DRIVER DROWSINESS DETECTION USING MACHINE LEARNING

A Report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

In

Electronics and Communication Engineering

Under the Supervision of

Ms. HIMANI

By

TANMAY -: 0115007321

BHUPENDER SINGH -: 0315007321

SATYAM -: 07715002820

MAHARAJA SURAJMAL INSTITUTE OF TECHNOLOGY

C-4, Janakpuri, New Delhi-58

Affiliated to Guru Gobind Singh Indraprastha University, Delhi

Dec, 2023

DECLARATION

We, students of B.Tech (Electronics & Communication Engineering) hereby declare that the project work done on "DRIVER DROWSINESS DETECTION" submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi in partial fulfilment of the requirement for the award of degree of Bachelor of Technology comprises of our original work and has not been submitted anywhere else for any other degree to the best of our knowledge.

TANMAY -: 0115007321

BHUPENDER SINGH -: 0315007321

SATYAM -: 07715002820

CERTIFICATE

This is to certify that the project work done on "DRIVER DROWSINESS DETECTION" submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi by "TANMAY, BHUPENDER SINGH and SATYAM" in partial fulfillment of the requirement for the award of degree of Bachelor of Technology, is a bonafide work carried out by him/her under my supervision and guidance. This project work comprises of original work and has not been submitted anywhere else for any other degree to the best of my knowledge.

Signature of Supervisor

Ms. Himani

Signature of HOD

Dr.Neeru Rathee

ACKNOWLEDGEMENT

Team effort together with precious words of encouragement and guidance makes daunting tasks achievable. It is a pleasure to acknowledge the direct and implied help we have received at various stages in the task of developing the project. It would not have been possible to develop such a project without the furtherance on part of numerous individuals. We find it impossible to express our thanks to each one of them in words, for it seems too trivial when compare to the profound encouragement that they extended to us.

We are grateful to Dr.Neeru Rathee, HOD, ECE, for having given us opportunity to do this project, which was of great interest to us.

Our sincere thanks to Ms. Himani, for believing in us and providing motivation all through. Without her guidance this project would not be such a success.

At last we thank the almighty, who had given the strength to complete this project on time.

Finally we would like to thank our parents, all friends, and well wishers for their valuable help and encouragement throughout the project.

(Signature1)

TANMAY -: 0115007321

(Signature2)

BHUPENDER SINGH -: 0315007321

(Signature3)

SATYAM -: 07715002820

Contents

CHAPTER 1
INTRODUCTION

1.1 Overview

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. The drowsiness detection system is capable of detecting drowsiness quickly. The driver behaviors are noticed in many conditions such as wearing spectacles and also in the dark condition inside the vehicle. The system is capable of detecting the drowsiness condition within the duration of more than two seconds. After the detection of abnormal behaviors, it is alerted to the driver through alarms and the parking lights will be on that will stop the vehicle which reduces the accidents due to drowsiness of the driver. A deep learning Architecture detects the face and eyes, based on the status of the eyes. If the eyes are closed more than usual time, it generates an alarm, intimating the driver.

Figure 1: Examples of Fatigue & Drowsiness Condition

Neglecting our duties towards safer travel has enabled hundreds of thousands of tragedies to get associated with this wonderful invention every year. In order to monitor and prevent a destructive outcome from such negligence, many researchers have written research papers on driver drowsiness detection systems. But at times, some of the points and observations made by the system are not accurate enough. Hence, to provide data and another perspective on the problem at hand, in order to improve their implementations and to further optimize the solution, this project has been done.

The development of technologies for detecting or preventing drowsiness while driving is a major challenge in the field of accident-avoidance system. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its affects.

