Driver Drowsiness Detection System

A project presented to the **National University** in partial fulfillment of the requirements for the degree of B.Sc. (Hon's) in Computer Science & Engineering.

Submitted by:

Animash Bishwas

Registration No: 17502004951

Session: 2017-18

Supervised by:

Nusrhat Jahan Sarker

Lecturer

Department of Computer Science & Engineering

Daffodil Institute of IT (DIIT)

Department of Computer Science & Engineering

Daffodil Institute of IT (DIIT)

Under National University (NU)- Dhaka, Bangladesh

Submission Date: 04-09-2023

Approval

This Project titled "**Driver Drowsiness Detection System**" is submitted to the Department of Computer Science & Engineering (CSE) of Daffodil Institute of IT (DIIT) under National University (NU). It has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor's in Computer Science & Engineering (CSE) & approved as to its styles & contents.

…….
Examiner

…….
Examiner

……
Nusrhat Jahan Sarker
Project Guide (Supervisor)
Lecturer
Department of CSE
Daffodil Institute of IT

…….
Md. Imran Hossain
Head
Department of CSE
Daffodil Institute of IT

Declaration

I hereby declare that the project work entitled "**Driver Drowsiness Detection System**" submitted to the degree of B.Sc. (Hon's) in Computer Science & Engineering (CSE) is a record of original work done by me. Except as acknowledged in the text and that the material has not been submitted, either in whole or in part, for a degree at this or any other university.

Submitted by:

……

Animash Bishwas

Registration No: 17502004951

Session: 2017-18

Acknowledgment

My sincere thanks to **Prof. Dr. Mohammed Shakhawat Hossain, Principal, DIIT** who has allowed me to do this project, and the encouragement given to me.

I express my gratitude to **Md. Imran Hossain**, Head of Department, Department of Computer Science & Engineering, DIIT, Dhaka, for his patronage and for giving me an opportunity to undertake this Project.

I would also like to thank my Project Supervisor, **Nusrhat Jahan Sarker**, Lecturer, Department of Computer Science & Engineering, DIIT, for his valuable guidance and support to meet the successful completion of my project.

I express my gratitude to **Poly Bhoumik**, Senior Lecturer, DIIT, Dhaka, for having provided us with the facilities to do the project successfully.

I also express my gratitude to **Saidur Rahman**, Senior Lecturer, DIIT, Dhaka, for having provided us with the facilities to do the project successfully.

I also express my gratitude to **Safrun Nesa Saira**, Senior Lecturer, DIIT, Dhaka, for having provided us with the facilities to do the project successfully.

I also express my gratitude to **Mizanur Rahman**, Lecturer, DIIT, Dhaka, for having provided us with the facilities to do the project successfully.

I also express my gratitude to **Moumita Akter**, Lecturer, DIIT, Dhaka, for having provided us with the facilities to do the project successfully.

I also say thanks to **Md. Mushfiqur Rahaman**, Lecturer, DIIT, Dhaka, for giving his valuable time to do the project successfully.

Finally, I extend my sincere thanks to my family members and my friends for their constant support throughout this project.

Abstract

Drowsiness detection system created to reduce the risk of accident while driving. The system will records image of driver then face and eyes will be detected. Results of eyes detection, each frame value will be analyzed if eyes are closed for 2 seconds. If eyes close for 2 seconds then system will decide that driver is sleepy and alarm will sound. From the experiment, average result for detection is 954ms, best position of camera is above the driver on the dashboard and for bright condition.

Table of contents

Approval **I**
Declaration **II**
Acknowledgement **III**
Abstract **IV**
Table of contents **V-VII**

Contents

CHAPTER- 01
INTRODUCTION **1-3**
1.1 Introduction 2
1.2 Motivation 2
1.3 Objective 2
1.4 Fact & Statistics 3
1.5 Business prospect of this project 3

CHAPTER-02
Proposed System **4-8**
2.1 Background of this Project 5
2.2 Limitation of the Existing system 6
2.3 Proposed System 7
2.4 Features of our System 8

CHAPTER- 03
Requirments and System Analysis **9-15**
3.1 Inroduction to Requirements 10
3.1.1 Hardware Requirements 10
3.1.2 Software Requirements 10
3.2 Python 11
3.3 OpenCV 11
3.4 Dlib 12
3.5 Imutils 12
3.6 Scipy 13
3.7 Pygame 13
3.8 Datetime 13
3.9 Pandas 13
3.10 Numpy 14
3.11 Matplotlib.pyplot 14
3.12 Seaborn 14

3.13 Matplotlib.dates 14
3.14 Sqlite3 15
3.15 DB Browser 15

CHAPTER- 04
System Design **16-27**
4.1 Proposed System Overview 17
4.2 Workflow Diagram 19
4.3 Flow Chart 20
4.4 Data Flow Diagram 21
4.4.1 DFD Level-0 21
4.4.2 DFD Level-1 22
4.4.3 DFD Level-2 23
4.5 Use CASE Diagram 24
4.6 Activity Diagram 26

CHAPTER-05
Implimentation **28-31**
5.1 Implimentation 29
5.1.1 Face Detect 29
5.1.2 Active Status 29
5.1.3 Drowsy Status 30
5.1.4 Sleeping Status 30
5.1.5 Database 31
5.1.6 Data Plot 32

CHAPTER-06
Conclusion **32-33**
6.1 Limitation of our system 33
6.2 Future Enhancement 33
6.3 Conclusion 33

References **34**

Appendix **35-37**

List of Figure

3.4.1 68 Face landmarks	13
4.1.1 EAR Formula	17
4.1.2 Drowsiness Detect	18
4.2.1 Workflow Diagram	19
4.3.1 Flow chart Diagram	21
4.4.1.1 Data Flow Diagram level-0	22
4.4.2.1 Data Flow Diagram level-1	23
4.4.3.1 Data Flow Diagram level-2	24
4.5.1 Use CASE Diagram	26
4.6.1 Activity Diagram	27

CHAPTER-01

INTRODUCTION

1.1 Introduction

Driver drowsiness detection is a car safety technology which helps prevent accidents caused by the driver getting drowsy. Various studies have suggested that around 20% of all road accidents are fatigue-related, up to 50% on certain roads. Some of the current systems learn driver patterns and can detect when a driver is becoming drowsy. Various technologies can be used to try to detect driver drowsiness. Primarily uses steering input from electric power steering system. Monitoring a driver this way only works as long as a driver actually steers a vehicle actively instead of using an automatic lane-keeping system. Uses a lane monitoring camera. Monitoring a driver this way only works as long as a driver actually steers a vehicle actively instead of using an automatic lane-keeping system. Uses computer vision to observe the driver's face, either using a built-in camera or on mobile devices. Requires body sensors to measure parameters like brain activity, heart rate, skin conductance, muscle activity, head movements etc.

1.2 Motivation

The purpose of this project is to develop the simulation of drowsiness detection system. The focus of the project is to design a system that will detect the drowsiness by detecting the closed eyes of the driver. By monitoring the state of the eyes, it is believed can detect the early symptom of the driver’s drowsiness, to avoid car accidents. The process of detecting the drowsiness between drivers is to detect the open and closed of the eyes.

1.3 Objective

The main objective of a driver drowsiness detection system using Python and OpenCV is to monitor a driver's level of drowsiness and alert them when they are at risk of falling asleep or losing concentration while driving. This system aims to enhance road safety by detecting signs of drowsiness in real-time and providing timely warnings to prevent accidents caused by driver fatigue.

Here are the key steps involved in such a system:

Face Detection: The system uses OpenCV's computer vision capabilities to detect and locate the driver's face in the video stream or images captured from a camera.

Eye Tracking: It uses techniques such as eye aspect ratio (EAR) or eyelid detection to determine the state of the eyes, such as whether they are open or closed.

Drowsiness Detection: By monitoring the driver's eye state and tracking patterns over time, the system can determine if the driver is becoming drowsy.

Alert Mechanism: When the system detects drowsiness, it triggers an alert to warn the driver.

1.5 Fact & Statistics

The Road Safety Foundation said 7,713 people were killed and 12,615 injured in 6,829 road crashes last year. Last year's figures were the highest since 2019, it added. The number of road crashes and deaths in 2021 were 6,284 and 5,371. Of these, at least 20% 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 Bangladesh 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.

