HW6 Report

Problem Description

In this homework, we will be implementing the Hough transform technique to detect lines in images. The Hough transform involves several steps. First, we convert the image to grayscale and apply edge detection using a suitable algorithm such as Canny edge detection. Next, we create a parameter space where each point represents a possible line in the image. The parameter space is typically represented as a 2D array, where each element corresponds to a particular line. The x-axis represents the angle of the line, while the y-axis represents the distance from the origin of the image to the line. Next, we iterate through each edge pixel in the image and increment the corresponding elements in the parameter space. Finally, we apply a threshold to the parameter space to identify the lines in the image.

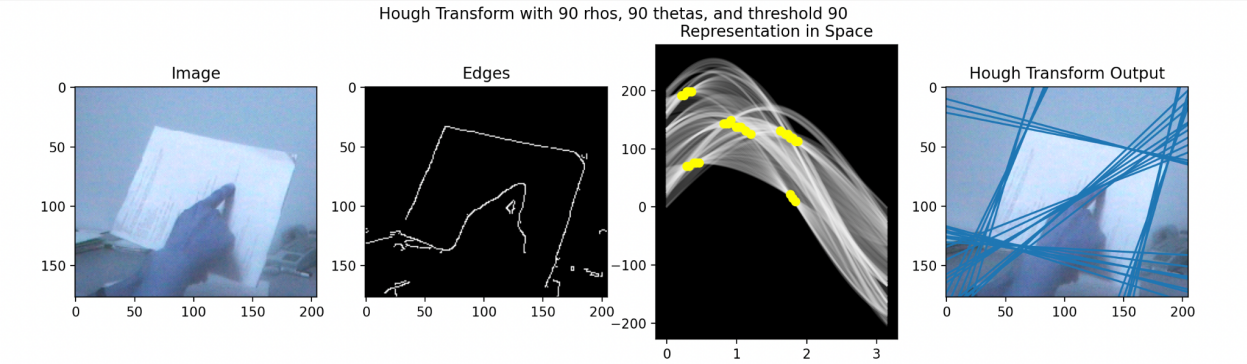
Proposed Algorithm

- We start by converting the image to grayscale and applying the canny edge detector, which will be passed to the function to return the hough transform, along with the hyperparameters, which include the number of thetas, number of rhos, and the gradient threshold to include the edge. To clarify, the number of thetas and the number of rhos
- We start by creating a grid that contains the different values of theta and rho based on the number of thetas and rhos desired, which is a parameter passed to the function. We also initialize the table for cosine and sine of the theta values that will be considered, as they will be needed later for computing rho when going from (x,y) to a line in polar coordinates with rho and alpha, and when recomputing the lines from the points in polar coordinates.
- After that, we go through the edge points and convert the points to a line in polar coordinates by calculating rho from cosine theta and sine theta, with the x,y coordinates of the edge point. The formula can be seen in the lecture slides. Based on the values we get for rho and theta, we update the “voting table” or the grid that contains the different values described in the point above.
- Finally, the code iterates through each pixel in the voting table and checks if it is above a threshold value (t_count). If the pixel is above the threshold value, it is considered as a potential line in the image. The code then calculates the parameters (rho and theta) for the line based on the indices of the pixel in the voting table and stores them in a list (all_rho and all_theta). The code also calculates the endpoints of the line in the image space and stores them in a list (lists). The endpoints are calculated using the rho and theta values and the sine and cosine of the theta value.
- I also implemented a section where I filter the endpoints to not be overlapping or really close as this is causing a generating of many lines that are really close to each other and intersection. Its value will be more obvious in the experiments.