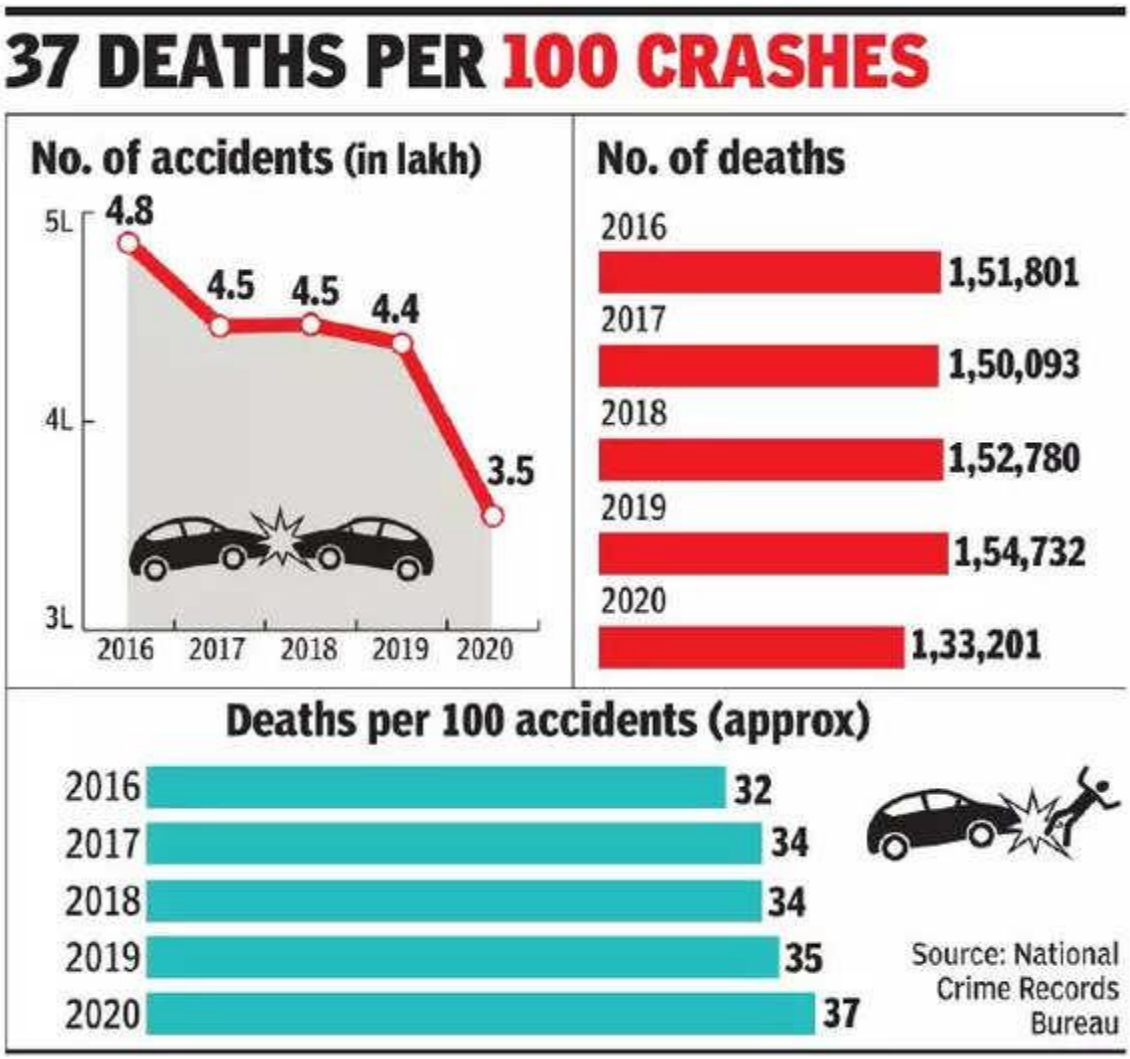


Figure 2: Statistic of Road Accident

1.2 Background of Study

Each year, there is an increase in road accidents cases involving cars and heavy vehicles like buses, lorries, and trucks in India. Drowsiness and fatigue condition is one of the prime factors contributing to road accidents. Driving in this condition may result terrible causes since it affects the driver's judgment and concentration. Falling asleep on the wheel can be avoid if the drivers take efforts such as getting enough sleep before driving, taking caffeine or stop for a while to rest when the signs of fatigue and drowsiness appears.

However, in many cases, drivers refuse to take one of these steps even when they know that they are suffering from fatigue, and will continue driving. Therefore, detecting drowsiness is important as one of the steps to prevent the road accidents. This project proposed that yawning and eyes detection is the obvious signs of fatigue and drowsiness.

1.3 Problem statement

Current drowsiness detection systems monitoring the driver's condition requires complex computation and expensive equipment, not comfortable to wear during driving and is not suitable for driving conditions; for example, Electroencephalography (EEG) and Electrocardiography (ECG), i. e. detecting the brain frequency and measuring the rhythm of heart, respectively.

A drowsiness detection system which uses a camera placed in front of the driver is more suitable to be use but the physical signs that will indicate drowsiness need to be located first in order to come up with a drowsiness detection algorithm that is reliable and accurate. Lighting intensity and while the driver tilts their face left or right are the problems occur during detection of eyes and mouth region.

Therefore, this project aims to analyze all the previous research and method, hence propose a method to detect drowsiness by using video or webcam. It analyzes the video images that have been recorded and come up with a system that can analyze each frame of the video.

1.4 Scope of Study

In this project, we will focus on these following procedures:

I. Basic concept of drowsiness detection system.
II. Determine the drowsiness from these parameters.
    - Eye blink
    - Area of the pupils detected at eyes.
III. Data collection and measurement.
IV. Integration of the methods chosen.
V. Software development and testing.

CHAPTER 2

Fundamentals of the Technology

2.1 Overview

Driver drowsiness detection technology has emerged as a critical component in enhancing road safety by addressing the risks associated with drowsy driving. Historically, the development of these systems has evolved in response to the alarming rates of accidents caused by driver fatigue. Key milestones in this evolution include advancements in sensor technologies, artificial intelligence, and signal processing. The basic principles of driver drowsiness detection involve the analysis of physiological and behavioral indicators that signify a driver's level of alertness. Physiological signals, such as eye movement patterns and heart rate variability, are often monitored to gauge the driver's state. Additionally, behavioral cues like steering patterns and lane deviations are analyzed to assess the risk of drowsiness. Understanding these fundamental principles is crucial for the effective design and implementation of driver drowsiness detection systems, contributing to the overall improvement of road safety.

2.2 Tools and Technology used

In the realm of driver drowsiness detection, various technologies are employed to monitor and analyze the physiological and behavioral signals indicative of a driver's alertness. The following technologies are commonly used.

I. **Convolutional Neural Networks (CNNs):**

Convolutional Neural Networks (CNNs) represent a pivotal advancement in the field of deep learning, particularly in the realm of computer vision. Deployed extensively for image and video analysis, CNNs have demonstrated remarkable efficacy in tasks such as facial recognition, object detection, and, notably, driver drowsiness detection. The strength of CNNs lies in their ability to automatically learn hierarchical features from input data through convolutional and pooling layers. In the context of drowsiness detection, a CNN can be trained on diverse datasets containing images or video frames of drivers exhibiting various levels of alertness. The network learns to discern subtle patterns, such as changes in facial expressions or eye movements, indicative of drowsiness.

The application of non-linear activation functions, such as ReLU, and pooling layers enables the network to capture intricate features while maintaining computational efficiency. Moreover, the concept of transfer learning, leveraging pre-trained CNN models on extensive datasets, has proven valuable in fine-tuning networks for specific tasks like identifying signs

of drowsiness. CNNs, with their capacity for learning intricate spatial hierarchies, stand as a robust technological foundation in the development of accurate and real-time driver drowsiness detection systems, contributing significantly to road safety.

II. Dlib:

Dlib, a powerful C++ library, is renowned for its versatility in addressing a variety of machine learning and computer vision applications. Developed by Davis King, it offers a comprehensive set of tools, encompassing image processing, machine learning algorithms, and facial recognition capabilities. One of the key features of Dlib is its facial landmarks detection, which involves identifying and locating specific points on a face. Dlib employs a shape prediction model trained on a diverse range of facial images, enabling precise identification of facial features such as eyes, nose, and mouth.

In the context of driver drowsiness detection, Dlib's facial landmarks detection capabilities can be leveraged to analyze key facial features associated with drowsiness, such as eye closure and head pose changes. Additionally, its support for deep learning allows for the integration of convolutional neural networks (CNNs) and other advanced models for more intricate analyses.

Dlib stands as a robust and versatile tool in the toolkit of computer vision enthusiasts and professionals, offering a seamless blend of machine learning and image processing capabilities. Its open-source nature and active community support make it a valuable asset for our applications, driver drowsiness detection systems.

Figure 2: Facial Landmarks

III. OpenCV:

OpenCV, an acronym for Open Source Computer Vision Library, stands as a cornerstone in the realm of computer vision and machine learning. This open-source software library, developed in C++ and supported by interfaces for various programming languages, provides an extensive suite of tools for real-time image and video processing. In the context of driver drowsiness detection, OpenCV proves invaluable due to its robust set of functionalities. Noteworthy features include facial recognition and landmark detection, enabling the identification of key facial features crucial for assessing drowsiness indicators. The library's object tracking capabilities further facilitate continuous monitoring of specific regions within video streams, contributing to the nuanced analysis of driver behavior. OpenCV seamlessly integrates with machine learning frameworks, allowing developers to incorporate advanced models for drowsiness detection. Its optimization for real-time applications ensures prompt responses, a critical aspect in the context of driver monitoring. Additionally, OpenCV's cross-platform compatibility and a supportive community provide a versatile and accessible environment for researchers and developers striving to enhance road safety through intelligent transportation systems, including the implementation of effective driver drowsiness detection solutions.