1.5 Business Prospect of this Project

The global driver drowsiness detection system market is expected to grow at a CAGR of 9.5% from 2022 to 2030. The growth in the market can be attributed to the increasing demand for passenger cars and commercial vehicles, and the rising awareness about road safety. The hardware devices segment is expected to dominate the market during the forecast period, owing to its low cost and easy installation.

CHAPTER- 02

Proposed System

2.1 Background of this Project

Driver drowsiness is a significant cause of road accidents and fatalities worldwide. Drowsy driving impairs a driver's ability to react quickly and make sound decisions, leading to an increased risk of accidents. To address this issue, researchers and engineers have developed driver drowsiness detection systems using computer vision and machine learning techniques. These systems aim to detect signs of drowsiness in real-time and provide alerts to prevent accidents caused by driver fatigue.

1. **Computer Vision and OpenCV:** Computer vision is a field of artificial intelligence that focuses on enabling machines to interpret and understand visual information from the world. OpenCV, an open-source computer vision library, provides a vast array of functions and tools for image and video processing, object detection, and facial recognition. It is widely used in various applications, including driver drowsiness detection, due to its efficiency and ease of integration.

2. **Eye Aspect Ratio (EAR):** The Eye Aspect Ratio (EAR) is a significant feature used in drowsiness detection systems. It is calculated based on the distances between specific eye landmarks, and it measures the ratio of the width to height of the eye. When a person's eyes are closed or partially closed due to drowsiness, the EAR decreases. By monitoring the EAR, the system can detect signs of eye closure, indicating potential drowsiness.

3. **Dlib Library:** The dlib library is a popular machine learning library that contains powerful tools for facial landmark detection and facial feature tracking. It provides pre-trained models for facial landmark detection, which are used to locate and track key facial features, including eyes and mouth, in real-time video streams.

4. **Database Integration**: Driver drowsiness detection systems often use databases to store data, such as EAR values and timestamps. These databases enable data logging for analysis, allowing researchers to study drowsiness patterns, driver behavior, and system performance over time.

5. **Alert Mechanism:** The alert mechanism in driver drowsiness detection systems aims to warn the driver when signs of drowsiness are detected. Common alert methods include audio signals, such as playing alert sounds or music, and visual indicators displayed on the dashboard or video frame.

2.2 Limitation of the Existing system

While driver drowsiness detection systems using Python and OpenCV can be effective, there are certain limitations to consider. Some of the common limitations of existing systems include:

- **Sensitivity to Lighting Conditions:** The system's accuracy might be influenced by changing lighting conditions, such as low light or glare, affecting the reliability of eye detection.

- **Limited to Face Orientation:** The system could struggle with detecting drowsiness accurately if the driver's face orientation changes significantly.

- **False Positives/Negatives:** The system might generate false alerts due to factors like sudden head movements, talking, or using the phone, impacting the user's trust in the system.

- **Dependency on Camera Quality:** The quality of the camera used can impact the system's ability to detect small eye movements, potentially leading to inaccurate results.

- **Hardware Requirements:** The system might demand a certain level of hardware capabilities, such as processing power, memory, and camera quality, which could limit its usability on some devices.

- **Real-Time Processing:** Depending on the efficiency of the algorithms used, there could be a slight delay in processing the frames, affecting real-time responsiveness.

2.2 Proposed System: A large number of the mishaps happen because of tiredness of drivers. It is one of the basic reasons for streets mishaps now-a-days. Most recent insights say that a considerable lot of the mishaps were caused on account of languor of drivers. Vehicle mishaps because of tiredness in drivers are causing demise to large number of lives. Over 30the anticipation of this, a framework is required which identifies the languor and alarms the driver which saves the life. In

this task, we present a plan for driver languor location. In this, the driver is ceaselessly observed through webcam. This model uses picture handling procedures which mostly centers on face and eyes of the driver. The model concentrates the drivers face and predicts the flickering of eye from eye area. We utilize a calculation to follow and investigate drivers face and eyes to gauge. If the flickering rate is high then the framework cautions the driver with a sound.

2.4 Features of our System

- Real Time Face Detection
- Open Eyes Detection
- Close Eyes Detection
- Drowsiness Alert
- Database
- Data Plot

CHAPTER- 03

System Requirement

3.1 Introduction to Requirements

The software requirements are description of features and functionalities of the target System. Requirements convey the expectations of users from the software product. The requirements can be obvious or hidden, known or unknown, expected or unexpected from client's point of view. The process to gather the software requirements from client, analyze and document them is known as requirement engineering. The goal of requirement engineering is to develop and maintain sophisticated and descriptive 'System Requirements Specification' document.

3.1.1 Hardware Requirements

1. **Laptop**
2. **Webcam**
3. **Processor:** An Intel Core i5 or i7 processor or equivalent would be suitable.
4. **Memory (RAM):** A minimum of 8GB of RAM is recommended, but 16GB or more may be required for more demanding applications.
5. **Storage:** An SSD with at least 256GB of storage capacity is recommended for fast data access and processing.
6. **Audio Output Devices:** Speakers or headphones are required for playing back drowsiness alert to end-users.
7. **Graphics Processing Unit (GPU):** A dedicated GPU with at least 4GB.
8. **Network Interface Card (NIC)**

3.1.2 Software Requirements

1. **Operating System:** Windows, Linux, or macOS.
2. **Programming Language:** Python, Java, C++.
3. **Libraries:** Dlib, Pygame, imutls, scipy, Pandas, Numpy, Matplotlib. pyplot, Seabon.
4. **Machine Learning Frameworks:** TensorFlow, Keras, and Scikit-Learn.
5. **Database Management System:** MySQL or SQLlite3.
6. **Integrated Development Environment (IDE):** IDEL, PyCharm, Visual Studio Code, or Eclipse, Jupyter Notebook, or Google Colab.
7. **Web Development Tools:** HTML, CSS, and Python.
8. **Web Application Frameworks:** Flask, Django, Ruby on Rails, and Node.js.

9. **Web Browser:** Google Chrome, Firefox, Safari, and other HTML5.

3.2 Python

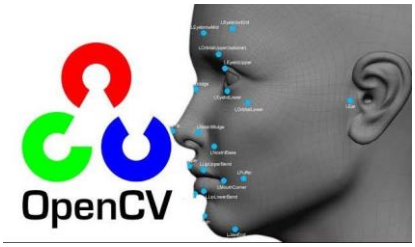
Python is a high-level programming language that is commonly used in driver drowsiness detection system projects. It is a versatile language that offers a wide range of libraries and frameworks for machine learning, and data analysis. Some of the key advantages of using Python for driver drowsiness projects include:

- **Easy to Learn:** Python is known for its simplicity and ease of use. Its syntax is easy to read and write, making it an ideal choice for beginners.
- **Rich Library Ecosystem:** Python offers a rich library ecosystem for machine learning, and data analysis, such as face Recognition, PyAudio, Praat, TensorFlow, Keras, and Scikit-Learn.
- **Large Community:** Python has a large and active community of developers who contribute to the development of libraries, frameworks, and tools. This makes it easier to find support and resources when building a driver drowsiness detection system.
- **Cross-Platform Compatibility:** Python code can run on different operating systems such as Windows, Linux, and macOS, making it a versatile choice for driver drowsiness detection projects.
- **Rapid Prototyping:** Python allows for rapid prototyping and experimentation, which is important for developing and testing different approaches to driver drowsiness detection system.

Overall, Python is a popular and effective choice for driver drowsiness detection system projects due to its ease of use, rich library ecosystem, and strong community support.

3.3 OpenCV

OpenCV is a nonprofit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.



3.4 Dlib

Dlib is one of the most powerful and easy-to-go open-source library consisting of machine learning library/algorithms and various tools for creating software. It was initially released in 2002. It has been used widely in many big industries, companies and for various big projects, etc. It also has many more types of algorithms that have a greater role in the real world.

Dlib is mostly used for face recognition purposes. They analyzed the object/face using the functions called HOG (Histogram of oriented gradients) and CNN (Convolutional Neural Networks). Face recognition nowadays are been used widely in many applications. In Our program dlib is used to find the frontal human face and estimate its pose using 68 face landmarks.