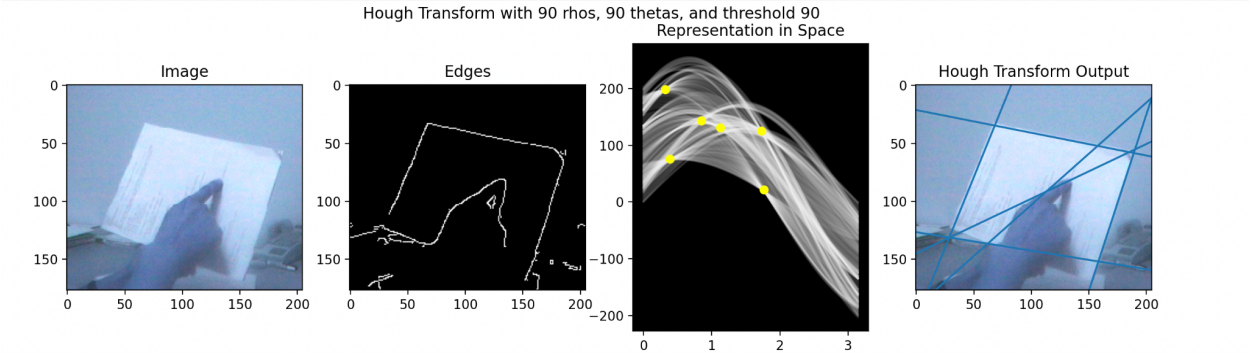
The filtering works by checking for all edge points above the threshold and checking if for every point, if any other point that is within a certain threshold (for theta and rho), we just keep the maximal value from the voting table. This helps us in filtering our results.

Experiments

We start by choosing num_thetas, num_rhos, and the threshold equal to 90.
We can start by seeing the results without the filtering, and then with the filtering

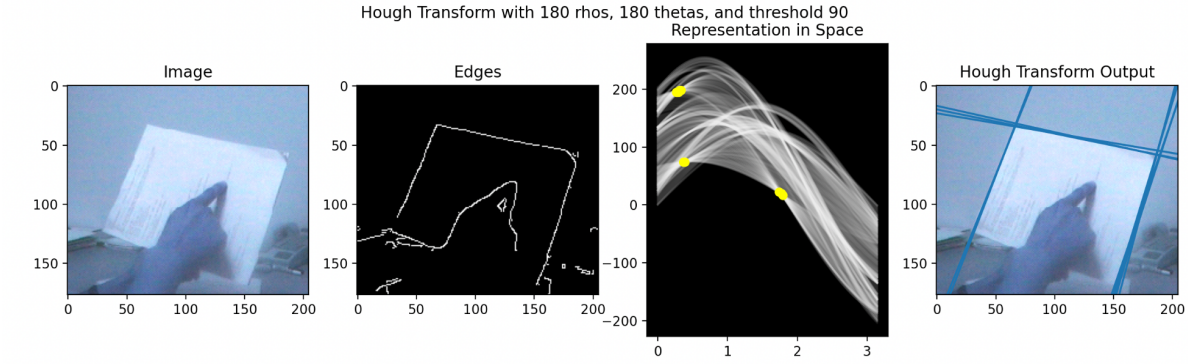


We can see that the results are logical as the lines are present where they should be. In the next image, we see the filtered results for the same hyperparameters.

