VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

Digital Assignment-IV, MAY 2021

B.Tech., Winter-2020-2021

COURSE CODE	ITE3999
COURSE NAME	Technical Answers to Real World Problems (TARP)
SLOT	TE1
FACULTY	Prof. VIJAYAN E.

TEAM 7 MEMBERS:

REG. NO.	NAME
18BIT0027	SAKINA HUSSAIN BANDOOKWALA
18BIT0231	KUSHAGRA AGARWAL
18BIT0272	PRIYAL BHARDWAJ

Problem Statement

Nowadays the driver safety in vehicles is one of the most wanted systems to avoid accidents. Many researches and surveys have estimated that about 1.35 million drivers have been involved in drowsiness related accidents in the past 5 years. Speaking of our country, up to 25% of road accidents are fatigue related. Drivers must keep a close eye on the road and be alert at all times so that they can react appropriately to sudden events.

Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving. The objective of this project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds as well as the number of blinks. This system will alert the driver when drowsiness is detected. At the same time, we will also be designing a mouth detection system to detect yawning.

Testing of the proposed approach

Proposed Approach:

- First of all, we have captured video from the camera and detected face from the video frames.
- From the extracted frames, we will derive the eyes from the face. This will be used in calculating the time for which eyes are closed.
- After that we have calculated Eyes Aspect Ratio (EAR) and compared that to threshold ratio. If the EAR is less than the threshold then an alert message will be displayed upon the screen.
- Along with Eye detection, the program will also detect mouth and keep a counter for number of yawns in a fixed interval of time. Yawn will be detected in a similar manner to blink detection.
- We have calculated Mouth Aspect Ratio (MAR) and compared that to threshold ratio. If the MAR is greater than the threshold, then it will be counted as a yawn and the yawn counter will increase. If the yawn counter crosses the threshold value, then an alert message will be displayed upon the screen.
- Also, apart from the message upon the screen, we will be sending a message on the phone of the emergency contact of driver along with the location of the driver.

Eye Detection

- In the system we have used facial landmark prediction for eye detection.
- The facial landmark detector included in the dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).
- This method starts by using:
    - A training set of labeled facial landmarks on an image. These images are manually labeled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
    - Priors, of more specifically, the probability on distance between pairs of input pixels.
- The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.
- We can detect and access both the eye region by the following facial landmark index show below
    - The right eye using [36, 42].
    - The left eye with [42, 48].
- These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on.
- Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data.

Eye Aspect Ratio (EAR)

- For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

$$EAR = \frac{||p2 - p6|| + ||p3 - p5||}{2||p1 - p4||}$$

  where p1, . . ., p6 are the 2D landmark locations.
- The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye.
- Aspect ratio of the open eye has a small variance among individuals, and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face.
- Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

Eye State Detection

- Finally, the decision for the eye state is made based on EAR calculated in the previous step. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”.
- The last step of the algorithm is to determine the person’s condition based on a pre-set condition for drowsiness. The average blink duration of a person is 100-400 milliseconds (i.e., 0.1-0.4 of a second).
- Hence if a person is drowsy his eye closure must be beyond this interval. We have set an interval of 48 frames which is nearly 3 seconds. If the eyes remain closed for three or more seconds, drowsiness is detected and alert regarding this is triggered.

Yawn Detection

- Yawning is characterized by a widely opened mouth.
- Like the eye closure detection, the facial landmarks are used to detect an open mouth.
- Mouth Aspect Ratio (MAR) is the parameter used to determine if the subject’s mouth is open.
- If MAR calculated from the frame is greater than the threshold, the subject is determined to be yawning.
- An alarm is raised if the subject has yawned more than the set boundary value consecutively.
- Small openings that in reality are taken as a result of talking, eating is ignored.

Drowsiness Detection

- The last step of the algorithm is to finally determine if the driver is drowsy or not.
- We will add up results of both the parameters, i.e., result from Eye state detection and Yawn Detection.
- If both the parameters cross the threshold value, we will sound an alarm to alert the driver.
- Along with alarm, we will also send a message to the emergency contact given by the driver which will contain the location of driver.
- If none of the parameter crosses the threshold value, then the system will keep repeating the process till the end of drive.

Following is our python code that can be run on any system such as laptop, mobile, etc. to start the driver drowsiness application.

