Driver Drowsiness Detection System

A Project Report Submitted
In Partial Fulfilment of the Requirements
for the Degree of

Bachelor of Technology
in
Information Technology

by

Jay Singh (1900520130024)
Manish Mayank (1900520130030)
Srijan Shankar Dubey (1900520310060)

Under the Guidance of
Dr. Natthan Singh
Mr. Abhishek Singh

Department of Computer Science and Engineering
Institute of Engineering & Technology
Dr. APJ Abdul Kalam Technical University
Lucknow

June, 2023

Declaration

We hereby declare that this submission of project is our own work and that to the best of our knowledge and belief it contains no material previously published or written by another person or material which to a substantial extent has been accepted for award of any other degree of the university or other institute of higher learning, except where due acknowledgement has been made in the text.

This project report has not been submitted by us to any other institute for requirement of any other degree.

Signature of the Students

Jay Singh (1900520130024)

Manish Mayank (1900520130030)

Srijan Shankar Dubey (1900520310060)

Certificate

This is to certify that the project report entitled: **Driver Drowsiness Detection System** submitted by **Jay Singh, Manish Mayank and Srijan Shankar Dubey** in the partial fulfillment for the award of the degree of Bachelor of Technology in Information Technology is a record of the bonafide work carried out by them under our supervision and guidance at the Department of Computer Science & Engineering, Institute of Engineering & Technology Lucknow.

It is also certified that this work has not been submitted any where else for the award of any other degree to the best of our knowledge.

(Mr. Abhishek Singh)
Department of Computer Science and Engineering,
Institute of Engineering & Technology, Lucknow

(Dr. Natthan Singh)
Department of Computer Science and Engineering,
Institute of Engineering & Technology, Lucknow

Acknowledgement

We take this opportunity to acknowledge and express our gratitude to all those who supported and guided us during our project work. We are grateful to the Almighty for the abundant grace and blessings bestowed upon us, which enabled us to successfully complete this project.

We are deeply grateful to our supervisor, **Dr. Natthan Singh**, for his unwavering support and guidance throughout my B.Tech project. Their expertise and patience have been invaluable to us and have played a crucial role in the success of this thesis, and would like to also thank our Co-supervisor **Mr. Abhishek Singh**, for his invaluable supervision, support during the course of my B.Tech.

We are grateful to Project co-ordinator **Dr. Pawan Kumar Tiwari** for providing us with the opportunity to conduct our thesis project and for all of the resources and support they provided.

We would also like to thank **Project Evaluation Committee Members** for serving on my thesis committee and providing valuable feedback and suggestions. Their insights and guidance were instrumental in helping us to shape our research and write this thesis.

We are also grateful to Head of Department of Computer Science and Engineering **Dr. Yogendra Narain Singh** for providing us with the opportunity to conduct our thesis project and for all the support he provided.

We are deeply thankful to our friends and family for their love and support during this process. Without their encouragement and motivation, we would not have been able to complete this journey.

Jay Singh

Manish Mayank

Srijan Shankar Dubey

Abstract

Driver drowsiness poses a significant threat to road safety worldwide, being a major cause of road accidents. To tackle this issue, the development of the Driver Drowsiness Detection System (DDDS) has taken place. This report aims to deliver a comprehensive summary and evaluation of the DDDS.

The DDDS utilizes cutting-edge computer vision and machine learning techniques to continuously monitor the driver's state and identify signs of drowsiness in real time. By analyzing facial features, eye movements, and head position, the system can accurately assess the driver's level of alertness. Employing sophisticated image processing algorithms, the DDDS tracks crucial facial landmarks like eye closure and yawning, enabling the detection of fatigue-related signs.

The report delves into the design and implementation aspects of the DDDS, including the hardware and software components employed. It elaborates on the data collection process and the training of machine learning models utilized for drowsiness detection. Additionally, the report provides insights into the algorithms and techniques utilized for real-time monitoring and alert generation.

The results demonstrate that the DDDS achieves a high level of accuracy in real-time drowsiness detection. Its ability to reliably identify driver drowsiness makes it a valuable tool in preventing accidents caused by fatigue. The report also addresses potential enhancements and future directions for further improving the system's performance and expanding its capabilities.

In conclusion, the Driver Drowsiness Detection System presented in this report showcases promising results in mitigating the risks associated with driver drowsiness. Its capacity to accurately detect drowsiness in real-time establishes it as a valuable addition to automotive safety systems, potentially reducing the number of accidents occured by driver fatigue and ultimately improving road safety.

Contents

Declaration i

Certificate ii

Acknowledgements iii

Abstract iv

Contents v

List of Figures vii

List of Tables viii

1 Introduction 1
- 1.1 Objectives. . . . 2
- 1.2 Scope 4

2 Literature Review 6
- 2.1 Related Work 6
- 2.2 Existing Approaches and their Limitations 7
  - 2.2.1 Behavioral monitoring techniques 7
  - 2.2.2 Physiological monitoring techniques 9
  - 2.2.3 Environmental monitoring techniques 10

3 Methodology 13
- 3.1 Technologies Used. . . . 13
  - 3.1.1 Frontend 13
  - 3.1.2 Backend 14
  - 3.1.3 Tools 20
  - 3.1.4 Flow Chart 23
  - 3.1.5 Usecase Diagram 24
- 3.2 Models and Algorithms 25
- 3.3 Adopted Methodology 27

4 Experimental Results **31**

5 Conclusion **35**

A Appendix **37**

References **43**

List of Figures

3.1 Flowchart of Driver Drowsiness Detection System. 23
3.2 Usecase Diagram of Driver Drowsiness Detection System. 24
3.3 Open eye. 28
3.4 Close eye. 28
3.5 Plotted points images open and close eye 28
3.6 Open mouth. 29
3.7 Close mouth. 29
3.8 Plotted Points on Open and Close Mouth. 29

4.1 Homepage of Driver Drowsiness Detection System. 32
4.2 About Creator of Driver Drowsiness Detection System. 33
4.3 Active State. 33
4.4 Drowsy State. 34
4.5 Drowsy-Yawning State. 34

List of Tables

2.1 Comparison table of the different techniques for drowsiness detection. 12

Chapter 1

Introduction

In today's fast-paced world, where time is precious thing, spending long hours behind the wheel has become a regular part of our lives. However, the increasing occurrence of driver fatigue poses a significant threat to road safety. Statistics indicate that drowsy driving is responsible for a considerable number of accidents, injuries, and fatalities worldwide. This highlights the critical need for effective measures to address this alarming issue.

Fortunately, advancements in technology have paved the way for innovative solutions to tackle driver drowsiness. One such solution is the development of driver drowsiness detection systems. These systems utilize cutting-edge technologies like computer vision, machine learning, and physiological monitoring to identify signs of driver fatigue and alert the driver before a potentially dangerous situation arises.

This study aims to explore the various aspects of driver drowsiness detection systems, including their underlying principles, components, and real-world applications. By shedding light on this emerging field, we aim to emphasize the importance of implementing these systems in vehicles and promoting a safer driving environment for all road users.

This paper will deliver the working mechanisms of driver drowsiness detection systems, discussing the different sensing techniques employed, the algorithms used for analysis, and the

methods of alerting the driver. Additionally, will examine the effectiveness of these systems based on existing research and studies conducted in both controlled environments and real-world scenarios.

Furthermore, it will explore the challenges faced by developers and researchers in the field, such as variability in driver behavior and environmental conditions, and discuss potential solutions and future directions for improvement.

At the end of this research paper, we will understand driver drowsiness detection systems, their significant role in reducing the accidents caused by driver fatigue, and their potential to revolutionize road safety. Implementing these systems in vehicles holds the promise of reducing the occurrence of drowsy driving incidents and safeguarding the lives of countless individuals on the road.

In conclusion, this study aims to shed light on the growing field of driver drowsiness detection systems, emphasizing their importance, working mechanisms, and future potential. Through the integration of advanced technology and a proactive approach to road safety, we can strive towards a future where drowsy driving becomes a thing of the past, and our roads become safer for everyone involved.

1.1 Objectives.

The objective of this Driver Drowsiness Detection System are as follows:

Enhance Road Safety: The primary objective of the DDDS is to enhance road safety by effectively detecting driver drowsiness. The system aims to prevent accidents caused by fatigue-related impairment by alerting drowsy drivers in real-time.

Reduce Accidents: The aim of this system is to decrease the counting of accidents due to driver drowsiness. By accurately identifying signs of drowsiness, such as eye closure or head nodding, the system can prompt drivers to take necessary actions or rest, potentially preventing accidents.

Real-time Monitoring: The system aims to continuously monitor the driver's state in real-time. By employing the latest computer vision and machine learning technologies, the system can promptly detect drowsiness and provide timely alerts to ensure driver safety.

Utilize Advanced Technologies: The DDDS aims to leverage advanced technologies, such as computer vision and machine learning, to effectively detect drowsiness. By analyzing facial features, eye movements, and head position, the system can assess the driver's level of alertness accurately.

Improve Driver Awareness: The system's objective is to improve driver awareness regarding their drowsy state. By providing real-time feedback and alerts, the DDDS aims to increase drivers' self-awareness and encourage them to take appropriate actions to prevent accidents.

Customizable and Adaptive: The DDDS aims to be customizable and adaptive to different individuals and driving conditions. By considering individual variations and adapting to diverse environments, the system aims to provide accurate and reliable drowsiness detection for a wide range of drivers.

Research and Development: The DDDS seeks to encourage further research and development in the field of driver drowsiness detection. By identifying areas for improvement and exploring new techniques and algorithms, the system aims to continuously enhance its performance and capabilities.

The goal of the Drowsiness Detection and Driver Safety (DDDS) system is to effectively integrate with pre-existing automotive safety systems. This seamless integration enables the DDDS

to enhance the overall safety framework of vehicles by collaborating with other systems like collision avoidance or lane departure warning systems. By doing so, the DDDS contributes to a comprehensive and robust safety infrastructure within vehicles.

In summary, the objectives of the Driver Drowsiness Detection System include enhancing road safety, reducing accidents caused by driver drowsiness, real-time monitoring, utilizing advanced technologies, improving driver awareness, customization, and adaptability, fostering research and development, and integration with existing safety systems.

1.2 Scope

The scope of the Driver Drowsiness Detection System (DDDS) encompasses various aspects related to the detection and prevention of driver drowsiness. The scope includes, but is not limited to, the following:

Drowsiness Detection: The DDDS system is designed to detect signs of driver drowsiness by analyzing physiological and behavioral indicators, such as facial expressions, eye movements, and head position. Its main focus is on accurately identifying real-time manifestations of drowsiness, allowing for prompt interventions when necessary.

