CS512 Assignment 3: Report

Kajol Tanesh Shah
Department of Computer Science
Illinois Institute of Technology
March 10, 2022

Abstract

In this assignment, we have performed operations on images like edge detection, line detection, smoothing, model fitting, etc. To do these operations, we have used the OpenCV library in python and other libraries like Numpy, Matplotlib, Scipy, etc. were also used.

1. Problem Statement

In this assignment, our aim was to perform various operations on images like converting to grayscale, smoothing, edge detection, detecting lines using hough transform, model fitting, etc. These image transformations are helpful as we can many times detect some features in the images after transformation which we are not able to identify in the original image. So, we have performed these transformations on images of different sizes.

2. Proposed solution

Using the OpenCV library and other supporting libraries like numpy, matplotlib, math, etc. we have implemented the program to do transformations on the images. Below are some of the functions that we performed and their working.

A)Grayscale
To perform various operations on the image, we first converted the image into grayscale using function COLOR_BGR2GRAY

B)Smoothing
Before performing edge detection, we performed smoothing on our image to remove noise using Gaussian filter i.e. using function GaussianBlur()

C)Edge detection
In order to detect lines in an image, we first performed edge detection using Canny Edge detection function Canny() which gave us thinned edges in the image

D)Hough Transform
We performed Hough transform to detect lines in the edge image. In this, for every pixel in the edge image, we checked whether the pixel was an edge pixel or not and calculated the corresponding theta and d values for it. We used the line equation formula x cos t + y sin t = d. Then, we stored these values in an array i.e. accumulator and incremented the accumulator. If the value in the accumulator was greater than threshold, we selected such

points and then used them to detect lines. Then, we displayed the parameter plane as an image and highlighted the intersection points.

E)Non-Maximum Supression
We detected k number of lines in an image based on the k value selected. Here, we detected peaks in the accumulator and performed non-maximum suppression

F)Fitting and plotting detected lines
After detecting k lines, we then fitted a line to the edge points belonging to the line and then plotted the lines.

3. Implementation details

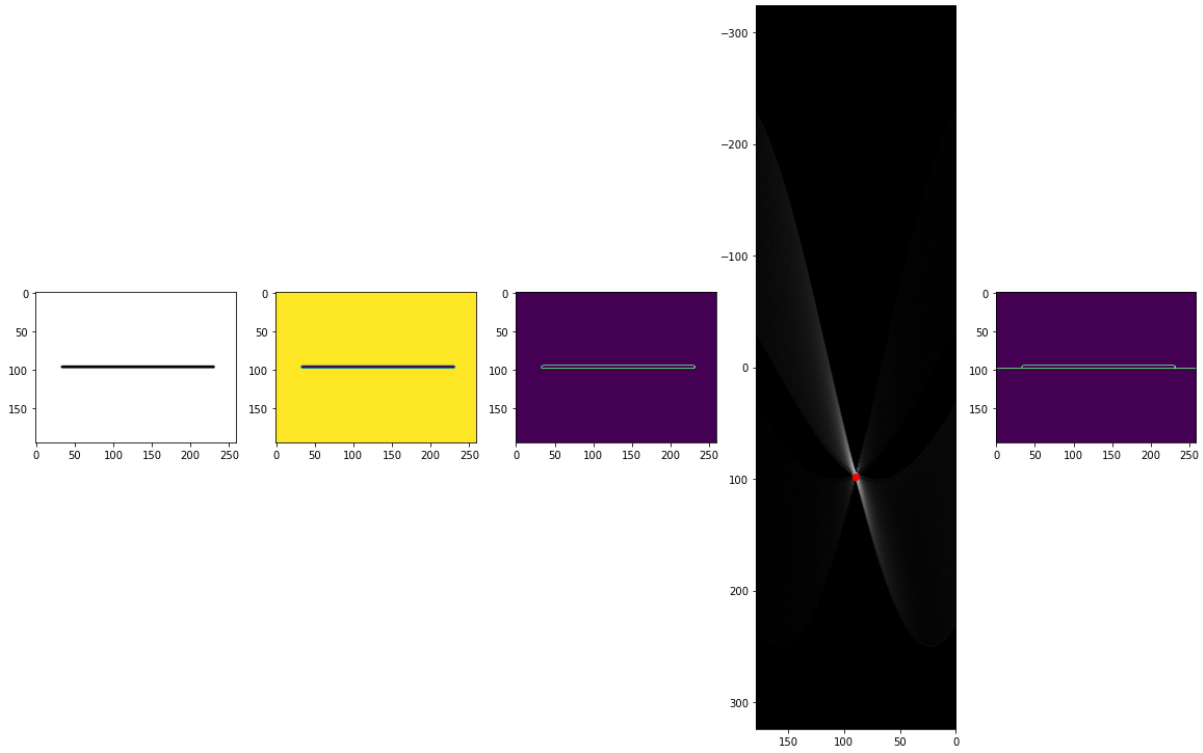
Some of the problems and design issues that were faced are as mentioned below:

- At first, the lines were not getting displayed correctly on the image due to incorrect variables used but then was able to find the issue and fix it
- Was getting array index out of bound error and had swapped the x and y variables by mistake. Was able to identify and fix the mistake
- One of the output subplots was overlapping the other one. It was due to the same index given for 2 subplots
- At first, was not able to get the intersection points in the parameter space but was able to figure it out

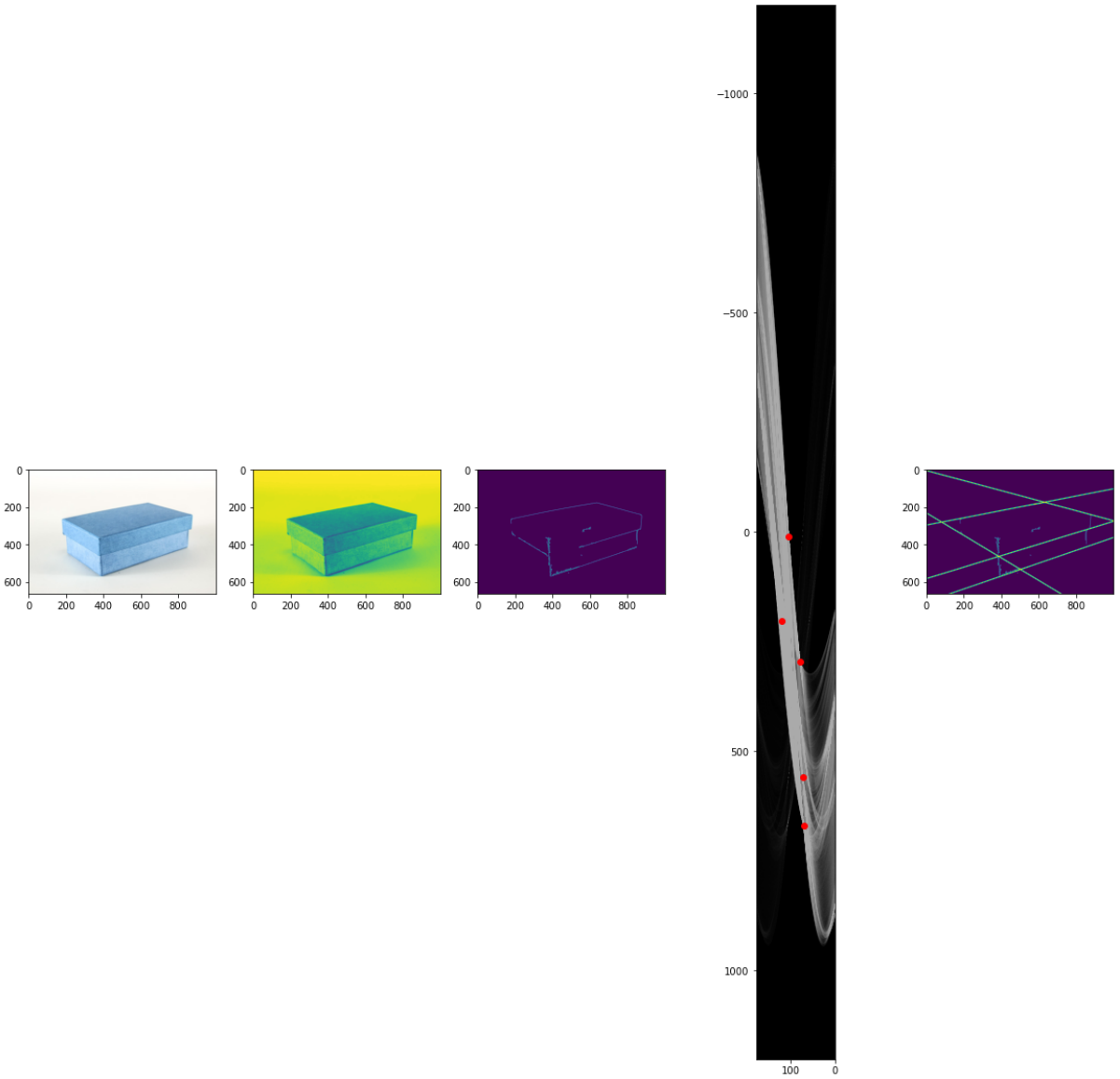
4. Results

Left to Right:
1st image: Original image
2nd image: Grayscale
3rd image: Canny Edge Detection
4th image: Parameter plane
5th image: Lines detected

