## SAMPLE CODE

### DROWSINESS DETECTION.PY

import numpy as np

import imutils

import time

import timeit

import dlib

import cv2

import matplotlib.pyplot as plt

from scipy.spatial import distance as dist

from imutils import face\_utils

from threading import Thread

from threading import Timer

from check\_cam\_fps import check\_fps

import make\_train\_data as mtd

import light\_remover as lr

import ringing\_alarm as alarm

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 init\_open\_ear() :

time.sleep(5)

print("open init time sleep")

ear\_list = []

th\_message1 = Thread(target = init\_message)

th\_message1.deamon = True

th\_message1.start()

for i in range(7) :

ear\_list.append(both\_ear)

time.sleep(1)

global OPEN\_EAR

OPEN\_EAR = sum(ear\_list) / len(ear\_list)

print("open list =", ear\_list, "\nOPEN\_EAR =", OPEN\_EAR, "\n")

def init\_close\_ear() :

time.sleep(2)

th\_open.join()

time.sleep(5)

print("close init time sleep")

ear\_list = []

th\_message2 = Thread(target = init\_message)

th\_message2.deamon = True

th\_message2.start()

time.sleep(1)

for i in range(7) :

ear\_list.append(both\_ear)

time.sleep(1)

CLOSE\_EAR = sum(ear\_list) / len(ear\_list)

global EAR\_THRESH

EAR\_THRESH = (((OPEN\_EAR - CLOSE\_EAR) / 2) + CLOSE\_EAR) #EAR\_THRESH means 50% of the being opened eyes state

print("close list =", ear\_list, "\nCLOSE\_EAR =", CLOSE\_EAR, "\n")

print("The last EAR\_THRESH's value :",EAR\_THRESH, "\n")

def init\_message() :

print("init\_message")

alarm.sound\_alarm("init\_sound.mp3")

#####################################################################################################################

#1. Variables for checking EAR.

#2. Variables for detecting if user is asleep.

#3. When the alarm rings, measure the time eyes are being closed.

#4. When the alarm is rang, count the number of times it is rang, and prevent the alarm from ringing continuously.

#5. We should count the time eyes are being opened for data labeling.

#6. Variables for trained data generation and calculation fps.

#7. Detect face & eyes.

#8. Run the cam.

#9. Threads to run the functions in which determine the EAR\_THRESH.

#1.

OPEN\_EAR = 0 #For init\_open\_ear()

EAR\_THRESH = 0 #Threashold value

#2.

#It doesn't matter what you use instead of a consecutive frame to check out drowsiness state. (ex. timer)

EAR\_CONSEC\_FRAMES = 20

COUNTER = 0 #Frames counter.

#3.

closed\_eyes\_time = [] #The time eyes were being offed.

TIMER\_FLAG = False #Flag to activate 'start\_closing' variable, which measures the eyes closing time.

ALARM\_FLAG = False #Flag to check if alarm has ever been triggered.

#4.

ALARM\_COUNT = 0 #Number of times the total alarm rang.

RUNNING\_TIME = 0 #Variable to prevent alarm going off continuously.

#5.

PREV\_TERM = 0 #Variable to measure the time eyes were being opened until the alarm rang.

#6. make trained data

np.random.seed(9)

power, nomal, short = mtd.start(25) #actually this three values aren't used now. (if you use this, you can do the plotting)

#The array the actual test data is placed.

test\_data = []

#The array the actual labeld data of test data is placed.

result\_data = []

#For calculate fps

prev\_time = 0

#7.

print("loading facial landmark predictor...")

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

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

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

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

#8.

print("starting video stream thread...")

vs = VideoStream(src=0).start()

time.sleep(1.0)

#9.

th\_open = Thread(target = init\_open\_ear)

th\_open.deamon = True

th\_open.start()

th\_close = Thread(target = init\_close\_ear)

th\_close.deamon = True

th\_close.start()

#####################################################################################################################

while True:

frame = vs.read()

frame = imutils.resize(frame, width = 400)

L, gray = lr.light\_removing(frame)

rects = detector(gray,0)

#checking fps. If you want to check fps, just uncomment below two lines.

#prev\_time, fps = check\_fps(prev\_time)

#cv2.putText(frame, "fps : {:.2f}".format(fps), (10,130), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (200,30,20), 2)

for rect in rects:

shape = predictor(gray, rect)

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

leftEye = shape[lStart:lEnd]

rightEye = shape[rStart:rEnd]

leftEAR = eye\_aspect\_ratio(leftEye)

rightEAR = eye\_aspect\_ratio(rightEye)

#(leftEAR + rightEAR) / 2 => both\_ear.

both\_ear = (leftEAR + rightEAR) \* 500 #I multiplied by 1000 to enlarge the scope.

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)

if both\_ear < EAR\_THRESH :

if not TIMER\_FLAG:

start\_closing = timeit.default\_timer()

TIMER\_FLAG = True

COUNTER += 1

if COUNTER >= EAR\_CONSEC\_FRAMES:

mid\_closing = timeit.default\_timer()

closing\_time = round((mid\_closing-start\_closing),3)

if closing\_time >= RUNNING\_TIME:

if RUNNING\_TIME == 0 :

CUR\_TERM = timeit.default\_timer()

OPENED\_EYES\_TIME = round((CUR\_TERM - PREV\_TERM),3)

PREV\_TERM = CUR\_TERM

RUNNING\_TIME = 1.75

RUNNING\_TIME += 2

ALARM\_FLAG = True

ALARM\_COUNT += 1

print("{0}st ALARM".format(ALARM\_COUNT))

print("The time eyes is being opened before the alarm went off :", OPENED\_EYES\_TIME)

print("closing time :", closing\_time)

test\_data.append([OPENED\_EYES\_TIME, round(closing\_time\*10,3)])

result = mtd.run([OPENED\_EYES\_TIME, closing\_time\*10], power, nomal, short)

result\_data.append(result)

t = Thread(target = alarm.select\_alarm, args = (result, ))

t.deamon = True

t.start()

else :

COUNTER = 0

TIMER\_FLAG = False

RUNNING\_TIME = 0

if ALARM\_FLAG :

end\_closing = timeit.default\_timer()

closed\_eyes\_time.append(round((end\_closing-start\_closing),3))

print("The time eyes were being offed :", closed\_eyes\_time)

ALARM\_FLAG = False

cv2.putText(frame, "EAR : {:.2f}".format(both\_ear), (300,130), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (200,30,20), 2)

cv2.imshow("Frame",frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

break

cv2.destroyAllWindows()

vs.stop()

#### RINGING ALARM.PY

import pygame

def select\_alarm(result) : if result == 0:

sound\_alarm("power\_alarm.wav")

elif result == 1 :

sound\_alarm("nomal\_alarm.wav")

else :

sound\_alarm("short\_alarm.mp3")

def sound\_alarm(path) : pygame.mixer.init() pygame.mixer.music.load(path) pygame.mixer.music.play()

#### LIGHT REMOVER.PY

import cv2 def light\_removing(frame) :

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) lab = cv2.cvtColor(frame, cv2.COLOR\_BGR2LAB)

L = lab[:,:,0]

med\_L = cv2.medianBlur(L,99) #median filter invert\_L = cv2.bitwise\_not(med\_L) #invert lightness composed = cv2.addWeighted(gray, 0.75, invert\_L, 0.25, 0) return L, compose