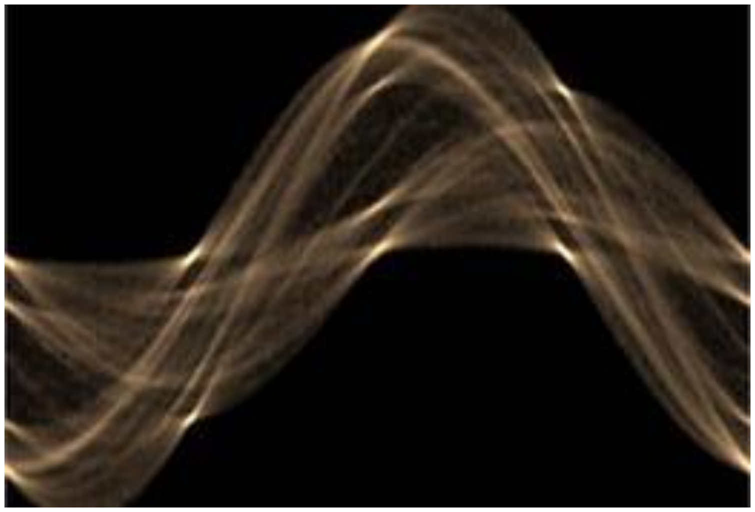
Project 1

Image Filtering and Hough Transform

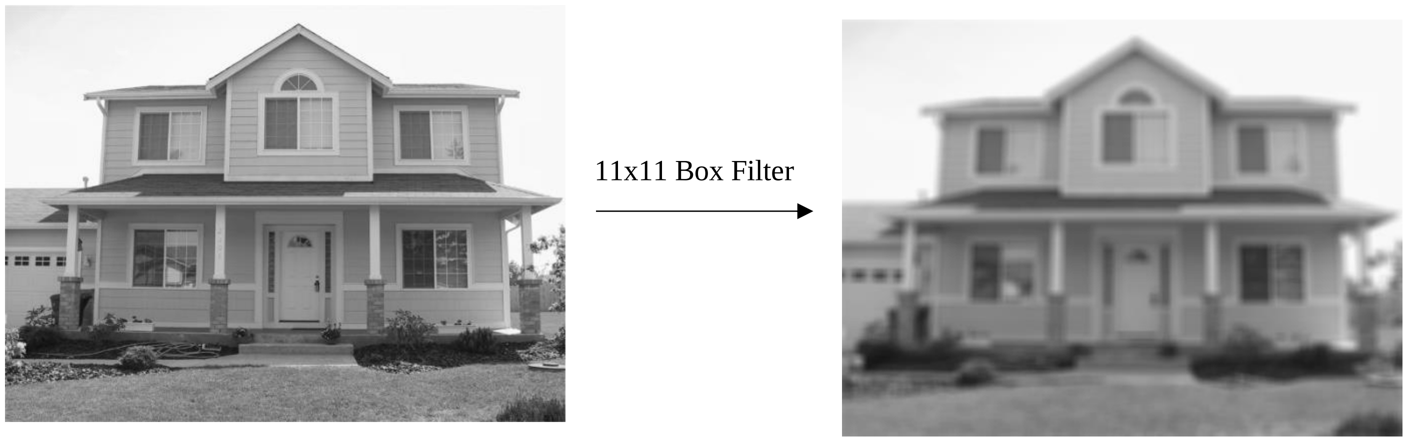
Computer Vision - CMPT 762

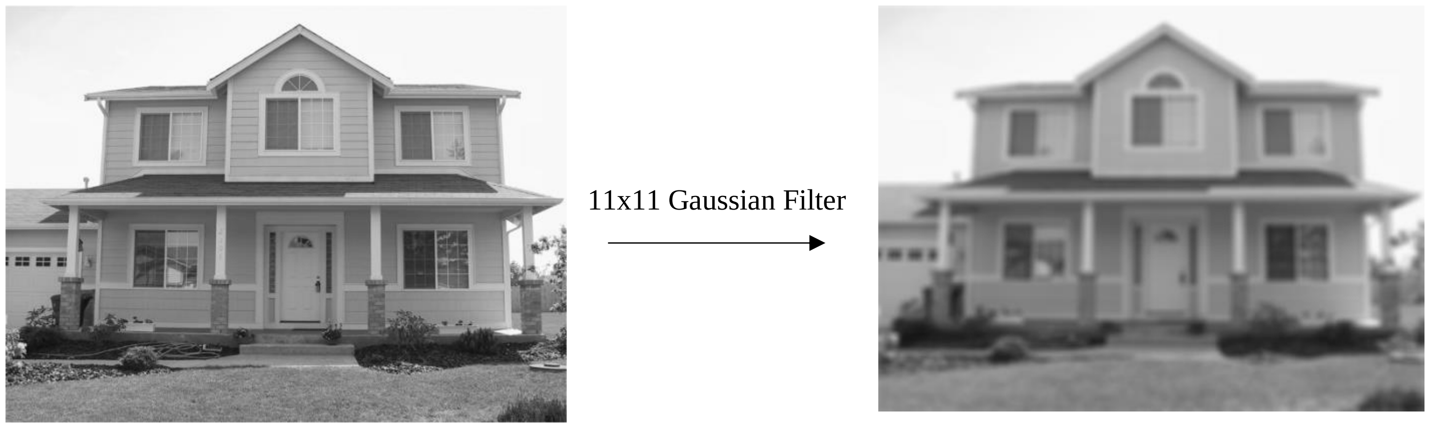


2.1 Convolution

Here we implemented convolution operation without using Matlab's function. As you can see we have padding and vectorized the implementation by converting for loop to matrix multiplication.

Some sample results:

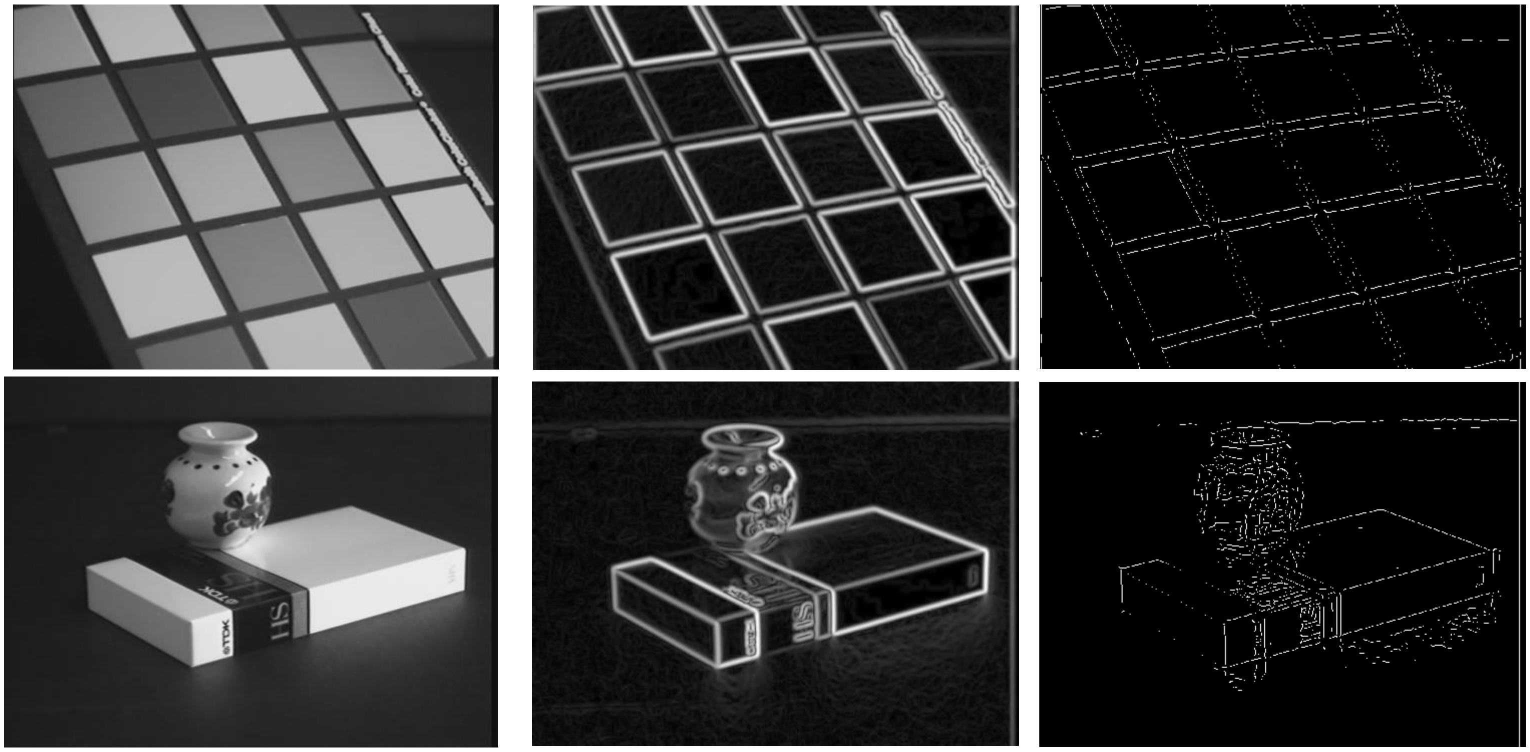




2.2 Edge detection

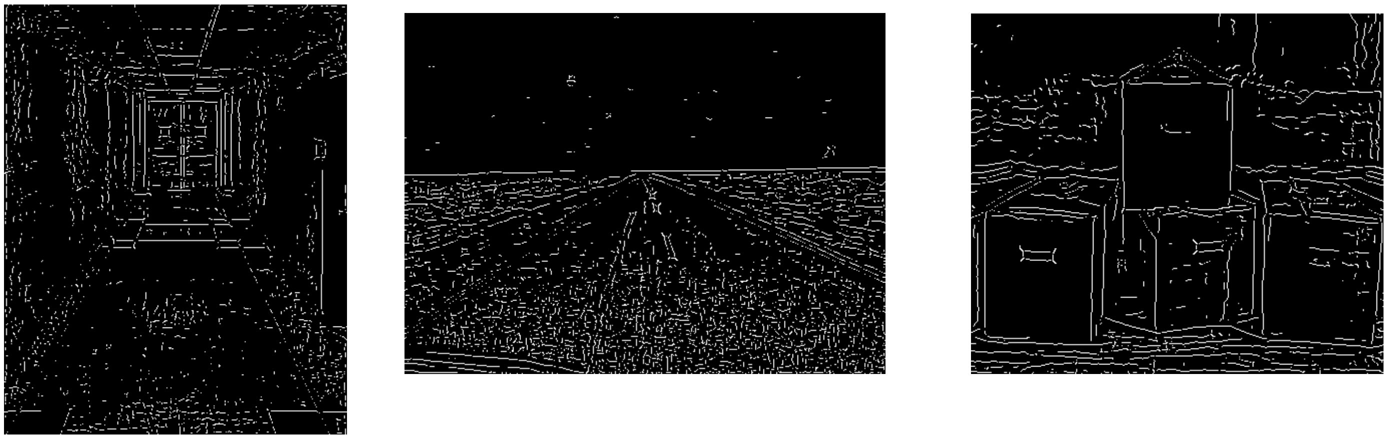
Here we implemented edge detection by applying Sobel filter after smoothing. Then we apply non-maximum suppression to have thin edges and show the results after applying the threshold.