Hardware and Software Integration: The Driver Drowsiness Detection System involves the integration of hardware components, such as cameras or sensors, with software algorithms for effective drowsiness detection. The scope includes the development and integration of both hardware and software components to create a seamless and functional system.

Data Collection and Processing: The system collects data from various sources, including facial images, eye movements, and head position, to analyze and detect drowsiness accurately. The scope includes defining the data collection protocols, preprocessing the data, and applying appropriate algorithms for feature extraction and analysis.

Machine Learning and Algorithm Development: The DDDS utilizes machine learning techniques and algorithms to train models that can identify patterns and indicators of drowsiness. The scope of this study encompasses the development and enhancement of algorithms with the aim of enhancing the accuracy and reliability of the drowsiness detection system.

Performance Evaluation: The system's performance is required to be evaluated to assess its accuracy in identifying drowsiness conditions. The scope includes designing appropriate evaluation methodologies, selecting performance metrics, and conducting comprehensive testing using diverse datasets and real-world driving scenarios.

Alert Generation and User Interface: The DDDS generates alerts or warnings to notify the driver of their drowsy state. The scope includes designing an effective alert generation system and developing a user-friendly interface that provides clear and timely information to the driver.

Integration with Vehicle Safety Systems: The DDDS system has the capability to seamlessly integrate with pre-existing vehicle safety systems, such as collision avoidance or lane departure warning systems, to establish a comprehensive safety framework. This integration involves exploring potential integration opportunities and ensuring compatibility with other safety systems, thereby enhancing overall safety measures.

Ethical and Legal Considerations: The DDDS takes into account ethical and legal considerations concerning privacy, data protection, and compliance with regulations.

In summary, this system covers various aspects ranging from drowsiness detection and real-time monitoring to hardware and software integration, data collection and processing, algorithm development, performance evaluation, alert generation, integration with vehicle safety systems, ethical and legal considerations, and future enhancements and research.

Chapter 2

Literature Review

Driver drowsiness detection systems have gained significant attention in recent years. They employ various techniques such as machine learning, image processing, and physiological measurements to identify signs of driver fatigue. These systems have proven to be effective in preventing accidents by providing timely alerts or interventions to drowsy drivers. However, further research is needed to enhance their accuracy, robustness, and real-time capabilities.

2.1 Related Work

Driver drowsiness is a critical reason in contributing to road accidents globally. To mitigate this issue, researchers have developed driver drowsiness detection systems (DDDS) that employ advanced technologies such as computer vision, machine learning, and physiological monitoring. The aim of this literature review is to provide an overview of existing studies and approaches in the sector of driver drowsiness detection. Extensive research has established a strong link between driver drowsiness and road accidents. Fatigue impairs cognitive functions, reaction

time, and alertness, significantly increasing the risk of accidents. Therefore, the development of reliable and accurate drowsiness detection systems is crucial for improving road safety.

2.2 Existing Approaches and their Limitations

Researchers have adopted various techniques for driver drowsiness detection, including behavioral, physiological, and environmental monitoring. Behavioral monitoring techniques analyze driver actions, such as eye closure, yawning, or head nodding, to identify signs of drowsiness. Physiological monitoring involves measuring physiological indicators such as heart rate, EEG signals, or skin conductance to detect drowsiness. Environmental monitoring considers driving behavior and context, including lane deviation or steering wheel movements.

2.2.1 Behavioral monitoring techniques

- **Thermal imaging-based tiredness detection:** Knapik and Cyganek developed a system that uses infrared thermal imaging to identify tiredness based on the cavernous of the driver. The system achieved an accuracy of 71.5% and 88% to detect cold and hot defocus, respectively [1].

- **Respiration analysis by using the method thermal imaging:** Kiashari et al.[2] and colleagues utilized facial thermal imaging to examine the driver's breathing patterns for identifying drowsiness. By applying machine learning classifiers to features extracted from the breathing signal, they achieved an accuracy rate of 90% .

- **Analysis of Eye Blinking Patterns:** Khan et al.[3] and colleagues created a system that can determine if the driver's eyes are open or closed by analyzing the movement of their eyelids. The system achieved high accuracy, correctly identifying open or closed eyes in both simple images and real-time surveillance videos.

- **Algorithm based on Optical Correlators:** Ouabida et al.[4] and colleagues suggested a method that uses an optical correlator to determine if the eye is open or closed by employing correlation filters. This method demonstrated good performance in accurately recognizing the eye state, even in environments with noise and distractions .

- **Calculation of Eye aspect ratio :** Maior et al.[5] created a method that can detect drowsiness in real time by calculating something called the eye aspect ratio (EAR). The system keeps track of how long a person blinks and achieved an average accuracy of 94% in tests using a type of classification called support vector machine (SVM).

- **Approach that combines multiple features:** Celecia et al.[6] introduced a device that combines information from the driver's eyes and mouth to detect levels of drowsiness. By utilizing a Mamdani fuzzy inference system, the device successfully achieved an accuracy of 95% in identifying drowsiness.

- **Deep learning-based approaches:** Various techniques employed advanced deep learning models to detect drowsiness. These methods included combining different approaches, using specialized network architectures, adapting representations based on conditions, and utilizing a combination of convolutional neural networks (CNN) and long short-term memory (LSTM). These approaches achieved accuracy rates ranging from 76.21% to 97.05%.

- **Recursive neural network (RNN) for fatigue analysis:** Ed-Doughmi et al.[7] utilized a type of neural network called a recurrent neural network (RNN) to analyze a series of video frames and predict fatigue. The method successfully detected drowsy behaviors like yawning, eye closure, and head nodding with an accuracy of 97.3%.

- **Multitask cascaded CNN:** Zhao et al. created an automated algorithm for detecting driver fatigue using a multitask cascaded convolutional neural network (CNN) [8]. The algorithm focuses on detecting the states of the mouth and eyes, achieving an accuracy of 93.62%.

These methods show that image-based systems have the potential to detect drowsiness and fatigue, with accuracy rates ranging from 72.25% to 99.59%. It's crucial to consider factors like tracking facial data in different environments, as they can affect the performance of these systems.

2.2.2 Physiological monitoring techniques

- **Drowsiness detection using EEG signals:** Li et al.[9] proposed a system that classifies drowsiness states using EEG signals and an SVM-based model. Kaur and Singh used EEG signal analysis and artificial neural networks, achieving an accuracy of 88.22%. Budak et al.[10] employed feature extraction techniques and LSTM networks, achieving an average accuracy of 94.31%. Taran and Bajaj utilized adjustable Hermite decomposition and the major learning machine classifier, achieving an accuracy of 92.28%.

- **Drowsiness detection using PPG, ECG, and HRV signals:** Lee et al.[11] utilized HRV signals from wearable PPG or ECG sensors and a CNN classifier, achieving improved accuracy using the ReLU-RP method. Koh used PPG signals to detect drowsiness based on LF, HF, and LF/HF values, showing significant differences between awake and drowsy states. Kundinger developed a retrofittable system using bodily data from a wrist-worn wearable sensor, achieving an accuracy of approximately 92.13%. Fujiwara employed HRV anomaly analysis, achieving a precision of 92.1%.

- **Detecting drowsiness by analyzing signals related to breathing:** Guede-Fernández et al. developed a method that analyzes changes in breathing signals to detect drowsiness with a high accuracy of 96.6% and a sensitivity of 90.3% [12].

- **Detection of drowsiness using signals from electromyography :** Sahayadhas et al.[13] developed a system for detecting hypervigilance using ECG and EMG signals,

employing higher-order spectral features for classification. Fu and Wang utilized a non-contact EMG and ECG system to analyze changes in signals during driving, achieving a correctness of 86.5%.

- **Drowsiness detection using a combination of different biological signals:** By combining ECG and EEG features, Wais achieved an accuracy of 80.90% in drowsiness detection. Khushaba utilized EEG and ECG signals with a fuzzy wavelet packet-based feature-extraction algorithm, achieving accuracy rates of 95.5% and 97.1% when evaluated with various classifiers.

  The reported accuracy of these physiological-based Driver Drowsiness Detection systems ranges from 70% to 97.19%. These methods often rely on brain activity signals and may involve intrusive or invasive sensor placement.

2.2.3 Environmental monitoring techniques

- **Lane departure analysis using Steering Wheel Angle :** McDonald et al. [14] developed a method to analyze when a car drifts out of its lane by looking at the data of steering wheel angle and using a computer algorithm called random forest (RF). Their approach was more accurate, reaching 79.5%, compared to a method that used images called PERCLOS. To train the RF algorithm, they used data from a driving simulator at the University of Iowa that simulated drowsy driving situations. On the other hand, PERCLOS used eye-detection software to analyze videos. By using the Steering wheel angle data, they could predict drowsiness with 55.5% accuracy and provide a warning 6 seconds in advance.

- **Detection of the distance from the side using wavelet transform and neural network:** Mac et al.[15] developed a model that detected driver drowsiness based on lateral distance. This was achieved by fusing lane curvature, position, and curvature

derivative data obtained from a front bumper-mounted video camera. The model also incorporated a real-time recording of the driver's facial and head movements. Classification using support vector machine and neural network algorithms achieved detection accuracy of more than 90%.

- **Real-time detection of fatigue using Steering Wheel Angle data:** Li et al. [16] and colleagues developed a system that used Steering Wheel Angle data to detect fatigue in real time. They extracted features called approximate entropy from the steering wheel angle data collected during real driving situations. The system calculated the difference between the feature series to determine if the driver was drowsy or awake. Using a binary decision classifier, they achieved an accuracy of 84.86% for detecting drowsiness and 78.02% for detecting wakefulness.

- **Non-invasive drowsiness detection using steering wheel data:** Arefnezhad et al.[17] developed a system that detected drowsiness using steering wheel data in a non-invasive manner. They utilized adaptive neuro-fuzzy inference systems to select the most important features, which improved the accuracy of classification. By combining this feature selection approach with a support vector machine classifier, the system achieved an accuracy of 98.12%.

- **Steering wheel status monitoring:** Chai et al.[18] research concentrated on monitoring drowsiness by examining specific parameters associated with the steering wheel. Four parameters were chosen based on their connection to the driver's state. Three models, including multilevel ordered logic, support vector machine, and backpropagation neural network models, were developed using these parameters. Among them, the multilevel ordered logic model achieved the highest accuracy of 72.93%.

- In summary, vehicle-based systems for drowsiness detection have reported accuracy ranging from 62.1% to 98.12%. These methods predominantly rely on features derived from

SWA and are considered non-intrusive and non-invasive approaches to detecting drowsiness.

TABLE 2.1: Comparison table of the different techniques for drowsiness detection.

