Assignment 1

Berk Karaimer, 21827541
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
b21827541@cs.hacettepe.edu.tr

March 24, 2021

1 Introduction

In this assignment we are supposed to take an image and find it's edges with preferred method, use Hough transform on found edges to find license plate lines from the image. With license plate lines found, we are asked to calculate the IOU value with given xml files to see how our program performs.

To make this program work i started with reading the given image files. I choose the image that I wanted to work with from given data set. On the image I changed the colors to gray scale and used gaussianblur function to remove noise from it to get better result from edge finding method. For edge finding method, I choose to work with Canny edge since it performs better than Sobel and other methods with finding better edged image output. Canny edge gives less noisy image output and performs well. After getting the edged image, I tried to use Hough transform on them to find lines with threshold and Hough accumulator. But the found lines from the image turned out to be not only license plate but all the edges found over threshold. I could not find a better thresholds for the images, so the image with lines gives output with unwanted lines. Finally, I could not find a way to find only the license plate lines from whole image. So program only gives edged image, Hough space and lines over thresholds.

2 Experiment

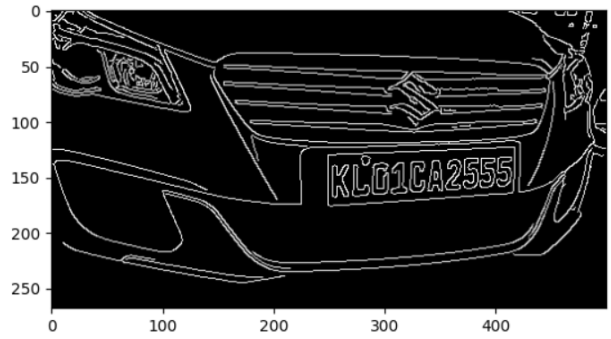
2.1 Part 1

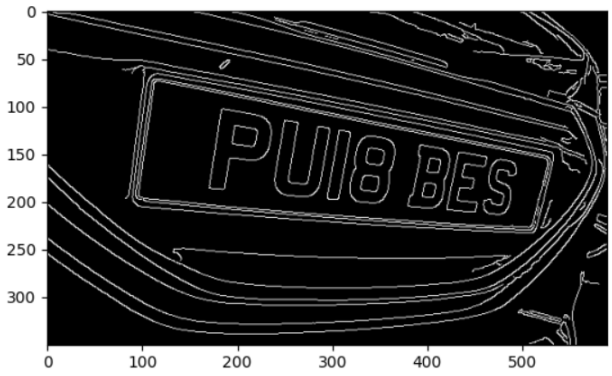
I choose to use Canny edge detection since it performs well when detecting edges with low error rate and catches edges as possible from the given picture. For parameters of canny function, I tried using Otsu's method for thresholds but it gave too many edges, so I tried to find low and high thresholds myself. First, i

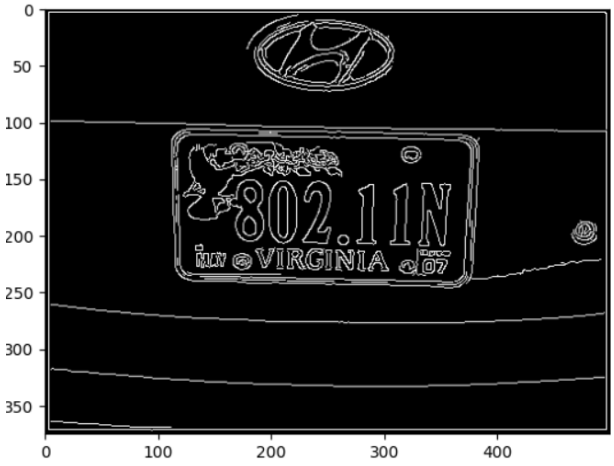
converted the colors of image to gray and used gaussianBlur function to remove noise. Next, to remove distortion from image i scaled it and used Canny edge detection as can be seen from following.

