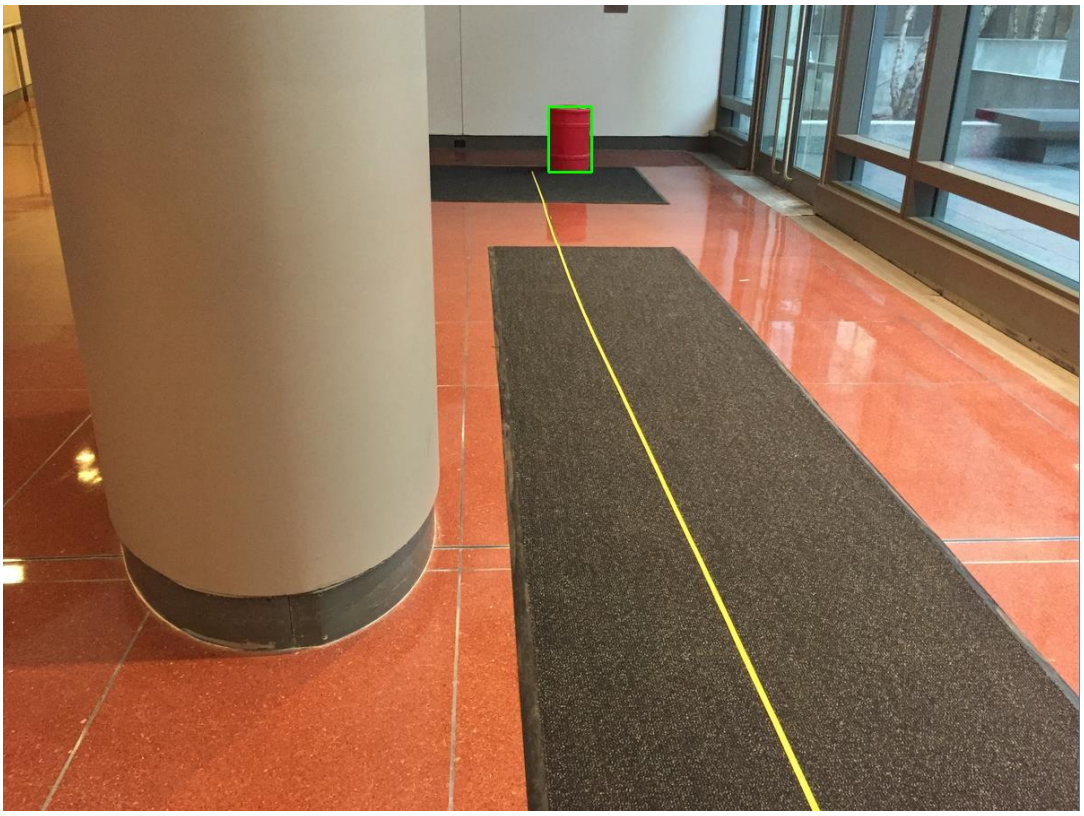
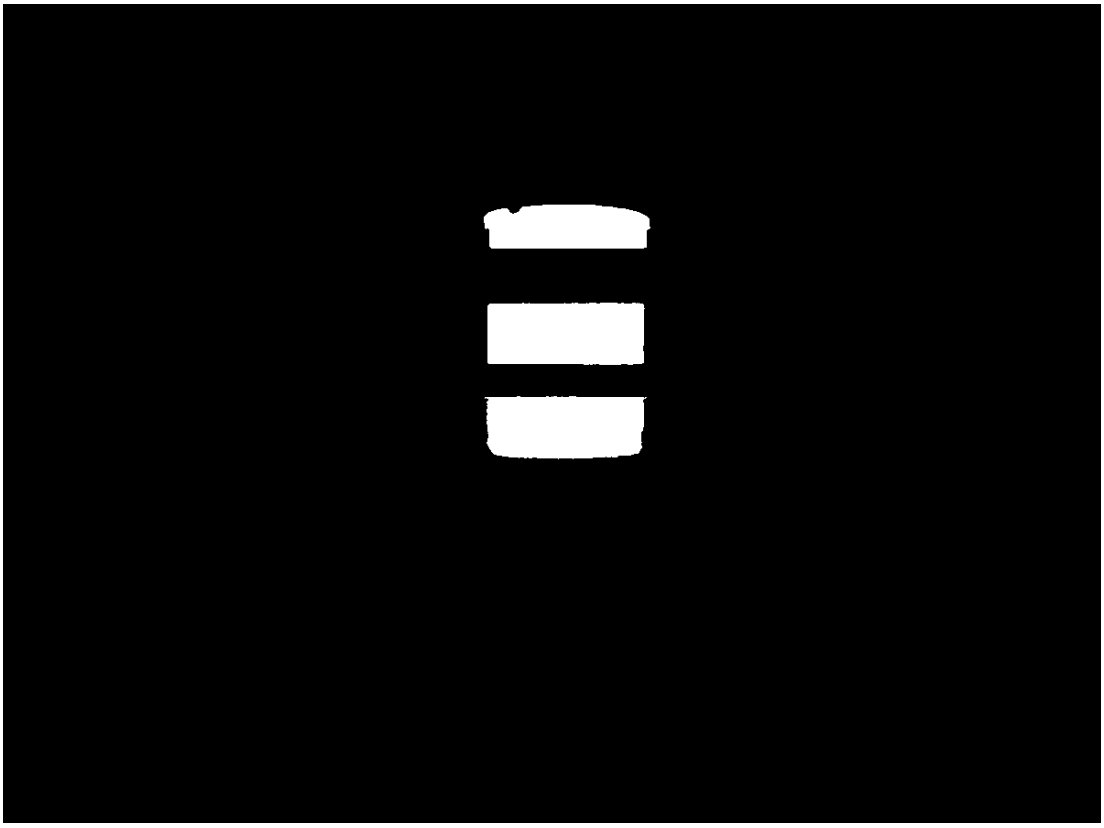