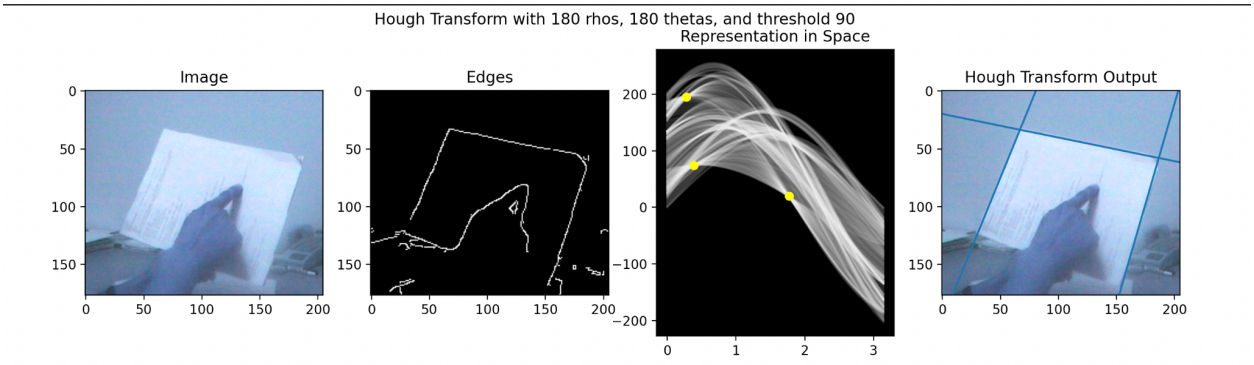


Here, we can see that the lines fit the image perfectly well. Also, we can see the relevant intersection points in the hough space and the selected points in yellow which correspond to lines in blue in the image.

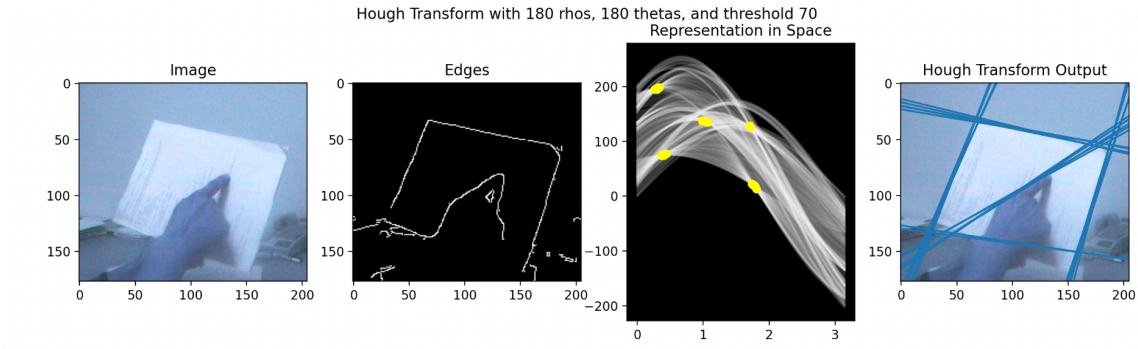
In the example above, we try with a less precise quantization.



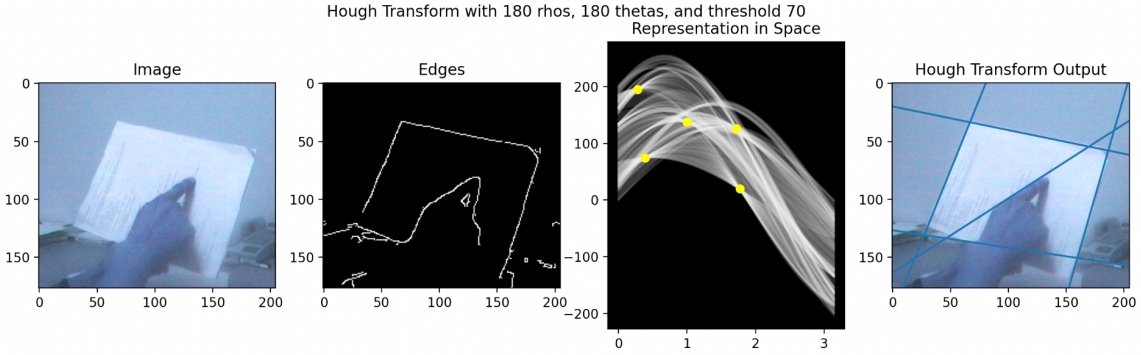
in space, which shows that we are missing a few lines that should have been detected, and the reason is that we raised the number of rhos and the number of thetas but kept the threshold constant. So by having a more divided voting table, it is harder to reach the same threshold specified above, therefore a smaller number of lines. The one above was done without filtering. The filtering is done on the photo below, which shows the proper lines with maximal votes:
Next thing, we are going to try to keep the number of rhos and thetas constant and change the



