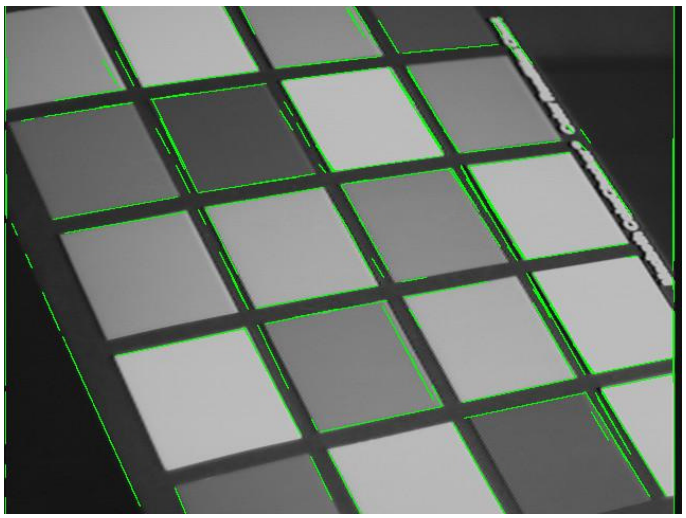
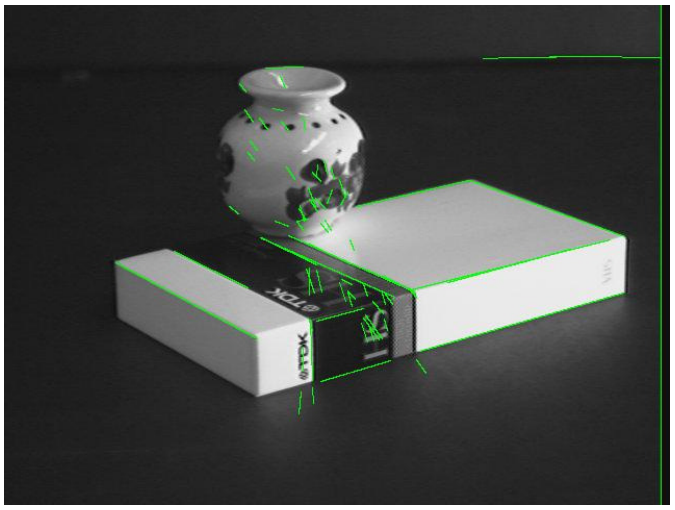
Shastri Ram
Computer Vision: Project Hough Transformation