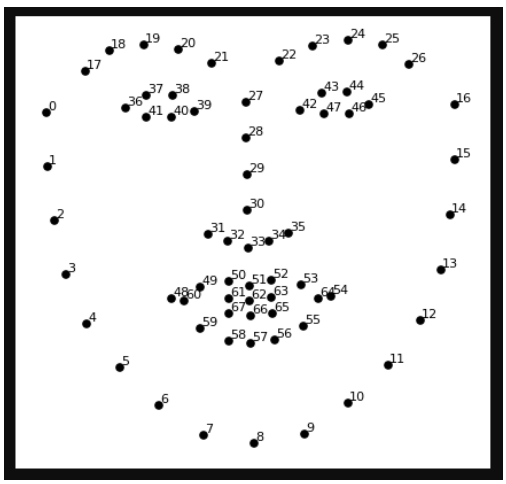


Fig 3.4.1: 68 face landmarks

3.5 Imutils

A series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

3.6 Scipy

SciPy is a scientific computation library that uses numpy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats and signal processing. Like NumPy, SciPy is open source so we can use it freely. SciPy was created by NumPy's creator Travis Oliphant.

3.7 Pygame

Python PyGame library is used to create video games. This library includes several modules for playing sound, drawing graphics, handling mouse inputs, etc. It is also used to create client-side applications that can be wrapped in standalone executables.

3.8 Datetime

The datetime module supplies classes for manipulating dates and times. Class datetime. datetime. A combination of a date and a time. Attributes: year, month, day, hour, minute, second.

3.9 Pandas

Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

3.10 Numpy

NumPy is a Python library for scientific computing that provides tools for working with arrays and matrices. It is a fundamental library for many scientific and data analysis tasks in Python, including Driver drowsiness detection system projects.

3.11 Matplotlib.pyplot

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

3.12 Seaborn

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

3.13 Matplotlib.date

Matplotlib provides sophisticated date plotting capabilities, standing on the shoulders of python datetime and the add-on module dateutil.

3.14 Sqlite3

SQLite is a C library that provides a lightweight disk-based database that doesn't require a separate server process and allows accessing the database using a nonstandard variant of the SQL query language. Some applications can use SQLite for internal data storage. It's also possible to prototype an application using SQLite and then port the code to a larger database such as PostgreSQL or Oracle.

3.15 DB Browser

DB Browser for SQLite is a high quality, visual, open source tool to create, design, and edit database files compatible with SQLite.

CHAPTER- 04
SYSTEM DESIGN

4.1 Proposed System Overview

The Driver Drowsiness Detection System is a software-based solution designed to enhance road safety by monitoring a driver's level of drowsiness and providing timely alerts to prevent accidents caused by fatigue-related impairments. The system utilizes Python and OpenCV (Open Source Computer Vision Library) to process video streams from a camera mounted inside the vehicle, enabling real-time monitoring of the driver's facial features and eye movements.

Functionalities:

Face Detection: The system uses OpenCV's face detection algorithms to locate the driver's face within the captured video frames.

Eye Tracking: After detecting the face, the system employs dlib, a machine learning library, to track the position of the driver's eyes within the facial region.

Eye Aspect Ratio (EAR) Calculation: By measuring the ratio of the distance between the eye landmarks, the system calculates the Eye Aspect Ratio (EAR) for each eye. The EAR is a measure of eye openness and is used to identify eye closure patterns.

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

Fig 4.1.1: EAR Formula

Drowsiness Detection: Based on the EAR values and predefined thresholds, the system determines whether the driver is drowsy. If the EAR drops below a certain threshold for a sustained period, it indicates drowsiness.

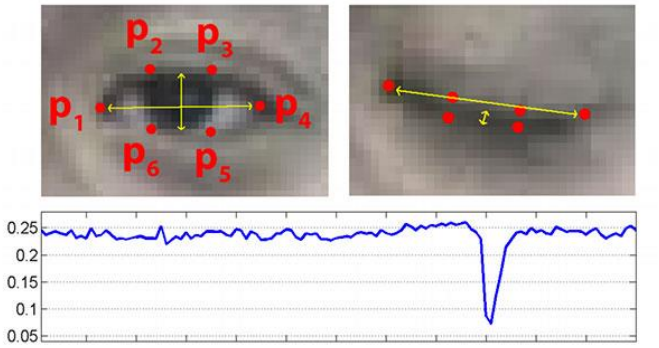


Fig 4.1.2: Detect Drowsiness

Alert Mechanism: When drowsiness is detected, the system triggers an alert to warn the driver and prevent potential accidents. This alert may consist of visual indicators displayed on the video frame and auditory signals, such as playing an alert sound.

Data Logging: The system records the EAR values along with timestamps in an SQLite database for further analysis and monitoring. This data can be useful for tracking drowsiness patterns and generating reports.

Proposed Workflow:

- The system initializes and starts capturing video frames from the camera.
- It processes each frame, detecting the driver's face and tracking the eyes.
- The EAR for each eye is calculated and analyzed to determine the driver's drowsiness level.
- If the drowsiness threshold is crossed, the system triggers an alert mechanism to warn the driver.
- EAR data, timestamps, and alerts are logged into an SQLite database for future analysis.
- The system continues to monitor the driver's drowsiness level in real-time.
- The driver can interact with the system through a user interface that allows starting and stopping the drowsiness detection process.

Benefits:

- **Enhanced Road Safety:** The system actively monitors driver drowsiness, helping prevent accidents caused by fatigue-related impairments.
- **Real-time Monitoring:** The system operates in real-time, providing timely alerts to keep the driver alert and awake.
- **Data Analysis:** The logged data enables post-analysis of drowsiness patterns, which can lead to insights and improvements in the system.
- **Easy Integration:** The system can be integrated with various vehicles, making it a potential safety feature in modern cars.

Limitations of existing system:

- The system may occasionally generate false alerts or miss drowsy states due to variations in lighting conditions, eye appearance, or other factors.
- Different individuals may have varying patterns of drowsiness, making it challenging to establish universal thresholds.
- The system might generate false alerts due to factors like sudden head movements, talking, or using the phone, impacting the user's trust in the system.

4.2 Workflow Diagram

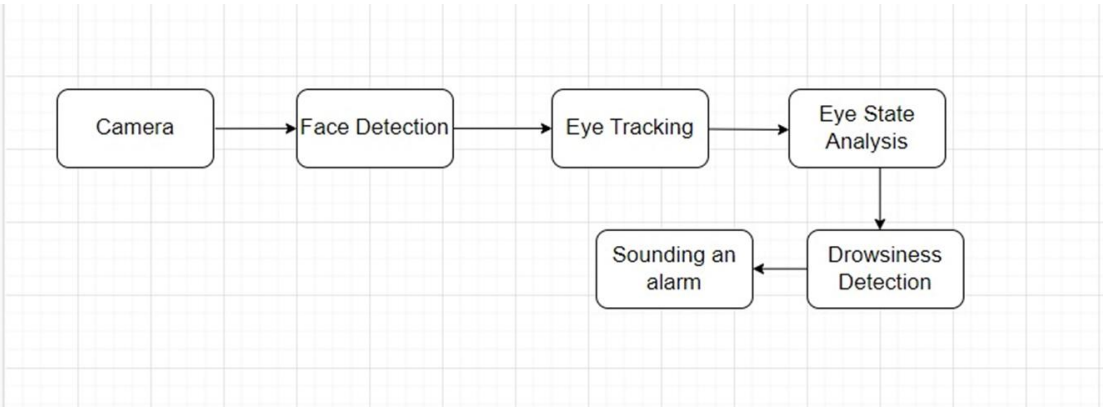


Fig 4.2.1: Workflow Diagram

- **Camera:** The system continuously captures frames from the video stream or images from the camera.
- **Face Detection:** Each captured frame is analyzed to detect and locate the driver's face using OpenCV's face detection algorithms.

