#3. Code Implementation for Lane detection through Hough Transform

import math

import cv2

import numpy as np

import matplotlib.pyplot as plt

def display\_lines(image,lines):

line\_image=np.zeros\_like(image)

if lines is not None:

for line in lines:

x1,y1,x2,y2=line.reshape(4)

cv2.line(line\_image,(x1,y1),(x2,y2),(255,0,0),10)

return line\_image

def make\_coordinates(image,line\_parameters):

slope,intercept=line\_parameteres

y1=image.shape[0]

y2=int(y1\*(3/5))

x1=int((y1-intercept)/slope)

x2=int((y2-intercept)/slope)

return np.array([x1,y1,x2,y2])

def average\_slope\_intercept(image,lines):

left\_fit=[]

right\_fit=[]

for line in lines:

x1,y1,x2,y2=line.reshape(4)

parameters=np.polyfit((x1,x2),(y1,y2),1)

slope=parameters[0]

intercept=parameters[1]

if slope<0:

left\_fit.append((slope,intercept))

else:

right\_fit.append((slope,intercept))

left\_fit\_avg=np.average(left\_fit,axis=0)

right\_fit\_avg=np.average(right\_fit,axis=0)

left\_line=make\_coordinates(image,left\_fit\_avg)

right\_line=make\_coordinates(image,right\_fit\_avg)

return np.array([left\_line,right\_line])

def region\_of\_interest(canny\_img):

height=canny\_img.shape[0]

polygons=np.array([

[(0,height),(canny\_img.shape[1],height),(canny\_img.shape[1]//2,250)]

])

mask=np.zeroes\_like(canny\_img)

cv2.fillPoly(mask,polygons,255)

## Bitwise & b/w mask and canny image,to show only region of interest traced

# by the polygon contour of the mask

masked\_img=cv2.bitwise\_and(canny\_img,mask)

return masked\_img

def lane\_detection(img):

## Color image to GrayScale image conversion

gray\_img = cv2.cvtColor(img, cv2.COLOR\_RGB2GRAY)

## Gaussian Blur

# Higher the kernel, the more blur the outcome image will be.

kernel\_size = 5

gauss\_img = cv2.GaussianBlur(gray\_img,(kernel\_size, kernel\_size), 0)

## Canny Edge Detection : It is used to detect boundaries of an image, through the gredients of the image

low\_threshold, high\_threshold = [200, 300]

canny\_img = cv2.Canny(gauss\_img, low\_threshold, high\_threshold)

cropped\_img=region\_of\_interest(canny\_img)

## Hough Transform: to connect the dots of images by transporting the images to the Parameter Space

# We have taken polar coordinates(rho,theta),in which we searched for intersecting lines.

lines = cv2.HoughLinesP(cropped\_img, rho=1, theta=math.pi/180,

threshold=15, np.array([]),

minLineLength=30,

maxLineGap=40)

averaged\_lines=average\_slope\_intercept(img,lines)

line\_image=display\_lines(img,averaged\_lines)

combo\_image=cv2.addWeighted(img,0.8,line\_image,1,1)

return combo\_image