Vehicle’s Driver Drowsiness Detection

A Project Report

in Partial Fulfillment of the Requirements

for the award of

Bachelor of Engineering

(Computer Science and Engineering)

By

Manshi (1930034)

Raushan Raj (1930035)

Under the guidance of

Prof. Damanpreet Singh

(Department of Computer Science and Engineering)

SANT LONGOWAL INSTITUTE OF ENGINEERING AND TECHNOLOGY,

LONGOWAL, SANGRUR – 148106, PUNJAB, INDIA

May, 2022

CERTIFICATE

This is certified that the project report on **"Vehicle's Driver Drowsiness Detection"** submitted by **"MANSHI, RAUSHAN RAJ"** in the partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science and Engineering, of SLIET, Longowal is an authentic work carried out by them under our supervision and guidance.

This project report has been approved as it satisfies the academic requirements and not been submitted to any other university or institution for the award of any degree.

…………………………………… ………………………………

(Sign. Of Project In Charge) **(Sign. Of Project Guide)**

………………………………

(Sign. Of H.O.D.)

ACKNOWLEDGEMENT

If words are considered as a symbol of approval and token of appreciation, then let the words play the heralding role expressing my gratitude. The satisfaction that accompanies the successful completion of any task would be incomplete without the mentioning the people whose ceaseless cooperation, constant guidance and encouragement made it possible. We are grateful to our project guide **Prof. Damanpreet Singh (CSE, SLIET)** for the guidance, inspiration and constructive suggestion that helped us in the preparation of this project. We would like to express our special gratitude and thanks to **Dr. Manoj Sachan (Head of Department).**

We also thank our project in charge **Jatinderpal Singh (AP, CSE)** and **Jagdeep Singh (AP, CSE)** for coordinating us in the successful completion of the project.

DECLARATION

We, "**MANSHI**" and "**RAUSHAN RAJ**" hereby declare that the project work entitled "**VEHICLE'S DRIVER DROWSINESS DETECTION**" is carried out by us and submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering in Computer Science Engineering, under Sant Longowal Institute of Engineering and Technology, Longowal during the academic year 2019-2022 and has not been submitted to any other university for the award of any kind of degree.

Manshi

Signature of the Student

Raushan Raj

Signature of the Student

The project Viva–Voce Examination has been held on __________and accepted.

....

Signature of Examiner

ABSTRACT

Driver fatigue is one of the major causes of accidents in the world. Detecting the drowsiness of the driver is one of the surest ways of measuring driver fatigue. In this project we aim to develop a prototype drowsiness detection system. This system works by monitoring the eyes and mouth of the driver and sounding an alarm when he/she is drowsy. The system so designed is a non-intrusive real-time monitoring system. The priority is on improving the safety of the driver without being obtrusive. In this project the eye blink, mouth movement of the driver is analyzed. If the driver's eyes remain closed for more than a certain period of time or found yawning repeatedly, the driver is said to be drowsy and an alarm is sounded.

Table of Contents

List of Figures

1. INTRODUCTION

Drowsiness of the drivers is one of the key issues for majority of road accidents. It threatens the road safety and causes severe injuries sometimes, resulting in fatality of the victim and economical losses. Drowsiness implies feeling lethargic, lack of concentration, tired eyes of the drivers while driving vehicles. Most of the accidents happen in India due to the lack of concentration of the driver. Performance of the driver gradually deteriorates owing to drowsiness. To avoid this anomaly, we developed a system that is able to detect the drowsiness nature of the driver and alert him immediately. This system captures images as a video stream through a camera, detects the face and localizes the eyes. Based on the result, the driver is alerted for drowsiness through an alarm system.

1.1 DIFFERENT APPROACHES TO DETECTING DROWSINESS:

There are different approaches to identify drowsiness state of the driver. They can be categorized into the following three main categories:

1.1.1 Behavioral Parameters-based Techniques:

Measuring the driver's fatigue without using non-invasive instruments comes under this category. Analyzing the behavior of the driver based on his/her eye closure ratio, blink frequency, yawning, position of the head and facial expressions. The current parameter used in this system is the eye-closure ratio of the driver.

1.1.2 Vehicular Parameters-based Techniques:

Measuring the fatigue nature of the driver through vehicle driving patterns comes under this category. These parameters include lane changing patterns, steering wheel angle, steering wheel grip force, vehicle speed variability and many more.

