AI ALGORITHM FOR DRIVERS’ YAWNING AND DROWSINESS ALERT

Rajdeep Shaw

Raspberry Pi camera module has been used here for proper detection of drowsiness and yawning of the person sitting in the driver’s seat. This is a measure to check if the driver is feeling asleep. Alert is provided if any error occurs hence reducing the chances of accident. Drowsiness detection works based on the theory of **Eye aspect ratio**. Euclidean distances between perpendicular eye positions are calculated and beyond a certain threshold it creates an alert. Yawning also detects the distance between upper lip and lower lip and decides whether if the driver is feeling sleepy. (N.B- driver has to keep his head in a particular distance from the camera, alerting the distances may change the lip gap and hence may be differential results). In that case, an alert is generated.

Drowsiness detection

- During drowsiness, a person’s eye tends to get closed. So drowsiness can be detected by the degree to which the eye of a person is closed.
- The degree to which the eye of the person is closed is detected by the principle of **Eye Aspect Ratio (EAR)**. Here it works by taking perpendicular distances between various points of eye. Here we take 6 points for eg. P_1, P_2, P_3, P_4, P_5, P_6.

So $EAR = (|P_2 – P_6|+|P_3 – P_5|)/(2|P_1 – P_4|)$.

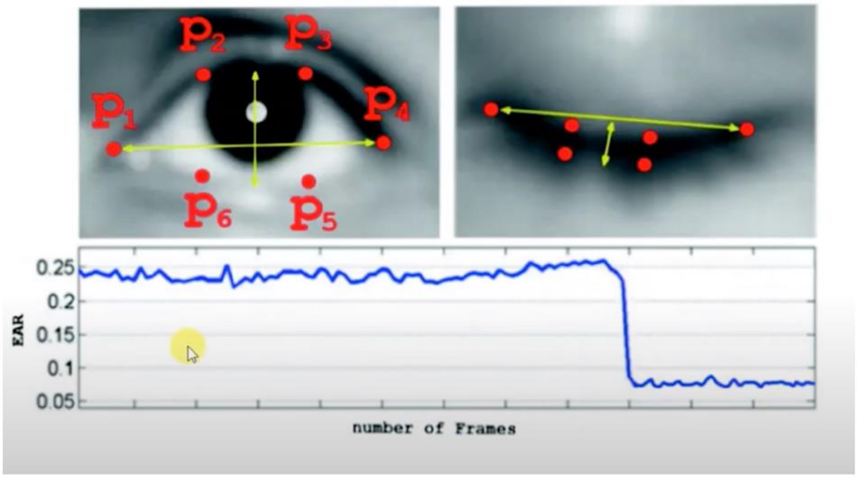

Fig 1. **EAR vs frame reception from Videostream**

- EAR value drops when eye is tending to get closed.
- Here we have set the threshold point of 0.15. The person is declared drowsy if EAR lies in the range [0.15, 1].

Code:-

```
#python drowniness_yawn.py --webcam webcam_index

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

def alarm(msg):
    global alarm_status
    global alarm_status2
    global saying

    while alarm_status:
        print('call')
        s = 'espeak "'+msg+'"'
        os.system(s)

    if alarm_status2:
        print('call')
        saying = True
        s = 'espeak "' + msg + '"'
        os.system(s)
        saying = False

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 = 30
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()
detector = 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)

while True:

    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #rects = detector(gray, 0)
    rects = detector.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:
                if alarm_status == False:
                    alarm_status = True
                    t = Thread(target=alarm, args=('wake up sir',))
                    t.deamon = True
                    t.start()

                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
2)

        else:
            COUNTER = 0
            alarm_status = False

        if (distance > YAWN_THRESH):
                cv2.putText(frame, "Yawn Alert", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
2)
                if alarm_status2 == False and saying == False:
                    alarm_status2 = True
                    t = Thread(target=alarm, args=('take some fresh air
sir',))
                    t.deamon = True
                    t.start()
        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

cv2.destroyAllWindows()
vs.stop()
```