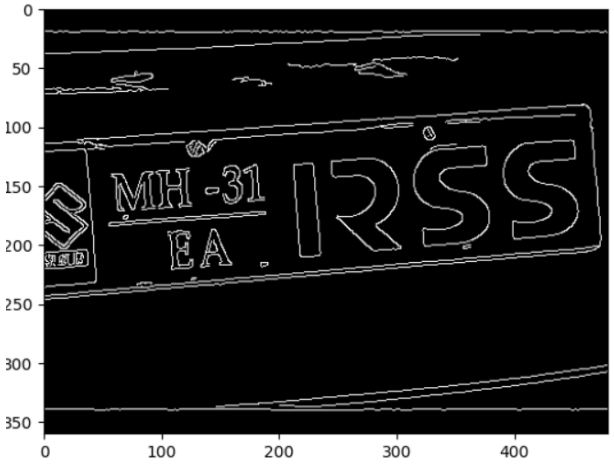
```
#gray scale convert
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
#noise remove
image = cv2.GaussianBlur(gray, (3, 3), 0)
image = (image*255).astype(np.uint8)
#canny edge detector
edged_image = cv2.Canny(image, 30, 250)
```

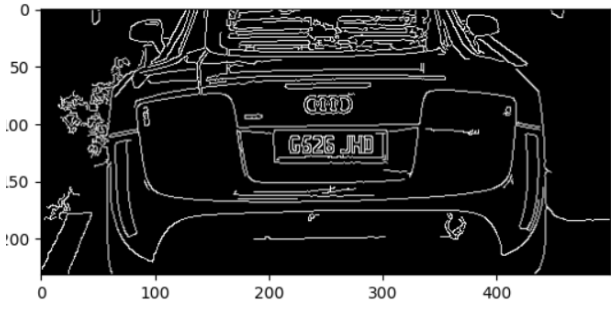
Here are 10 outputs of edged images:



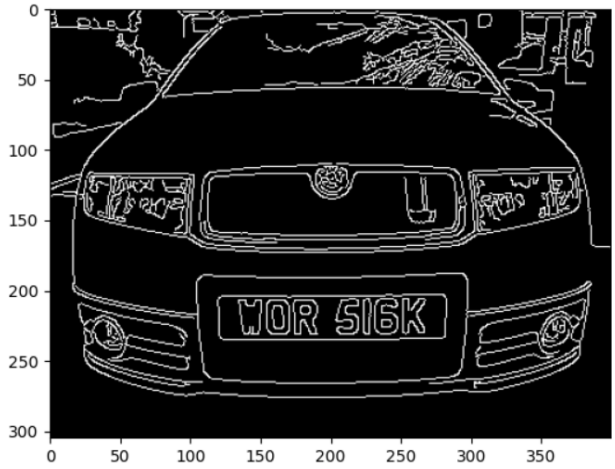


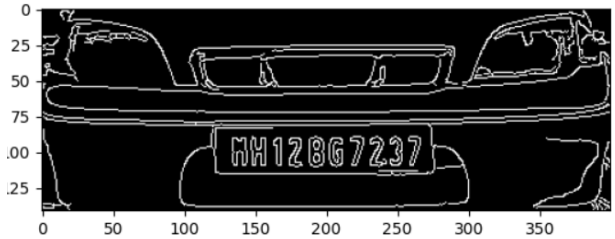

802.11N
VIRGINIA

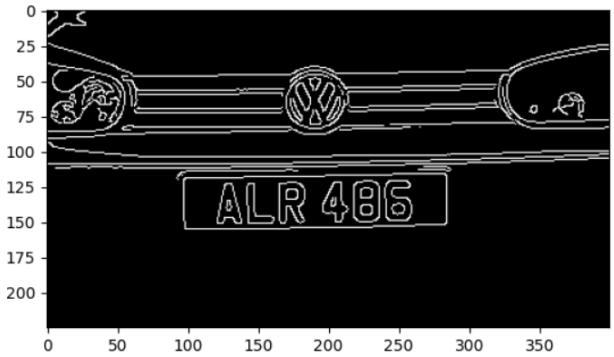

MH -31
EA
RSS

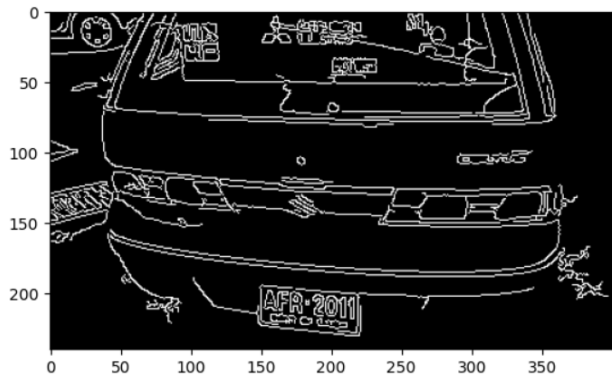

G626 JHD


MH01AY8866


MOR 516K


MH12BG7237


ALR 486


AFR 2011

2.2 Part 2

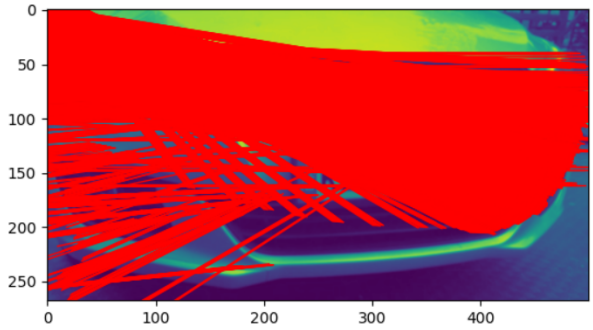
Before Hough transform, I clipped borders of the image by 5 pixel to get rid of unnecessary lines as following.

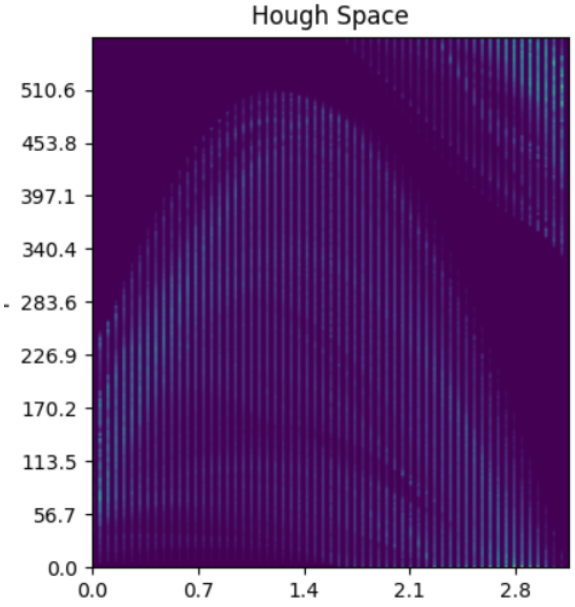
```
edged_image[0:borderLen, 0:leny] = 0
edged_image[lenx-borderLen:lenx, 0:leny] = 0
edged_image[0:lenx, 0:borderLen] = 0
edged_image[0:lenx, leny-borderLen:leny] = 0
```

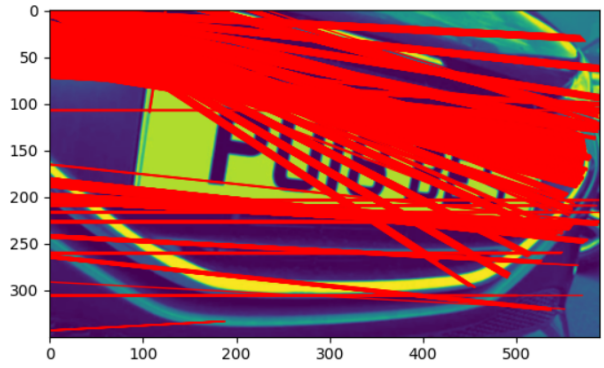
After, I assigned minimum and maximum values for rho and theta from image shape and created Hough space filled with zeros. With these done, i moved on to voting of Hough space and after Hough space added a threshold for list of lines.

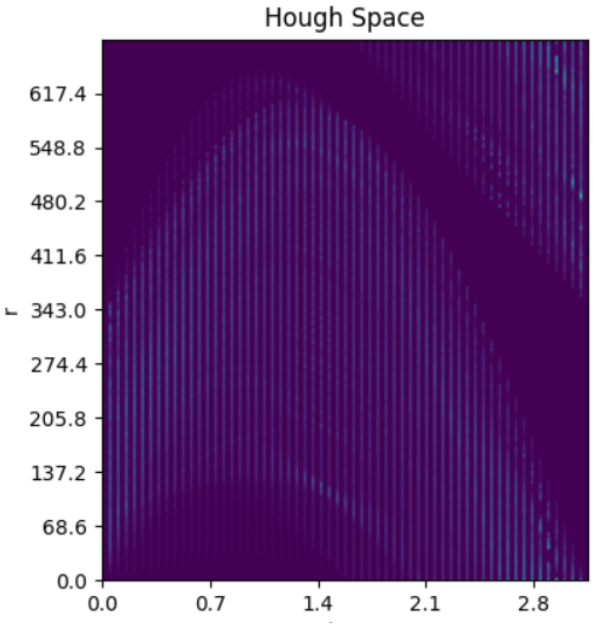
```
#read image values
    for i in range(x_max):
        for j in range(y_max):
            #check white pixels in edged
            if edged_image[i, j] == 255:
                #Hough transform for every theta value
                for idx_theta in range(0, theta_dim, 3):
                    #theta to rad
                    theta = idx_theta * theta_max / theta_dim
                    r = i * math.cos(theta) + j * math.sin(theta)
                    ir = int(r_dim * r / r_max)
                    hough_space[ir, idx_theta] = hough_space[ir, idx_theta]
                    #take values higher than treshold
                    if hough_space[ir, idx_theta] > img_shape[0]/1.6 \
                    and idx_theta not in list_lines \
                    and ir not in list_lines:
                        list_lines.append([r, theta, j])
```