1.1.3 Physiological Parameters-based Techniques:

Measuring the drowsiness of the driver based on the physical conditions of the driver fall under this category. Such parameters may be respiration rate, heart-beat rate, body temperature and many more.

Among other various approaches, these physiological parameters provide the most accurate results since they are based on the biological nature of the driver.

All the above approaches have their own advantages and disadvantages. Based on the desired result accuracy, any approach can be used. Physiological approach includes wearing of the equipment on the driver's body. This equipment includes electrodes to detect the pulse rate of the driver which might make the driver uncomfortable while driving. This also can't be assured that the driver always wears such equipment while driving which may result in inefficient results. Hence there is a hindrance using the physiological approach. Vehicular-based approach is always based on the efficiency of the driver and his condition. There are also constraints like the road condition and the type of vehicle which may change regularly. Hence it is best to follow the behavioral based approach through visual assessment of the driver from a camera. There shall be no equipment attached to the driver. Hence this technique is always the best approach and can be implemented in any vehicle without any modifications.

1.2 DIGITAL IMAGE PROCESSING

The term digital image processing generally refers to processing of a two dimensional picture by a digital computer. In a broader context, it implies digital processing of any two- dimensional data. A digital image is an array of real numbers represented by a finite number of bits. The principle advantage of Digital Image Processing methods is its versatility, repeatability and the preservation of original data precision.

1.2.1 Pixel:

Pixel is the smallest element of an image. Each pixel corresponds to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255. The values of a pixel at any point correspond to the intensity of the light photons striking at that point. Each pixel stores a value proportional to the light intensity at that particular location.

1.2.2 Digital image:

A digital image is nothing more than data numbers indicating variations of red, green, and blue at a particular location on a grid of pixels.

1.2.3 Gray level:

The value of the pixel at any point denotes the intensity of image at that location, and that is also known as gray level.

1.3 MOTIVATION

Now-a-days, there is huge increase in private transportation day by day in this modernize world. It is tedious and boring driving for a long distance. One of the main causes behind the driver's lack of alertness is due to long time travelling without sleep and rest. Tired driver can get drowsy while driving. Every fraction of seconds drowsiness can turn into dangerous and life-threatening accidents may lead to death also. To prevent this type of incidents, it is required to monitor driver's alertness continuously and when it detects drowsiness, the driver should be alerted. Through this we can reduce significant number of accidents and can save lives of people.

1.4 PROBLEM STATEMENT

Many of the road accidents occurs due to drowsiness of the driver. Drowsiness can be detected by monitoring the driver through continuous video stream with a mobile or camera. The general objective is to create a model that will indicate whether a person is feeling drowsy or not. The model takes image for every frame and check for eyes and mouth movements. If the eye is closed for certain amount of time or mouth is wide open indicating a yawn, then it will alert driver through a sound.

2. LITERATURE SURVEY

In computer science, image processing is the use of computer algorithms to perform image processing on images. As a subcategory or field of digital signal processing, image processing has many advantages over analog image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the buildup of noise and signal distortion during processing. Since images are defined over two dimensions digital image processing may be modelled in the form of multidimensional system.

2.1 DROWSINESS DETECTION THROUGH REGION OF INTEREST:

Region of interest (ROI) can detect a driver's face. As can be seen in the blue rectangle is the region of interest. The way to create an ROI area is to first obtain the green rectangle area from the Haar Cascade Classifier in the first frame, which includes height, width. Then, the rectangle is scaled up to create region of interest. There are several steps to calculate the ROI area and we have to calculate ROI for each and every region of interest

Disadvantages of ROI:

1. It is uses extra frames or squares to detect face detection.
2. It can't find in low light.
3. Why to use again region of interest while Haar cascade classifier can do the same process?
4. It can't detect while using glasses in driving.

2.2 DETECTION OF DROWSINESS THROUGH LBPH:

In this algorithm the faces are detected by using local binary patterns histograms (LBPH). The first computational step in lbph is to create an intermediate image that describes the original image in a binary format. The image is converted into matrix form and we need to take a central value of the matrix to be used as and threshold value. This value is used to define neighboring values which can be set to either 0 or 1. The values which are 1 in the matrix form are to be considered and the remaining values are discarded. The values represent each pixel. Through this the region of face can be detected.

