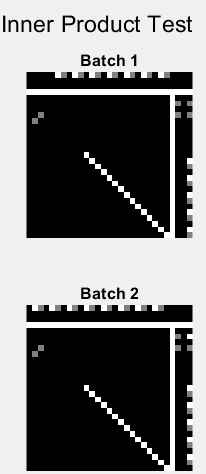
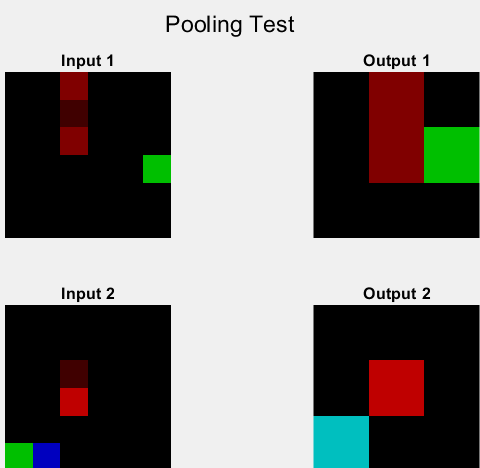
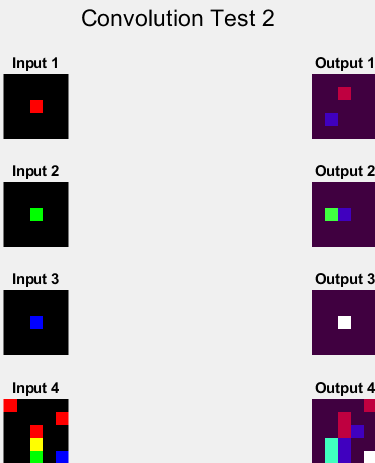
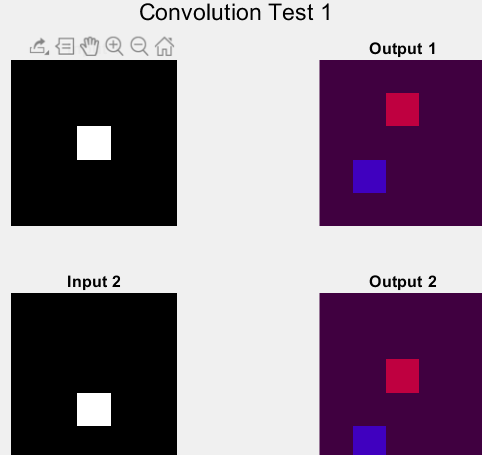
**Computer Vision**

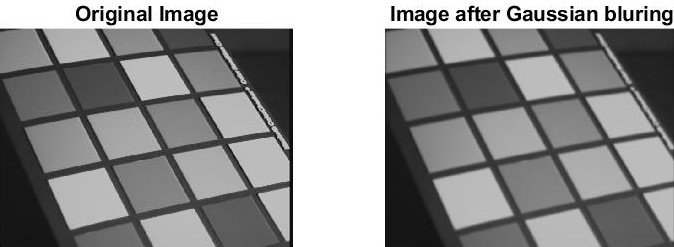
Project 5 Report

1. Figures below show the visualization results of the convolution, max pooling and fully connected forward functions. To do the max pooling efficiently I used [1].

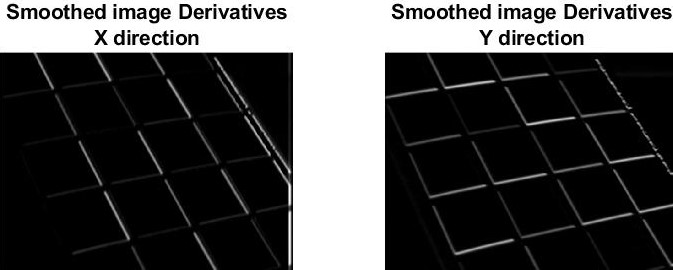


1. Figures below show the output of myEdgeFilter function and the intermediate output. The final output has also been compared to the MATLAB *“edge”* function.

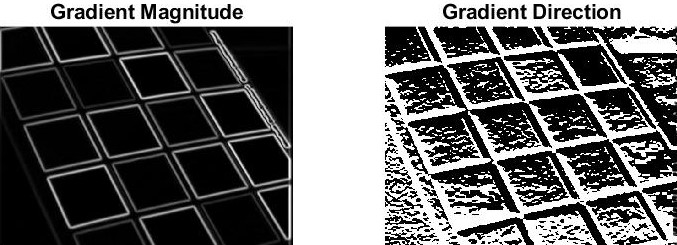
First step: Smoothing the input image using Gaussian kernel.



