Unconsciousness detection alarm for driver

PROJECT REPORT

Submitted by

Aman Chaturvedi
Akshay Kumar Kushwaha

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

AIT CSE

Chandigarh University

Unconsciousness detection alarm for driver

PROJECT REPORT

Submitted by

Aman Chaturvedi (20BCS6814)

Akshay Kumar Kushwaha (20BCS6815)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE & ENGINEERING

Chandigarh University

BONAFIDE CERTIFICATE

Certified that this project report " Unconsciousness detection alarm for driver " is the bonafide work of Aman and Akshay Kumar.

who carried out the project work under my/our supervision.

SIGNATURE

Anubhav Kumar Sir

SUPERVISOR

Computer Science & Engineering

ACKNOWLEDGEMENT

In the completion of the project based on algorithm analysis a lot of guidance, training and practice was required. First and foremost we would thank God for equipping me with such knowledge and desire to achieve the goal.

In the successful completion of this project and the work undertaken we would like to thank Chandigarh University for providing the opportunity to work on such projects under the guidance of skilled mentors and teachers.

We would like to express our deep gratitude to to our project Supervisor , Department of Computer Science and Engineering, Chandigarh University, Mohali, Punjab, India, for their guidance, insightful comments, and constructive suggestions to improve the quality of the project work.

At last we would like to thank all the engineers and researchers to develop such platform for us to conduct such tasks through their software and hardware.

TABLE OF CONTENTS

List of Figures.....6

List of Tables.....8

Abstract.....10

Abbreviations.....11

Chapter 1. Introduction

1.1 Identification of client and need.....12

1.2 Background of study.....15

1.3 Identification of Problem.....15

1.4 Identification of Tasks.....16

1.5 Timeline.....17

Chapter 2. Literature Review

2.1 Drowsiness and Fatigue.....19

2.2 Proposed Solutions (Electroencephalography for Drowsiness Detection).....19

2.3 Drowsiness detection using face detection system.....23

2.4 Review Summary.....24

2.5 Problem Definition.....24

2.6 Goals/ Objectives.....25

Chapter 3. Design Flow/ Process

3.1 Evaluation & Selection of Specification/ Features.....26

3.2 Design Limitations.....26

3.3 Refinement of Boundary Based Analysis and Features.....29

3.4 Project Work Flow.....30

Chapter 4. Result Analysis & Validation

4.1 Implementing Solution.....36
4.2 Tools.....36
4.3 Software Implementation.....38
4.4 Software Modeling.....41

Chapter 5. Conclusion & Future Work

5.1 Conclusion.....46
5.2 Future Work.....49

Appendix.....52
References.....55
User Manual.....59

LIST OF FIGURES

Figure 1 : Statistic of Road Accident from 1970 to 2020.....1
Figure 2 : Blue Print of the proposed System.....2
Figure 3 : Examples of Fatigue & Drowsiness Condition.....3
Figure 4 : Cascade of Classifiers.....8
Figure 5 : Examples of Person in Normal and Yawning Condition.....9
Figure 6 : Examples of Eyelid Movement.....10
Figure 7 : Feature Used in Viola-Jones.....15
Figure 8 : Cascade of Classifiers.....16
Figure 9 : Flow chart of System.....17
Figure 10 : Flow Chart of Project Progress.....18
Figure 11 : DFD Level 0.....18
Figure 12 : DFD Level 1.....19
Figure 13 : DFD Level 2.....20
Figure 14 : Use-case Diagram.....20
Figure 15 : Sequence Diagram.....21
Figure 16 : Face Detection Result.....25
Figure 17 : Eye Detection Result.....27
Figure 18 : Mouth Detection Result.....29
Figure 19: Flow Process of Detection Algorithm.....30
Figure 20 : True Positive Eyes and Mouth Detection.....31
Figure 21 : Flow Process of Face Detection System.....32
Figure 22 : Flow Process of Mouth Detection System.....33
Figure 23 : Flow Process of Eye Detection System.....34
Figure 24 : Flow Process of Drowsiness Detection System.....35

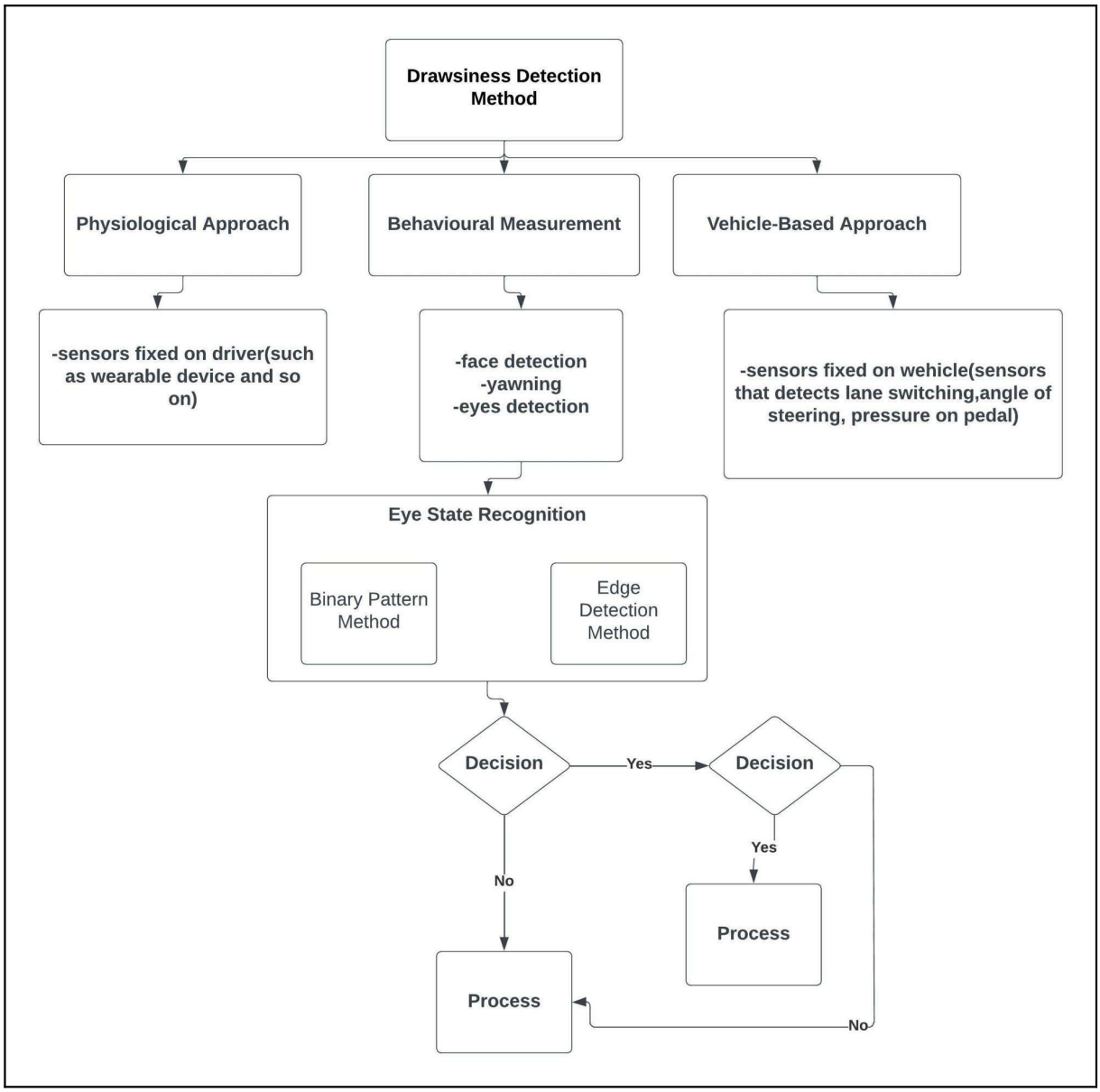
LIST OF TABLES

Table 1 : Gantt Chart.....6

ABSTRACT

Today, the number of traffic accidents caused by microsleep has been alarming. When microsleep, people can snore without even realizing it. Vehicle drowsiness detection system has not been a major concern for many decades, although it turns out to be one of the necessary features that could prevent micro-sleep mode and therefore should be implemented in all vehicles to ensure the safety of drivers and other vehicles on. the road. To our knowledge, the introduction of driving restrictions while drowsy has not yet been implemented. The absence of such a system in the current traffic systems puts drivers at great risk, especially at night, because there is a high probability of accidents due to sleepy and tired drivers at night. Therefore, this project proposes a real-time drowsiness detection system for vehicles with immobilizer to reduce accidents. An eye blink sensor is built into the wearable glasses and a heart rate sensor detects driver drowsiness. The system also includes an SMS notification system for relatives or friends, which includes the location information of the drunk driver. This project can detect and respond to three levels of sleep alerting the driver with an audible alarm. A power lock turns on when a high level of sleep is detected. As a result, the vehicle slows down and then stops when dangerous drowsiness is detected for safety reasons.

GRAPHICAL ABSTRACT



ABBREVIATIONS

EKG	Electrokardiogram
ECG	Electrocardiogram
EEG	Electroencephalography
IRTE	India Institute of Road Safety
EAR	Eye shape ratio
MAR	Mouth shape ratio
CNN	Convolutional Neural Network
BSD	Berkeley Source Distribution
NUI	Natural user interface
UI	User Interface
PERCLOS	Final Year Project
FYP	PERcentage of eye CLOSure

CHAPTER-1
INTRODUCTION

1.1 Identification of client and need

A person who is drowsy has a strong desire to sleep and is in a condition of approaching sleep. It can refer to both the routine state just before falling asleep and the chronic condition that refers to remaining in that state without regard to a regular rhythm. When performing activities that demand continual attention, such as operating a vehicle, sleepiness might be harmful. A person will feel drowsy while driving if they are sufficiently exhausted, which increases the risk of a traffic collision.

The number of vehicles involved in traffic accidents is increasing every year.

Figure 1: Statistic of Road Accident from 1970 to 2020

Developing technologies to detect or prevent drowsiness while driving is a major challenge in the field of accident prevention systems. Because of the danger of sleeping on the road,

methods must be developed to combat its effects.

The huge increase in traffic accidents worldwide has led to serious researchers. Many different solutions have been proposed to improve driver safety. However, most of the existing drowsiness detection systems have some drawbacks. Most systems can only detect drowsiness and issue warnings to warn drivers, while they can ignore the warnings and continue driving while drowsy. In this case, traffic accidents can still happen and the danger these drivers pose to other road users is still great. Most of the existing work focuses on both indoor and outdoor vehicles to prevent crashes due to drowsiness. Vehicle interior device research primarily concerns drivers.

This is because 20% of accidents and 30% of fatal accidents in the world are caused by driver drowsiness and lack of driver concentration. The built-in system alerts the driver by giving certain warnings when the driver is already drowsy.

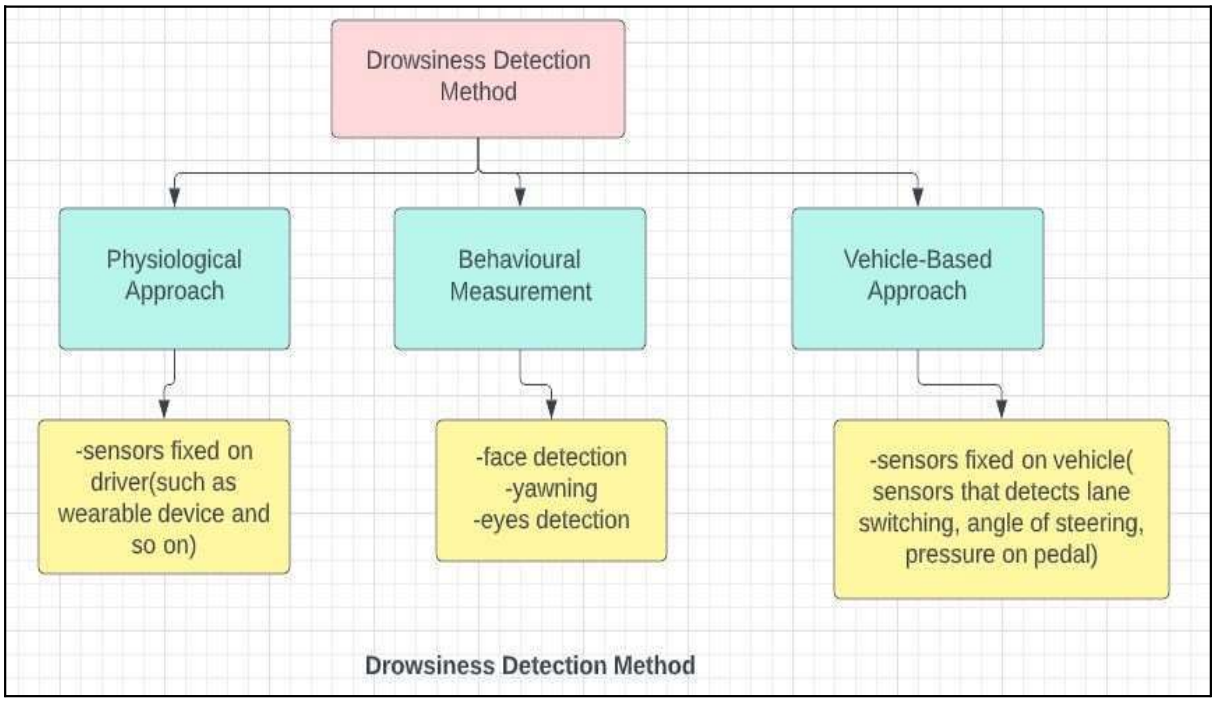


Figure 2: Blue Print of the proposed System

The driver's heart rate is analyzed by an EKG sensor for simulation purposes. The ECG signal is tested to compare it with a normal heart rate (60 > rate and <100). Finally, it is decided whether the driver is drowsy by matching both camera outputs and heart rate. The driver is constantly monitored and warned through a loudspeaker to avoid accidents if the

Driver drowsiness and fatigue are major causes of traffic accidents. Every year, the number of deaths and deaths worldwide is increasing. The system presents modules for driver assistance systems in order to reduce the number of accidents caused by driver fatigue and increase road safety. The system automatically detects driver drowsiness based on visual information and artificial intelligence.

The proposed OpenCV algorithm helps to effectively normalize human faces while causing most of the accidents related to car crashes. The algorithm first recognizes the head in a color image based on the color and texture variations of the human face and background.

Multiple facial and body gestures, such as tired eyes and yawning, are considered signs of driver drowsiness and fatigue. These characteristics indicate a bad driver.

1.4 Identification of Tasks

The task of this system can be divided into two parts:

- Face recognition or localization.
- Prediction of important regions on detected faces.

After landmark prediction, we only use the eye and mouth landmarks to determine the eye shape ratio (EAR) and mouth shape ratio (MAR) to check whether the person is sleepy.

- We must use the dlib library inside the library, which contains a pre-trained face orientator, which is used to estimate the location of x, y coordinates related to the facial structure of the face
- The coordinates represent important areas of the face, such as the mouth, left eyebrow, right eyebrow, left eye, right eye, nose and chin.
- Eye aspect ratio (EAR) is calculated using Euclidean distance. EAR is the ratio of the maximum horizontal length of the eyes to the maximum length of the eyes. The algorithm searches for eye marks, measures the absolute distance between points 37 and 40, 43 and 46, and selects the most important distance from the front of the camera eye.
- Once the coordinates are obtained, we must calculate the proportion of the mouth.

- Images were taken at 6 frames per second as a final detection step.
- If the network estimates that the probability of closing the eyes for more than 12 consecutive images is more than 50%, a hazard warning and an audio signal are sent to the driver.
- To save the prints, we also have to save the frames where the person felt sleepy.

Regarding the user interface, the following tasks must be considered:

- All functions offered to the user should be easily accessible to the user. We have to take care of it. To this end, we will continue to improve the user interface of our project in every way. We listen to our users and their suggestions and comments to ensure that our project runs smoothly and correctly.
- The basic functions and proper operation and calculations that we mentioned in the first point are never provided by other functions that we provide and must be kept in a fast charging state. We don't want our users to wait a long time to get an alert that they are sleeping. To do this, we optimize our code so that it takes very less time and is very fast, computational and responsive.

1.5 Timeline

Building this software on time requires following a strict schedule for each task. His first two weeks were spent completing the project and briefing the team. Additional months are divided into tasks such as problem identification, design flow and process, simulations, literature review, performing analysis and gathering results.

A detailed Gantt chart below will show the exact proportion of time provided for each task:

Table 1: Gantt Chart (Timeline of the Project)

Activities Planned		August				September				October				November			
		Week															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Phase 1	Project finalization and understanding																
	Title																
	Abstract																
	Deciding Tools/Software																
	Literature Review																
Phase 2	Basic design and undertaking																
	Simulation																
	Stimulating various model changes																
	Gathering results and analysis																
	Report and PPT Review by management																

CHAPTER-2
LITERATURE REVIEW

There are many previous studies on driver drowsiness detection systems that can be used as references for developing driver drowsiness detection systems that detect driver drowsiness in real time. There are also several ways to use different approaches to spot signs of fatigue.

2.1 Drowsiness and Fatigue

Antoine Picot et al stated that drowsiness is when a person is awake and in a state of sleep. Such a situation leads to the fact that the leader does not pay full attention to his leadership. Therefore, the vehicle can no longer be driven because the driver is semi-conscious. According to Gianluca Borghini et al., mental fatigue is one of the factors of sleep and caused the inability of the experienced person to perform because it reduces the brain's ability to respond to unexpected events.

2.2 Proposed Solutions (Electroencephalography for Drowsiness Detection)

Step 1: Overview of Required Hardware

In this method, the camera is in front of the user and the image is sent to the processor. Facial feature recognition is a template rearrangement operation. This operation involves comparing facial expressions, especially eyes, and images.

Step 2: Face Reconstruction Using the Viola-Jones Algorithm

The Viola-Jones Algorithm was introduced in 2001 by POWEL VIOLA and MICHEL JONES. This algorithm is commonly used for object detection, but is widely used for eye detection. This algorithm takes a long time to train. However, the discovery process is very quick and easy. This algorithm is used to detect objects in real time. Some classes are used in sequential order to facilitate and speed up the identification process.

