

Driver Drowsiness Detection System

A graduate project report submitted to AKTU in partial fulfilment
of the requirement for the award of the degree of

Bachelor of Technology
In
Computer Science and Engineering

SUBMITTED BY

Shivangi Singh	1906410108002
Jayati Maurya	1864110030
Anjali Srivastava	1864110010
Divyanshu Srivastava	1864110023
Diksha Maurya	1864110022

UNDER THE GUIDANCE OF

Mr. Gaurav Ojha
(Assistant Professor)



Department of Computer Science & Engineering
Ashoka Institute of Technology and Management

(A constitute of Dr. A.P.J. Abdul Kalam Technical University)

Varanasi – 221007



May, 2022

CERTIFICATE

This is to certify that report entitled “**Driver Drowsiness Detection System**” which is submitted by **Shivangi Singh (1906410108002), Jayati Maurya (1864110030), Anjali Srivastava(1864110010), Divyanshu Srivastava (1864110023), Diksha Maurya (1864110022)** in partial fulfilment of the requirement for the award of degree **Bachelor of Technology in Computer Science & Engineering** to **Ashoka Institute Of Technology & Management**, affiliated to **A.K.T.U, Lucknow** is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Supervisor:
Mr. Gaurav Ojha
(Asst. Professor)

Dept. of Computer Science & Engineering

Forwarded By:
Mr. Arvind Kumar
(Asst. Professor & Head)
Dept. of Computer Science & Engineering

Date:

DECLARATION

We hereby declare that the submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by any another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Sign.:

Shivangi Singh

Roll No. 1906410108002

Date:

Sign.:

Jayati Maurya

Roll No. 1864110030

Sign.:

Anjali Srivastava

Roll No. 1864110010

Sign.:

Divyanshu Srivastava

Roll No. 1864110023

Sign.:

Diksha Maurya

Roll No. 1864110022

ACKNOWLEDGEMENT

We wish to express my heartfelt gratitude to all the people who have played a crucial role in the research for this project without their active cooperation, the preparation of this project could not have been completed within the specified time limit.

We are profoundly grateful to my project guide Mr. Gaurav Ojha for his expert guidance and continuous encouragement throughout to see that this project meets its target since its commencement to its completion.

We are thankful to Mr. Arjun Mukherjee of the CSE department for his valuable advice and guidance.

Sign.:

Shivangi Singh

Roll No. 1906410108002

Date:

Sign.:

Jayati Maurya

Roll No. 1864110030

Sign.:

Anjali Srivastava

Roll No. 1864110010

Sign.:

Divyanshu Srivastava

Roll No. 1864110023

Sign.:

Diksha Maurya

Roll No. 1864110022

ABSTRACT

Many of the accidents occur due to drowsiness of drivers. It is one of the critical causes of road accidents now-a-days. Latest statistics say that many of the accidents were caused because of the drivers falling asleep. For the prevention of this, a system is introduced which detects drowsiness and alerts the drivers. This system uses image processing techniques which mainly focuses on the face and eyes of the driver. The model extracts the driver's face and predicts the blinking of the eye from the eye region.

This document is a review report on the research conducted and the project made in the field of computer engineering to develop a system for driver drowsiness detection to prevent accidents from happening because of driver fatigue and sleepiness. The report proposed the results and solutions on the limited implementation of the various techniques that are introduced in the project. Whereas the implementation of the project gives the real-world idea of how the system works and what changes can be done to improve the utility of the overall system.

Furthermore, the paper states the overview of the observations made by the authors to help further optimization in the mentioned field to achieve the utility at a better efficiency for a safer road.

Keywords-Driver drowsiness; eye detection; face detection; fatigue.

LIST OF TABLES

S. NO.	TOPIC	PAGE NO.
1	Table 1: Test Cases	40

LIST OF FIGURES

S. NO.	TOPIC	PAGE NO.
1	Figure 1: Libraries Used	16
2	Figure 2: System Model	18
3	Figure 3: Facial Landmarks	20
4	Figure 4: Eye Landmarks of Open & Closed Eyes	21
5	Figure 5: Define EAR	21
6	Figure 6: Use Case Diagram	23
7	Figure 7: Activity Diagram	24
8	Figure 8: Class Diagram	25
9	Figure 9: Block Diagram of Face Training	29
10	Figure 10: Block Diagram of Face Recognition Process	30
11	Figure 11: Working of Landmark Detector	31
12	Figure 12: Application of Convex Hull	32
13	Figure 13: Set of Points	33
14	Figure 14: Convex Hull	33
15	Figure 15: Code	34
16	Figure 16: Code	34
17	Figure 17: Code	35
18	Figure 18: EAR Mapped	37
19	Figure 19: Demonstration	40

LIST OF ABBREVIATIONS

S. NO.	TOPIC	PAGE NO.
1	EAR: Eye Aspect Ratio	37
2	TED: Truly Eye Detect	37
3	FED: Falsely Eye Detect	37
4	TAD: Truly Alarm Detect	37
5	FAD: Falsely Alarm Detect	37

TABLE OF CONTENTS

TOPIC	PAGE NO.
Certificate	I
Declaration	II
Acknowledgement	III
Abstract	IV
List of tables	V
List of figures	VI
List of abbreviations	VII
Chapter 1 – Introduction	1-3
1.1 Motivation	2
1.1.1 Human Psychology with Current Technology	2
1.1.2 Facts & Statistics	2
1.2 Intended Audience	3
1.3 Product Scope	3
1.4 Problem Definition	3
Chapter 2 - Literature Survey	4-10
2.1 System Review	4
2.2 Literature Review	4
2.2.1 Face & Eye Detection by Machine Learning Algorithm	4
2.2.2 Behavioral Based Techniques	4
2.3 Proposed System	6
2.4 Technologies Used	7
Chapter 3 – Requirements Specification	11-12
3.1 Software	12

3.2 Hardware	12
Chapter 4 - Requirement Analysis	13-16
4.1 Python	14
4.2 Libraries	14
4.3 Operating System	16
4.4 Laptop	16
4.5 Webcam	16
Chapter 5 - Project Planning	17-21
5.1 System Model	18
5.2 Drowsy Eye & Face Detection	19
Chapter 6 - System Design	22-25
6.1 Use Case Diagram	23
6.2 Activity Diagram	24
6.3 Class Diagram	25
Chapter 7 - Approach & Implementation	26-35
7.1 Approach	27
7.2 Implementation	27
7.3 Methodology	27
7.3.1 Image Processing	27
7.3.2 Frontal Face Detector	28
7.3.3 Shape Predictor	30
7.3.4 Video Capture	31
7.3.5 Color Image to Grayscale	31
7.3.6 Convex Hull	32
Chapter 8 - Result Discussion & Performance Analysis	36-38
8.1 Prediction of Drowsiness	37
8.2 Performance Analysis	37
Chapter 9 - System Testing	39-40
9.1 Test Cases & Test Results	40

Chapter 10 – Conclusion and Future Implementation	41
10.1 Conclusion	42
10.2 Future Implementation	42
References	43

Chapter 1

INTRODUCTION

1.1 MOTIVATION

Every year, the National Highway Traffic Safety Administration (NHTSA) and World Health Organisation (WHO) have reported that approximately 1.35 million people die due to vehicle crashes across the world. Generally, road accidents mostly occur due to inadequate way of driving. These situations arise if the driver is addicted to alcohol or in drowsiness. The maximum types of lethal accidents are recognised as a severe factor of tiredness of the driver. When drivers fall asleep, the control over the vehicle is lost. There is a need to design smart or intelligent vehicle system through advanced technology. This paper implements a mechanism to alert the driver on the condition of drowsiness or daydreaming.

1.1.1 Human Psychology with Current Technology

Humans have always invented machines and devised techniques to ease and protect their lives, for mundane activities like travelling to work, or for more interesting purposes like aircraft travel. With an advancement in technology, modes of transportation kept on advancing and our dependence on it started increasing exponentially. It has greatly affected our lives as we know it. In modern times, almost everyone in this world uses some sort of transportation every day. Some people are rich enough to have their own vehicles while others use public transportation. However, there are some rules and codes of conduct for those who drive irrespective of their social status. One of them is staying alert and active while driving.

Neglecting our duties towards safer travel has enabled hundreds of thousands of tragedies to get associated with this wonderful invention every year. It may seem like a trivial thing to most folks but following rules and regulations on the road is of utmost importance. While on road, an automobile wields the most power and in irresponsible hands, it can be destructive and sometimes, that carelessness can harm lives even of the people on the road. One kind of carelessness is not admitting when we are too tired to drive. In order to monitor and prevent a destructive outcome from such negligence, many researchers have written research papers on driver drowsiness detection systems. In current years, drowsy driver detection is the most necessary procedure to prevent any road accidents, probably worldwide. The aim of this project is to construct a smart alert technique for building intelligent vehicles that can automatically avoid drowsy driver impairment. But drowsiness is a natural phenomenon in the human body that happens due to different factors. Hence, it is required to design a robust alert system to avoid the cause of the mishap. In this proposed paper, we address a drowsy driver alert system that has been developed using such a technique in which the Video Stream Processing (VSP) is analysed by eye blink concept through an Eye Aspect Ratio (EAR) and Euclidean distance of the eye.