IV. Imutils:

Imutils, a Python-based image processing utility library, significantly enhances the capabilities of developers working with computer vision applications. Serving as a valuable extension to OpenCV, imutils simplifies common tasks and optimizes workflows. Its utility functions, such as `resize` and `rotate`, streamline image preprocessing by providing easy-to-use tools for adjusting dimensions and orientation. The library's aspect ratio-preserving resizing ensures that facial features crucial for drowsiness detection maintain accurate representation. Imutils seamlessly integrates with OpenCV, offering a cohesive environment for efficient image processing pipelines.

Particularly useful for object tracking, imutils includes functions for computing centroids, contributing to the precise analysis of facial features indicative of driver drowsiness. The library's support for threaded video streams facilitates real-time processing, aligning with the requirements of applications like driver monitoring systems. Imutils, with its focus on

simplicity and compatibility, significantly contributes to the development of effective and readable code for complex computer vision tasks, making it a valuable asset in the toolbox of developers and researchers alike.

V. Numpy:

Numpy, a fundamental library in the Python ecosystem, plays a pivotal role in numerical computing and data manipulation. Specifically designed for efficient handling of large, multi-dimensional arrays and matrices, Numpy provides an essential foundation for mathematical operations crucial in various scientific and machine learning applications. In the context of driver drowsiness detection, Numpy's array manipulation capabilities are invaluable for handling image data, which is often represented as arrays. Its powerful functions for mathematical operations, statistical analysis, and linear algebra provide a solid groundwork for implementing algorithms involved in image processing and machine learning tasks. For instance, when working with image frames from a driver monitoring system, Numpy facilitates tasks such as pixel-level manipulation, matrix transformations, and statistical analysis of image data.

Its efficiency in numerical operations, combined with seamless integration with other libraries like OpenCV, contributes to the development of robust and high-performance algorithms for driver drowsiness detection systems. The versatility and performance optimization offered by Numpy make it an indispensable tool for researchers and developers working on advanced computer vision applications.

VI. MideaPipe:

MediaPipe, an open-source framework developed by Google, has emerged as a powerful tool for machine learning solutions in the domain of multimedia applications. Offering a versatile and cross-platform environment, MediaPipe simplifies the development of applications that involve real-time video and camera input analysis. Notably, its hand tracking module stands out, providing robust and accurate tracking of hand movements, enabling applications to interpret gestures in real time. The framework also encompasses modules for face detection, facial landmark recognition, pose estimation, and holistic

tracking, allowing developers to build applications that understand and respond to various aspects of human movement.

MediaPipe's modular architecture and integration with TensorFlow Lite offer customization and extensibility, making it suitable for a wide range of applications, from fitness tracking to augmented reality experiences. Its optimization for real-time processing ensures low-latency analysis of video input, and being open-source, it fosters collaboration and community contributions, supported by comprehensive documentation. As a result, MediaPipe has become a valuable asset for developers working on innovative multimedia applications that require efficient and accurate machine learning solutions.

VII. Pygame:

Python is the most popular programming language or nothing wrong to say that it is the next-generation programming language. In every emerging field in computer science, Python makes its presence actively. Python has vast libraries for various fields such as Machine Learning (Numpy, Pandas, Matplotlib), Artificial intelligence (Pytorch, TensorFlow), and Game development (Pygame,Pyglet)

CHAPTER 3

LITERATURE REVIEW

3.1 Introduction

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

There are some researches on Driver's Drowsiness Detection, for the proper outcome of the subject and usage of it. The researches use different approaches for the application and the requirement processes.

I. **Drowsiness Detection Based on Driver Temporal Behaviour** (31-March 2021, F. Faraji, F. Lotfi, J. Khorramdel, A. Najafi, A. Ghaffari). In this research YOLOv3 CNN is applied as a pretrained network, which is proved to be utilized as a powerful means for object detection. LSTM (Long-Short Term Memory) neural network is employed to learn driver temporal behaviours including yawning and blinking time period as well as sequence classification. One of the main factors of the temporal behaviour is that the driver becomes gradually diverted from the road and road traffic. Hence detection is not always accurate.

II. **Driver Drowsiness Detection** (21-09-2020, V B Navya Kiran, Raksha R, Anisoor Rahman, Varsha K N, Dr. Nagamani N P).The detection system includes the processes of face image extraction, yawning tendency, blink of eyes detection, eye area extraction etc. This paper provides a comparative study on papers related to driver drowsiness detection and alert system. It is designed in such a way where system does not continuously record or retain any data.

III. **Driver Drowsiness Detection System** (12 December 2019, Pratyush Agarwal) This paper analyses the method used to detect driver's drowsiness and proposes the results & solutions on the limited implementation of the various techniques that are used in such embedded systems. Driver Drowsiness Detection System(May 2019, Muhammad Faique Shakeel and Nabita Bajwa). In this article, they propose a novel deep learning methodology based on Convolutional Neural Networks (CNN) to tackle the Project. In the trained model, we only use 250 low-light images.

IV. **A Survey on State-of-The-Art-Driver Drowsiness Detection Techniques** (1 December 2020, FHikmat Ullah Khan).The detection system includes the processes of face image extraction, yawning tendency, blink of eyes detection, eye area extraction etc. The percentage of the eyelid closure of the algorithms over the pupil over time is relatively very low.

V. **A Survey on Driver Drowsiness Detection Techniques** (01 December 2020, Reshma , Ishwarya, and Sai Vennala). Drowsiness Detection System, the detection system includes the processes of face image extraction, yawning tendency, blink of eyes etc.

3.2 Existing system

Vision based techniques:

NO EYE DETECTION – most critical sign of drowsiness Yawning and nodding are not always practical. Varies from person to person – some may not yawn when they are sleepy sometimes.

Physiological sensors:

More accurate solutions Needs to be attached to the human body

- If driver forgets to wear it?

- May hesitate to wear

Easy wear biosensors are developed.

- Head bands, headphones (Signal-Channel EEG)

- Portal glasses for EOG

- Hand bands for PPG

3.3 Drowsiness and Fatigue

Drowsiness is where a person is in the middle of awake and sleepy state. This situation leads the driver to not giving full attention to their driving. Therefore, the vehicle can no longer be controlled due to the driver being in a semi-conscious state. According to **Mental fatigue** is a

factor of drowsiness and it caused the person who experiences to not be able to perform because it decreases the efficiency of the brain to respond towards sudden events.

3.4 Electroencephalography (EEG) for Drowsiness Detection

Electroencephalography (EEG) is a method that measures the brain electrical activity. As shown in Figure 3, it can be used to measure the heartbeat, eye blink and even major physical movement such as head movement.