Disadvantages of LBPH:

1. It produces less urate results
2. The computational time is high.
3. This will work only if the data samples are less.

2.3 REAL TIME ANALYSIS USING EYE AND YAWNING

The basic purpose of the proposed method is to detect the close eye and open mouth simultaneously and generates an alarm on positive detection. The system firstly captures the real time video using the camera mounted in front of the driver. Then the frames of captured video are used to detect the face and eyes by applying the viola-jones method, with the training set of face and eyes provided in OpenCV. Small rectangle is drawn around the center of eye and matrix is created that shows that the Region of Interest (ROI) that is eyes used in the next step. Since the both eyes blink at the same time that's why only the right eye is examined to detect the close eye state. If the eye is closed for certain amount of time it will be considered as closed eye. To determine the eye state, firstly the eye ball color is acquired by sampling the RGB components on the center of eye pixel. Then the absolute thresholding is done on the eye ROI based on eye ball color and intensity map is obtained on Y-axis that show the distribution of pixels on y-axis which gives the height of eye ball and compared that value with threshold value which is 4 to distinguish the open and close eye. After that, if the eye blink is detected in each frame it will be considered as 1 and stored in the buffer and after the 100 frames, eye blinking rate is calculated. Then to detect the yawning motion of the mouth, contour finding algorithm is used to measure the size of mouth. If the height is greater than the certain threshold. It means person is taking yawning. To evaluate the performance of the proposed system, system has been measured under different conditions like persons with glasses, without glasses, with moustache and without moustache for 20 days in different timings. The system performs best when the drivers are without glasses.

2.4 EXISTING SYSTEM

The current drowsiness detection systems include the usage of the devices that detect the respiration rate, heart rate, blood pressure, etc. These devices can cause the driver to be uncomfortable for driving. Cannot be assured that the drivers wear these devices all the time while driving. May get lost or improper functioning which may lead to low accuracy in the result.

3. PROPOSED METHODOLOGY

3.1 FLOW CHART

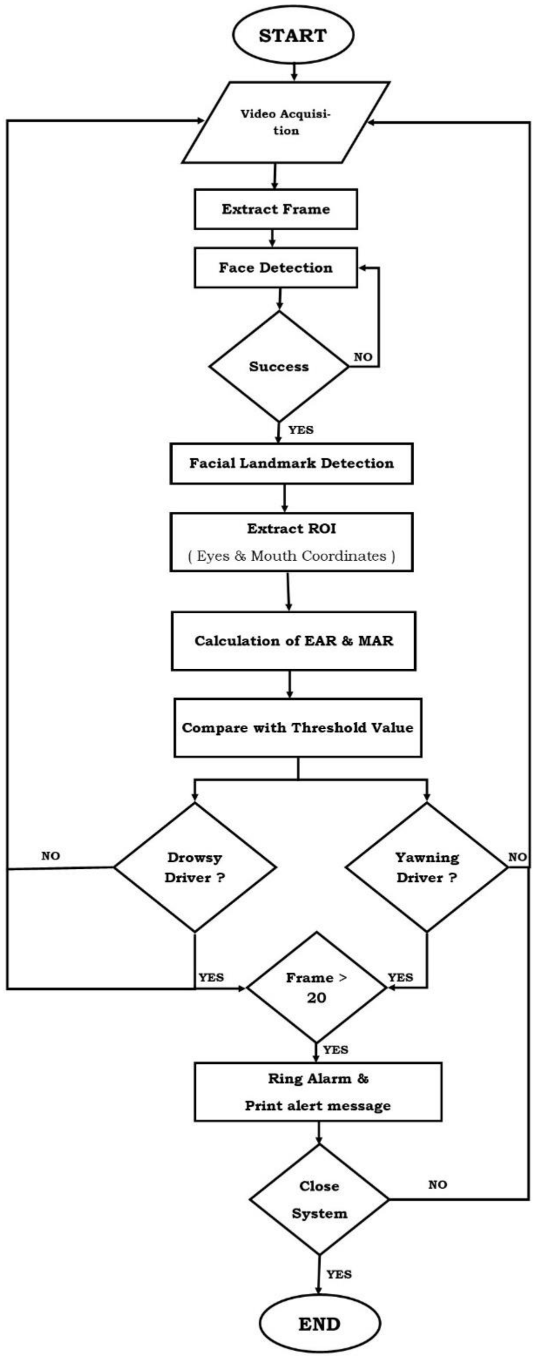


Fig 3.1: Flow Chart of the driver drowsiness detection system

3.2 MODULAR DIVISION

The entire architecture is divided into following modules.