- **Eye Tracking:** Once the face is detected, the system focuses on the driver's eyes within the face region.
- **Eye State Analysis:** The system analyzes the state of the driver's eyes, such as open, closed, or blinking, by applying eye-tracking algorithms or eye aspect ratio (EAR) calculations.
- Drowsiness Detection: Based on the eye state analysis and predefined thresholds, the system determines if the driver is drowsy or alert. This can involve comparing the eye state with predefined drowsiness indicators or using machine learning algorithms to classify drowsiness patterns.
- **Alert Trigger:** If the system detects drowsiness, it triggers an alert mechanism to warn the driver. This can include sounding an alarm, flashing lights, or other means of notification to grab the driver's attention and prevent accidents.

4.3 Flow Chart

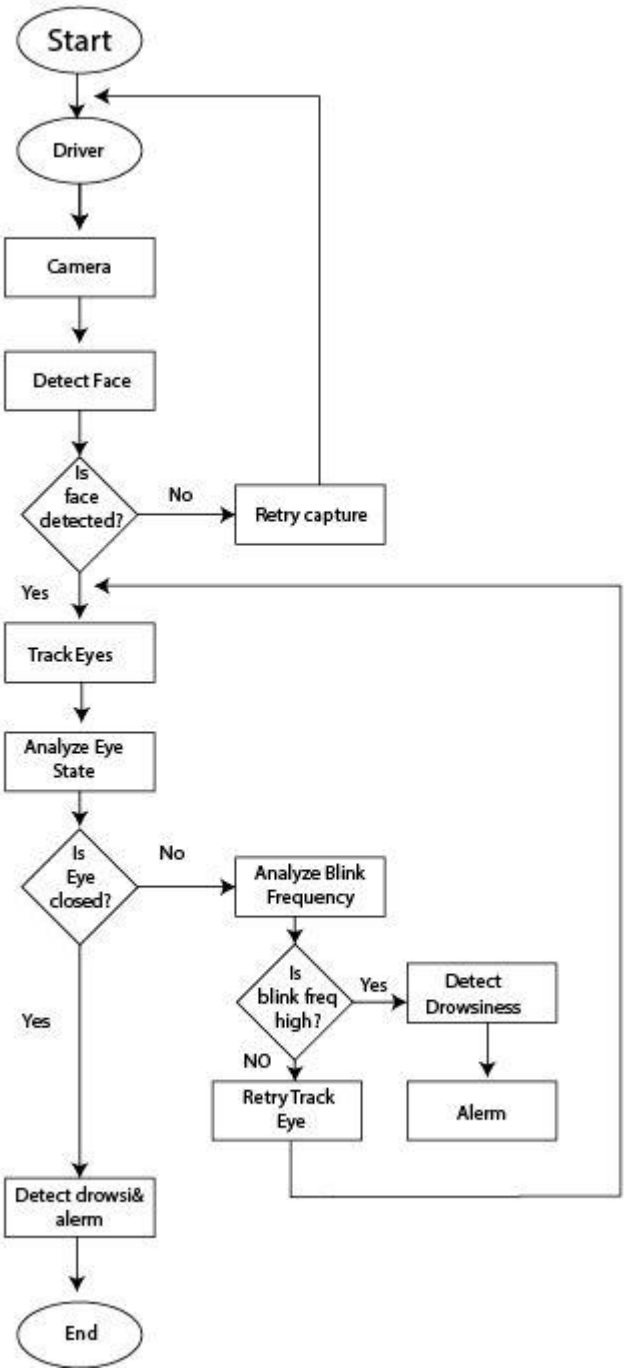


Fig 4.3.1: Flow chart Diagram

This flowchart represents the high-level steps involved in a driver drowsiness detection system using Python and OpenCV. It starts by capturing frames and detecting the face. If a face is detected, the system proceeds to track the eyes and analyze their state, determining if the eyes are closed. If the eyes are closed, the system analyzes the blink frequency to determine if it is high. If the blink frequency is high, an alert is triggered indicating that the driver is drowsy. Regardless of the result, the system continues

monitoring the driver. When the monitoring session ends, the system stops monitoring and the process ends.

4.4 Data Flow Diagram

Data Flow Diagrams (DFDs) are graphical representations of a system that show how data flows through different processes and entities. In the context of Driver Drowsiness Detection System, DFDs can be used to represent how the system processes and analyzes Driver Drowsiness Detection System.

- **DFD Level 0:** This diagram provides an overall view of the system, showing the input and output data flows and the main processes of the system.
- **DFD Level 1:** This diagram provides a more detailed view of the system by breaking down the main process of the system into subprocesses or entities.
- **DFD Level 2:** This diagram provides a further detailed view of the sub-processes from Level 1, by breaking them down into even more detailed subprocesses or entities

Overall, DFDs provide a structured way to represent the data flow and processes of a system, which is particularly useful for complex systems such as Driver Drowsiness Detection System. By breaking down the system into subprocesses and entities, DFDs help to understand how the system works at different levels of detail.

4.4.1 DFD Level-0

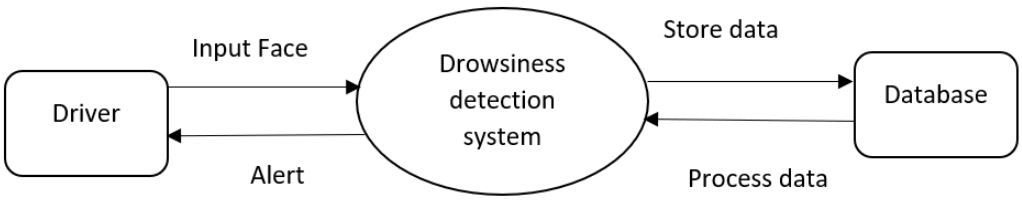


Fig 4.4.1.1: Data Flow Diagram level-0

In this level-0 DFD, the main components are:

1. **Input Data:** This component represents the data sources used by the system, which includes the video stream from the camera and the database for storing EAR data.
2. **Drowsiness Detection System**: This component handles the video frame processing tasks, including face detection and eye tracking using OpenCV and dlib and calculates the EAR, which is a measure used to detect eye openness.
3. **Detected Output**: This component determines whether the driver is drowsy based on the EAR value and predefined thresholds and responsible for triggering alerts, such as visual messages on the frame and audio alerts using the mixer module from Pygame.

4.4.2 DFD Level-1

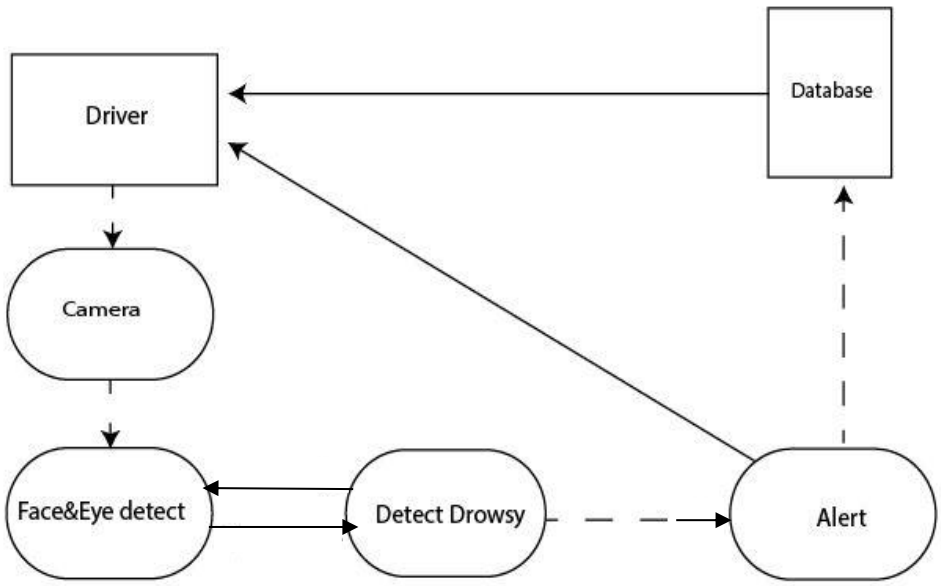


Fig 4.4.2.1: Data Flow Diagram level-1

In this level-1 DFD, the main components are:

1. **Input Data (Video Stream):** This component represents the input data sources used by the system, which include the video stream from the camera and the database (SQLite) for storing EAR data.
2. **Video Frame Processing (Face & Eye Detection)**: This component processes the video frames, performs face detection, and tracks the driver's eyes using OpenCV and dlib.
3. **EAR Data Management:** This component stores the EAR data along with timestamps in the database for further analysis and monitoring.
4. **Audio Alert (Audio Playback):** This component plays an audio alert (e.g., music) when drowsiness is detected to wake up the driver.
5. **Visual Alert:** This component displays a visual alert on the video frame when drowsiness is detected to grab the driver's attention.