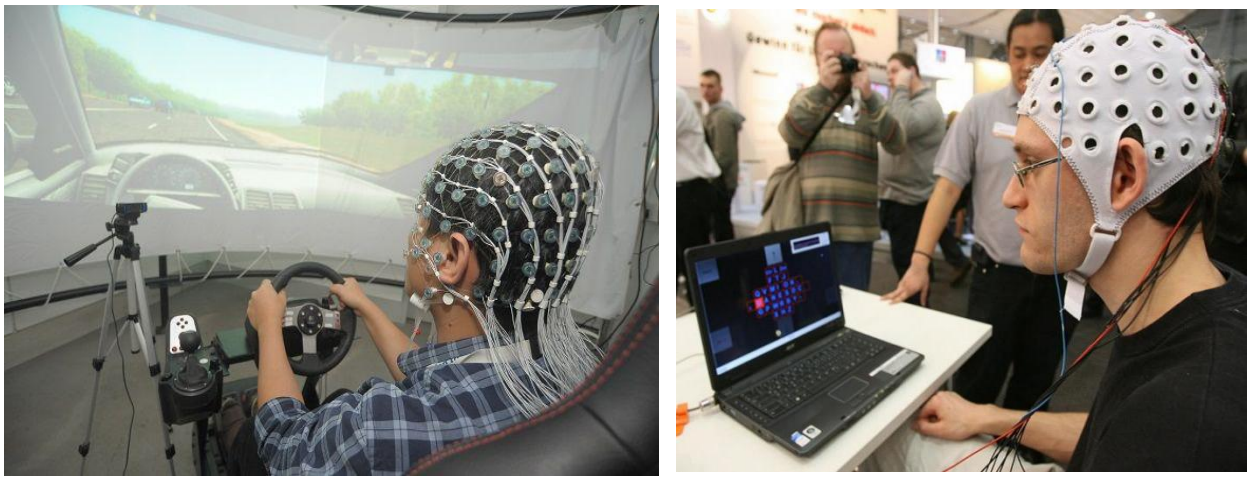


Figure 3: Examples of EEG Data Collecting

It can be used on human or animal as subjects to get the brain activity. It uses a special hardware that place sensors around the top of the head area to sense any electrical brain activity. It has been implemented by the previous researcher to detect drowsiness signs, the EEG method is best to be applied for drowsiness and fatigue detection.

In the method, EEG have four types of frequency components that can be analyzed, i.e. alpha (α), beta (β), theta (θ) and delta (δ).

When the power is increased in alpha (α) and delta (δ) frequency bands, it shows that the driver is facing fatigue and drowsiness.

The disadvantages of this method are, it is very sensitive to noise around the sensors. For example, when the person is doing the EEG experiment, the surrounding area must be completely silent. The noise will interfere with the sensors that detect the brain activity. Another disadvantage of this method is that even if the result might be accurate, it is not suitable to use for real driving

application. Imagine when a person is driving and he is wearing something on his head with full of wires and when the driver moves their head, the wire may strip off from their place. Even though it is not convenient to be used for real-time driving but for experiment purposes and data collection, it is one of the best methods so far.

CHAPTER 4

METHODOLOGY

4.1 Introduction

Drowsiness can be detected by using face area detection. The methods to detect drowsiness within face area are vary due to drowsiness sign are more visible and clearer to be detected at face area. From the face area, we can detect the eyes' location. From eyes detection, there are four types of eyelid movement that can be used for drowsiness detection. They are complete open, complete close, and in the middle where the eyes are from open to close and vice versa.

The algorithm processes the images captured in grey-scale method; where the color from the images is then transformed into black and white. Working with black and white images is easier because only two parameters have to be measured. Then it performs the edge detection to detect the edges of eyes so that the value of eyelid area can be calculated.

Figure 4: Examples of Eyelid Movement

The problem occurring with this method is that the size area of eye might vary from one person to another. Someone may have small eyes and looks like it is sleepy but some are not. Other than that, if the person is wearing glasses, there is obstacle to detect eye region. The images that being captured must be in certain range from the camera because when the distance is far from the camera, the images are blurred.

I. **Facial Landmark Detection:**

The project begins by employing Mediapipe's facial landmark detection model and Dlib python library to accurately identify key facial landmarks, such as the eyes, nose, and mouth, from a video feed captured by a camera installed inside the vehicle.

II. **Eye Blink Analysis:**

The system then tracks the driver's eye movements, specifically monitoring the frequency and duration of eye blinks. An algorithm is implemented to distinguish between normal blinking patterns and prolonged or frequent blinks, which may be indicative of drowsiness.

III. **Yawn Detection:**

Another crucial aspect of drowsiness detection is the recognition of yawns. The project utilizes facial landmark data to detect when the driver yawns, as yawning can be a strong indicator of fatigue.

Figure 5: Yawning detection

4.2 Eye Aspect Ratio

We will use the simple yet robust Eye Aspect Ratio (EAR) technique introduced in Real-Time Eye Blink Detection using Facial Landmarks paper.

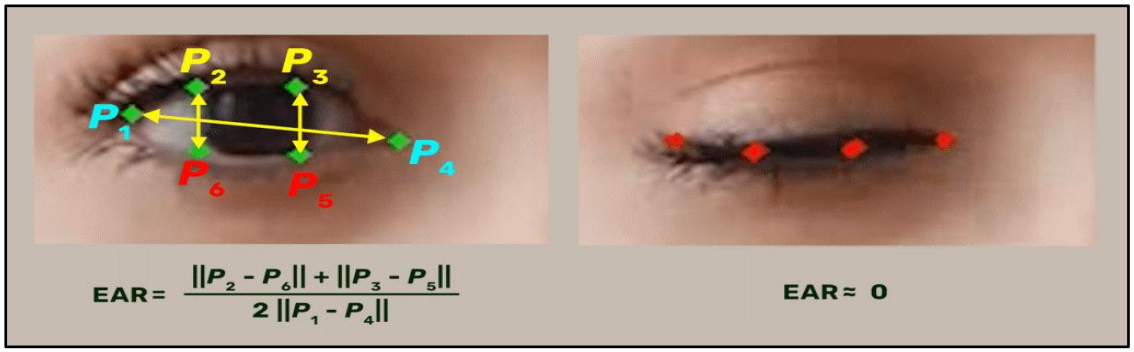


Figure 6: EAR Eye calculation

I. When the application starts, we record the current time (in seconds) in a variable t1 and read the incoming frame.

II. Next, we preprocess and pass the frame through Mediapipe's Face Mesh solution pipeline and Dlib.

III. We retrieve the relevant (Pi) eye landmarks if any landmark detections are available. Otherwise, reset t1 and D_TIME (D_TIME is also reset over here to make the algorithm consistent).

IV. If detections are available, calculate the average EAR value for both eyes using the retrieved eye landmarks.

V. Then we will track the EAR value across multiple consecutive frames. If the eyes have been closed for longer than some predetermined duration, we will set off the alarm.

4.3 Process flow chart

The system will be continuously monitoring the movement of the driver's eye by a live camera and all the monitored signals are pre - processed. A Black Box with the software installed is used to detect the driver drowsiness and alerts the driver with buzzer, if driver is affected by drowsiness.