1.1.2 Facts & Statistics

Our current statistics reveal that just in 2015 in India alone, 148,707 people died due to car related accidents. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. Fatigue, in general, is very difficult to measure or observe unlike alcohol

and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem as its implementation is very difficult and costly.

1.2 INTENDED AUDIENCE

The intended audiences for this document are the development team, the project evaluation jury, and other tech-savvy enthusiasts who wish to further work on the project.

1.3 PRODUCT SCOPE

There are many products out there that provide the measure of fatigue level in the drivers which are implemented in many vehicles. The driver drowsiness detection system provides similar functionality but with better results and additional benefits. Also, it alerts the user on reaching a certain saturation point of the drowsiness measure.

1.4 PROBLEM DEFINITION

Fatigue is a safety problem that has not yet been deeply tackled by any country in the world mainly because of its nature. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative.

Chapter 2

LITERATURE SURVEY

2.1 SYSTEM REVIEW

This survey is done to comprehend the need and prerequisite of the general population, and to do as such, we went through different sites and applications and looked for the fundamental data. Based on these data, we made an audit that helped us get new thoughts and make different arrangements for our task. We reached the decision that there is a need for such an application and felt that there is a decent extent of progress in this field too.

2.2 LITERATURE REVIEW

Drowsiness of the driver can be determined with different aspects using vehicle-based, psychological, and behavioural measurements implemented through different predictive algorithms as discussed in the following sections.

2.2.1. Face and Eye Detection by Machine Learning (ML) and Deep Learning (DL) Algorithms

Convolutional Neural Network (CNN) technique of the ML algorithm to detect microsleep and drowsiness. In this paper, detection of driver's facial landmarks can be achieved through a camera that is then passed to this CNN algorithm to properly identify drowsiness. Here, the experimental classification of eye detection is performed through various data sets like without glasses and with glasses in day or night vision. So, it works for effective drowsiness detection with high precision with android modules. The algorithm of Deep CNN was used to detect eye blink and its state recognition. It specially focuses on construction of real-time fatigue detection to prevent roadside accidents. This system formulates a number of drivers' faces, which works on multi-layered 3D CNN models to identify drowsy drivers and provide a 92-percentage acceptance rate.

2.2.2. Behavioural Based Techniques

The different techniques used in behavioural based parameters are:

- Eye tracking and dynamic template matching
To avoid road accidents, a real time driver fatigue detection system based on vision is proposed. Firstly, the system detects the face of the driver from the input images and then the operator is used to locate the eyes positions and gets the images of the eye as the dynamic template for the tracking of the eye. Then the obtained images are used to determine whether the eyes are closed or open to judge the drowsiness of the driver.
- Mouth and yawning analysis
Fatigue is the main reason for road accidents. To avoid the issue, the driver fatigue detection system based on mouth and yawning analysis was proposed. Firstly, the system locates and tracks the mouth of a driver using a cascade of classifier training and mouth detection from the input images. Then, the images of mouth and yawning are trained using SVM. In the

end, SVM is used to classify the regions of the mouth to detect the yawning and alerts the fatigue.

- Facial expression method

Laboratory condition using Finite Element Analysis is used by the researchers which is a complex system that contains the databases of facial expression as a template and detects drowsiness based on results from the database. Similarly, the hardware-based driver drowsiness detection system based on facial expression is presented. The hardware system uses infrared light as it has given many benefits like ease of use, independent of lightning conditions of the environment. The system firstly uses the technique of background subtraction to determine the face region from the input images. Then using horizontal projection and template matching, facial expressions are obtained. After that in the tracking phase, elements found earlier are followed up using template matching and then investigates the incidence of sleepiness using the determination of facial states from the change in facial components. Changes in the three main elements such as eyebrow rising, yawning and eye closing for the certain period are taken as the initial indications for the drowsiness and the system generates the alert. The experiment was performed in the real driving scenario. For testing, images are acquired by the webcam under different conditions of lighting and from different people.

- Eye closure and head posture method

In the driver drowsiness detection system using eye closure and head posture method, firstly, video is captured using a webcam. To detect ROI (face and eyes), viola-jones method is used. The face is partitioned into three areas and the top one presenting the eye area is browsed. Then to detect the eye state, a neural network is used to train the images then the coefficients learning images are compared with the coefficients of the testing images and tells which class it belongs to. When the closed eye is identified in the frames then the eye closure duration is calculated, if the value exceeds the pre-defined time, then the drowsiness state is detected. Then the developed system estimates the head movements which are: left, right, backward, forward inclination and left or right rotation. The captured video is segmented into frames and extracts the images of head and determines the coordinates of images. Then the images are compared to determine the inclined state of head and same case with other head postures. Finally, the system combines the eye closure duration and head posture estimation to measure the drowsiness.

2.3 PROPOSED SYSTEM

The proposed system here is designed to minimise the occurrence of countless mishaps due to the drowsy driver. Nowadays, driver fatigue causes road accidents every now and then across the world. So, these activities should be required to automatically handle an implementation of smart alert system or vigilance in a vehicle which is an objective of this system. To analyse different behavioural or visual-based attitudes of the driver, face movement and eye blink are measured to study the state of the driver. Here, eye blink is mainly focused to detect drowsiness of the driver. The threshold value of an EAR lies above 0.25 without any effect of exhaustion. When a driver automatically shuts down, then the threshold value of EAR falls below the given range. A threshold value of a drowsy eye blink sample represents the number of video frames

of the driver's closed eyes. If the consecutive counting frames increase above the range of the threshold value, then the drowsiness of the driver is detected. Here, a Pi camera is used to regularly record the total movement of an eye through which the threshold value of an EAR is calculated. A counter is also included in it for counting occurrence of frames

2.4 TECHNOLOGY USED

a. **PYTHON** - Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its 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 supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python programming language (latest Python 3) is being used in web development, Machine Learning applications, along with all cutting-edge technology in the Software Industry. The biggest strength of Python is huge collection of standard libraries which can be used for the following:

- Machine Learning
- GUI Applications (like Kivy, Tkinter, PyQt etc.)
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like OpenCV, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia
- Scientific computing
- Text processing and many more.

b. **JUPYTER Lab** - Project Jupyter is a non-profit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data. Its flexible interface allows users to configure and arrange workflows in data science, scientific computing, computational journalism, and machine learning.

c. **IMAGE PROCESSING** - In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images.

Image processing is the process of transforming an image into a digital form and performing certain operations to get some useful information from it. The image processing system usually treats all images as 2D signals when applying certain predetermined signal processing methods.

There are five main types of image processing:

- Visualization - Find objects that are not visible in the image
- Recognition - Distinguish or detect objects in the image
- Sharpening and restoration - Create an enhanced image from the original image
- Pattern recognition - Measure the various patterns around the objects in the image
- Retrieval - Browse and search images from a large database of digital images that are like the original image

Applications of Image Processing

• **Traffic Sensing Technologies**

In the case of traffic sensors, we use a video image processing system or VIPS. This consists of a) an image capturing system b) a telecommunication system and c) an image processing system. When capturing video, a VIPS has several detection zones which output an “on” signal whenever a vehicle enters the zone, and then output an “off” signal whenever the vehicle exits the detection zone. These detection zones can be set up for multiple lanes and can be used to sense the traffic in a particular station. Besides this, it can auto record the license plate of the vehicle, distinguish the type of vehicle, monitor the speed of the driver on the highway and lots more.

• **Face Detection**

One of the most common applications of image processing that we use today is face detection. It follows deep learning algorithms where the machine is first trained with the specific features of human faces, such as the shape of the face, the distance between the eyes, etc. After teaching the machine these human face features, it will start to accept all objects in an image that resemble a human face. Face detection is a vital tool used in security, biometrics and even filters available on most social media apps these days.

• **Image Reconstruction**

Image processing can be used to recover and fill in the missing or corrupt parts of an image. This involves using image processing systems that have been trained extensively with existing photo datasets to create newer versions of old and damaged photos.

Benefits of Image Processing

The implementation of image processing techniques has had a massive impact on many tech organizations. Here are some of the most useful benefits of image processing, regardless of the field of operation:

- The digital image can be made available in any desired format (improved image, X-Ray, photo negative, etc)
- It helps to improve images for human interpretation
- Information can be processed and extracted from images for machine interpretation
- The pixels in the image can be manipulated to any desired density and contrast
- Images can be stored and retrieved easily
- It allows for easy electronic transmission of images to third-party providers

d. **MACHINE LEARNING** - Machine learning is the scientific study of algorithms and statistical models that computer systems use to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", to make predictions or decisions without being explicitly told.

Machine learning is an important component of the growing field of data science. Using statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Working of Machine Learning

UC Berkeley breaks out the learning system of a machine learning algorithm into three main parts:

Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabelled, your algorithm will produce an estimate about a pattern in the data.

Error Function: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.

Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

Machine learning classifiers fall into three primary categories

- **Supervised machine learning**

Supervised machine learning is defined by its use of labelled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross-validation process to ensure that the model avoids overfitting or underfitting.

- **Unsupervised machine learning**

Unsupervised machine learning uses machine learning algorithms to analyse and cluster unlabelled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention.

- **Semi-supervised learning**

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labelled data set to guide classification and feature extraction from a larger, unlabelled data set. Semi-supervised learning can solve the problem of having not enough labelled data (or not being able to afford to label enough data) to train a supervised learning algorithm.