4.4.3 DFD Level-2

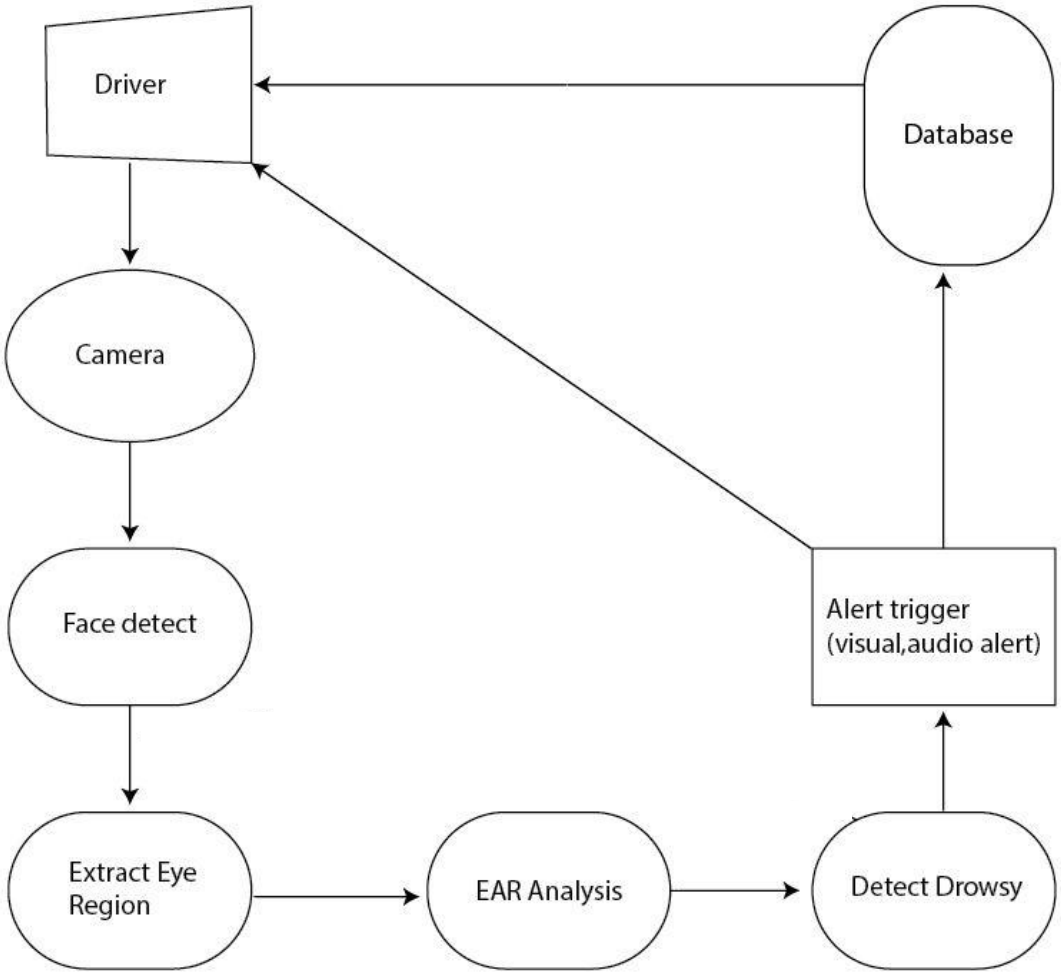


Fig4.4.3.1: Data Flow Diagram level-2

In this level-2 DFD, the main components are:

1. **Input Data (Video Stream, Database):** This component represents the input data sources used by the system, which include the video stream from the camera and the database (SQLite) for storing EAR data.
2. **Video Frame Processing (Face & Eye Detection, EAR Calculation):** This component processes the video frames, performs face detection, and tracks the driver's eyes using OpenCV and dlib. It also calculates the Eye Aspect Ratio (EAR) to determine eye openness.

3. **EAR Data Management (Store in Database)**: This component is responsible for storing the EAR data along with timestamps in the database for further analysis and monitoring.
4. **Drowsiness Detection (EAR Analysis, Alert Trigger):** This component analyzes the EAR data to detect drowsiness. If drowsiness is detected based on predefined thresholds, it triggers an alert to wake up the driver.
5. **Audio/Visual Alert Interface (Visual Alert, Audio Playback):** This component handles the interface for triggering audio and visual alerts when drowsiness is detected. It displays a visual alert on the video frame and plays audio (e.g., music) to wake up the driver.

4.5 Use Case Diagram

In the Unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) and their interactions with the system. To build one, you'll use a set of specialized symbols and connectors. An effective use case diagram can help your team discuss and represent:

- Scenarios in which your system or application interacts with people, organizations, or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