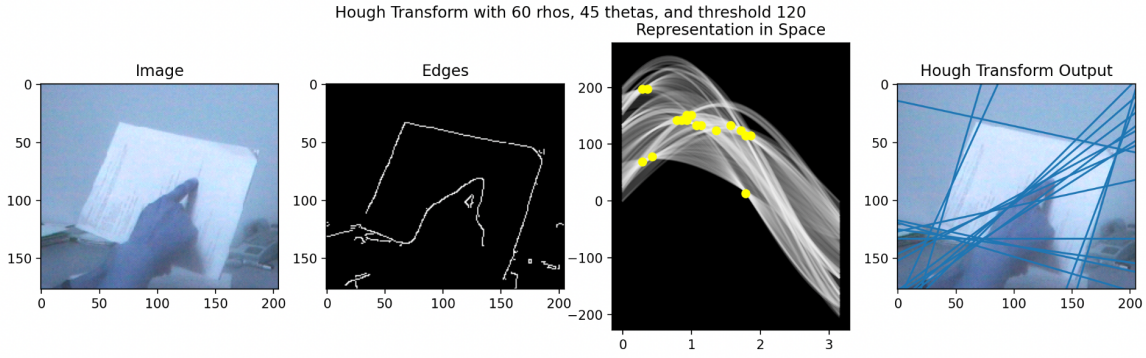
threshold:



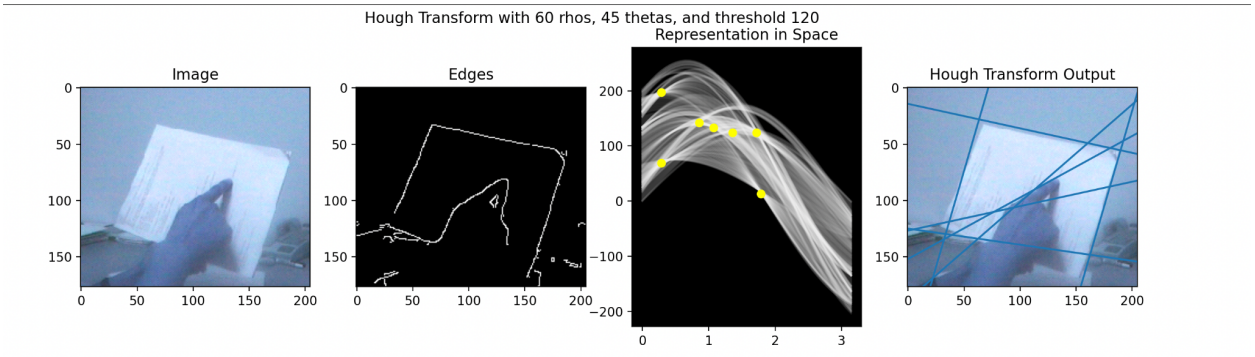
As expected, when we lower the threshold, we get the needed lines again, and because we have a finer quantization space, the lines are more accurate, and fall exactly where they should. This can be seen when the filtering is turned on:



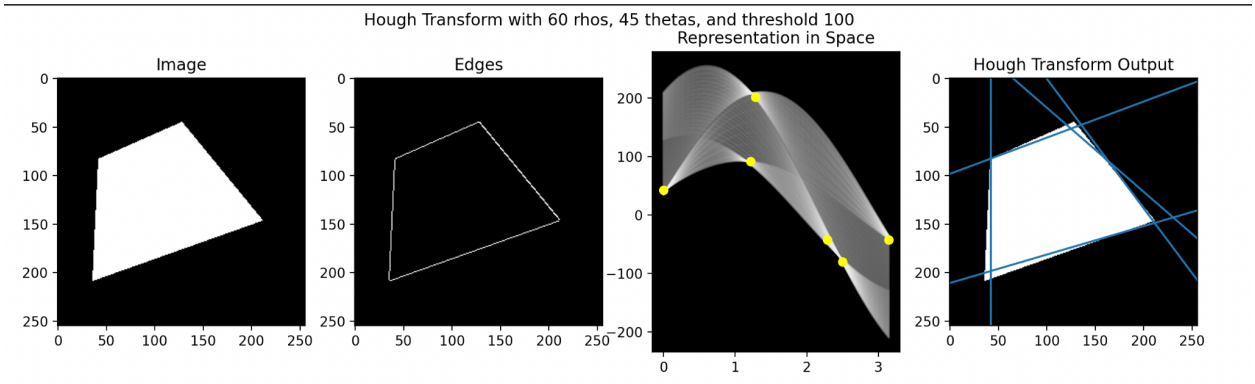
We are also going to try to lower the quantization space and have a higher threshold to see what would happen. We are also going to use different values for rho and theta:

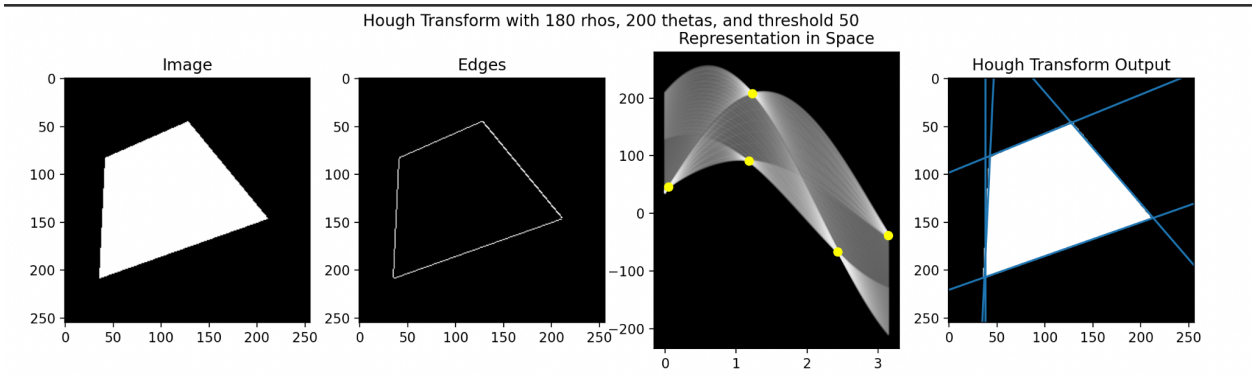


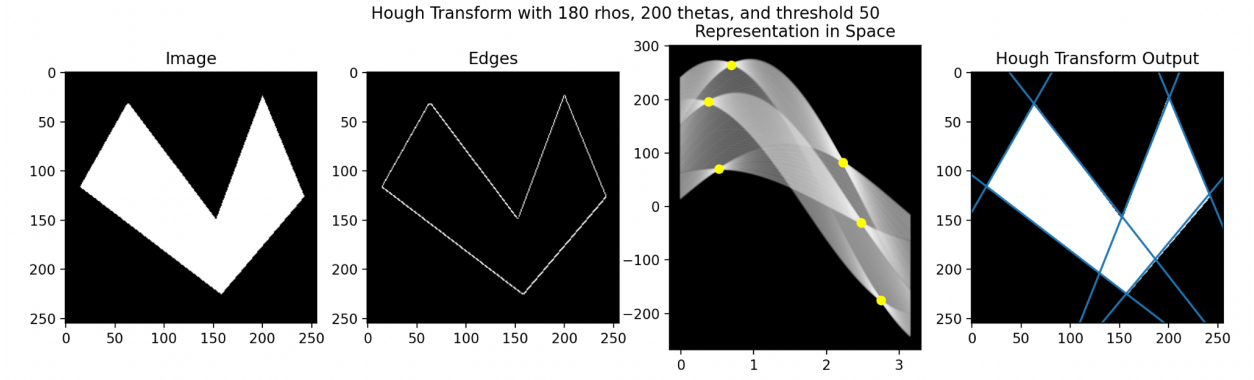
We can see the divergent lines which don't really fit the edges in the image. This is because we are using a smaller quantization space and it is causing these results: smaller quantization will cause the lines to be less fitting to the image and to be less accurate. This is even more obvious