Image 008 –



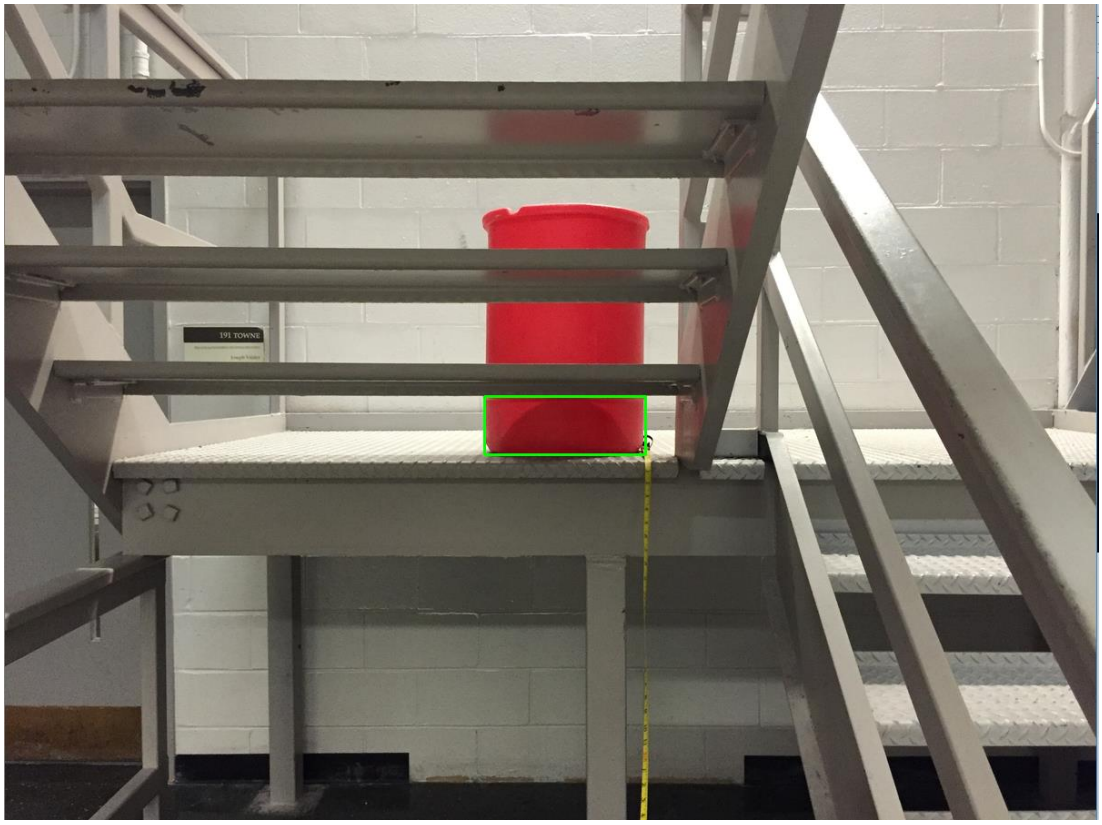


Image 009 –