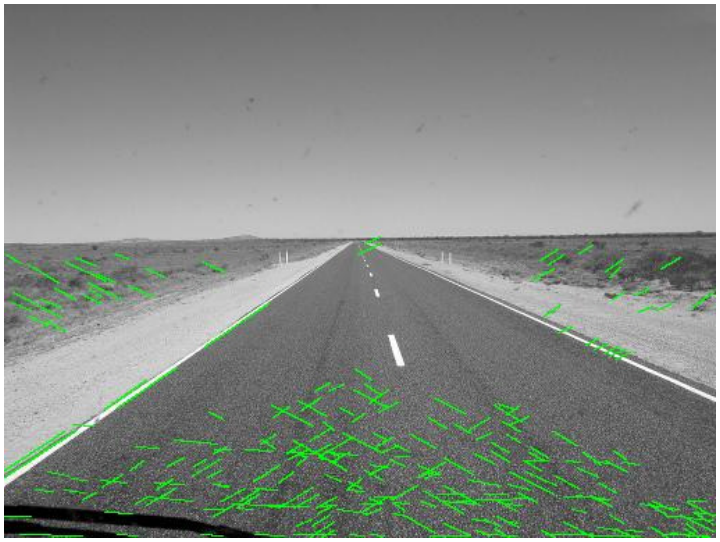
Experiments

The following image are the result of the initial run of houghScript.m



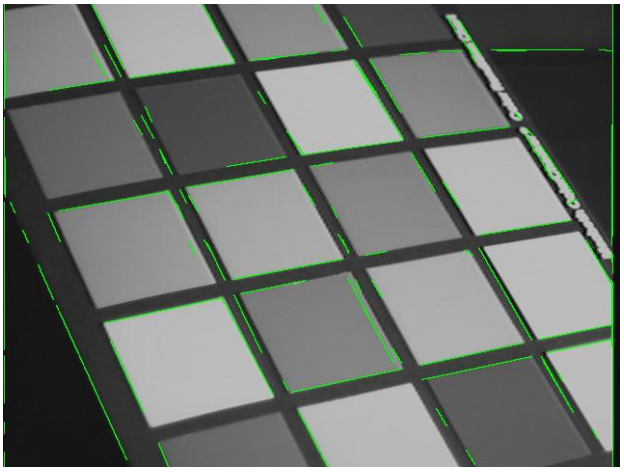


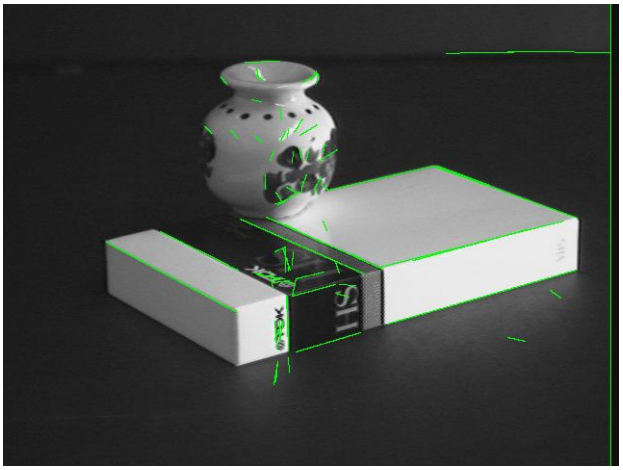

HEINZ
HEINZ TOMATO KETCHUP
24 GIANT SIZE PKG.
New!
Brillo
soap pads
SHINES ALUMINUM
Campbell's
TOMATO JUICE
362
FREESTONE
PEACH HALVES



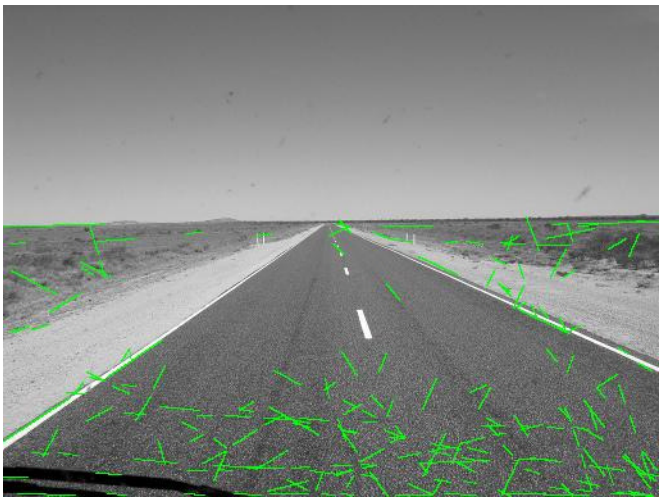
A couple of things can be noticed from these images. Firstly, there are some images which have multiple lines on top of each other like image 1 (from top left). This suggests that even after NMS, there are some peaks that are close to each other. Secondly, in the majority of images, there are many small lines all over the images like in images 3-8. This suggests that the parameters need to be tweaked some more. Lastly, some of the obvious lines in some images like images 2-4 are not showing up. Again this suggests some parameter adjustment such as the number of lines, or the resolution.

First, I was attempt to remove the duplicate lines by using the imdilate function with `strel('ball',5,5);` in MATLAB and then running the NMS algorithm. The results have less small line segments and more actual lines that represent the image. Additionally, the duplicate lines seem to have been mostly removed. The results are shown below.