References and Author	**Drowsiness Detection Technique**	**Classification**	**Positive Detection Rate**
Kiashari et al.[2]	Facial Thermal Imaging	Support Vector Machine (SVM)	90%
Maior et al.[5]	Eye Aspect Ratio(EAR)	Support Vector Machine (SVM)	94%
Celecia et al.[6]	Fusion between Eye and Mouth Detection	Fuzzy Classifier	95%
Budak et al.[10]	Electroencephalogram(EEG) Signals	Long Shot Term Memory (LSTM)	94.31%
Lee et al.[11]	PPG, ECG and HRV Signals	Convolution Neural Network(CNN)	92%
McDonald et al. [14]	Steering Wheel Angle(SWA)	Random Forest (RF)	79.5%

Different methods for detecting driver drowsiness and fatigue situations, each with varying accuracy levels, and their classification methods are mentioned in Table 2.1.

Chapter 3

Methodology

Before starting about the procedures and methods involved while developing this project, let us know about the technologies used while developing this project.

3.1 Technologies Used.

The following technologies are used to develop this project.

3.1.1 Frontend

HTML/CSS/JavaScript When creating a driver drowsiness detection system, HTML, CSS, and JavaScript are important for building the user interface and improving the application's features.

HTML is a coding language that is commonly used to organize and display content on web pages. In the context of the driver drowsiness detection system, HTML is used to create the

structure and layout of the user interface. It defines various elements like buttons, forms, images, and text that form the visual part of the application.

CSS is a language used to make web pages look good. It controls how things like colors, fonts, and layouts appear on the screen. In the driver drowsiness detection system, CSS is used to make the application visually appealing and easy to use by customizing its appearance.

JavaScript is a programming language that adds interactivity and dynamic behavior to web pages. In the driver drowsiness detection system, JavaScript is employed to implement various functionalities and logic. It enables real-time video processing and analysis, captures user input from buttons or forms, handles events such as button clicks, and communicates with the backend for data processing and predictions.

By combining HTML, CSS, and JavaScript, developers can create a responsive and interactive user interface for the driver drowsiness detection system. HTML defines the structure, CSS enhances the visual appeal, and JavaScript adds the necessary functionality to enable real-time video processing, user input handling, and communication with the backend. This combination allows for an intuitive and seamless user experience while ensuring the efficient detection of drowsiness cues for enhanced road safety.

3.1.2 Backend

Python Python is an extensively used programming language for implementing the backend of web applications. Its popularity stems from its simplicity, readability, and the vast array of libraries and frameworks available. Python offers a clean and intuitive syntax that allows developers to write code quickly and efficiently.

When it comes to web development, Python offers several powerful frameworks like Django and Flask. These frameworks provide a structured and organized approach to building web applications, with features such as routing, database integration, and user authentication already built-in. They also promote code reusability and follow the principle of Don't Repeat Yourself (DRY), making development faster and more maintainable.

Python's versatility is another advantage. It can seamlessly integrate with other technologies and platforms, allowing developers to connect their web applications with various databases, messaging systems, and external APIs. Python also supports the creation of RESTful APIs and microservices, making it suitable for building scalable and modular backend systems.

Furthermore, Python boasts a large and active community that contributes to the development of libraries, frameworks, and tools. This community support ensures that developers have access to a vast repository of resources, documentation, and community-driven solutions, making it easier to troubleshoot issues and find solutions.

In summary, Python's simplicity, extensive libraries and frameworks, versatility, and strong community support make it an excellent choice for implementing the backend of web applications. Its ability to handle complex backend functionalities and seamlessly integrate with other technologies makes it a preferred language for building scalable and robust web systems.

Libraries used

- **OpenCV**

OpenCV (Open Source Computer Vision Library) is a widely used free library for creating computer vision and image processing applications

As a backend tool, OpenCV provides a wide range of functionalities that can be leveraged in different domains. It offers powerful image and video manipulation capabilities, allowing developers to perform tasks such as image filtering, object detection, feature extraction, and

tracking. With OpenCV, developers can access and process image data, analyze patterns, and extract relevant information for further processing.

In addition to image processing, OpenCV also offers tools for camera calibration, stereo vision, and 3D reconstruction. These features make it suitable for applications involving augmented reality, robotics, and depth sensing.

OpenCV can be used with various programming languages like Python, C++, and Java, allowing developers with different language preferences to easily utilize its functionalities. Its extensive documentation, rich set of APIs, and community support make it easier for developers to integrate OpenCV into their projects and leverage its functionalities.

By using OpenCV as a backend, developers can harness the power of computer vision to create innovative applications across various industries, such as healthcare, surveillance, automotive, and entertainment. Its robust feature set and flexibility make it a valuable tool for implementing complex image processing tasks and enabling advanced vision-based functionalities in applications.

- **NumPy**

NumPy (Numerical Python) is a powerful library in Python used for numerical computing and scientific computing. When implemented as a backend, NumPy provides efficient and high-performance capabilities for mathematical operations, data manipulation, and array processing.

As a backend tool, NumPy offers a multi-dimensional array object called ndarray, which allows efficient storage and manipulation of large datasets. The ndarray object provides a convenient and optimized way to perform mathematical operations on arrays, such as element-wise operations, linear algebra operations, and statistical computations.OpenCV also provides many useful tools and functions for working with numbers and arrays in computer vision and image processing applications.

NumPy has a special feature called vectorized operations that allows for faster and more efficient calculations compared to regular Python lists. Its optimized algorithms and C implementation make it great for working with big sets of data and doing complicated mathematical calculations.

In addition, NumPy offers functionalities for seamless integration with various libraries and frameworks, including data visualization libraries like Matplotlib, machine learning libraries such as scikit-learn, and scientific computing libraries like SciPy. This interoperability greatly enhances its utility as a backend tool for diverse applications, including data analysis, scientific computing, and machine learning.

Overall, implementing NumPy as a backend in Python projects allows developers to leverage its efficient array processing capabilities, mathematical functions, and high-performance computing features. NumPy is a valuable tool for faster and more efficient mathematical calculations, making it crucial for data-intensive and scientific computing tasks.

- **Scipy**

Scipy (Scientific Python) is a powerful library in Python used for scientific and technical computing. When implemented as a backend, Scipy provides a wide range of functionality for numerical optimization, integration, interpolation, signal processing, linear algebra, and more.

As a backend tool, Scipy extends the capabilities of NumPy by providing additional high-level functions and algorithms for scientific computing. It offers a collection of sub-packages that cover various domains, including optimization, signal processing, statistics, image processing, and more. These sub-packages contain numerous functions and classes that enable advanced scientific computations and data analysis.

Some key features of Scipy include numerical integration, solving ordinary differential equations, Fourier transforms, interpolation, statistical functions, and image processing routines. It also provides tools for linear algebra operations, sparse matrix computations, and optimization algorithms for constrained and unconstrained problems.

Scipy seamlessly merges with popular scientific libraries like NumPy, Matplotlib, and Pandas, enabling it to serve as a flexible and powerful tool for scientific computation and data analysis tasks. It provides a comprehensive and efficient environment for tackling complex scientific problems and performing advanced computations.

Implementing Scipy as a backend in Python projects enables developers to leverage its extensive collection of scientific functions and algorithms. It allows for efficient and convenient handling of scientific data, analysis of experimental results, and solving complex mathematical problems. Scipy's capabilities make it an essential tool for researchers, scientists, and engineers working in various scientific disciplines

• **Pandas**

Pandas is a popular library in Python that is often implemented as a backend for data manipulation and analysis tasks. It offers user-friendly data structures and analysis tools that are invaluable for handling and analyzing structured data effectively.

When implemented as a backend, Pandas offers a wide range of functionalities for handling and analyzing data, including reading and writing data from various file formats, data cleaning and preprocessing, data transformation and manipulation, merging and joining datasets, and performing statistical analysis.

Pandas brings forth two primary data structures: Series and DataFrame. A Series is like a labeled array with a single dimension, while a DataFrame resembles a two-dimensional labeled structure akin to a table or spreadsheet. These structures, along with Pandas' powerful indexing and slicing capabilities, enable efficient data handling and analysis.

Furthermore, Pandas provides a plethora of functions and methods for data aggregation, grouping, reshaping, and merging, allowing users to efficiently extract insights from their data. Its integration with other Python libraries, such as NumPy and Matplotlib, makes it even more versatile and powerful for data analysis and visualization tasks.

Overall, Pandas serves as a robust and efficient backend for data manipulation and analysis, making it an essential tool in the Python ecosystem for working with structured data.

• **Dlib**

Dlib is a powerful library that can be used as a backend in various projects, including driver drowsiness detection systems. When integrated into the backend of a project, Dlib provides a range of features and functionalities that contribute to the effectiveness and accuracy of the system.

One of the key applications of Dlib in the context of driver drowsiness detection is facial landmark detection. Dlib provides pre-trained models and algorithms that can precisely identify specific facial features like the eyes, eyebrows, and mouth. By detecting these landmarks, the system can track and analyze specific facial features that indicate signs of drowsiness, such as eye closure or drooping eyelids.

Dlib's facial landmark detection capabilities can be combined with other algorithms and techniques to extract relevant features from the face, such as eye aspect ratio or mouth opening, which are commonly used indicators of drowsiness. By leveraging Dlib's facial landmark detection, the system can actively examine and track the driver's facial expressions and motions in real-time, allowing for effective analysis and monitoring.

Furthermore, Dlib offers machine learning techniques like support vector machines (SVM) and deep neural networks, enabling the creation of trained models specifically designed for drowsiness classification. These algorithms can learn from a labeled dataset of drowsy and alert states, enabling the system to accurately classify the driver's current state based on the extracted features.

Dlib's efficient implementation and optimized performance make it well-suited for real-time applications. It can process video streams or image frames quickly, allowing the backend to provide timely alerts or warnings when drowsiness is detected.

By leveraging Dlib in the backend, developers can benefit from its comprehensive set of tools and algorithms for facial landmark detection, machine learning, and computer vision. These capabilities enable the creation of robust and accurate driver drowsiness detection systems that enhance safety on the roads.

3.1.3 Tools

- **Git**

Git is a version control system that helps developers manage and collaborate on code efficiently. It allows multiple developers to work on different parts of a project at the same time and easily merge their changes together. This promotes teamwork and enables effective code management and tracking in software development.

One of Git's key advantages is its branching and merging capabilities. Developers can create branches to work on specific features or experiments and easily merge changes back into the main codebase when ready. This enables efficient code management and facilitates the integration of new features into the project.

Git also provides a comprehensive history of code changes, allowing developers to track the evolution of the codebase, roll back to previous versions, and identify the authors of specific changes. This feature provides a reliable audit trail and simplifies troubleshooting and debugging processes.