Code

```
#python drowsiness_yawn.py --webcam 0 --alarm alarm.wav

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 serial
import playsound
from twilio.rest import Client
import geocoder
g = geocoder.ip('me')
print(str(g.latlng))

account_sid = 'ACc5166e7f1310f5d7fba4d6a96bfdb9a2'
auth_token = 'd02288d9d634561e51b2de0086b74a95'
client = Client(account_sid, auth_token)


###ser = serial.Serial('COM6', 9600, timeout=1)
#ser.open()
###ser.setDTR(False)
###ser.close()

def sound_alarm(path):
      # play an alarm sound
      playsound.playsound('alarm.wav', 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("-a", "--alarm", type=str, default="", help="path alarm .WAV
file")
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
sms_eye = 1
sms_yawn = 1

# initialize the frame counter as well as a boolean used to
# indicate if the alarm is going off
COUNTER = 0
ALARM_ON = False

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)
```

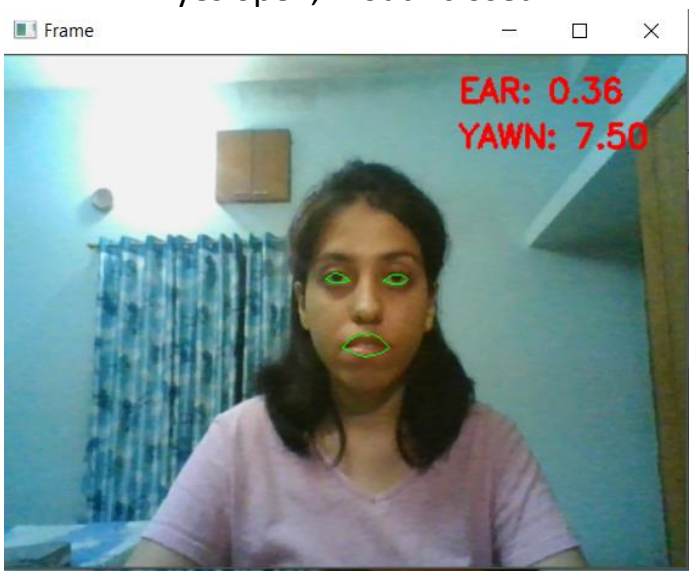
```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    #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 not ALARM_ON:
                    ALARM_ON = True
                    if(sms_eye==1):
                        message = client.messages \
                                    .create(
                                        body = "Drowsy driver detected at
location " + str(g.latlng),
                                from_='+18306269708',
                                        to='+919424157184'
                                        )
                        print(message.sid)
```

```
                        sms_eye = 0
                        print('SMS Sent')

                    if args["alarm"] != "":
                        t = Thread(target=sound_alarm, args=(args["alarm"],))
                        t.deamon = True
                        t.start()



##                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
            sms_eye = 1
            ALARM_ON = False

        if (distance > YAWN_THRESH):
                cv2.putText(frame, "Yawn Alert", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                if not ALARM_ON:
                    ALARM_ON = True
                    if(sms_yawn==1):
                        message = client.messages \
                                    .create(
                                        body = "Drowsy driver detected at
location " + str(g.latlng),
                                from_='+18306269708',
                                        to='+919424157184'
                                        )
                        print(message.sid)
                        sms_yawn = 0
                        print('SMS Sent')
                    if args["alarm"] != "":
                        t = Thread(target=sound_alarm, args=(args["alarm"],))
                        t.deamon = True
                        t.start()
        else:
            sms_yawn = 1
            ALARM_ON = False
```