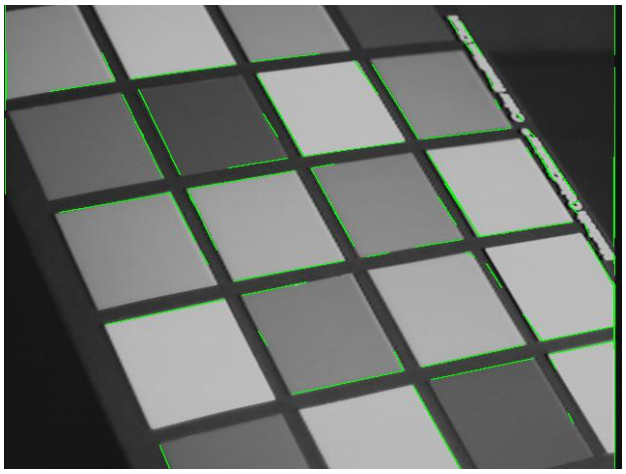


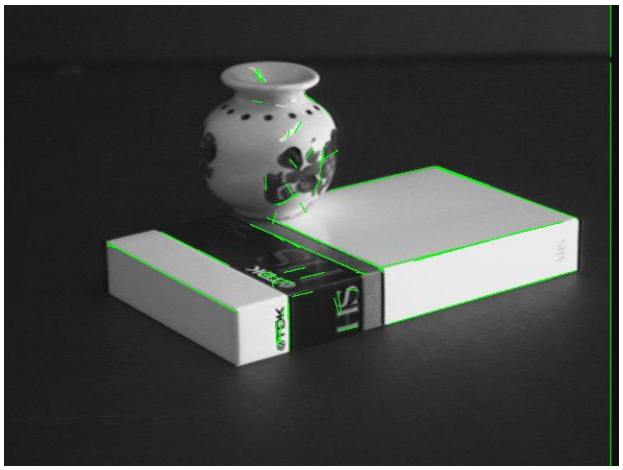


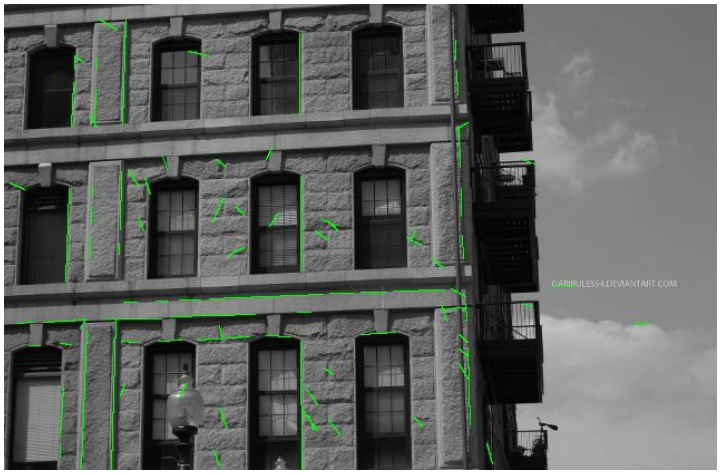

HEINZ
24 GIANT SIZE PKG
New!
Brillo
soap pads
Campbell's
TOMATO JUICE
FREESTONE
PEACH HALVES
1069



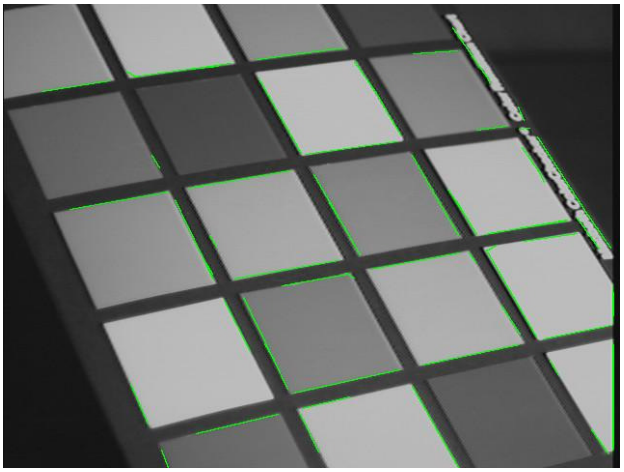
The next aim is to get rid of the small line segments. I will first adjust the threshold to get rid of the smaller line magnitude values. A value of 0.1 appears to filter out a lot of the unwanted line small line segments. The results are shown below.

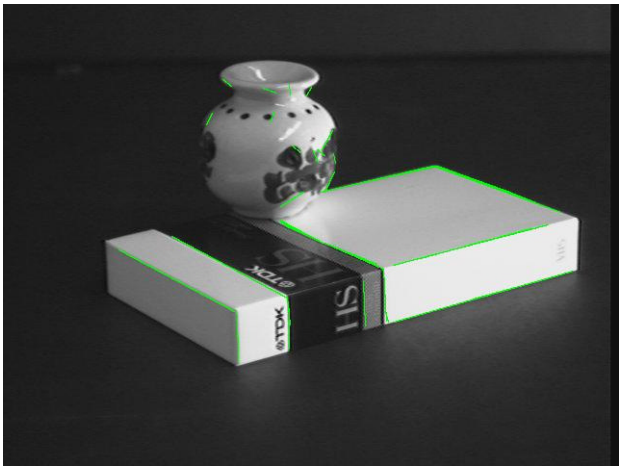






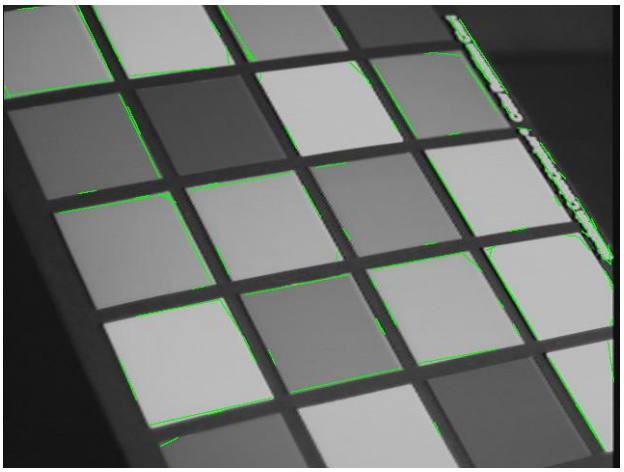
It can be seen that adjusting the threshold eradicated a lot of the noisy lines, that is, the small line segments. However in some of the images, for instance, the images of the buildings there is still a lot of noise. The Gaussian filter applied at the beginning is supposed to help suppress some of the noise. The initial value of sigma was 2. This value should be increased to get a larger averaging effect of the filter. After experimentation, a sigma of 5 was found to produce good results as shown below. A lot of the noise was reduced and the edges highlighted in green corresponded well to the actual edges in the image.