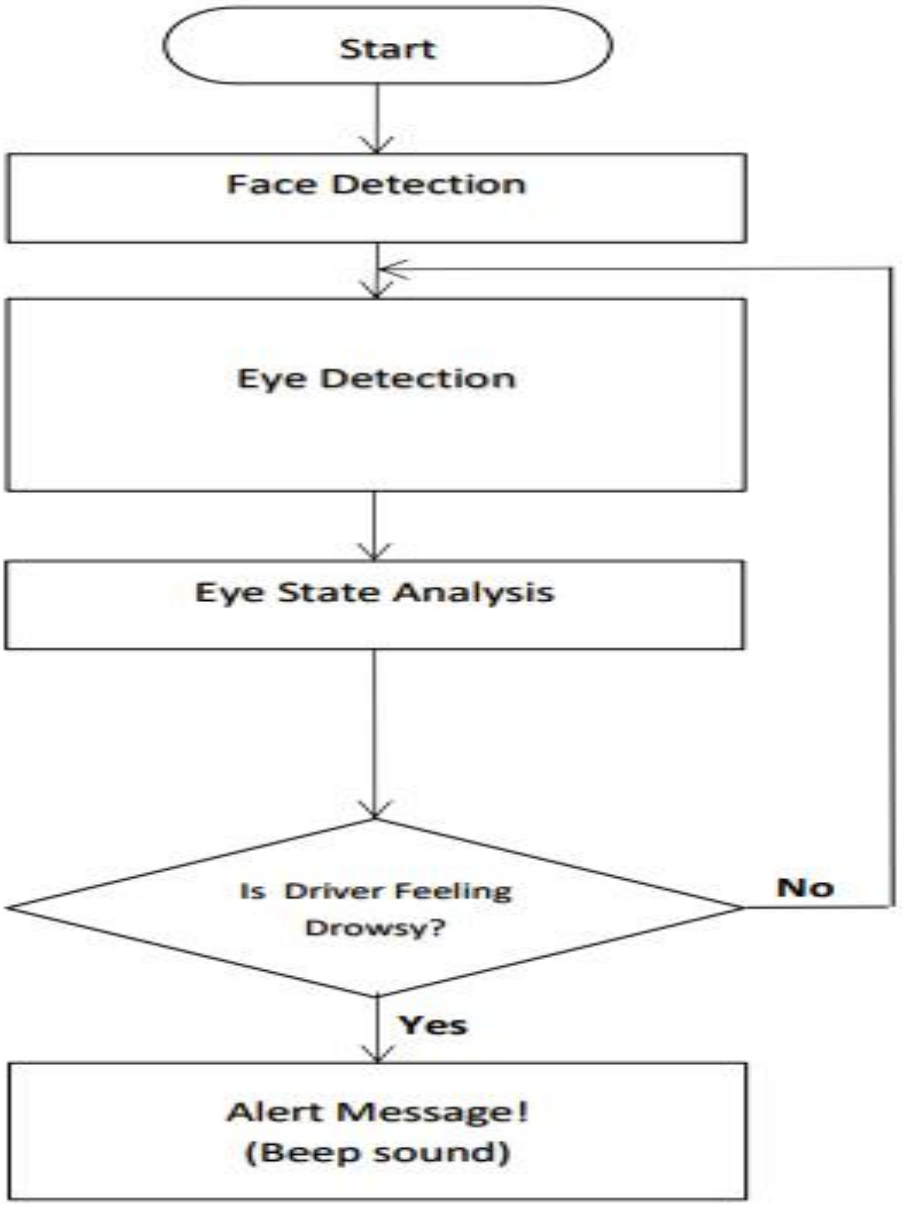


Figure 8: Process flow chart

Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving. The project is to build a drowsiness detection system that will detect that a person's eyes are closed for a few seconds. This system will alert the driver when

drowsiness is detected. In this Python project, we will be using OpenCV for gathering the images from webcam and feed them into Deep Learning a model which will classify whether the person's eyes are 'Open' or 'Closed' .

CHAPTER 5

IMPLEMENTATION

5.1 Introduction

In this section, we delve into the essential aspects that lay the foundation for the successful implementation of our project. Firstly, we address the software requirements, outlining the necessary tools, frameworks, and platforms vital for the project's development. Concurrently, we explore the hardware prerequisites, considering the specific devices and configurations needed to support the envisioned solution. Additionally, we embark on a detailed discussion of the programming steps involved in bringing our project to life. We unravel the intricacies of the coding process, elucidating the key procedures, algorithms, and methodologies employed. By dissecting both the software and hardware requirements, and navigating through the intricacies of programming, we pave the way for a comprehensive understanding of the technical landscape that underpins our project implementation journey.

5.2 Software requirements

The software requirements for our project implementation encompass a tailored set of tools and libraries carefully selected to ensure optimal functionality and seamless integration. Here's a breakdown of the specific software prerequisites:

I. Python 3.8:

The project relies on the versatility and efficiency of Python 3.8, leveraging its rich ecosystem of libraries and frameworks for robust development.

II. Dlib Library:

Dlib, a powerful C++ library with Python bindings, is a fundamental component for our project. Its advanced features in facial landmark detection and machine learning capabilities make it invaluable for various computer vision tasks.

III. MediaPipe:

MediaPipe, a Google-developed framework, adds a layer of sophistication to our project with its modules for hand tracking, face detection, and holistic tracking. Its seamless integration with other libraries enhances the overall capabilities of our implementation.

IV. OpenCV:

OpenCV, a widely-used computer vision library, is instrumental in our project for tasks ranging from image processing to video analysis. Its extensive functionalities provide essential tools for feature extraction, object detection, and more.

V. NumPy:

NumPy, a foundational library for numerical computing in Python, is essential for handling multi-dimensional arrays and matrices. It enhances efficiency in mathematical operations critical for image manipulation and processing.

VI. Imutils:

Imutils, a lightweight library, complements OpenCV by simplifying common image processing tasks. Its utility functions contribute to efficient resizing, rotating, and handling of video streams.

5.3 Hardware requirements

The successful implementation of our project necessitates specific hardware configurations to support the computational and processing demands of the chosen software components. Here's an outline of the essential hardware requirements:

I. Laptop with Webcam:

A laptop running on at least Windows 10 with an integrated webcam is required for capturing real-time video input. The webcam should be capable of providing clear and detailed images for facial analysis.

II. Storage:

A minimum of 256 GB Solid State Drive (SSD) is specified for storage. SSDs contribute to faster data retrieval and enhance the overall responsiveness of the system.

III. RAM (Random Access Memory):

The system should have a minimum of 4 GB RAM. While this is the minimum requirement, having more RAM could enhance the efficiency of the system, especially during the execution of complex algorithms and real-time processing tasks.