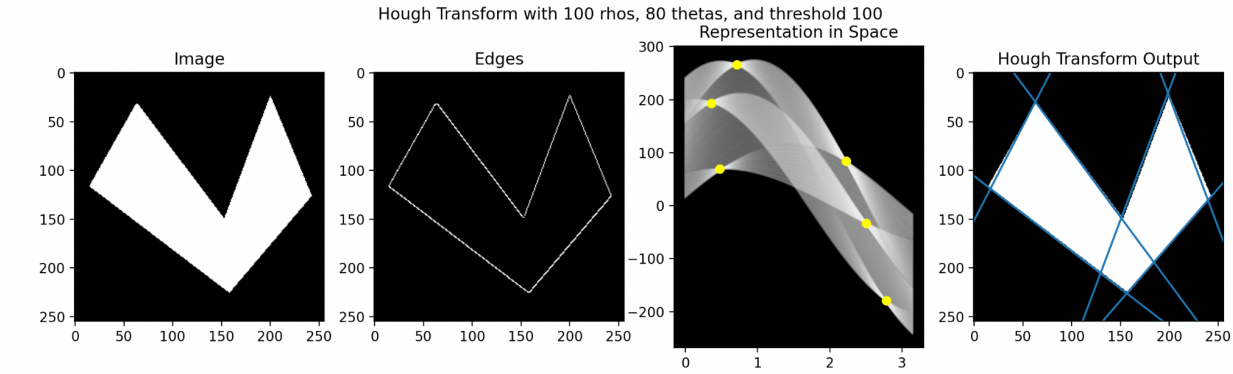


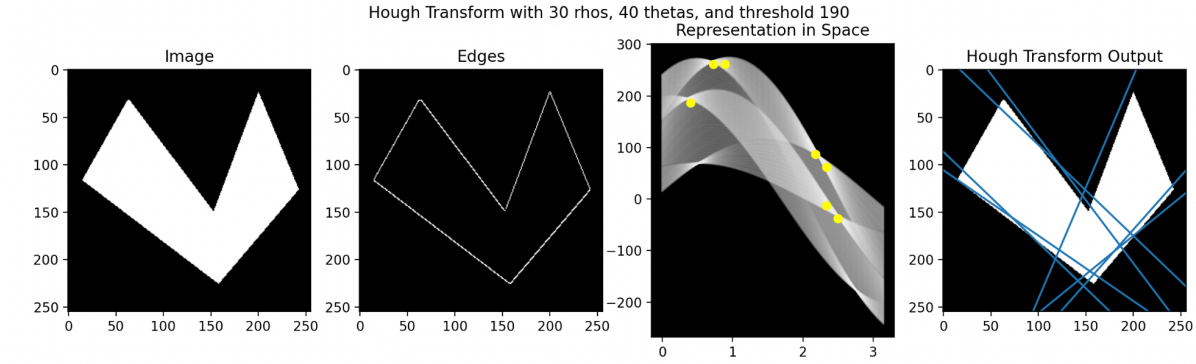
in the filtered version below. Also, it is seen that it is not necessary to have the same values for rhos and thetas. We will show a few other examples for the 2 different images we have. The same conclusions we got to above still hold. Starting with the filtered version:

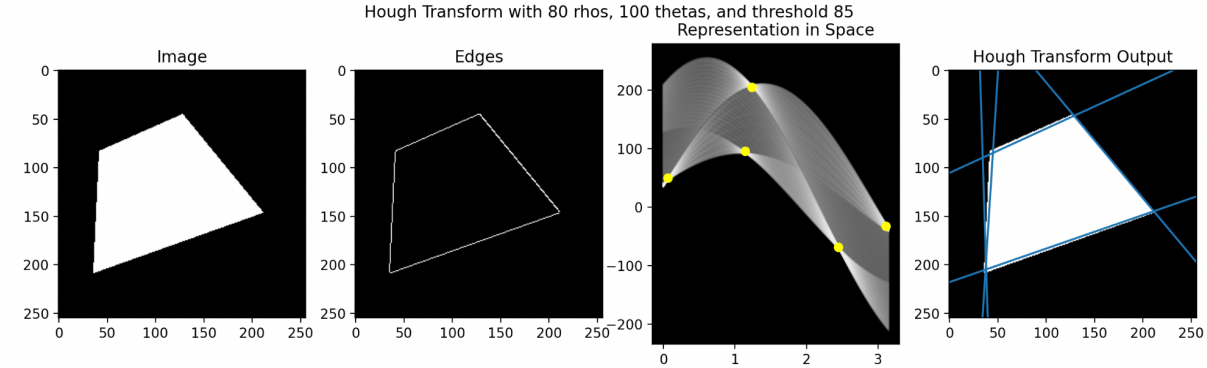

Hough Transform with 60 rhos, 45 thetas, and threshold 100
Image
Edges
Representation in Space
Hough Transform Output


Hough Transform with 180 rhos, 200 thetas, and threshold 50
Image
Edges
Representation in Space
Hough Transform Output

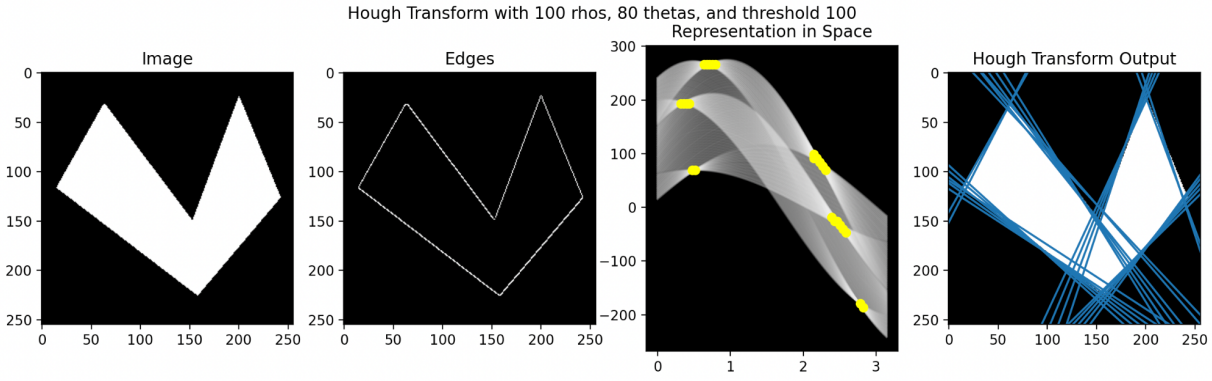

Hough Transform with 180 rhos, 200 thetas, and threshold 50
Image
Edges
Representation in Space
Hough Transform Output