Chapter 3

REQUIREMENTS SPECIFICATIONS

3.1 SOFTWARE

(a) Python

- Python 3

(b) Libraries

- NumPy
- SciPy
- Playsound
- Dlib
- Imutils
- OpenCV, etc.

(c) Operating System

- Windows or Ubuntu

3.2 HARDWARE

(a) Laptop with basic hardware

(b) Webcam

Chapter 4

REQUIREMENT ANALYSIS

4.1 PYTHON

Python is the basis of the program that we wrote. It utilizes many of the python libraries.

4.2 LIBRARIES

• NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. It is applied for scientific programming in Python, especially for numbers. It comprises multidimensional objects in arrays and a package of integrating tools for Python implementation. NumPy is basically a mix of C and Python used as an alternative for traditionally used MATLAB programming, where data in the form of numerals are treated as arrays for multidimensional functions and rearrangement operations.

Features

- It is a combination of C and python.
- Multidimensional homogeneous arrays.
- Various functions for arrays.
- Reshaping of arrays.
- Python can be used as an alternative to MATLAB.

Advantages

1. NumPy arrays take less space: NumPy's arrays are smaller in size than Python lists. A python list could take upto 20MB size while an array could take 4MB. Arrays are also easy to access for reading and writing.
2. The speed performance is also great. It performs faster computations than python lists: As it is open-source, it doesn't cost anything, and it uses a very popular programming language, Python, which has high-quality libraries for almost every task. Also, it is easy to connect the existing C code to the Python interpreter.

• SciPy

SciPy is an open-source scientific library of Python that is distributed under a BSD license. It is used to solve complex scientific and mathematical problems. It provides many user-friendly and effective numerical functions for numerical integration and optimization. The SciPy library supports integration, gradient optimization, special functions, ordinary differential equation solvers, parallel programming tools, and many more. We can say that SciPy implementation exists in every complex numerical computation. The SciPy is a

data-processing and system-prototyping environment as similar to MATLAB. It is easy to use and provides great flexibility to scientists and engineers.

SciPy contains significant mathematical algorithms that provide easiness to develop sophisticated and dedicated applications. Being an open-source library, it has a large community across the world to the development of its additional module, and it is much beneficial for scientific application and data scientists.

- **Playsound:** Used for sounding the alarm

- **Dlib**

DLib is an open-source C++ library implementing a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction.

DLib also features utility functionality including

- Threading,
- Networking,
- Numerical Algorithms,
- Image Processing,
- and Data Compression and Integrity algorithms.

Dlib is a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face

- **Imutils**

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

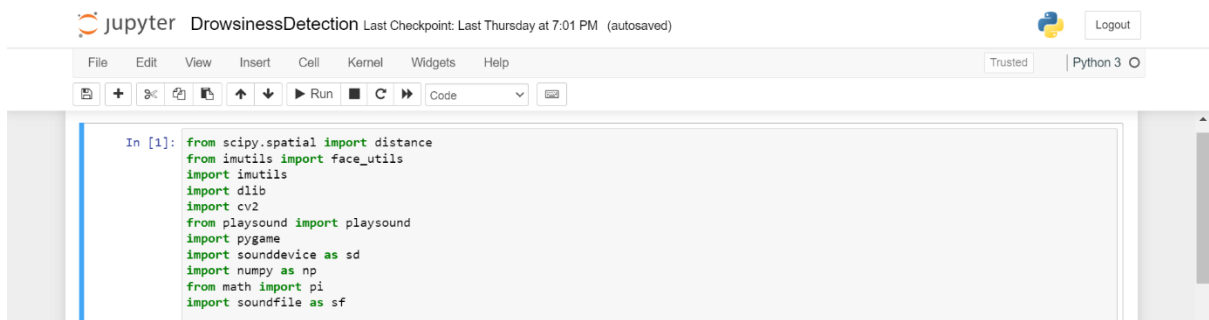
- **OpenCV**

OpenCV is the huge open-source library for computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When integrated with various libraries, such as NumPy, python can process the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

Applications of OpenCV

There are lots of applications which are solved using OpenCV, some of them are listed below:

- Face recognition
- Automated inspection and surveillance
- Number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition



The screenshot shows a Jupyter Notebook window titled "DrowsinessDetection". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a code editor. The code cell contains the following imports:

```
In [1]: from scipy.spatial import distance
        from imutils import face_utils
        import imutils
        import dlib
        import cv2
        from playsound import playsound
        import pygame
        import sounddevice as sd
        import numpy as np
        from math import pi
        import soundfile as sf
```

Figure 1: Libraries Used

4.3 OPERATING SYSTEM

Program is tested on Windows 10 build 1903 and PopOS 19.04

4.4 LAPTOP

Used to run our code.

4.5 WEBCAM

Used to get the video feed.

Chapter 5

PROJECT PLANNING

5.1 SYSTEM MODEL

The framework is created utilizing the incremental model. The centre model of the framework is first created and afterwards augmented in this way in the wake of testing at each turn. The underlying undertaking skeleton was refined into expanding levels of ability.

At the following incremental level, it might incorporate new execution backing and improvement.

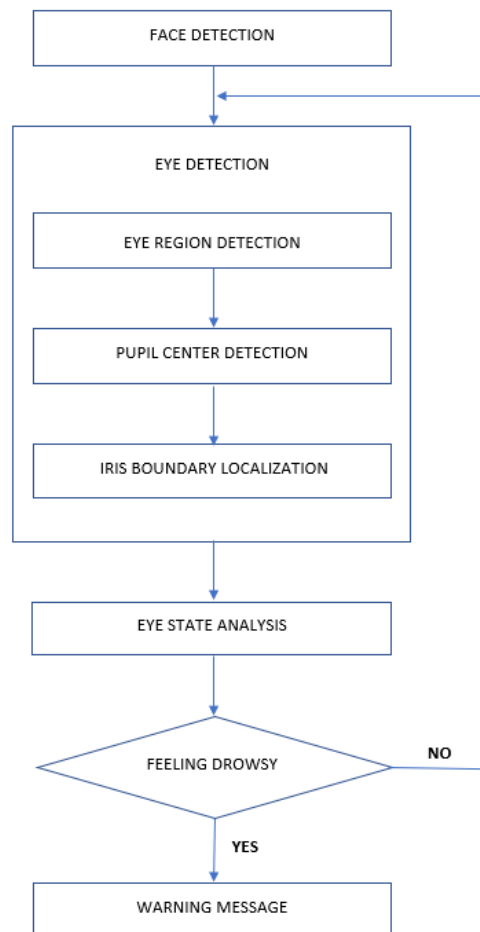
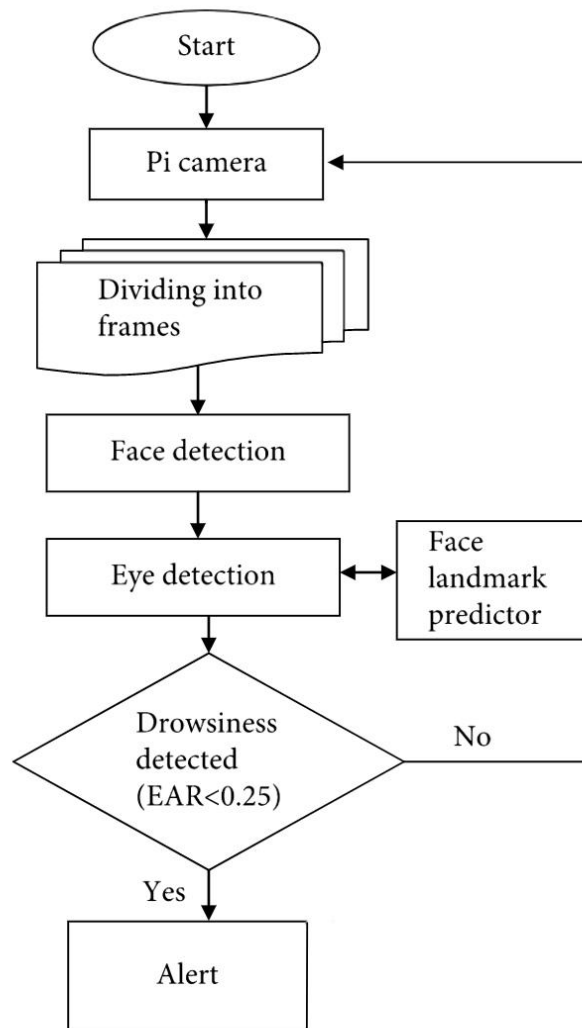


Figure 2: System Model

5.2. DROWSY EYE AND FACE DETECTION

The step-by-step methods for detection of drowsy are as follows:

- (i) Step 1: video recording
- (ii) Step 2: face detection
- (iii) Step 3: eye detection
- (iv) Step 4: drowsiness detection (combination of steps 2 and 3)



The camera captures a continuous video stream which is converted into several frames which are forwarded to the face detection step. The face detection is analysed by face landmark algorithm, which can detect eye, mouth and nose from facial appearance as shown in figure 1.