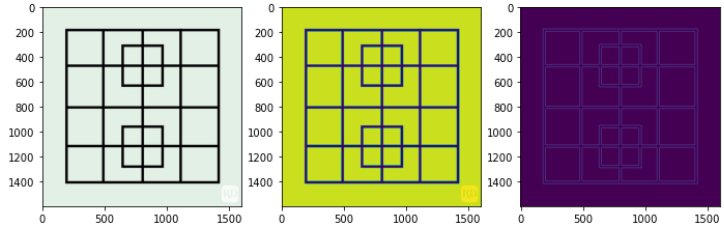
A)In the below image, you can see that as it contained a single edge, a single intersection point in red can be seen in the parameter space and a line was detected for the edge

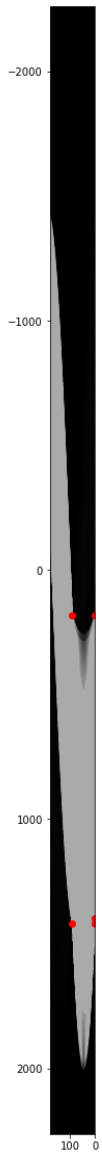


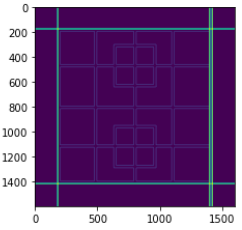
B)In the below image, you can see that as it contained many edges, more than one i.e. 5 intersection points in red can be seen in the parameter space and 5 lines were detected for the edges. Here, as we had selected k = 5, only 5 lines with max votes were selected



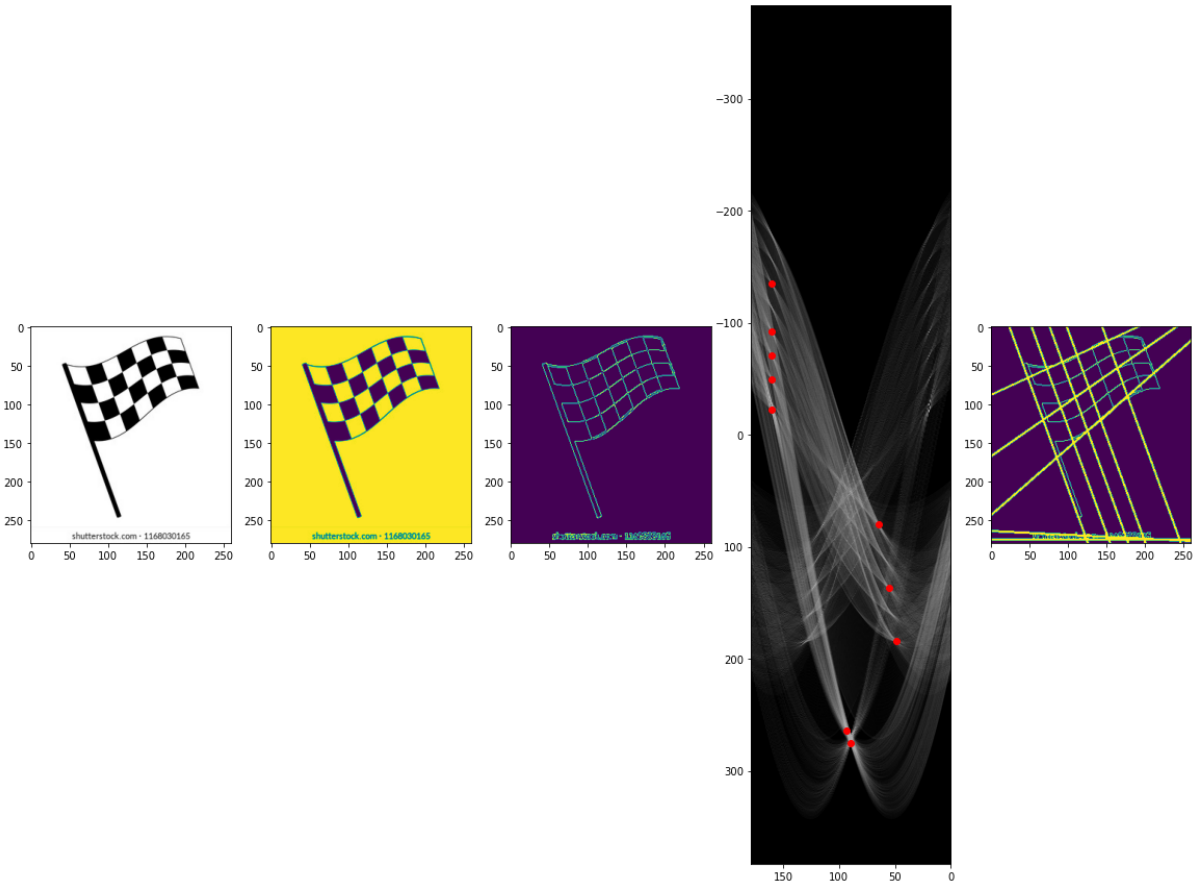
C)k=5







D) For k = 10, 10 lines were detected in the image



E)For k=5, 5 lines were detected