Laura Machlab
MP6 Report
Computer Vision
April 27, 2023

Task

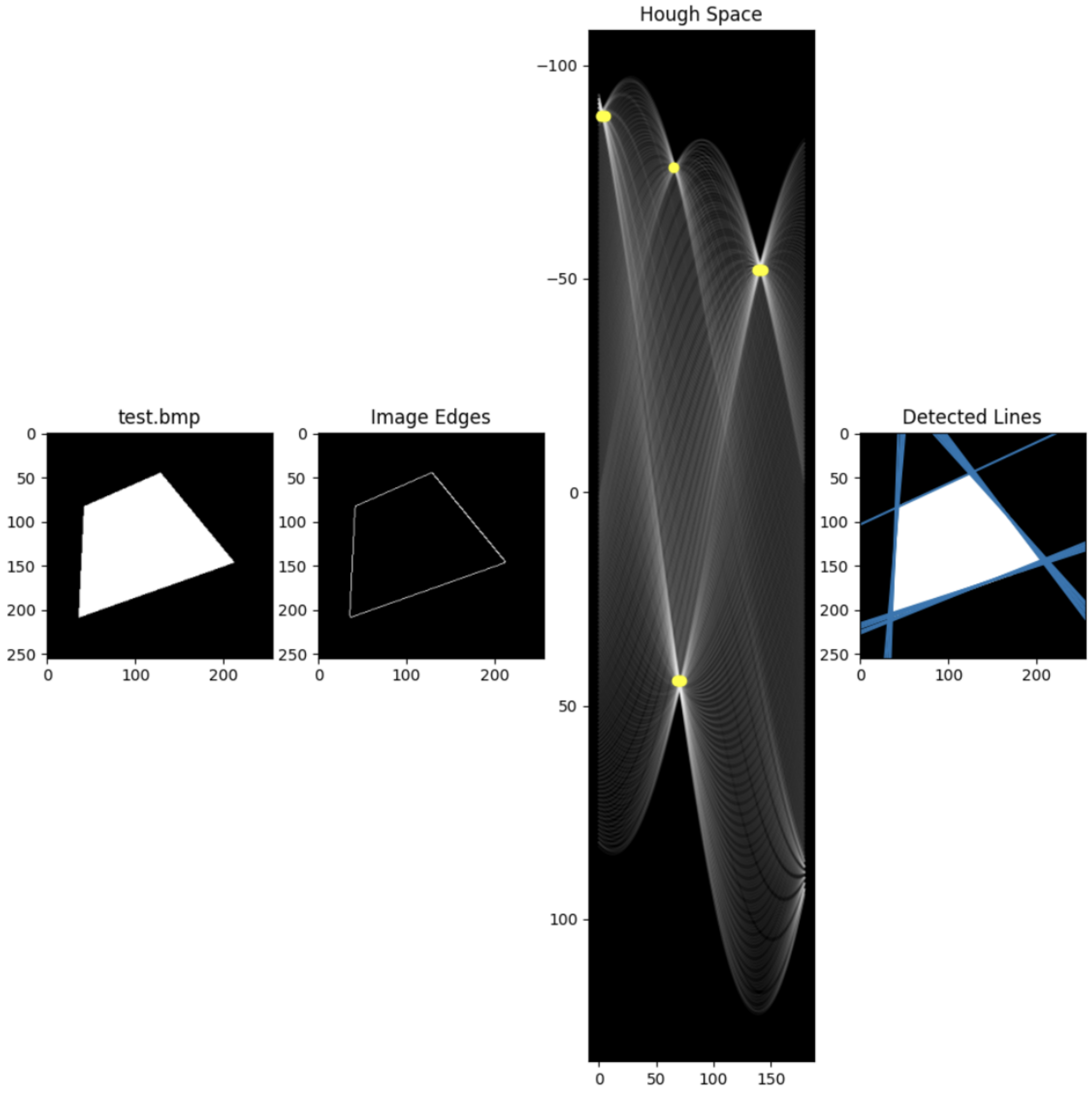
The task for this MP was to use the Hough Transform algorithm to carry out line detection for the given input images. This means that the function made will identify lines on edges in the image.