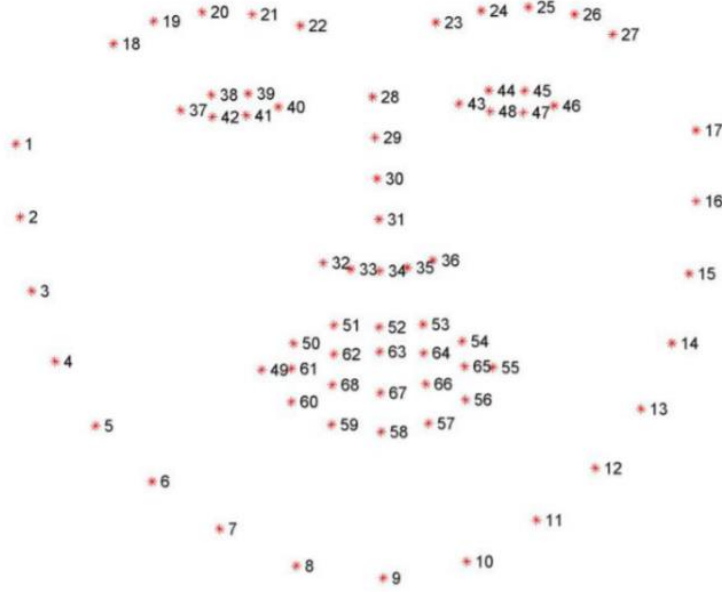


Figure 3: Facial Landmarks

This detection technique with python-based library packages like OpenCV. OpenCV is used for real-time image processing which is implemented by computer vision algorithms. When the facial features are successfully detected, the next step, eye detection of drowsy drivers, is possible to focus through facial landmark predictor algorithms. So, it can convert that image frame format to grayscale level, where detected eye areas are traced by six coordinates. Now it is needed to calculate EAR which measures the distance between vertical and horizontal eye landmark points by using Euclidean Distance (ED) method. The calculation of the distance of the eyelids section is made as

$$iED(X_i, Y_i) = \sqrt{\sum_{n=1}^n (Y_i - X_i)^2},$$

where, ED (X, Y) is denoted as a Euclidean distance between X_i and Y_i ; these are two cartesian coordinate points. So, it is represented in python program which is explained below:

- (i) $A = \text{distance.euclidean}(\text{eye}[1], \text{eye}[5])$
- (ii) $B = \text{distance.euclidean}(\text{eye}[2], \text{eye}[4])$
- (iii) $C = \text{distance.euclidean}(\text{eye}[0], \text{eye}[3])$

From the above statement, distance is an object of distance package that belongs to the SciPy library file which is invoked as a Euclidean (X_i, Y_i). Here, there are two coordinate points of an eye landmark. The variables (A, B, and C) are used for calculation of the EAR value of an

eye. In order to find the EAR by using detected eye landmark coordinate values from every video frame, it is being computed as

$$EAR = \frac{\|x_2 - x_6\| + \|x_3 - x_5\|}{2\|x_1 - x_4\|},$$

where x_1, x_2, x_3, x_4, x_5 and x_6 are landmark coordinates.

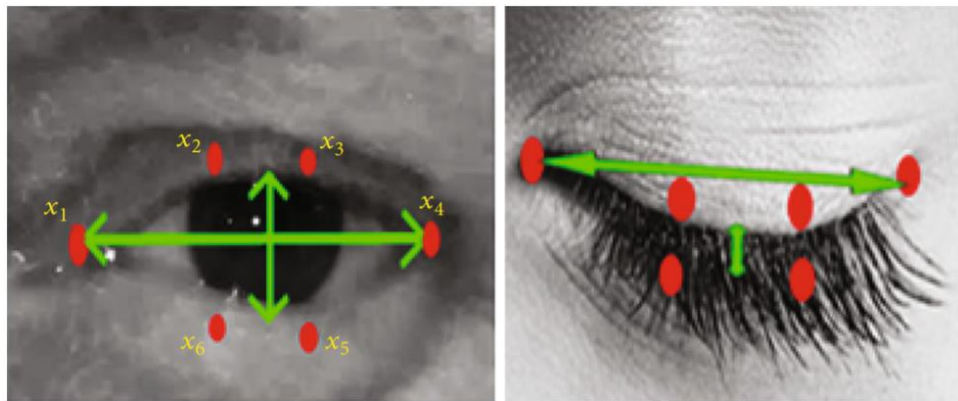


Figure 4: Eye landmarks (6 coordinates) of open and closed eyes

The EAR value of the left and right eyes is averaged during synchronous eye blink. The threshold value of EAR remains constant when both eyes continue to be open, where values will be randomly changed during eye blink. The threshold range of EAR is above 0.25 that means the driver's eyes are unlatched. If any drowsiness of the driver is detected from video frames, it happens due to a drop of its threshold value below 0.25.

```
def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear
```