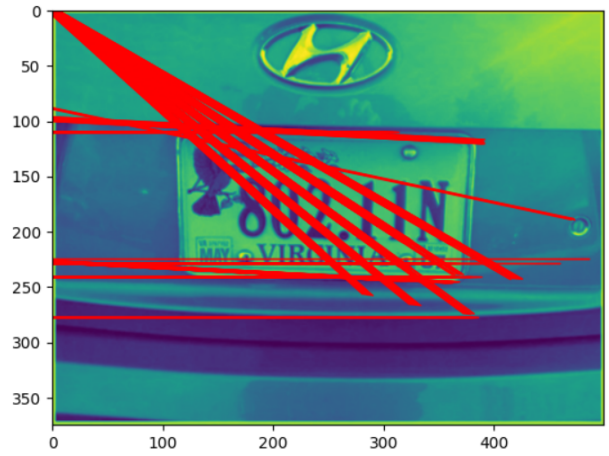
So, at the end of this part I had lines and Hough space. But I could not find a good threshold for the Hough transform and the image ended up with unwanted lines. Finally, i tried to locate the plate by finding intersections but could not manage it and this was where i ended up. Here are the output images of Hough :

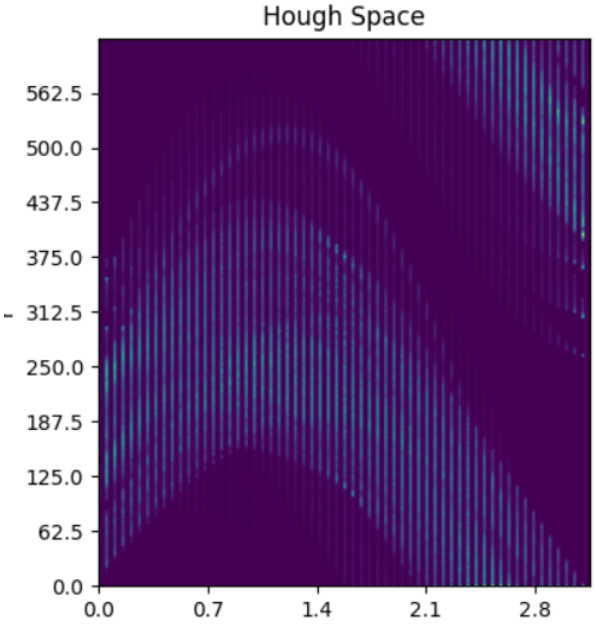

0
50
100
150
200
250
0
100
200
300
400

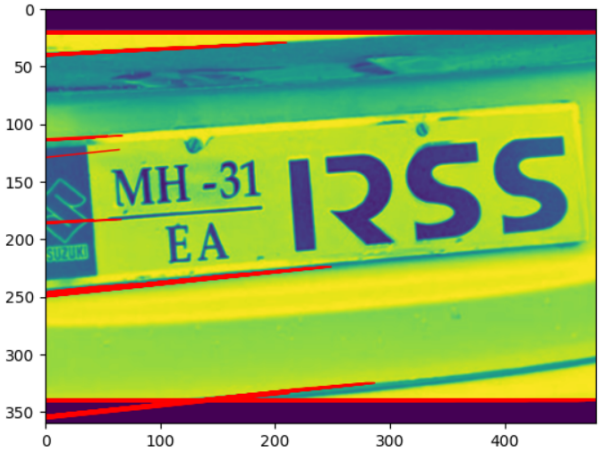

Hough Space
510.6
453.8
397.1
340.4
283.6
226.9
170.2
113.5
56.7
0.0
0.0
0.7
1.4
2.1
2.8

