

*An autonomous car crash prevention system based on behavioral
assessment of driver*



Author

MAHAD ARIF

Registration Number

00000277767

Supervisor

Dr. Muhammad Jawad Khan

ROBOTICS AND INTELLIGENT MACHINE ENGINEERING
SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
ISLAMABAD
JUNE 2022

An autonomous car crash prevention system based on behavioral
assessment of driver

Author

MAHAD ARIF

Registration Number

00000277767

A thesis submitted in partial fulfillment of the requirements for the degree of
MS Robotics & Intelligent Machine Engineering

Thesis Supervisor

Dr. Muhammad Jawad Khan

Thesis Supervisor's Signature: _____



A handwritten signature in black ink, appearing to read 'Jawad Khan', is written over a horizontal line. The signature is stylized and cursive.

ROBOTICS AND INTELLIGENT MACHINE ENGINEERING
SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
ISLAMABAD
JUNE 2022

National University of Sciences & Technology**MASTER THESIS WORK**

We hereby recommend that the dissertation prepared under our supervision by: Mahad Arif, NUST Regn # 00000277767, Titled: "An autonomous car crash prevention system based on behavioral assessment of driver" be accepted in partial fulfillment of the requirements for the award of **MS Robotics and Intelligent Machine Engineering** degree in B+ Grade.

Examination Committee Members

1. Name: Dr. Hasan Sajid Signature: 
2. Name: Dr. Karam Dad Kallu Signature: 
3. Name: _____ Signature: _____

Supervisor's Name: Dr. Muhammad Jawad Khan

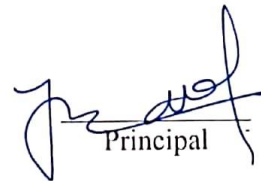
Signature: 


Head of Department

30/6/2022
Date

COUNTERSIGNED

Date: 30-6-22


Principal

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis written by Mr. Mahad Arif, (Registration No.00000277767), of SMME, NUST has been vetted by undersigned, found complete in all respects as per NUST Statutes /Regulations, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS Degree. It is further certified that necessary amendments as pointed out by GEC members of the scholar have also been incorporated in the said thesis.

Signature: _____

Name of Supervisor/ Dr. Muhammad Jawad Khan

Date: 29/6/22

Signature (HOD): _____

Date: 30/6/2022

Signature (Principal): _____

Date: 30-6-22

Declaration

The following research effort, named " *An autonomous car crash prevention system based on behavioral assessment of driver* " is my original work, which I certify. The work has not been submitted to another institution for evaluation. All other sources of information have been appropriately recognized and cited for the content that has been utilized.



Signature of Student

MAHAD ARIF

00000277767

Plagiarism Certificate (Turnitin Report)

This thesis has been checked for Plagiarism and the Turnitin plagiarism report endorsed by supervisor is attached.



Signature of Student

MAHAD ARIF

00000277767



Signature of Supervisor

Dr. Muhammad Jawad Khan

Copyright Statement

- The student author retains ownership of all intellectual property rights in the text of this thesis. It is only in line with the author's instructions that copies (by any method), either in whole or of excerpts, may be created and stored in the Library of the NUST School of Mechanical & Manufacturing Engineering (SMME). The Librarian can provide you with further information. This page must be included in all copies of the document that are created. It is not permitted to make more copies (by any means) without the express written consent of the author.
- Except where otherwise agreed, ownership of any intellectual property rights described in this thesis is vested in the NUST School of Mechanical & Manufacturing Engineering, subject to any prior agreement to the contrary. Intellectual property rights of this thesis may not be used by any third party without the written permission of the SMME, which reserves the right to prescribe the terms of any such agreement.
- In addition to this, the Library of the NUST School of Mechanical & Manufacturing Engineering in Islamabad has further information on the situations under which disclosures and exploitation are permitted.

Acknowledgements

I am obliged to the Almighty Allah for always blessing me with the best in everything. Without His guidance, I could not have completed my Master's degree and this Thesis. I also want to thank my thesis supervisor, Dr. Muhammad Jawad Khan, for his backing, encouragement, and accessibility during my research work for this thesis. Without his support and trust the whole process of research would have been impossible. His efforts and devotion throughout my research work were crucial in achieving this goal. Here I would also like to thank other faculty members of NUST especially Dr. Hasan Sajid and Dr. Karam Dad Kallu for their mentoring. Special thanks to everyone who helped me in my journey.

I am grateful to my parents for always supporting and praying for me throughout my life. They have always inspired me by setting a good example of hard work, honesty, devotion, and commitment in all aspects of their lives.

This page is intentionally left blank

Dedicated to my parents, wife, teachers, friends, and family

Abstract

Many road-side accidents occur due to the driver being not in the emotional state of driving. i.e., the driver is fatigued or is not alert. Computer Vision is one of the widely used fields in the world right now. The amount of work being carried out in this field is enormous and very helpful as well. One of such works is detecting human mood at any given time by analyzing the facial expressions of that person. The mood can be of these types. i.e., Alert, Fatigued, Happy, Sad, Drowsy etc. The "OpenCV" open-source Computer Vision library makes it possible to analyze facial expressions.

In this thesis, different behavioral assessments are made on a car driver's video recordings to detect drowsiness. These behavioral assessments include Eye Blinks detection, Yawning Detection, Percentage Eye Closure (PERCLOS) and Pose Estimation. All these are ensembled together to give a more accurate prediction of a driver being drowsy. It was concluded that the number of false positives increase during night-time and thus the accuracy of the system goes down when the lighting conditions are low. Also, camera for driver's video recording should be placed just behind the left of steering wheel for maximum number of true detections. The system also works in real-time thus making it more useful.