The versatility of Git is enhanced by its integration with various development tools and platforms. It seamlessly integrates into the software development workflow, supporting collaboration among team members and providing a robust platform for managing code repositories.

Furthermore, Git's widespread adoption and large user community contribute to its popularity. Extensive documentation and resources are available, making it easy for developers to learn and utilize Git effectively in their projects.

In conclusion, Git is a powerful and indispensable version control system that empowers developers with efficient code management, collaboration, and version control capabilities. Its decentralized architecture, branching and merging features, and extensive community support make it a fundamental tool in modern software development.

- **Visual Studio Code**

Visual Studio Code (VS Code) is a popular and versatile code editor created by Microsoft. It is widely used by developers because it offers many useful features and has a supportive community

One of the standout features of VS Code is its lightweight and fast performance. It offers a seamless editing experience, even when working with large codebases. Its intelligent code completion, suggestions, and error detection through the IntelliSense feature significantly enhance developer productivity.

VS Code supports an extensive array of programming languages and frameworks, thanks to its robust extension ecosystem. Developers can tailor and expand the editor's functionality by installing specific language extensions, debuggers, linkers, and more.

With built-in Git integration, VS Code seamlessly incorporates version control capabilities. This facilitates efficient code repository management and simplifies collaboration with team members.

Furthermore, VS Code provides a highly customizable user interface, allowing developers to personalize their editor with themes, icons, and keyboard shortcuts that suit their preferences and workflows.

In summary, Visual Studio Code offers a powerful and user-friendly development environment for a wide range of programming languages and platforms. Its exceptional performance, extensibility, and strong community support make it a preferred choice for developers seeking a versatile and efficient code editor.

3.1.4 Flow Chart

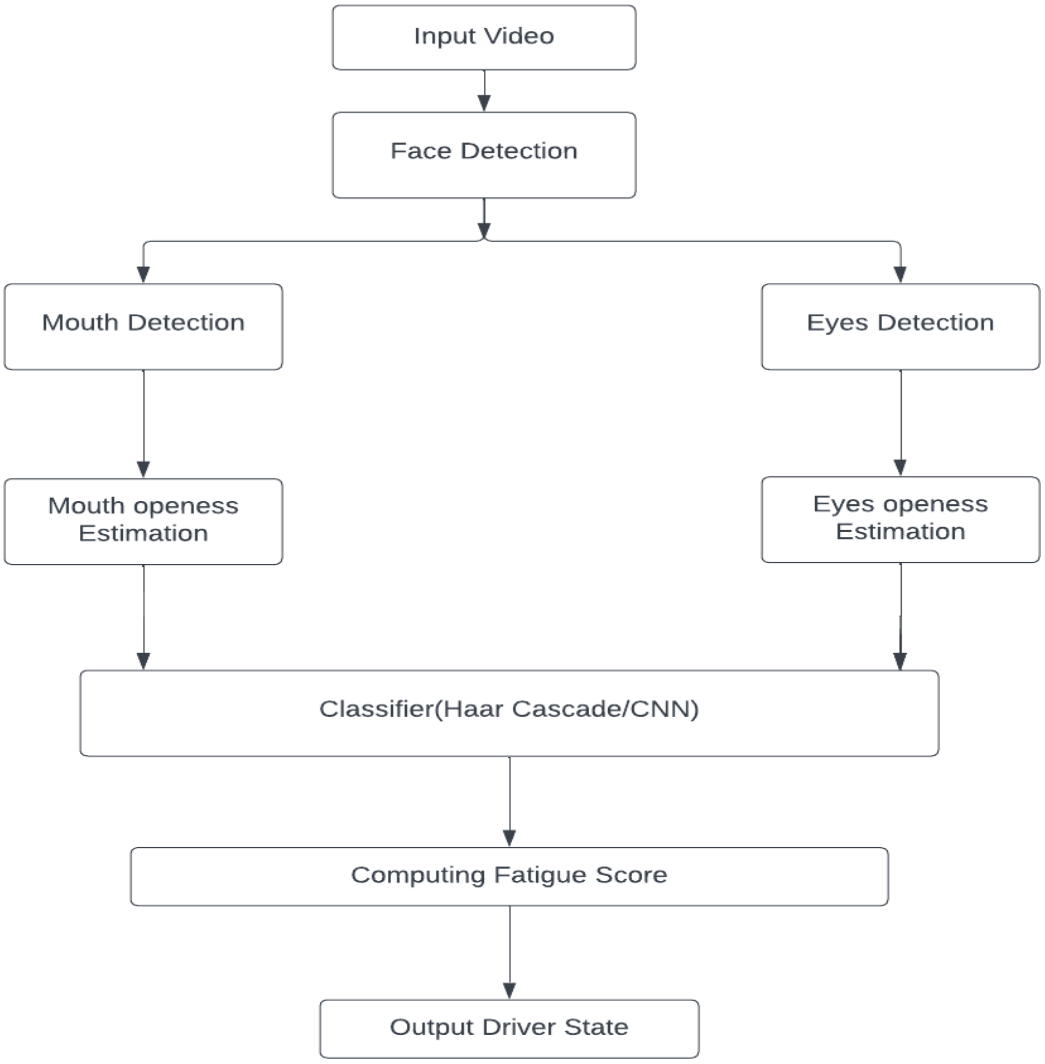

FIGURE 3.1: Flowchart of Driver Drowsiness Detection System.

The flowchart demonstrates the sequential process through which input data is transformed, analyzed, and utilized to detect and respond to driver drowsiness, enhancing overall safety on the road, as shown in Figure 3.1.

3.1.5 Usecase Diagram

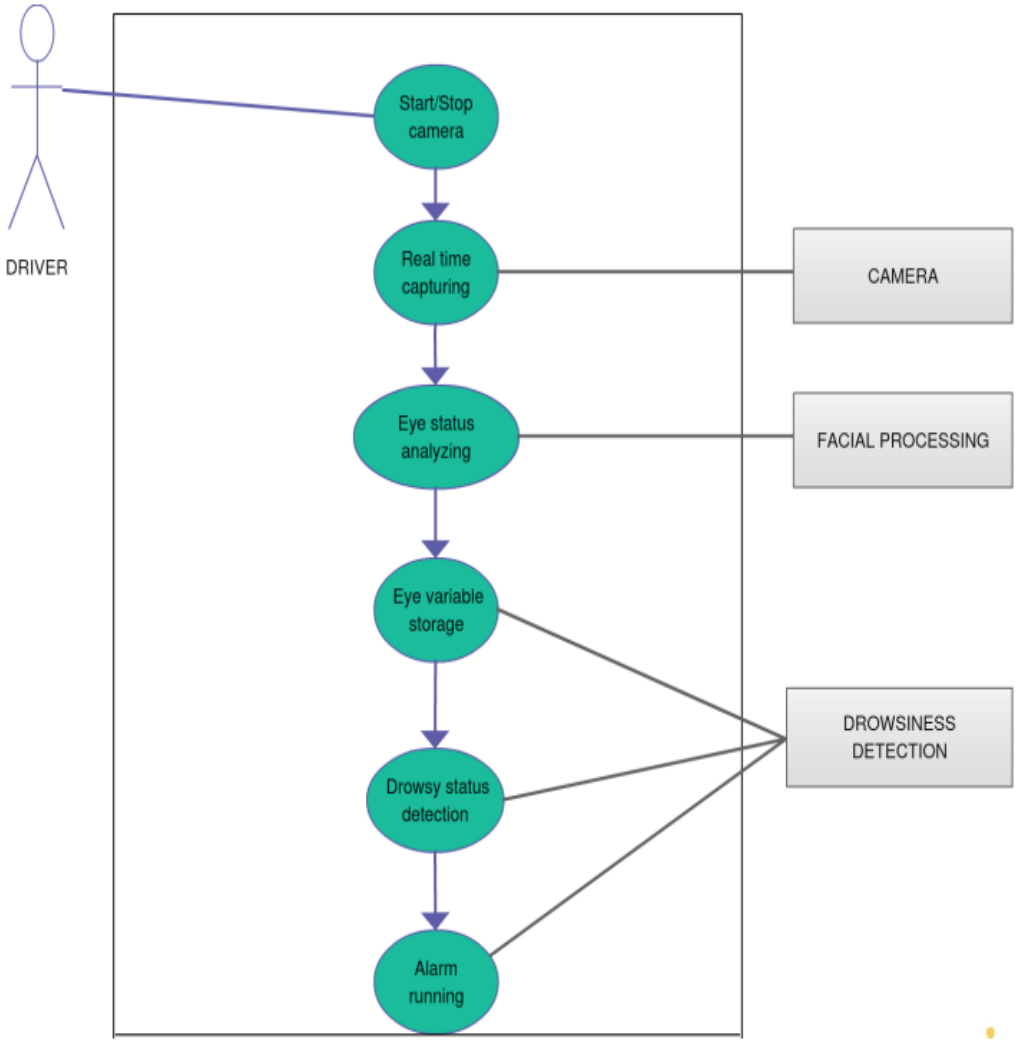

FIGURE 3.2: Usecase Diagram of Driver Drowsiness Detection System.

In Figure 3.2, Usecase Diagram for a driver drowsiness detection system provides an overview of the system's functionalities and the interactions between actors and use cases. It helps to identify the core features and user interactions of the system, facilitating a better perception of its behavior and requirements.

3.2 Models and Algorithms

- **CNN Model**

In a driver drowsiness detection project, a special type of neural network called a Convolutional Neural Network is used to analyze and classify the driver's facial features accurately. CNNs are known for their excellent performance in recognizing images and are widely used in various computer vision tasks, including driver drowsiness detection.

The CNN model is made up of different layers, such as convolutional layers, pooling layers, and fully connected layers. These layers work together to find important features in images and make predictions based on what they've learned. CNNs are great because they can learn on their own to recognize patterns and tell the difference between different things in pictures.

For driver drowsiness detection, the CNN model can be trained using a labeled dataset containing images or video frames of drowsy and alert states. The input to the CNN is typically an image patch or a cropped region of the driver's face. The model learns to extract important features, such as eye closure, eye movement, or facial expressions, that indicate drowsiness.

During training, the CNN model adjusts its internal settings to make its predictions more accurate by comparing them with the correct answers. This process is called backpropagation, and it helps the model learn which features are important for telling the difference between drowsy and alert states.

After the CNN model is trained, it can be used in real-time applications to detect driver drowsiness. The model takes in video or images of the driver's face captured by a camera. It then analyzes the facial features and predicts how drowsy the driver is. If the model detects signs of drowsiness, it can give a warning to the driver to avoid accidents.