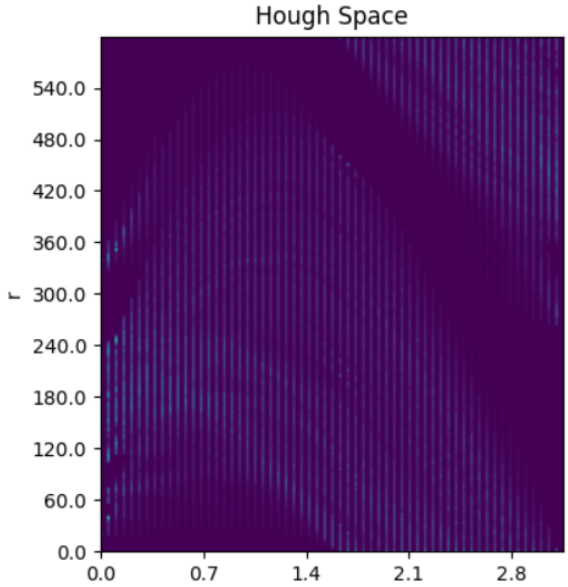

0
50
100
150
200
250
300
0
100
200
300
400
500

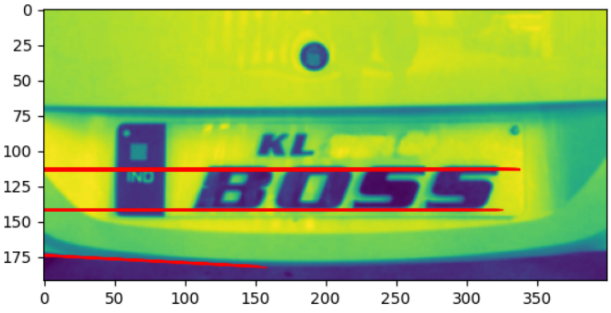

Hough Space
617.4
548.8
480.2
411.6
343.0
274.4
205.8
137.2
68.6
0.0
r
0.0
0.7
1.4
2.1
2.8

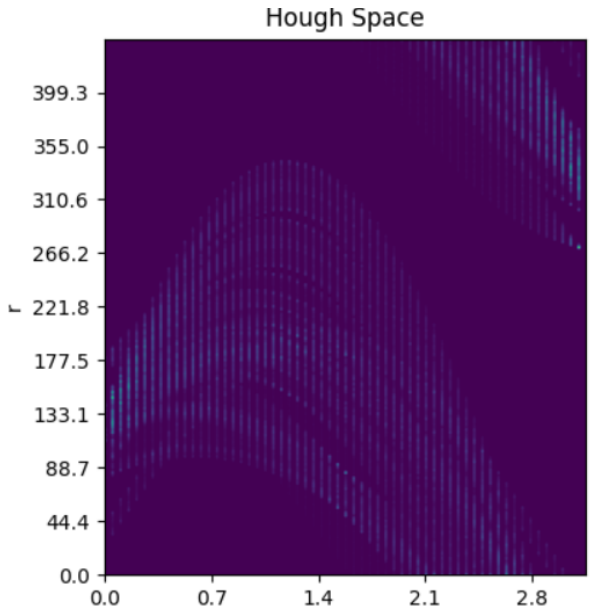

0
50
100
150
200
250
300
350
0
100
200
300
400


Hough Space
562.5
500.0
437.5
375.0
312.5
250.0
187.5
125.0
62.5
0.0
0.0
0.7
1.4
2.1
2.8


0
50
100
150
200
250
300
350
0
100
200
300
400
MH -31
EA
IRSS


Hough Space
r
540.0
480.0
420.0
360.0
300.0
240.0
180.0
120.0
60.0
0.0
0.0
0.7
1.4
2.1
2.8


0
25
50
75
100
125
150
175
0
50
100
150
200
250
300
350
KL
BOSS


Hough Space
r
399.3
355.0
310.6
266.2
221.8
177.5
133.1
88.7
44.4
0.0
0.0
0.7
1.4
2.1
2.8

3 Conclusion

I have learned to use Hough Transformation algorithm and what it's uses are. Since I could not detect the license plate lines from all the lines I have found, I could not include IOU scores for the program. I believe that thresholds from my Hough transformation is not working well for these images and with that the program fails to give proper lines for the license plates. Another reason might be that while looking for theta values the range is incremented by 3, so it might be skipping some important lines, but since the threshold values i gave did not work well, without that increment, image gets full of lines.