Keywords: *Drowsiness Detection, OpenCV, Dlib, MediaPipe, Eye Blinks, Yawning, PERCLOS, Pose, Fatigue, Drowsy Driving, Car Crash Prevention System, Behavioral Assessment, Real-time*

Table of Contents

COPYRIGHT STATEMENT.....	I
ACKNOWLEDGEMENTS.....	II
ABSTRACT.....	V
TABLE OF CONTENTS.....	VI
LIST OF FIGURES	IX
LIST OF EQUATIONS	X
LIST OF ACRONYMS	XI
1. INTRODUCTION.....	1
1.1 PREVIOUS WORK.....	2
1.2 PROBLEM STATEMENT	4
1.2 OBJECTIVE	4
1.3 AREAS OF APPLICATION.....	5
1.4 THESIS OVERVIEW	5
2. THEORY	6
2.1 TYPES OF ASSESSMENTS	6
2.1.1 <i>Vehicular Based Techniques</i>	6
2.1.2 <i>Physiological Parameters-Based Techniques</i>	6
2.1.3 <i>Behavioral Based Techniques</i>	7
2.2 MAJOR LIBRARIES TO USE.....	8
2.2.1 <i>OpenCV</i>	8
2.2.2 <i>Dlib</i>	10
2.2.3 <i>MediaPipe</i>	12
2.3 OTHER USEFUL LIBRARIES.....	13

2.3.1	<i>NumPy</i>	13
2.3.2	<i>IMUTILS</i>	13
2.3.3	<i>SciPy</i>	13
2.3.4	<i>Math</i>	14
3.	METHODOLOGY AND IMPLEMENTATION	15
3.1	PARTICIPANTS / SUBJECTS.....	15
3.2	EXPERIMENTAL SETUP.....	15
3.3	SOFTWARE AND HARDWARE SPECIFICATIONS.....	16
3.4	WORKFLOW.....	17
3.5	FRAME PRE-PROCESSING.....	19
3.6	DROWSINESS DETECTION.....	20
3.6.1	<i>Face Detection</i>	20
3.6.2	<i>Mouth Detection</i>	21
3.6.3	<i>Yawning Detection</i>	22
3.6.4	<i>Eyes Detection</i>	23
3.6.5	<i>Blinks Detection</i>	23
3.6.6	<i>PERCLOS Estimation</i>	24
3.6.7	<i>Pose Estimation</i>	25
3.6.8	<i>Drowsiness Alert</i>	28
4.	RESULTS AND DISCUSSION	29
4.1	YAWNING.....	29
4.2	BLINKING.....	31
4.3	PERCLOS.....	33
4.4	POSE.....	34
4.5	DROWSINESS ALERT.....	36
4.6	REAL-TIME DETECTION.....	38
4.7	LIMITATIONS.....	39

5. CONCLUSION.....	41
6. FUTURE WORK.....	42
7. REFERENCES	43

List of Figures

FIGURE 1: A 14-CHANNEL EMOTIV EEG HEADSET	7
FIGURE 2: AN EXAMPLE OF IMAGE PROCESSING WITH OPENCV	10
FIGURE 3: DLIB'S 68 FACIAL LANDMARKS.....	11
FIGURE 4: POSE LANDMARKS OF MEDIAPIPE.....	12
FIGURE 5: RECORDED FRAME OF SUBJECT WHILE DRIVING	16
FIGURE 6: PROPOSED ARCHITECTURE.....	18
FIGURE 7: FRAME AFTER GRAYSCALE CONVERSION	20
FIGURE 8: EYE LANDMARKS.....	23
FIGURE 9: NOSE SHOULDER ANGLE MEASUREMENT	26
FIGURE 10: SUBJECT YAWNING	29
FIGURE 11: YAWN DETECTION ACCURACY BY DAY TIME	30
FIGURE 12: YAWN DETECTION ACCURACY BY CAMERA ANGLE	31
FIGURE 13: SUBJECT BLINKING.....	32
FIGURE 14: BLINKS DETECTION ACCURACY BY DAY TIME	32
FIGURE 15:BLINKS DETECTION ACCURACY BY CAMERA ANGLE	33
FIGURE 16: PERCLOS AT THE START OF VIDEO, AFTER 20 SECONDS, 30 SECONDS, 40 SECONDS	34
FIGURE 17: SUBJECT'S POSE WHILE DRIVING.....	34
FIGURE 18: POSE DETECTION ACCURACY BY DAY TIME.....	35
FIGURE 19: POSE DETECTION ACCURACY BY CAMERA ANGLE	36
FIGURE 20: DROWSINESS ALERT BASED ON INCORRECT POSE	37
FIGURE 21: DROWSINESS DETECTION ACCURACY BY DAY TIME.....	37
FIGURE 22: DROWSINESS DETECTION ACCURACY BY CAMERA ANGLE.....	38
FIGURE 23: REAL-TIME PROCESSING.....	39
FIGURE 24: NIGHT-TIME DETECTION	40
FIGURE 25: SIDE POSE DETECTION	40

List of Equations

EQUATION 1: FORMULA FOR CALCULATING EAR 24

EQUATION 2: ANGLE BETWEEN 2 VECTORS..... 27

List of Acronyms

Computer Vision	:	CV
Machine Learning	:	ML
Deep Learning	:	DL
Artificial Intelligence	:	AI
Percentage Eye Closure	:	PERCLOS
Eye Aspect Ratio	:	EAR
Mouth Aspect Ratio	:	MAR
Open-Source Computer Vision	:	OpenCV
Electroencephalography	:	EEG
Heart-Rate Variability	:	HRV
Integrated Development Environment	:	IDE
Scientific Python Development Environment	:	SPYDER

1. Introduction

One of the main reasons of many roadside accidents is the drowsiness. It is a natural thing to feel drowsy, but it becomes a nightmare while you are driving a car or any other vehicle for that matter. Some statistics from around the world are cited below which can help us understand how serious of an issue this actually is. European Sleep Research Society reported that about 25-30% of accidents in UK are due to driver's drowsiness. 35% drivers in Netherlands, 70% in Spain reported to have fallen asleep while they were driving. While in France, out of total 3,970 fatal accidents on the road, 85% were due to drivers being drowsy in 2011. Drowsy drivers in Germany caused 25% of all fatal road traffic accidents. In north countries such as Finland for example, 17% of fatal vehicle accidents were related to fatigued drivers; and they caused 18% of deaths on the road between 2006 to 2010. In United States, the National Highway Traffic Safety Administration (NHTSA) has reported that each year about 1550 deaths, 71,000 injuries, and 12.5 billion dollars of losses are due to crashes caused by drowsy driving [1]. These are some alarming statistics for the authorities to act upon.

