CMPT 412
Assignment 1 Report

Maoshun Shang

301280479

Introduction

This assignment mainly focuses on the technique studied in lectures, including basics about image filtering, Gaussian and Sobel filter, hough transformation, and line detections. The programming language used is MATLAB, which helps with reducing the complexity of heavy computation on matrix and vector calculations. In the implementation, the code transfers a colored 24-bit image to greyscale, then perform edge detection, hough transformation, and line detection respectively, finally output an image with green lines indicating where the detected lines are.

How to run the code

The main entrance for running the whole project is the file named houghScript.m, please prepare input images in data folder as ".jpg" format. After running houghScript, the final output will be stored in "result" folder. Each input image corresponds to four output images.

What Effects the Parameters Have

The parameters include sigma, threshold, rhoRes, thetaRes, and nLines.

The sigma controls the size of the Gaussian kernel, the larger sized kernel has better effect of reducing the noise. However, an over-sized kernel may ignore the signals who represent actual lines in the original image.

Threshold is a limit of whether each signal from edge magnitude image could be a line on the final image. A larger threshold reduces the number of extracted lines.

rhoRes and thetaRes are two parameters that indicate how accurate the algorithm is. However, the more accurate the algorithm is, the higher running time.

nLines is a parameter to control how many lines we want to see on the line extraction result image. For example, if nLines = 30, the final image shows 30 lines with the top signal magnitude. Having more nLines increase the potential of showing lines which are actually not a line in the original image.

Sample outputs

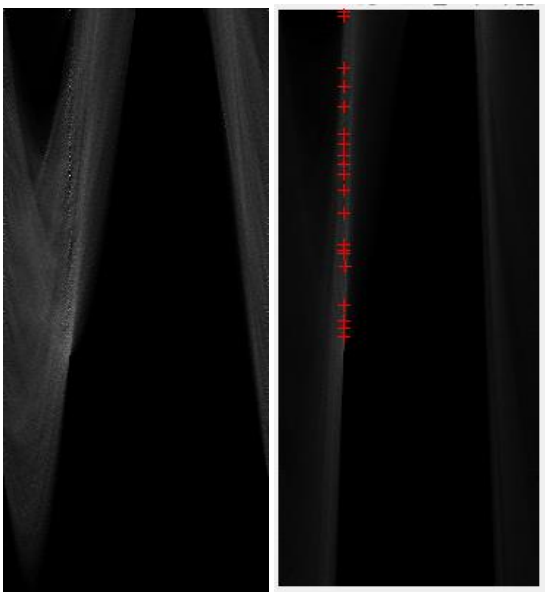
With provided images

The script has been tested with multiple images, one out of these sample outputs is displayed below. They represent the original image, image after line filtering, the edge magnitude image after filtering out signals below threshold, the hough lines (red plus sign indicates the peak), and the greyscale image with detected lines.

Original

Edge Filtering

Edge Filtering with Threshold



Hough Transform

Greyscale Image with Extracted Lines

According the images above, the code did detect some of the lines in the image. However, the result is not satisfying. The main unsatisfying reason is the final image shows too many “lines” that are should not be recognized as lines. For example, the grass is being extracted as many lines, but it will not be indicated as lines when human detect them manually.

The problem is the settings of the parameters. By analyzing the intermediate outputs, there are numerous noises on the image after edge filtering. By analyzing the second image(image after edge filtering), the grass is mapped to massive number of thin lines, then wrongly recognized as a set of connected lines. This error occurs at the process of edge filtering with Gaussian kernel. Therefore, by tuning the parameters, change the sigma from 2 to 6, the final output image makes more sense due to reduced noise.

The images below show the final image with new set of parameters. The new image does not contain the extra lines on the grass.

Sigma = 2

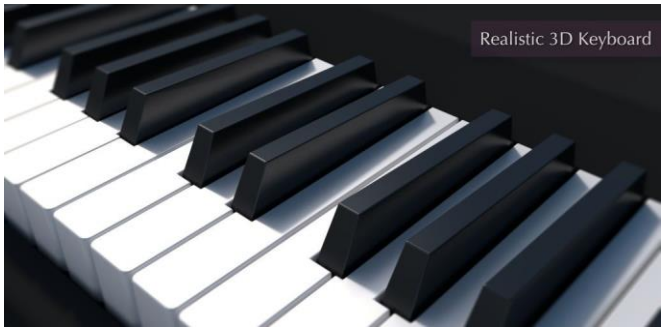
Sigma = 4

This also applies to other images as well. Below shows the difference between final output with original parameter (sigma = 2) and new parameters (sigma = 4).