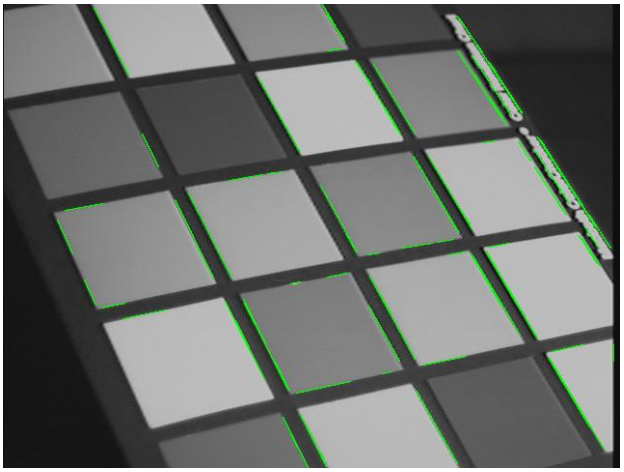


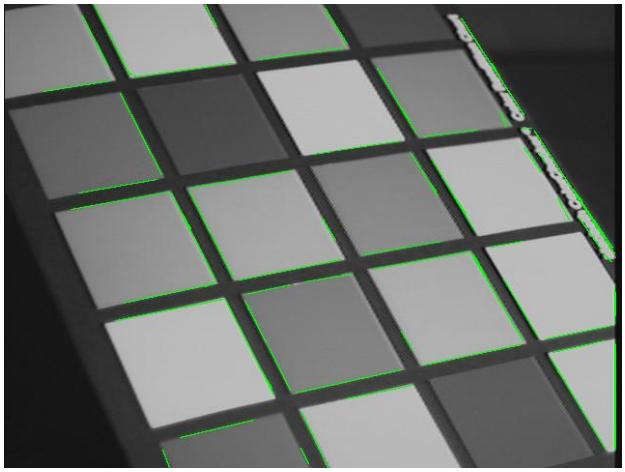

57 HEINZ 57
5124
HEINZ TOMATO KETCHU
24 GIANT SIZE PKG
New!
Brillo
soap pads
SHINES ALUMINUM FAS
Campbell's
36-17½ OZ. CANS
362
TOMATO JUICE
Del Monte
24 No. 2½ CANS
FREESTONE
PEACH HALVES
1069

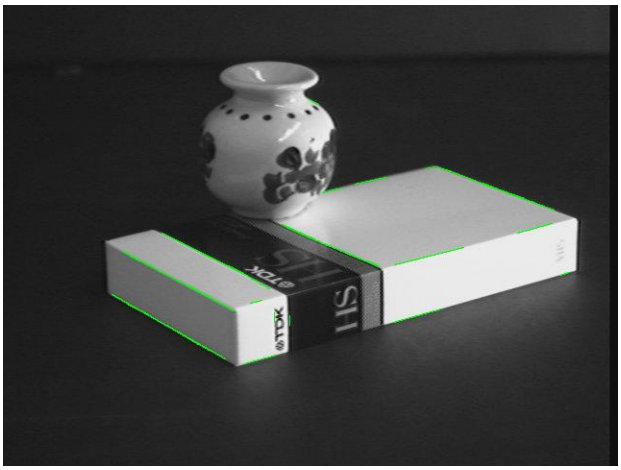
These results seem to indicate the edges well. However I would like to show more edges, so the next step is to adjust the resolution of rho and theta. While keeping the previous parameters constant, I adjusted the resolution of rho and theta. Changing the rho resolution from its original value of 2 seemed to have the effect of producing a lot of smaller line segments. This was good for the images which had a lot of small edges like the image with the cuboid boxes, but bad for the images which had more clearly defined edges like the image with a lot of squares. These two images are shown below.