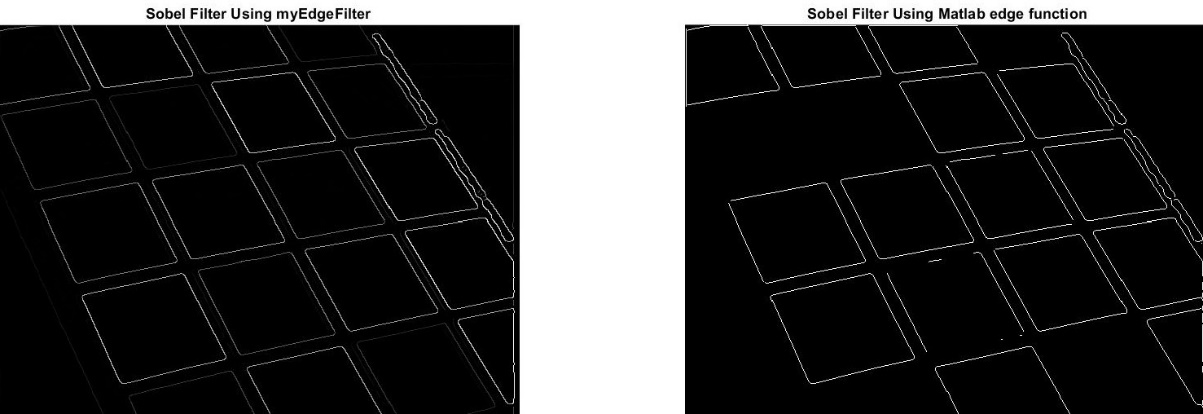
Second step: Calculating the smoothed image derivatives along x (imgx) and y (imgy) directions using Sobel filter.



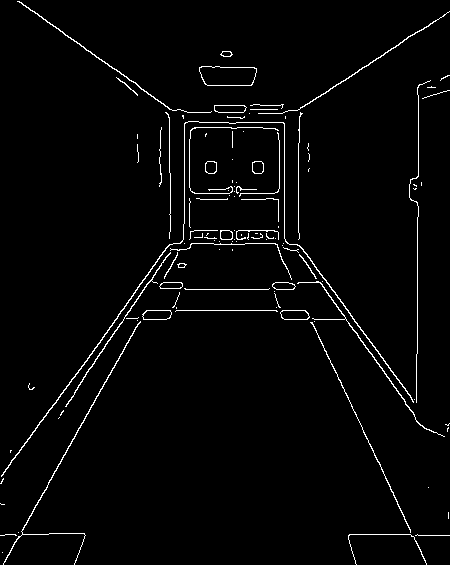
Third step: Calculating the magnitude and direction of the image gradient using imgx and imgy from previous step.



Final step: Perform edge thinning with non-maximum suppression algorithm. Figure shows the final result of myEdgeFilter function compared to MATLAB *“edge”* function.



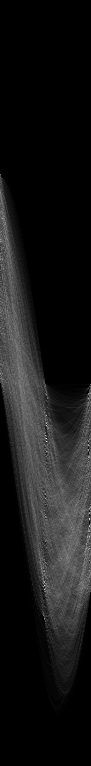
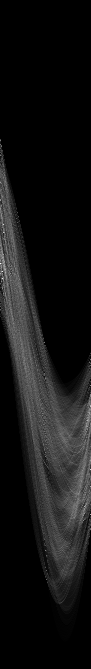
Figures show the edge images from picture 3, 4, 7 respectively (left to right).