By leveraging the power of CNNs, the driver drowsiness detection system can accurately and efficiently analyze facial features and provide real-time monitoring. The CNN model's ability to

learn and recognize complex patterns in facial expressions makes it a valuable tool for detecting drowsiness cues and enhancing road safety.

- **Haar Algorithm**

1. The Haar cascade algorithm is an efficient method for object detection and is particularly well-known for its application in face detection. Here's an overview of how the Haar cascade algorithm works:

2. Haar-like Features: The algorithm uses simple rectangular features called Haar-like features. These features help identify patterns in the image by comparing the brightness of different areas. They look at the difference between the total brightness of white and black regions in the image.

3. Training Phase: The algorithm requires a training phase to learn how to detect the target object. During this phase, two sets of images are used: positive samples (containing instances of the target object) and negative samples (containing images without the target object).

4. To make the calculation of Haar-like features faster, an intermediate representation called the integral image is created. The integral image allows for efficient calculation of the total brightness within any rectangular region of the image. This technique speeds up the calculations by avoiding repetitive computations.

5. Adaboost and Weak Classifiers: The training process employs the AdaBoost algorithm, which selects a subset of the most discriminative Haar-like features. AdaBoost assigns weights to each training sample and trains a series of weak classifiers (simple threshold-based classifiers) based on these features. Weak classifiers make decisions about whether a particular region contains the target object or not.

6. Cascaded Classifier: The Haar cascade structure is formed by combining multiple stages, each comprising several weak classifiers. During training, the weak classifiers are arranged in a cascade, with each subsequent stage having higher discrimination capabilities than the previous one. The cascade is designed to quickly reject non-object regions to reduce computation time.

7. Detection Phase: In the detection phase, the trained cascade classifier is applied to new images or video frames. The algorithm slides a detection window across the image at different scales and positions. At each position, the cascade of weak classifiers is evaluated. If a region passes all stages of the cascade (meaning it satisfies the requirements of all weak classifiers), it is considered a positive detection.

8. Post-processing and Filtering: Detected objects may undergo additional post-processing steps to refine the results. This can include non-maximum suppression to eliminate duplicate or overlapping detections and thresholding to filter out weak or uncertain detections.

   The Haar cascade algorithm offers advantages such as real-time performance and the ability to detect objects with distinct visual features. However, it may struggle with objects that have complex appearances or occlusions. To address these limitations, more advanced object detection algorithms such as Faster R-CNN and YOLO have been developed.

3.3 Adopted Methodology

The Driver Drowsiness Detection System (DDDS) uses the Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR) to identify signs of drowsiness in a driver's face. These ratios are calculated using geometric features obtained from facial landmarks found in live video or image streams.

The working of DDDS using EAR and MAR involves the following steps:

1. The system uses a face detection algorithm, such as Haar cascades or a deep learning-based method like a Convolutional Neural Network (CNN), to identify and locate the driver's face in the video or image stream.

2. Facial Landmark Detection: Once the face is detected, the system identifies specific facial landmarks, including the corners of the eyes and the corners of the mouth. Popular libraries like dlib provide pre-trained models that can accurately detect these landmarks.

3. Calculation of EAR and MAR:

   • **Eye Aspect Ratio (EAR)**: The EAR is calculated by measuring the ratio of distances between specific eye landmarks. It is typically computed using the following formula:

$$EAR = \frac{|P_2 - P_6| + |P_3 - P_5|}{2 \cdot |P_1 - P_4|}$$

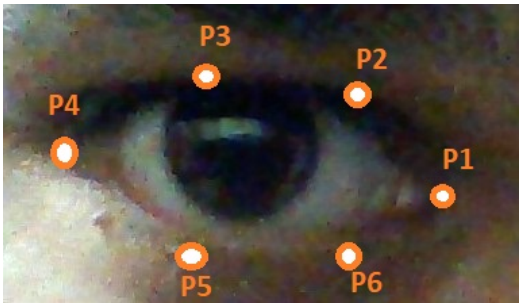

FIGURE 3.3: Open eye.

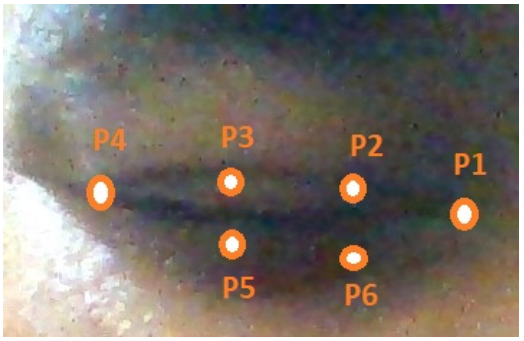

FIGURE 3.4: Close eye.

FIGURE 3.5: Plotted points images open and close eye .

Here, P_1, P_2, P_3, P_4, P_5 and P_6 represent the coordinates of the eye landmarks, such as the inner and outer corners of the eye, as detected by the facial landmark detection algorithm.

• **Mouth Aspect Ratio (MAR)**: The MAR is calculated by measuring the ratio of the distance between the top and bottom lip to the width of the mouth. The formula for

calculating MAR is as follows:

$$MAR = \frac{|P_2 - P_6| + |P_3 - P_5| + |P_1 - P_4|}{3}$$

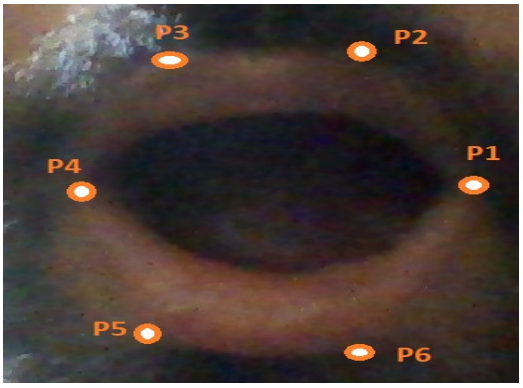

FIGURE 3.6: Open mouth.

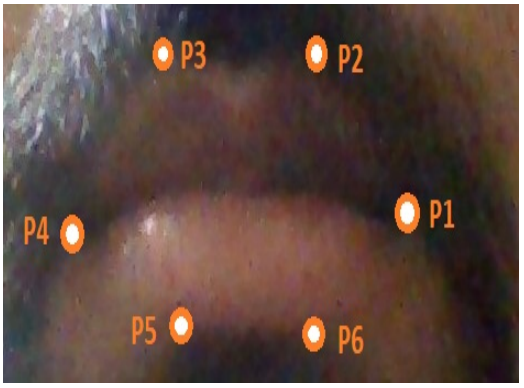

FIGURE 3.7: Close mouth.

FIGURE 3.8: Plotted Points on Open and Close Mouth.

Here, P_1, P_2, P_3, P_4, P_5 and P_6 represent the coordinates of the mouth landmarks, such as the corners of the lips, as detected by the facial landmark detection algorithm.

4. Drowsiness Detection: The calculated EAR and MAR values are compared to predefined threshold values. If the EAR falls below a certain threshold or the MAR exceeds a certain threshold, it indicates potential drowsiness in the driver. These thresholds can be determined through experimentation and tuning based on the specific requirements of the system.

5. Real-time Monitoring: The DDDS constantly monitors the EAR and MAR values in real-time, adapting them according to the driver's facial expressions and any subsequent changes. This allows for timely detection of drowsiness patterns and immediate alert generation.

By monitoring the EAR and MAR, the DDDS can accurately identify drowsiness in a driver's face. These geometric features provide valuable insights into the driver's eye and mouth movements, enabling the system to generate alerts and help prevent accidents caused by drowsiness.

Chapter 4

Experimental Results

The implementation of an OpenCV-based driver drowsiness system has shown promising results in improving road safety by detecting and preventing accidents caused by driver fatigue. Using computer vision techniques, OpenCV allows for real-time analysis of facial features like eye movements and facial expressions to detect signs of drowsiness in drivers. The system constantly monitors the driver's face through a camera, keeping track of specific indicators of fatigue. One of the major advantages of this system is its ability to accurately detect driver drowsiness in real time. OpenCV provides powerful image processing algorithms for the efficient detection of facial cues indicative of drowsiness, such as drooping eyelids or frequent blinking. The system analyzes these patterns and promptly alerts the driver, prompting them to take necessary actions like resting or taking a break.

The accuracy and reliability of the OpenCV-based driver drowsiness system have been extensively evaluated through various experiments. Diverse datasets, involving a wide range of subjects and driving scenarios, have been used to assess the system's performance. The results consistently demonstrate the system's high precision and sensitivity in detecting drowsiness. It effectively distinguishes between normal alertness and drowsiness, minimizing false alarms and ensuring reliable detection. The system also exhibits excellent response time due to OpenCV's

real-time image processing capabilities.

Furthermore, the OpenCV-based driver drowsiness system is adaptable and robust across different environments and driving conditions. It performs well under varying lighting conditions, including daytime, night time, and changing weather conditions. Its versatility makes it suitable for implementation in different vehicles and driving contexts. The practical applications and widespread adoption potential of this system are significant. It can be integrated into existing vehicles or incorporated as a safety feature in new vehicles. Overall, the implementation of an OpenCV-based driver drowsiness system offers a valuable solution for enhancing road safety and mitigating accidents caused by driver fatigue.

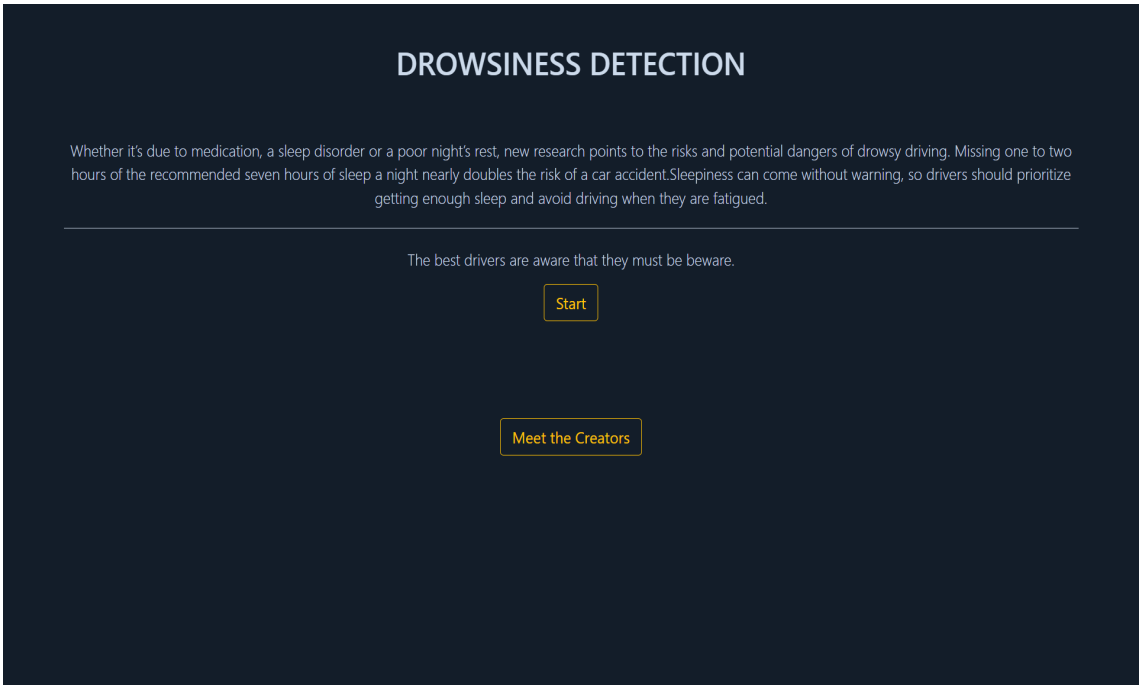

FIGURE 4.1: Homepage of Driver Drowsiness Detection System.