Figure 5: Define EAR

Chapter 6

SYSTEM DESIGN

6.1 USE CASE DIAGRAM

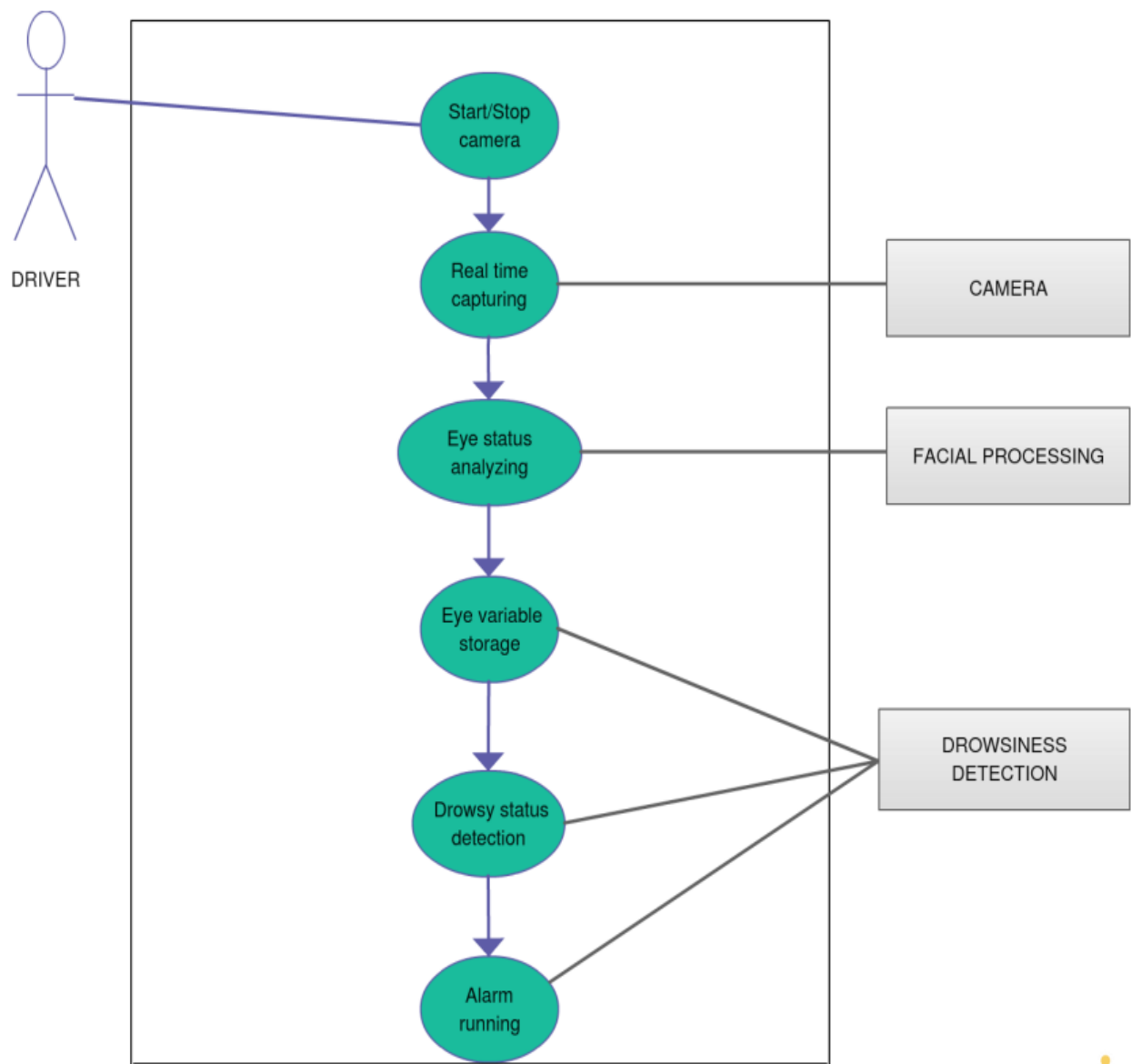


Figure 6: Use Case Diagram

6.2 ACTIVITY DIAGRAM

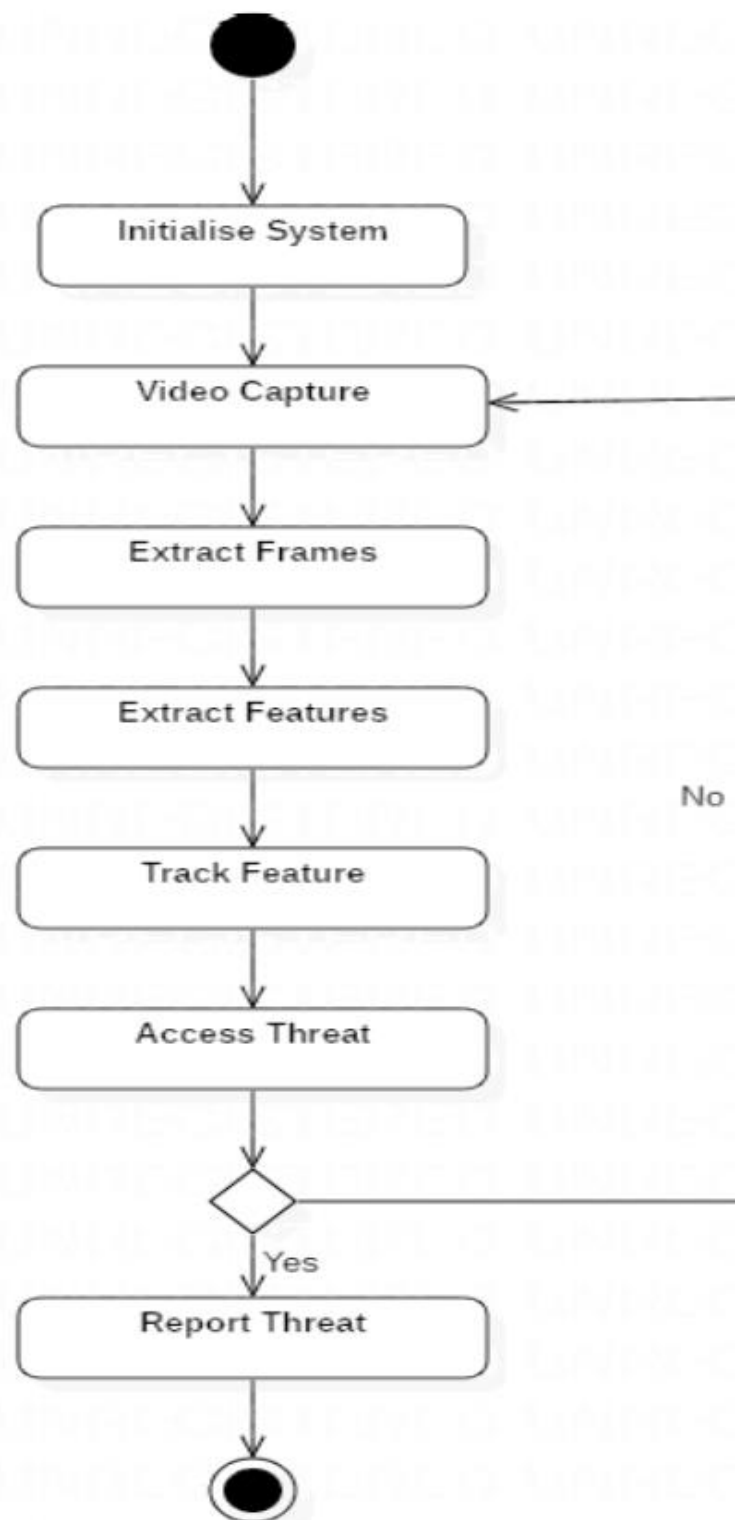


Figure 7: Activity Diagram

6.3 CLASS DIAGRAM

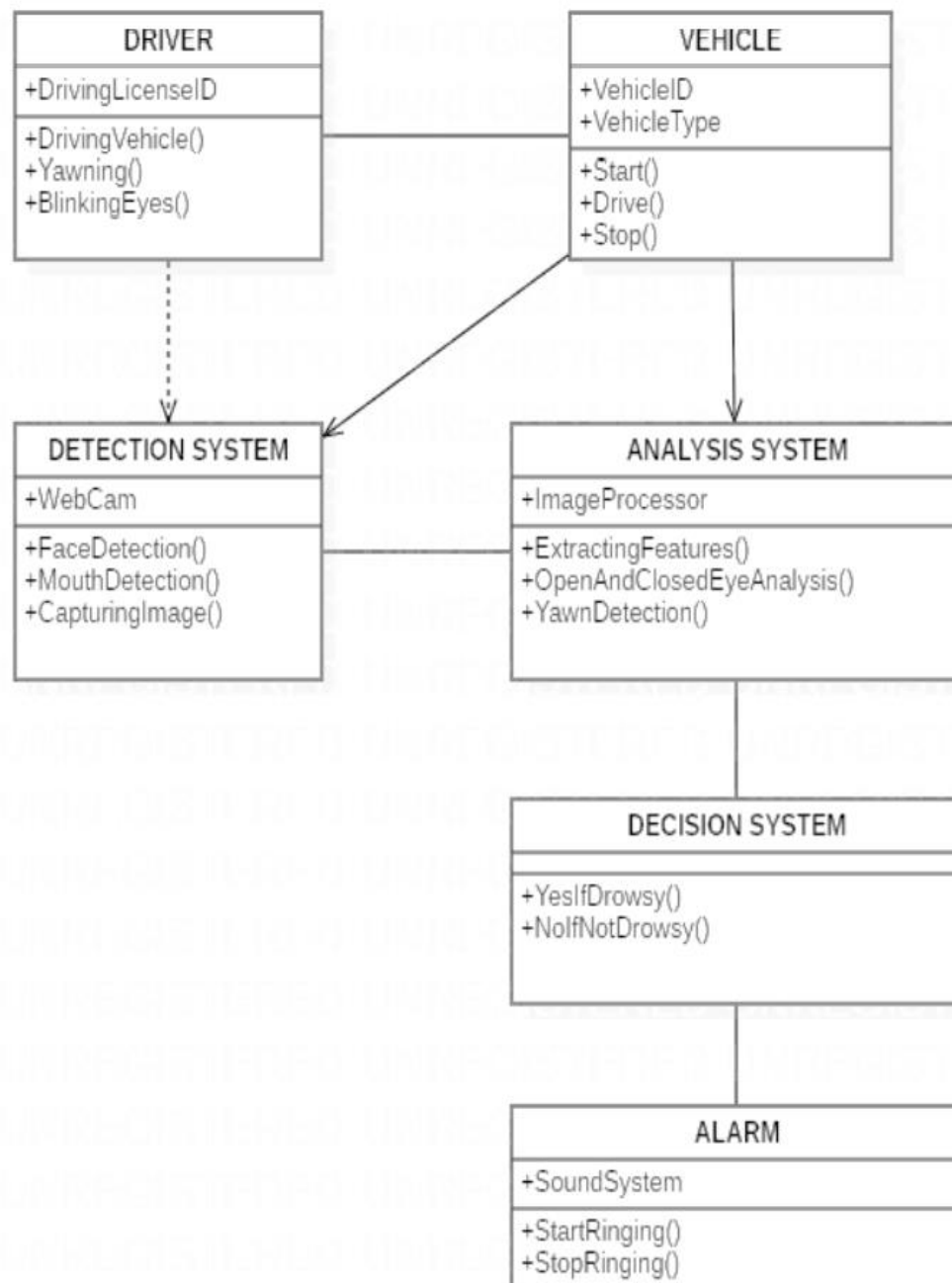


Figure 8: Class Diagram

Chapter 7

APPROACH & IMPLEMENTATION

7.1 APPROACH

The approach we will be using for this Python project is as follows:

Step 1 – Take image as input from a camera.

Step 2 – Detect the face in the image and create a Region of Interest (ROI).

Step 3 – Detect the eyes from ROI and feed it to the classifier.

Step 4 – Classifier will categorize whether eyes are open or closed.

Step 5 – Calculate score to check whether the person is drowsy.

7.2 IMPLEMENTATION

- In our program we used Dlib, a pre-trained program on the dataset to detect human faces using the predefined 68 landmarks.
- After passing our video feed to the Dlib frame by frame, we can detect left eye and right eye features of the face.
- Now, we drew contours around it using OpenCV.
- Using Scipy's Euclidean function, we calculated the sum of both eye aspect ratios, which is the sum of two distinct vertical distances between the eyelids divided by its horizontal distance.
- Now we check if the aspect ratio value is less than 0.25 (0.25 was chosen as base case after some tests). If it is less, an alarm is sounded, and the user is warned.

7.3 METHODOLOGY

7.3.1. Image Processing

Image processing is a technique to carry out a particular set of actions on an image for obtaining an enhanced image or extracting some valuable information from it. The input is an image, and output may be an improved image or characteristics/features associated with the same.

Image processing algorithms commonly used for complete image capture can be categorized into:

- Low-level techniques, such as color enhancement and noise removal
- Medium-level techniques, such as compression and binarization
- Higher-level techniques involving segmentation, detection, and recognition algorithms extract semantic information from the captured data.

7.3.1.1. Image Processing Algorithms

Contrast Enhancement algorithm: Colour enhancement algorithm is further subdivided into

Histogram equalization algorithm: Using the histogram to improve image contrast.

Adaptive histogram equalization algorithm: It is the histogram equalization which adapts to local changes in contrast.

Connected-component labeling algorithm: It is about finding and labeling disjoint regions.

Feature detection algorithm: Feature detection consists of

Marr–Hildreth algorithm: It is an early edge detection algorithm

Canny edge detector algorithm: Canny edge detector is used for detecting a wide range of edges in images.

7.3.2. Frontal Face Detector

Face recognition is one of the most sought-after technologies in the field of machine learning. In recent times, the use cases for this technology have broadened from specific surveillance applications in government security systems to wider applications across multiple industries in such tasks as user identification and authentication, consumer experience, health, and advertising.