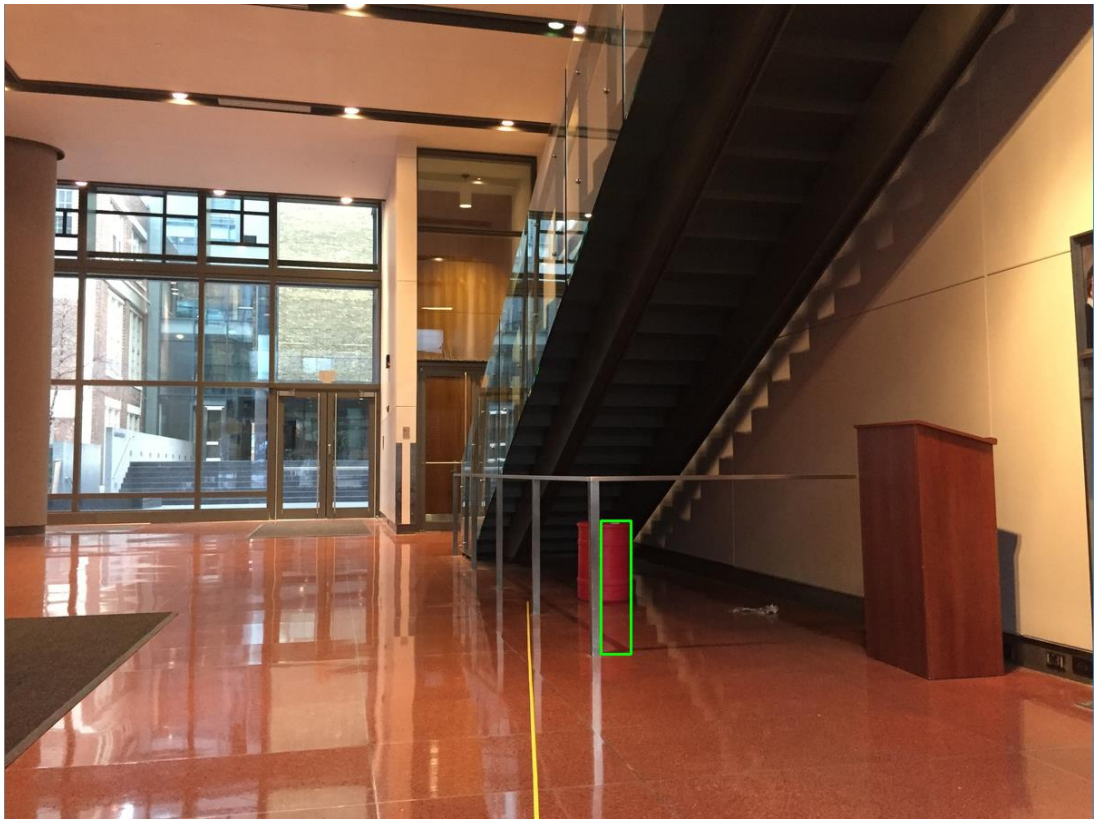