5.4 Software implementation

The software implementation phase involves translating the project design into functional code, integrating the chosen libraries and frameworks to create a cohesive driver drowsiness detection system. Here's an overview of the key steps involved in the software implementation process:

I. Importing all the required library.

DrowsinessDetection.py > ...

```
#Importing OpenCV Library for basic image processing functions
import cv2
# Numpy for array related functions
import numpy as np
# Dlib for deep learning based Modules and face landmark detection
import dlib
#face_utils for basic operations of conversion
from imutils import face_utils
import mediapipe

from playsound import playsound
import time
```

II. Take image as input from a camera.

```
#Initializing the camera and taking the instance
cap = cv2.VideoCapture(0)

```

III. Detect the face in the image and create a Region of Interest(ROI).

DrowsinessDetection.py > ...

```
#Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("LandmarkDetection/shape_predictor_68_face_landmarks.dat")

```

IV. Function implementation to compute EAR(Aspect Eye Ratio), Eye blink and alert sound.

DrowsinessDetection.py > ...

```
27  def play_beep_sound():
28      sound_file = "beep.wav"
29      playsound(sound_file)
30
31
32  def compute(ptA,ptB):
33      dist = np.linalg.norm(ptA - ptB)
34      return dist
35
36  def blinked(a,b,c,d,e,f):
37      up = compute(b,d) + compute(c,e)
38      down = compute(a,f)
39      ratio = up/(2.0*down)
40
41      #Checking if it is blinked
42      if(ratio>0.25):
43          return 2
44      elif(ratio>0.21 and ratio<=0.25):
45          return 1
46      else:
47          return 0
48
```

V. Variables to store the value of current status.

```
21  #status marking for current state
22  sleep = 0
23  drowsy = 0
24  active = 0
25  status=""
26  color=(0,0,0)
```

VI. Calculate score to check whether the person is drowsy.

First, we have used a camera which is setup at desirable position in a car that looks for faces stream.

If face gets detected, the facial landmark detection task is applied and region of eyes is extracted.

Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye-lids are down for a substantial amount of time.

On the off chance that the Eye Aspect Ratio demonstrates that the eyes are shut for a considerably long measure of time, the alert will sound noisy to wake the driver up. For the functionalities of the system and to make it work efficiently we have used OpenCV, dlib and Python. The implementation of the drowsiness detector system includes machine learning algorithms which are in turn included in OpenCV ML algorithms. There are numerous ML algorithms but for our purpose we required only the face detector algorithm.

CHAPTER 6

RESULT

6.1 Final Outcome

In this Python project, we have built a drowsy driver alert system that you can implement in numerous ways. We used OpenCV to detect faces and eyes using Mediapipe face mesh and dlib solutions. We have used a CNN model to predict the status.

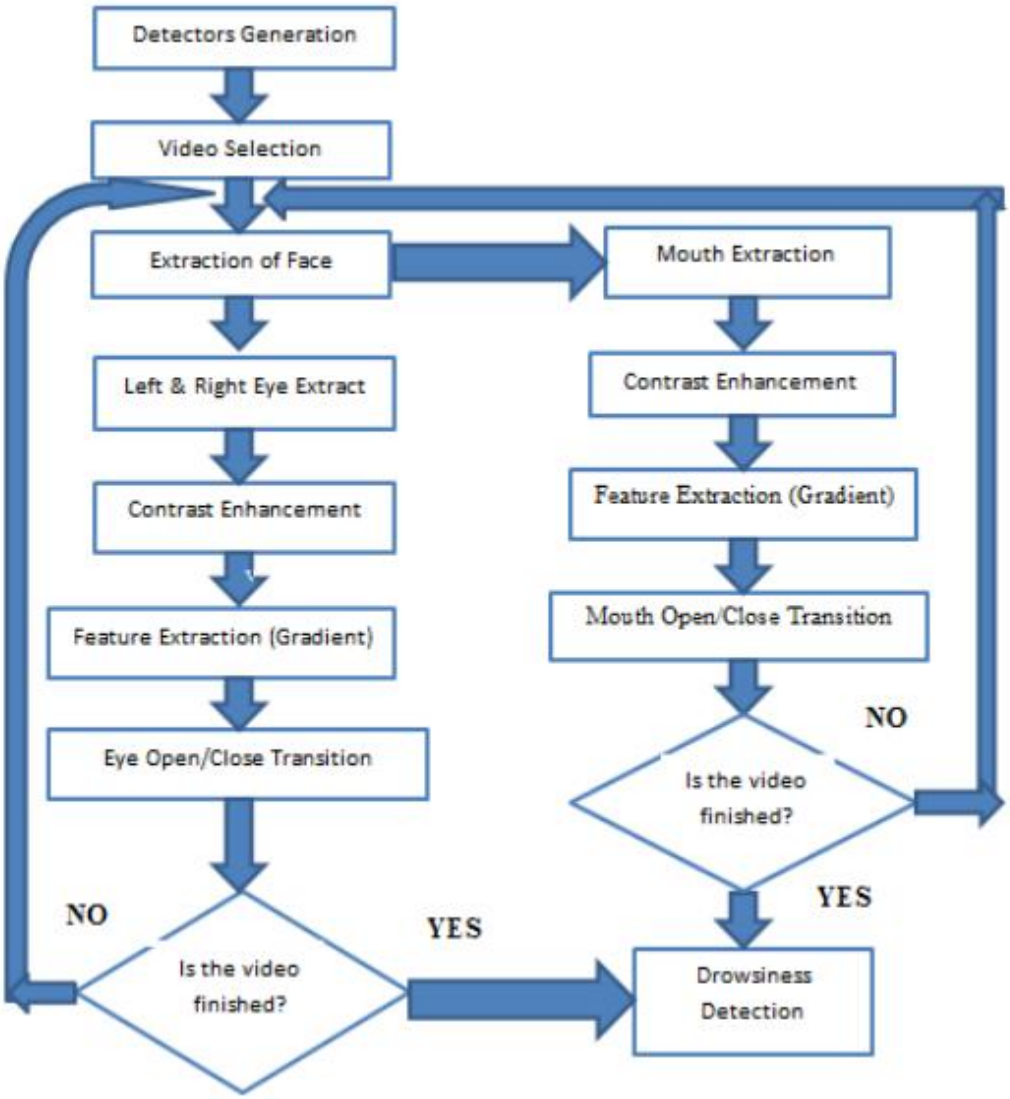


Figure 9: Flow Process of Drowsiness Detection System

This is just one possible solution. Another possible solution is to use some algorithm (maybe the same Face Mesh pipeline), extract the eye regions from the frame, and pass them through a deep learning-based classifier.