Algorithm

To do this, the first step is finding the edges in the image. I did this by using Canny edge detection. The way that the Hough transformation algorithm works is that it uses the basic line equation, y=mx+b, and rearranges it to the form b=y-mx. Then, the idea is to map (x,y) points of the image onto (m, b) points in the parameter space. This means that a line in the image space is a point in the parameter space and vice versa. However, because the parameter space has an infinite size, the relationship of rho = x*cos(theta) + y*sin(theta) is leveraged. This is the hough space. If you draw a line that goes through the origin and normal to a line in the parameter space, the length of this line is rho, and the angle of this line is theta. The goal is to find all values of rho and theta in the hough space for a given x and y in the image. When this is plotted, the points in the hough space with the greatest overlap of lines give the rho and theta values – that can then be converted to m and b values which determine a line in the original image. So, to carry this out, after getting the edges in grayscale of the original image, I step though each pixel, and if that pixel is on an edge, I calculated all possible values for rho and theta. I tracked these values on a matrix called the accumulator matrix. After doing this for all pixels, I applied a threshold where any rho-theta combinations in the accumulator matrix have a number of instances greater than the threshold were selected as lines. Rho and theta were used to find two (x,y) points and these points were used to map a line onto the original image.

Results

I applied the Hough transformation algorithm to the three given images. For each image, there are groups of similar lines that cluster around one another. I found that when I increased the threshold value, less lines showed up. However, in some cases the decrease in lines came with loss of important lines in the image. If I were to continue with this, I would try clustering the edges to choose one average one per group. In the first three shown below, I played around to find the threshold value that included all needed lines but minimized the size of the groups of lines. These three also each have a parameter space of 180 possible rho values and 180 possible theta values.