For implementing non-maximum suppression, we used matrix multiplication instead of for loop to increase the speed. Please refer to matlab code for more details about implementation



Input image Gradient magnitude after Sobel filter non-maximum suppression

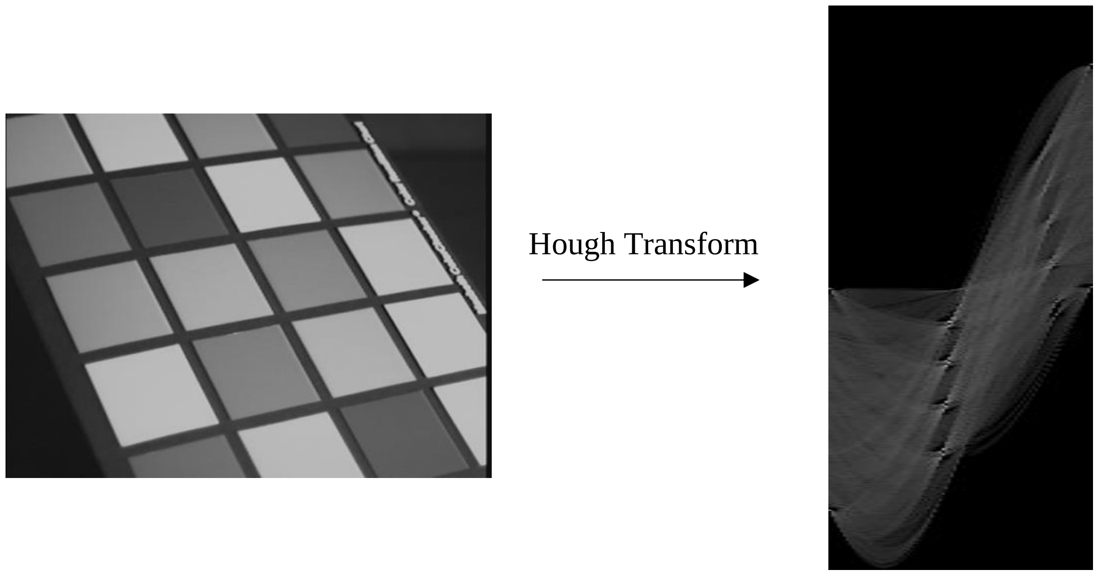
+ threshold



2.3 Hough Transform

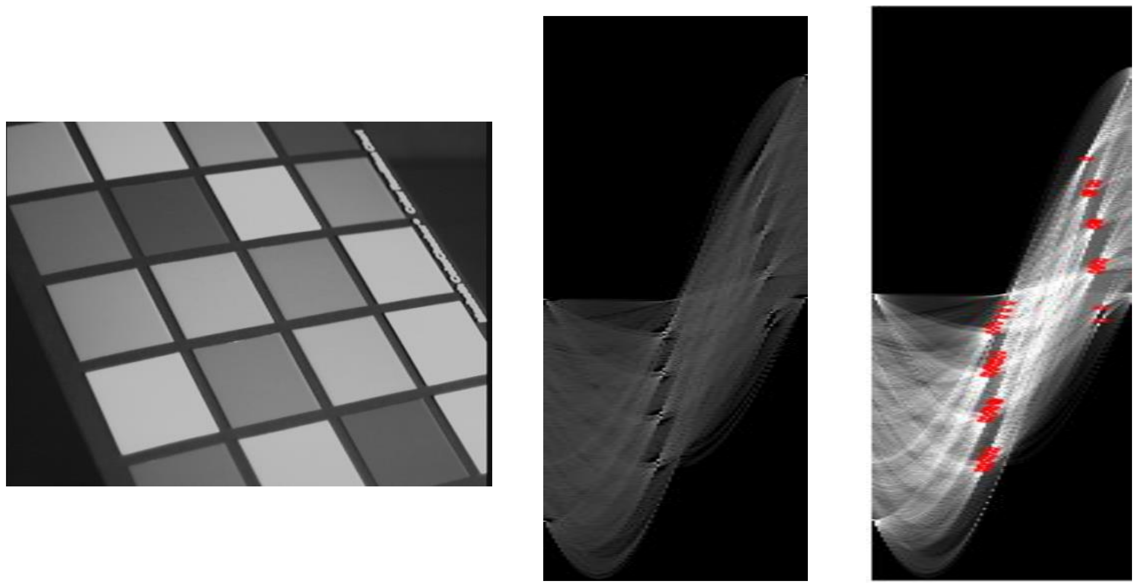
We implemented Hough transform from scratch. You can refer to the Matlab code for more details.