The concentration level of driver is key to avoid any accident at all but while being drowsy that cannot be possible. Driving is meant to be a risk free and effective way of traveling but while there is a chance of a person driving while being drowsy, it cannot be risk free or without danger, not only for themselves but also for other people driving with them on the road. So, while this is in the back of someone's mind, how can they drive properly without worrying about not getting into an accident. To come up with a solution for this problem, many research have been made and to very effective ends as well. We will look at them in the below chapter.

1.1 Previous Work

[2] Picot et al., calculated degree to which elevated-speed camera to be placed. EOG is used for mining the blink attributes to detect drowsiness. They present a method for detecting and characterizing the blinks. Found high correlation between length, occurrence, PERCLOS and vibrant characteristics derived from EOG and video indicators. Rate of frame effect on various mined attributes is also reviewed. [3] Flores et al., used Gabor filter. They did video-based somnolence detector joined with EEG centered somnolence detection. [4] Singh et al., developed driver fatigue monitoring system that detect the drowsiness based on eye tracking. Physiological and behavioral measures such as heart proportion and unevenness, rate of respiration, movement of head (blink length, frequency and PERCLOS) and reported driving activity such as time-to-lane crossing, speediness, angle of the steering, lane direction were assessed. [5] Shen et al., used red eye effect and texture detection method for drowsiness detection. [6] Mbouna and Kong, calculated the elucidated Position of Eye and Pose of Head Visual Analysis for driver vigilance examining. The visible features like pupil activity (PA), HP and eye index were proposed in order to retrieve information on driver vigilance. The sequence of video frames are used to classify vigilant and non-vigilant driving events by SVM. [7] Tadesse et al., used Viola Jones algorithm to spot face. Color histograms in conjunction with LBP are used in order to detect the face beyond frames. [8] Anizy et al., projected a fully automatic system efficient for identifying drowsiness and distraction of driver by using SVM and Haar classifier for face identification and eye recognition. [9] Khunpisuth et al., measured degree of drowsiness by using Raspberry Pi Camera and a Raspberry Pi 3 module that was able to measure degree of somnolence in drivers. The head tipping and eye twitch was used to assess whether a driver felt somnolent or not. [10] Mohsen et al., detect drowsiness at early stage by measuring the heart rate variability using logistic regression based machine learning algorithm. [11] P. Huynh and Kim used Deep Neural Network and semi-supervised gradient boosting algorithm in order to identify face. Then traces the face features by

merging the Kernelized Correlation Filter with a Kalman filter in order to strongly trace the face. Retrieved face regions are then moved to 3-d Convolutional Neural Network that is trailed over a gradient boosting machine for categorization. [12] Chellapa et al., used Eye Aspect Ratio (EAR) and Haar Cascade Classifier to detect drowsiness but the system doesn't work under low lighting conditions. [13] Galarza et al., used Percentage Eye Closure (PERCLOS) with HCI classifier for successful drowsiness detection. [14] Wisaroot et al., measured driver's eye closing and yawning by infrared camera. They used metrics like face identification, identification of eye, identification of mouth, eye closing, and detection of yawning in order to correctly detect drowsy state of the driver. [15] Lahoti et al., measured Eye Aspect Ratio (EAR) and Mouth Aspect Ratio (MAR). They used Naive Bayes Classifier, Random Forest and SVM classifier for this purpose. [16] Gwak et al., measured the level of drowsiness, driving efficiency, physiological signals and behavioral indicators of driver using a driving simulant. [17] Their research is based on real-time detection of the driver's head, face, and mouth (Haar Classifier). There are 3 stages in this research. 1st: Image Capturing (extract driver's facial expressions and detecting head movement, Collecting samples of eyes closed/open). 2nd: Analysis (Image is then preprocessed using various 'Image Processing' techniques for drowsiness detection like Blurring, RGB to HSV Conversion, HSV Thresholding Blob Detection). 3rd: Alert (If found in abnormal state, alarm is activated to alert the driver). SVM is used. [18] used 2 methods. CBS (Cumulative Blink Signal Analysis) and IFD (Intinsic Functions Decomposition). CBS locates and analyzes the eyes blinking from nonstationary and nonlinear signal to detect the drowsiness state. IFD technic is based on the nonstationary and nonlinear signal to analyze the nonstationary and nonlinear signal by using the combination between the two methods: Empirical Mode Decomposition (EMD) and Band Power (BP). Analysis is confirmed by SVM. [19] Utilized 10 ocular parameters, namely blink duration, blink rate, percentage of eye closure (PERCLOS), microslepp, SEM, saccade amplitude, saccade duration, saccadic PV, pupil diameter, and fixation duration. They concluded Blink duration, PERCLOS, and saccadic PV

demonstrated high accuracy, sensitivity, and specificity to detect fatigue. [20] made use of Physiological and Facial Analysis. Smartphone camera was used for heart rate by tracking color on fingertip. Google Vision API (head position, blinking duration, yawning frequency) through eye opening and mouth opening probabilities. Heart rate, blinking, yawning, and speeding were used as indicators of drowsiness. [21] recorded video of driver and detected them with image processing techniques in each frame. The facial characteristics of the detected face are pointed and the aspect ratio, the mouth opening ratio and the nose longitation relationship are calculated and drowsiness is detected in accordance with their values. Some other papers [22]–[28] talked about driver’s drowsiness detection based on behavioral analysis which uses driver’s facial features, yawning, eye closure, PERCLOS for drowsiness detection. These techniques are most relative to our thesis as it also covers behavior-based drowsiness detection.

1.2 Problem Statement

Different parameters are analyzed for drowsiness detection whether behavioral or physiological or vehicular based. Some may not give accurate results, and some may give very accurate results. To compensate for this, we need to analyze more parameters together. Can the accuracy increase if we analyze 4 behavioral parameters together? Also, what is the best angle for capturing driver’s video to detect maximum number of blinks, yawns?