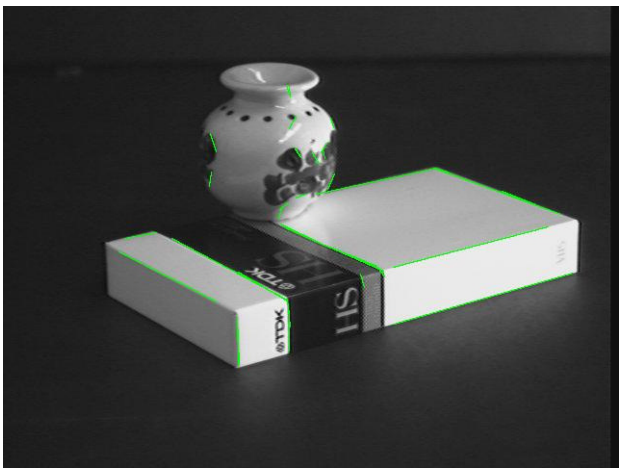


It is interesting to note that too high a resolution of rho and theta caused the algorithm to produce lower quality results. This is probably because since there were more buckets to vote on, there were smaller line segments which would have been filtered out from the thresholding and NMS and hence less edges would be produced in the output. Below shows a comparison of two sets of images with different resolutions for rho and theta. On the left the images have a rho resolution of 0.5 and a theta resolution of pi/360. On the right the images have a rho resolution of 2 and a theta resolution of 180.






TDK
HS


TDK
HS


GABIRULES4.DEVIANTART.COM

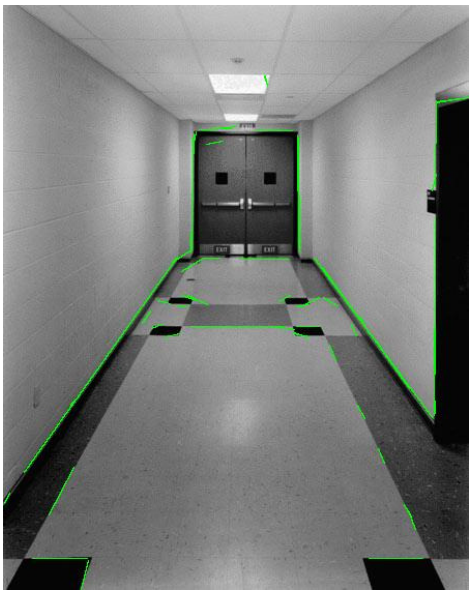

GABIRULES4.DEVIANTART.COM


www.dexigner.com


www.dexigner.com

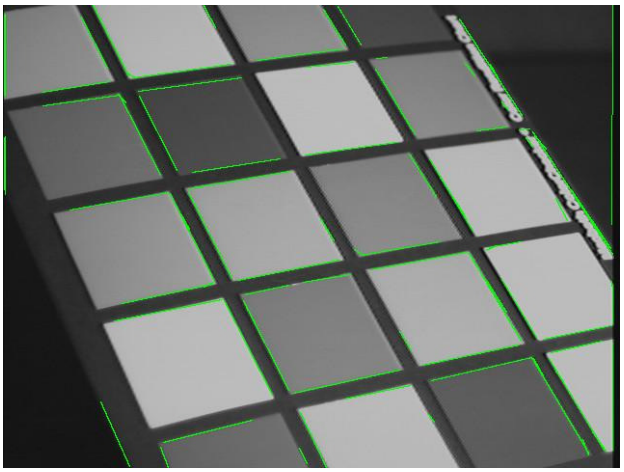

57 HEINZ 57
5124
HEINZ TOMATO KETCHU
24 GIANT SIZE PKG
New!
Brillo
soap pads
SHINES ALUMINUM FAS
Campbell's
36-17½ OZ. CANS
362
TOMATO JUICE
TOMATO JUICE
DelMonte
24 No. 2½ CANS
FREESTONE
PEACH HALVES
1069

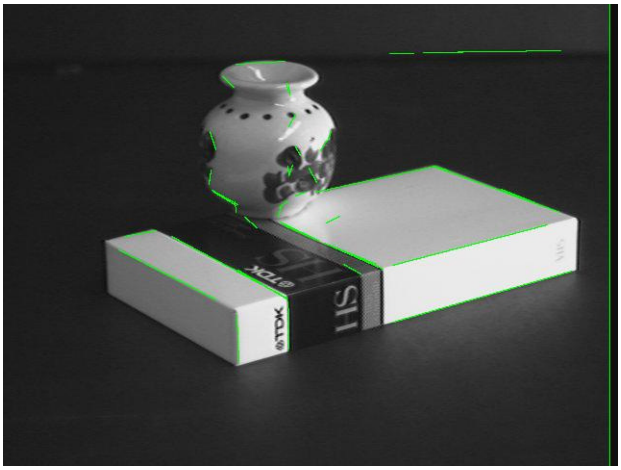

57 HEINZ 57
5124
HEINZ TOMATO KETCHU
24 GIANT SIZE PKG
New!
Brillo
soap pads
SHINES ALUMINUM FAS
Campbell's
36-17½ OZ. CANS
362
TOMATO JUICE
TOMATO JUICE
DelMonte
24 No. 2½ CANS
FREESTONE
PEACH HALVES
1069



From the results, it can be seen that the right column yields better results.