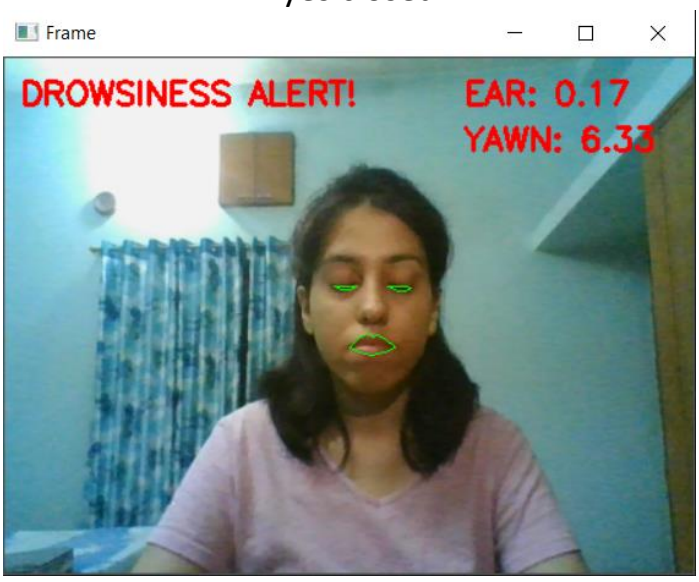
```
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    if key == ord("q"):
        break

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

Results

Eyes open, Mouth closed

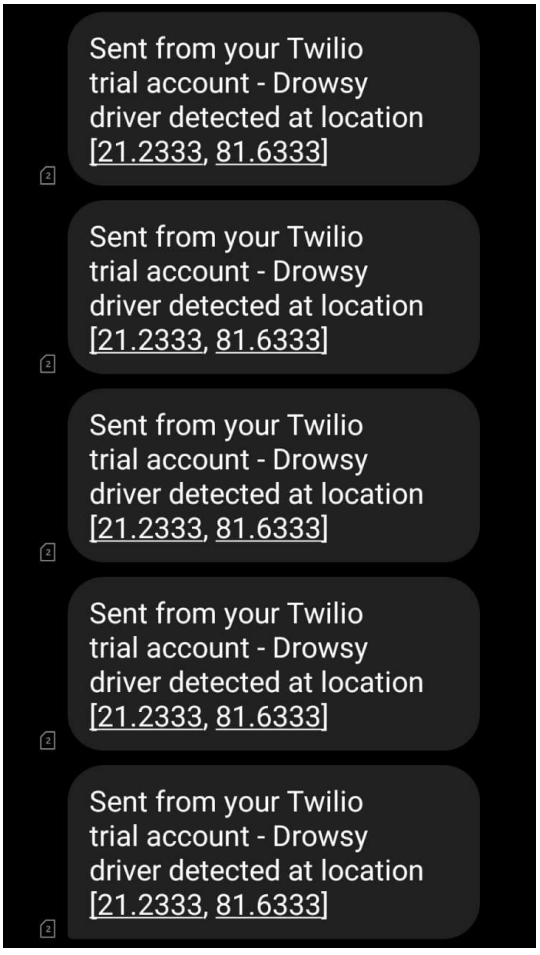


Eyes closed



Mouth open

Below is a sample of the message received on mobile of emergency contact given by the driver. Message comprises of the location of the drowsy driver and can be further customised as per the driver choice.



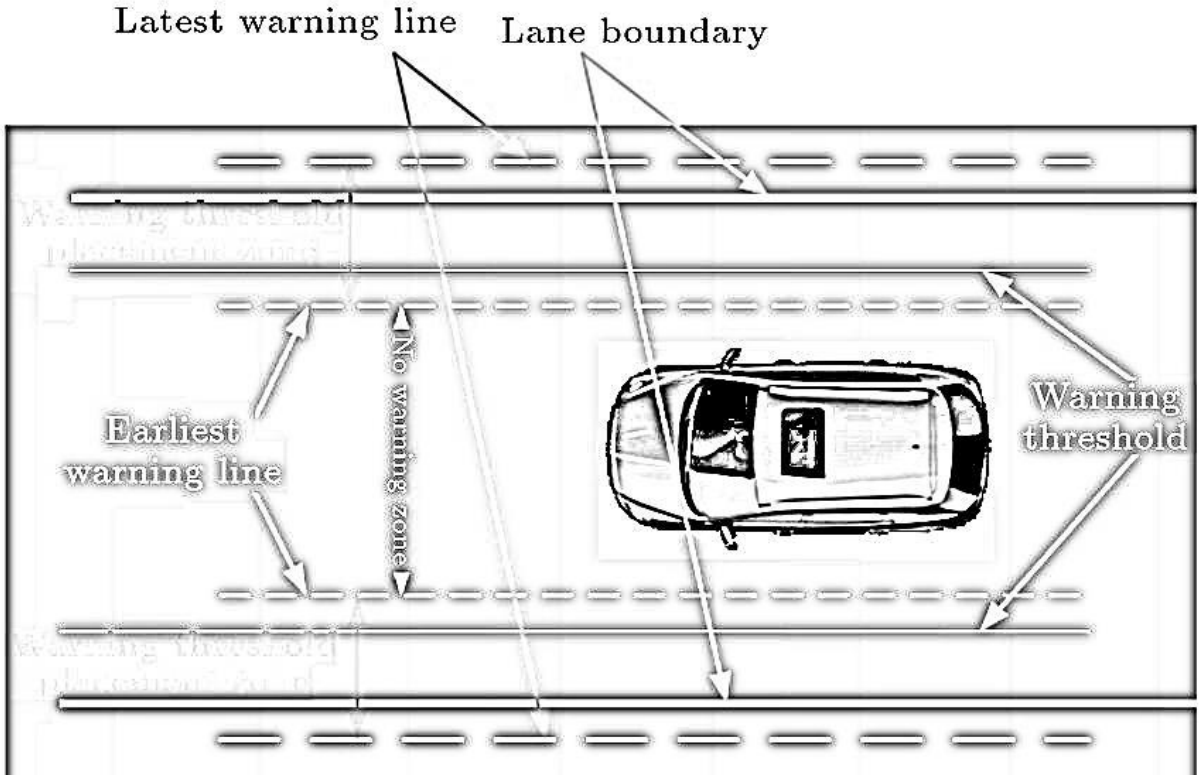
A sound alarm is also integrated in this hybrid approach to instantly alert the driver if they are detected to be drowsy.

Comparative analysis with others approach

From the literature survey which we had done in the beginning of our project, we got to know various approaches to detect driver drowsiness. Some of those are:

1) **Lane Detection system**

In this approach, the system monitors the car's position. If the car abruptly changes the lane or it changes lane way too fast, there may be possibility that the driver is drowsy and is not able to steer the car well thus, causing abrupt change in lane. If it detects such change, it sounds an alarm to wake up the driver and in some advanced systems, an auto correction system is also applied so that it will steer the car back into the right lane and direction.



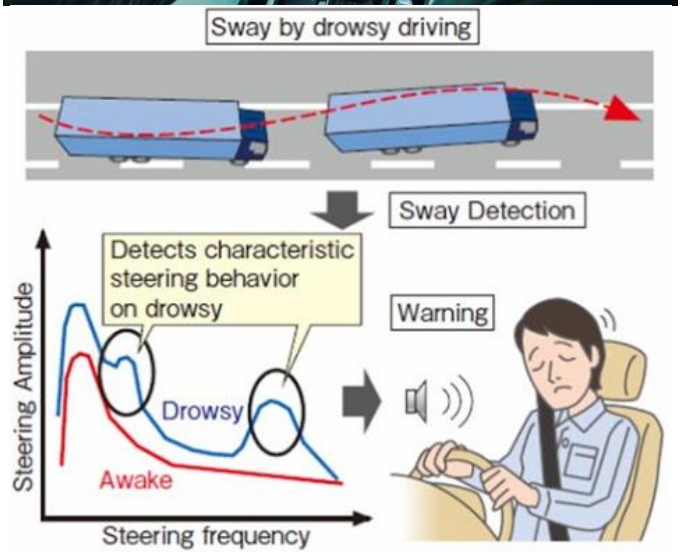
Problems:

- There may be case in which driver is awake and has to change to lanes abruptly to maybe prevent some accident. In that case, this system will falsely sound the alarm.

- There may be roads which have lot of ditches in them and to prevent any damage to car, the driver has to constantly change the lane to avoid any damage to the car.

2) Sensitive Steering Wheel