1.2 Objective

Objectives of the thesis include: -

- Developing a system that can detect real-time drowsiness.
- Analyzing 4 parameters (Blinks, Yawns, PERCLOS, Head Pose) together.
- Real-time drowsiness alert to be given if drowsiness occurs and detected.

1.3 Areas of Application

This work can be used for car drivers, train drivers, pilots etc. Similarly, it can be used on person doing very accurate and tough tasks such as a surgeon doing a very demanding surgery and can be deemed to rest if they are feeling drowsy by the intensive work.

1.4 Thesis Overview

The thesis has been divided into 6 Sections. Section 1 contains the introduction and the preliminaries. Section 2 briefly explains the theory behind the chosen method. It contains all the details of the theory. Section 3 contains the complete methodology and work done for this thesis.

Section 4 includes the complete results acquired after implementing the proposed method. Section 5 consists of discussion of the complete work and also concludes the entire work. Section 6 describes all the possible future work which can be held in this domain.

2. Theory

2.1 Types of Assessments

There are three main cutting-edge approaches used to detect drowsiness. These methods include those based on vehicles, those based on physiological parameters, and those based on behavioral parameters. Below, we'll take a closer look at all three.

2.1.1 Vehicular Based Techniques

The type of techniques in which we analyze the vehicle instead of the one driving it are vehicular based techniques. This includes the lane changing, steering wheel angle, vehicle speed variability, accelerator movement, grip force and so on. These methods are used to detect downfall of the driving performance due to fatigue and drowsiness. Steering behavior of the driver is measured by an angle sensor which is mounted on the steering wheel. When the driver is drowsy, the number of micro-movements reduces as a result. Lane changing is measured by detecting the lanes by Hough Transform first and then detect the driver's face and eyes by Viola-Jones Algorithm [29]. The advantage of this technique is that it is a non-intrusive method. The disadvantage in this type of technique is that it is affected by the external factors such as weather conditions and road conditions.

2.1.2 Physiological Parameters-Based Techniques

In these type of techniques, a driver's physical and environmental conditions are considered. These conditions being heart-rate variability, body temperature, pulse rate, brain activity etc. One of the most used methods for this type of technique is Electroencephalogram (EEG) based drowsiness detection. In this method an EEG headset is mounted on the head of the driver while they are driving thus gathering the electrical activity of the brain. The brain signals are then analyzed to identify drowsy state. The other method commonly used for this type of technique is pulse sensor

method which measures the heart pulse rate using infrared heart-pulse sensor placed on the wrist of the driver. By measuring the amount of blood flowing through the hand and the oxygen level, infrared light is reflected back to the transmitter, where Arduino is utilized as a microcontroller to receive the oxygen fluctuation given by the sensor. Processing the frequency of Heart Rate Variability (HRV) allows us to see this. As a result, it was discovered that a person's low to high frequency ratio decreases when they become sleepy. [29].

The advantage of this technique is that it is very effective and reliable. The disadvantage is that it is intrusive and hinders the driving which is very bad considering we use it in real life environment.



Figure 1: A 14-channel EMOTIV EEG headset

2.1.3 Behavioral Based Techniques

These techniques considers the drivers behavioral parameters such as facial expression changes, eye blinks, eye aspect ratio, yawning etc. These techniques help in measuring the geometric changes in the shape of a person's face. Facial expressions are caught and then measured against the template of the already saved facial expressions of a person. In yawning detection, geometric

shape of mouth is measured, a person is considered yawning if the value increases a certain threshold. Eye blinking detection is another popular method of this technique which measures the eye blinks of the person and relates it to the normal frequency of a person blinking per minute [29]. Eye blinks are measured by getting the ratio of height of eyes to the length of eyes, which if less than a certain threshold, an eye blink will be considered. Another method to detect drowsiness by this technique is PERCLOS estimation which is basically the ratio of eye closed to total frames in that window.

The advantage of this technique is that it is non-intrusive and easy to use while its disadvantage is that it is affected by illumination and low lighting conditions.

Technique	Parameters Used	Advantages	Disadvantages
Vehicular Parameters-Based	Lane Changing, Steering Wheel Angle	Non-Intrusive	Affected by external conditions, Unreliable
Physiological Parameters-Based	Heart Rate Variability, Pulse Rate, Brain Activity	Reliable, Effective	Intrusive
Behavioral Parameters-Based	Facial Expressions, Eye Blinks, Yawning	Non-Intrusive, Easy to use	Affected by illumination, lighting environments

Table 1: Comparison of different types of drowsiness detection techniques

By comparing these all techniques, their pros, and cons, it was decided to use the Behavioral Assessments to detect the drowsiness of the driver for this thesis.

2.2 Major Libraries To Use

2.2.1 OpenCV

An massive open-source library called OpenCV (Open-Source Computer Vision) is used for many different things, including Computer Vision (CV), Image Processing, Machine Learning (ML), Deep Learning (DL), and Artificial Intelligence (AI). It is a highly optimized library that plays an

important role in real-time operations which is a need of modern day's systems. The fact that it is a cross-platform library whose C++, JAVA and Python interfaces support Linux, Windows, MacOS, iOS and Android is a game changer and puts it at the very top and makes it one of the most used libraires across the world. It covers both conventional and cutting-edge CV and ML methods, totaling more than 2500 optimized algorithms. The work of these algorithms include face recognition, object identification, action identification, vehicle license plate identification, text scanning, camera movement tracking, object movement tracking, gender detection, age detection, extracting 3D models of objects, stitching images together to make a big picture of the whole scene, finding similar images, smart attendance system via face recognition, removing red-eye effect, following eye movements and many more. To see its popularity and usage this is the list of well-established companies which are using this library: Google, Yahoo, Intel, IBM, Microsoft, Honda, Toyota. It has an extensive amount of documentation and online support which is a super thing to have when you are working with a library. In this thesis, this library is used to get a video input that will be used for drowsiness detection. Then through this, video is processed frame by frame converting each one into grayscale for further analysis. Once done with the analysis, the video is shown back on the screen with text put on it through OpenCV.