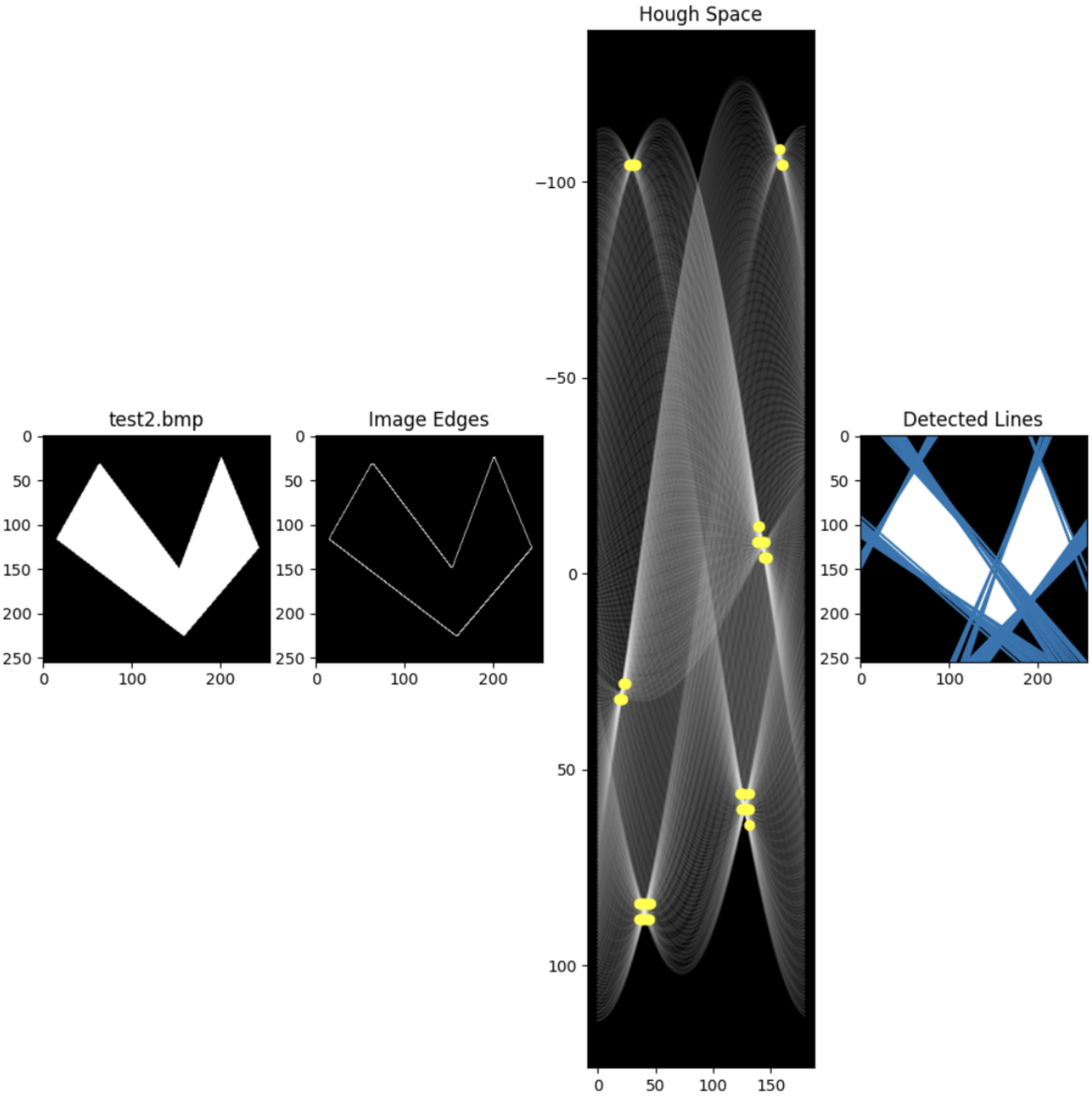


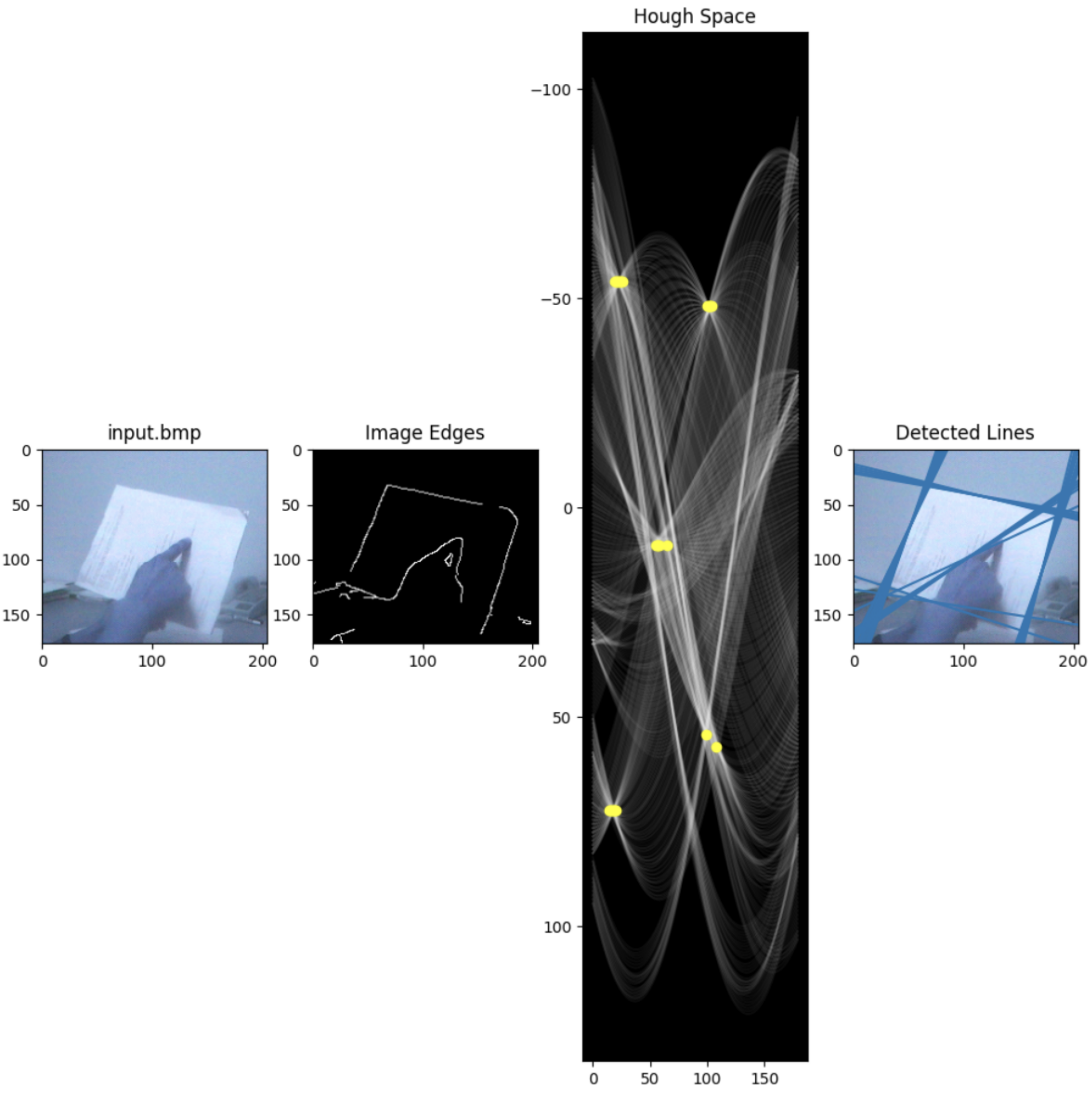
test.bmp

Image Edges