In one of the papers, we got to know about another system that works on the grip pressure on the steering wheel. This system works by placing a sensor on the steering wheel to detect how much pressure is being applied on it. If it is above optimum level then the driver is assumed to be not drowsy and if the pressure is less than the optimum level, then driver is assumed to be drowsy. On detection of drowsiness, the system will either sound the alarm or flash it on some display to wake up the driver.



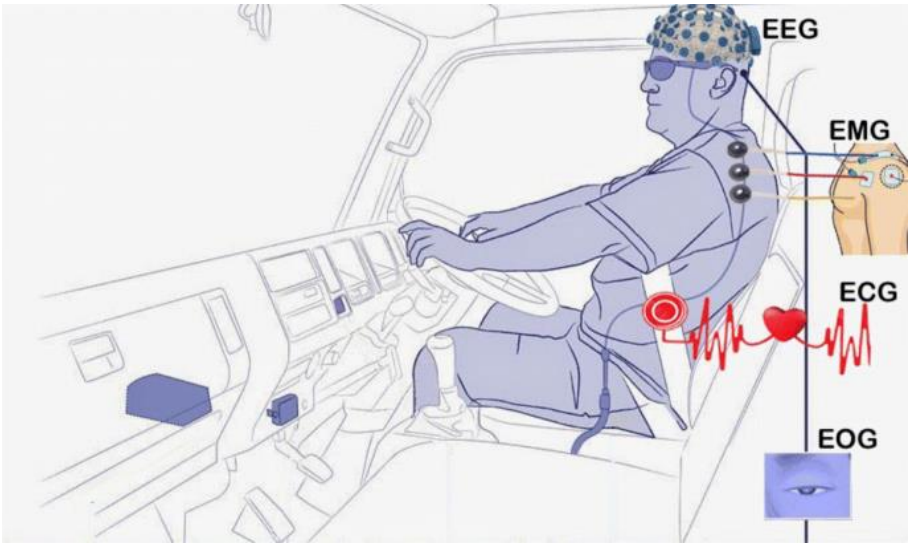
Problems:

- Experienced drivers tend to hold the steering lightly whereas the beginners or the new leaners tend to hold the steering more tightly. In such cases, this system may give some false results.

- Some drivers prefer to wear gloves while driving, this may also lead to varied pressure on the steering wheel.
- Other conditions such as sensitivity to factors like dirt, sweat and temperature may also change the pressure on the steering wheel.

3) Physiological sensors

Physiological signals (i.e., ECG, EEG, etc) have been commonly used to study drowsiness and sleep disorders. A conventional clinic measurement system requires the electrodes to be in contact with the human body by use of coupling gel. The system can be deployed in a vehicular environment to provide driver assistance. While drowsiness detection was the primary goal of this project, such a system can also be utilized for other beneficial purpose, e.g., health monitoring of drivers.



Problems:

- This interferes with the normal driver operation. All the wires from the sensors hinders the normal movement of a driver.
- It is not feasible for long term monitoring purposes. Time to time replacement of batteries or replacement of sensors due to various conditions does not help in an economical and long-term process.

4) EOG equipment for eye detection

EOG and video recordings are the main techniques used in drowsy driver detection studies to record the blinking behaviour of a driver.