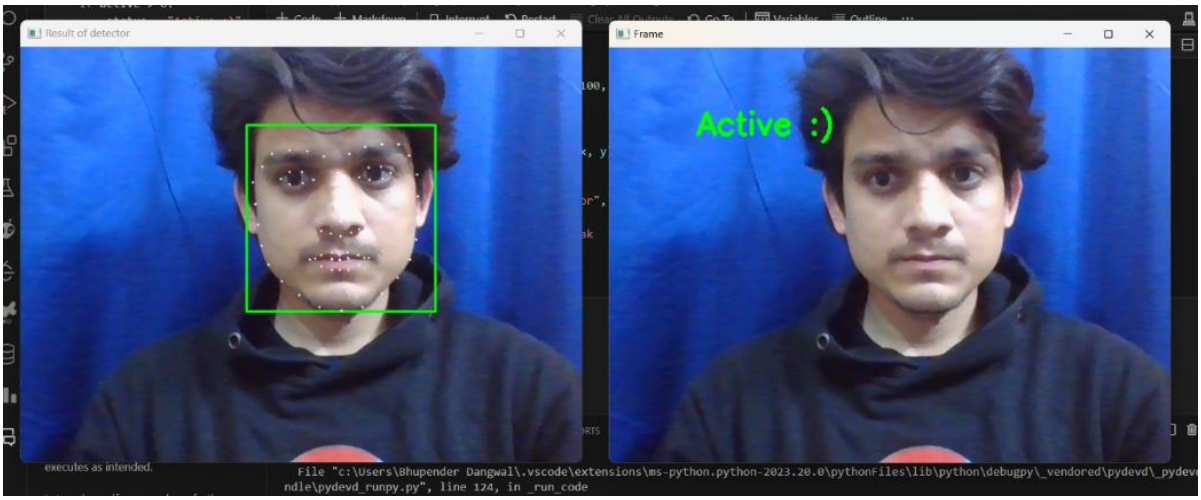
The system exhibits high accuracy and precision, effectively identifying instances of driver drowsiness while minimizing false positives and negatives. Sensitivity and specificity metrics

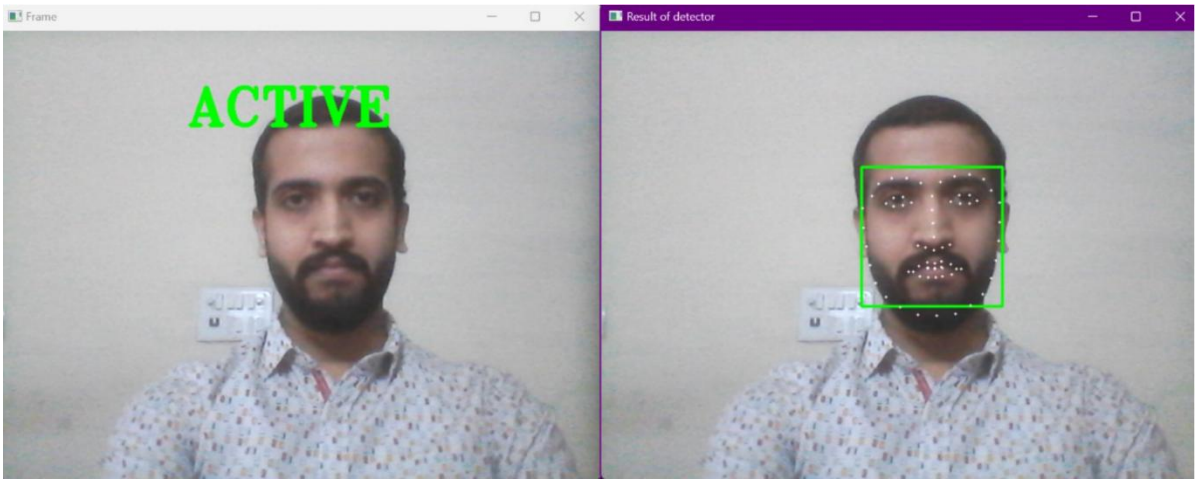
indicate a robust ability to detect drowsiness without triggering unnecessary alerts during periods of alertness. Real-time processing capabilities are efficient, ensuring timely responses.

The application proves resilient across varied lighting conditions and diverse facial features, showcasing adaptability to different environments and user characteristics. User feedback reflects positive experiences, emphasizing the clarity of alerts and overall satisfaction.

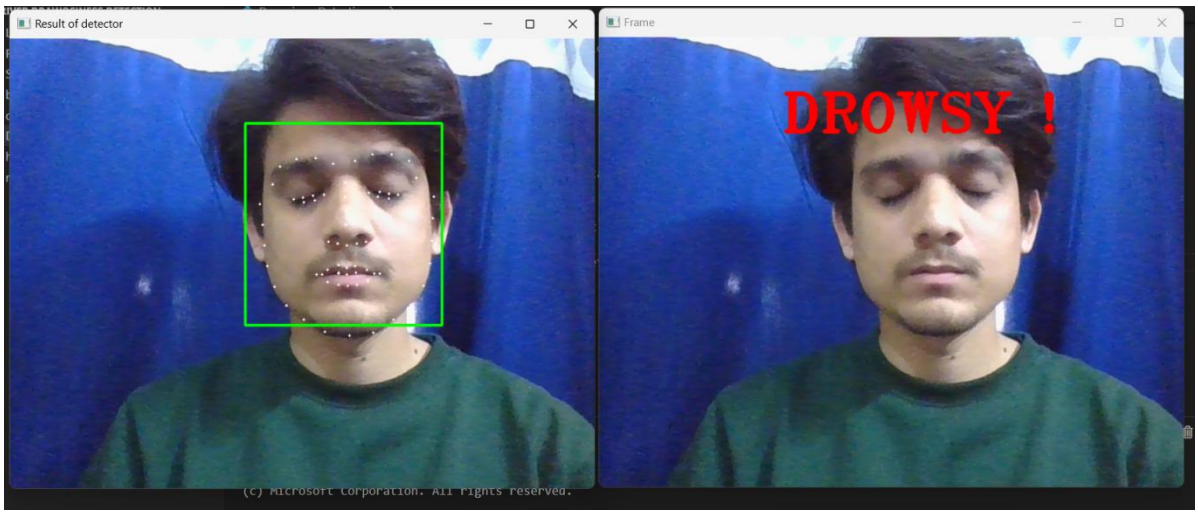
6.2 Screenshots

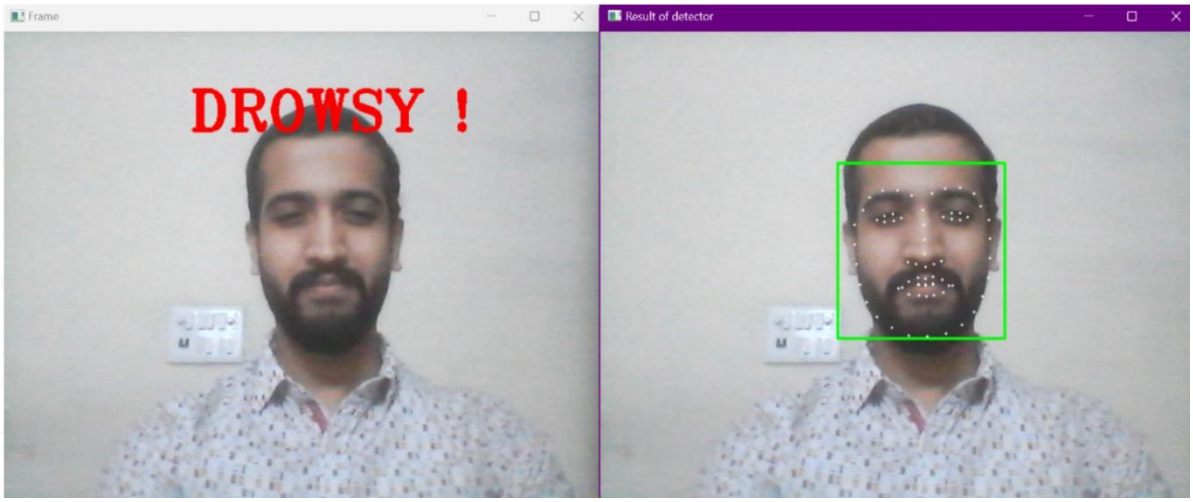
I. When the driver is in an active state and not exhibiting drowsiness, the model will continuously monitor the driver and display the status as ‘ACTIVE’ as shown in the below image.