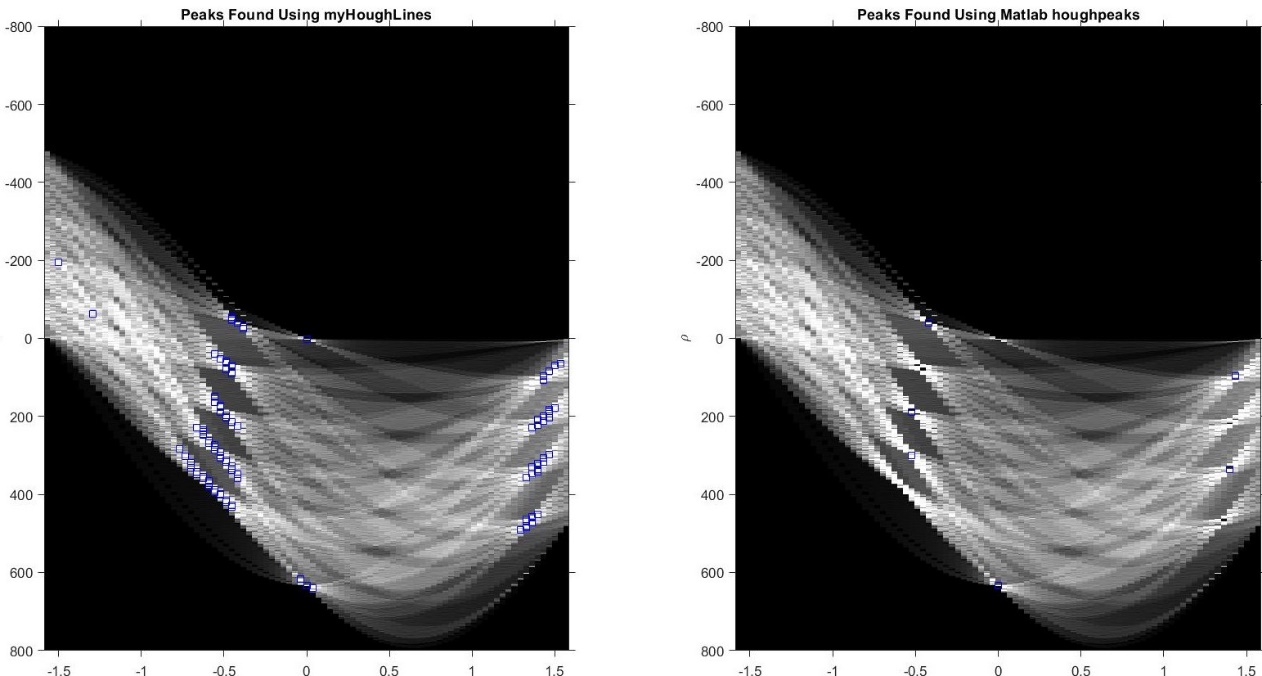
1. Figures below shows the outputs of the myHoughTransform function compared to MATLAB *“hough”* function. The to be compatible with MATLAB hough transform implementation. The where . Because the longest line that can lay inside an image is the diagonal line coming from one corner of the image to the other. The output of the function (Matrix H) has been normalized for the sake of the visualization.



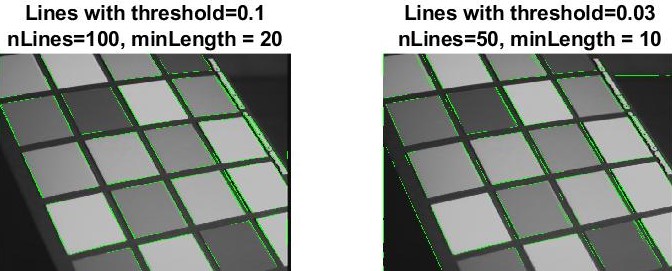
Figures below show the hough images of input images 5,6,8 respectively (left to right).

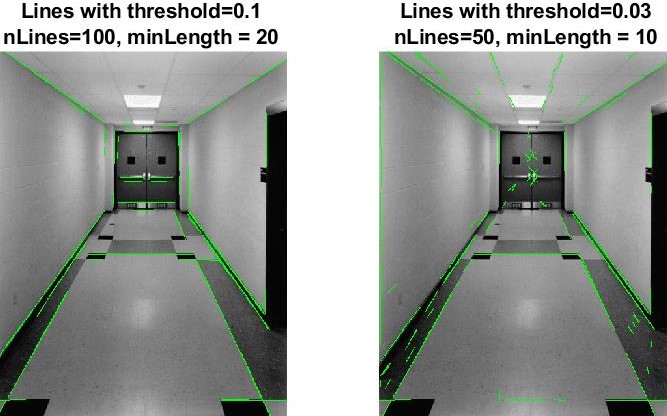


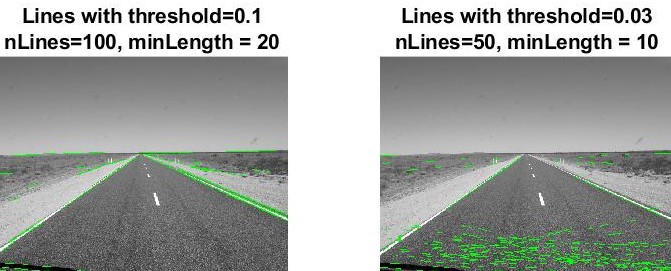
1. Figure below shows the peaks found in image 1 by myHoughLine function . Also the result has been compared to MATLAB *“houghPeaks”* function (MATLAB built-in function found 5 peaks).



1. Figures below show how the line segment changed when threshold, number of peaks and the length of lines parameters changed. Increasing length of the lines (minLength) reduced the noise in found lines by removing very short lines and keeping longer ones. While increasing the threshold caused the algorithm too keep stronger edges which have the more probability to be stronger lines in the image resulting in finding more of the real lines rather than noises. Results are for images 1,3 and 4 respectively (top down).