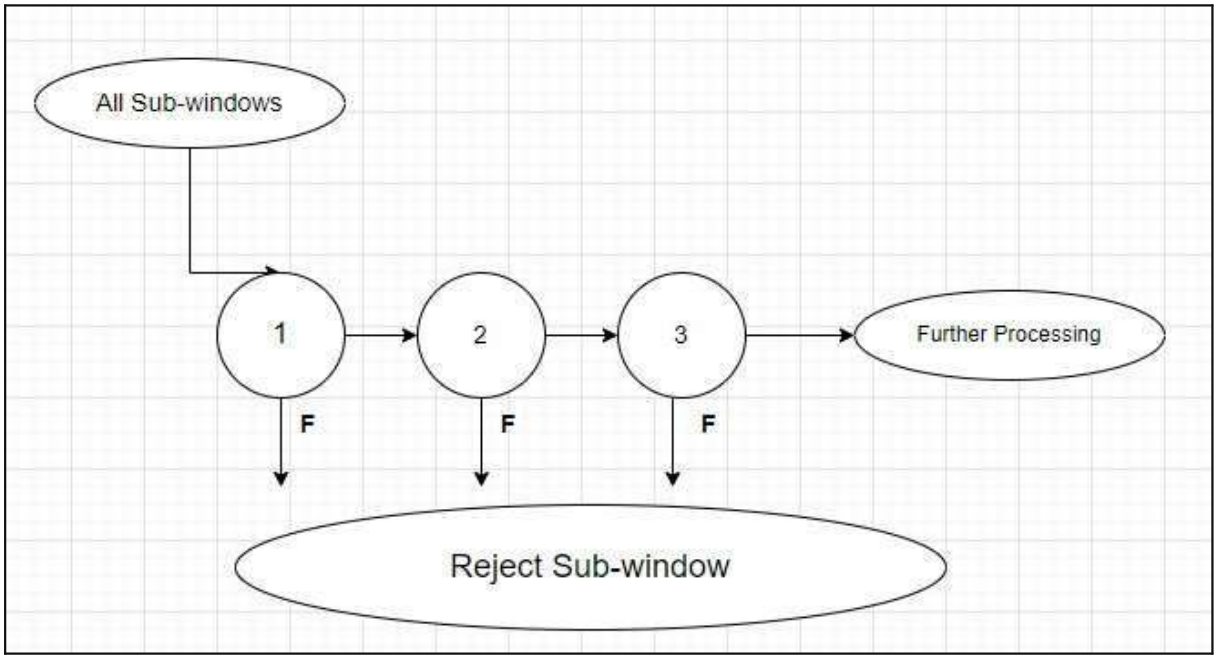


Figure 4: Cascade of Classifiers

Didactic example and results of detection are as follow:

Step 3: Find eyes on the face

Many methods have been developed to find eye locations. Eye detection often involves identifying the driver's eyes. This task is accomplished using various methods. A simple way is to use the eye properties. By describing different races, different people have different eye color spectrum. The RGB format is not suitable for this as 1 pixel is used to determine the color code. Because in addition to hue, other factors such as lighting and saturation are also important. Therefore, other forms of lighting, saturation, hue, etc.

Step 4: Extraction of features

After detecting the eyes, we convert the eye images to binary format to detect closed and open eyes and blinks. This conversion reduces the amount of data processed in the system. The image is then divided into top and bottom. When the eyes are open, the pupils and eyelashes are dark, so the ratio of dark pixels at the top to the bottom is higher than when the eyes are closed. This is because the eyelids cover the eyes and the eyelashes are at the bottom. The lighting of the image should be normalized before the binary conversion. To normalize the image, it is converted to gray-scale. Then, paying attention to the V middle, the lighting

is demonstrated. The image is then converted to gray-scale. Then, setting a threshold, we convert the image to binary format.

Step 5: Drowsiness detection:

Drowsiness level was determined by information such as whether the eyes were closed or open, blink frequency, blink duration, and use of a neural network.

Electroencephalography (EEG) is a method that measures the electrical activity of the brain. It can be used to measure heartbeats, eye blinks and even large physical movements such as head movements. It can be used to increase brain activity in humans or animals. It uses special hardware that places sensors on top of the head to detect electrical activity in the brain.

The disadvantage of this method is that it is very sensitive to the noise surrounding the sensors. For example, when a person takes an EEG test, the environment must be completely quiet. Noise interferes with the work of sensors that detect brain activity. Another disadvantage of this method is that even if the result is accurate, it is not suitable for use in a real driving application.

Imagine if a person is driving with something full of wires on his head, and if the driver moves his head, the wire can come off. Although it is not convenient to use for real-time driving, but for testing and data collection, it is one of the best methods so far.

Yawn Detection Method: A person's sleepiness can be detected by looking at their face and behavior. Some people cover their mouths with their hands when they yawn. The obstacle to getting good pictures is when a person closes their mouth while yawning, but yawning is definitely a sign of sleep and fatigue.

Figure 5: Examples of Person in Normal and Yawning Condition

After reviewing research and existing methods, this project proposed using an eye and yawn detection method. Blink duration indicates that the longer people keep their eyes closed, the more sleepy they feel. when a person is asleep. His eyes are closed longer than normal blinks. Apart from that, yawning is one of the symptoms of drowsiness, and if yawning is a sign that you are feeling sleepy or tired, it is a normal human reaction.

2.3 Drowsiness detection using face detection system

Drowsiness can be detected by face area detection. There are different ways to detect facial drowsiness. Drowsiness signs are more clearly visible in the facial area. You can see the position of the eyes from the part of the face. Regarding eye detection, the authors said that there are four types of eyelid movements that can be used to detect drowsiness: fully open, fully closed, and open eyes. to closed and vice versa.

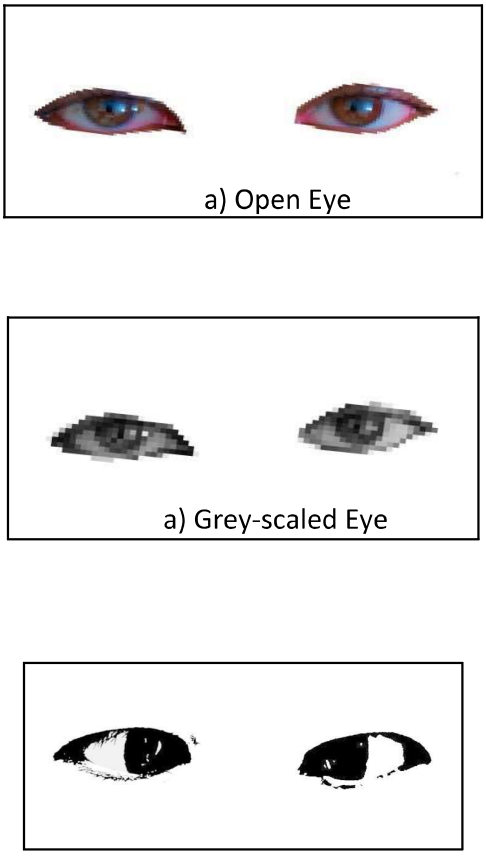


Figure 6: Examples of Eyelid Movement

The algorithm processes captured images using a gray-scale process. Then the colors of the image

are converted to black and white. It is easier to work with black and white images as only two parameters need to be measured. Then edge detection is performed to find the edges of the eyes and calculate the eyelid region values.

The problem with this method is that the size of the eye area may vary from person to person. Some people have small eyes and look sleepy, others don't. Apart from that, if a person wears spectacles, they have trouble perceiving the eye area. The image captured must be within a certain range from the camera, as the image will be blurry at greater distances from the camera.

2.4 Review Summary

A safety tool called fatigue detection can prevent accidents caused by drivers falling asleep at the wheel.

The goal of this intermediate Python project is to develop a drowsiness detection system that can detect when someone closes their eyes for a short period of time. A system that warns the driver when drowsiness is detected.

In this Python project I created a sleepy driver alert system that can be used in a variety of ways. After detecting faces and eyes with a hair cascade classifier using OpenCV, we used a CNN model to predict states.

Drowsiness and fatigue lead to road accidents in India. Webcam driver drowsiness detection is introduced to minimize and reduce the number of car, truck and truck accidents. It recognizes the signs of drowsiness and warns you when your driver is tired.

2.5 Problem Definition

Current driver drowsiness detection systems require complex calculation and expensive devices that are not convenient to use while driving and are not suitable for driving conditions; for example electroencephalography (EEG) and electrocardiography (ECG), i.e. e) detection of brain frequency and measurement of heart rhythm.

Earlier, we saw that most quests to satisfy fail for unexpected reasons. Sleepiness can also be considered as one of the problems due to which companies face task failure. First of all, sleepy employees are much less efficient, they react especially slowly, make more mistakes and may forget to do things that can adversely affect the productivity of a particular business enterprise. Sleepy workers also have poorer adaptability. This means that they are no longer able to

understand how to deal with the changing situation and new challenges that may become increasingly unusual in the current work environment.

In addition, they may have difficulty multitasking or quickly switching between different tasks, even in daily work. And according to the Fatigue Calculator, developed by Brigham Health as part of the National Safety Council's sleep-related projects, it estimated that a medium-sized Fortune 500 company with about 52,000 employees would lose about eighty million dollars a year due to tired workers. This type of damage can lead to the loss of the business itself. So there must be a system or software that maintains a real-time file and alerts each person to complete the work on time.

Therefore, the purpose of this project is to analyze all previous studies and methods, so a method to detect drowsiness using a video or web camera is proposed. It analyzes the recorded video images and offers a system that can analyze each frame of the video.

2.6 Goals/Objectives

The project focuses on the objectives, which are:

- Proposing ways to detect fatigue and drowsiness while driving.
- To examine the eyes and mouth from the video images of the participants in the IRTE driving simulation test, which can be used as an indicator of fatigue and sleep.
- Investigate physical changes in fatigue and sleepiness.
- Develops a system that uses eye closure and yawning to detect fatigue and sleepiness.

In this project, the author focuses on the following procedures:

- Basic concept of drowsiness detection system
- Get familiar with symptoms of drowsiness
- Define drowsiness based on these parameters
    - Blinking
    - Detected pupil area. eyes

- Yawn

- The Data collection and the measurement.
- Development and testing of coding.
- Full testing and improvement.

CHAPTER-3
DESIGN FLOW

3.1 Specification/Feature Evaluation and Selection

During this phase, it was determined that one of the best ways to detect eyes and yawns was an algorithm. Some of the current algorithms associated with this project are being reviewed to support the development of the project. In, the proposed method measures the time that a person keeps their eyes closed, and if the time that the eyes are closed is longer than the normal blinking time, the person is likely to fall asleep. Based on a study of human blinks, we found that the average human blink time is about 202.24 milliseconds, while a sleepy person's blink time is about 258.57 milliseconds.

Once the methods used in this project are defined, the author will receive a video of the experiments performed by his IRTE. In this video, participants drive in a simulated environment and are recorded throughout the session. Experiment time is about 60 to 90 minutes. Drowsiness detection analysis is done manually by watching a full-length video and noting signs of drowsiness. The data parameters are:

Drowsiness, yaw, and other signs are displayed at start and end times. This is used to calculate the duration of the characters that occurred.

3.2 Design Limitations

Several algorithms and techniques were used for face, eye, and mouth detection. The algorithm and technique used is the Cascade Object Detector. The Cascade Object Detector uses the Viola-Jones algorithm to detect a person's face, nose, eyes, mouth, or torso.

Viola-Jones Face Recognition Algorithm The Viola-Jones object recognition framework can be used to recognize a wide range of object classes, but is more focused on face and facial feature recognition. This algorithm uses the concept of rectangular features, which involves summing pixels within a rectangular area. In Figure 8, the sum of pixels contained in the white rectangle is subtracted from the sum of pixels contained in the gray rectangle. The value of a feature with two rectangles, represented by A and B, is the sum difference of the

pixels within the two rectangular regions. The regions are the same size and shape. Also, they are arranged horizontally or vertically and are adjacent to each other. A feature with three rectangles, represented by C, is calculated by subtracting the sum in the two outer rectangles from the sum in the middle rectangle. Finally, the four rectangular features, denoted as D, compute the difference between pairs of rectangular diagonals.

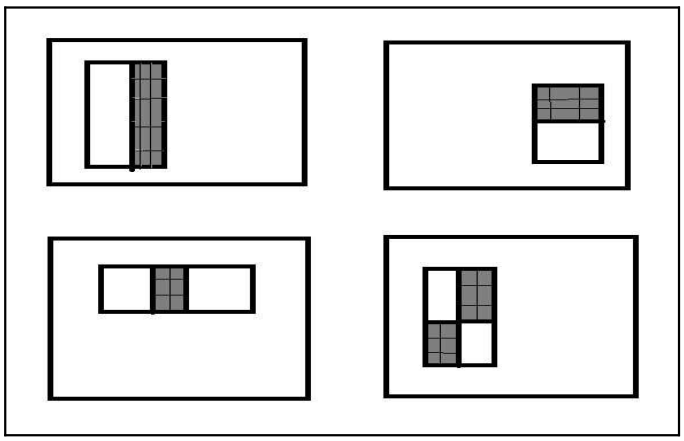


Figure 7: Feature Used in Viola-Jones

Rectangular features can be quickly computed using a representation of the image called the integral image.

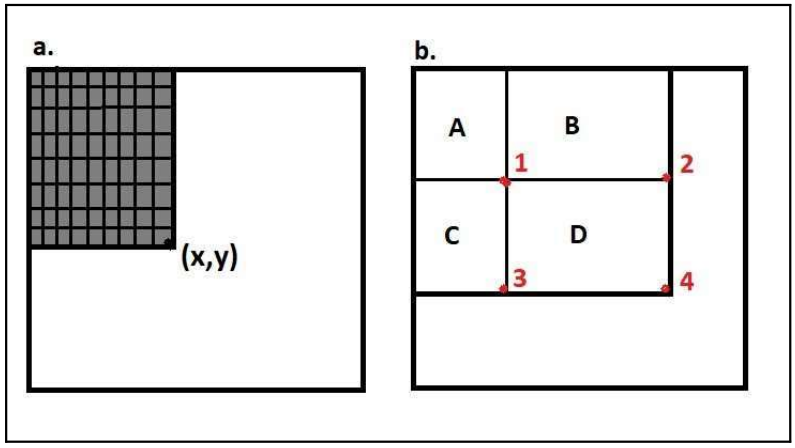


Figure 8: Feature Used in Viola-Jones

The integrated image value at (x, y) is the sum of all pixels above and to the left. Based on the integrated image, the pixel sum of rectangle D can be calculated using four matrix references. The value of the integrated image at position 1 is the sum of the pixels in rectangle A. The value at position 2 is A B, at position 3 is A C, and at position

is A B C D. For the face recognition task, the selected features for face location detection are shown in Figure 10. These two functions are shown in the top row and are superimposed on a regular training surface in the bottom row. The first function measures the difference in intensity between the eye area and the top of the cheek. This is because the area around the eyes is often darker than the top of the nose and cheeks. The second function compares the intensity in the eye regions with the intensity across the bridge of the nose.

Cascade of Classifiers A total of 45,396 possible features are recognized in a standard 24×24 pixel sub-window. This is too big and prohibitively expensive to evaluate. To improve recognition performance, features should be added to the classifier. However, this step directly increases computation time and slows down the detection process significantly. Thus, a cascade of classifiers is constructed, improving recognition performance while significantly reducing computational time.

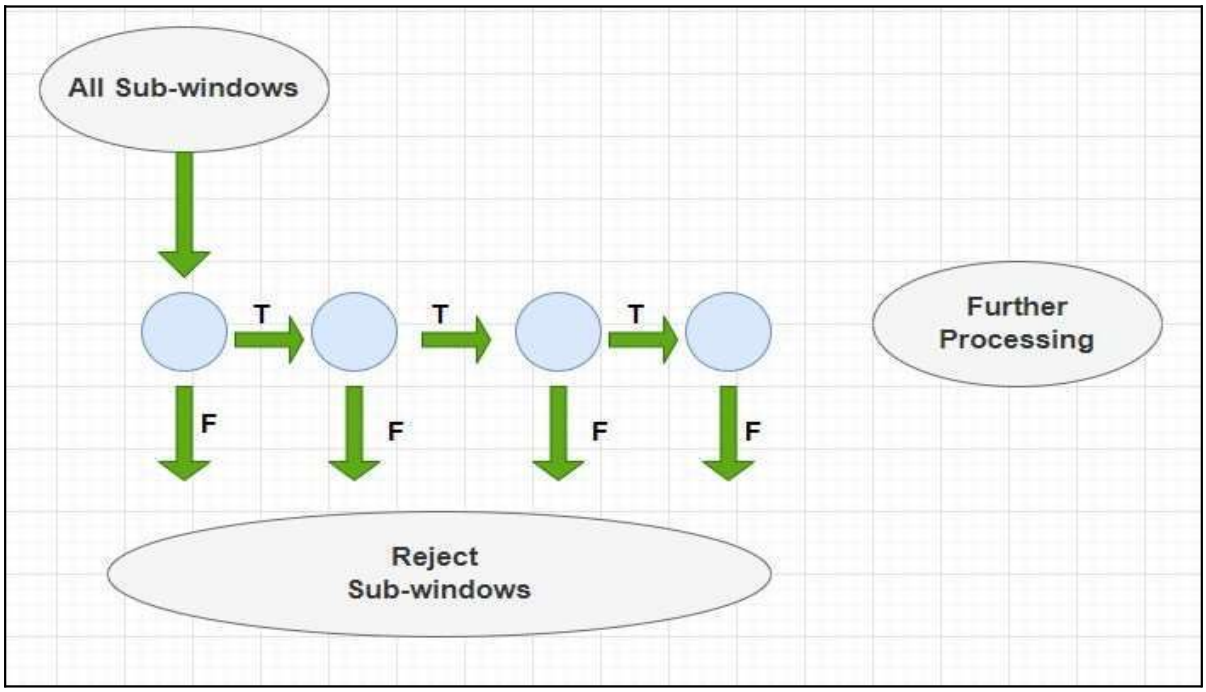


Figure 9: Cascade of Classifiers

Evaluation of the strong classifiers produced by the learning process can be done quickly, but not fast enough to run in real time. Strong classifiers are therefore arranged in a cascade in order of complexity. Each subsequent classifier is trained only on selected samples that pass the previous classifier. At any point in the cascade, if the classifier rejects the inspected sub-window, no further processing is performed and the search continues for the next sub-

window.

3.3 Refinement of Boundary Based Analysis and Features

To recognize the eye and mouth region, the face region must be recognized first. However, this step reduces the performance and speed of the system due to the large detection range. The aim of the project is to identify signs of sleepiness, i.e. the eye and mouth area. Therefore, this project limited the detection range through eyes and mouth. This improves system performance. The cascade object detector algorithm is tested using MATLAB® software to obtain the detection area used by the development system. Testing must be done to ensure that it meets the required parameters.

