R eport of MP6

1. Hough Transform

It is used to isolate shapes from a image, commonly used in line, circle detection.

2. Algorithm Implementation

this mp focus on line detection, and whole program consist of 4 functions.

Edge_Detection: use canny edge detection to find edge

Parameter_Space: initial the space matrix and define the range of θ and ρ. Then iterate through the edge pixels, accumulate the θ and ρ calue in the m-c space.

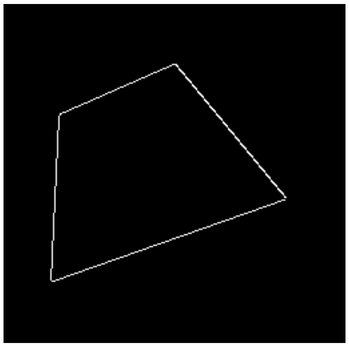
Intersection Detection: Use maximum and minimum filters to find local optical points in m-c space. Apply threshold filter to retain significant local maxima points

Line detection: use the maxima points list to correspond the parameters of the line, covert the points from m-c space back to x-y space, draw line in the original image.

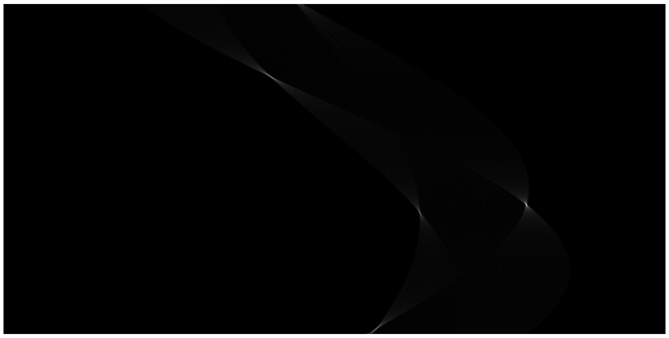
3. Result analysis

3.1 The images refer to edge image, parameter space, intersection points, and the detect line image:

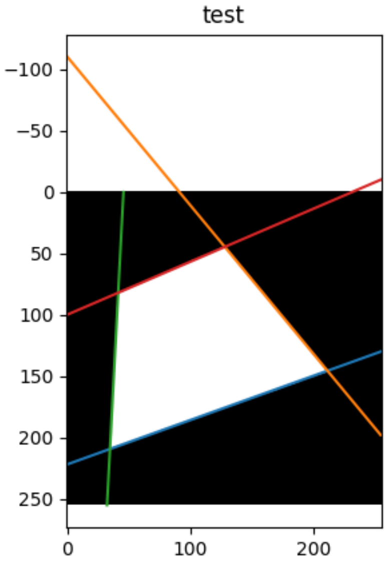
Test: for test.bmp, from para_space iamge we get 4 light points, combine with 4 white pixel in the intersection iamge, we can see this should have 4 lines in the final detect line image.



(edge)

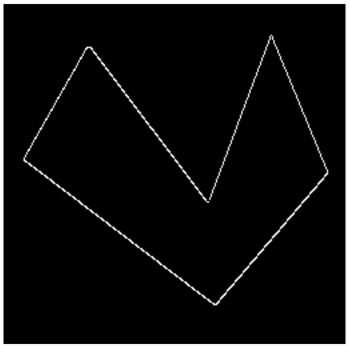


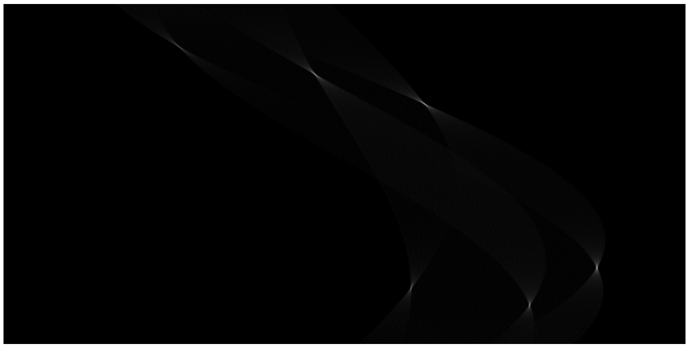
(para_space)



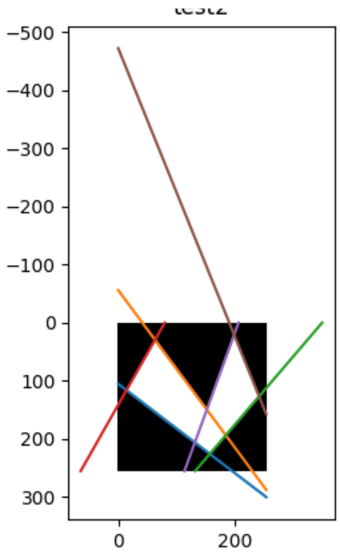
(intersection) (detect line)

Test2: for test2.bmp, from para_space iamge we get 6 light points, combine with 6 white pixel in the intersection iamge, we can see this should have 6 lines in the final detect line image.