The dlib library is arguably one of the most utilized packages for face recognition. A Python package appropriately named `face_recognition` wraps dlib's face recognition functions into a simple, easy to use API. dlib includes two face detection methods built into the library:

1. A **HOG + Linear SVM face detector** that is accurate and computationally efficient.
2. A **Max-Margin (MMOD) CNN face detector** that is both highly accurate and very robust, capable of detecting faces from varying viewing angles, lighting conditions, and occlusion.

The dlib library provides two functions that can be used for face detection:

1. **HOG + Linear SVM:** `dlib.get_frontal_face_detector()`
2. **MMOD CNN:** `dlib.cnn_face_detection_model_v1(modelPath)`

7.3.2.1. Support Virtual Machine (SVM)

Support vector machines (SVMs) are supervised machine learning models that divide and classify data. SVMs are widely used for applications such as face detection, classification of images, handwriting recognition, etc. An SVM model can be considered as a point space wherein multiple classes are isolated using hyperplanes.

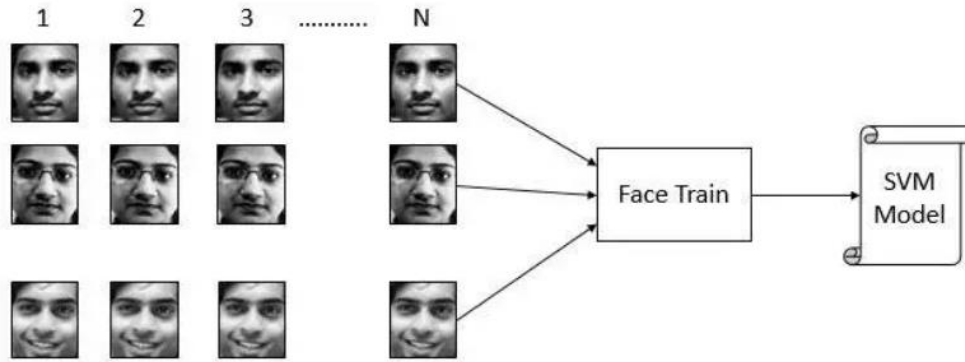


Figure 9: Block Diagram of Face Training

7.3.2.2. Histogram of Oriented Gradients (HOG)

A HOG is a feature descriptor generally used for object detection. HOGs are widely known for their use in pedestrian detection. A HOG relies on the property of objects within an image to possess the distribution of intensity gradients or edge directions. Gradients are calculated within an image per block. A block is considered as a pixel grid in which gradients are constituted from the magnitude and direction of change in the intensities of the pixel within the block.

The face sample images of a person are fed to the feature descriptor extraction algorithm, i.e., a HOG. The descriptors are gradient vectors generated per pixel of the image. The gradient for each pixel consists of magnitude and direction, calculated using the following formulae:

$$g = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \arctan \frac{g_y}{g_x}$$

G_x and G_y are respectively the horizontal and vertical components of the change in the pixel intensity. A window size of 128 x 144 is used for face images since it matches the general aspect ratio of human faces. The descriptors are calculated over blocks of pixels with 8 x 8 dimensions. These descriptor values for each pixel over 8 x 8 block are quantized into 9 bins, where each bin represents a directional angle of gradient and value in that bin, which is the summation of the magnitudes of all pixels with the same angle. Further, the histogram is then normalized over a 16 x 16 block size, which means four blocks of 8 x 8 are normalized together to minimize light conditions. This mechanism mitigates the accuracy drop due to a change in light. The SVM model is trained using several HOG vectors for multiple faces.

7.3.2.3. Face Recognition

The recognition of a face in a video sequence is split into three primary tasks: Face Detection, Face Prediction, and Face Tracking. The tasks performed in the Face Capture program are performed during face recognition as well. To recognize the face obtained, a vector of HOG features of the face is extracted. This vector is then used in the SVM model to determine a matching score for the input vector with each of the labels. The SVM returns the label with the maximum score, which represents the confidence to the closest match within the trained face data.

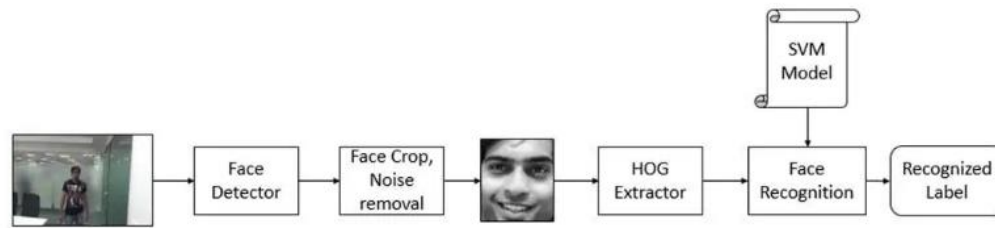


Figure 10: Block Diagram of Face Recognition Process

Human face recognition plays a very important role as part of modern surveillance and security applications. With the help of HOG and SVM models, one can achieve high-performance levels in recognizing human faces and analysing facial features, even in scenes containing complex backgrounds. The HOG + Linear SVM face detector takes ≈ 0.1 seconds, implying that ≈ 10 frames can be processed in a video stream scenario.

7.3.3. Shape Predictor

Shape predictors, also called landmark predictors, are used to predict key (x, y) -coordinates of a given “shape”. It is a tool that takes in an image region containing some object and outputs a set of point locations that define the pose of the object. The classic example of this is human face pose prediction, where you take an image of a human face as input and are expected to identify the locations of important facial landmarks such as the corners of the mouth and eyes, tip of the nose, and so forth.

7.3.3.1. Working of shape predictor

To estimate the landmark locations, the algorithm:

- Examines a sparse set of input pixel intensities (i.e., the “features” to the input model)
- Passes the features into an Ensemble of Regression Trees (ERT)
- Refines the predicted locations to improve accuracy through a cascade of regressors

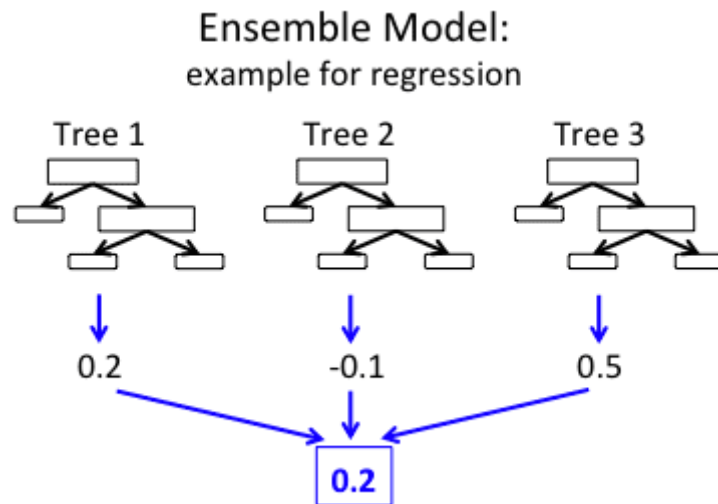


Figure 11: Working of Landmark Detector

7.3.4. Video Capture

Python provides various libraries for image and video processing. One of them is OpenCV. OpenCV is a vast library that helps in providing various functions for image and video operations. With OpenCV, we can capture a video from the camera. It lets you create a video capture object which is helpful to capture videos through webcam and then you may perform desired operations on that video.

Steps to capture a video

- Use `cv2.VideoCapture()` to get a video capture object for the camera
- Set up an infinite while loop and use the read method to read the frames using the above created object
- Use `cv2.imshow()` method to show the frames in the video
- Breaks the loop when the user clicks a specific key

`cv2.VideoCapture` object is instantiated by passing in the path to input video file. Then a loop is started, calling the `.read` method of `cv2.VideoCapture` to poll the next frame from the video file so that it can be processed in pipeline.

7.3.5. Color Image to Grayscale

The human eye is most sensitive to green light, then red, then blue. The grayscale value is calculated as the weighted average of the three channel values. `cv2.cvtColor()` method is used to convert an image from one color space to another.

The main reason why grayscale representations are often used for extracting descriptors instead of operating on color images directly is that grayscale simplifies the algorithm and reduces computational requirements.

While converting the color image to grayscale, each channel is weight differently to account

for how much each color is perceived:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

The grayscale representation of an image is often used when we have no use for color (such as in detecting faces or building object classifiers where the color of the object does not matter). Discarding color thus allows us to save memory and be more computationally efficient.

7.3.6. Convex Hull

A convex hull is the smallest convex polygon containing all the given points. Input is an array of points specified by their x and y coordinates. The output is the convex hull of this set of points.

A few of the applications of the convex hull are:

- **Collision avoidance:** If the convex hull of a car avoids collision with obstacles, then so does the car. Since the computation of paths that avoid collision is much easier with a convex car, then it is often used to plan paths.



Figure 12: Application of Convex Hull

- **Shape analysis:** Shapes may be classified for the purposes of matching by their "convex deficiency trees", structures that depend for their computation on convex hulls.

7.3.6.1. Algorithm for Convex Hull

Given the set of points for which we must find the convex hull. Suppose we know the convex hull of the left half points and the right half points, then the problem now is to merge these two

convex hulls and determine the convex hull for the complete set. This can be done by finding the upper and lower tangent to the right and left convex hulls.

