Title-Real time lane detector using python.

Coding:

import matplotlib.pyplot as plt

import numpy as np

import cv2

import os

import matplotlib.image as mpimg

from moviepy.editor import VideoFileClip

import math

def interested\_region(img, vertices):

if len(img.shape) > 2:

mask\_color\_ignore = (255,) \* img.shape[2]

else:

mask\_color\_ignore = 255

cv2.fillPoly(np.zeros\_like(img), vertices, mask\_color\_ignore)

return cv2.bitwise\_and(img, mask)

def lines\_drawn(img, lines, color=[255, 0, 0], thickness=6):

global cache

global first\_frame

slope\_l, slope\_r = [],[]

lane\_l,lane\_r = [],[]

α =0.2

for line in lines:

for x1,y1,x2,y2 in line:

slope = (y2-y1)/(x2-x1)

if slope > 0.4:

slope\_r.append(slope)

lane\_r.append(line)

elif slope < -0.4:

slope\_l.append(slope)

lane\_l.append(line)

#2

img.shape[0] = min(y1,y2,img.shape[0])

# to prevent errors in challenge video from dividing by zero

if((len(lane\_l) == 0) or (len(lane\_r) == 0)):

print ('no lane detected')

return 1

#3

slope\_mean\_l = np.mean(slope\_l,axis =0)

slope\_mean\_r = np.mean(slope\_r,axis =0)

mean\_l = np.mean(np.array(lane\_l),axis=0)

mean\_r = np.mean(np.array(lane\_r),axis=0)

if ((slope\_mean\_r == 0) or (slope\_mean\_l == 0 )):

print('dividing by zero')

return 1

x1\_l = int((img.shape[0] - mean\_l[0][1] - (slope\_mean\_l \* mean\_l[0][0]))/slope\_mean\_l)

x2\_l = int((img.shape[0] - mean\_l[0][1] - (slope\_mean\_l \* mean\_l[0][0]))/slope\_mean\_l)

x1\_r = int((img.shape[0] - mean\_r[0][1] - (slope\_mean\_r \* mean\_r[0][0]))/slope\_mean\_r)

x2\_r = int((img.shape[0] - mean\_r[0][1] - (slope\_mean\_r \* mean\_r[0][0]))/slope\_mean\_r)

#6

if x1\_l > x1\_r:

x1\_l = int((x1\_l+x1\_r)/2)

x1\_r = x1\_l

y1\_l = int((slope\_mean\_l \* x1\_l ) + mean\_l[0][1] - (slope\_mean\_l \* mean\_l[0][0]))

y1\_r = int((slope\_mean\_r \* x1\_r ) + mean\_r[0][1] - (slope\_mean\_r \* mean\_r[0][0]))

y2\_l = int((slope\_mean\_l \* x2\_l ) + mean\_l[0][1] - (slope\_mean\_l \* mean\_l[0][0]))

y2\_r = int((slope\_mean\_r \* x2\_r ) + mean\_r[0][1] - (slope\_mean\_r \* mean\_r[0][0]))

else:

y1\_l = img.shape[0]

y2\_l = img.shape[0]

y1\_r = img.shape[0]

y2\_r = img.shape[0]

present\_frame = np.array([x1\_l,y1\_l,x2\_l,y2\_l,x1\_r,y1\_r,x2\_r,y2\_r],dtype ="float32")

if first\_frame == 1:

next\_frame = present\_frame

first\_frame = 0

else :

prev\_frame = cache

next\_frame = (1-α)\*prev\_frame+α\*present\_frame

cv2.line(img, (int(next\_frame[0]), int(next\_frame[1])), (int(next\_frame[2]),int(next\_frame[3])), color, thickness)

cv2.line(img, (int(next\_frame[4]), int(next\_frame[5])), (int(next\_frame[6]),int(next\_frame[7])), color, thickness)

cache = next\_frame

def hough\_lines(img, rho, theta, threshold, min\_line\_len, max\_line\_gap):

lines = cv2.HoughLinesP(img, rho, theta, threshold, np.array([]), minLineLength=min\_line\_len, maxLineGap=max\_line\_gap)

line\_img = np.zeros((img.shape[0], img.shape[1], 3), dtype=np.uint8)

lines\_drawn(line\_img,lines)

return line\_img

def weighted\_img(img, initial\_img, α=0.8, β=1., λ=0.):

return cv2.addWeighted(initial\_img, α, img, β, λ)

def process\_image(image):

global first\_frame

gray\_image = cv2.cvtColor(image, cv2.COLOmean\_r[0][1] - (slope\_mean\_r \* mean\_r[0][0])GR2GRAY)

img\_hsv = cv2.cvtColor(image, cv2.COLOR\_RGB2HSV)

lower\_yellow = np.array([20, 100, 100], dtype = "uint8")

upper\_yellow = np.array([30, 255, 255], dtype="uint8")

mask\_yellow = cv2.inRange(img\_hsv, lower\_yellow, upper\_yellow)

mask\_white = cv2.inRange(gray\_image, 200, 255)

mask\_yw = cv2.bitwise\_or(mask\_white, mask\_yellow)

mask\_yw\_image = cv2.bitwise\_and(gray\_image, mask\_yw)

gauss\_gray= cv2.GaussianBlur(mask\_yw\_image, (5, 5), 0)

canny\_edges=cv2.Canny(gauss\_gray, 50, 150)

imshape = image.shape

lower\_left = [imshape[1]/9,imshape[0]]

lower\_right = [imshape[1]-imshape[1]/9,imshape[0]]

top\_left = [imshape[1]/2-imshape[1]/8,imshape[0]/2+imshape[0]/10]

top\_right = [imshape[1]/2+imshape[1]/8,imshape[0]/2+imshape[0]/10]

vertices = [np.array([lower\_left,top\_left,top\_right,lower\_right],dtype=np.int32)]

roi\_image = interested\_region(canny\_edges, vertices)

theta = np.pi/180

line\_image = hough\_lines(roi\_image, 4, theta, 30, 100, 180)

result = weighted\_img(line\_image, image, α=0.8, β=1., λ=0.)

return result

Name -Swastik Gupta(2014111045)

Sankalp Kanikar(2014111056)

Vinay Gupta(2014111047)