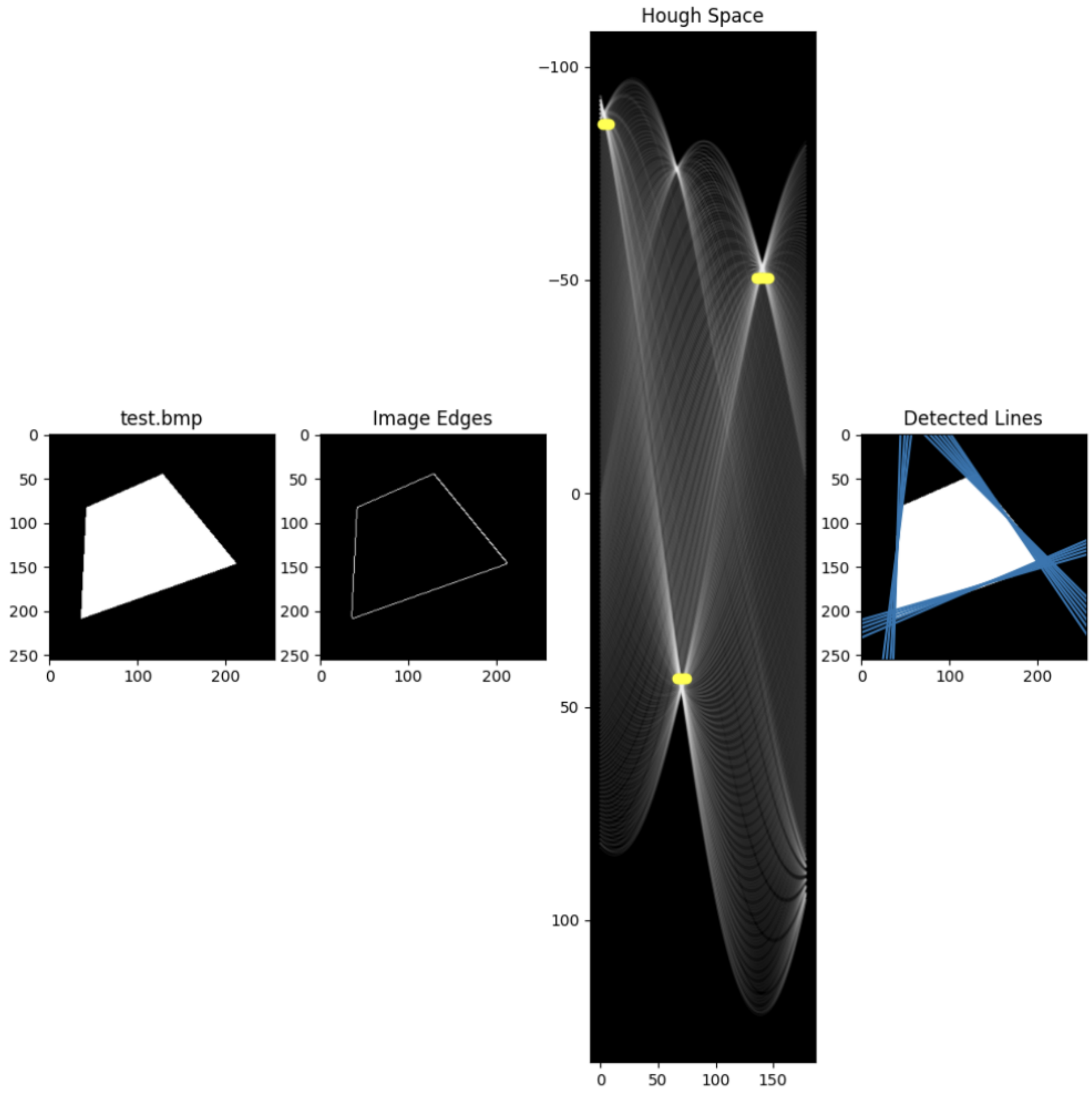
Hough Space

Detected Lines

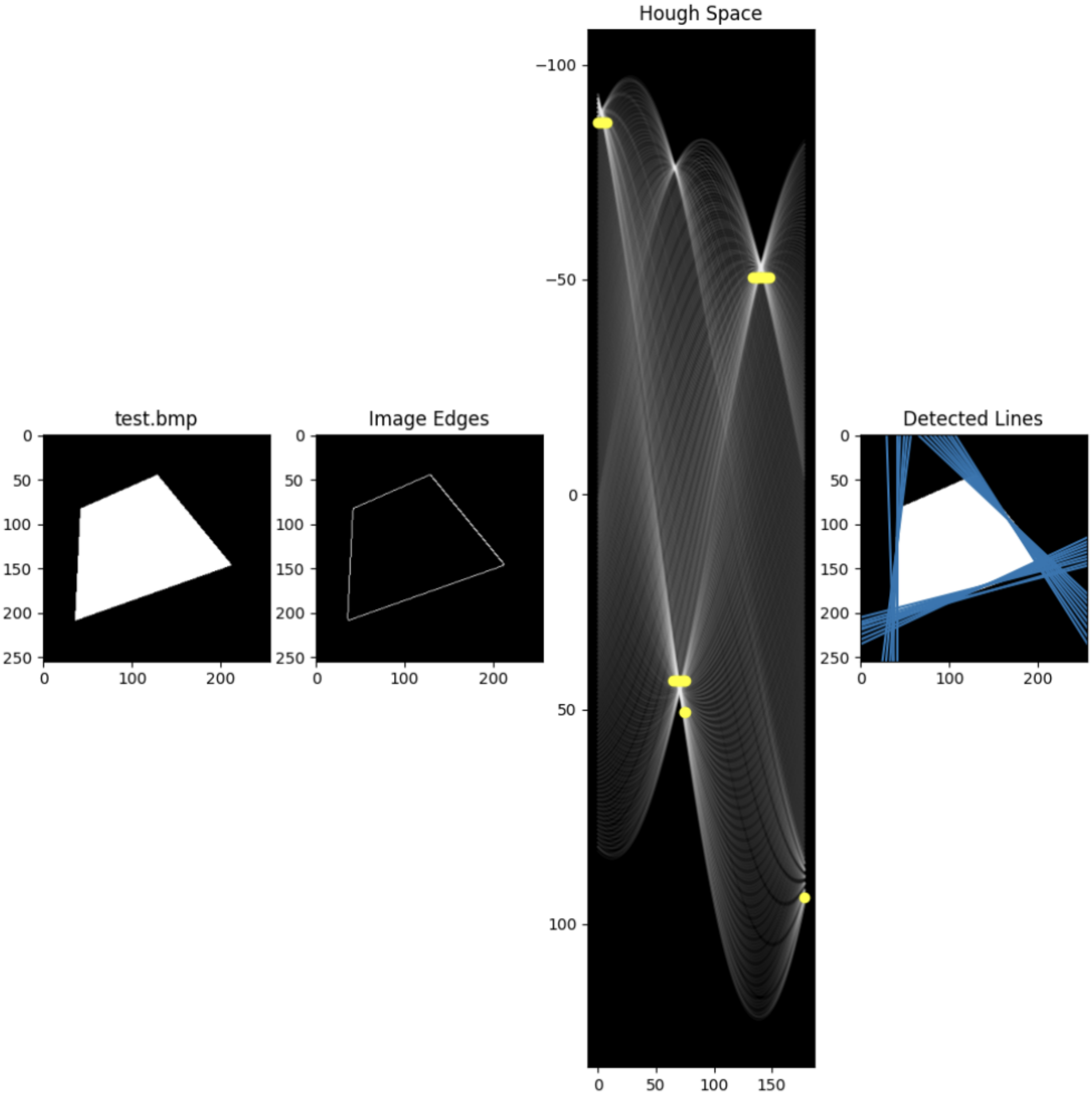

test2.bmp
Image Edges
Hough Space
Detected Lines
0
50
100
150
200
250
−100
−50
0
50
100
0
50
100
150