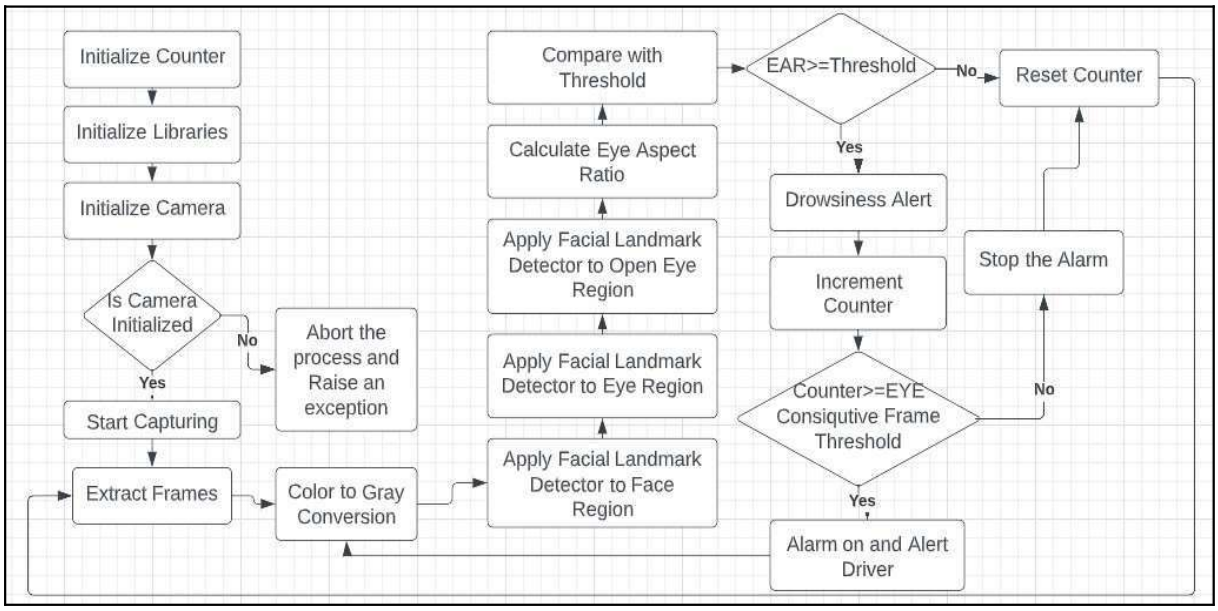


Figure 10: Flow chart of System

3.4 Project Work Flow

Flow Chart

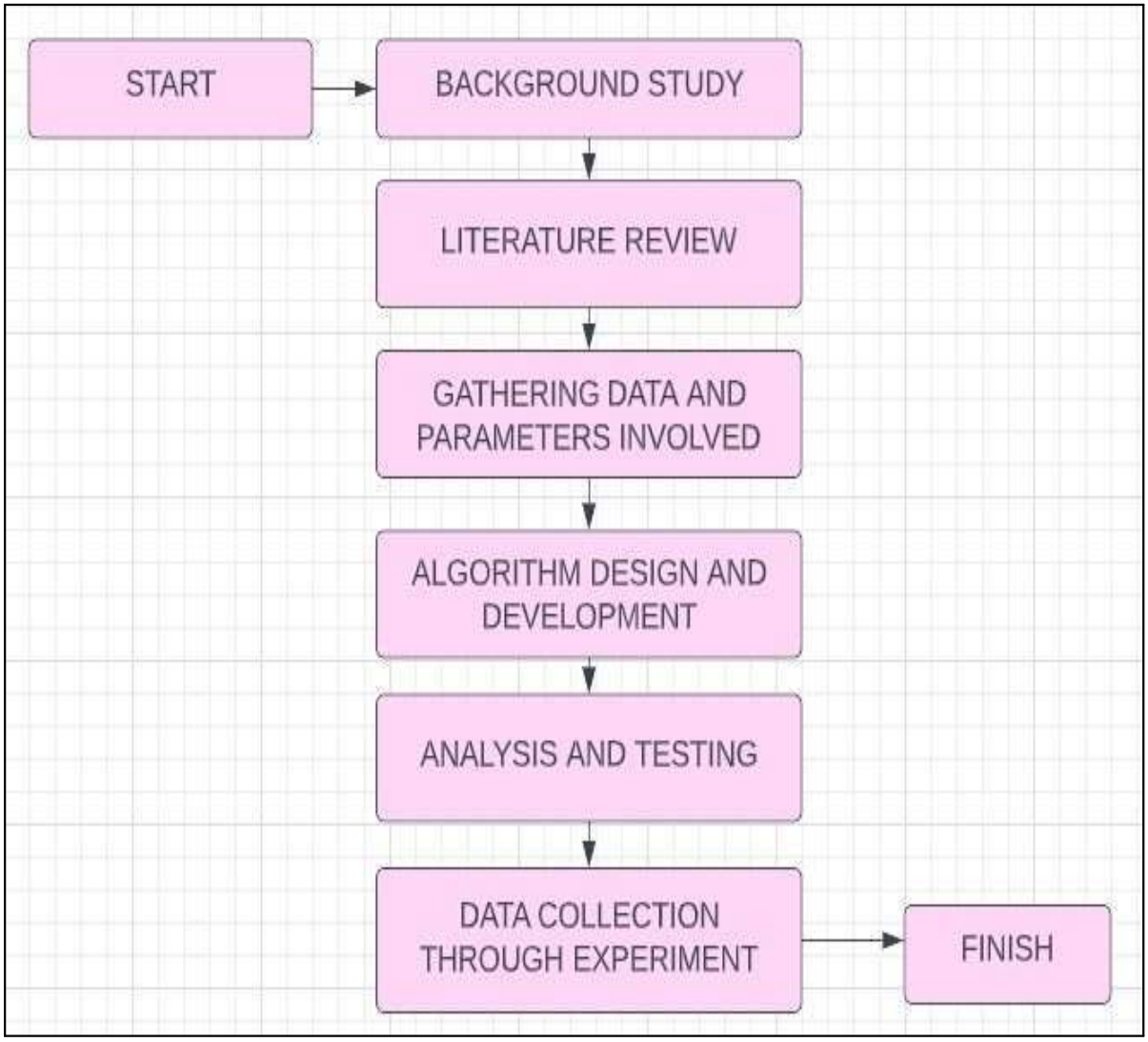


Figure 11: Flow Chart of Project Progress

Data Flow Diagrams

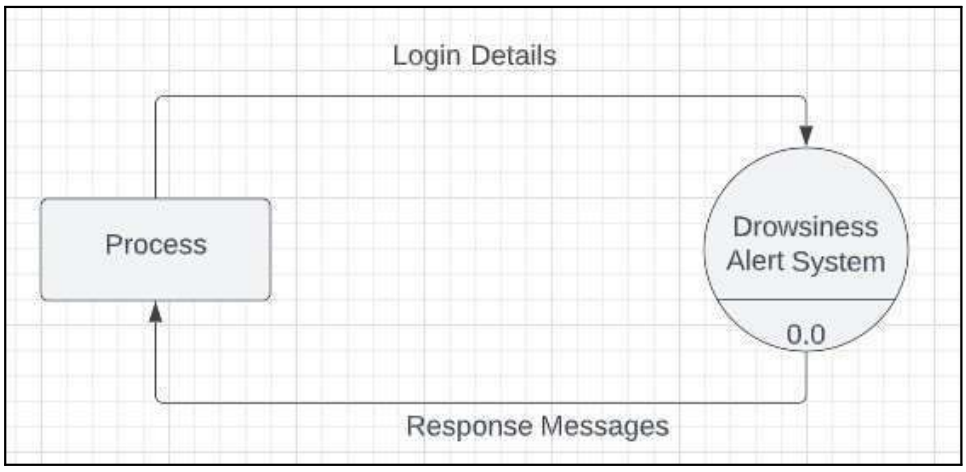


Figure 12: DFD level 0

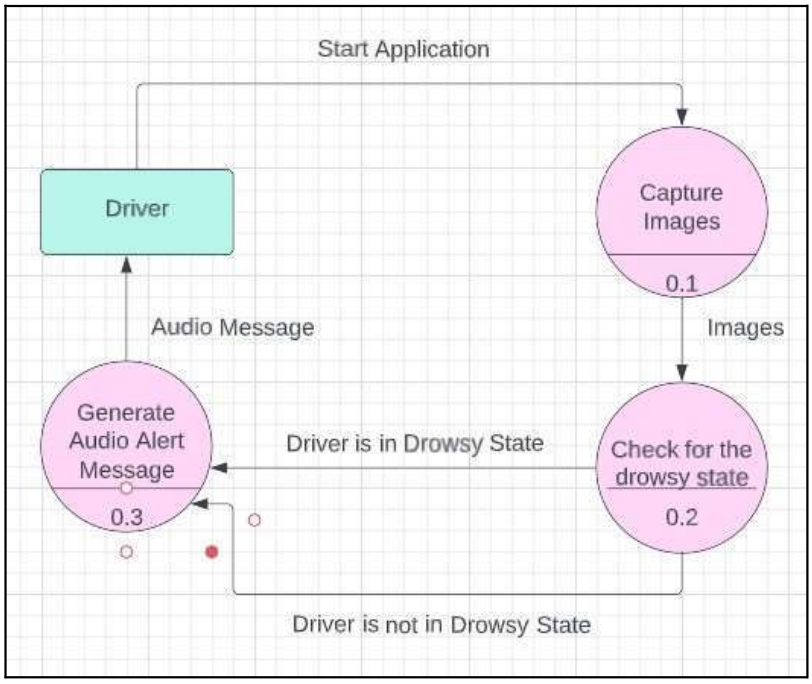


Figure 13: DFD level 1

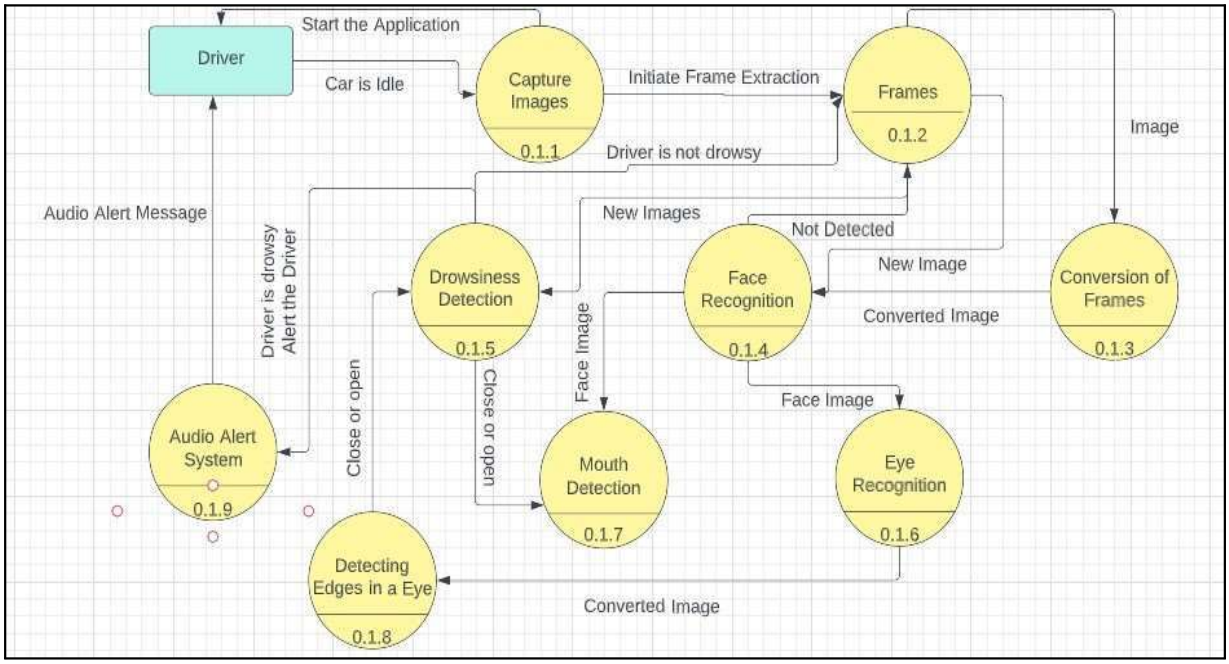


Figure 14: DFD level 2

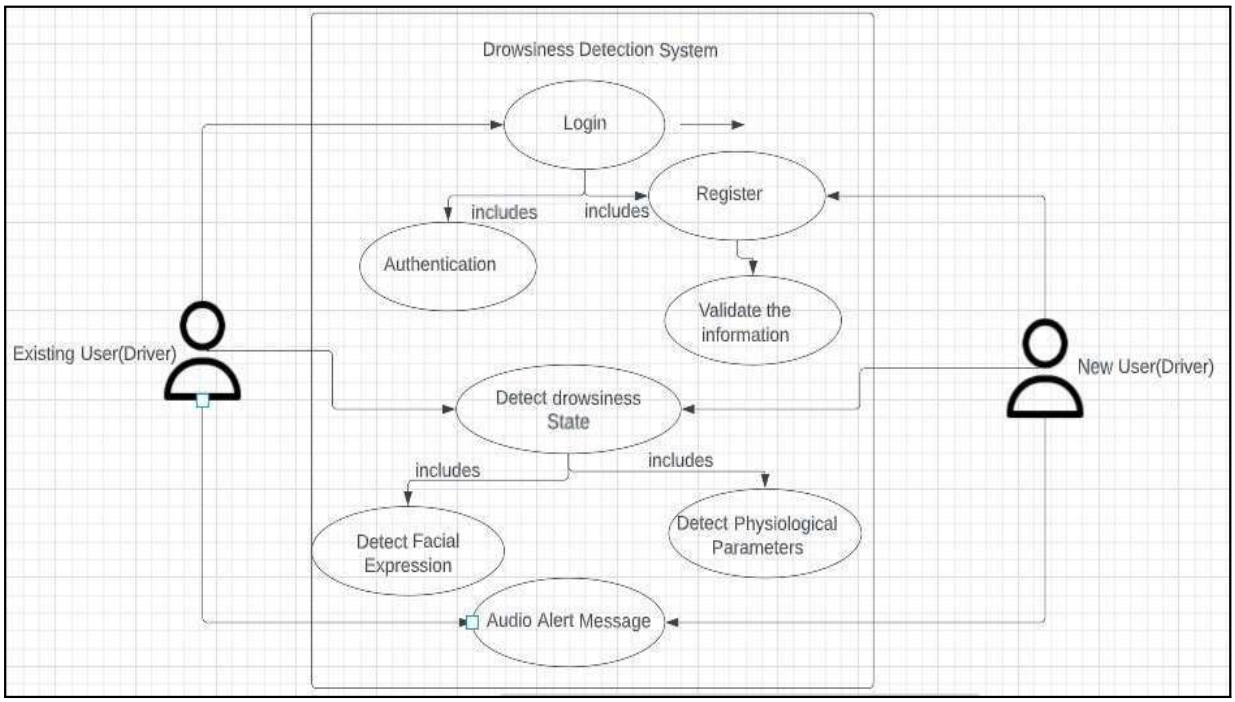


Figure 15: Use-case Diagram

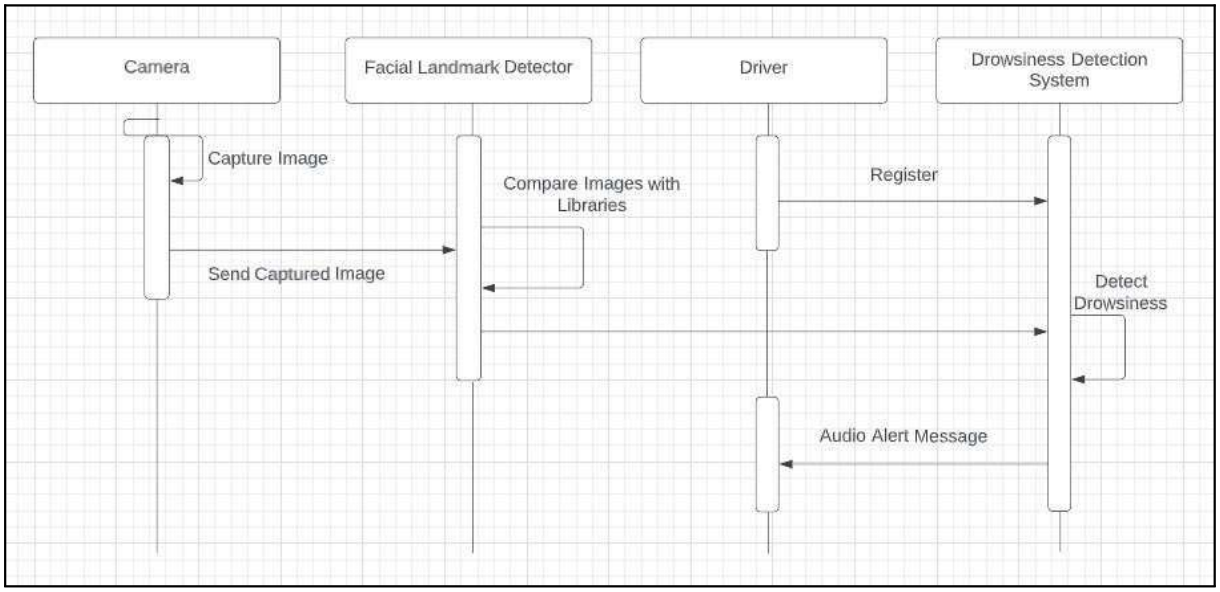


Figure 16: Sequence Diagram

3.5 Methodology

Various types of methods have been developed to examine sleepiness.

Physiological Level Approach:

This technique is an invasive method that uses electrodes to obtain information about pulse rate, heart rate, and brain activity. ECG is used to calculate heart rate variability and detect various states of drowsiness.

Behavior-Based Approach:

In this technique, a camera monitors a person's blink frequency, head position, etc. and alerts the person when these signs of drowsiness are detected.

The various technologies available are described below.

TensorFlow:

IT is an open source software library for data flow programming for various tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. Used for both research and production.

TensorFlow computations are represented as stateful data flow graphs. The name TensorFlow

comes from the operations such neural networks perform on multidimensional data arrays. These arrays are called "tensors".

Machine Learning:

Machine Learning is a form of programming that gives computers the ability to automatically learn from data without being explicitly programmed.

In other words, these programs change their behavior by learning from the data. Python is clearly one of the best machine learning languages. Python includes special machine learning libraries such as scipy, pandas, and numpy that are great for doing linear algebra and learning machine learning methods for kernels.

OpenCV:

OpenCV stands for Open Source Computer Vision. This is an open source, BSD licensed library containing hundreds of advanced computer vision algorithms optimized to use hardware acceleration. OpenCV is widely used for machine learning, image processing, image manipulation, etc. OpenCV has a modular structure. There are shared and static libraries and a CV name-space.

In a nutshell, an application uses OpenCV to easily load a bitmap file containing landscape images, perform a blending operation between the two images, and display one image behind another. be done.

Kivy:

Kivy is an open source Python library for developing mobile apps and other multi-touch application software with a natural user interface (NUI). It can run on Android, iOS, Linux, OS X, and Windows. Kivy is distributed under the terms of the MIT license and is free and open source software. Kivy is the main framework developed by the Kivy organization, along with Python for Android, Kivy iOS, and several other libraries used on all platforms.