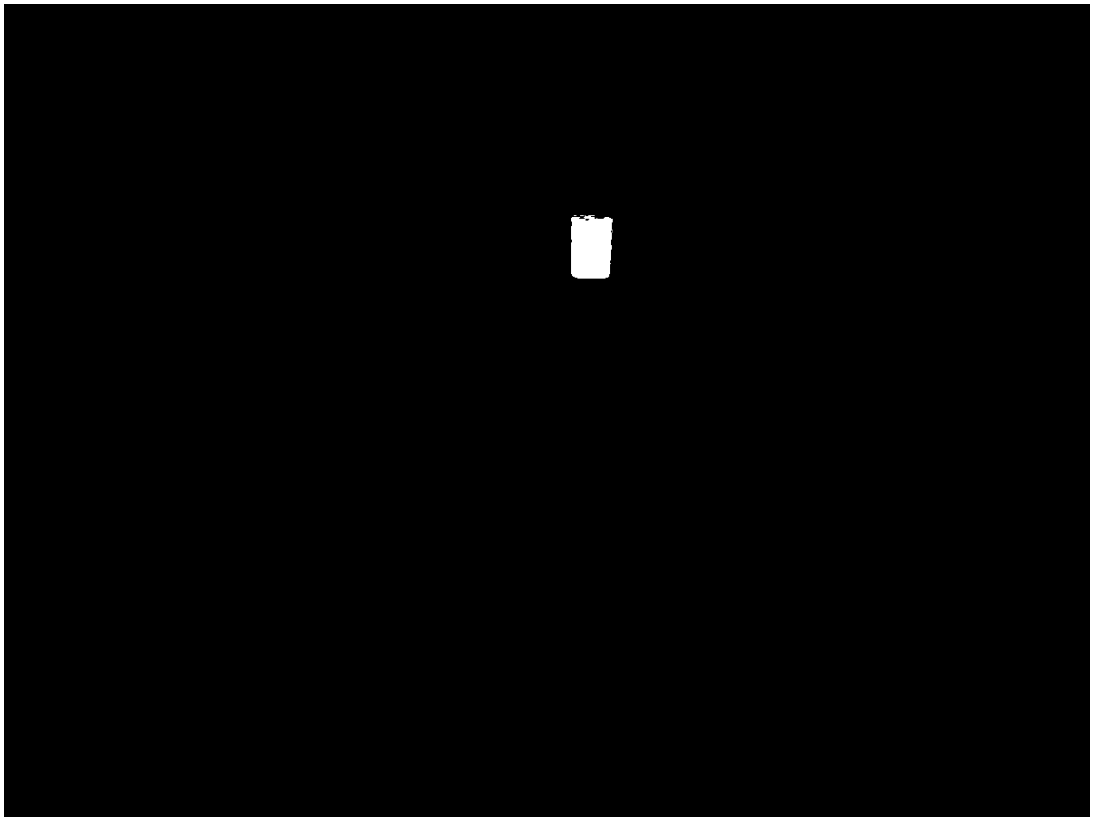
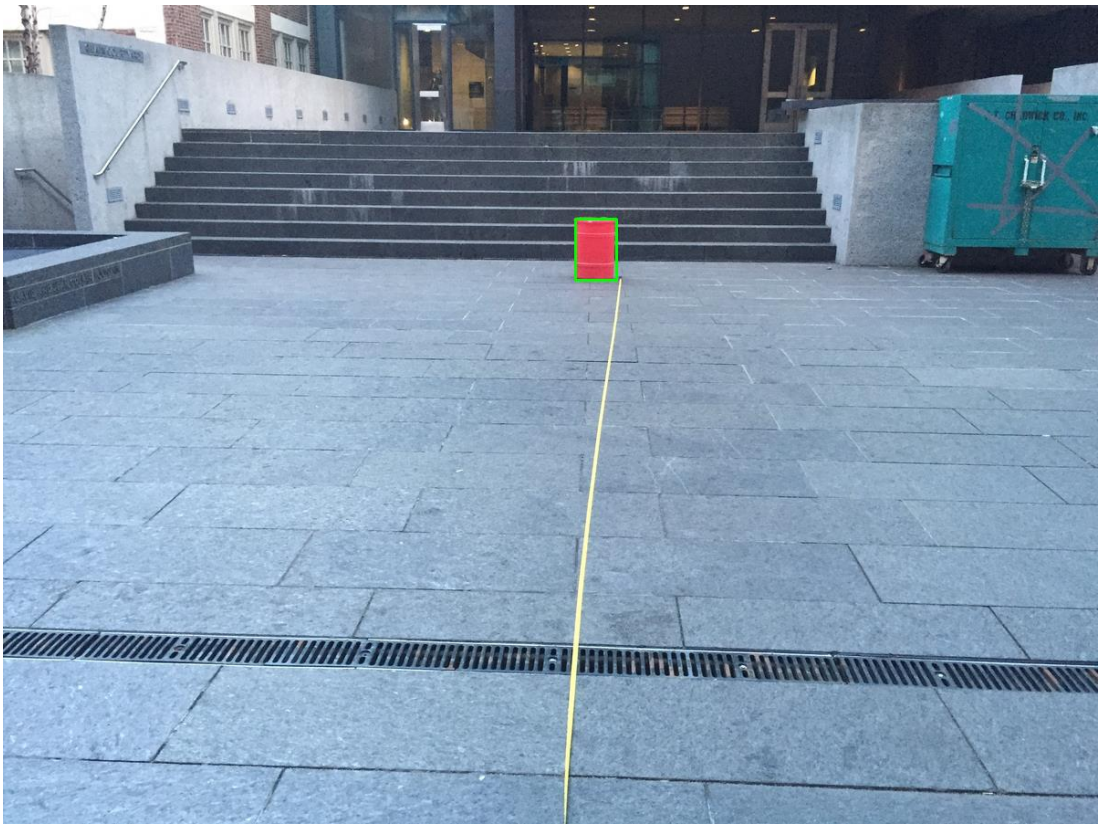