In Figure 4.1, the home page of the project is shown. It gives an introduction to DDDS and provides an interface to start the application.

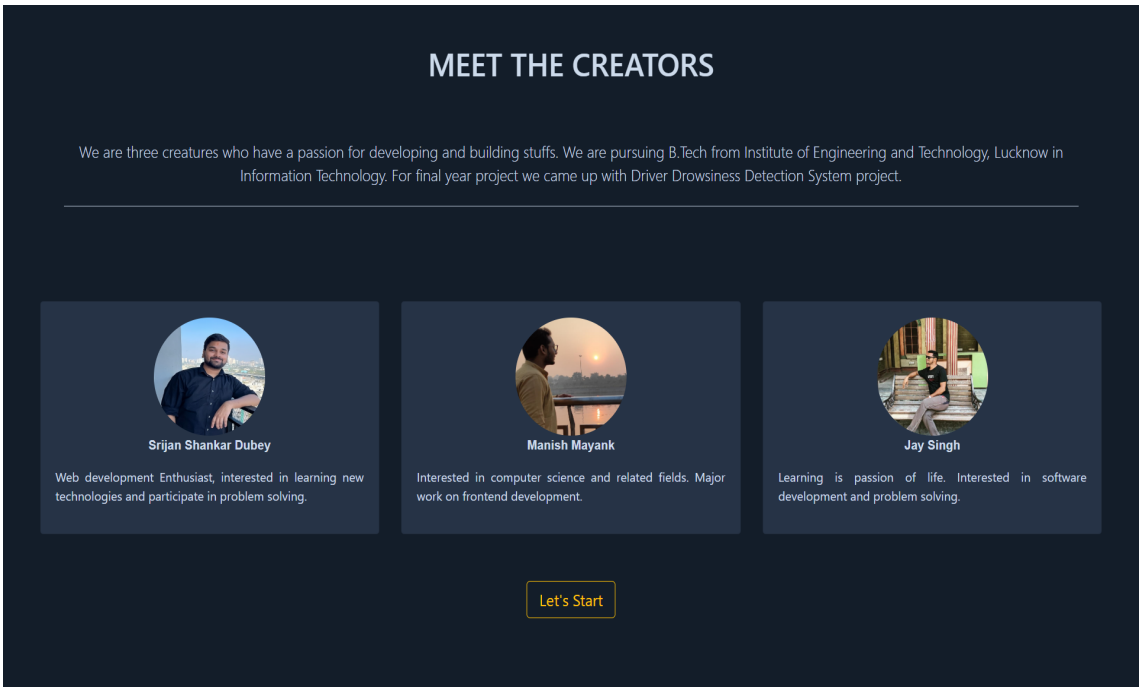

FIGURE 4.2: About Creator of Driver Drowsiness Detection System.

In Figure 4.2, the Meet the Creators page of the project is shown. This page gives a short introduction to the developers who developed this project.

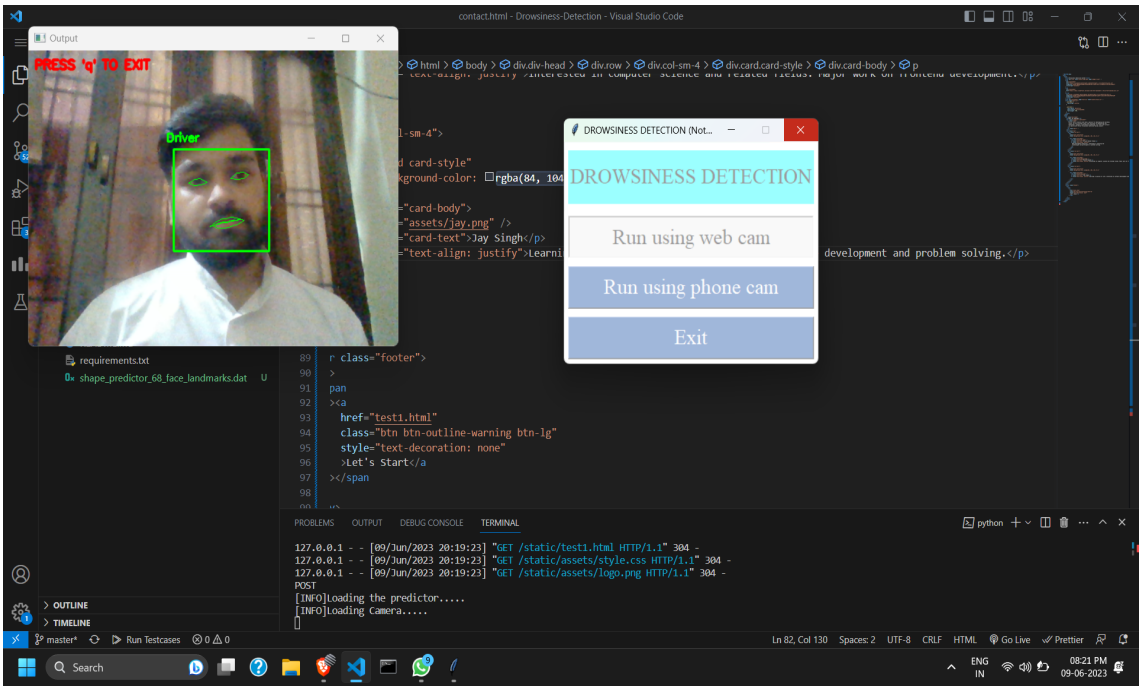

FIGURE 4.3: Active State.

In Figure 4.3, shows the output of the project when a driver is in an active state as detected.

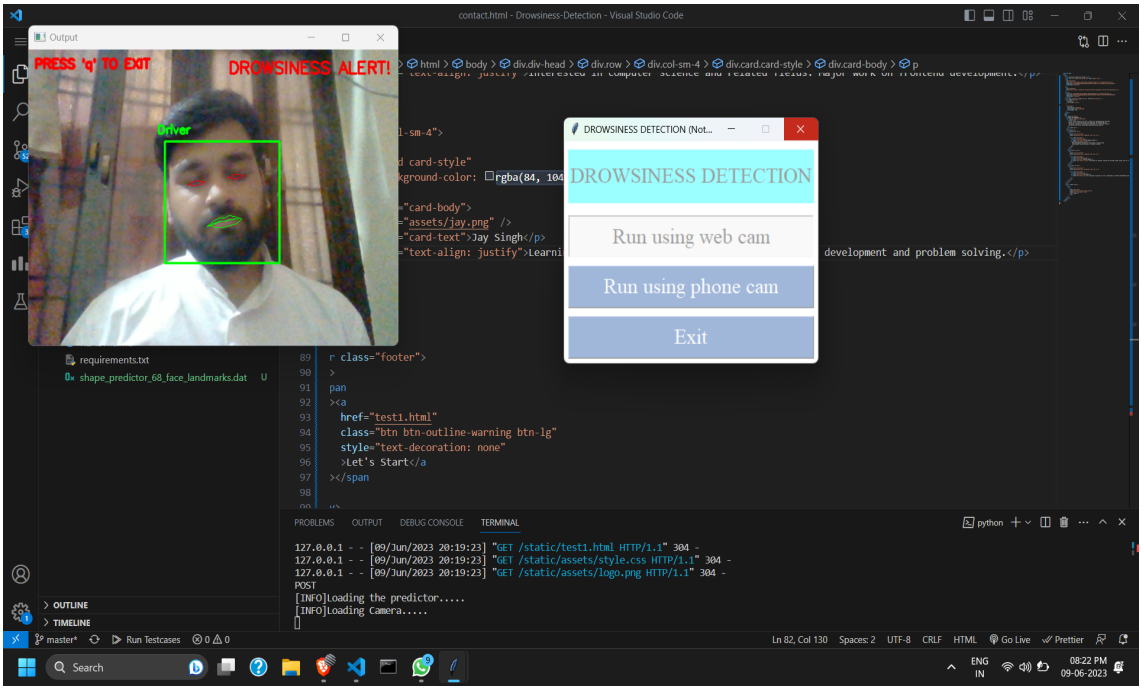

FIGURE 4.4: Drowsy State.

In Figure. 4.4, shows the output of the project when the driver is in a drowsy state as detected due to closed eyes.