To obtain a signal for eye blink detection with EOG, several surface electrodes are positioned around the eyes. Since the cornea has a positive electric potential in reference to the fundus of the eye, a natural occurring eyelid movement during a blink affects the electric potential between the two electrodes positioned above and below the eye. So, a blink can be measured as change in the potential distribution of the eye.

Problems:

- Although EOG is considered to be a quite reliable method because of its high frame rate, video-based assessment has become more popular for its practicability in the industry. This is due to its ability to measure contact free.

- The driver might feel embarrassed to wear EOG equipment in real traffic.
- EOG method is not very straightforward and requires measuring electric potential around the eyes for blinks whereas using video recordings, eyelid movement is visible in the images and can be assessed using image processing methods.
- Video-based assessments have more advantages like unobtrusive sensor, installation in instrumented car relatively unproblematic, possible to use over longer time periods if no recalibration necessary, tracking quality can be influenced by temperature shifts, movement of cameras, etc

References:

Prakash Choudhary, Rahul Sharma, Gautam Singh, Smarjeet Das “A Survey Paper on Drowsiness Detection & Alarm System for Drivers”, 2016 International Research Journal of Engineering and Technology (IRJET)

Y. Katyal, S. Alur and S. Dwivedi, "Safe driving by detecting lane discipline and driver drowsiness," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, 2014, pp. 1008-1012, doi: 10.1109/ICACCCT.2014.7019248.

Li, Zuojin et al. “Online Detection of Driver Fatigue Using Steering Wheel Angles for Real Driving Conditions.” Sensors (Basel, Switzerland) vol. 17,3 495. 2 Mar. 2017, doi:10.3390/s17030495

Anna Anund, Katja Kircher “Advantages and disadvantages of different methods to evaluate sleepiness warning systems” 2009 VTI Rapport 664A

****