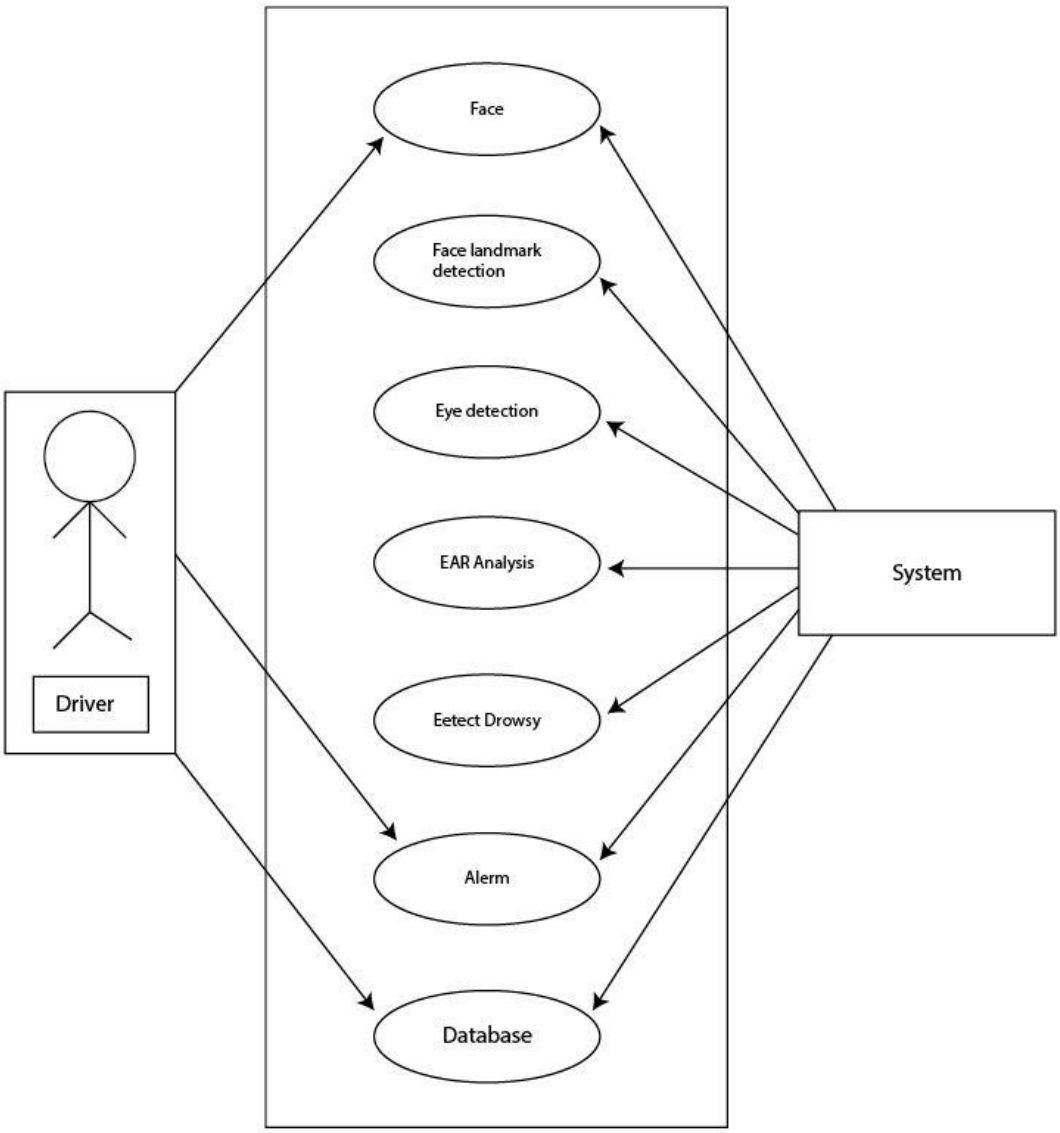


Fig 4.5.1: Use CASE Diagram

4.6 Activity Diagram

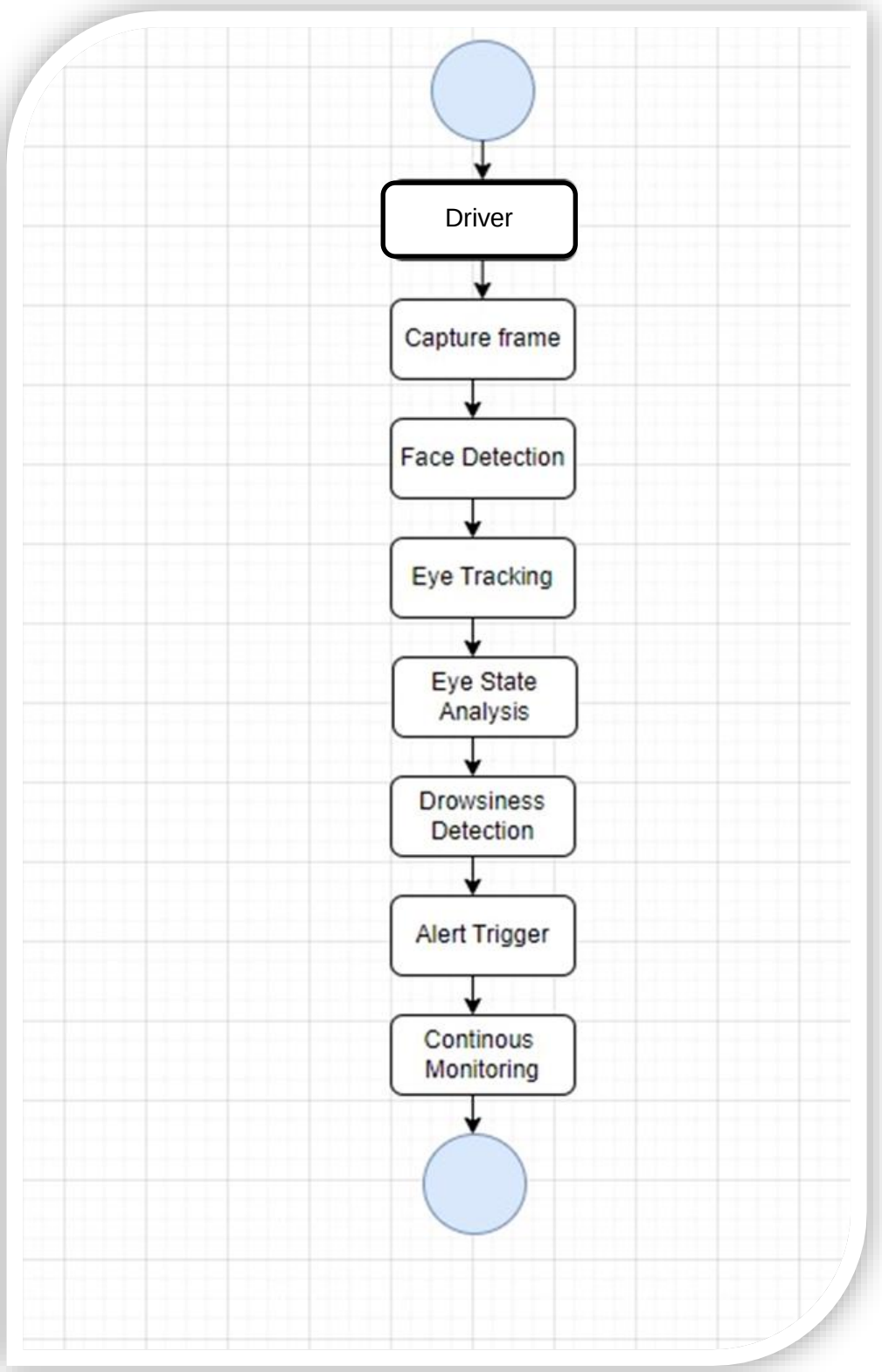


Fig 4.6.1: Activity Diagram

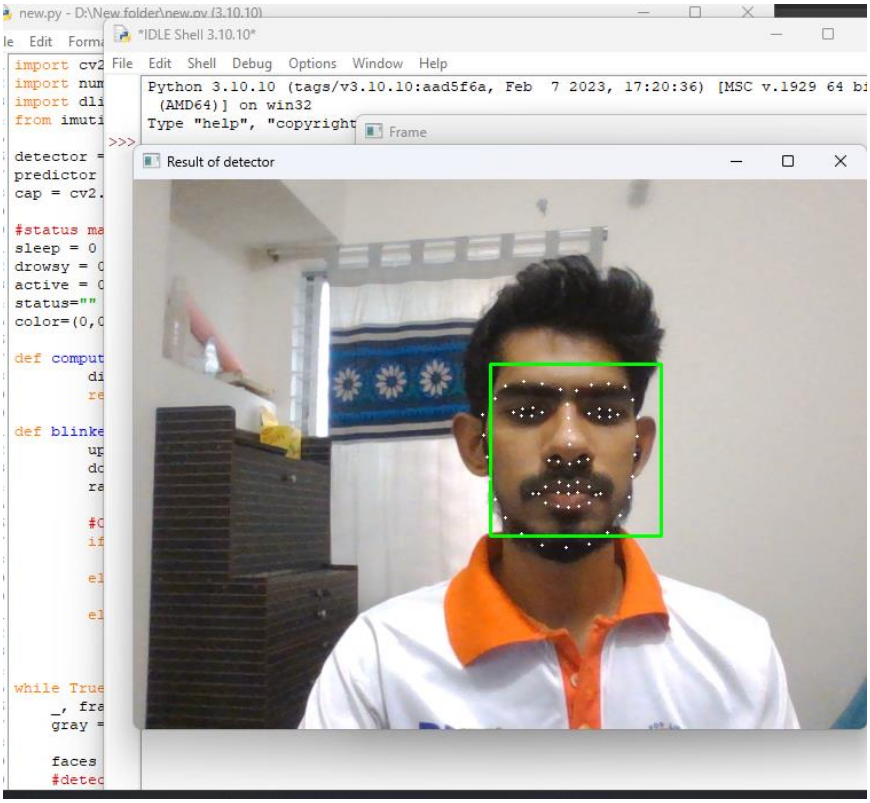
- **Driver**: The system initializes and starts monitoring the driver's drowsiness.
- **Capture Frame**: The system captures a frame from the video stream or an image from the camera.
- **Face Detection**: The system detects and locates the driver's face within the captured frame.
- **Eye Tracking**: The system focuses on the driver's eyes within the detected face region.
- **Eye State Analysis**: The system analyzes the state of the driver's eyes, such as open, closed, or blinking.
- **Drowsiness Detection:** Based on the eye state analysis and predefined drowsiness indicators, the system determines if the driver is drowsy or alert.
- **Alert Trigger:** If the system detects drowsiness, it triggers an alert mechanism to warn the driver. This can involve sounding an alarm, flashing lights, or other means of notification.
- **Continuous Monitoring:** The system continuously monitors the driver's drowsiness level and repeats steps 2-7 to provide real-time alerts and warnings.