CHAPTER-4

RESULTS ANALYSIS AND VALIDATION

4.1 Implementing Solution

In the early stages of this final project, it was intended to use MATLAB® to develop the algorithms for the system. Using MATLAB®, we have used a software toolbox to simplify the process of writing code. As a result of the experiment, the face, eyes and mouth were recognized.

4.2 Tools

A number of tools were used to detect faces and remove eye and mouth regions.

4.2.1 Logitech C310 web camera

Used as a tool to capture the driver's face before narrowing them around the eyes and mouth.

4.2.2 Video Cutter

Used to cut the video of the exam conducted by IRTE. Video must be cropped for use in MATLAB® software.

4.2.3 Format Factory Software

- Used to convert video by combining multiple video files to get one video.
- Used to convert video formats to meet MATLAB® requirements.
- Supports conversion of any video file format.

4.2.4 MATLAB® R2013a Software

[1] Used as one of the means of implementing algorithms for blink detection systems.

[2] Compatible with various programming languages such as C, C++ and Java.

[3] Computer Vision System Toolbox is used to provide algorithms and tools for computer vision and video processing design and simulation. The system includes algorithms for feature extraction, motion detection, object tracking, video processing, and video analysis.

4.3 Software Implementation

In the early stages of this 4th grade project, the intention was to use MATLAB® to develop the algorithms for the system.

4.3.1 Face Detection

To detect faces, the author uses an algorithm that is part of the Computer Vision Toolbox System, the Vision Cascade Detector. It creates a system object detector that uses the Viola-Jones method to detect objects. By default the detector is configured to detect faces.

For the face Detection it uses Haar feature-based cascade classifiers is an efficient object detection methodology planned by Paul Viola and Michael Jones in their paper, "Rapid Object Detection employing a Boosted Cascade of easy Features" in 2001. it's a machine learning based mostly approach wherever a cascade perform is trained from plenty of positive and negative pictures. it's then accustomed observe objects in alternative pictures.

Here we are going to work with face detection. Initially, the formula wants plenty of positive pictures (images of faces) and negative pictures (images while not faces) to coach the classifier. Then we'd like to extract options from it. For this, Haar options shown within the below image square measure used. they're rather like our convolutional kernel. every feature could be a single worth obtained by subtracting total of pixels below the white parallelogram from total of pixels below the black parallelogram.

A cascaded Adaboost classifier with the Haar-like functions is exploited to find out the face location. First, the compensated photograph is segmented into numbers of rectangle regions, at any position and scale inside the original photo because of the difference of facial feature, Haar-like function is green for real-time face detection. these can be calculated consistent with the difference of sum of pixel values within rectangle regions. The functions can be represented via the specific composition of the black place and white area. A cascaded Adaboost classifier is a strong classifier which is a mixture of numerous susceptible

classifiers. every susceptible classifier is skilled by way of Adaboost set of rules. If a candidate sample passes thru the cascaded Adaboost classifier, the face vicinity may be located. nearly all of face samples can skip via and nonface samples can be rejected

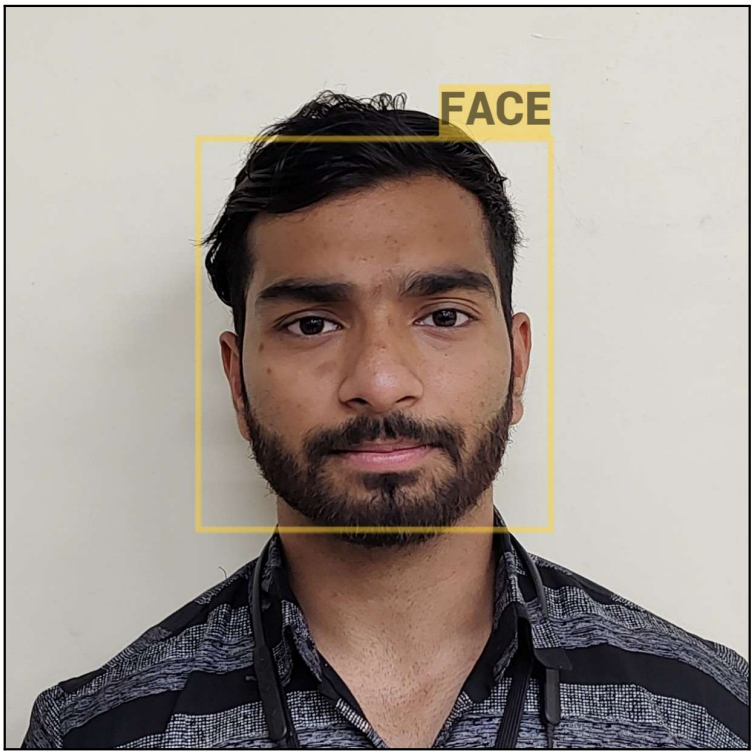


Figure 17: Face Detection Result

Face region detection properties are as follows:

1. Define and set a cascaded object detector using the constructor. The builder uses the built-in Viola-Jones algorithm to recognize the face, nose, eyes, mouth and upper body.

2. Play the video or selected image and run the face detector.

3. Draw a bounding box around the detected faces. The bounding box is the region of the desired detected face. This is the jump box around the face area.

4.3.2 Eye detection

In the system we've used facial landmark prediction for eye detection Facial landmarks square measure wont to localize and represent salient regions of the face, such as:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

Facial landmarks were successfully carried out to stand alignment, head pose estimation, face swapping, blink detection and much greater. within the context of facial landmarks, our intention is detecting critical facial systems on the face using shape prediction techniques. Detecting facial landmarks is therefore a two-step method:

- Localize the face inside the photo.
- stumble on the important thing facial systems on the face ROI.

Localize the face in the photograph: The face photo is localized by using Haar characteristic-based totally cascade classifiers which become discussed within the first step of our algorithm i.e. face detection, detect the key facial structures on the face ROI:

There are a selection of facial landmark detectors, however all strategies essentially try to localize and label the subsequent facial areas:

- Mouth
- right eyebrow
- Left eyebrow
- proper eye
- Left eye
- nose

The facial landmark detector blanketed in the dlib library is an implementation of the only Millisecond Face Alignment with an Ensemble of Regression bushes paper with the aid of

Kazemi and Sullivan (2014).

This approach starts by using using:

1. A schooling set of categorized facial landmarks on an image. these pictures are manually classified, specifying particular (x, y)-coordinates of areas surrounding every facial structure.

2. Priors, of more particularly, the possibility on distance among pairs of input pixels.

The pre-trained facial landmark detector in the dlib library is used to estimate the place of sixty eight (x, y)-coordinates that map to facial systems at the face.

Those annotations are part of the 68 factor iBUG 300-W dataset which the dlib facial landmark predictor became educated on. It's crucial to notice that different flavors of facial landmark detectors exist, consisting of the 194 point model that may be educated at the HELEN dataset irrespective of which dataset is used, the equal dlib framework may be leveraged to train a form predictor at the enter training facts.

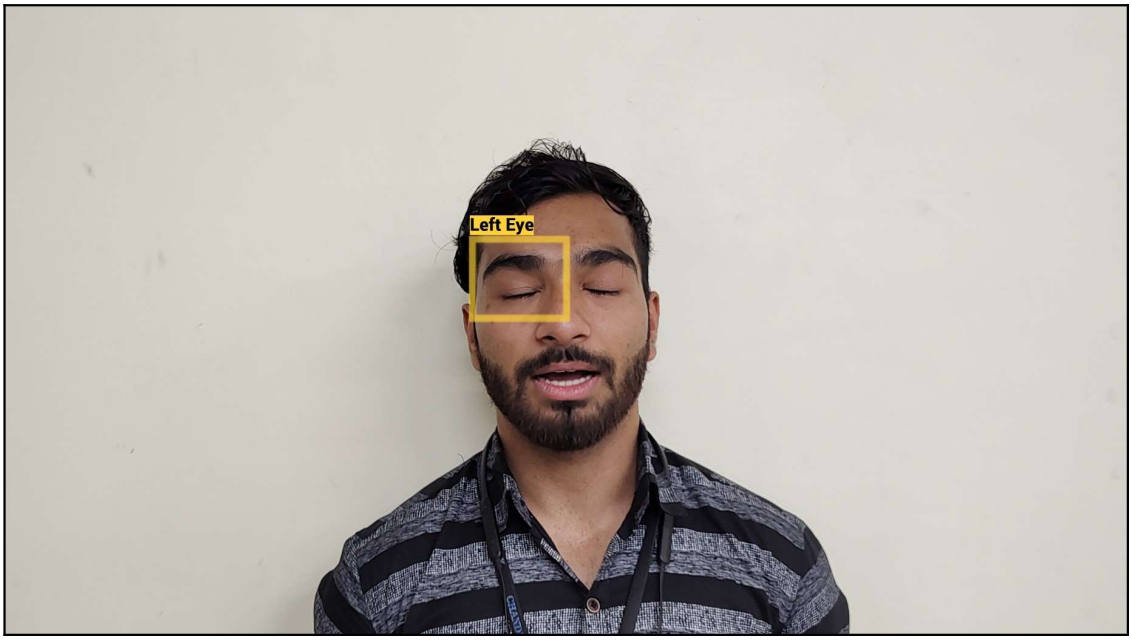


Figure 18: Eye Detection Result

Eye Aspected magnitude relation Calculation:

$$EAR = (||p2 - p6|| + ||p3 - p5||) / (2||p1 - p4||)$$

For every video frame, the attention landmarks ar detected. the attention ratio (EAR) between height and dimension of the attention is computed.

Eye area can be estimated based on optical flow, sparse tracking or inter-frame intensity discrimination and adaptive threshold. And finally, the decision is made whether the eyes are covered with eyelids or not. A different approach is to infer the state of eye opening from a single image, such as correlation matching with open and closed eye models, heuristic projection of horizontal or vertical image intensity over the eye region, parametric model suitable for eyelid detection. or using active form models. The main disadvantage of the previous approaches is that they usually implicitly impose too strict requirements on the layout, which means the relative face position (head orientation) of the camera, image resolution, lighting, motion dynamics, etc. In particular, heuristic methods that produce an image. intensity is likely to be very sensitive despite their real-time performance.

Therefore, we propose a simple but effective algorithm for eye blink detection using a modern facial landmark detector. A single scalar quantity is derived from the landmarks that reflects the level of eye opening. Finally, eye blinks are detected using an SVM classifier

trained on blink and non-blink patterns.

Finally, the choice for the attention state is formed supported EAR calculated within the previous step. If the distance is zero or is about to zero, the attention state is assessed as “closed” otherwise the attention state is known as “open”.

4.3.3 Mouth Detection

The purpose of Mouth Detection is to detect the symptoms of drowsiness, yawning. A cascade object detector that detects rectangular objects using the Viola-Jones algorithm was used for mouth detection. Here is the function for detecting the mouth region:

1. Use the constructor to define and configure the cascading object detector. The constructor uses the built-in Viola-Jones algorithm to detect faces, noses, eyes, mouths, and torsos.

2. Draw a bounding box around the detected mouth.

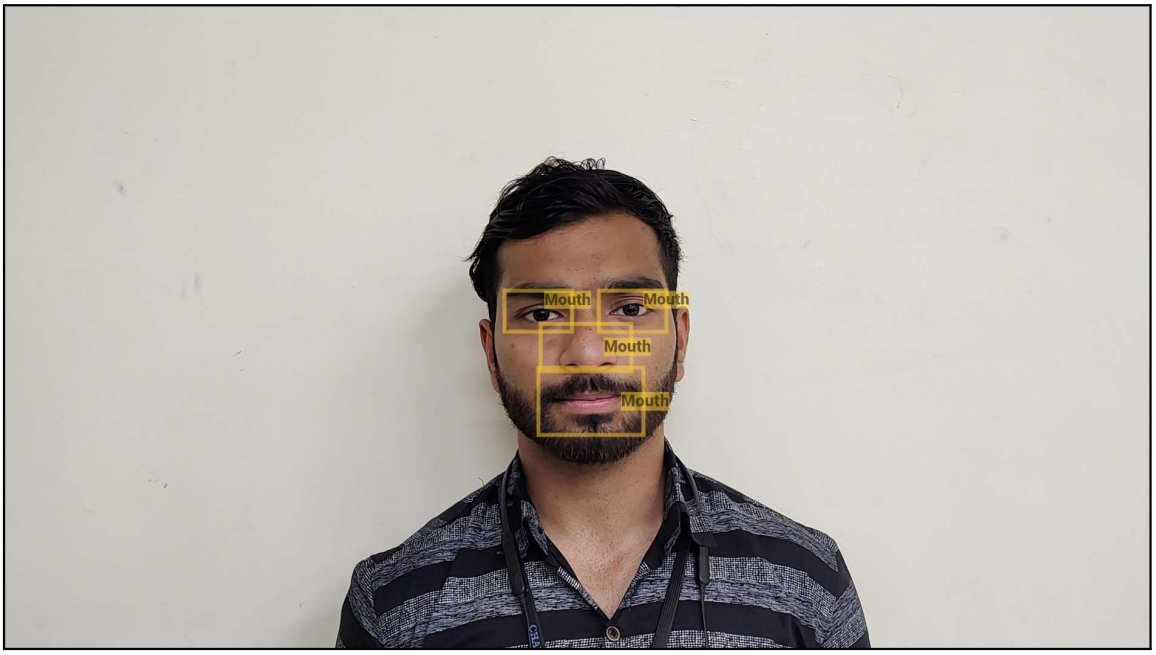


Figure 19: Mouth Detection Result

4.4 Software Modeling

After an algorithm is developed through experimentation, it must be refined to make the system meet the goals of the project. The algorithm successfully detects eyes and mouth in the video; so the result can be obtained.

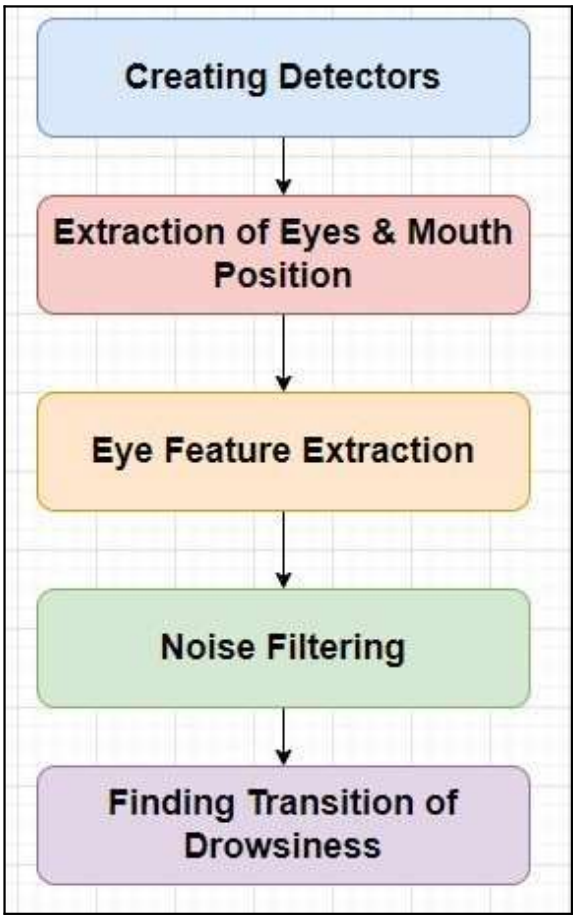


Figure 20: Flow Process of Detection Algorithm

4.4.1 Creating Detectors

In the first section, all cascade detectors for face, eyes and mouth were created. This is because the whole algorithm is related to other detection features.

4.4.2 Eye and Mouth Location Extraction

After creating the detector, the face bounding box, all eye and mouth regions are preserved. Each bounding box contains the parameters of the recognition part. A cascaded filter was

used to extract the detected facial features by preserving the centroid of the bounding box of the face. This is obtained by dividing the face bounding box (x, y) value by 2 and is marked with a red dashed line in the image. The value used and divided by 3.

Subtraction is done to ensure that the eye detection boundary is at the top of the face and the mouth detection bounding box is below the nose or at the bottom of the face. Figure 26 shows the resulting true positive expression.

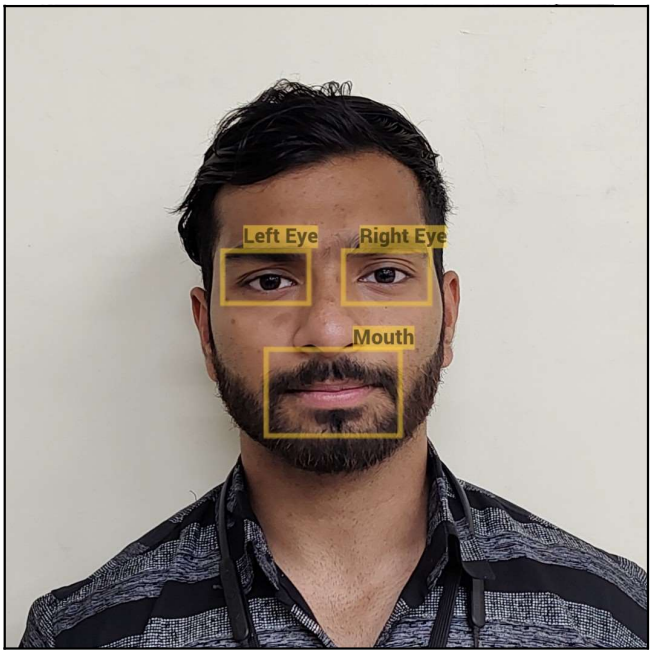


Figure 21: True Positive Eyes and Mouth Detection.

4.4.3 Eye Feature Extraction

After each eye bounding box is found, a histogram smoothing filter is applied to solve the problem of luminance variation. Each gray scale image, with one sample value for each pixel, contains only intensity data, with black being the weakest and white being the strongest. Luminescence is a measure of the wavelength-weighted power emitted by a light source in a given direction per unit solid angle. Histogram smoothing is a method of adjusting image intensity to improve contrast.

The properties seem directly related to closing the eyes. The amount of gradient in the eye image is supposed to be proportional to the aperture of the eyes. For example, when a person's eyes are open, the magnitude of the gradient may be greater than the gradient when

the eyes are closed. Here, the detection of the cumulative amount in the time domain informs us about the opening or closing of the eye.

