IMAGE PROCESSING PROJECT REPORT

(CSE 464)

Introduced by:

Sec (3)

Names:	**Codes:**
1-Mohamed Hassan Abd El Aleem	1701177
2-Mohamed Sayed Abbas	1701205
4-Mohamed Khaled Anwer	1701185

1- Python code

```
import matplotlib.pylab as plt
import cv2
import numpy as np

def region_of_interest(img, vertices):
    mask = np.zeros_like(img)
    #channel_count = img.shape[2]
    match_mask_color = 255
    cv2.fillPoly(mask, vertices, match_mask_color)
    masked_image = cv2.bitwise_and(img, mask)
    return masked_image

def drow_the_lines(img, lines):
    img = np.copy(img)
    blank_image = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

    for line in lines:
        for x1, y1, x2, y2 in line:
            cv2.line(blank_image, (x1,y1), (x2,y2), (0, 255, 0), thickness=10)

    img = cv2.addWeighted(img, 0.8, blank_image, 1, 0.0)
    return img

image = cv2.imread('road.jpg')
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
print(image.shape)
height = image.shape[0]
width = image.shape[1]
region_of_interest_vertices = [
    (0, height),
    (width/2, height/2),
    (width, height)
]
gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
canny_image = cv2.Canny(gray_image, 100, 200)
cropped_image = region_of_interest(canny_image,
                np.array([region_of_interest_vertices], np.int32),)
lines = cv2.HoughLinesP(cropped_image,
                        rho=6,
                        theta=np.pi/180,
                        threshold=160,
                        lines=np.array([]),
                        minLineLength=40,
                        maxLineGap=25)
image_with_lines = drow_the_lines(image, lines)
plt.imshow(image_with_lines)
plt.show()
```

https://gist.github.com/pknowledge/8933224beea63ffd818f72da76b18f3e

2- Code process

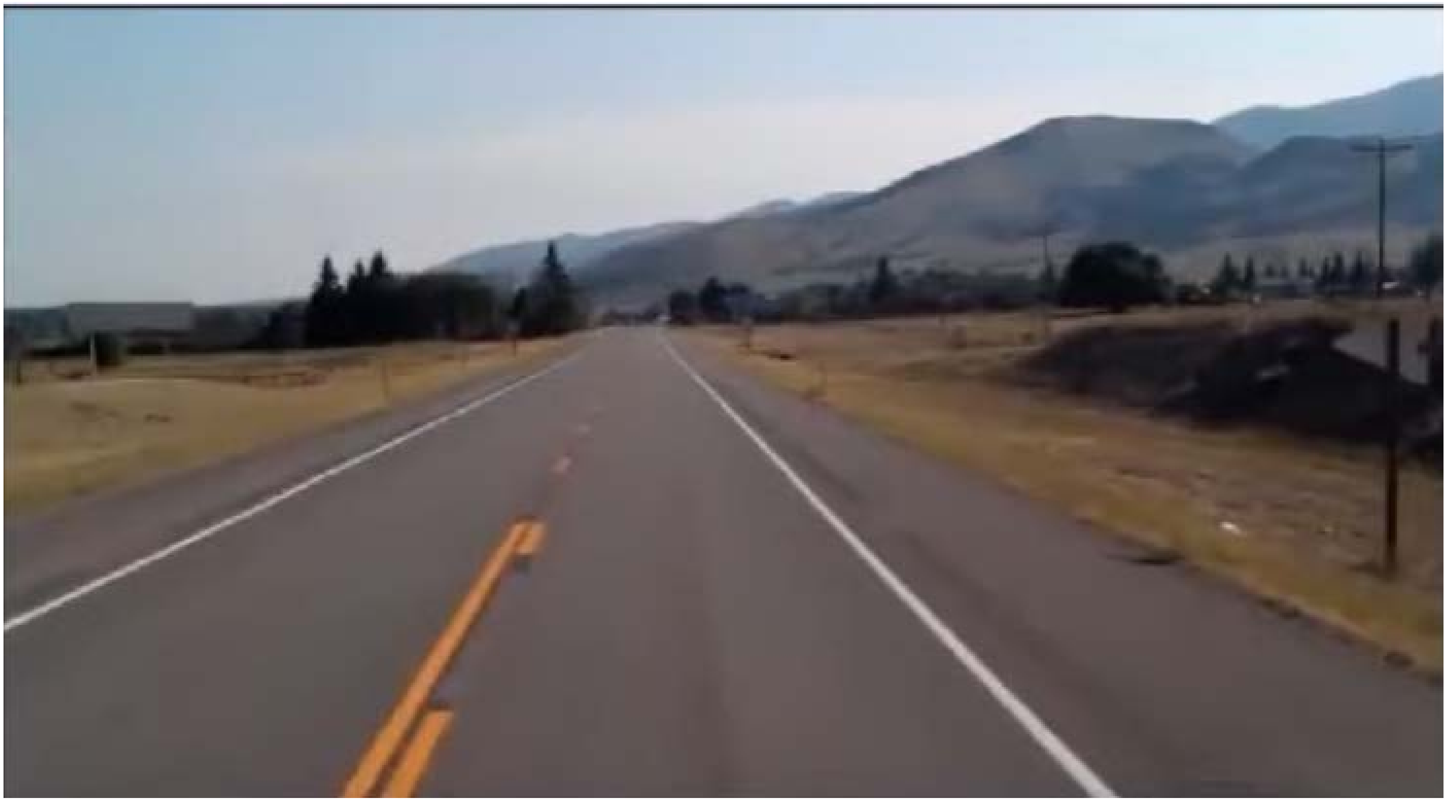
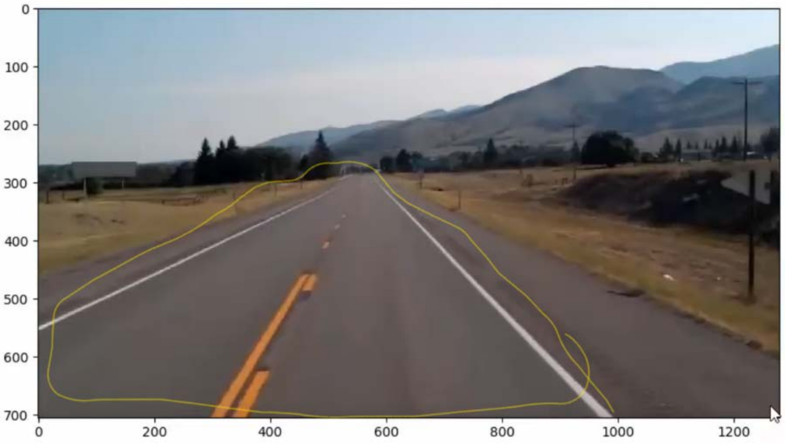