- Face Detection
- Face Landmark Detection
- EAR Calculation
- MAR Calculation

3.2.1 Face Detection:

This module takes input from the camera and tries to detect a face in the video input. The detection of the face is achieved through the Haar classifiers mainly, the **Haar Cascade Classifier**. The face is detected in a rectangle format and converted to grayscale image and stored in the memory which can be used for further manipulation.

3.2.2 Face Landmark Detection:

The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.

Below we can visualize what each of these 68 coordinates map to:

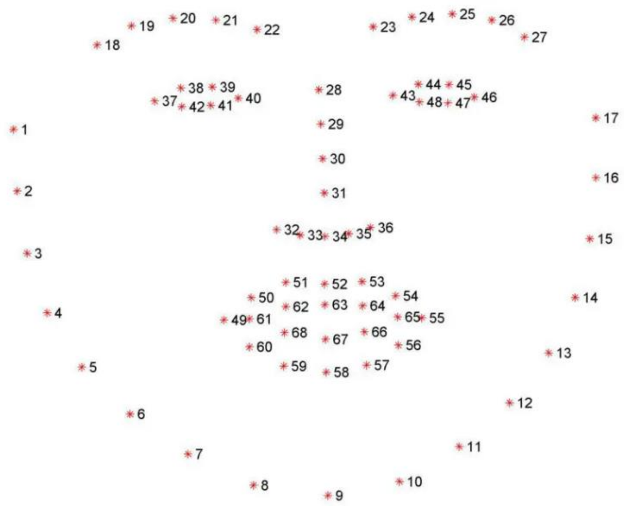


Fig 3.2: The full set of facial landmarks

3.2.3 EAR Calculation:

Eye Aspect Ratio function is defined to calculate the distance between the eye landmarks taken vertically and distances between the eye landmarks taken horizontally. So, when the eye is open, the value returned for the eye aspect ratio will be a constant approximately. The value will rapidly decrease approaching zero in case of an eye closure.