Image 010 –





With the corresponding outputs –

```
"ImageNo - ""001.jpg"", TopLeftX - "566", TopLeftY - "370", BottomRightX - "662", BottomRightY - "525", Distance - "6.159839860158966
"ImageNo - ""002.jpg"", TopLeftX - "581", TopLeftY - "469", BottomRightX - "643", BottomRightY - "529", Distance - "7.324451879180413
"ImageNo - ""003.jpg"", TopLeftX - "632", TopLeftY - "88", BottomRightX - "791", BottomRightY - "531", Distance - "0.36213825256267906
"ImageNo - ""004.jpg"", TopLeftX - "475", TopLeftY - "271", BottomRightX - "799", BottomRightY - "624", Distance - "0.734897583382421
"ImageNo - ""005.jpg"", TopLeftX - "613", TopLeftY - "408", BottomRightX - "681", BottomRightY - "510", Distance - "6.9888432543441255
"ImageNo - ""006.jpg"", TopLeftX - "344", TopLeftY - "167", BottomRightX - "944", BottomRightY - "713", Distance - "-26.474342092205468
"ImageNo - ""007.jpg"", TopLeftX - "608", TopLeftY - "113", BottomRightX - "656", BottomRightY - "186", Distance - "7.346992756967925
"ImageNo - ""008.jpg"", TopLeftX - "527", TopLeftY - "432", BottomRightX - "704", BottomRightY - "496", Distance - "2.277799796755321
"ImageNo - ""009.jpg"", TopLeftX - "657", TopLeftY - "570", BottomRightX - "690", BottomRightY - "717", Distance - "7.206425338543024
"ImageNo - ""010.jpg"", TopLeftX - "627", TopLeftY - "236", BottomRightX - "672", BottomRightY - "303", Distance - "7.398022799736875
```

Images 1, 2, 4, 5, 7, and 10 did very well. Whereas images 3, 6, 8 and 9 did not do very well for various reasons.

Image 3 did not do well because of the coke machine looking like a barrel in the mask that was generated.

Image 6 did not do well because perhaps my pixel data was not trained well enough because it picked up the reflections as part of the red barrel.

Images 8 and 9 did not do well because they were obstructed by something, which resulted in the contour to be separated into multiple sections which did not bound the whole barrel when generating the bounding box associated with these contours.

I noticed some issues with the distance as well. In particular the distance for Image 6 resulted in a negative value, I believe this was due to the area of the bounding box being much larger than ones that we had in testing. I also noticed that the distance was off by a meter or two sometimes in extreme cases. When training my Linear Regression I noticed that despite the distance of the images always increasing in meters, sometimes the area of the barrel in the image would be larger or smaller than the one before depending on the orientation. I think this causes some error in calculating the distance due to the area and distance not being as linear as they should be to generate a strong Linear Regression.

Overall my program did very well though, the masks generated were very good in most cases, ignoring most background noise. The bounding box that I generated fit very well to the barrel as well in cases where it was not obstructed. And the distance was fairly close in most cases, and was off only when the bounding box generated was larger or smaller than the actual barrel itself.