4.4.4 Noise Filter

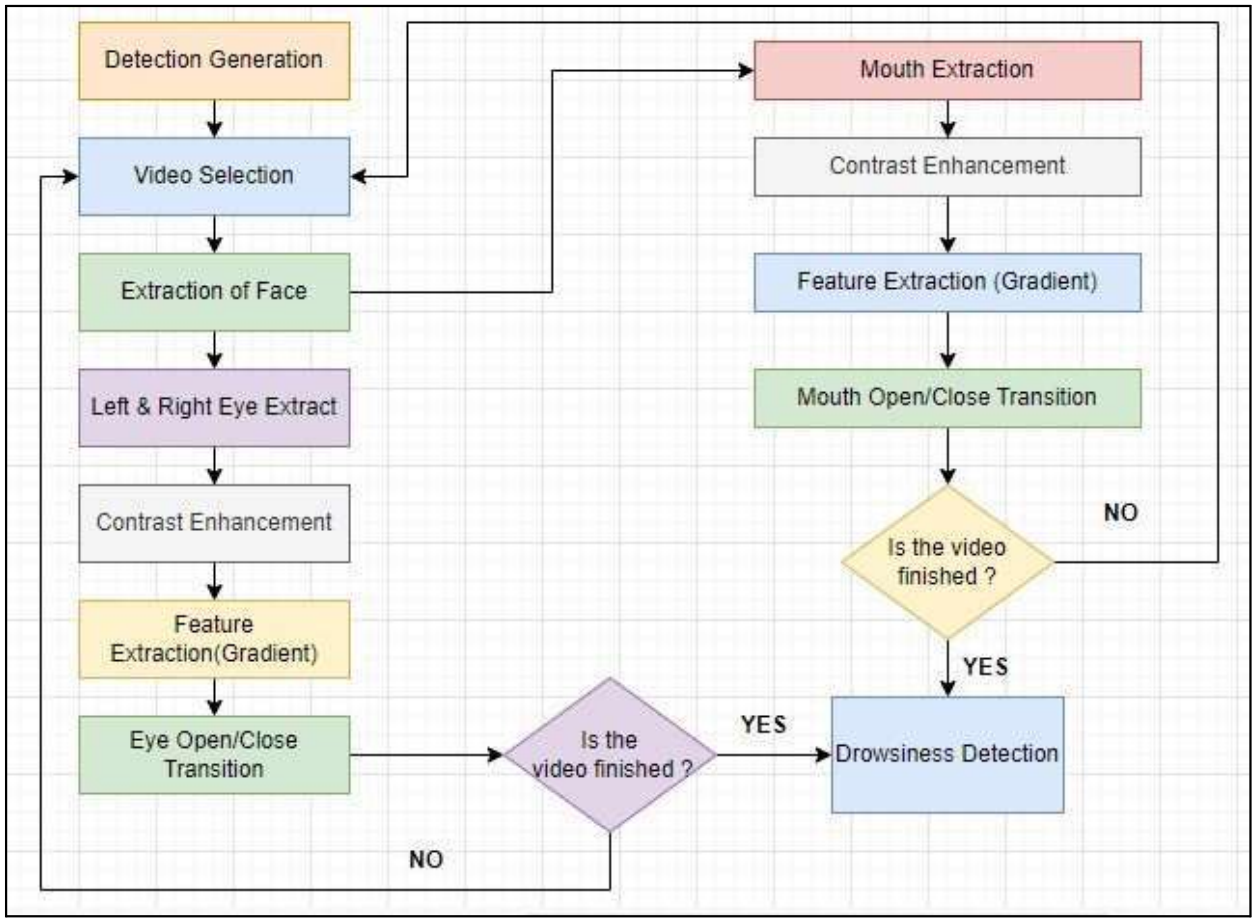


Figure 22: Flow Process of Drowsiness Detection System

During data collection, the algorithm uses a median filter to filter out noise so that the final plot is smooth. We then use the global parameter method to obtain the eye open to eye closed transition made by the product energy of each region. Finally, global parameters are derived using DoG. The same method is used in the mouth opening detection algorithm.

4.5 Modeling Results

After completing the modeling of the algorithm, we performed a simulation of the algorithm using the IRTE test procedure video. The video has been split into his 3 minute videos due to the length of the video which cannot be run in MATLAB® due to insufficient memory in the processor. Each video will not contain more than 400 images per video. He has 10 participants to complete the driving test, but he only picks up one participant to show the results.

Based on the experiment and modeling, this project will successfully detect each eye and mouth detection sleep system to achieve the goals of the final year project. This system is reliable because it detects the sleepy state transition, where algorithms detect every eye-mouth exchange in every frame second. Therefore, a webcam-based drowsiness detection system using MATLAB® software was successful. This chapter presents all the information and results of the simulation algorithm.

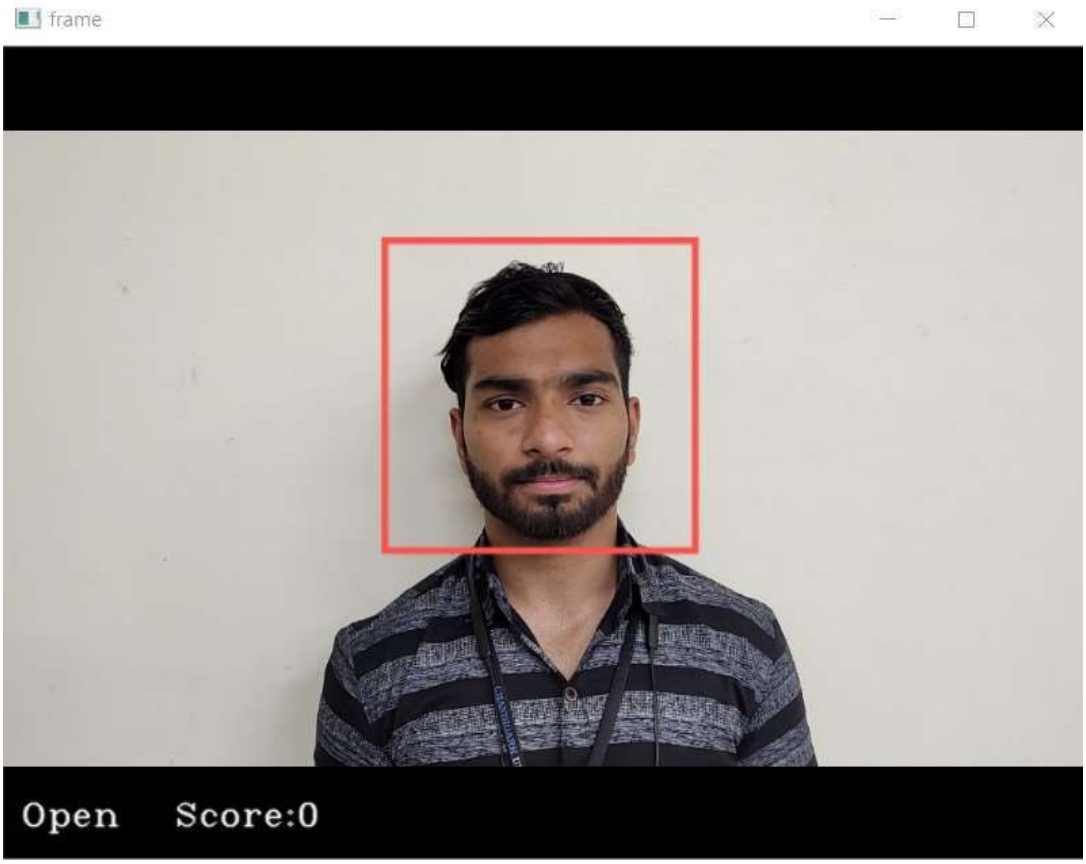


Figure 23: Flow Process of Drowsiness Detection System

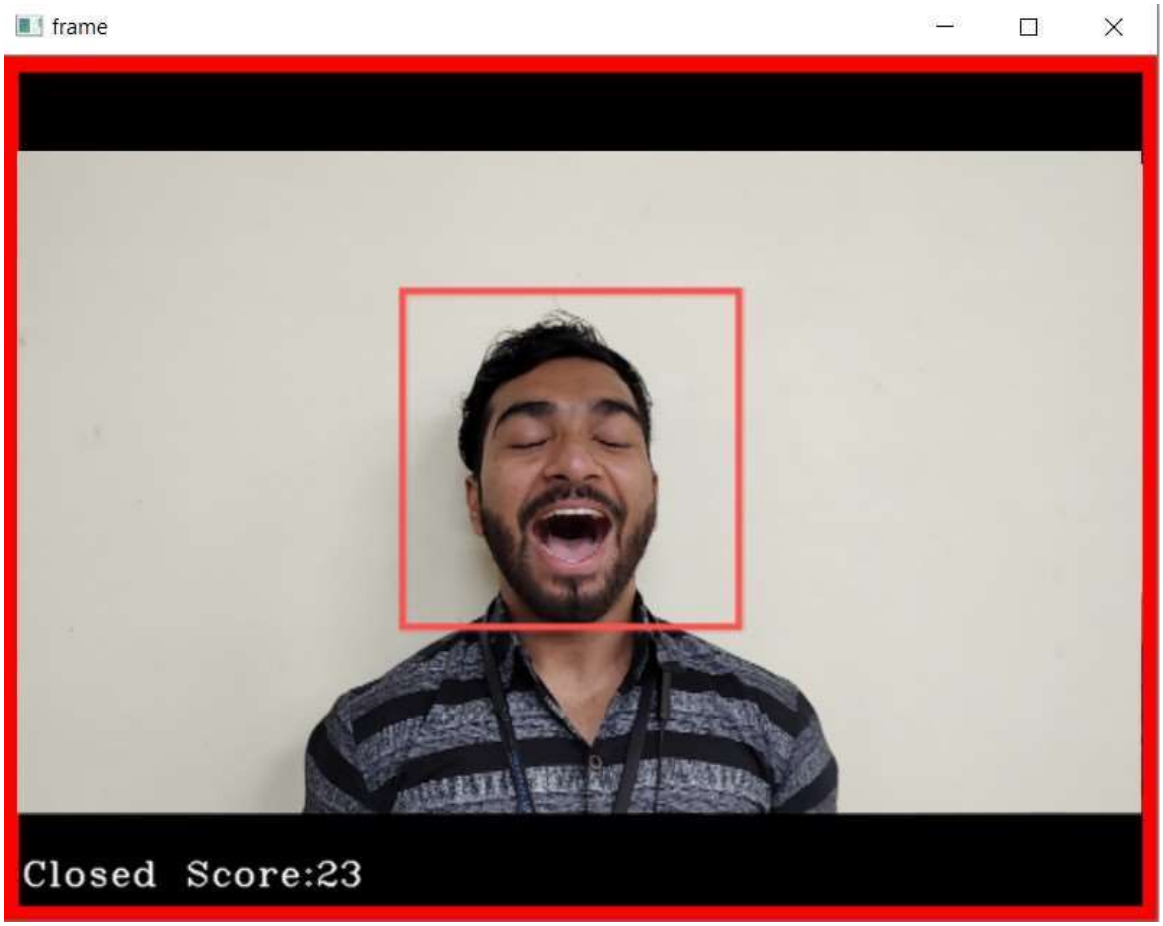


Figure 24: Flow Process of Drowsiness Detection System

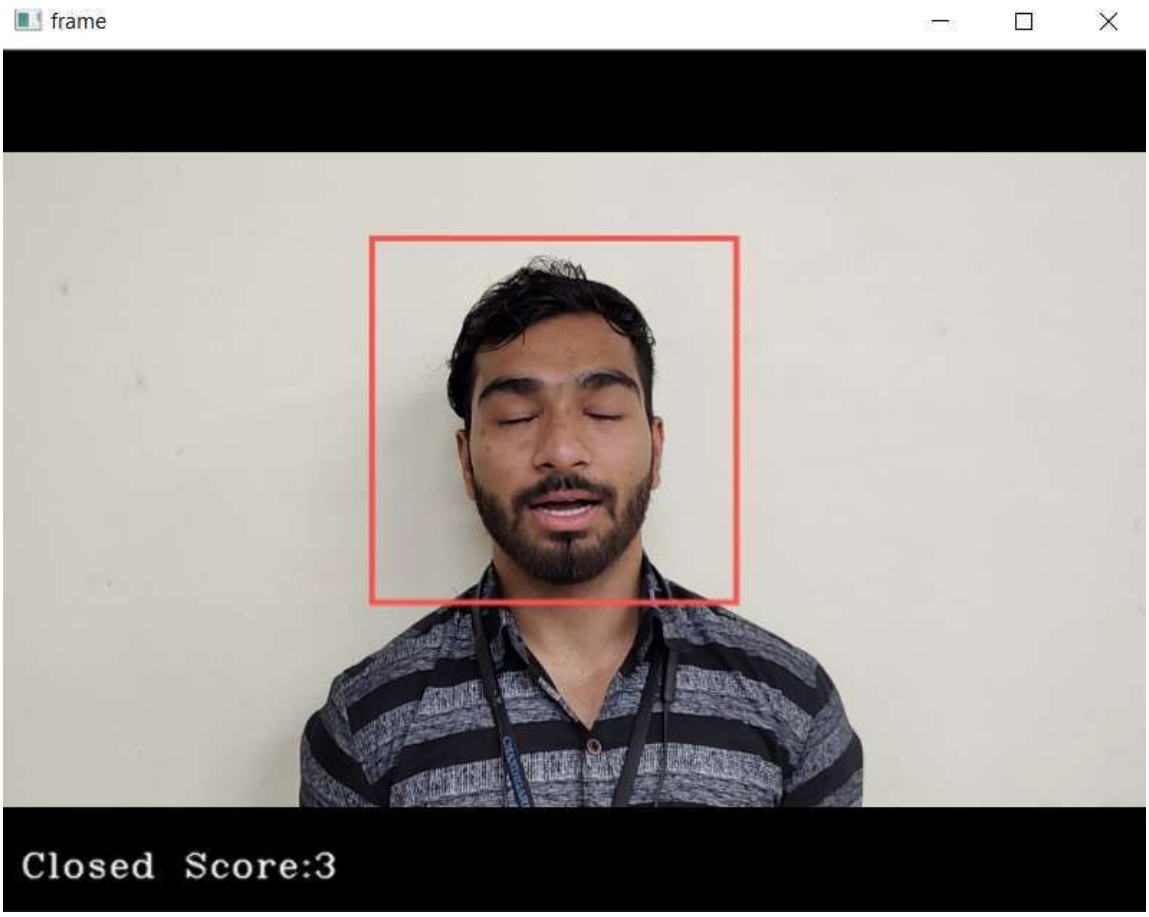


Figure 25: Flow Process of Drowsiness Detection System

CHAPTER 5.
CONCLUSION AND FUTURE WORK

5.1. Conclusion

A real-time blink detection algorithm was presented. Quantitatively proved this Cascade classifier based on hair features and facial landmark detector based on regression It is accurate enough to reliably estimate positive face images and eye openness. Robust in low image quality (mostly low image resolution) and in the wild.

Limitations:

Use of glasses: Difficulty in recognizing the condition when wearing glasses eye. Since it is highly dependent on light, the reflection from the glasses is closed eyes than open eyes. This requires close eye contact with the camera to avoid light.

Multiple face issue: Camera may detect more faces if multiple faces are displayed in the window. The number of faces may appear as unwanted output. Because different facial conditions are different. Therefore, only the driver's face should be within the camera's range. Similarly, Recognition speed decreases due to multi-faceted operation.

5.2. Future work

Our model is designed for detection of drowsy state of eye and give and alert signal or warning in the form of audio alarm. But the response of driver after being warned may not be enough to stop causing the accident meaning that if the driver is slow in responding towards the warning signal then accident may occur. Hence to avoid this we can design and fit a motor driven system and synchronize it with the warning signal so that the vehicle will slow down after getting the warning signal automatically.

We can also provide the user with an Android application which will provide with the information of his/her drowsiness level during any journey. The user will know Normal state, Drowsy State, the number of times blinked the eyes according to the number of frames captures.

APPENDIX

Code Written for Image Processing:

1) Dataset Image Generation:

```
import os
import shutil
import glob
from tqdm import tqdm

Raw_DIR= r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\MRL Eye Data\mrlEyes_2018_01'
for dirpath, dirname, filenames in os.walk(Raw_DIR):
    for i in tqdm([f for f in filenames if f.endswith('.png')]):
        if i.split('_')[4]=='0':
            shutil.copy(src=dirpath+'/'+i, dst=r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\Close Eyes')

        elif i.split('_')[4]=='1':
            shutil.copy(src=dirpath+'/'+i, dst=r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\Open Eyes')
```

2) Model Training:

```
import tensorflow as tf
from tensorflow.keras.applications import InceptionV3
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout,Input,Flatten,Dense,MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator

tf.test.is_gpu_available()

Batchsize=8
```

```
train_datagen= ImageDataGenerator(rescale=1./255, rotation_range=0.2,shear_range=0.2,
    zoom_range=0.2,width_shift_range=0.2,
    height_shift_range=0.2, validation_split=0.2)

train_data= train_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron Intership
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\train',

target_size=(80,80),batch_size=batchsize,class_mode='categorical',subset='training' )

validation_data= train_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron
Intership Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\train',
                        target_size=(80,80),batch_size=batchsize,class_mode='categorical',
subset='validation')

test_datagen = ImageDataGenerator(rescale=1./255)

test_data = test_datagen.flow_from_directory(r'D:\Python37\Projects\iNeuron Intership
Projects\CV_Driver_Drowsiness_Detection\MRL Eye Data\Prepared_Data\test',
                        target_size=(80,80),batch_size=batchsize,class_mode='categorical')

bmodel = InceptionV3(include_top=False, weights='imagenet',
input_tensor=Input(shape=(80,80,3)))
hmodel = bmodel.output
hmodel = Flatten()(hmodel)
hmodel = Dense(64, activation='relu')(hmodel)
hmodel = Dropout(0.5)(hmodel)
hmodel = Dense(2,activation= 'softmax')(hmodel)

model = Model(inputs=bmodel.input, outputs= hmodel)
for layer in bmodel.layers:
    layer.trainable = False

model.summary()
```

```
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping,
ReduceLROnPlateau

checkpoint = ModelCheckpoint(r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\models\model.h5',
                        monitor='val_loss',save_best_only=True,verbose=3)

earlystop = EarlyStopping(monitor = 'val_loss', patience=7, verbose= 3,
restore_best_weights=True)

learning_rate = ReduceLROnPlateau(monitor= 'val_loss', patience=3, verbose= 3, )

callbacks=[checkpoint,earlystop,learning_rate]

model.compile(optimizer='Adam', loss='categorical_crossentropy',metrics=['accuracy'])

model.fit_generator(train_data,steps_per_epoch=train_data.samples//batchsize,
                    validation_data=validation_data,
                    validation_steps=validation_data.samples//batchsize,
                    callbacks=callbacks,
                     epochs=5)

acc_tr, loss_tr = model.evaluate_generator(train_data)
print(acc_tr)
print(loss_tr)

acc_vr, loss_vr = model.evaluate_generator(validation_data)
print(acc_vr)
print(loss_vr)

acc_test, loss_test = model.evaluate_generator(test_data)
print(acc_tr)
print(loss_tr)
```