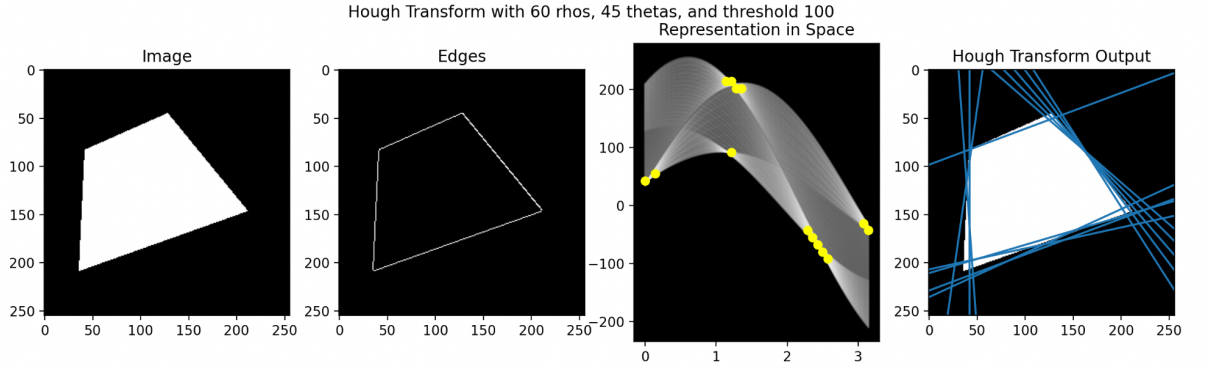


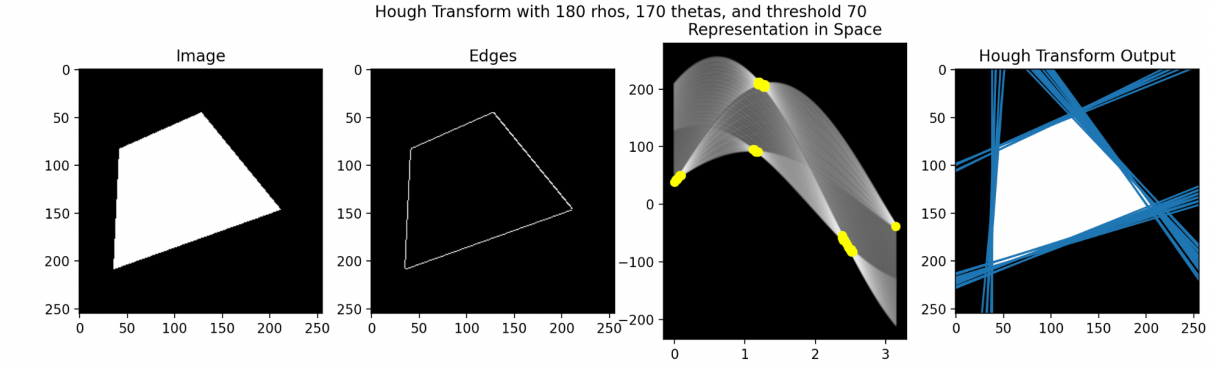


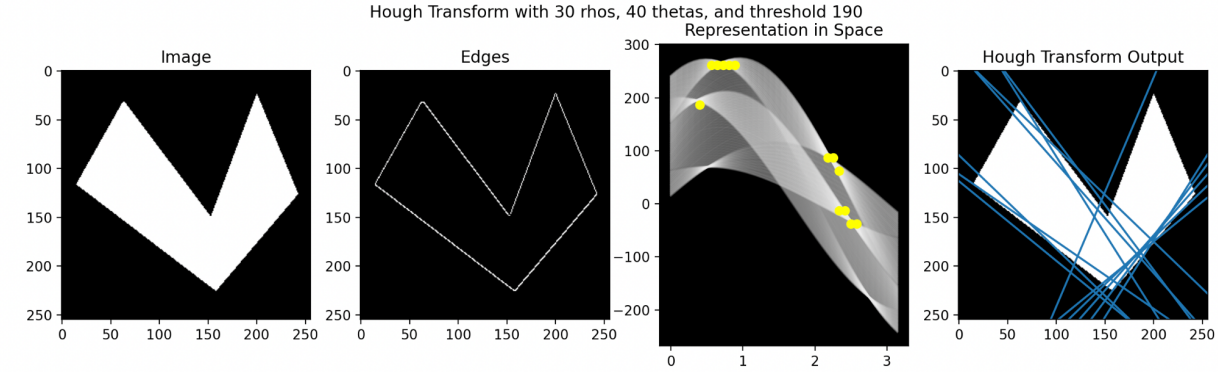


Followed by the unfiltered versions:


Hough Transform with 100 rhos, 80 thetas, and threshold 100
Image
Edges
Representation in Space
Hough Transform Output


Hough Transform with 60 rhos, 45 thetas, and threshold 100
Image
Edges
Representation in Space
Hough Transform Output


Hough Transform with 180 rhos, 170 thetas, and threshold 70
Image
Edges
Representation in Space
Hough Transform Output


Hough Transform with 30 rhos, 40 thetas, and threshold 190
Image
Edges
Representation in Space
Hough Transform Output