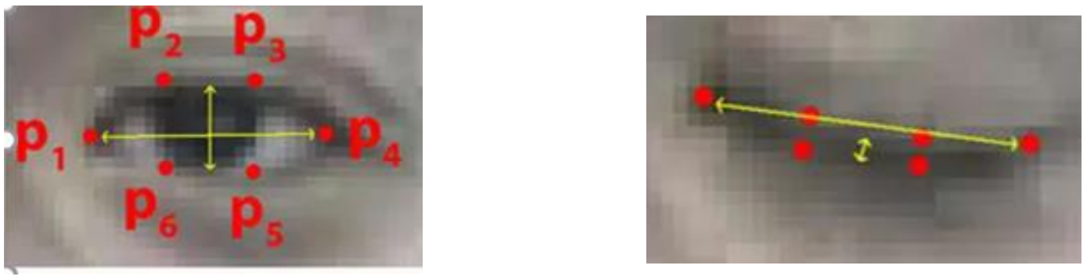


Fig 3.3: A visualization of eye landmarks when then the eye is open and closed

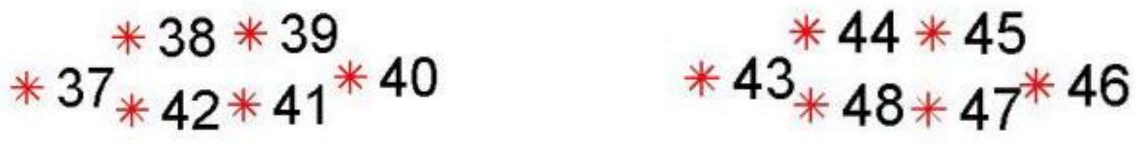


Fig 3.4: Eye Coordinates

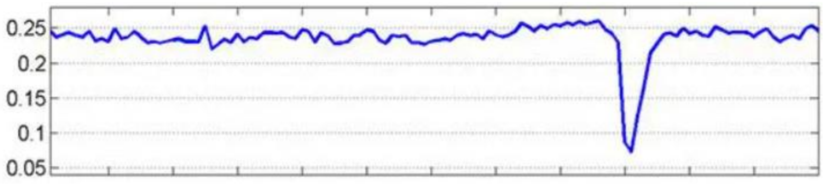


Fig 3.5: Graph showing blink

$$\text{EAR} = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$$

Fig 3.6: Formula for calculating EAR

3.2.4 MAR Calculation:

Mouth Aspect Ratio function is defined to calculate the distance between the mouth landmarks taken vertically and distances between the mouth landmarks taken horizontally. So, when the mouth is closed, the value returned for the mouth aspect ratio will be a constant approximately. The value will rapidly increase in case of mouth being wide open indicating yawning.

Fig 3.7: A visualization of mouth landmarks when then the mouth is open and closed

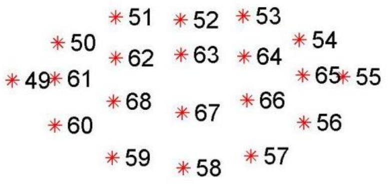


Fig 3.8: Mouth Coordinates

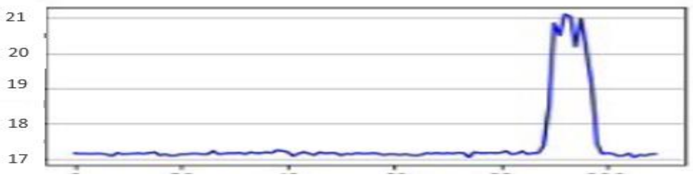


Fig 3.9: Graph showing yawn

$$\mathrm{MAR} = \frac{\|p_2 - p_8\| + \|p_3 - p_7\| + \|p_4 - p_6\|}{2\,\|p_1 - p_5\|}$$

Fig 3.10: Formula for calculating MAR

3.3 TECHNOLOGY USED.

3.3.1 Python

Python is an interpreted, high-level, general purpose programming language created by Guido Van Rossum and first released in 1991, Python's design philosophy emphasizes code Readability withits notable use of significant Whitespace. Its language constructs and object oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed, and garbage collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

3.3.2 OpenCV

OpenCV (Open-source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by willow garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open source BSD license. OpenCV supports some models from deep learning frameworks like TensorFlow, Torch, PyTorch (after converting to an ONNX model) and Caffe according to a defined list of supported layers. It promotes Open Vision Capsules which is a portable format, compatible with all other formats.

3.3.3 PyGame

PyGame is a Python wrapper module for the SDL multimedia library. It contains python functions and classes that will allow you to use SDL's support for playing cdroms, audio and video output, and keyboard, mouse and joystick input.

3.3.4 Numpy

NumPy is a library for the python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim with contributions from several other developers. In 2005, Travis created NumPy by incorporating features of the competing Num array into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

3.3.5 Dlib

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open-source licensing allows you to use it in any application, free of charge.

3.3.6 imutils

It is a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV.

3.3.7 Thonny

Thonny is an integrated development environment for Python that is designed for beginners. It supports different ways of stepping through the code, step-by-step expression evaluation, detailed visualization of the call stack and a mode for explaining the concepts of references and heap.

3.3.8 PyCharm

PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the python language. It is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data science with Anaconda. PyCharm is cross platform with windows, macOS and Linux.

4. EXPERIMENTAL ANALYSIS AND RESULTS

4.1 SYSTEM CONFIGURATION

4.1.1 Software requirements:

These are the software requirements for running this project.

- Operating System: Raspberry Pi OS, Windows 10/8/7 (incl. 64-bit), Mac OS, Linux
- Language: Python 3
- IDE: Thonny, JetBrains PyCharm Community Edition 2019.1.3 x64 or above.

4.1.2 Hardware requirements

- Processor: 64 bit, quad-core, 2.5 GHz minimum per core
- RAM: 4 GB or more.
- HDD: 20 GB of available space or more.
- Display: Dual XGA (1024 x 768) or higher resolution monitors.
- Camera: A webcam.
- Keyboard: A standard keyboard.
- Mouse: A standard mouse.

