#Importing Libraries

import cv2

import dlib

import time

import math

#Classifier File

carCascade = cv2.CascadeClassifier("vech.xml")

#Video file capture

video = cv2.VideoCapture("carsVideo.mp4")

# Constant Declaration

WIDTH =1280

HEIGHT = 720

#estimate speed function

def estimateSpeed(location1, location2):

d\_pixels = math.sqrt(math.pow(location2[0] - location1[0], 2) + math.pow(location2[1] - location1[1], 2))

ppm = 8.8

d\_meters = d\_pixels / ppm

fps = 18

speed = d\_meters \* fps \* 3.6

return speed

#tracking multiple objects

def trackMultipleObjects():

rectangleColor = (0, 255, 255)

frameCounter = 0

currentCarID = 0

fps = 0

carTracker = {}

carNumbers = {}

carLocation1 = {}

carLocation2 = {}

speed = [None] \* 1000

out = cv2.VideoWriter('outTraffic.avi', cv2.VideoWriter\_fourcc('M','J','P','G'), 10, (WIDTH, HEIGHT))

while True:

start\_time = time.time()

rc, image = video.read()

if type(image) == type(None):

break

image = cv2.resize(image, (WIDTH, HEIGHT))

resultImage = image.copy()

frameCounter = frameCounter + 1

carIDtoDelete = []

for carID in carTracker.keys():

trackingQuality = carTracker[carID].update(image)

if trackingQuality < 7:

carIDtoDelete.append(carID)

for carID in carIDtoDelete:

print("Removing carID " + str(carID) + ' from list of trackers. ')

print("Removing carID " + str(carID) + ' previous location. ')

print("Removing carID " + str(carID) + ' current location. ')

carTracker.pop(carID, None)

carLocation1.pop(carID, None)

carLocation2.pop(carID, None)

if not (frameCounter % 10):

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

cars = carCascade.detectMultiScale(gray, 1.1, 13, 18, (24, 24))

for (\_x, \_y, \_w, \_h) in cars:

x = int(\_x)

y = int(\_y)

w = int(\_w)

h = int(\_h)

x\_bar = x + 0.5 \* w

y\_bar = y + 0.5 \* h

matchCarID = None

for carID in carTracker.keys():

trackedPosition = carTracker[carID].get\_position()

t\_x = int(trackedPosition.left())

t\_y = int(trackedPosition.top())

t\_w = int(trackedPosition.width())

t\_h = int(trackedPosition.height())

t\_x\_bar = t\_x + 0.5 \* t\_w

t\_y\_bar = t\_y + 0.5 \* t\_h

if ((t\_x <= x\_bar <= (t\_x + t\_w)) and (t\_y <= y\_bar <= (t\_y + t\_h)) and (x <= t\_x\_bar <= (x + w)) and (y <= t\_y\_bar <= (y + h))):

matchCarID = carID

if matchCarID is None:

print(' Creating new tracker' + str(currentCarID))

tracker = dlib.correlation\_tracker()

tracker.start\_track(image, dlib.rectangle(x, y, x + w, y + h))

carTracker[currentCarID] = tracker

carLocation1[currentCarID] = [x, y, w, h]

currentCarID = currentCarID + 1

for carID in carTracker.keys():

trackedPosition = carTracker[carID].get\_position()

t\_x = int(trackedPosition.left())

t\_y = int(trackedPosition.top())

t\_w = int(trackedPosition.width())

t\_h = int(trackedPosition.height())

cv2.rectangle(resultImage, (t\_x, t\_y), (t\_x + t\_w, t\_y + t\_h), rectangleColor, 4)

carLocation2[carID] = [t\_x, t\_y, t\_w, t\_h]

end\_time = time.time()

if not (end\_time == start\_time):

fps = 1.0/(end\_time - start\_time)

for i in carLocation1.keys():

if frameCounter % 1 == 0:

[x1, y1, w1, h1] = carLocation1[i]

[x2, y2, w2, h2] = carLocation2[i]

carLocation1[i] = [x2, y2, w2, h2]

if [x1, y1, w1, h1] != [x2, y2, w2, h2]:

if (speed[i] == None or speed[i] == 0) and y1 >= 275 and y1 <= 285:

speed[i] = estimateSpeed([x1, y1, w1, h1], [x1, y2, w2, h2])

if speed[i] != None and y1 >= 180:

cv2.putText(resultImage, str(int(speed[i])) + "km/h", (int(x1 + w1/2), int(y1-5)), cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 0, 100) ,2)

cv2.imshow('result', resultImage)

out.write(resultImage)

if cv2.waitKey(1) == 27:

break

cv2.destroyAllWindows()

out.release()

if \_\_name\_\_ == '\_\_main\_\_':

trackMultipleObjects()