1. Finding the contours, and

2. Applying the contour operations to obtain desired results

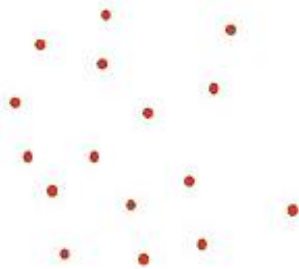


Figure 13: Set of Points

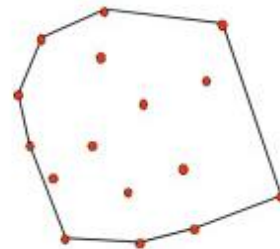


Figure 14: Convex Hull

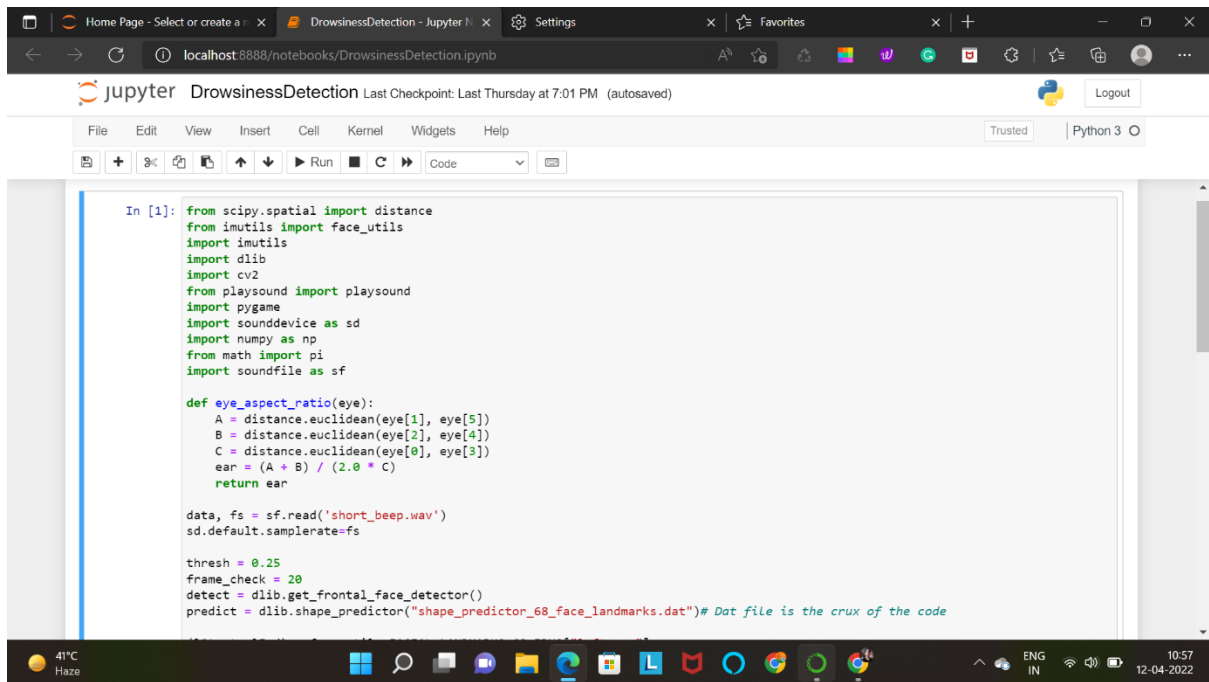
Step 1: Read the Input Image

Step 2: Binarize the input image. We perform binarization in three steps:

- Convert to grayscale
- Blur to remove noise
- Threshold to binarize image

Step 3: Use findContour to find contours

Step 4: Find the Convex Hull using convexHull. Since we have found the contours, we can now find the Convex Hull for each of the contours. This can be done using convexHull function.



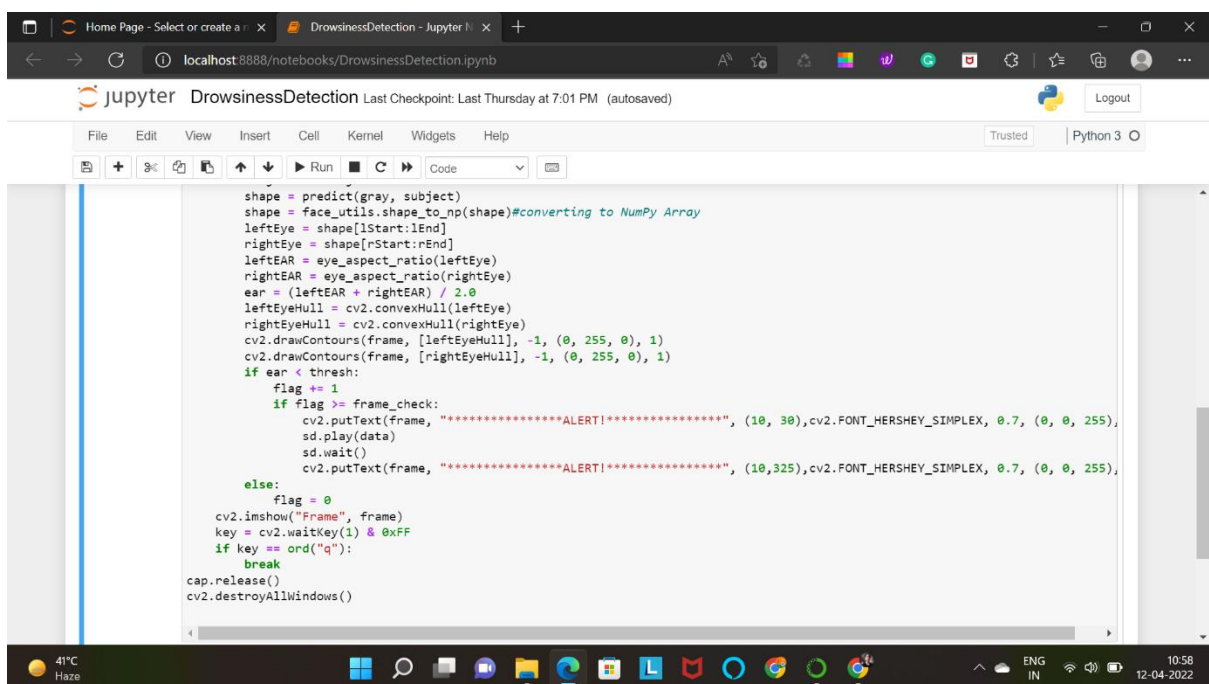
```
In [1]: from scipy.spatial import distance
from imutils import face_utils
import imutils
import dlib
import cv2
from playsound import playsound
import pygame
import sounddevice as sd
import numpy as np
from math import pi
import soundfile as sf

def eye_aspect_ratio(eye):
    A = distance.euclidean(eye[1], eye[5])
    B = distance.euclidean(eye[2], eye[4])
    C = distance.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

data, fs = sf.read('short_beep.wav')
sd.default.samplerate=fs

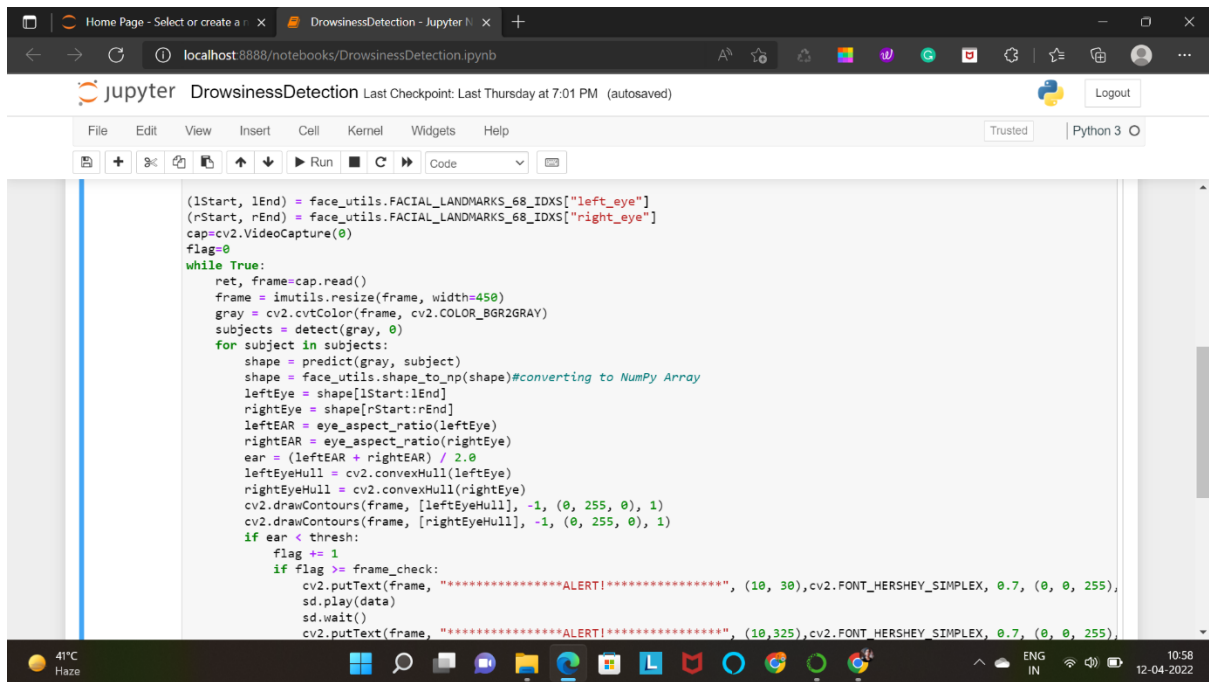
thresh = 0.25
frame_check = 20
detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")# Dat file is the crux of the code
```

