from scipy.spatial import distance as dist

from imutils.video import VideoStream

from imutils import face\_utils

from threading import Thread

import numpy as np

import argparse

import imutils

import time

import dlib

import cv2

import os

import cv2

import numpy as np

import dlib

from math import hypot

from imutils import face\_utils

from twilio.rest import Client

import serial

data = serial.Serial(

'COM3',

baudrate = 9600,

parity=serial.PARITY\_NONE,

stopbits=serial.STOPBITS\_ONE,

bytesize=serial.EIGHTBITS,

# )

timeout=1 # must use when using data.readline()

)

account\_sid = "ACf9e6b8bcae65d1eecff8b00e438f1e22"

auth\_token = "54f3136be063a85c1bbcd575a31ce8f9"

client = Client(account\_sid, auth\_token)

def eye\_aspect\_ratio(eye):

A = dist.euclidean(eye[1], eye[5])

B = dist.euclidean(eye[2], eye[4])

C = dist.euclidean(eye[0], eye[3])

ear = (A + B) / (2.0 \* C)

return ear

def final\_ear(shape):

(lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]

(rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]

leftEye = shape[lStart:lEnd]

rightEye = shape[rStart:rEnd]

leftEAR = eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye)

ear = (leftEAR + rightEAR) / 2.0

return (ear, leftEye, rightEye)

def lip\_distance(shape):

top\_lip = shape[50:53]

top\_lip = np.concatenate((top\_lip, shape[61:64]))

low\_lip = shape[56:59]

low\_lip = np.concatenate((low\_lip, shape[65:68]))

top\_mean = np.mean(top\_lip, axis=0)

low\_mean = np.mean(low\_lip, axis=0)

distance = abs(top\_mean[1] - low\_mean[1])

return distance

ap = argparse.ArgumentParser()

ap.add\_argument("-w", "--webcam", type=int, default=0,

help="index of webcam on system")

args = vars(ap.parse\_args())

EYE\_AR\_THRESH = 0.3

EYE\_AR\_CONSEC\_FRAMES = 10

YAWN\_THRESH = 20

alarm\_status = False

alarm\_status2 = False

saying = False

COUNTER = 0

print("-> Loading the predictor and detector...")

detector = dlib.get\_frontal\_face\_detector()

detector1 = cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml") #Faster but less accurate

predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')

print("-> Starting Video Stream")

vs = VideoStream(src=args["webcam"]).start()

#vs= VideoStream(usePiCamera=True).start() //For Raspberry Pi

time.sleep(1.0)

##global tempFlag

##global hrFlag

while True:

frame = vs.read()

frame = imutils.resize(frame, width=450)

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

#rects = detector(gray, 0)

rects = detector1.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30),flags=cv2.CASCADE\_SCALE\_IMAGE)

#for rect in rects:

for (x, y, w, h) in rects:

rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

shape = predictor(gray, rect)

shape = face\_utils.shape\_to\_np(shape)

eye = final\_ear(shape)

ear = eye[0]

leftEye = eye [1]

rightEye = eye[2]

distance = lip\_distance(shape)

leftEyeHull = cv2.convexHull(leftEye)

rightEyeHull = cv2.convexHull(rightEye)

cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)

cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

lip = shape[48:60]

cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

if ear < EYE\_AR\_THRESH:

COUNTER += 1

if COUNTER >= EYE\_AR\_CONSEC\_FRAMES:

print('DROWSINESS ALERT!')

data.write(str.encode('$'))

cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

client.api.account.messages.create(

to="+91-7019951952",

from\_="+18722013952" , #+1 210-762-4855"

body="Drowsiness Detected" )

else:

COUNTER = 0

if (distance > YAWN\_THRESH):

print('Yawn ALERT!')

cv2.putText(frame, "Yawn Alert", (10, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

else:

alarm\_status2 = False

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

elif key == ord("e"):

print('Emergency ')

client.api.account.messages.create(

to="+91-7019951952",

from\_="+18722013952" , #+1 210-762-4855"

body="Emergency Detected" )

cv2.destroyAllWindows()

vs.stop()