Here is a sample result for this function:



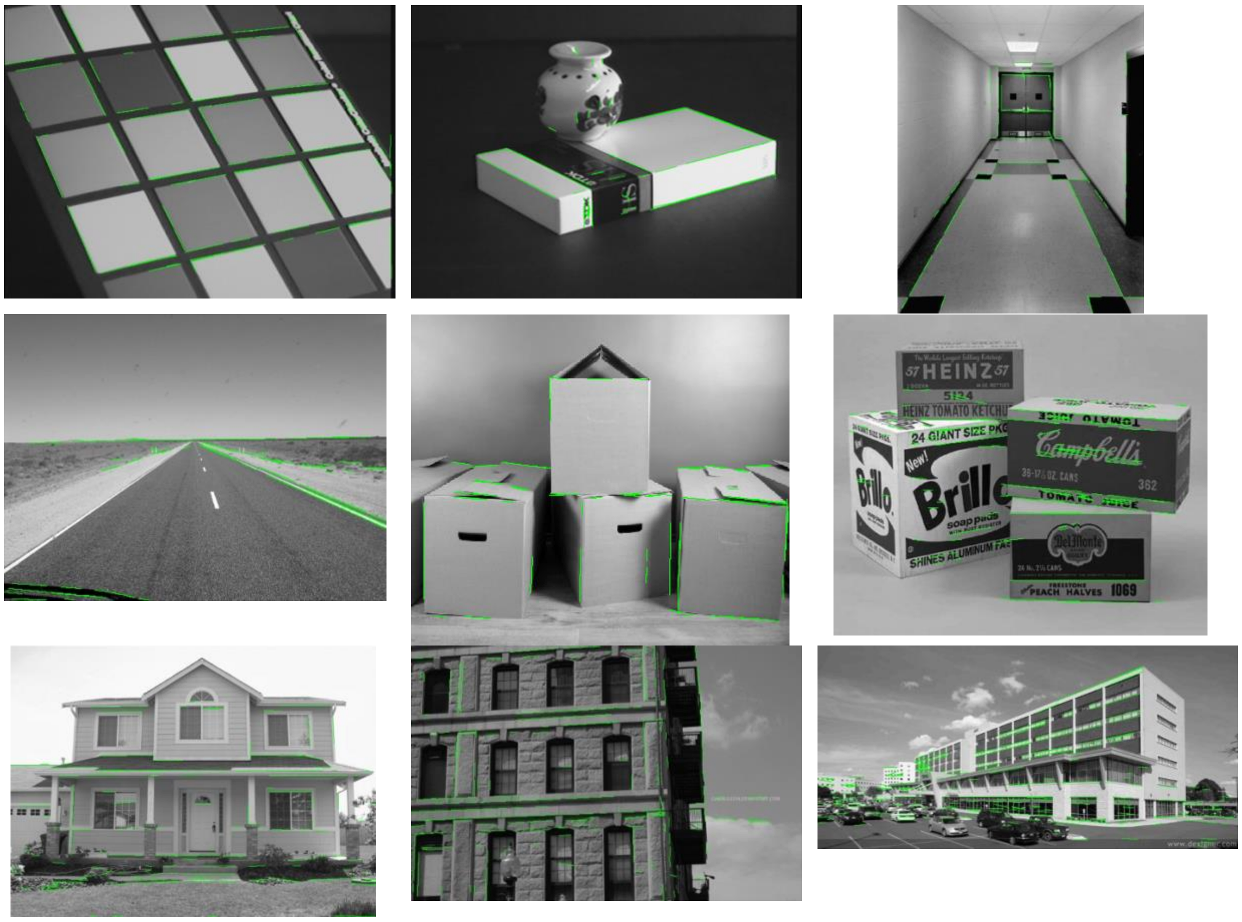
2.4 Finding lines

I implemented non-maximal suppression by myself. You can find the implementation in matlab directory in `nonMaximalSuppression.m` file.