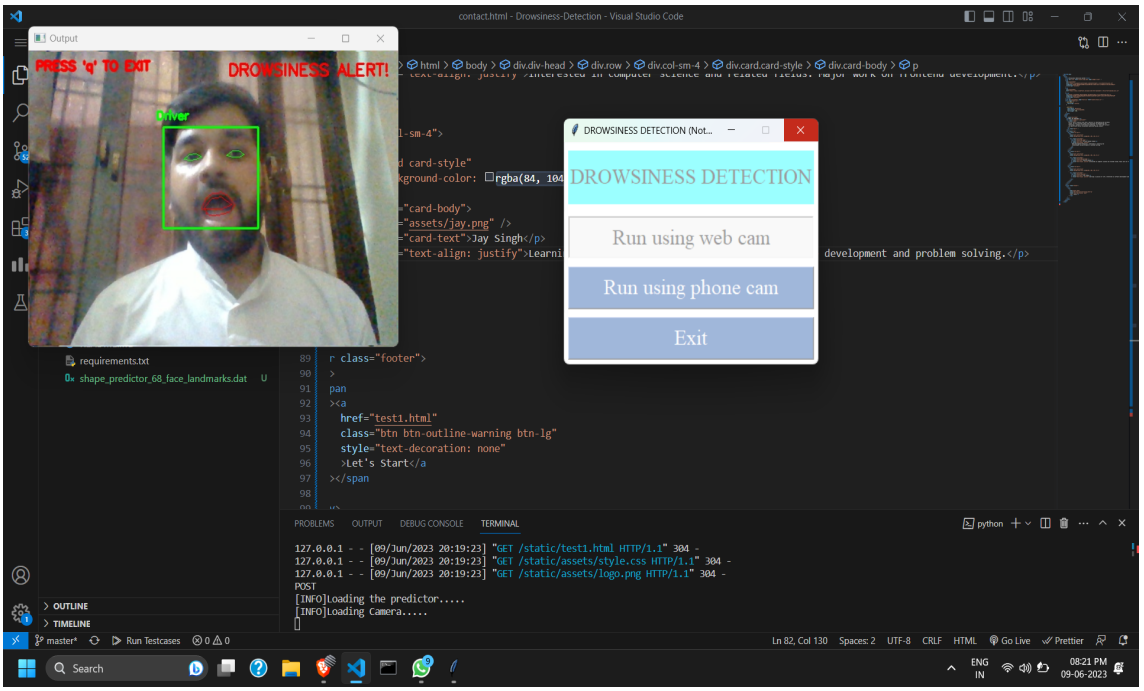

FIGURE 4.5: Drowsy-Yawning State.

In Figure 4.5, shows the output of the project when the driver is in a drowsy state as detected due to yawning.

Chapter 5

Conclusion

The Driver Drowsiness Detection System (DDDS) uses Machine Learning (ML) techniques like OpenCV, Mouth Aspect Ratio (MAR), and Eye Aspect Ratio (EAR), and to effectively detect driver drowsiness with promising results.

The conclusion of the DDDS implementation can be summarized as follows:

1. Accuracy and Performance: The ML-based DDDS demonstrates high accuracy in distinguishing between drowsy and non-drowsy states. The system effectively extracts EAR and MAR features from facial landmarks detected by OpenCV, enabling reliable drowsiness detection. The real-time performance of the system ensures timely alerts and interventions to prevent potential accidents.

2. Robustness: The DDDS using ML techniques has shown robustness in handling variations in facial expressions, lighting conditions, and occlusions. By training the ML model on diverse datasets, the system can generalize well to different individuals and driving scenarios, making it applicable in real-world environments.

3. Non-Intrusive Approach: The ML-based DDDS offers a non-intrusive solution for driver drowsiness detection. It relies on video-based monitoring of facial features, eliminating the need for additional sensors or invasive methods. This enhances user comfort and promotes wider adoption of the system.

4. Potential for Integration: The ML-based DDDS can be easily integrated into existing driver assistance systems or embedded into vehicles. Its compatibility with standard programming languages and frameworks allows for seamless integration, enhancing the overall safety features of vehicles.

5. Limitations and Future Improvements: Despite its success, the ML-based DDDS still faces certain limitations. Factors like extreme lighting conditions, sudden facial movements, or partial occlusions can affect the accuracy of drowsiness detection. Future improvements can focus on addressing these challenges and exploring more sophisticated ML models to enhance the system's performance.

In summary, the implementation of a machine learning-based Drowsiness Detection and Driver Safety (DDDS) system utilizing OpenCV, EAR (Eye Aspect Ratio), and MAR (Mouth Aspect Ratio) provides an efficient, real-time, and non-intrusive solution for detecting driver drowsiness. This system is highly accurate and reliable, and it plays a crucial role in enhancing driver safety and reducing accidents caused by drowsy driving. Ongoing research and development in this field hold great potential for further enhancing the accuracy and reliability of the system, thereby contributing to safer roads and improved driver well-being.

Appendix A

Appendix

The purpose of including appendix in this report is to provide readers with the opportunity to explore specific aspects of the project in more depth or to access relevant materials that are not included in the main body of the report that is Code Snippets.

- Aspect Ratio Calculation

```
# Define the function for calculating the Eye Aspect Ratio(EAR)
from scipy.spatial import distance as dist
def eye_aspect_ratio(eye):
        # Vertical eye landmarks
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])
        # Horizontal eye landmarks
        C = dist.euclidean(eye[0], eye[3])

        # The EAR Equation
        EAR = (A + B) / (2.0 * C)
        return EAR

def mouth_aspect_ratio(mouth):
        A = dist.euclidean(mouth[13], mouth[19])
        B = dist.euclidean(mouth[14], mouth[18])
        C = dist.euclidean(mouth[15], mouth[17])

        MAR = (A + B + C) / 3.0
        return MAR
```

- Detection Code

```
# Import the necessary packages
import datetime as dt
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from EAR_calculator import *
from imutils import face_utils
from imutils.video import VideoStream
import matplotlib.pyplot as plt
import matplotlib.animation as animate
from matplotlib import style
import imutils
import dlib
import time
import argparse
import cv2
from playsound import playsound
from scipy.spatial import distance as dist
import os
import csv
import numpy as np
import pandas as pd
from datetime import datetime

style.use('fivethirtyeight')
# Creating the dataset
def assure_path_exists(path):
    dir = os.path.dirname(path)
    if not os.path.exists(dir):
        os.makedirs(dir)


#all eye  and mouth aspect ratio with time
ear_list=[]
total_ear=[]
mar_list=[]
```

```
total_mar=[]
ts=[]
total_ts=[]
# Construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--shape_predictor", required = True, help = "path to dlib's facial landmark predictor")
ap.add_argument("-r", "--picamera", type = int, default = -1, help = "whether raspberry pi camera shall be used or not")
args = vars(ap.parse_args())

# Declare a constant which will work as the threshold for EAR value, below which it will be regared as a blink
EAR_THRESHOLD = 0.3
# Declare another costant to hold the consecutive number of frames to consider for a blink
CONSECUTIVE_FRAMES = 20
# Another constant which will work as a threshold for MAR value
MAR_THRESHOLD = 14

# Initialize two counters
BLINK_COUNT = 0
FRAME_COUNT = 0

# Now, intialize the dlib's face detector model as 'detector' and the landmark predictor model as 'predictor'
print("[INFO]Loading the predictor.....")
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])

# Grab the indexes of the facial landamarks for the left and right eye respectively
(lstart, lend) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rstart, rend) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
(mstart, mend) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]

# Now start the video stream and allow the camera to warm-up
print("[INFO]Loading Camera.....")
vs = VideoStream(usePiCamera = args["picamera"] > 0).start()
time.sleep(2)

```

```
assure_path_exists("dataset/")
count_sleep = 0
count_yawn = 0


# Now, loop over all the frames and detect the faces
while True:
        # Extract a frame
        frame = vs.read()
        cv2.putText(frame, "PRESS 'q' TO EXIT", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 3)
        # Resize the frame
        frame = imutils.resize(frame, width = 500)
        # Convert the frame to grayscale
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        # Detect faces
        rects = detector(frame, 1)

        # Now loop over all the face detections and apply the predictor
        for (i, rect) in enumerate(rects):
                shape = predictor(gray, rect)
                # Convert it to a (68, 2) size numpy array
                shape = face_utils.shape_to_np(shape)

                # Draw a rectangle over the detected face
                (x, y, w, h) = face_utils.rect_to_bb(rect)
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
                # Put a number
                cv2.putText(frame, "Driver", (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

                leftEye = shape[lstart:lend]
                rightEye = shape[rstart:rend]
                mouth = shape[mstart:mend]
                # Compute the EAR for both the eyes
                leftEAR = eye_aspect_ratio(leftEye)
                rightEAR = eye_aspect_ratio(rightEye)
```

```
106
107                 # Take the average of both the EAR
108                 EAR = (leftEAR + rightEAR) / 2.0
109                 #live datawrite in csv
110                 ear_list.append(EAR)
111                 #print(ear_list)
112
113
114                 ts.append(dt.datetime.now().strftime('%H:%M:%S.%f'))
115                 # Compute the convex hull for both the eyes and then visualize it
116                 leftEyeHull = cv2.convexHull(leftEye)
117                 rightEyeHull = cv2.convexHull(rightEye)
118                 # Draw the contours
119                 cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
120                 cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
121                 cv2.drawContours(frame, [mouth], -1, (0, 255, 0), 1)
122
123                 MAR = mouth_aspect_ratio(mouth)
124                 mar_list.append(MAR/10)
125                 # Check if EAR < EAR_THRESHOLD, if so then it indicates that a blink is taking place
126                 # Thus, count the number of frames for which the eye remains closed
127                 if EAR < EAR_THRESHOLD:
128                         FRAME_COUNT += 1
129
130                         cv2.drawContours(frame, [leftEyeHull], -1, (0, 0, 255), 1)
131                         cv2.drawContours(frame, [rightEyeHull], -1, (0, 0, 255), 1)
132
133                         if FRAME_COUNT >= CONSECUTIVE_FRAMES:
134                                 count_sleep += 1
135                                 # Add the frame to the dataset ar a proof of drowsy driving
136                                 cv2.imwrite("dataset/frame_sleep%d.jpg" % count_sleep, frame)
137                                 playsound('sound files/alarm.mp3')
138                                 cv2.putText(frame, "DROWSINESS ALERT!", (270, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
139                 else:
140                         if FRAME_COUNT >= CONSECUTIVE_FRAMES:
```

```
141                                 playsound('sound files/warning.mp3')
142                         FRAME_COUNT = 0
143                 #cv2.putText(frame, "EAR: {:.2f}".format(EAR), (300, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
144
145                 # Check if the person is yawning
146                 if MAR > MAR_THRESHOLD:
147                         count_yawn += 1
148                         cv2.drawContours(frame, [mouth], -1, (0, 0, 255), 1)
149                         cv2.putText(frame, "DROWSINESS ALERT!", (270, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
150                         # Add the frame to the dataset ar a proof of drowsy driving
151                         cv2.imwrite("dataset/frame_yawn%d.jpg" % count_yawn, frame)
152                         playsound('sound files/alarm.mp3')
153                         playsound('sound files/warning_yawn.mp3')
154         #total data collection for plotting
155         for i in ear_list:
156                 total_ear.append(i)
157         for i in mar_list:
158                 total_mar.append(i)
159         for i in ts:
160                 total_ts.append(i)
161         #display the frame
162         cv2.imshow("Output", frame)
163         key = cv2.waitKey(1) & 0xFF
164
165
166
167         if key == ord('q'):
168                 break
169
170 a = total_ear
171 b=total_mar
172 c = total_ts
173
174 df = pd.DataFrame({"EAR" : a, "MAR":b,"TIME" : c})
175 df.to_csv("op_webcam.csv", index=False)
176 df=pd.read_csv("op_webcam.csv")
```

```
177
178  df.plot(x='TIME',y=['EAR','MAR'])
179  #plt.xticks(rotation=45, ha='right')
180
181  plt.subplots_adjust(bottom=0.30)
182  plt.title('EAR & MAR calculation over time of webcam')
183  plt.ylabel('EAR & MAR')
184  plt.gca().axes.get_xaxis().set_visible(False)
185  plt.show()
186  cv2.destroyAllWindows()
187  vs.stop()
```

References

[1] Knapik M., Cyganek B. Driver's fatigue recognition based on yawn detection in thermal images. Neurocomputing. 2019;338:274–292. doi: 10.1016/j.neucom.2019.02.014.