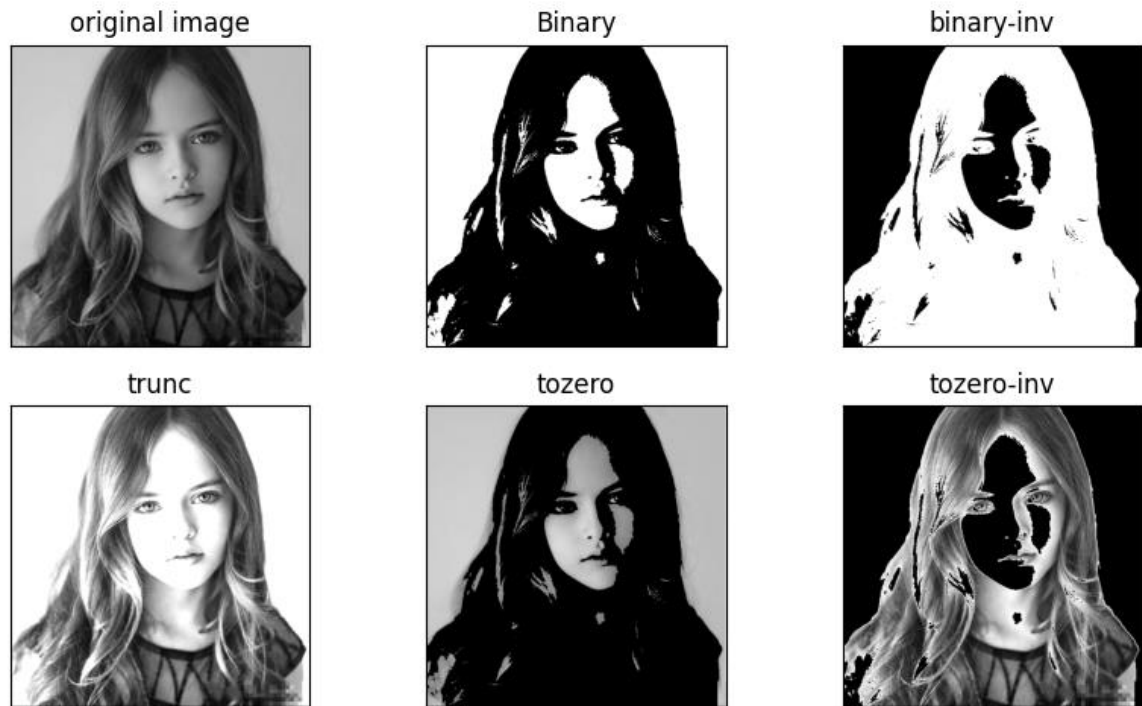


Figure 2: An Example of Image Processing with OpenCV

2.2.2 Dlib

Machine Learning algorithms are included in the general-purpose, cross-platform C++ library known as Dlib. In fields including robotics, embedded technology, mobile phones, and extremely large-scale computer systems, it is widely employed in both industry and academics. It is open source, therefore anyone can use it for free in any application. One of the main advantages of this library is that it has a large community and online support available for it. Also, it provides full documentation of each and every function and class in it. This is such a big help for the developers as they can implement these functions and classes almost immediately just after reading its comprehensive documentation thoroughly. The fact that this library features debugging modes that verify the functions' documented preconditions, when activated, will catch the great majority of faults that were brought on by improper use of the function. This library is mostly used for Machine Learning (ML) techniques, such as SVM for classification and regression, general-purpose multiclass classification, multiclass SVM, kernel RLS regression, clustering algorithms, and many

others. For this thesis, Dlib's facial landmark detector is used to detect a human face. It has a pretrained model which identified 68 landmarks on a human face that tells us its features. Basically these 68 landmarks are the coordinate positions (x, y) that map the facial points on a person's face like eyes, nose, mouth etc. This makes it easy to use and correctly detect the face and its features without much hassle.