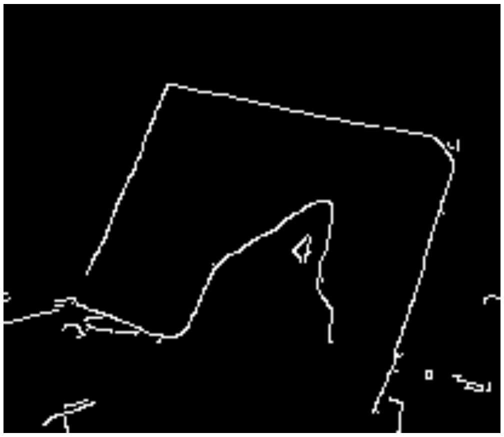


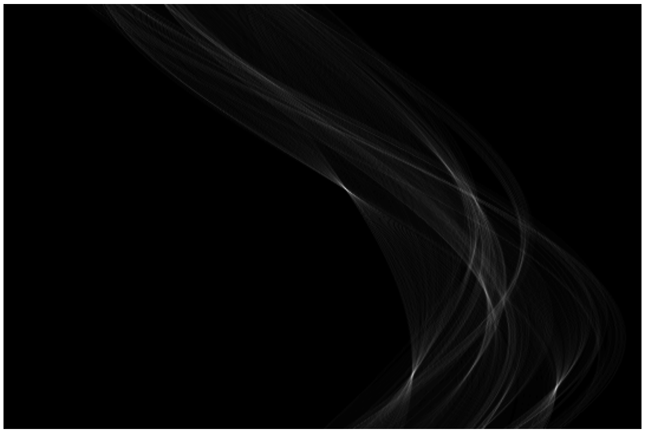
(edge) (para_space)



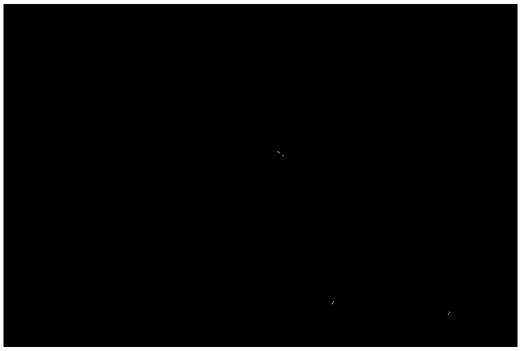
(intersection) (detect line)

Input: for input.bmp, from para_space iamge we get 5 light points, combine with 5 white pixel in the intersection iamge, we can see this should have 5 lines in the final detect line image. However, the performace for input.bmp is not so good, still need to update the steeing of θ and ρ and threshold.

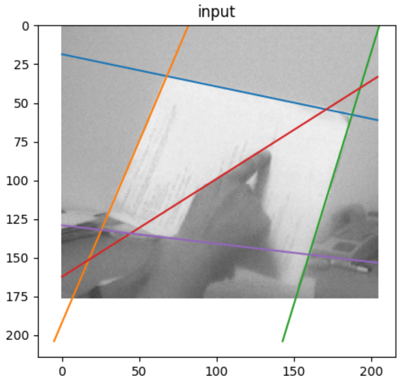




(edge) (para_space)



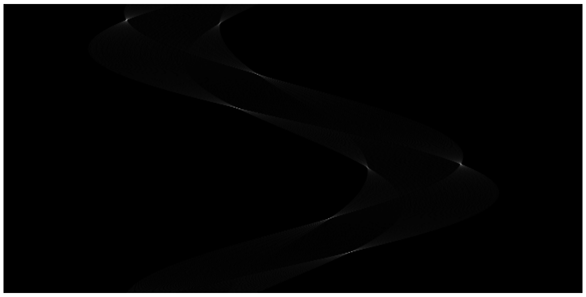
(intersection)



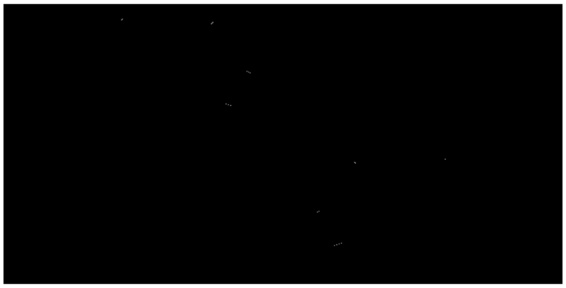
(detect line)

4. different quantization in the parameter space

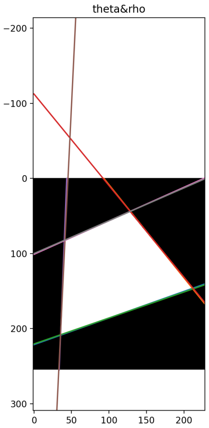
in the above test, I set θ range [-90° ,90°], ρ correspond [- √ (height^2 + width^2), √ (height^2 + width^2)]. When I change the value of them, I found, with the increase of them, there are more light spots and white pixels found in para_space image and intersection image. And the final detect lines fit the shape better. Take the test.bmp as example, we reset I set θ range [-180° ,180°], ρ correspond 2*[- √ (height^2 + width^2), √ (height^2 + width^2)].



(para_space)



(intersection)



from result, we see there are more lights spots, that that is because a larger value allows the system to detect more straight lines with different angles and more straight lines with different lengths. But processing more data points also requires greater computing resources.