4.1.3 Raspberry Pi

The Raspberry Pi is a small single-board computer develops by the Raspberry Pi Foundation. It features a Broadcom System on Chip (SoC) with a 700MHz, ARM1176JZF-S processor, ARMv6 instruction set, and 4GB of RAM. It also includes Video Core IV GPU. For booting & storing data SD card is used, no hard disk or solid state drive is provided. On the board there are many interfaces, for example USB, Ethernet, video and audio and 26 (GPIO) General Purpose Input Output pins. Python is main programming language for Raspberry Pi.

Our experiment is based on Raspberry Pi board. We capture the video of the driver using the extended camera on the Raspberry Pi board using OpenCV module of Python and then convert the video into frames and do the further manipulation.

Component Model Configuration

- Board: Raspberry Pi 4 Ram 4 GB
- CPU: 64-bit Quad Core Cortex-A72
- Memory Card: 32 GB
- Camera: Raspberry Pi Camera V2

Fig 4.1: Raspberry Pi 4 Ram 4 GB

Fig 4.2: Raspberry Pi Camera V2

4.2 SAMPLE CODE

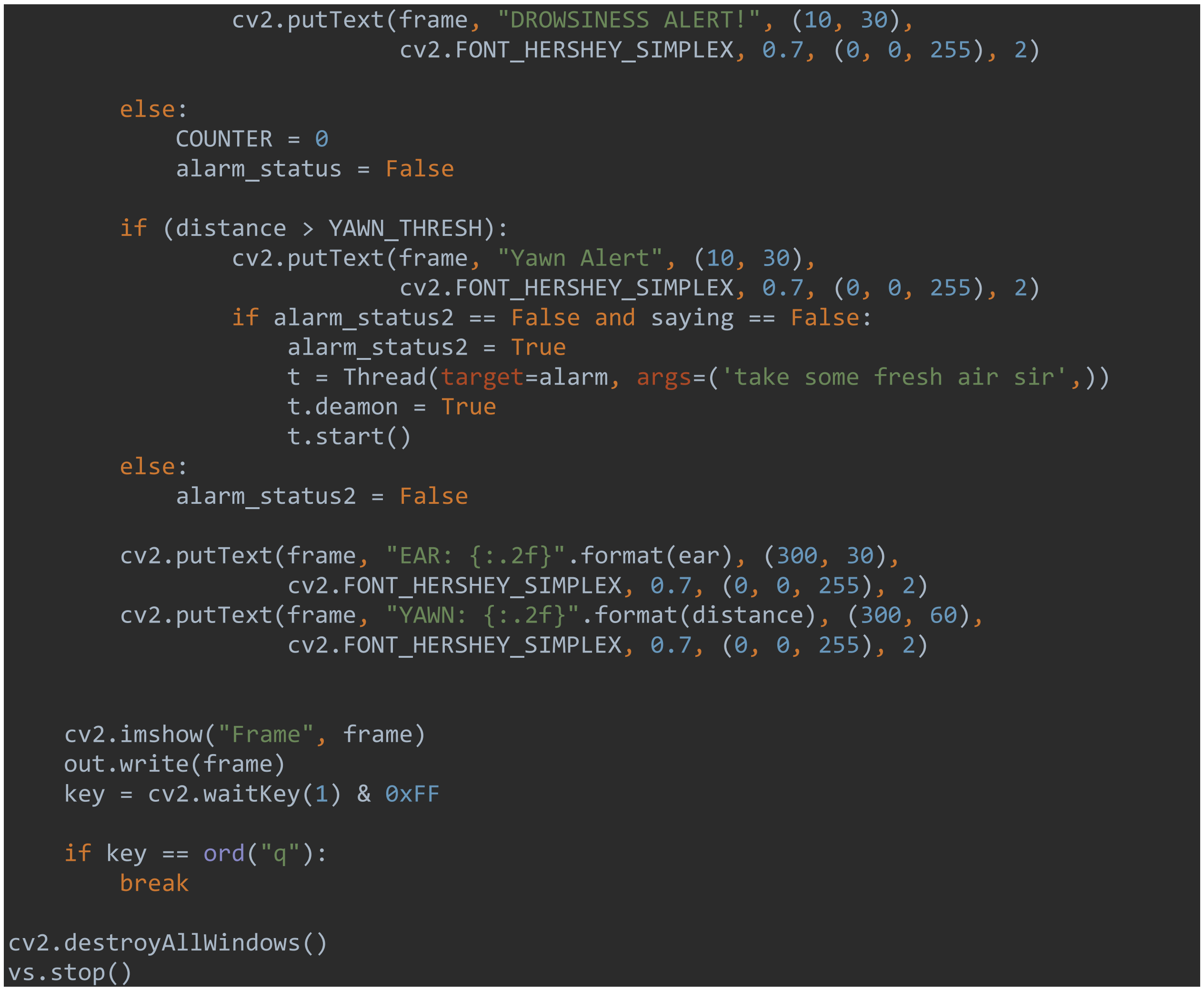
To start our implementation, we have created a python file and written the following code.