Figure 1:sample road image

1. sample image import

first we would import the test image ‘road.jpg’ to begin the processing and to define the image dimensions to define the **region of interest**



after defining the region of interest after obtaining the width and Height we will mask every other unneeded elements in the image

Figure 2:defining the region of interest

using (mask = np.zeros_like(img)) and defining the channel count and color and fill everything other that the region of interest in black

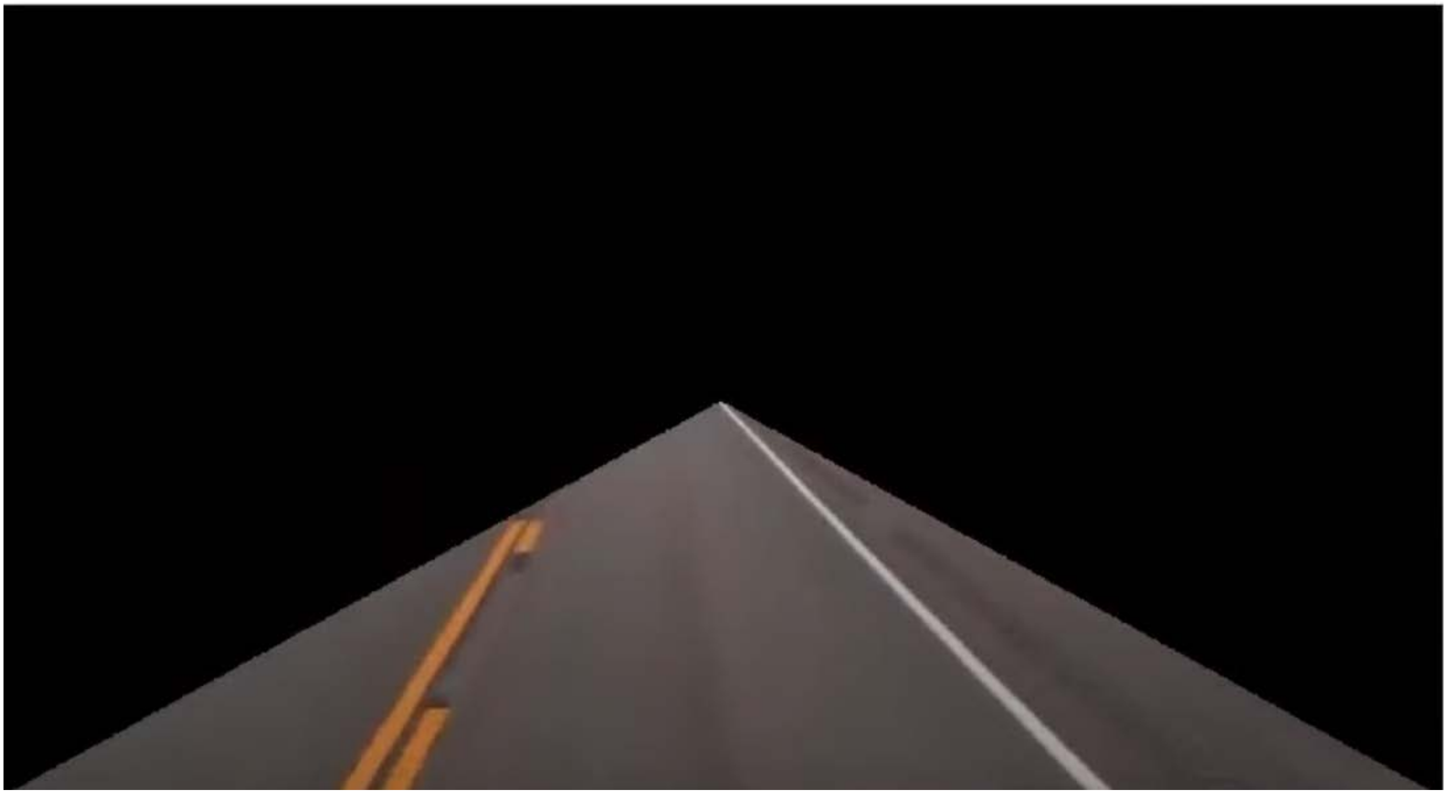


Figure 3:Masked image

2. Canny Edge detection

Then we will convert our image from RGB to Grey scale and pass it to Canny detecting method resulting as follows

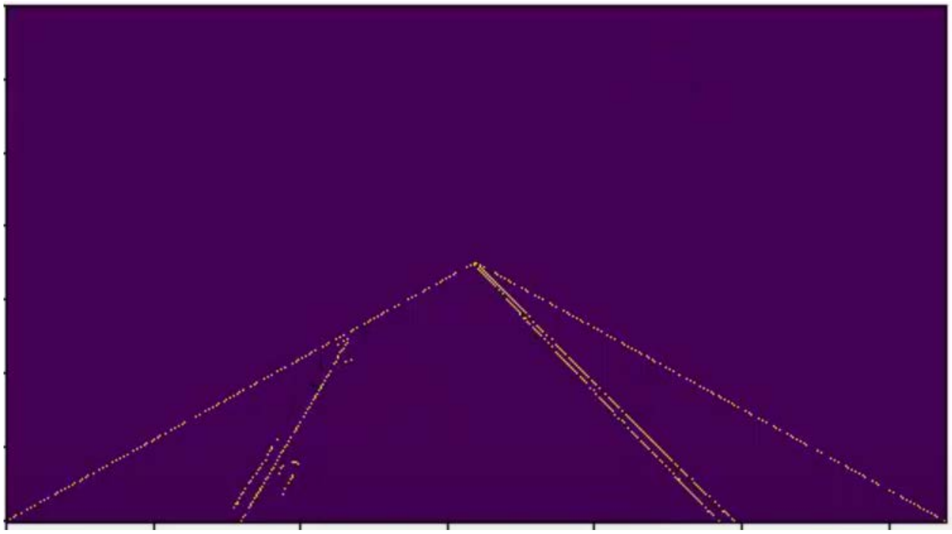


Figure 4:Canny Edge result

But as shown beside the road lanes the region of interest edges are also shown which we have no interest in

And to solve this problem we can use the canny edge detection before masking the region of interest