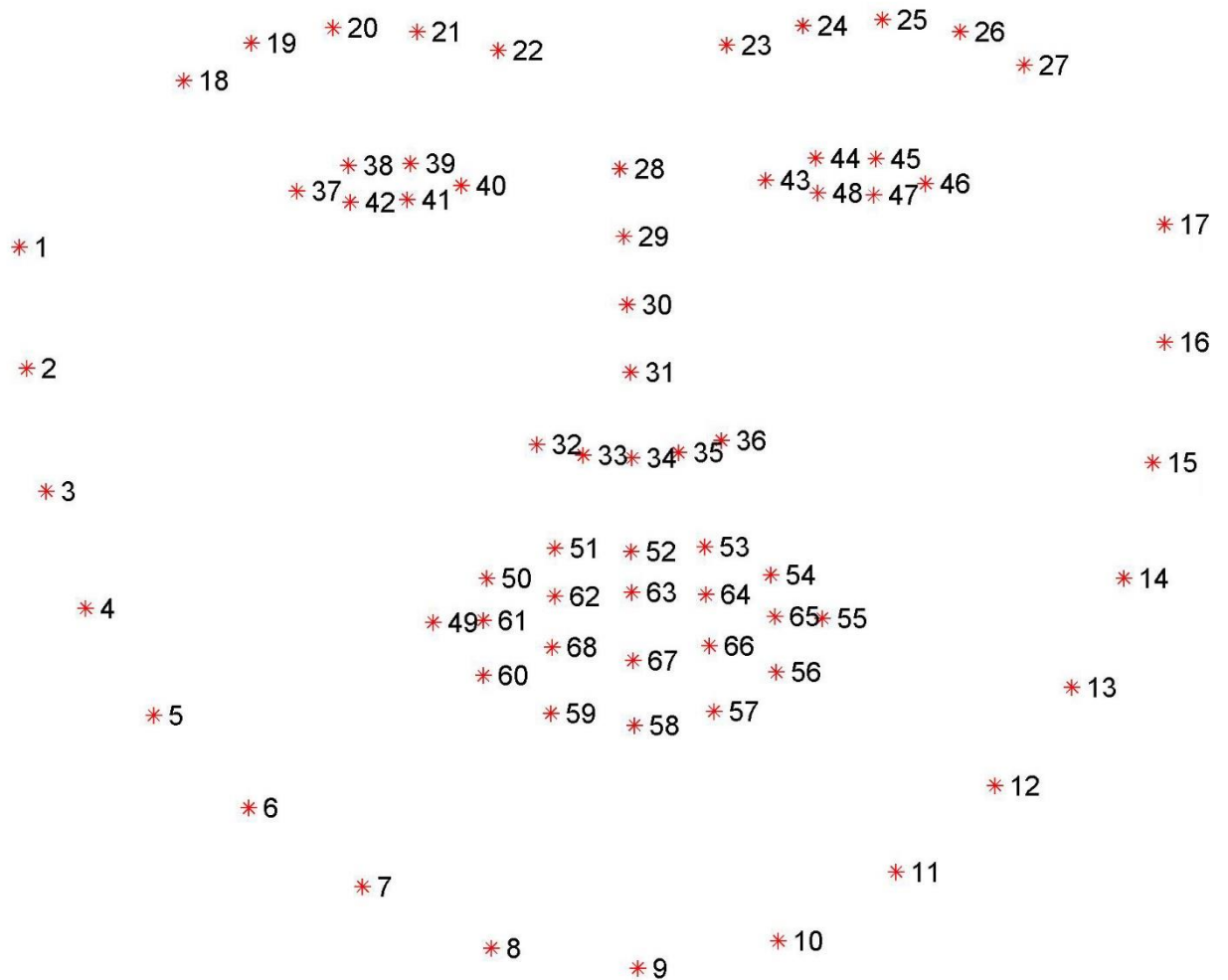


Figure 3: Dlib's 68 facial landmarks [30]

As shown in Figure 2, there are 68 marks for the face detection. Point numbers 1-17 describes the outline of the face while point numbers 18-22 and point numbers 23-27 tells us the coordinates of right and left eyebrows respectively. Points 28-31 marks the nose front (vertically) while points 32-36 marks bottom of the nose (horizontally). Point numbers 37-42 and point numbers 43-48 maps the right and left eyes respectively. Finally, point numbers 49-68 are for the mouth. These

all 68 points combine to give a facial structure that can be used to detect a human face and its components.

2.2.3 MediaPipe

MediaPipe is another cross-platform Machine Learning (ML) solutions library for live and streaming media. Some of its ML solutions are human pose detection and tracking, hand tracking, object detection and tracking, face detection, face mesh, hair segmentation and many more. This is also a free and open-source library which is fully customizable and can be deployed anywhere as its solutions work across android, iOS, desktop, cloud, web and IoT. In this thesis, MediaPipe library is used for Pose Estimation which will tell us about a drivers pose while driving and will alert us if it isn't correct. Refer figure 3 for full body pose landmarks by MediaPipe.

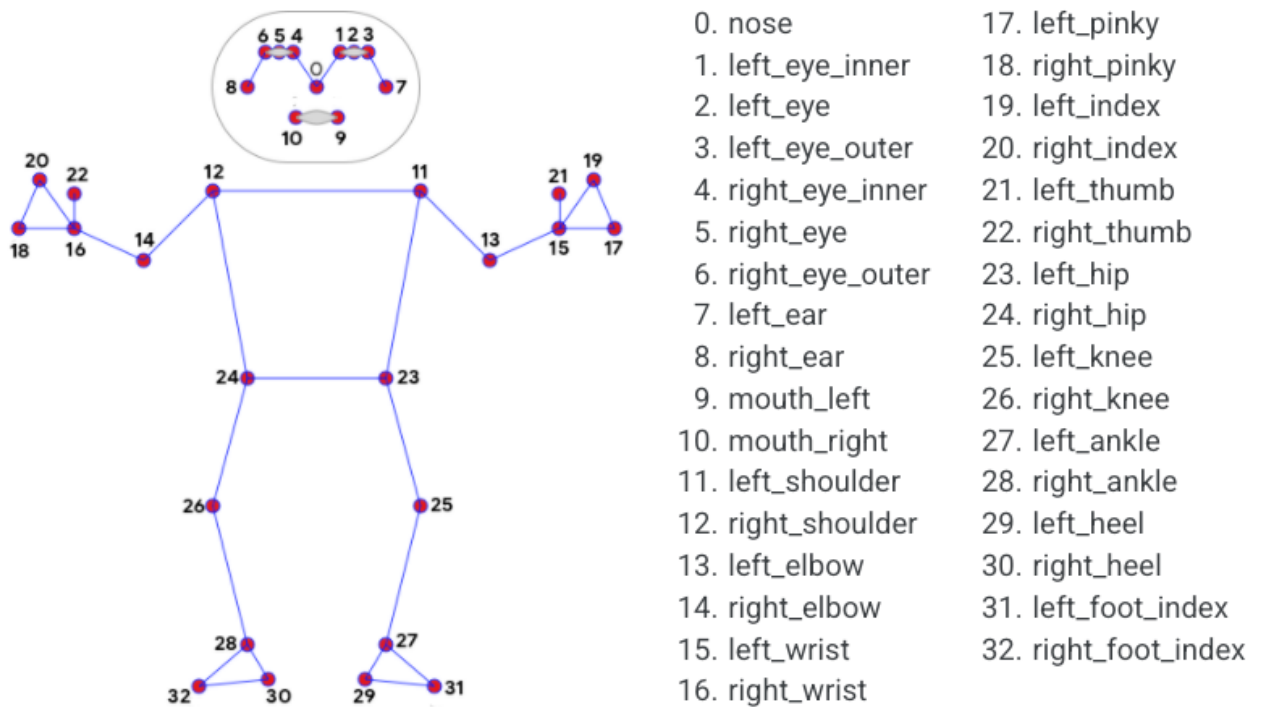


Figure 4: Pose Landmarks of MediaPipe [31]

For our purpose, we would need the points from 1 up to 22 as these are the points which will be visible of a driver driving a car when a camera is placed on the front dashboard.