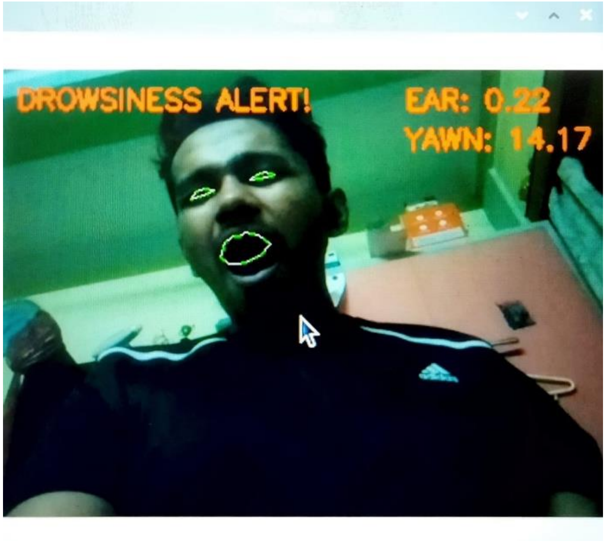

Fig: - Drowsiness alert using EAR and yawn detection

Yawn Detection

- **CONCEPT OF YAWNING:-**
    - Distance of upper lip and lower lip → Greater than threshold value →the yawning ALERT
    - Distance between user and camera should remain intact in the process otherwise there can be falsified results.
- **PSEUDO-CODE:-**

```
 //install all libraries
//Avgparse:- used to take arguments from command line

      Eye_threshold_value=0.3
     EYE_AR_THRES=0.3
      Frame_check=30 frames/sec
     Yawn_Threshold=2

//Detector:-It detects face from frames
                ➢ “CV2.Cascade_classifier” has been used for faster but less accurate
                  instead of dlib detectors  with the help of it ‘.xml’ file has been invoked.

    //Predictor:- (dlib shape predictor used to invoke “.dat” file
                ➢ If ( frames==found) then

                       Predictor=frames     //to find different land marks
```

- **Command for frame reception from camera:-**
    - ➢ **VS=Videostream(use Picamera=TRUE).start()**

- **Since CV2 classifier is used → we detect faces from grey-scale image**
    - ➢ **rects=detector.detect_Multiscale**