We have used various used various packages in order to build the project which are explained below:

- **SciPy package:** To compute the Euclidean distance between facial landmarks points in the eye aspect ratio and mouth aspect ratio calculation.
- **imutils package:** To make working with OpenCV easier in computer vision and image processing functions.
- **Thread class:** To play alarm in a separate thread from the main thread to ensure our script doesn't pause execution while the alarm sounds.
- **pygame package:** To play alarm.
- **dlib package:** It contains our implementation of facial landmark detection.

```python
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
from pygame import mixer


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

    mixer.init()
    sound = mixer.Sound(r"J:\Project\alarm.wav")
    while alarm_status:
        sound.play()
        print('call')
        s = 'espeak "'+msg+'"'
        os.system(s)

    if alarm_status2:
        sound.play()
        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 = 0.25
EYE_AR_CONSEC_FRAMES = 20
YAWN_THRESH = 17
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
```

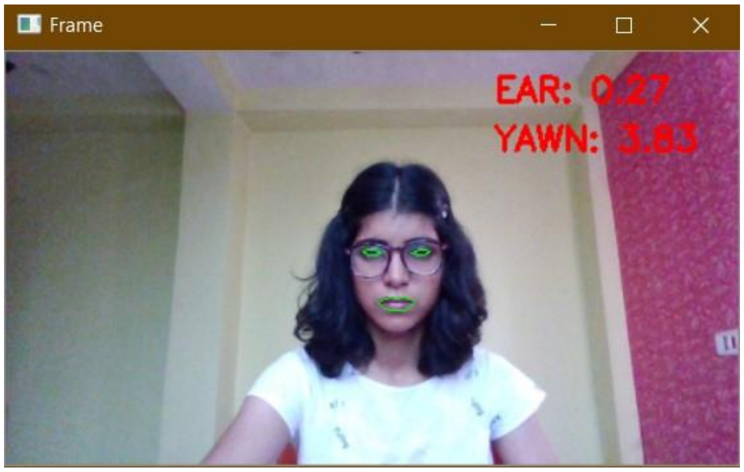
```
but less accurate
predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')


print("-> Starting Video Stream")
vs = VideoStream(src=args["webcam"]).start()
#vs= VideoStream(usePiCamera=True).start()       //For Raspberry Pi
time.sleep(1.0)
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('output.avi', fourcc, 20.0, (640, 450))
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)
    out.write(frame)
    key = cv2.waitKey(1) & 0xFF

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

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

Fig 4.3: Code for detecting drowsiness

4.3 RESULTS