3) Data Prep Drowsiness:

```
import os
import glob
import shutil
import random
from tqdm import tqdm

raw_data = '/home/delixus/Desktop/drowsiness_detection/mrlEyes_2018_01'
for dirpath, dirname, filename in os.walk(raw_data):
    for file in tqdm([f for f in filename if f.endswith('.png')]):
        if file.split('_')[4] == '0':
            path='/home/delixus/Desktop/drowsiness_detection/data/train/closed'
            if not os.path.exists(path):
                os.makedirs(path)
            shutil.copy(src=dirpath + '/' + file, dst= path)
        elif file.split('_')[4] == '1':
            path='/home/delixus/Desktop/drowsiness_detection/data/train/open'
            if not os.path.exists(path):
                os.makedirs(path)
            shutil.copy(src=dirpath + '/' + file, dst= path)

def create_test_closed(source, destination, percent):
    path, dirs, files_closed = next(os.walk(source))
    file_count_closed = len(files_closed)
    percentage = file_count_closed * percent
    to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

    for f in enumerate(to_move):
        if not os.path.exists(destination):
            os.makedirs(destination)
        shutil.move(f[1], destination)
    print(f'moved {int(percentage)} images to the destination successfully.')
```

```
def create_test_open(source, destination, percent):
    path, dirs, files_open = next(os.walk(source))
    file_count_open = len(files_open)
    percentage = file_count_open * percent
    to_move = random.sample(glob.glob(source + "/*.png"), int(percentage))

    for f in enumerate(to_move):
        if not os.path.exists(destination):
            os.makedirs(destination)
        shutil.move(f[1], destination)
    print(f'moved {int(percentage)} images to the destination successfully.')

create_test_closed('/home/delixus/Desktop/drowsiness_detection/data/train/closed',
                   '/home/delixus/Desktop/drowsiness_detection/data/test/closed',
                   0.2)
create_test_open('/home/delixus/Desktop/drowsiness_detection/data/train/open',
                   '/home/delixus/Desktop/drowsiness_detection/data/test/open',
                   0.2)
```

4) Main File:

```
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
import numpy as np
from pygame import mixer

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
model = load_model(r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\models\model.h5')

mixer.init()
```

```
sound= mixer.Sound(r'D:\Python37\Projects\iNeuron Intership Projects\
CV_Driver_Drowsiness_Detection\alarm.wav')
cap = cv2.VideoCapture(0)
Score = 0
while True:
    ret, frame = cap.read()
    height,width = frame.shape[0:2]
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= face_cascade.detectMultiScale(gray, scaleFactor= 1.2, minNeighbors=3)
    eyes= eye_cascade.detectMultiScale(gray, scaleFactor= 1.1, minNeighbors=1)

    cv2.rectangle(frame, (0,height-50),(200,height),(0,0,0),thickness=cv2.FILLED)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,pt1=(x,y),pt2=(x+w,y+h), color= (255,0,0), thickness=3 )

    for (ex,ey,ew,eh) in eyes:
        #cv2.rectangle(frame,pt1=(ex,ey),pt2=(ex+ew,ey+eh), color= (255,0,0), thickness=3 )

        # preprocessing steps
        eye= frame[ey:ey+eh,ex:ex+w]
        eye= cv2.resize(eye,(80,80))
        eye= eye/255
        eye= eye.reshape(80,80,3)
        eye= np.expand_dims(eye,axis=0)
        # preprocessing is done now model prediction
        prediction = model.predict(eye)

        # if eyes are closed
        if prediction[0][0]>0.30:
            cv2.putText(frame,'closed',(10,height-
20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255
),
```

```
                thickness=1,lineType=cv2.LINE_AA)
        cv2.putText(frame,'Score'+str(Score),(100,height-
20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255
),
                thickness=1,lineType=cv2.LINE_AA)
        Score=Score+1
        if(Score>15):
          try:
            sound.play()
          except:
            pass

      # if eyes are open
      elif prediction[0][1]>0.90:
        cv2.putText(frame,'open',(10,height-
20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255
),
                thickness=1,lineType=cv2.LINE_AA)
        cv2.putText(frame,'Score'+str(Score),(100,height-
20),fontFace=cv2.FONT_HERSHEY_COMPLEX_SMALL,fontScale=1,color=(255,255,255
),
                thickness=1,lineType=cv2.LINE_AA)
        Score = Score-1
        if (Score<0):
          Score=0



  cv2.imshow('frame',frame)
  if cv2.waitKey(33) & 0xFF==ord('q'):
    break

cap.release()
cv2.destroyAllWindows()
```

5) Drowsiness Detection:

```
import cv2
import os
import load_model
import numpy as np
import mixer
import time

mixer.init()
sound = mixer.Sound('alarm.wav')

face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
leye = cv2.CascadeClassifier('haar cascade files\haarcascade_lefteye_2splits.xml')
reye = cv2.CascadeClassifier('haar cascade files\haarcascade_righteye_2splits.xml')



lbl=['Close','Open']

model = load_model('models/cnncat2.h5')
path = os.getcwd()
cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
count=0
score=0
thicc=2
rpred=[99]
lpred=[99]

while(True):
    ret, frame = cap.read()
    height,width = frame.shape[:2]
```

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

faces = face.detectMultiScale(gray,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
left_eye = leye.detectMultiScale(gray)
right_eye =  reye.detectMultiScale(gray)

cv2.rectangle(frame, (0,height-50) , (200,height) , (0,0,0) , thickness=cv2.FILLED )

for (x,y,w,h) in faces:
    cv2.rectangle(frame, (x,y) , (x+w,y+h) , (100,100,100) , 1 )

for (x,y,w,h) in right_eye:
    r_eye=frame[y:y+h,x:x+w]
    count=count+1
    r_eye = cv2.cvtColor(r_eye,cv2.COLOR_BGR2GRAY)
    r_eye = cv2.resize(r_eye,(24,24))
    r_eye= r_eye/255
    r_eye=  r_eye.reshape(24,24,-1)
    r_eye = np.expand_dims(r_eye,axis=0)
    rpred = model.predict_classes(r_eye)
    if(rpred[0]==1):
        lbl='Open'
    if(rpred[0]==0):
        lbl='Closed'
    break

for (x,y,w,h) in left_eye:
    l_eye=frame[y:y+h,x:x+w]
    count=count+1
    l_eye = cv2.cvtColor(l_eye,cv2.COLOR_BGR2GRAY)
    l_eye = cv2.resize(l_eye,(24,24))
    l_eye= l_eye/255
    l_eye=l_eye.reshape(24,24,-1)
```

```
        l_eye = np.expand_dims(l_eye,axis=0)
        lpred = model.predict_classes(l_eye)
        if(lpred[0]==1):
            lbl='Open'
        if(lpred[0]==0):
            lbl='Closed'
        break

    if(rpred[0]==0 and lpred[0]==0):
        score=score+1
        cv2.putText(frame,"Closed",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)
    # if(rpred[0]==1 or lpred[0]==1):
    else:
        score=score-1
        cv2.putText(frame,"Open",(10,height-20), font, 1,(255,255,255),1,cv2.LINE_AA)



    if(score<0):
        score=0
    cv2.putText(frame,'Score:'+str(score),(100,height-20), font, 1,
(255,255,255),1,cv2.LINE_AA)
    if(score>15):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
```

```
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

6) Result:

```
#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



#Initializing the face detector and landmark detector
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("/content/drive/MyDrive/collab/Driver-Drowsiness-
Detection-master/dlib_shape_predictor/shape_predictor_68_face_landmarks.dat")

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

```
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


pic = '/content/drive/MyDrive/collab/ss.jpeg';
pic2 = cv2.imread('photo.jpg', cv2.IMREAD_UNCHANGED)
# pic2 = cv2.imread(pic, cv2.IMREAD_UNCHANGED)
gray = cv2.cvtColor(pic2, cv2.COLOR_BGR2GRAY)

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

                face_frame = pic2.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):
                status="SLEEPING !!!"
                color = (255,0,0)

elif(left_blink==1 or right_blink==1):
        sleep=0
        active=0
        drowsy+=1
        if(drowsy):
                status="Drowsy !"
                color = (0,0,255)

else:
        drowsy=0
        sleep=0
        active+=1
        if(active):
                status="Active :)"
```

```
                color = (0,255,0)

        cv2.putText(pic2, status, (100,100), cv2.FONT_HERSHEY_SIMPLEX, 1.2,
color,3)