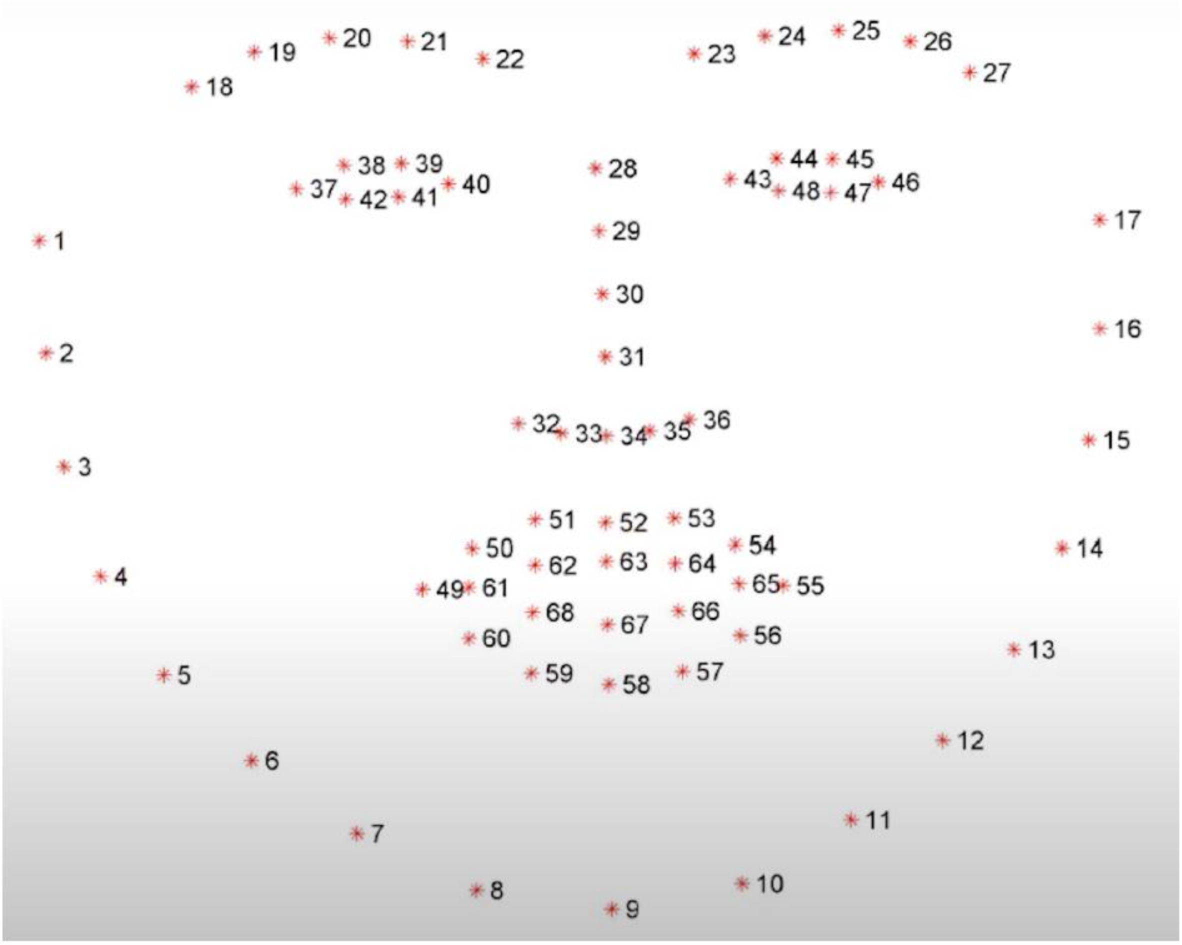

Fig.:- Facial landmark positions

- **<u>After locating shapes→it is converted into “numpy” for easy calculation</u>**
- **<u>Positions and configurations for each eye is set(left_eye,right_eye)-→the EAR_value calculated.</u>**

- **<u>FOR YAWNING :--</u>**

<u>Lip distance calculation:- </u> Lip distance is calculated by using lip distance function<u> .</u>

- ➢ Finding positions and configuration landmarks of the upper lip and lower lip
- ➢ <u>Mean </u>is taken of the all the six points of the lip region
- ➢ Then a distance is calculated hence about <u> threshold too.</u>

- <u>CVR convex HULL :-</u> It is used to plot the outline on our object of interest that is in this case the lip region

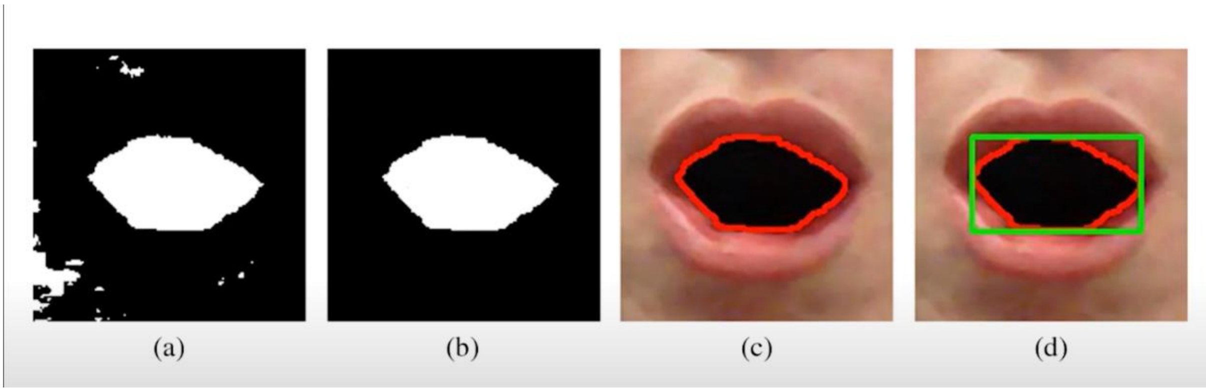

Fig:- Lip distance measurement for yawn detection

- **LOGIC FOR ALARM:-**

```
➢ If( 'ear_value' > 'threshold_value')  then,
            Counter_value++;
             If( 'Counter_value' > 'consequent_frames_value')  then,
                    Alarm=true(1)
           else
                  Alarm=false(0)
➢ If( 'lip_distance' > 'threshold_value')  then,
            Counter_value++;
             If( 'Counter_value' > 'consequent_frames_value')  then,
                    Alarm=true(1)
           else
                  Alarm=false(0)
```

#################END###