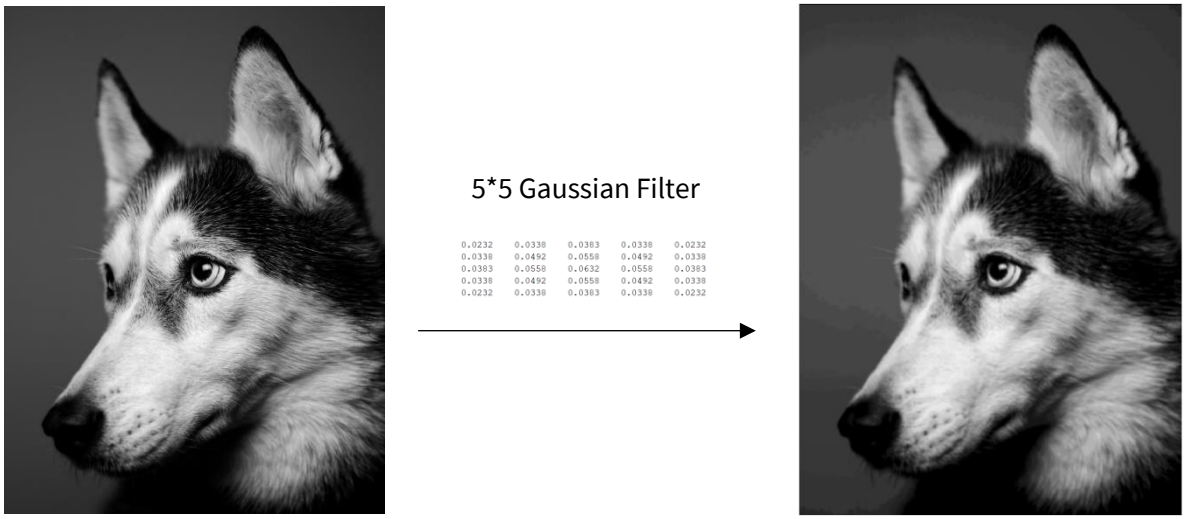
Computer Vision – Assignment 1

Ali Gholami (301403061)

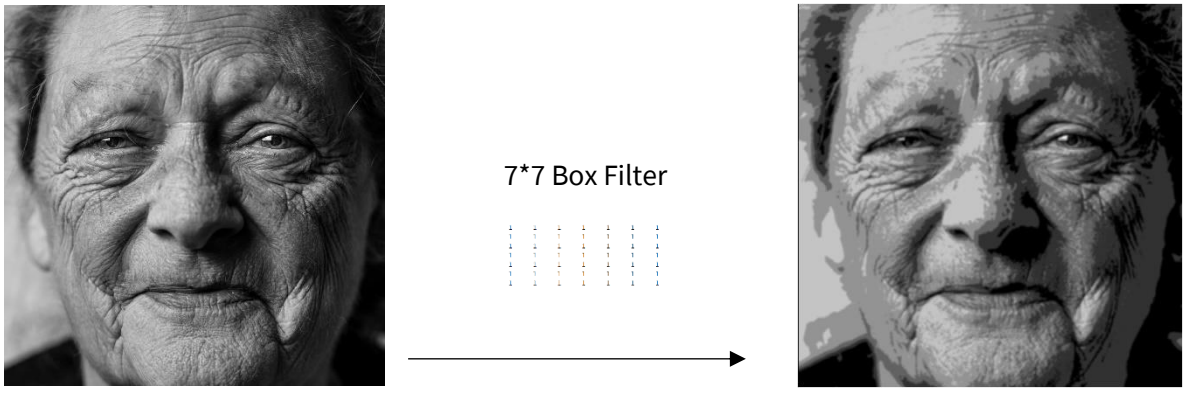
This is the report for the computer vision course, CMPT762, taught by Dr. Furukawa at the Simon Fraser University, Spring 2020. The source code of this assignment is provided under the **matlab** directory.

1. This question is regarding the Convolution operation and its implementation. The code is provided under the 'myImageFilter.m' file. This code is implemented from scratch, performs padding on the input image, and supports vectorized inputs. Some of the output samples are provided below:



Input Image

myImageFilter Output

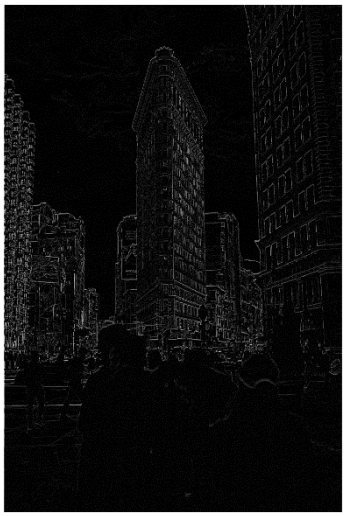


Input Image

myImageFilter Output

2. This question asks to implement an edge detection method, based on the horizontal and vertical Sobel filters. The result given below is first smoothed with a Gaussian filter, and then the magnitude of gradients along x and y axes is calculated.

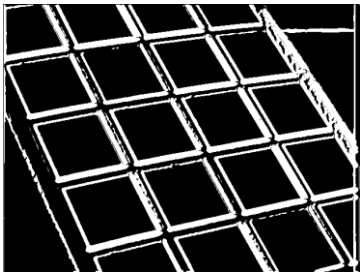
Gaussian + Sobel



Input Image

myImageFilter
Output

3. This part focuses on Hough transform, which is used to find lines in an image. This method converts image space to a different parameter space to represent lines with points. The resulting Hough matrix is provided below for a sample image:

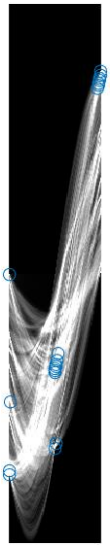


Hough Matrix

Input Image

myHoughTransform
output

4. This part of the assignment focuses on the line detection problem. A part of this code which perform non-maximum suppression and finds local maximums is borrowed from the following link: https://github.com/sfoerster/matlab/blob/master/hough/hough_peaks.m
A sample output of the peaks is provided below:

5. Line segment fitting visualizations are provided below:


6. Please refer to the results folder for high quality intermediate results. A sample of intermediate results is given here:




Did your code work well on all the image with a single set of parameters? Yes.

How did the optimal set of parameters vary with images? There are a set of parameters like sigma, that can affect the performance of the system. For example, using a small sigma for images that have a very small set of edges can be quite enough. Instead, for the images with lots of edges, we need a little bit more smoothing, therefore we should set sigma to a higher value in that case.

Which step of the algorithm causes the most problems? I think the most problematic part is changing an image to its gradient magnitudes. A lot of variations in intensity may be regarded as edges, which will cause problems like detecting false lines down the road.

Did you find any changes you could make to your code or algorithm that improved performance? Yes. A lot of the times using a built-in function by Matlab can have a far better performance than implementing it. For example, using find(x) instead of finding the actual indexes by looping and applying conditions can be faster. Also, I was highly aware of variable duplicates while coding, which is to take a copy of a variable once and use it everywhere, instead of calculating it every time. On top of that, sometimes using an on the fly variable notation can cause less resource allocation which is a good thing from a performance point of view. I know that there are ways to improve the speed of convolution operation which is not implemented in this project.

Here is the final results on three images (Husky, Urban and Senior):