2.3 Other useful libraries

2.3.1 NumPy

NumPy stands for Numerical Python, an open-source project that can be used freely. It is a Python library offering a selection of functions for dealing with multidimensional arrays. In python, we have lists which are slow to process and takes a lot of time to go through all the elements. To make this problem go away, NumPy arrays were introduced which are 50% faster than the python lists and saves a lot more time where time is of essence. Also, it provides a lot of supporting functions that make working with NumPy arrays very easy. It provides comprehensive mathematical functions, linear algebra routines, Fourier transforms, matrices and much more. In short, it is used for numerical computing, image processing, signal processing, statistical computing, graphs and networks, architecture, engineering and many more.

2.3.2 IMUTILS

With the aid of OpenCV and Python, IMUTILS is a set of practical methods that greatly simplify fundamental image processing tasks including translation, rotation, resizing, displaying images, and far more.

2.3.3 SciPy

It is an open-source scientific python library which is used to perform scientific, mathematical, and engineering computations. It depends on NumPy which is explained above in section 2.4.1. This library offers numerous effective numerical operations like methods for numerical method and optimization and is designed to be used with NumPy arrays. It is organized in sub-packages which covers different scientific domains. For example, `scipy.linalg` for Linear Algebra routines, `scipy.interpolate` for Interpolation, `scipy.io` for Input and Output and many more like this. We used `scipy.spatial` which is used for Spatial Data Structures and Algorithms to calculate the Euclidean distance for eyes which will be explained later in the thesis.

2.3.4 Math

Math is a python library which is used for mathematical calculations as its name suggests. Some of the calculations we can do with this library are ceil, floor, factorial, absolute value, lowest common multiple, square root, exponent, log, power, hypot and many more.

3. Methodology and Implementation

3.1 Participants / Subjects

For the experiment, we needed subjects to record a video of while they were driving a car. Five males participated in the experiment. The subjects were of different ages and were all licensed drivers. None of the participants had any vision impairment or any other health problem that could potentially harm their driving. Furthermore, all of them were inquired about any neurological or psychological problems they might be facing. The subjects were briefly explained about the experiment and its setup after which they gave their consent to participate in the experiment.

3.2 Experimental Setup

To analyze changes in the facial expressions of a person, we need to record their face clearly. So, it was decided that the optimal place to put the camera for recording of a person driving a car was on the dashboard just besides the steering wheel. The clear view of the face was absolutely necessary for our behavioral analysis to detect drowsiness in the driver. The recordings were carried out during different day times and in different lighting conditions so that our system can be checked in almost every situation and whether it is giving right results or not.

The camera used for recording was of a smart phone which can be mounted on the dashboard by a stand so that it doesn't move much and give us as stable of a recording as we can get. The resolution for each video recording was different. For each participant, the recording time spanned over 30 minutes. The amount of time was decided so that we don't miss the facial expressions of subject during drowsy state if there was, in fact, drowsiness occurring. So, 30 minutes seemed like a pretty good amount of time to analyze the changes in the facial expressions of a person driving a car.



Figure 5: Recorded frame of subject while driving

As can be seen in figure 5, the recorded video contains a clear image of subject's face and thus, it can be used to analyze his facial expressions.

3.3 Software And Hardware Specifications

For analyzing the recorded videos, we used programming language “PYTHON 3”. To make use of this programming language, a distribution for Python and R programming languages for scientific computing “ANACONDA3” (developed by Anaconda, Inc.), was installed on the machine. This distribution includes much software for both python and R that can be installed on the machine and can be worked upon through Anaconda's Navigator. For our use of python 3, an IDE (Integrated Development Environment) namely “SPYDER (Scientific PYthon Development EnviRonment)” v5.1.5 was installed on Anaconda Navigator and was subsequently used for coding in python.

The coding, processing and analyzation of recorded videos were done using HP Envy TS m6 Sleekbook. It has an Intel® Core™ i5-4200U with 4 CPUs @ 1.60GHz each, 12 GB RAM and an “Intel® HD Graphics Family” Graphics Card with 2 GB memory. The system was running

Microsoft® Windows 10 operating system and handled all the processes completely and efficiently.

3.4 Workflow

The proposed workflow is as follows.

1. Video Recording
2. Frame Preprocessing
3. Face Detection
 - a. Mouth Detection
 - i. Yawning Detection
 - b. Eyes Detection
 - i. Blinks Detection
 - ii. PERCLOS Estimation
 - c. Pose Estimation
4. Drowsiness Alert

All these steps will be explained one by one in detail to give an in-depth view of this work. Figure 6 shows the entire architecture. It involves basic objectives explained earlier in this document.