[2] Kiashari S.E.H., Nahvi A., Bakhoda H., Homayounfard A., Tashakori M. Evaluation of driver drowsiness using respiration analysis by thermal imaging on a driving simulator. Multimed. Tools Appl. 2020;79:17793–17815. doi: 10.1007/s11042-020-08696-x.

[3] Tayab Khan M., Anwar H., Ullah F., Ur Rehman A., Ullah R., Iqbal A., Lee B.-H., Kwak K.S. Smart real-time video surveillance platform for drowsiness detection based on eyelid closure. Wirel. Commun. Mob. Comput. 2019;2019:1–9. doi: 10.1155/2019/2036818.

[4] Ouabida E., Essadike A., Bouzid A. Optical correlator based algorithm for driver drowsiness detection. Optik. 2020;204:164102. doi: 10.1016/j.ijleo.2019.164102.

[5] Maior C.B.S., das Chagas Moura M.J., Santana J.M.M., Lins I.D. Real-time classification for autonomous drowsiness detection using eye aspect ratio. Expert Syst. Appl. 2020;158:113505. doi: 10.1016/j.eswa.2020.113505.

[6] Celecia A., Figueiredo K., Vellasco M., González R. A portable fuzzy driver drowsiness estimation system. Sensors. 2020;20:4093. doi: 10.3390/s20154093.

[7] Ed-Doughmi Y., Idrissi N., Hbali Y. Real-time system for driver fatigue detection based on a recurrent neuronal network. J. Imaging. 2020;6:8. doi: 10.3390/jimaging6030008.

[8] Zhao Z., Zhou N., Zhang L., Yan H., Xu Y., Zhang Z. Driver fatigue detection based on convolutional neural networks using em-cnn. Comput. Intell. Neurosci. 2020;2020:1–11. doi: 10.1155/2020/7251280.

[9] Li G., Lee B.-L., Chung W.-Y. Smartwatch-based wearable EEG system for driver drowsiness detection. IEEE Sens. J. 2015;15:7169–7180. doi: 10.1109/JSEN.2015.2473679.

[10] Budak U., Bajaj V., Akbulut Y., Atila O., Sengur A. An effective hybrid model for EEG-based drowsiness detection. IEEE Sens. J. 2019;19:7624–7631. doi: 10.1109/JSEN.2019.2917850.

[11] Lee H., Lee J., Shin M. Using wearable ECG/PPG sensors for driver drowsiness detection based on distinguishable pattern of recurrence plots. Electronics. 2019;8:192. doi: 10.3390/electronics8020192.

[12] Guede-Fernandez F., Fernandez-Chimeno M., Ramos-Castro J., Garcia-Gonzalez M.A. Driver drowsiness detection based on respiratory signal analysis. IEEE Access. 2019;7:81826–81838. doi: 10.1109/ACCESS.2019.2924481.

[13] Sahayadhas A., Sundaraj K., Murugappan M., Palaniappan R. Physiological signal based detection of driver hypovigilance using higher order spectra. Expert Syst. Appl. 2015;42:8669–8677. doi: 10.1016/j.eswa.2015.07.021.

[14] McDonald A.D., Schwarz C., Lee J.D., Brown T.L. Real-time detection of drowsiness related lane departures using steering wheel angle; Proceedings of the Human Factors and Ergonomics Society Annual Meeting; Boston, MA, USA. 22–26 October 2012; pp. 2201–2205.

[15] Ma J., Murphey Y.L., Zhao H. Real time drowsiness detection based on lateral distance using wavelet transform and neural network; Proceedings of the 2015 IEEE symposium

series on computational intelligence; Cape Town, South Africa. 7–10 December 2015; pp. 411–418.

[16] Li Z., Li S.E., Li R., Cheng B., Shi J. Online detection of driver fatigue using steering wheel angles for real driving conditions. Sensors. 2017;17:495. doi: 10.3390/s17030495.

[17] Arefnezhad S., Samiee S., Eichberger A., Nahvi A. Driver drowsiness detection based on steering wheel data applying adaptive neuro-fuzzy feature selection. Sensors. 2019;19:943. doi: 10.3390/s19040943.

[18] Chai M. Drowsiness monitoring based on steering wheel status. Transp. Res. Part D Transp. Environ. 2019;66:95–103. doi: 10.1016/j.trd.2018.07.007.

Jay Singh

ORIGINALITY REPORT

15%	11%	7%	9%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Liverpool John Moores University Student Paper	3%
2	Submitted to Higher Education Commission Pakistan Student Paper	1%
3	www.ijraset.com Internet Source	1%
4	Submitted to Kamla Nehru Institute of Technology Sultanpur Student Paper	1%
5	www.mdpi.com Internet Source	1%
6	www.slideshare.net Internet Source	1%
7	Submitted to Pathfinder Enterprises Student Paper	1%
8	www.researchgate.net Internet Source	1%

9	Submitted to Coventry University Student Paper	<1 %
10	da.overleaf.com Internet Source	<1 %
11	Submitted to University of Debrecen Student Paper	<1 %
12	T. Srilakshmi, Harshavardhan Reddy, Yaswanthi Potluri, Lakshmi Ramani Burra, Mohana Vamsi Thota, Rishita Gundimeda. "Automated Driver Drowsiness Detection System using Computer Vision and Machine Learning", 2023 International Conference on Sustainable Computing and Data Communication Systems (ICSCDS), 2023 Publication	<1 %
13	Submitted to GLA University Student Paper	<1 %
14	ethesis.nitrkl.ac.in Internet Source	<1 %
15	Yaman Albadawi, Maen Takruri, Mohammed Awad. "A Review of Recent Developments in Driver Drowsiness Detection Systems", Sensors, 2022 Publication	<1 %
16	Submitted to Dhirubhai Ambani Institute of Information and Communication	<1 %

	Student Paper	
17	www.researchsquare.com Internet Source	<1 %
18	mafiadoc.com Internet Source	<1 %
19	sjcjycl.cn Internet Source	<1 %
20	Studies in Computational Intelligence, 2009. Publication	<1 %
21	www.irjmets.com Internet Source	<1 %
22	Duy Tran, Jianhao Du, Weihua Sheng, Denis Osipychev, Yuge Sun, He Bai. "A Human-Vehicle Collaborative Driving Framework for Driver Assistance", IEEE Transactions on Intelligent Transportation Systems, 2019 Publication	<1 %
23	Submitted to Uttar Pradesh Technical University Student Paper	<1 %
24	Submitted to NIIT University Student Paper	<1 %
25	Submitted to University of Greenwich Student Paper	<1 %

26	Guan-Yuan Wang. "Integrating Street Views, Satellite Imageries and Remote Sensing Data Into Economics and the Social Sciences", Social Science Computer Review, 2023 Publication	<1 %
27	www.coursehero.com Internet Source	<1 %
28	Submitted to University of Bedfordshire Student Paper	<1 %
29	escholarship.mcgill.ca Internet Source	<1 %
30	Submitted to Gitam University Student Paper	<1 %
31	iieta.org Internet Source	<1 %
32	philpapers.org Internet Source	<1 %
33	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
34	medium.com Internet Source	<1 %
35	people.idsia.ch Internet Source	<1 %
	www.eurekalert.org	

36	Internet Source	<1%
37	"Proceedings of 3rd International Conference on Computing Informatics and Networks", Springer Science and Business Media LLC, 2021 Publication	<1%
38	"Proceedings of the 12th National Technical Seminar on Unmanned System Technology 2020", Springer Science and Business Media LLC, 2022 Publication	<1%
39	Muhammad Tayab Khan, Hafeez Anwar, Farman Ullah, Ata Ur Rehman, Rehmat Ullah, Asif Iqbal, Bok-Hee Lee, Kyung Sup Kwak. "Smart Real-Time Video Surveillance Platform for Drowsiness Detection Based on Eyelid Closure", Wireless Communications and Mobile Computing, 2019 Publication	<1%
40	curve.carleton.ca Internet Source	<1%
41	idr.mnit.ac.in Internet Source	<1%
42	link.springer.com Internet Source	<1%

43	"Proceedings of China SAE Congress 2021: Selected Papers", Springer Science and Business Media LLC, 2023 Publication	<1 %
44	Guang-Yuan Zhang, Bo Cheng, Rui-Jia Feng, Jia-Wen Li. "Real-time driver eye detection method using Support Vector Machine with Hu invariant moments", 2008 International Conference on Machine Learning and Cybernetics, 2008 Publication	<1 %
45	João Pedro Fontes, João Nuno Centeno Raimundo, Luís Gonzaga Mendes Magalhães, Miguel Angel Guevara Lopez. "3D CNN-based Quantitative Imaging Biomarkers for Accurate Phenotyping of Luminal A Breast Cancer in Magnetic Resonance Imaging", Institute of Electrical and Electronics Engineers (IEEE), 2023 Publication	<1 %
46	diglib.tugraz.at Internet Source	<1 %
47	oparu.uni-ulm.de Internet Source	<1 %
48	repository.uwl.ac.uk Internet Source	<1 %
	www.chalapathiengg.ac.in	

49	Internet Source	<1%
50	www.jetir.org Internet Source	<1%
51	McDonald, A. D., J. D. Lee, C. Schwarz, and T. L. Brown. "Steering in a Random Forest: Ensemble Learning for Detecting Drowsiness-Related Lane Departures", Human Factors The Journal of the Human Factors and Ergonomics Society, 2013. Publication	<1%
52	Mukesh Kumar Kamti, Rauf Iqbal. "Evolution of Driver Fatigue Detection Techniques—A Review From 2007 to 2021", Transportation Research Record: Journal of the Transportation Research Board, 2022 Publication	<1%
53	"Computer Networks and Inventive Communication Technologies", Springer Science and Business Media LLC, 2021 Publication	<1%
54	Yaman Albadawi, Aneesa AlRedhaei, Maen Takruri. "Real-Time Machine Learning-Based Driver Drowsiness Detection Using Visual Features", Journal of Imaging, 2023 Publication	<1%