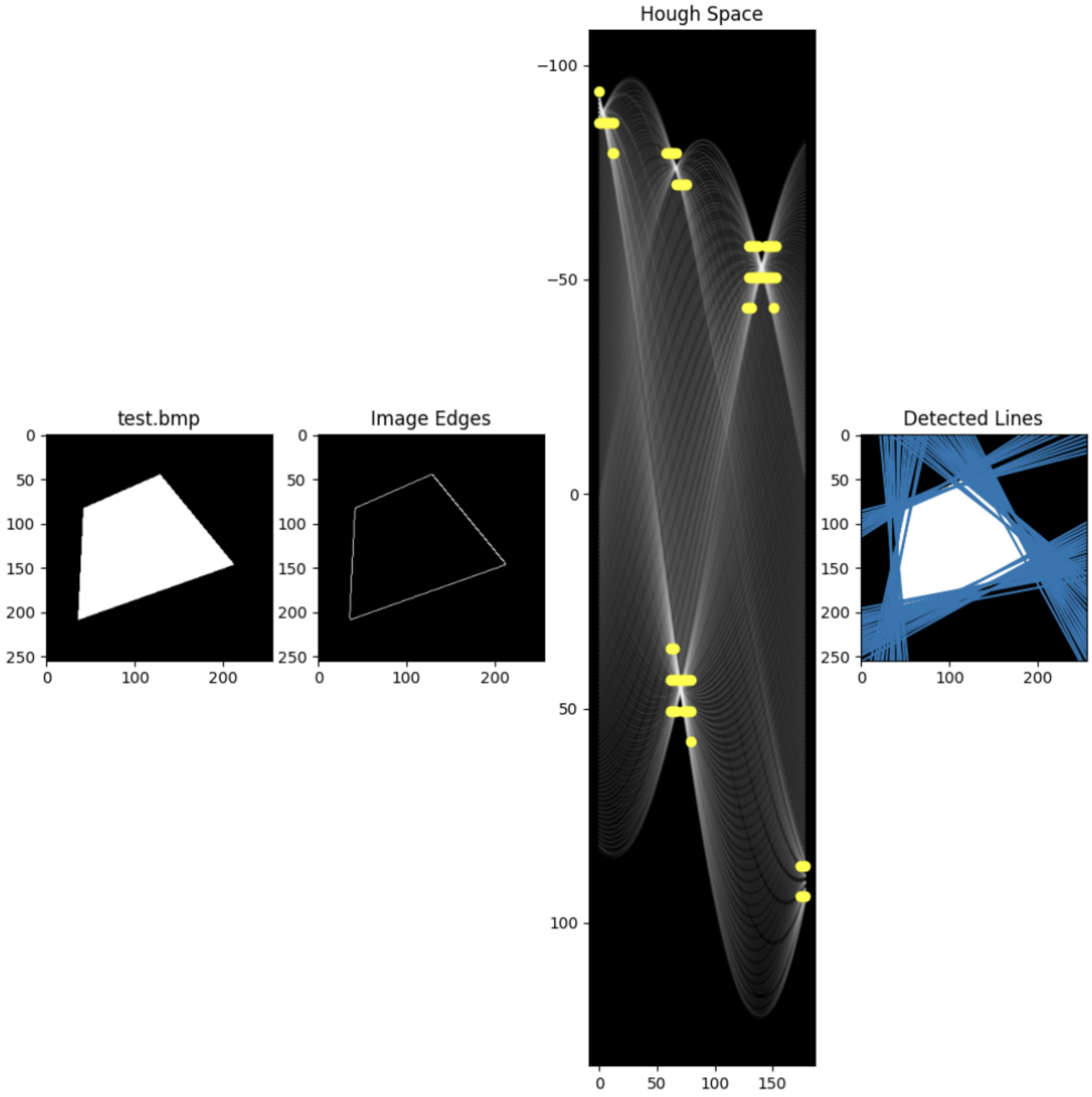
In the following example for the test.bmp image, I decreased the parameter space to only have 100 rho values and 100 theta values. When I maintained the threshold at 92 (as I had it set for the first line detection of the same image) you can see that less lines are identified.



Even after decreasing the threshold to 75, the smaller parameter space still does not catch the upper edge as a line. However, it can be seen that the important intersection is still there on the hough space graph, just not meeting the threshold.



Below is with the threshold at 50. The upper line is detected.


test.bmp
0
50
100
150
200
250
0
100
200
Image Edges
0
50
100
150
200
250
0
100
200
Hough Space
−100
−50
0
50
100
0
50
100
150
Detected Lines
0
50
100
150
200
250
0
100
200