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

cv2_imshow(pic2)
cv2_imshow(face_frame)
```

REFERENCES

[1] Rahul Atul Bhope, "Computer Vision based totally drowsiness detection for motorized cars with Web Push Notifications", IEEE 4th International Conference on Internet of Things, IEEE, Ghaziabad, India, 2019.

[2] Jasper S. Wijnands, Jason Thompson, Kerry A. Nice, Gideon D. P Aschwanden & Mark Stevenson, "Real-time tracking of driver drowsiness on cell systems the use of 3D neural networks", Neural Computing and Applications, 2019.

[3] Chris Schwarz, John Gaspar, Thomas Miller & Reza Yousefian, "The detection of drowsiness the usage of a driver monitoring machine" , in Journal of Traffic Injury Prevention (Taylor and Francis Online), 2019.

[4] Aditya Ranjan, Karan Vyas, Sujay Ghadge, Siddharth Patel, Suvarna Sanjay Pawar, "Driver Drowsiness Detection System Using Computer Vision.", in International Research Journal of Engineering and Technology(IRJET), 2020.

[5] B.Mohana, C.M.Sheela Rani, "Drowsiness Detection Based on Eye Closure and Yawning Detection", in International Research Journal of Engineering and Technology(IRJET), 2019.

[6] Driver Alert Control (DAC). (2016, Feb 10). Retrieved from http://assist.Volvocars.Com/united kingdom/motors/Pages/owners-manual.Aspx? Mc=Y555&my=2015&sw=14w20&article=2e82f6fc0d1139c2c0a801e800329d4e

[7] Z. Mardi, S. N. Ashtiani, and M. Mikaili, "EEG-based totally drowsiness detection for safe driving the use of chaotic capabilities and statistical tests," Journal of Medical Signals and Sensors, vol. 1, pp. A hundred thirty–137, 2011.

[8] T. Danisman, I.M. Bilasco, C. Djeraba and N. Ihaddadene, "Drowsy driving force detection machine using eye blink styles," Universite Lille 1 & Telecom Lille 1, Marconi, France, 2010.

[9] B. Hariri, S. Abtahi, S. Shirmohammadi, and L. Martel, "A yawning size method to discover driver drowsiness," Distributed and Collaborative Virtual Environments Research Laboratory, University of Ottawa, Ottawa, ON, Canada, 2011.

[10] L. Li, Y. Chen and Z. Li, "Yawning detection for tracking driver fatigue based totally on cameras", Proceedings of the 12th International IEEE Conference Intelligent Transportation Systems., pp. 1-6, Oct. 2009.

[11] S. Abtahi, B. Hariri and S. Shirmohammadi, "Driver drowsiness tracking based totally on yawning detection", Proceedings of the IEEE International Control, Measurement and Instrumentation (CMI), IEEE, pp. 1-4, May 2011.

[12] X. Fan, B. Yin, and Y. Sun, "Yawning detection for monitoring driving force fatigue", Proceedings of the International Conference on Machine Learning and Cybernet, vol. 2, pp. 664-668, Aug. 2007. Thirteen. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, "Squeezenet: Alexnet-degree accuracy with 50x fewer parameters

and <zero.5Mb model length", Proceedings of the International Conference of Learning Representations, pp. 1-13, 2017.

[13] K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint face detection and alignment the usage of multitask cascaded convolutional networks", IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499-1503, Oct. 2016.

[14] D. B. Lucas and T. Kanade, "An iterative photograph registration method with an application to stereo vision", Proceedings of the 7th International Joint Conference on Artificial Intelligence, vol. 2, pp. 674-679, 1981.

[15] D. S. Bolme, J. R. Beveridge, B. A. Draper and Y. M. Lui, "Visual object monitoring the usage of adaptive correlation filters", Proceedings of the IEEE Computer Society Conference on Computer Vision Pattern Recognition., pp. 2544-2550, Jun. 2010.

[16] F. J. Henriques, R. Caseiro, P. Martins and J. Batista, "Exploiting the circulant structure of tracking-by means of-detection with kernels", Proceedings of the European Conference Computer Vision, pp. 702-715, 2012.

[17] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter out tracker with feature integration", Proceedings of the European Conference Computer Vision, pp. 254-265, 2015.

[18] M. Danelljan, G. Häger, F. S. Khan and M. Felsberg, "Discriminative scale area tracking", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 8, pp. 1561-1575, Aug. 2017.

[19] N. Wang and D.-Y. Yeung, "Learning a deep compact image representation for visual monitoring", Proceedings of the twenty sixth International Conference on Neural Information Processing Systems., vol. 1, pp. 809-817, 2013.

[20]Maven machines, “Maven co-pilot,” http://mavenmachines.Com/co-pilot/.

[21] S. Hu and G. Zheng, “Driver drowsiness detection with eyelid associated parameters through support vector gadget,” Expert Systems with Applications, vol. 36, no. 4, pp. 7651–7658, 2009.

[22] Evgeniy Abdulin and Oleg Komogortsev, “User eye fatigue detection via eye motion behavior,” in Proceedings of the thirty third Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems. ACM, 2015, pp. 1265–1270.

[23] M. Zhe, Y. Xin-Ping, and W. Chao-Zhong, “Driving fatigue identification approach based totally on physiological signals,” in Seventh International Conference of Chinese Transportation Professionals, 2008, pp. 341–352.

[24] Ye Sun, Xiong Yu, Jim Berilla, Zhen Liu, and Guangxi Wu, “An in-car physiological sign monitoring system for driving force fatigue detection,” in 3rd International Conference on Road Safety and Simulation, 2011.

[25] C. Liu, S. Hosking, and M. Lenn, "Predicting motive force drowsiness the use of vehicle measures: Recent insights and destiny challenges," J. Safety Res., vol. 40, no. 1, pp. 239–245, 2009.

[26] S. Otmani, T. Pebayle, J. Roge, and A. Muzet, "Effect of using duration and partial sleep deprivation on next alertness and overall performance of vehicle drivers," Physiol. Behav, vol. Eighty four, no. 1, pp. 715–724, 2005.

[27] Caterpillar Inc., "Cat smartband optimizing shift schedules to reduce hazard," https: //protection.Cat.Com/cda/documents/4788162/7/AEXQ1521-01%20CSBp.C20Onesheet.Pdf.

[28] Ries Simons, Marieke Martens, Jan Ramaekers, Arno Krul, Ineke Kl¨opping-Ketelaars, and Gisela Skopp, "Effects of dexamphetamine with and with out alcohol on simulated using," Psychopharmacology, vol. 222, no. Three, pp. 391–399, 2012.

[29] Divya Das, Shiyu Zhou, and Jonah D Lee, "Differentiating alcohol-prompted using behavior using guidance wheel signals," Intelligent Transportation Systems, IEEE Transactions on, vol. Thirteen, no. Three, pp. 1355–1368, 2012.

[30] Monique AJ Mets, Esther Kuipers, Lieke M Senerpont Domis, Maartje Leenders, Berend Olivier, and Joris C Verster, "Effects of alcohol on toll road driving in the stisim driving simulator," Human Psychopharmacology: Clinical and Experimental, vol. 26, no. 6, pp. 434–439, 2011. Esra Vural, Video based totally detection of driving force fatigue, Ph.D. Thesis, Sabanci University, 2009.

[31] J. Gomez-Clapers and R. Casanella, "A rapid and easy-to-use ecg acquisition and coronary heart fee tracking gadget the use of a wireless steering wheel," IEEE Sensors, vol. 12, no. Three, pp. 610–616, 2012.

[32] J. B. Hyun, S. C. Gih, K. Ko, and S. P. Kwang, "A clever health tracking chair for nonintrusive size of organic alerts.," IEEE Transactions on Information Technology in Biomedicine, vol. 16, no. 1, pp. One hundred fifty–158, 2012.

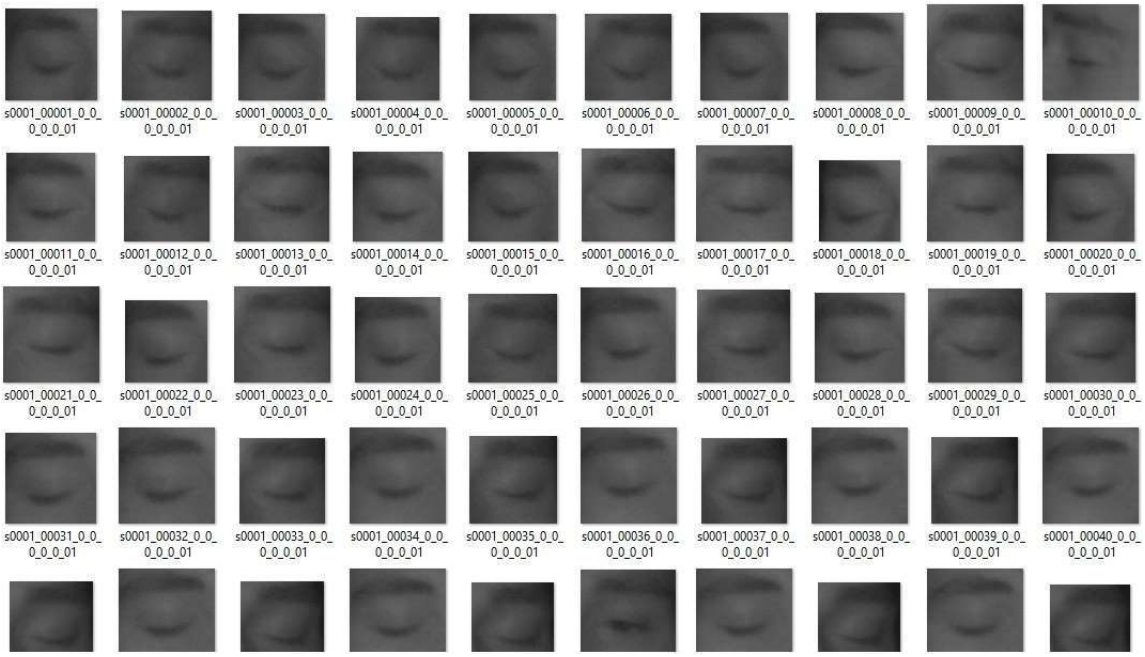
[33] Shuyan Hu and Gangtie Zheng, "Driver drowsiness detection with eyelid associated parameters through aid vector machine," Expert Systems with Applications, vol. 36, no. 4, pp. 7651–7658, 2009.

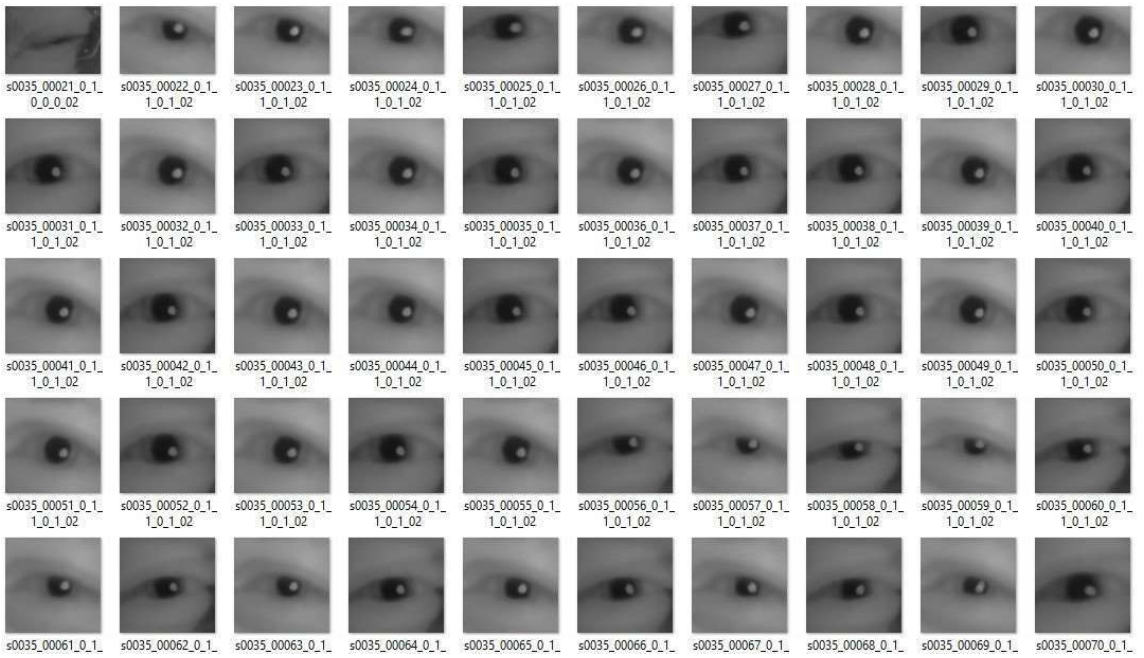
[34] Evangelia Portouli, Evangelos Bekiaris, Vassilis Papakostopoulos, and Nicos Maglaveras, "On-street experiment for amassing driving behavioural facts of sleepy drivers," Somnologie-Schlafforschung und Schlafmedizin, vol. 11, no. Four, pp. 259–267, 2007.

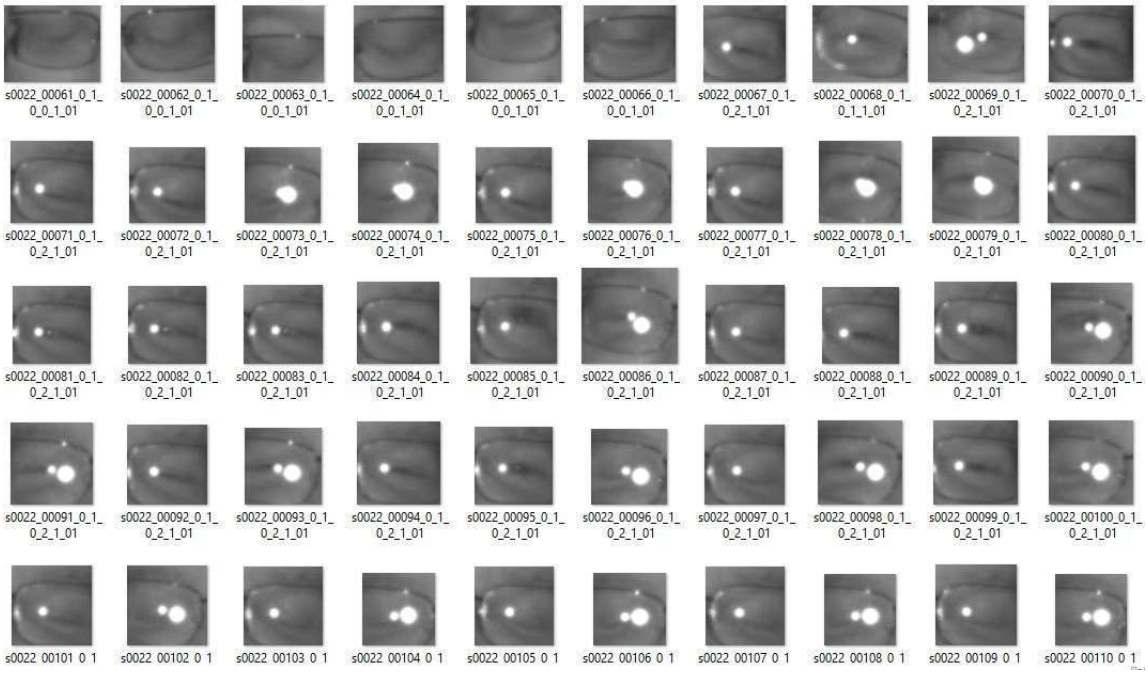
[35] Michael Ingre, TorbjORn ¨ °AKerstedt, BjORn Peters, Anna Anund, and G ¨ ORan Keck- ¨ lund, "Subjective sleepiness, simulated driving performance and blink length: examining man or woman variations," Journal of sleep research, vol. 15, no. 1, pp. 47–53, 2006.

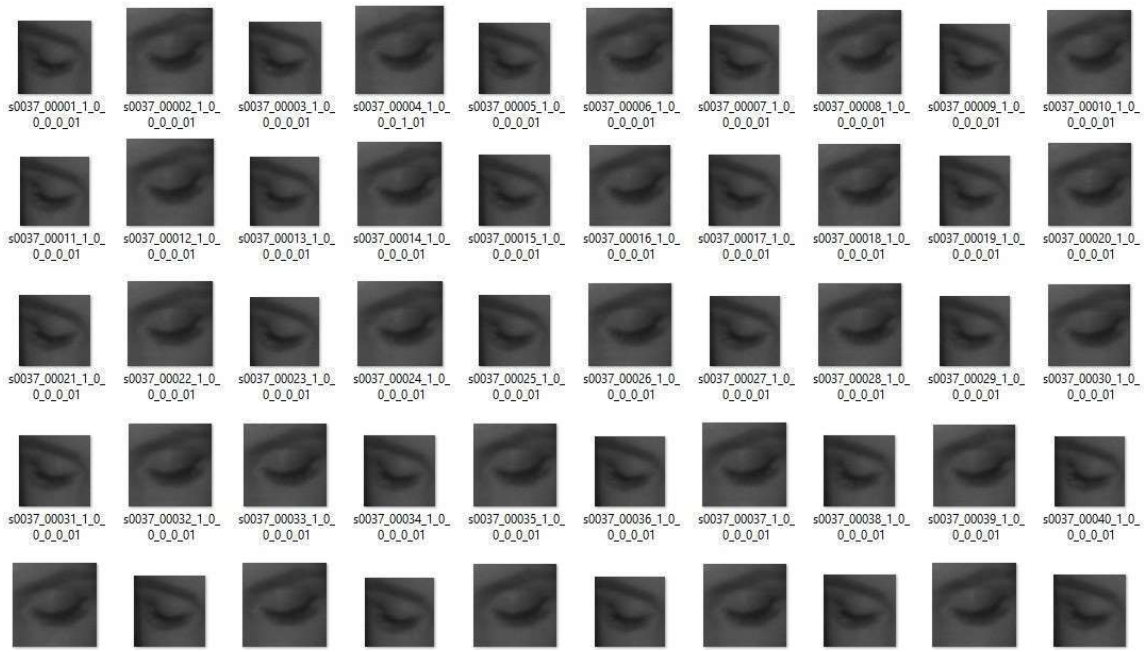
USER MANUAL

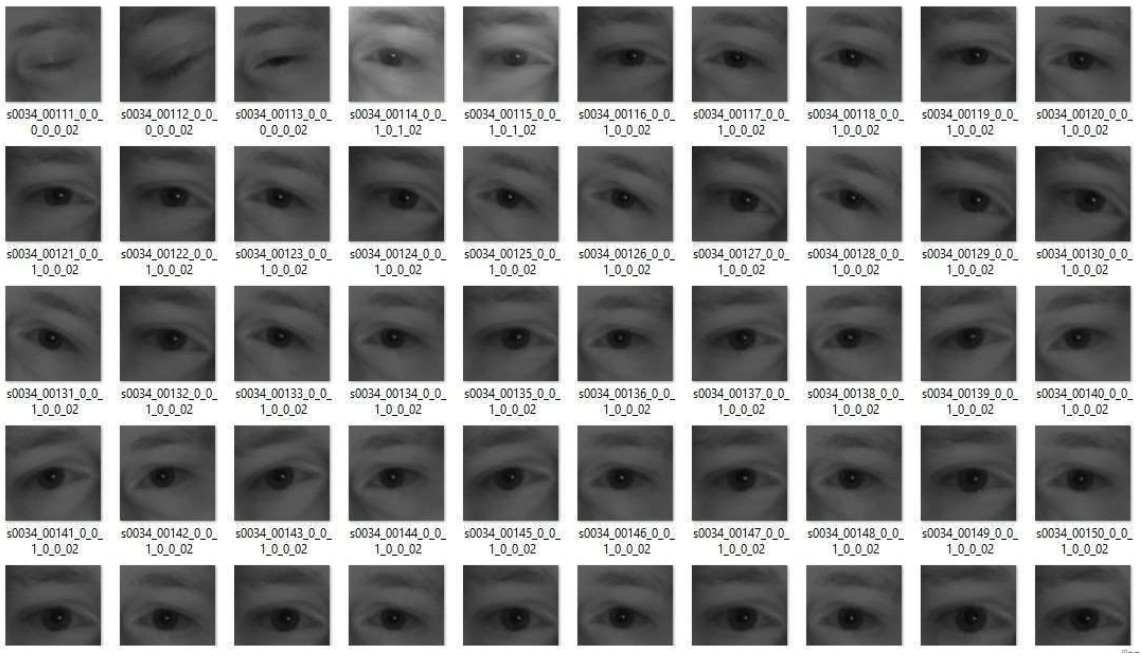
Dataset:

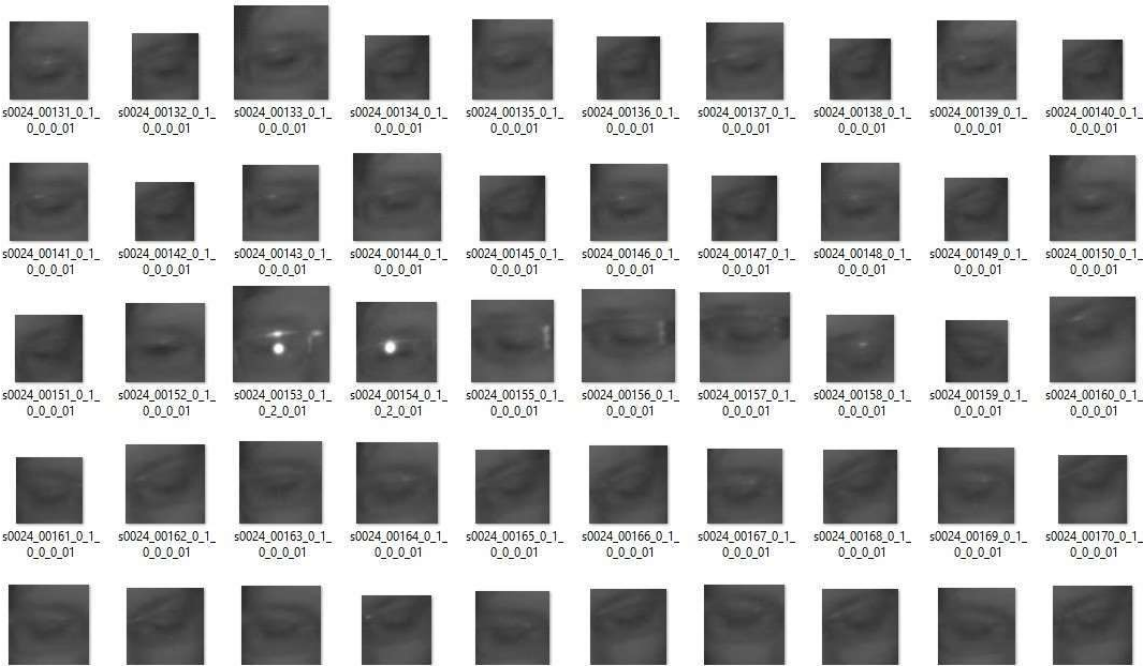





s0022_00061_0_1_0_0_1_01
s0022_00062_0_1_0_0_1_01
s0022_00063_0_1_0_0_1_01
s0022_00064_0_1_0_0_1_01
s0022_00065_0_1_0_0_1_01
s0022_00066_0_1_0_0_1_01
s0022_00067_0_1_0_2_1_01
s0022_00068_0_1_0_1_1_01
s0022_00069_0_1_0_2_1_01
s0022_00070_0_1_0_2_1_01
s0022_00071_0_1_0_2_1_01
s0022_00072_0_1_0_2_1_01
s0022_00073_0_1_0_2_1_01
s0022_00074_0_1_0_2_1_01
s0022_00075_0_1_0_2_1_01
s0022_00076_0_1_0_2_1_01
s0022_00077_0_1_0_2_1_01
s0022_00078_0_1_0_2_1_01
s0022_00079_0_1_0_2_1_01
s0022_00080_0_1_0_2_1_01
s0022_00081_0_1_0_2_1_01
s0022_00082_0_1_0_2_1_01
s0022_00083_0_1_0_2_1_01
s0022_00084_0_1_0_2_1_01
s0022_00085_0_1_0_2_1_01
s0022_00086_0_1_0_2_1_01
s0022_00087_0_1_0_2_1_01
s0022_00088_0_1_0_2_1_01
s0022_00089_0_1_0_2_1_01
s0022_00090_0_1_0_2_1_01
s0022_00091_0_1_0_2_1_01
s0022_00092_0_1_0_2_1_01
s0022_00093_0_1_0_2_1_01
s0022_00094_0_1_0_2_1_01
s0022_00095_0_1_0_2_1_01
s0022_00096_0_1_0_2_1_01
s0022_00097_0_1_0_2_1_01
s0022_00098_0_1_0_2_1_01
s0022_00099_0_1_0_2_1_01
s0022_00100_0_1_0_2_1_01
s0022 00101 0 1
s0022 00102 0 1
s0022 00103 0 1
s0022 00104 0 1
s0022 00105 0 1
s0022 00106 0 1
s0022 00107 0 1
s0022 00108 0 1
s0022 00109 0 1
s0022 00110 0 1


s0037_00001_1_0_0_0_0_01
s0037_00002_1_0_0_0_0_01
s0037_00003_1_0_0_0_0_01
s0037_00004_1_0_0_0_1_01
s0037_00005_1_0_0_0_0_01
s0037_00006_1_0_0_0_0_01
s0037_00007_1_0_0_0_0_01
s0037_00008_1_0_0_0_0_01
s0037_00009_1_0_0_0_0_01
s0037_00010_1_0_0_0_0_01
s0037_00011_1_0_0_0_0_01
s0037_00012_1_0_0_0_0_01
s0037_00013_1_0_0_0_0_01
s0037_00014_1_0_0_0_0_01
s0037_00015_1_0_0_0_0_01
s0037_00016_1_0_0_0_0_01
s0037_00017_1_0_0_0_0_01
s0037_00018_1_0_0_0_0_01
s0037_00019_1_0_0_0_0_01
s0037_00020_1_0_0_0_0_01
s0037_00021_1_0_0_0_0_01
s0037_00022_1_0_0_0_0_01
s0037_00023_1_0_0_0_0_01
s0037_00024_1_0_0_0_0_01
s0037_00025_1_0_0_0_0_01
s0037_00026_1_0_0_0_0_01
s0037_00027_1_0_0_0_0_01
s0037_00028_1_0_0_0_0_01
s0037_00029_1_0_0_0_0_01
s0037_00030_1_0_0_0_0_01
s0037_00031_1_0_0_0_0_01
s0037_00032_1_0_0_0_0_01
s0037_00033_1_0_0_0_0_01
s0037_00034_1_0_0_0_0_01
s0037_00035_1_0_0_0_0_01
s0037_00036_1_0_0_0_0_01
s0037_00037_1_0_0_0_0_01
s0037_00038_1_0_0_0_0_01
s0037_00039_1_0_0_0_0_01
s0037_00040_1_0_0_0_0_01


s0034_00111_0_0_0_0_0_02
s0034_00112_0_0_0_0_0_02
s0034_00113_0_0_0_0_0_02
s0034_00114_0_0_1_0_1_02
s0034_00115_0_0_1_0_1_02
s0034_00116_0_0_1_0_0_02
s0034_00117_0_0_1_0_0_02
s0034_00118_0_0_1_0_0_02
s0034_00119_0_0_1_0_0_02
s0034_00120_0_0_1_0_0_02
s0034_00121_0_0_1_0_0_02
s0034_00122_0_0_1_0_0_02
s0034_00123_0_0_1_0_0_02
s0034_00124_0_0_1_0_0_02
s0034_00125_0_0_1_0_0_02
s0034_00126_0_0_1_0_0_02
s0034_00127_0_0_1_0_0_02
s0034_00128_0_0_1_0_0_02
s0034_00129_0_0_1_0_0_02
s0034_00130_0_0_1_0_0_02
s0034_00131_0_0_1_0_0_02
s0034_00132_0_0_1_0_0_02
s0034_00133_0_0_1_0_0_02
s0034_00134_0_0_1_0_0_02
s0034_00135_0_0_1_0_0_02
s0034_00136_0_0_1_0_0_02
s0034_00137_0_0_1_0_0_02
s0034_00138_0_0_1_0_0_02
s0034_00139_0_0_1_0_0_02
s0034_00140_0_0_1_0_0_02
s0034_00141_0_0_1_0_0_02
s0034_00142_0_0_1_0_0_02
s0034_00143_0_0_1_0_0_02
s0034_00144_0_0_1_0_0_02
s0034_00145_0_0_1_0_0_02
s0034_00146_0_0_1_0_0_02
s0034_00147_0_0_1_0_0_02
s0034_00148_0_0_1_0_0_02
s0034_00149_0_0_1_0_0_02
s0034_00150_0_0_1_0_0_02

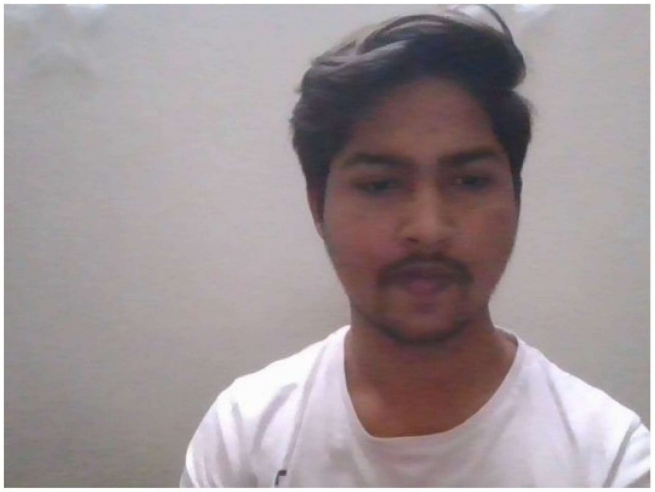

s0024_00131_0_1_0_0_0_01
s0024_00132_0_1_0_0_0_01
s0024_00133_0_1_0_0_0_01
s0024_00134_0_1_0_0_0_01
s0024_00135_0_1_0_0_0_01
s0024_00136_0_1_0_0_0_01
s0024_00137_0_1_0_0_0_01
s0024_00138_0_1_0_0_0_01
s0024_00139_0_1_0_0_0_01
s0024_00140_0_1_0_0_0_01
s0024_00141_0_1_0_0_0_01
s0024_00142_0_1_0_0_0_01
s0024_00143_0_1_0_0_0_01
s0024_00144_0_1_0_0_0_01
s0024_00145_0_1_0_0_0_01
s0024_00146_0_1_0_0_0_01
s0024_00147_0_1_0_0_0_01
s0024_00148_0_1_0_0_0_01
s0024_00149_0_1_0_0_0_01
s0024_00150_0_1_0_0_0_01
s0024_00151_0_1_0_0_0_01
s0024_00152_0_1_0_0_0_01
s0024_00153_0_1_0_2_0_01
s0024_00154_0_1_0_2_0_01
s0024_00155_0_1_0_0_0_01
s0024_00156_0_1_0_0_0_01
s0024_00157_0_1_0_0_0_01
s0024_00158_0_1_0_0_0_01
s0024_00159_0_1_0_0_0_01
s0024_00160_0_1_0_0_0_01
s0024_00161_0_1_0_0_0_01
s0024_00162_0_1_0_0_0_01
s0024_00163_0_1_0_0_0_01
s0024_00164_0_1_0_0_0_01
s0024_00165_0_1_0_0_0_01
s0024_00166_0_1_0_0_0_01
s0024_00167_0_1_0_0_0_01
s0024_00168_0_1_0_0_0_01
s0024_00169_0_1_0_0_0_01
s0024_00170_0_1_0_0_0_01

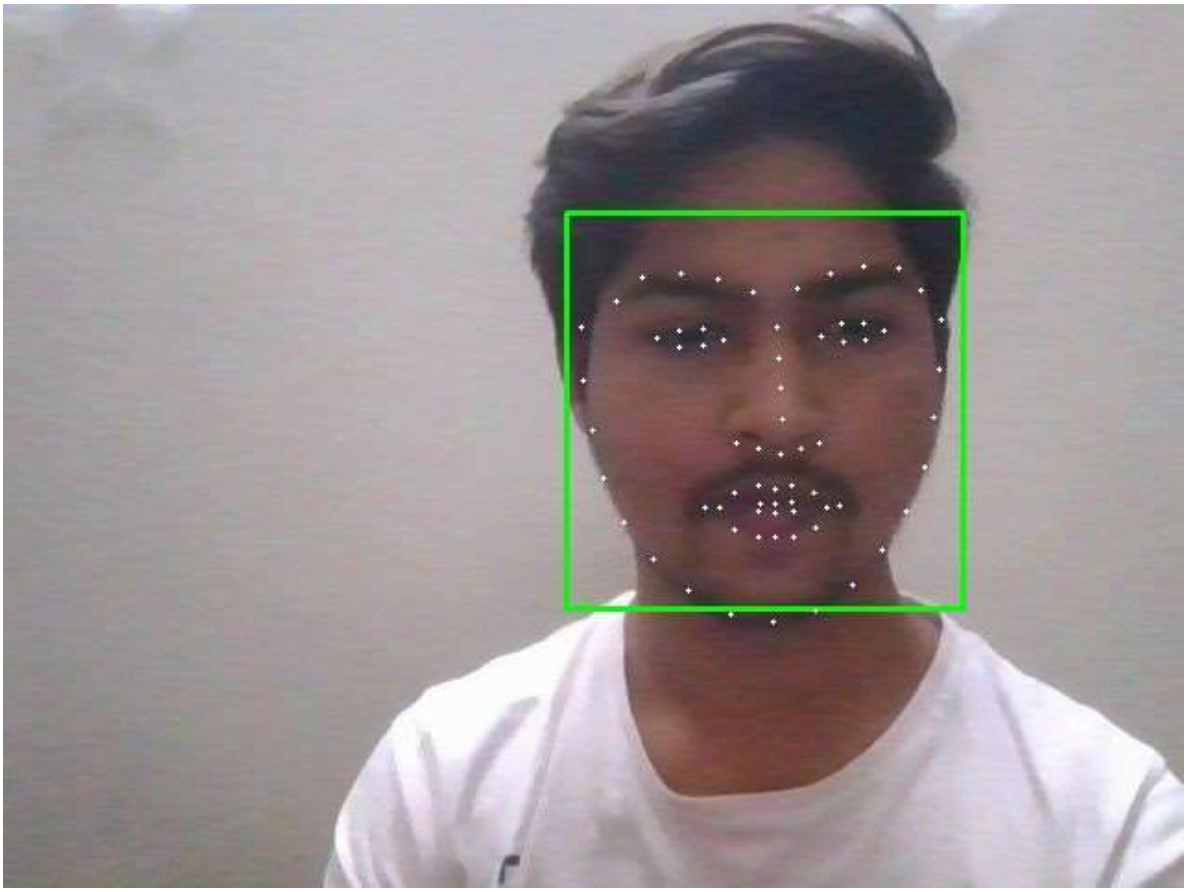
In [15]:

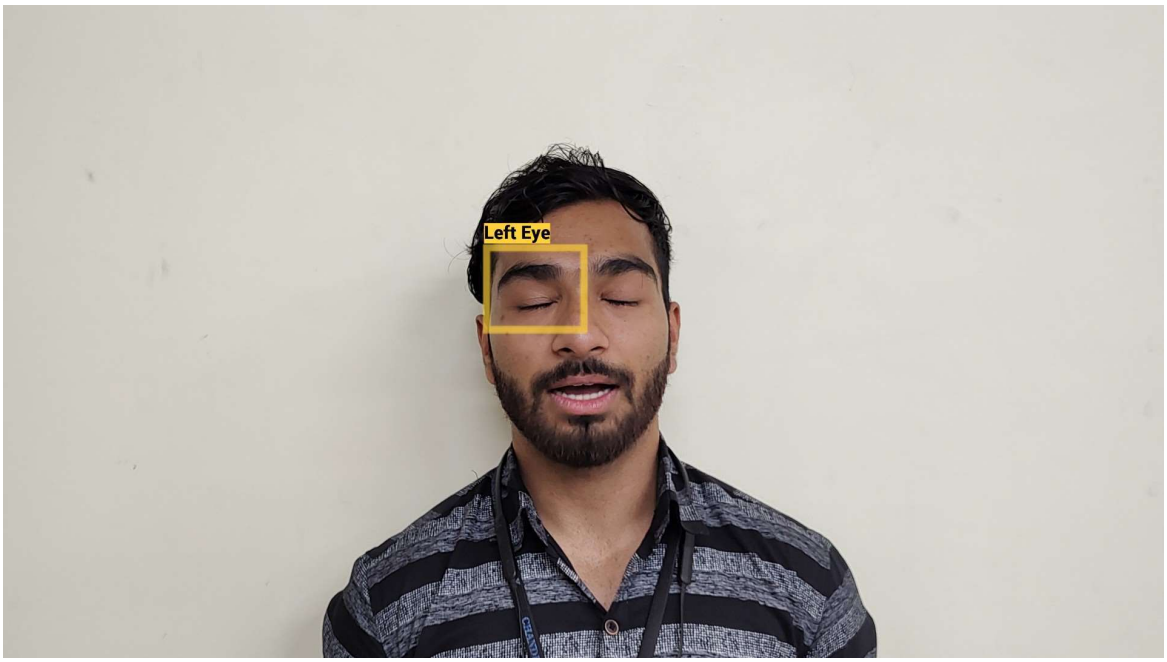
```
from IPython.display import Image
try:
  filename = take_photo()
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they do not
  # grant the page permission to access it.
  print(str(err))
```