After tuning these parameters, I was quite satisfied with the result. However I thought back to PID controls tuning, and one of the techniques taught is that after you tune the parameters, it is worth going back and re-tuning. I decided to see what would happen if I reduced the threshold once again to 0.03 while keeping the other parameters constant. The results are shown below. While there seems to be the introduction of smaller, noisy line segments, I would say the output looks quite good because it highlights edges that a human would easily identify.




TDK
SII


GABIRULES54.DEVIANTART.COM


www.dexigner.com


The Worlds Largest Selling Ketchup
57 HEINZ 57
5124
HEINZ TOMATO KETCHU
24 GIANT SIZE
New!
Brillo
soap pads
SHINES ALUMINUM
TOMATO JUICE
Campbell's
36-17½ OZ. CANS
362
TOMATO JUICE
Del Monte
24 No. 2½ CANS
FREESTONE
PEACH HALVES
1069

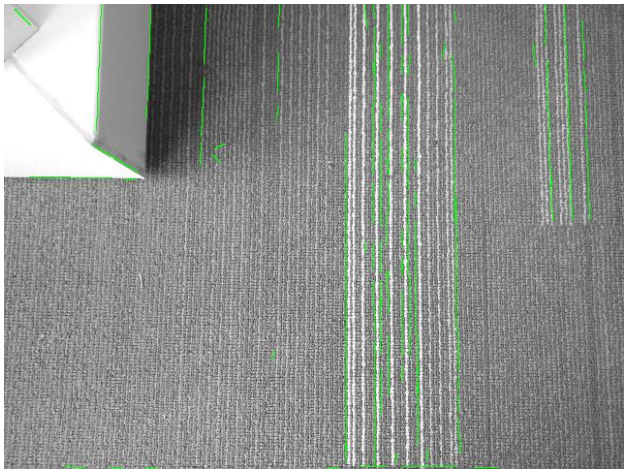
Thus, after experimentation, I would say the optimal values for the images are sigma =5, threshold = 0.03, rhoRes = 2, thetaRes = pi/180. These parameters seem to be optimal for these set of images. However as noted before, the higher resolution seems to be better for the images which had a higher number of smaller edges. The lower resolution seemed to be better for the images which had a higher number of long edges. However, highly textured images such as the one of the building with the stone brick façade and the desert scene would produce a large number of small edges and a lower resolution is preferred here than a higher resolution.

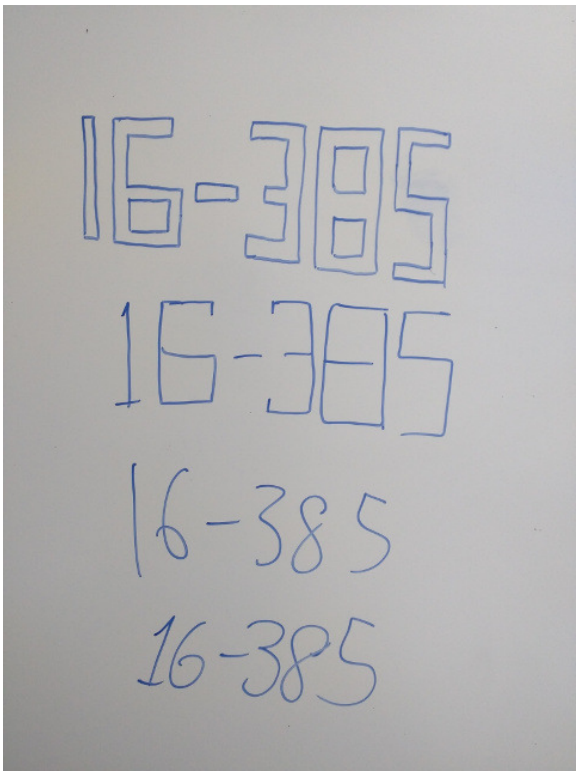
Though the code worked well with these single set of parameters, it is possible to finely tune the parameters to get even better output for each image. However, the downside is that the output for the other images may be terrible. Thus for a general edge detector, the chosen parameters would work. For more specific instances, it is up to the programmer to choose the parameters that would maximize the output for the particular application.