Figure 15: Code



```
shape = predict(gray, subject)
shape = face_utils.shape_to_np(shape)#converting to NumPy Array
leftEye = shape[1Start:1End]
rightEye = shape[rStart:rEnd]
leftEAR = eye_aspect_ratio(leftEye)
rightEAR = eye_aspect_ratio(rightEye)
ear = (leftEAR + rightEAR) / 2.0
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 ear < thresh:
    flag += 1
    if flag >= frame_check:
        cv2.putText(frame, "*****ALERT!*****", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
        sd.play(data)
        sd.wait()
        cv2.putText(frame, "*****ALERT!*****", (10, 325), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
    else:
        flag = 0
cv2.imshow("Frame", frame)
key = cv2.waitKey(1) & 0xFF
if key == ord("q"):
    break
cap.release()
cv2.destroyAllWindows()
```

Figure 16: Code



```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS["right_eye"]
cap=cv2.VideoCapture(0)
flag=0
while True:
    ret, frame=cap.read()
    frame = imutils.resize(frame, width=450)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    subjects = detect(gray, 0)
    for subject in subjects:
        shape = predict(gray, subject)
        shape = face_utils.shape_to_np(shape)#converting to NumPy Array
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        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 ear < thresh:
        flag += 1
        if flag >= frame_check:
            cv2.putText(frame, "*****ALERT!*****", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
            sd.play(data)
            sd.wait()
            cv2.putText(frame, "*****ALERT!*****", (10,325),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),
```

Figure 17: Code

Chapter 8

RESULT DISCUSSION AND PERFORMANCE ANALYSIS

This section provides a successful experimental consequence in which the drowsiness is calculated through the observation of EAR of the driver.

8.1 PREDICTION OF DROWSINESS

When the EAR value is greater than 0.25, it indicates the eyes are open. This test shows that the driver has not fallen into drowsiness that is graphically represented in figure 7 and the detected face is not recognized as a drowsy one. Similarly, the drowsiness of the driver is detected due to EAR value being less than 0.25 as graphically plotted in figure 7 and finally, a drowsy face is detected. The EAR values are frequently changed due to movements of the eyelids and when the drowsiness is detected, then the driver is alerted with repeated voice sound.

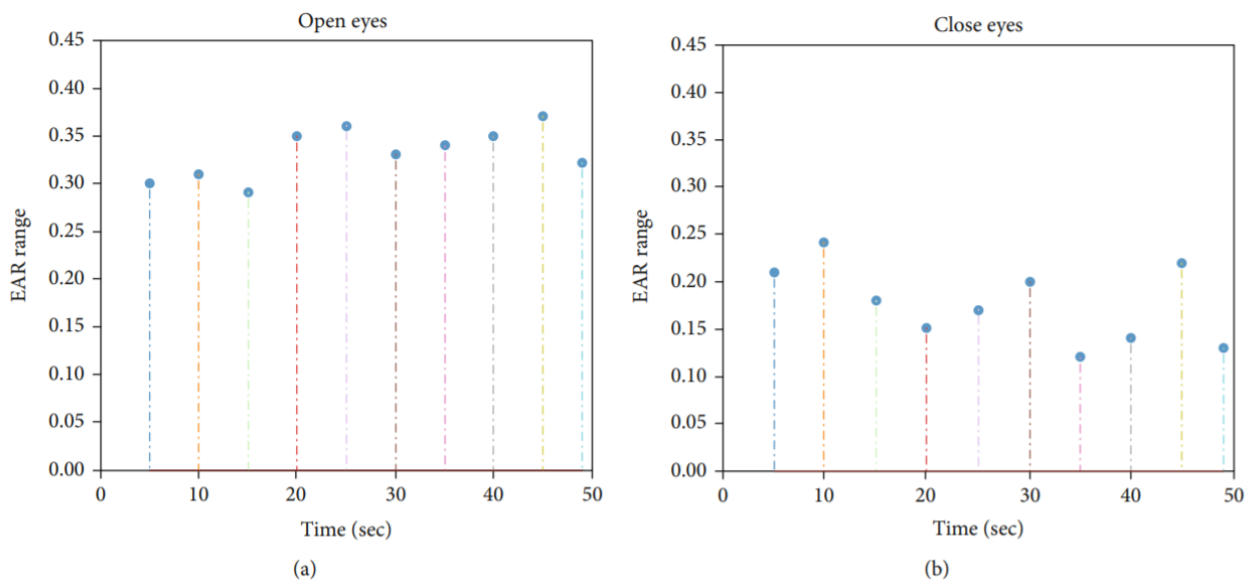


Figure 18: EAR Mapped: (a) when eyes are opened; (b) when eyes are closed

8.2 PERFORMANCE ANALYSIS

This system experimented on various conditions like using glasses, without glasses as an alternative view. So, the performance is measured on different levels such as eye accuracy and drowsiness accuracy detection. So, the blinking eyes and drowsiness were detected on several frames which are used to observe accuracy from video streams. The overall analysis was performed on frames with TED (Truly Eye Detect), FED (Falsely Eye Detect), TAD (Truly Alarm Detect), FAD (Falsely Alarm Detect), TMD (Truly Mail Detect), and FMD (Falsely Mail Detect).

The performance of eye accuracy and drowsiness can be accurately calculated and are measured by a given set of frames on various situations. Consider Eye Accuracy Detection as EAD and Drowsy Accuracy Detection as DAD.

$$EAD = \frac{TED}{(TED + FED)},$$

$$DAD = \frac{(TAD + TMD)}{(TAD + FAD + TMD + FMD)}.$$

From the overall analysis of the tested result, it can be concluded that using face landmarks does not change under any conditions and its performance is more accurate than the cascaded methods. But this process takes a little bit more time to load at deep night vision than the cascade technique. Thus, this proposed system has provided an efficient and successful drowsy detection using the facial landmark method.

Chapter 9

SYSTEM TESTING

9.1 TEST CASES AND TEST RESULTS

TEST ID	TEST CONDITION	SYSTEM BEHAVIOUR	EXPECTED RESULT
T01	Straight face, good light	Not drowsy	Not drowsy
T02	Closed eyes, good light	Drowsy	Drowsy

Note: Testing is performed manually

Table 1: Test Cases



Figure 19: Demonstration

Chapter 10

CONCLUSION AND FUTURE IMPLEMENTATION

10.1 CONCLUSION

This research provides a robust method for detecting drowsiness of drivers. This method generally combines two different systems in one integrated system. But the existing techniques are based on psychological or vehicle-based approaches to detect drowsiness of drivers, but such techniques are highly intruding as well as fully turns on the physical environment. So, the proposed system is used to construct a non-intruding technique for measuring drowsiness of the driver. The main component of the system is the camera module that is used for persistent recording of face landmarks that are localized through facial landmark points then to calculate EAR. However, if the calculated EAR value increases from the threshold range, then the eyes are kept open and no change in the state of the system occurs. Similarly, if the EAR value falls from the threshold range, then an alert is generated.

10.2 FUTURE IMPLEMENTATION

The future work may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc, for fatigue measurement. Driver drowsiness poses a major threat to highway safety, and the problem is particularly severe for commercial motor vehicle operators. Twenty-four-hour operations, high annual mileage, exposure to challenging environmental conditions, and demanding work schedules all contribute to this serious safety issue. Monitoring the driver's state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem. Currently there is no adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized.

After necessary field trials and validation, the Government, transporters, taxis, and cabs can adopt it. The Auto industry, especially the car segment needs this as a standard feat

REFERENCES

- [1] “Fundamentals of Computer Vision”, Wesley E. Snyder (North Carolina State University), Hairong Qi (University of Tennessee), Cambridge University Press.
- [2] “Review of Classification Algorithms in Image Processing”, Manjula Kamalahasan, K.Vijayarekha , P.Vimaladevi (SRC, SASTRA University).
- [3] “Digital Image Processing”, Rafael C. Gonzalez and Richard E. Woods.
- [4] “Structured Prediction and Learning in Computer Vision”, Sebastian Nowozin and Christoph H. Lampert 2011
- [5] L. B. Chen, W. J. Chang, J. P. Su et al., “A wearable-glasses based drowsiness-fatigue-detection system for improving road safety,” in 2016 IEEE 5th Global Conference on Consumer Electronics, p. 12, IEEE, 2016.
- [6] S. W. Jang and B. Ahn, “Implementation of detection system for drowsy driving prevention using image recognition,” Sustainability, vol. 12, no. 7, p. 3037, 2020.
- [7] Cheng B., Zhang W., Lin Y., Feng R., Zhang X., “Driver drowsiness detection based on multisource information”, Hum. Factors Ergon. Manuf. Serv. Indust.
- [8] Vural E. Sabanci University; Istanbul, Turkey: 2009. “Video Based Detection of Driver Fatigue”.
- [9] “Drowsy driver detection through facial movement analysis” Lew M., Sebe N., Huang T.