CHAPTER- 05

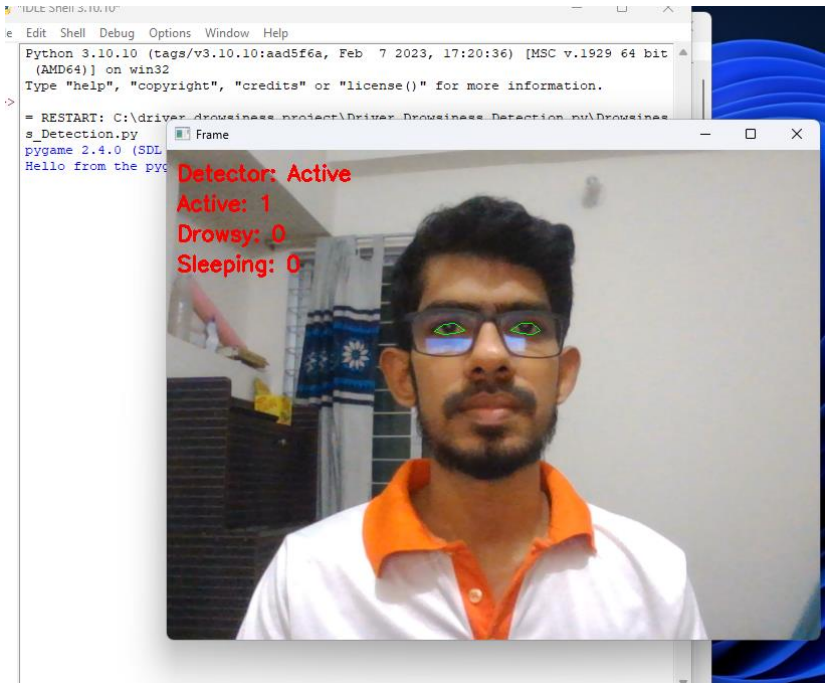
Implementation

Implementation

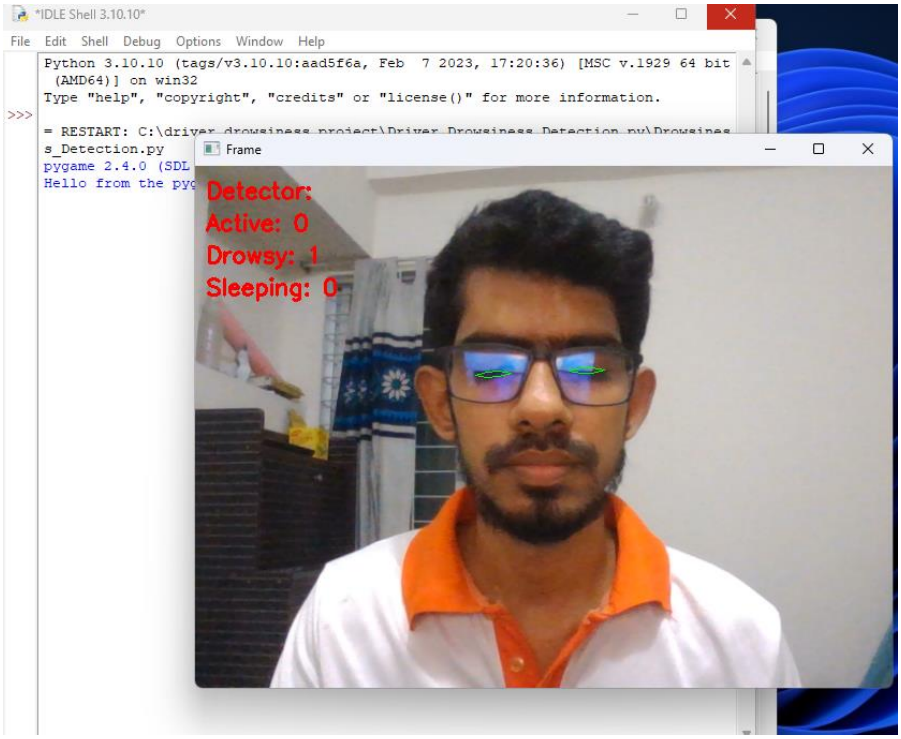
5.1 Face detect:



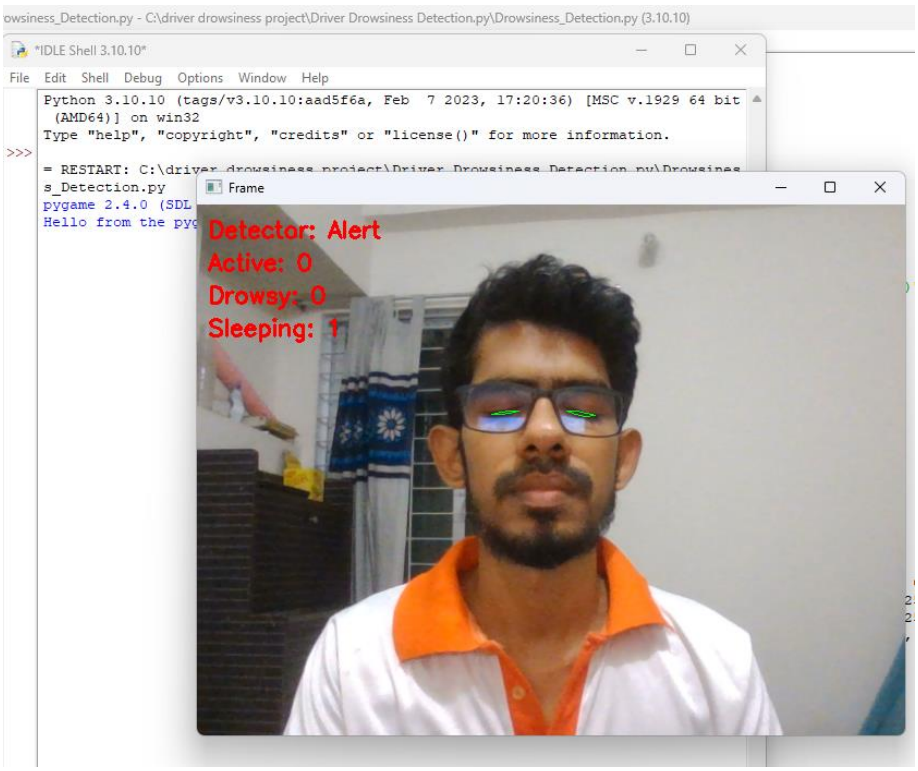
5.2 Active Status:



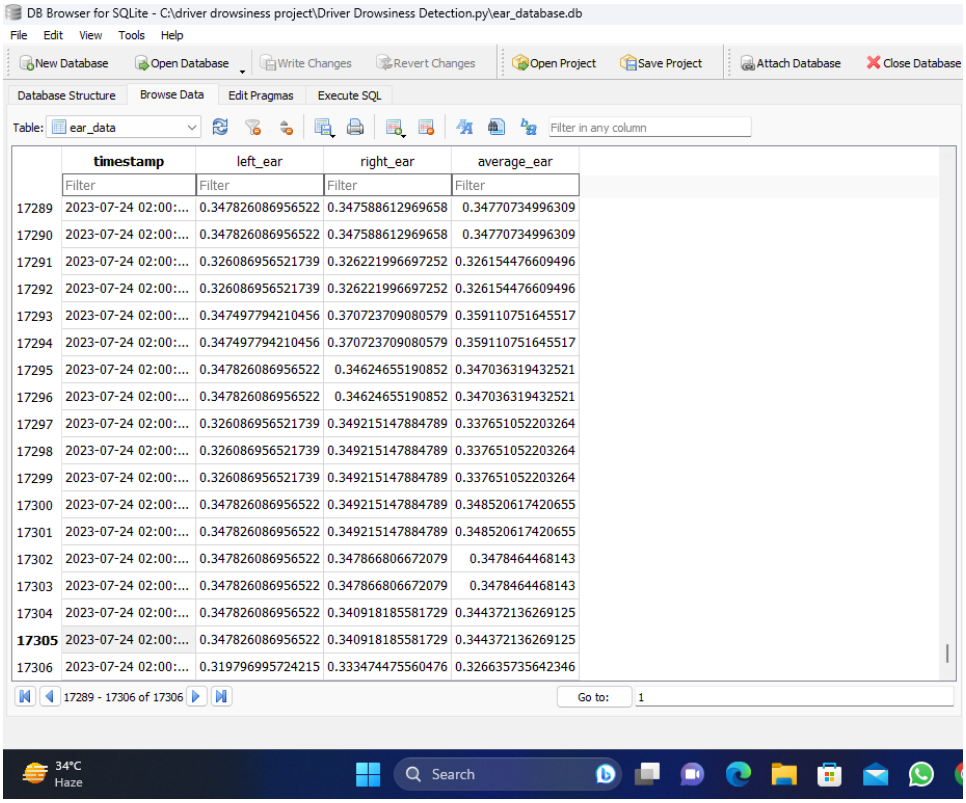
5.3 Drowsy Status:



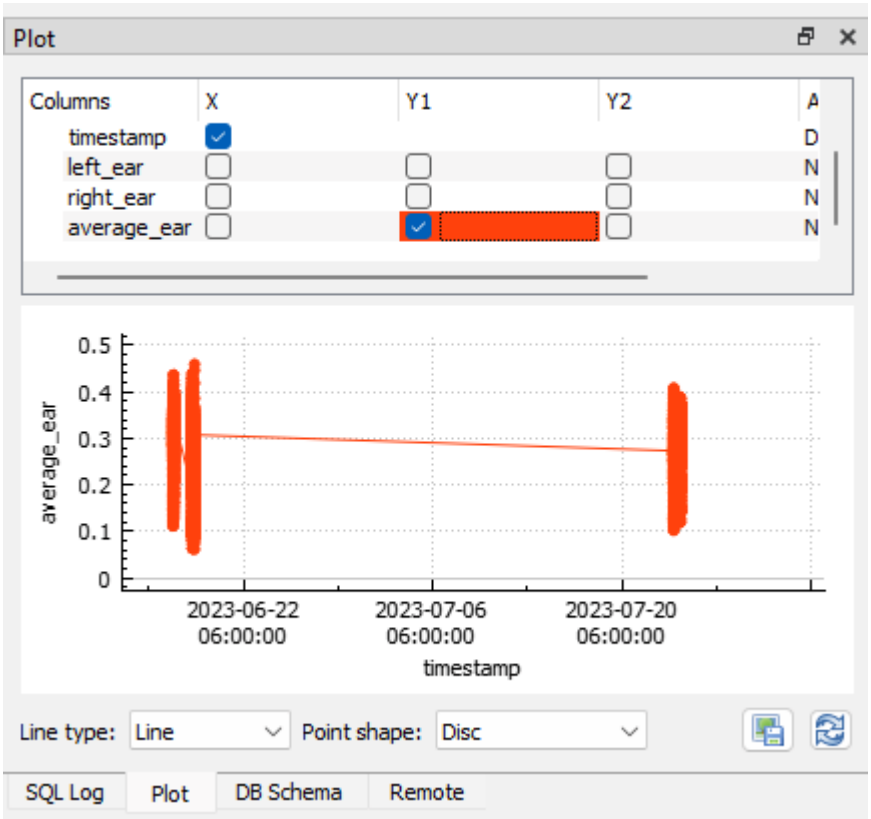
5.4 Sleeping Status:



5.5 Database:



5.6 Data Plot:



CHAPTER- 06

Conclusion

6.1 Limitation of our System

Here are some of the limitation of our Project:

- Dependency on Lighting
- Process time might be a slight delay.

6.2 Future Enhancement

- When Detect Drowsy a few time's in 5 minute the system automatic control the Car Engine and car switch off.
- System will be use in Mobile App and Web Browser.

6.3 Conclusion

The Driver Drowsiness Detection System aims to improve road safety by monitoring driver drowsiness in real-time. By leveraging Python, OpenCV, and dlib, the system detects signs of drowsiness, triggers timely alerts, and logs data for analysis. Although it has certain limitations, the proposed system holds significant potential for reducing accidents related to driver fatigue and contributing to safer driving experiences. The system will detect drowsiness by observing eye blinking patterns that is achieved by using Euclidean distance ratio eye blinking ratio.

References

[1] Eye Aspect Ratio (EAR) Method by Soukupová and Čech: "Real-time eye blink detection using facial landmarks." In Proceedings of the 2016 conference on empirical methods in natural language processing (EMNLP), 2016.

[2] Vandna Saini, Rekha Saini "Driver Drowsiness Detection System and Techniques", IJCSIT, Vol. 5 (3), 2014.

[3] K.Srijayathi, M.Vedachary "Implementation of the Driver Drowsiness Detection System", IJSETR, Volume 2, Issue 9, September 2013.

[4] Chisty, Jasmeen Gill, "Driver Drowsiness Detection System", IJCST, Volume 3, Issue 4, Jul-Aug 2015.

[5] Divya Chandan, "Drowsiness Detection Using MATLAB", IJCST, Volume 9, Issue 3, March-2018.

[6] Arun Sahayadhas, Kenneth Sundaraj & Murugappan Murugappan "Detecting Driver Drowsiness Based on Sensors", Sensors, 7 December 2012.

[7] Patil M.N., Brijesh Iyer, Rajeev Arya (2016) Performance Evaluation of PCA and ICA Algorithm for Facial Expression Recognition Application. In: Pant M., Deep K., Bansal J., Nagar A., Das K. (eds) Proceedings of Fifth International Conference on Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing, vol 436, pp 965-976. Springer, Singapore.

[8] Mitharwal Surendra Singh L., Ajgar Bhavana G., Shinde Pooja S., Maske Ashish M. "Eye Tracking Based Driver Drowsiness Monitoring & Warning System", IJTRA, Volume 3, Issue 3, May-June 2015.

Appendix

```
import cv2
import imutils
from imutils import face_utils
import dlib
from scipy.spatial import distance
from pygame import mixer
import sqlite3
import datetime
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.dates as mdates

mixer.init()
mixer.music.load("music.wav")

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

thresh = 0.25
flag = 0
frame_check = 20
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS['left_eye']
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_68_IDXS['right_eye']

detect = dlib.get_frontal_face_detector()
predict = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
cap = cv2.VideoCapture(0)

conn = sqlite3.connect('ear_database.db')
cursor = conn.cursor()

query = "SELECT timestamp, average_ear FROM ear_data"
df = pd.read_sql_query(query, conn)
```

```
cursor.execute("'CREATE TABLE IF NOT EXISTS ear_data (timestamp TEXT, left_ear REAL, right_ear REAL,average_ear REAL, drowsy INT)'")
conn.commit()

while True:
    ret, frame = cap.read()
    frame = imutils.resize(frame, width=650)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    subjects = detect(gray, 0)

    detected_status = ""
    active_status = 0
    drowsy_status = 0
    sleeping_status = 0

    for subject in subjects:
        shape = predict(gray, subject)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEar = eye_aspect_ratio(leftEye)
        rightEar = eye_aspect_ratio(rightEye)
        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)

        timestamp = datetime.datetime.now().strftime("%Y-%m-%d %I:%M:%S %p")
        cursor.execute("INSERT INTO ear_data (timestamp, left_ear, right_ear,average_ear, drowsy)VALUES (?, ?, ?, ?, ?)",
                       (timestamp, leftEar, rightEar, ear, 1 if ear < thresh else 0))
        conn.commit()

        if ear < thresh:
            flag += 1
            if flag >= frame_check:
                detected_status = "Alert"
                sleeping_status += 1
                mixer.music.play()
        else:
            flag = 0
            detected_status = "Active"
            active_status += 1
```

```
    drowsy_status = len(subjects) - active_status - sleeping_status

    cv2.putText(frame, "Detector: {}".format(detected_status), (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    cv2.putText(frame, "Active: {}".format(active_status), (10, 60),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    cv2.putText(frame, "Drowsy: {}".format(drowsy_status), (10, 90),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
    cv2.putText(frame, "Sleeping: {}".format(sleeping_status), (10, 120),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    cv2.imshow("Frame", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

df['timestamp'] = pd.to_datetime(df['timestamp'])
df.set_index('timestamp', inplace=True)

df_hourly = df.resample('H').mean()

plt.figure(figsize=(12, 6))
sns.lineplot(x=df_hourly.index, y=df_hourly['average_ear'], color='blue')
plt.xlabel('Time')
plt.ylabel('Average Drowsiness Level')
plt.title('Driver Drowsiness Over Time')
plt.xticks(rotation=45)
plt.gca().xaxis.set_major_formatter(mdates.DateFormatter("%Y-%m-%d  %I:%M:%S
%p"))
plt.tight_layout()
plt.show()
```