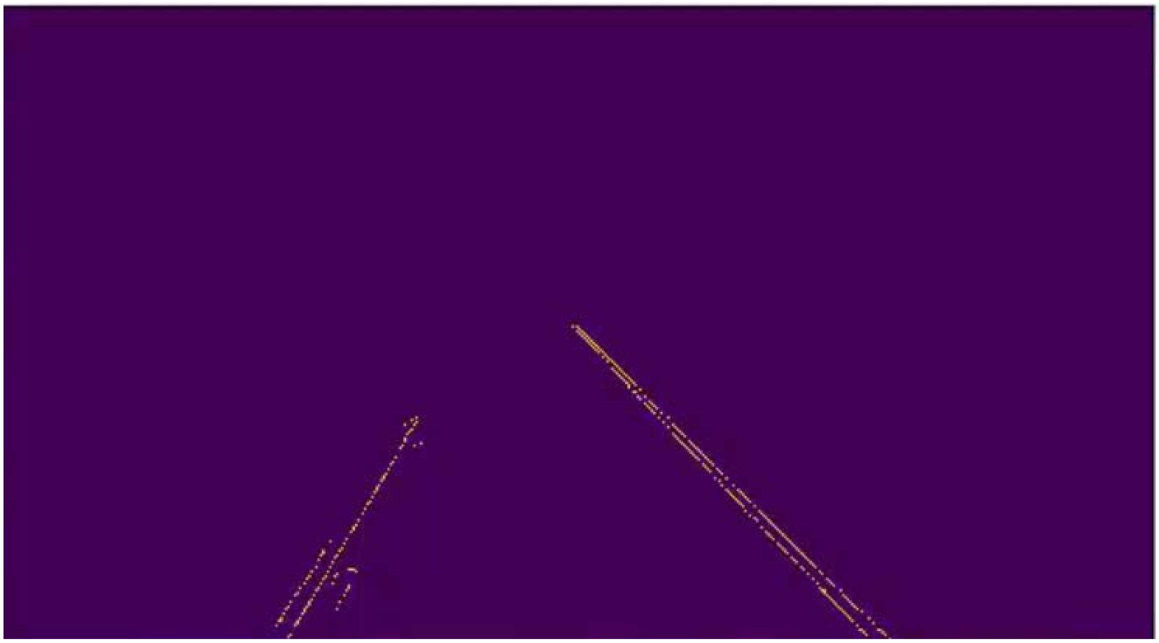


Figure 5:corrected Edge detection image

3. Drawing the lines using HLT

Using the Hough line transform method which requires passing the Edge detection image , the Rho = 6 , the value of theta = np.pi/60 , threshold = 160, lines = none(by default),minimum line length = 40 and the maxlinegap= 25) and then merge the result with the original image

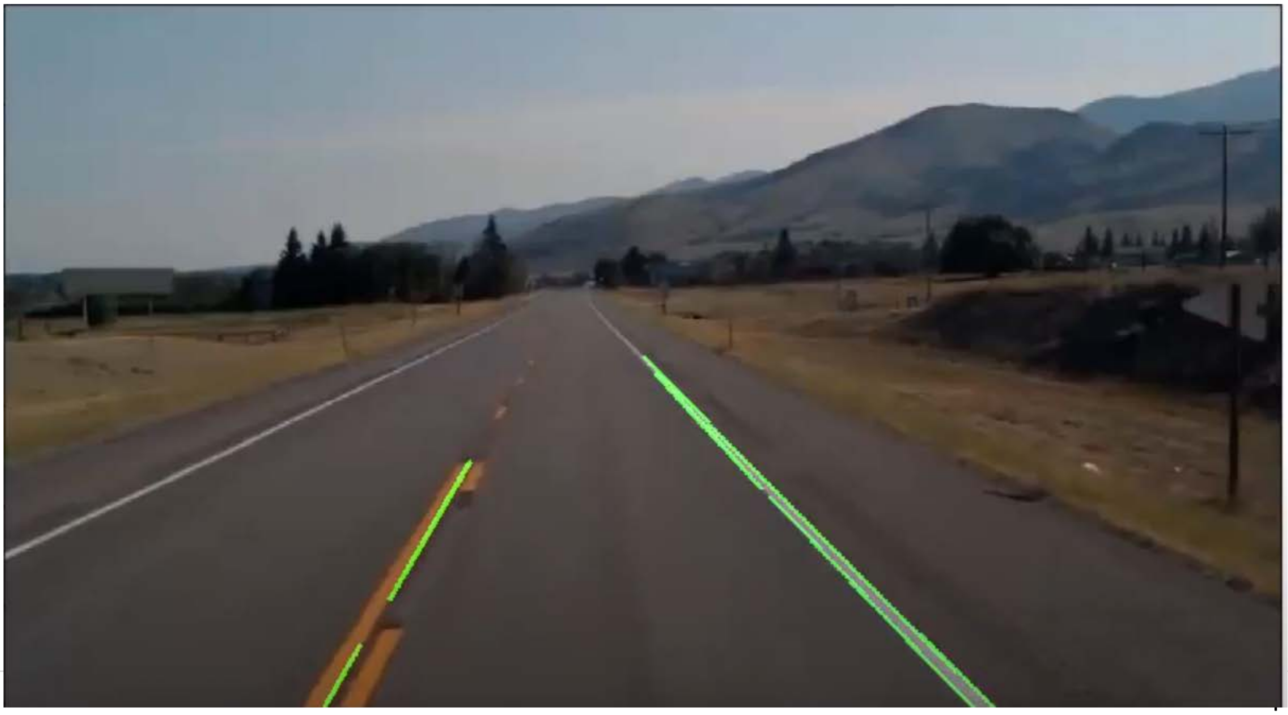


Figure 6:the end result of the detected lines