1. I have changed the threshold, nLines and minLength (minimum length of the found lines in matlab houghLine function). Larger threshold helped algorithm find stronger edges resulting in segmenting more defined lines in the picture. While increasing the minLength helped with reducing the number of noises assumed as small lines. Increasing nLines simply increased the options to segment resulting in finding more defined lines in the picture.

The algorithm still has some problem in finding vertical lines regardless of changing the parameters.

Implementing the hough transform was the most problematic part for me. I encountered some precision issues in functions like ceil, round and floor. I had to change them in the way that made the results the best.

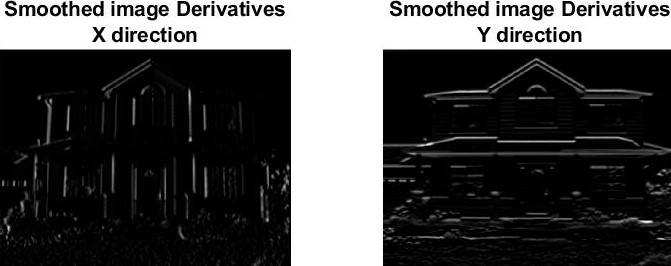
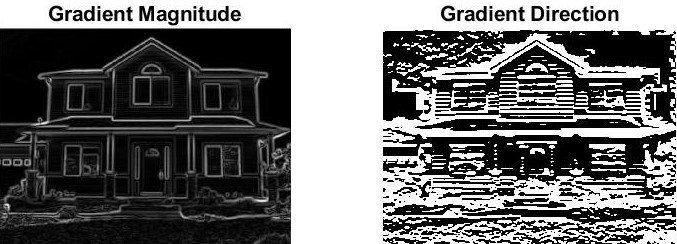
The code also works better on the images with lesser information in them. As you can see in the figures below the result for the left picture which has less detail is better than the right which has a lot of details.

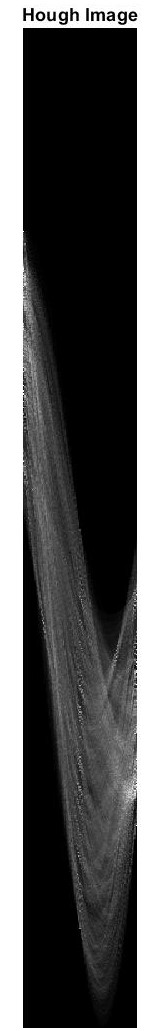
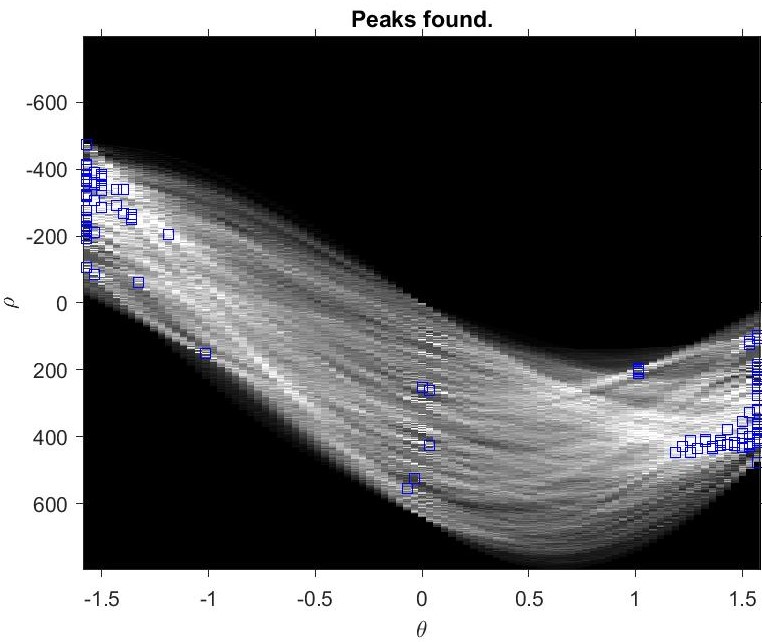


As expected the algorithm also performs poorly in finding the lines which have the low contrast with their background. I used small sigma for these pictures to avoid blurring the low contrast edges in the background. As you can see in the figures below the lines of the boxes near the wall in left picture and the edges of the laptop or its keyboard in the right picture has not been detected, while high contrast lines like one on the laptop monitor has been detected perfectly.



Figures below show the intermediate and final result of the houghScript on image 7.





Figures below shows the results on two taken Images other than the provided data set.