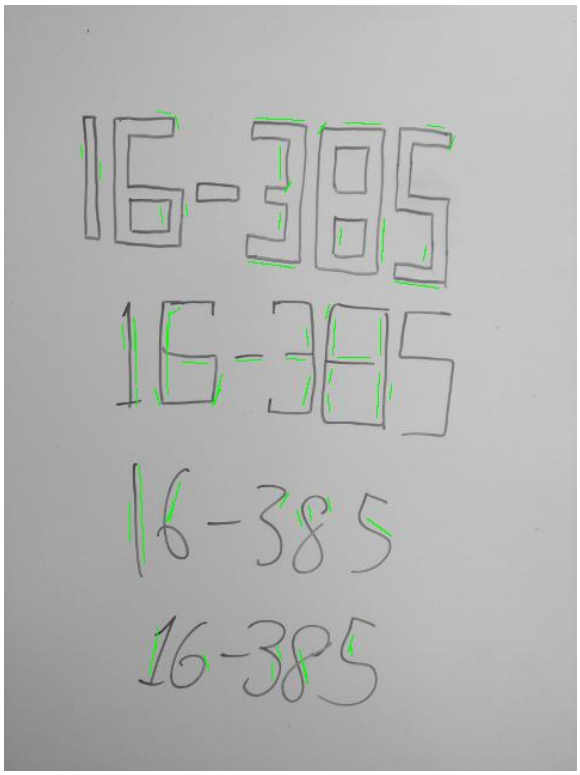
The step of the algorithm that caused the most problems was the myEdgeFilter. In particular, when performing the NMS, finding which pair of pixels to choose depending on the angle of the middle pixel was tricky. At first I incorrectly thought of the coordinate frame as the traditional Cartesian coordinate frame and I was getting output that looked good for the horizontal and vertical lines and bad for the angled lines. I initially thought that the y axis extended upwards, however, I realized instead that the origin is at the top left and the x-axis extends to the right and the y-axis to the bottom. It changed the choice of pixels depending on the angle and that solved the problem.

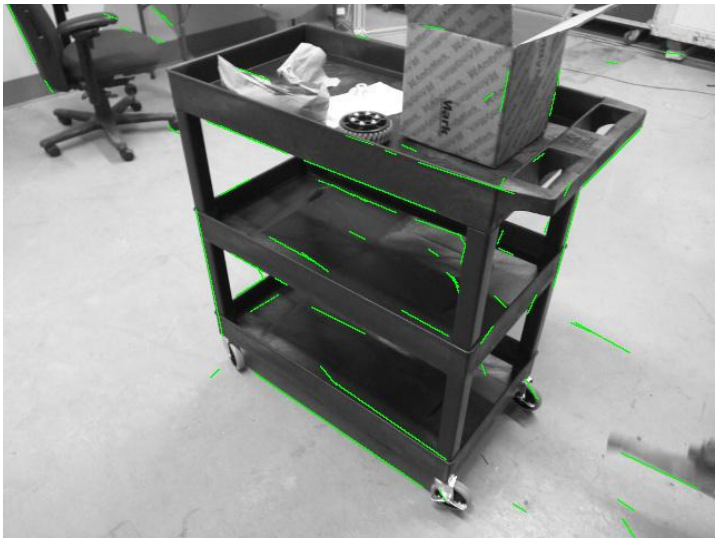
My Own Images!

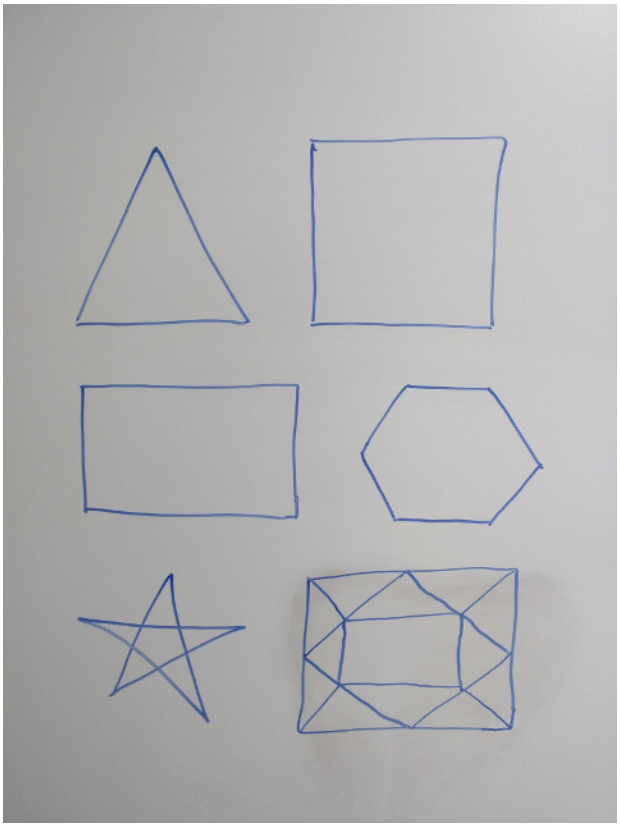
Though the write-up asked to use 5 images, I was quite interested to see how my code would deal with hand writing and specific shapes so I included one extra image. Below shows my images along with their results. The parameters used to produce this output were sigma = 5, threshold = 0.03, rhoRes = 2, thetaRes = pi/180 and nLines = 50.