```
<IPython.core.display.Javascript object>
```

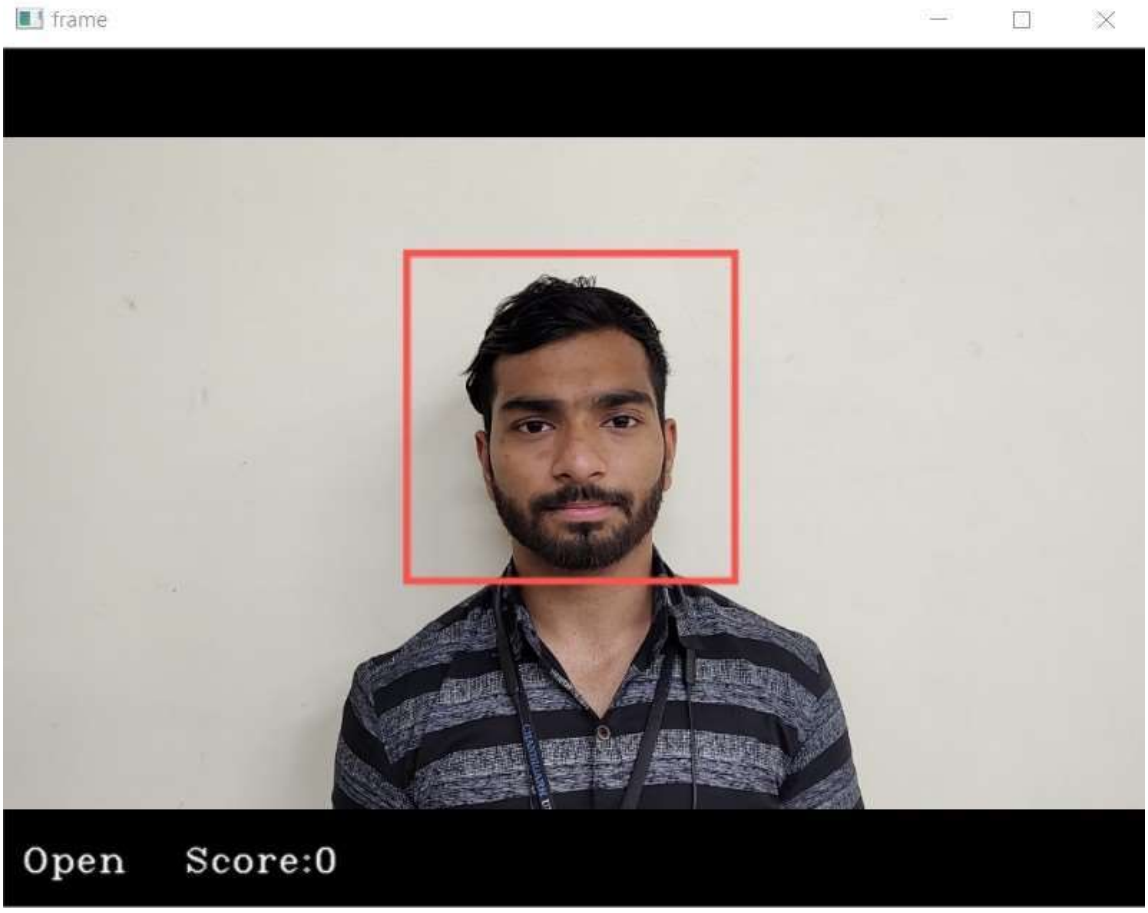
```
Saved to photo.jpg
```

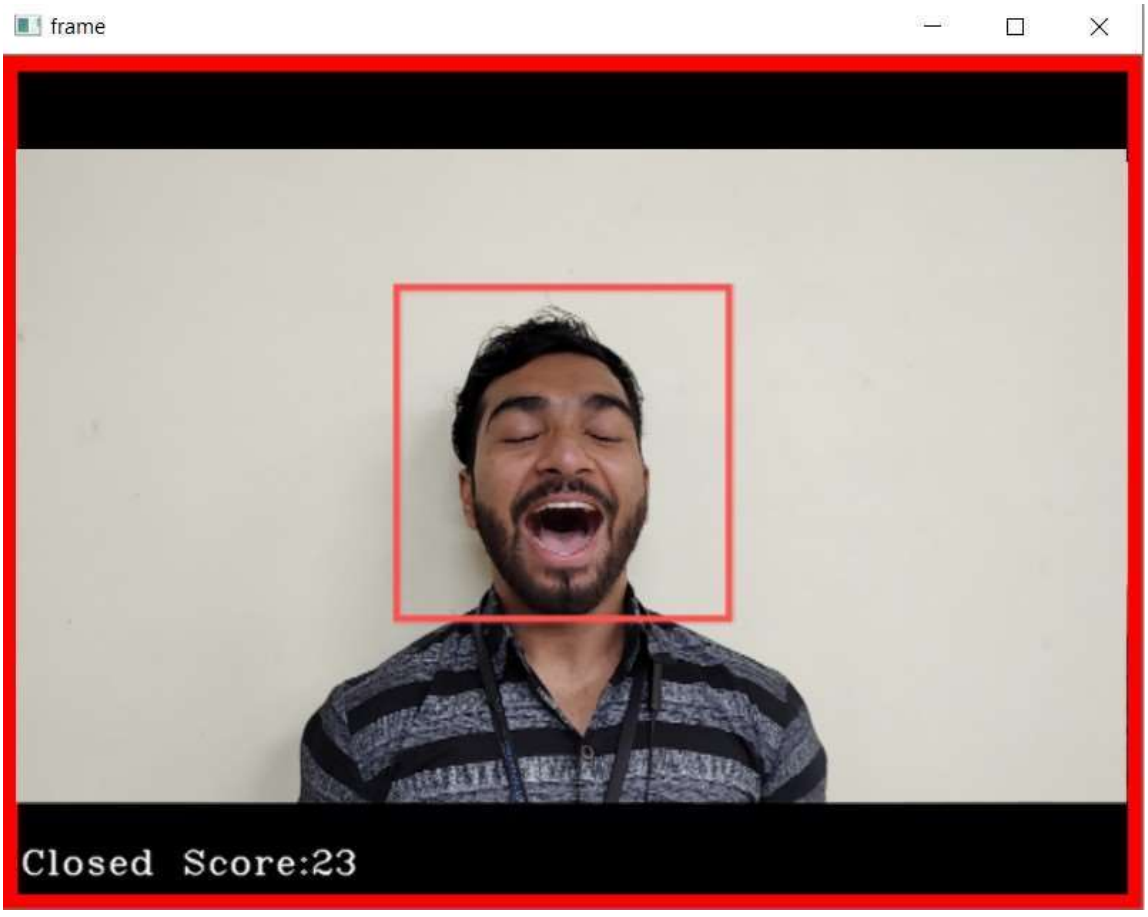


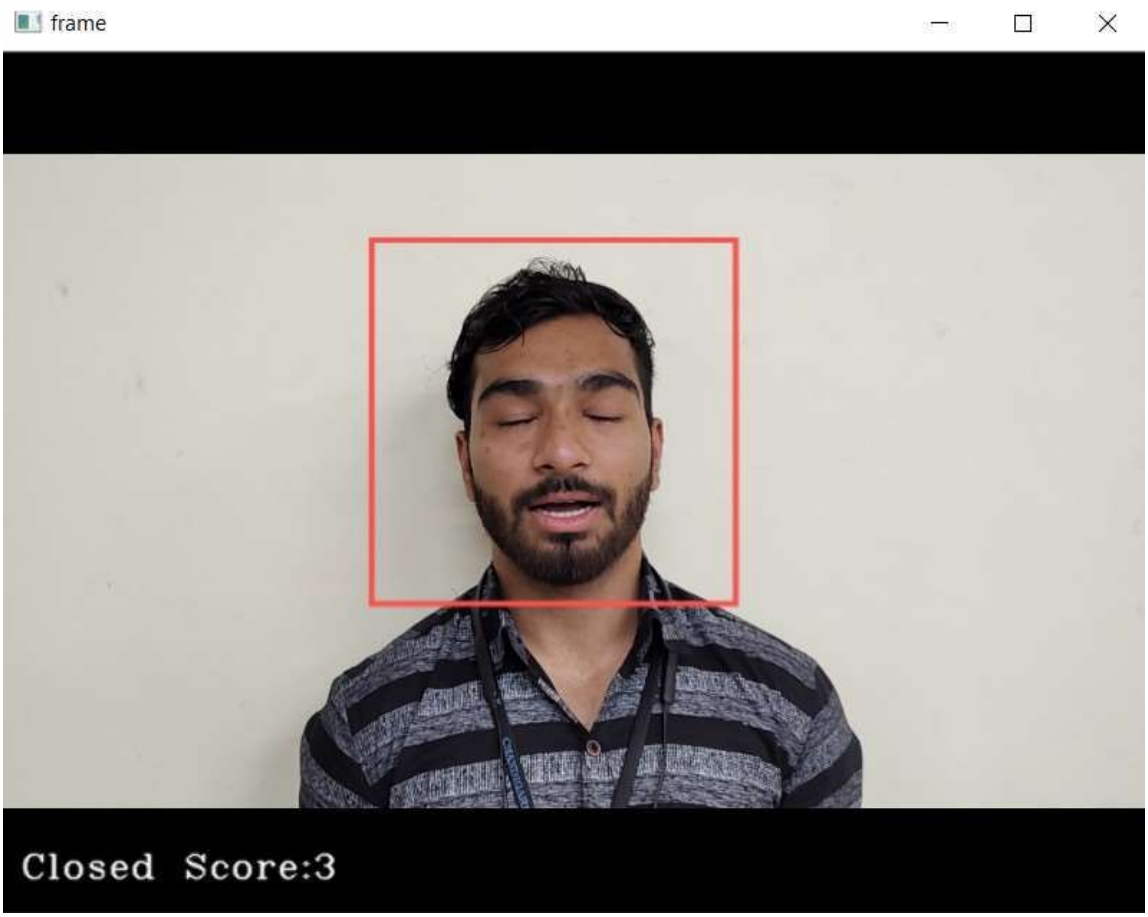



Left Eye


Right Eye


frame
Open Score:0


frame
Closed Score:23


frame
Closed Score:3