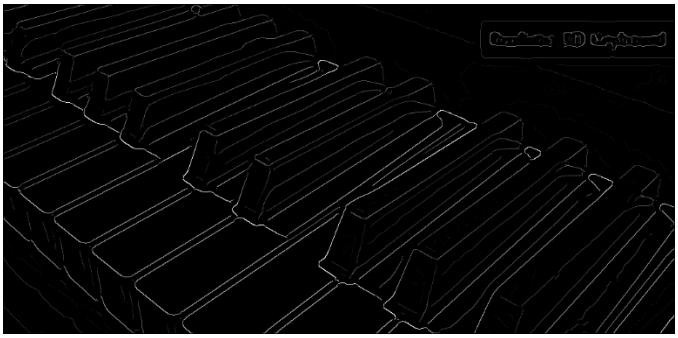
Sigma = 2

Sigma = 4

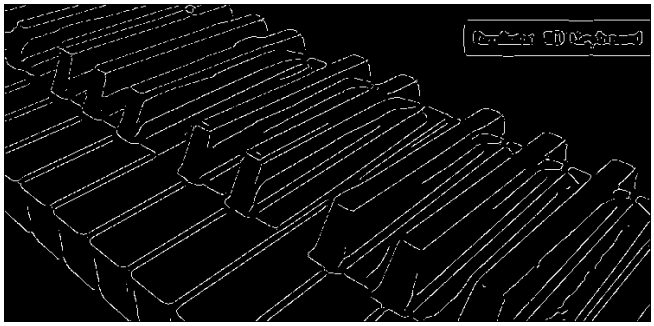
With my own images



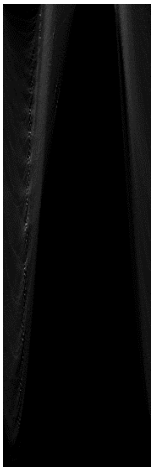
Original Image



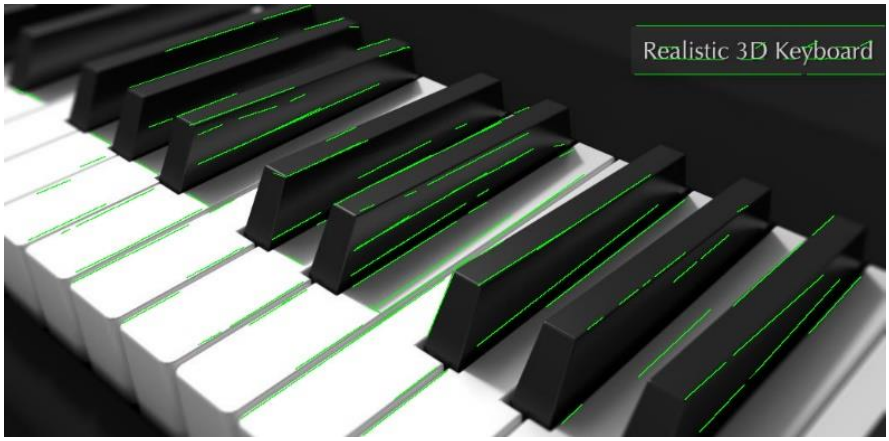
Edge Filtering



Edge Filtering with Threshold

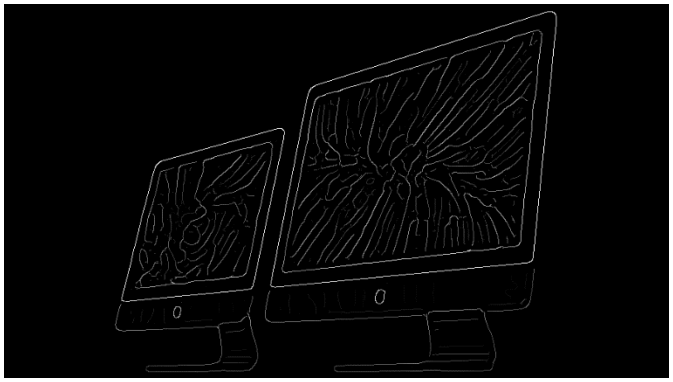


Hough Transform

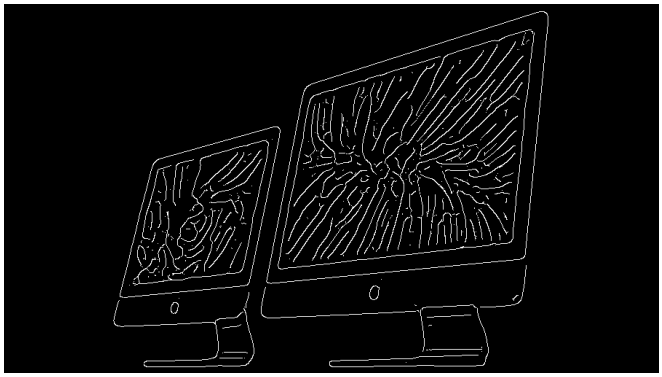


Greyscale Image with Extracted Lines

Original Image



Edge Filtering



Edge Filtering with Threshold

Hough Transform

Greyscale Image with Extracted Lines

It can be observed that some of the image have better result. For example, the piano keyboard had a better result than that of iMac. This is due to the lines on the original image of the keyboard are more observable and clearer than that on iMac. Thus, the quality of original image could play a role in terms of final line detection quality.