```
EAR: 0.27
YAWN: 3.83
```

Fig 4.4: Normal Mode

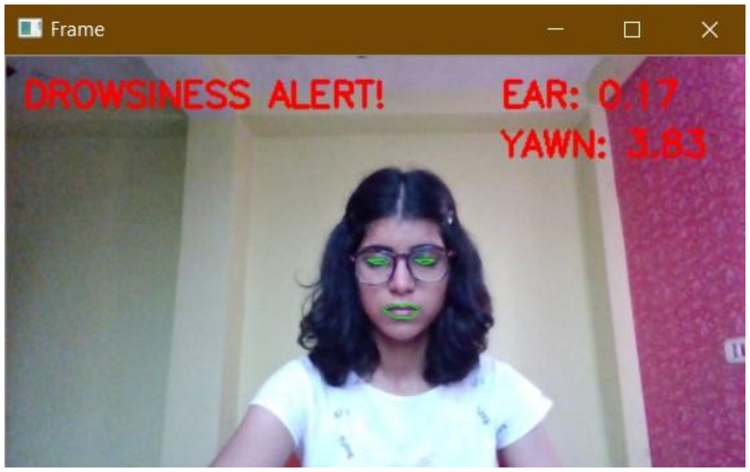


Fig 4.5: Drowsiness Alert

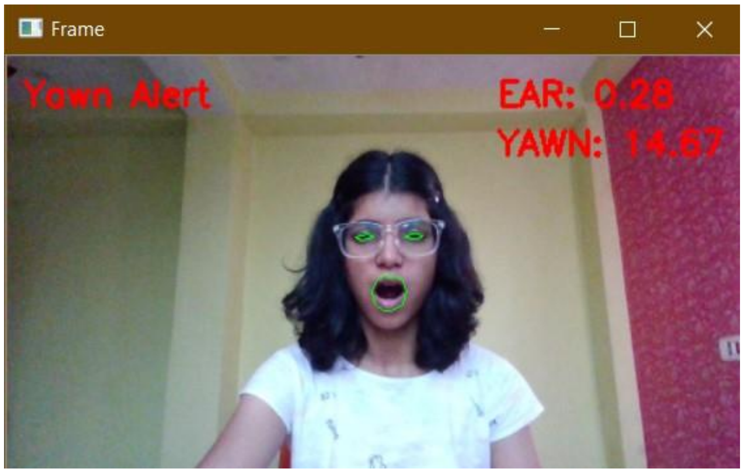


Fig 4.6: Yawn Alert

5.CONCLUSION AND FUTURE SCOPE

5.1 CONCLUSION

The current study developed an automated system for detecting drowsiness of the driver. This is a car safety technology which helps prevent accidents caused by the driver getting drowsy. This system will capture images as a video stream through a camera, detect the face and localize the eyes and mouth. The eyes and mouth will then be analyzed for drowsiness detection. Based on the result, the driver will be alerted for drowsiness through an alarm system.
The Drowsiness Detection System developed based on eye closure of the driver can differentiate normal eye blink and drowsiness also yawning. The system works well even in case of drivers wearing spectacles and even under low light conditions if the camera delivers better output.

5.2 DEPENDENCIES

- **Distance of camera from driver face:** The driver should be seated at a well distance from the camera such that his/her face is properly visible in the frame. Sitting too far or to close to the camera will not produce accurate result.

- **Use of Goggles**: In case the user uses colorful goggles then it is difficult for the algorithm to detect the state of the eye. As it hugely depends on eyes for any calculation thus can produce inaccurate result.

- **Multiple face detection:** If multiple face arises in the window then the camera may detect more number of faces undesired output may appear. Because of different condition of different faces. So we need to make sure that only the driver face come within the range of the camera. Also the speed of detection reduces because of operation on multiple faces.

- **Dependence on ambient light:** The model developed for this purpose strongly depends on the ambient light condition. It requires certain minimum level of light conditions for the camera to produce valid input otherwise it becomes very difficult to detect. To avoid this error, we can use either LED light for better detection or we can use an infrared camera.

5.3 FUTURE WORK

Our model is designed for detection of drowsy state of eye and give and alert signal or warning may be in the form of audio or any other means. But the response of driver after being warned may not be sufficient enough to stop causing the accident meaning that if the driver is slow in responding towards the warning signal then accident may occur. Hence to avoid this we can design and fit a motor driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically. Also we can avoid the use of Raspberry Pi which is not so fast enough for video processing by choosing our own mobile phone as the hardware. This can be done by developing a proper mobile application which will perform the same work as Raspberry Pi and response will be faster and effective.

6. References:

- https://pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
- https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/
- http://www.ee.ryerson.ca/~phiscock/thesis/drowsydetector/drowsydetector.pdf
- http://www.cnblogs.com/skyseraph/archive/2011/02/24/1963765.html
- http://www.scribd.com/doc/15491045/Learning-OpenCV-Computer-Vision-with-theOpenCV-Library
- http://opencv.willowgarage.com/documentation/reading_and_writing_images_and_video.html
- http://www.scribd.com/doc/46566105/opencv
- https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/
- http://note.sonots.com/SciSoftware/haartraining.html