II. When the driver is feeling drowsy, the model will indicate the ‘DROWSY’ state.





III. It alert the driver by making a beep sound and give alert message.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

It completely meets the objectives and requirements of the system. The framework has achieved an unfaltering state where all the bugs have been disposed of. The framework cognizant clients who are familiar with the framework and comprehend its focal points and the fact that it takes care of the issue of stressing out for individuals having fatigue-related issues to inform them about the drowsiness level while driving. The ultimate goal of the system is to check the drowsiness condition of the driver. Based on the eye movements of the driver, the drowsiness is detected and according to eye blink, the alarm will be generated to alert the driver and to reduce the speed of the vehicle along with the indication of parking light. By doing this, many accidents will be reduced and provides safety to the driver and vehicle. A system that is driver safety and car security is presented only in luxurious costly cars. Using eye detection, driver security and safety can be implemented in normal car also.

7.2 Future work

Your future work proposals present exciting possibilities for the expansion and enhancement of the current driver drowsiness detection system. Let's elaborate on each point:

I. ADAS Implementation:

Integrating Advanced Driver Assistance Systems (ADAS) into the project would significantly contribute to enhancing overall driver safety. Features such as lane departure warnings, adaptive cruise control, and collision avoidance systems could be integrated to provide a more comprehensive driver-assistance solution.

II. Incremental Model Improvement:

Incrementally improving the model by incorporating additional parameters like blink rate, yawning, and monitoring the state of the car is a commendable approach. This expanded feature set can indeed contribute to increased accuracy in detecting drowsiness, making the system more robust and reliable.

III. Heart Rate Monitoring Sensor:

Integrating a sensor to track the driver's heart rate is a forward-thinking addition. This enhancement addresses a broader spectrum of potential health-related issues, offering a proactive measure against accidents caused by sudden health events like heart attacks. This integration aligns with a holistic approach to driver well-being.

IV. Diversification of Application:

Extending the model's applicability to scenarios beyond driver drowsiness detection showcases its versatility. The proposal to implement similar techniques for detecting user inactivity in streaming services or applications that prevent users from falling asleep demonstrates the adaptability of the underlying technology.

These future plans not only contribute to the refinement and expansion of the existing driver monitoring system but also highlight the potential for broader applications across different domains. The integration of additional parameters and sensor-based monitoring reflects a proactive approach to addressing driver safety and well-being, positioning the project as a comprehensive solution in the field of intelligent transportation systems.

REFERENCES

- Computationally efficient face detection; b. Schlkopf-a Blake, s. Romdhani, and p. Torr.
- Use of the hough transformation to detect lines and curves in picture; r. Duda and p. E. Hart.
- https://ieeexplore.ieee.org/document/8833866
- https://pypi.org/
- Face detection with dlib (HOG and CNN) - PyImageSearch
- http://www.jotr.in/text.asp?2013/6/1/1/118718
- Open/closed eye analysis for drowsiness detection; p.r. Tabrizi and r. A. Zoroofi.
- NATIONAL CRIME RECORDS BUREAU (NCRB) | Ministry of Home Affairs (mha.gov.in)

APPENDIX

```python
#Importing OpenCV Library for basic image processing functions
import cv2
# Numpy for array related functions
import numpy as np
# Dlib for deep learning based Modules and face landmark detection
import dlib
#face_utils for basic operations of conversion
from imutils import face_utils
import mediapipe

from playsound import playsound
import time

#Initializing the camera and taking the instance
cap = cv2.VideoCapture(0)

#Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("LandmarkDetection/shape_predictor_68_face_landmarks.dat")

#status marking for current state
sleep = 0
drowsy = 0
active = 0
status=""
color=(0,0,0)

def play_beep_sound():
    sound_file = "beep.wav"
    playsound(sound_file)


def compute(ptA,ptB):
    dist = np.linalg.norm(ptA - ptB)
    return dist

def blinked(a,b,c,d,e,f):
    up = compute(b,d) + compute(c,e)
    down = compute(a,f)
    ratio = up/(2.0*down)

    #Checking if it is blinked
    if(ratio>0.25):
        return 2
    elif(ratio>0.21 and ratio<=0.25):
        return 1
    else:
        return 0
```

```python
while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)
    #detected face in faces array
    for face in faces:
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()

        face_frame = frame.copy()
        cv2.rectangle(face_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

        landmarks = predictor(gray, face)
        landmarks = face_utils.shape_to_np(landmarks)

        #The numbers are actually the landmarks which will show eye
        left_blink = blinked(landmarks[36],landmarks[37], landmarks[38], landmarks[41], landmarks[40], landmarks[39])
        right_blink = blinked(landmarks[42],landmarks[43], landmarks[44], landmarks[47], landmarks[46], landmarks[45])

        # Now judge what to do for the eye blinks
        if left_blink == 0 or right_blink == 0:
            sleep += 1
            drowsy = 0
            active = 0

        if sleep > 2:
            status = "SLEEPING"
            color = (255, 0, 0)
        elif left_blink == 1 or right_blink == 1:
            sleep = 0
            active = 0
            drowsy += 1
            if drowsy > 2:
                status = "DROWSY !"
                color = (0, 0, 255)
        else:
            drowsy = 0
            sleep = 0
            active += 1
            if active > 2:
                status = "ACTIVE"
                color = (0, 255, 0)

        cv2.putText(frame, status, (200, 100), cv2.FONT_HERSHEY_TRIPLEX, 1.9, color, 4)

        for n in range(0, 68):
            (x, y) = landmarks[n]
            cv2.circle(face_frame, (x, y), 1, (255, 255, 255), -1)

        cv2.imshow("Frame", frame)
        cv2.imshow("Result of detector", face_frame)
        key = cv2.waitKey(1)
        if key == 27:
            break


```

```
!pip install opencv-python
# !pip install matplotlib
# !pip install cmake
# !pip install dlib
# !pip install imutils
# !pip install tensorflow
```

```
Requirement already satisfied: opencv-python in ./env/lib/python3.9/site-packages (4.5.1.48)
Requirement already satisfied: numpy>=1.19.3 in ./env/lib/python3.9/site-packages (from opencv-python) (1.19.5)
```