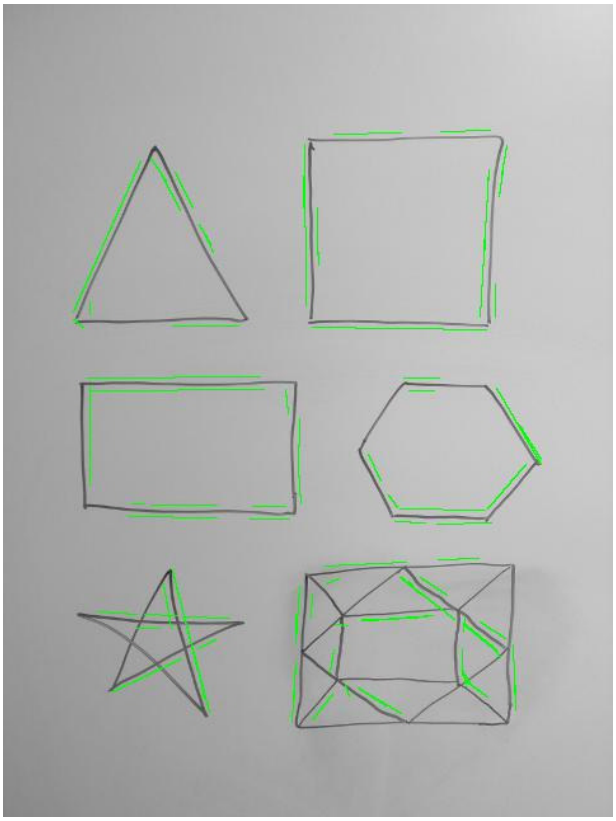



16-385
16-385
16-385
16-385

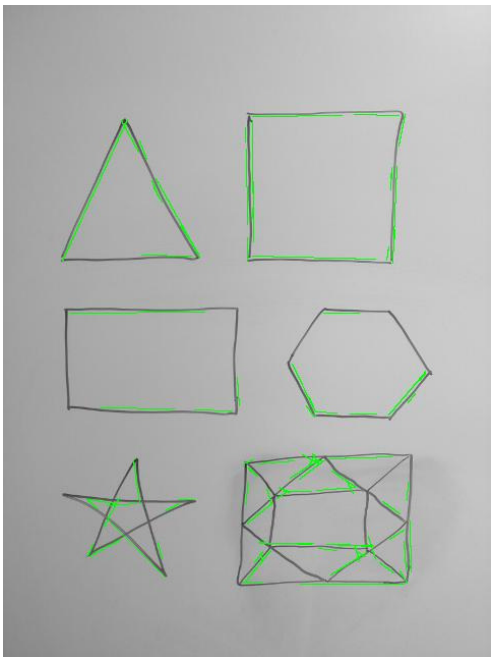

16-385
16-385
16-385
16-385

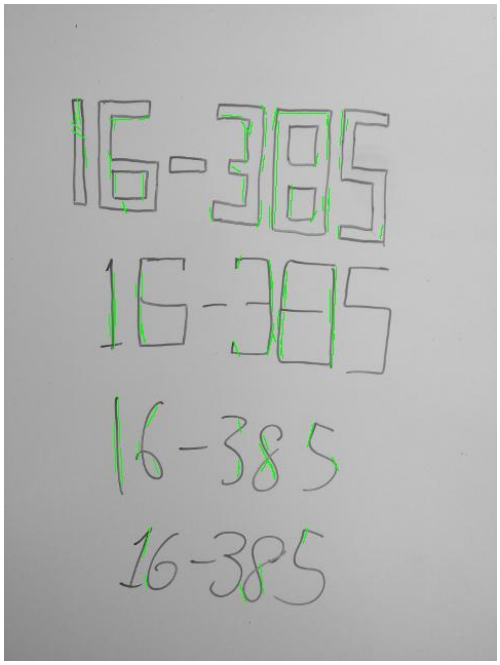






The output for these images seem pretty reasonable given the values of the parameters. They seem to follow a certain pattern similar to the test data given for the project. Of particular interest is the handwritten image of 16-385 and the drawn shape. Their edges are around the actual location of the edge. I hypothesize that this is due to the Gaussian filter being applied initially. Since these images are so clean, a Gaussian with a small sigma needs to be applied. As such the images below shows the result with a Gaussian of sigma = 2.5.





The detected edges seem to be closer to the actual edge but some are still ‘around’ the edge. Below shows the images with sigma = 1. While the detected lines fall on the edge itself, there are a lot of small lines which suggests some further filtering needs to be done.