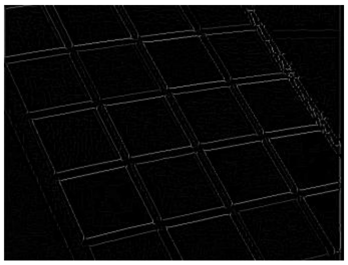
2.5 Fitting line segments for visualization

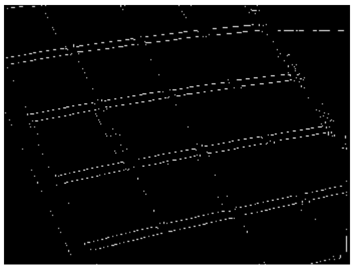
Here are results for this section after fitting line segments. (threshold = 0.13)



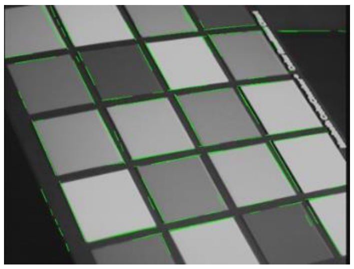
2.6 Experiments

Sample of an intermediate output images:

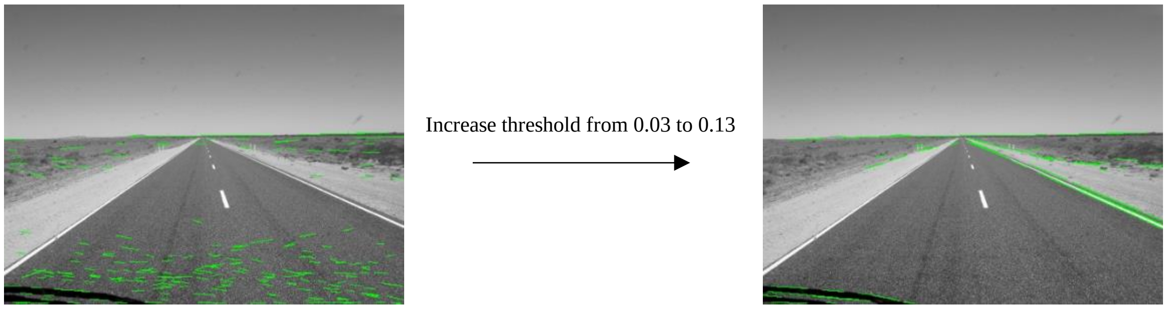

edge


threshold

hough

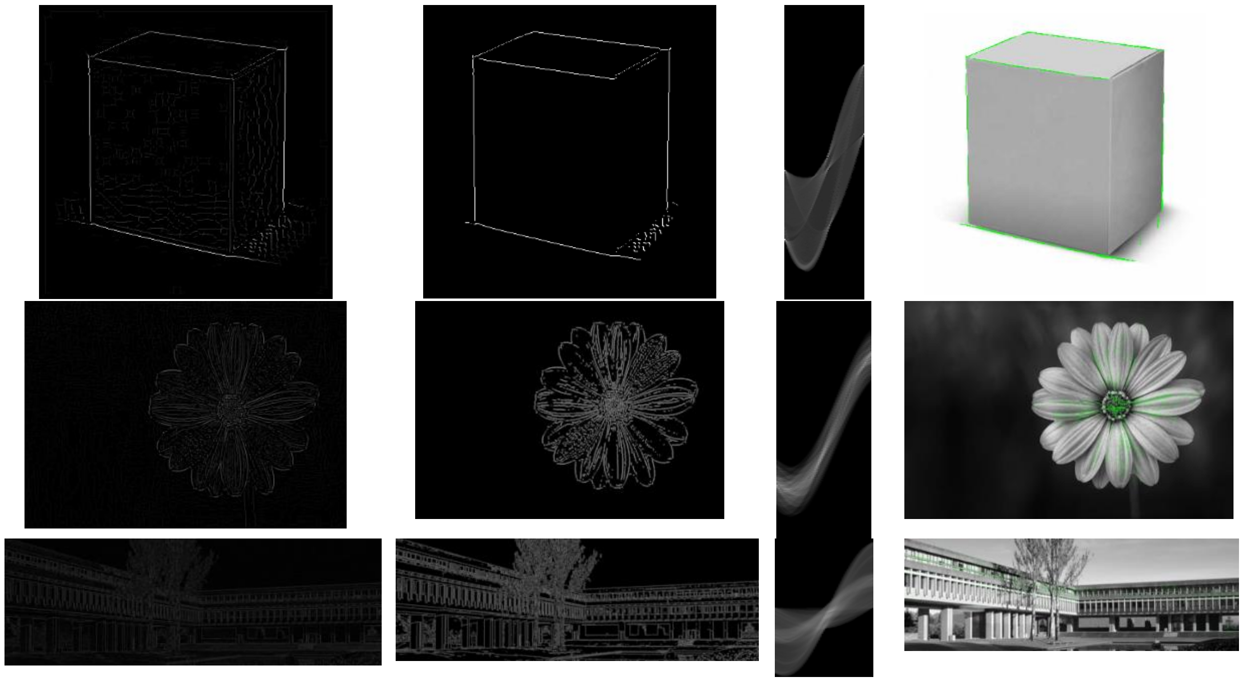

lines

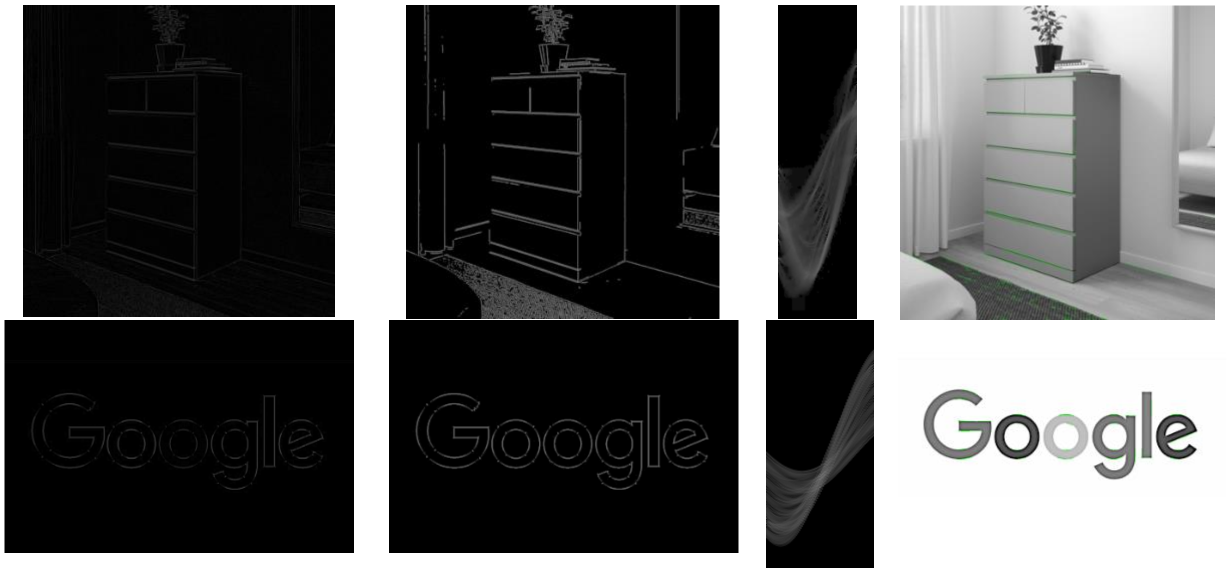
The code doesn’t work perfect on all the images with a single set of parameters (but the results can capture a good portion of edges) because the optimal set of parameters vary with the images and it depends on different factors like resolution, number of edges in an image, distance between edges and so on. For example, for img02.png we have problem for the vase because it can’t represent by just straight lines. Also, by increasing threshold we can eliminate strange lines in img04.png



By implementing different algorithms using vectorization we can improve the performance and speed, or we can use built-in function in Matlab. In addition, setting optimal parameters for every image can be helpful. So, I implemented convolution and non-maximal suppression using vectorization and matrix multiplication. Also, I tried to find the optimal parameters for every image to increase the performance.

Here you can see the results for my own images in ec directory.




edge
threshold
hough
lines