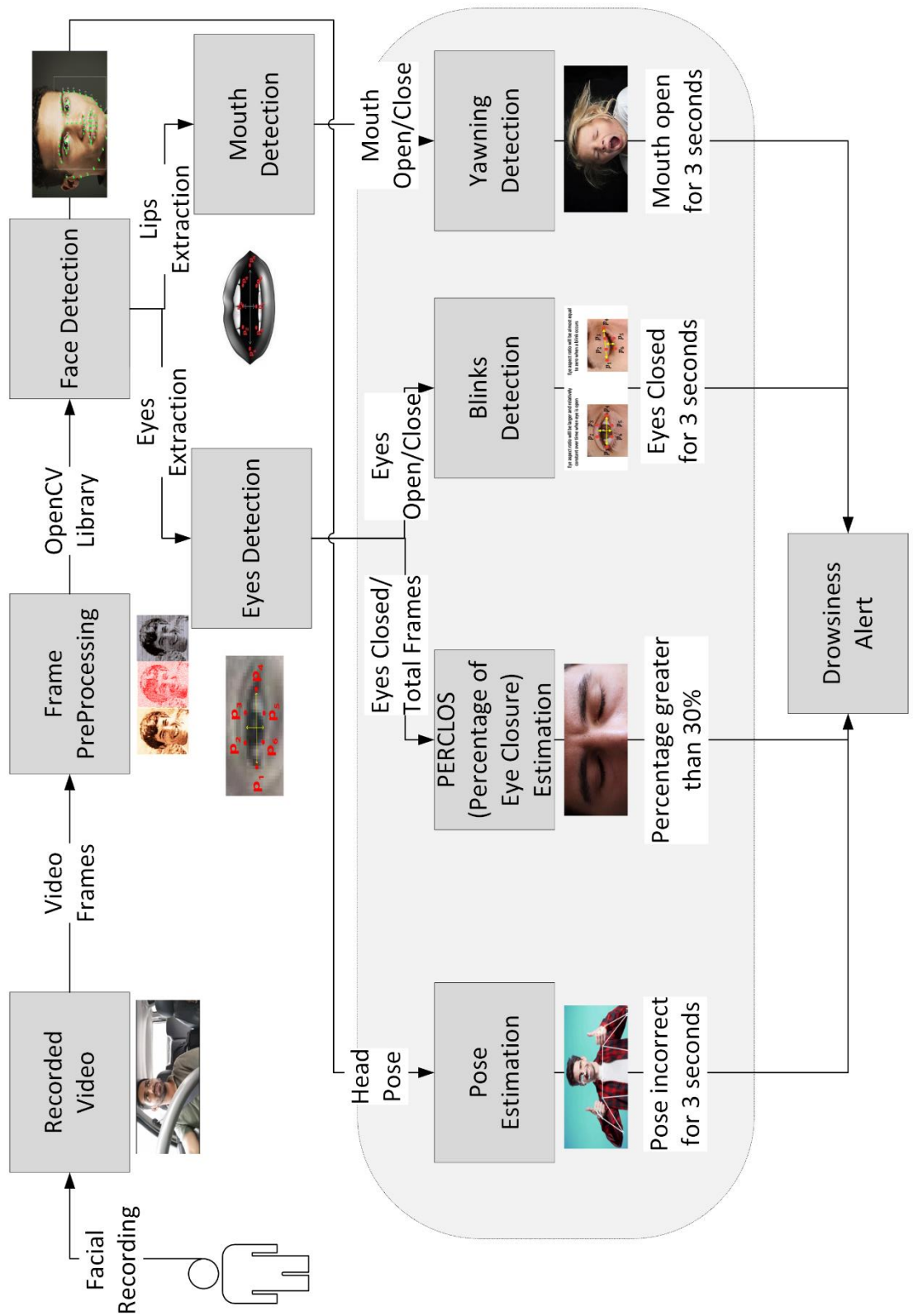


Figure 6: Proposed Architecture

3.5 Frame Pre-Processing

First and foremost, we needed to capture the video we needed to analyze. For this purpose, we used OpenCV's function "cv2.VideoCapture" and gave the path to the video which we needed to analyze. If we use 0 as the parameter for the video, it starts the webcam and gets the live camera feed as video input. Whatever video source we use, video input is saved in a variable (for e.g., 'cap') which we will use for the pre-processing afterwards. The OpenCV library takes the video as frames and apply processes on those frames one by one instead of applying them on whole video at once. So, everything we do after this will be applied on one frame at a time till all frames are ran through. But before we get to that, we need to get the total FPS of the recorded video to use afterwards inside our main loop. We do that via get() function on the cap variable which was capturing the recorded video and save it in a variable FPS. The get() function takes in an argument cv2.CAP_PROP_FPS which is responsible for getting the total FPS of a recorded video.

A while loop is started which will run for each and every frame until the frames are read or we press an exit key. The exit key was set to be 'q' and is defined at the end of while loop. In the while loop, first we check whether the video frames are ended i.e., if the video is completed to make sure that the processing never stops on its own except only if we press the exit key. If that's the case, we reset the frames to the very first frame i.e., at the start of the video. Otherwise, if the frames are continued, we do our processing on every frame one by one. First, we capture the frames from the variable (cap) we first created by OpenCV's function 'cap.read()'. This function returns 2 values, the first one being a Boolean value which will be 1: for frame read and 0: for no frame read (this is also the value which keeps on running the main while loop which we started before at the beginning). The second value 'cap.read()' returns is the frame itself. We gather both these values in 2 separate variables i.e., 'success' for Boolean value and 'img' for the current frame we need to process.

Now we need to resize the frame width to a fixed value of 640 pixels and work with it. This is done by 'imutils.resize()' function of IMUTILS library. Every process we do is commonly applied to grayscale images as it is easy to gather things from grayscale pictures. OpenCV captures the frame in BGR (Blue, Green, Red) format by default. Therefore, in order to process the frame further, we must convert it to grayscale. To do that, 'cv2.cvtColor()' function is used, and we give in the parameter of 'cv2.COLOR_BGR2GRAY' to convert it into grayscale format. The grayscale frame is saved in a variable (imgGray) and will be used for further processing.

This is the processing we needed to do before we get to the analyzation stage of the video. That stage will be discussed in next topics.



Figure 7: Frame after Grayscale conversion

3.6 Drowsiness Detection

3.6.1 Face Detection

The first major thing to do for drowsiness detection with behavioral parameters is detecting whether there is a face present in the video which we can analyze or not. Our code will check for a face in the video and other detections will take place on that face. If there isn't a face present in

the video input, then there is no point of any detections afterwards and we should get an error message as a result.

The way to go for face detection was with the help of 'Dlib' library whose details are already shared before in the document. With this library's 68 facial landmarks, we are going to predict facial landmarks in the input video. First, we need to detect a face. For this, we use Dlib's function 'get_frontal_face_detector()' which gets the face information i.e., points where the face is present in the video and assign it to a Dlib object type variable (detector). After detecting the face, we have to predict the facial landmarks of that face. That is done with the help of Dlib's already trained model file for 68 landmarks. The file is available as 'shape_predictor_68_face_landmarks.dat' online and is used throughout the world for the sole purpose of predicting facial landmarks in a video or an image. Dlib's function 'shape_predictor()' predicts the facial landmarks and it takes in the path to the 68 facial landmarks file. We assign the result in another Dlib object type variable (predictor).

There can be many faces in the video, so in the while loop we started earlier, we use the 'detector' object and pass the Grayscale frame (imgGray) into it. The detector detects the faces and returns an array of rectangles (points) corresponding to each face detected in it. This array is stored in a variable (faces) which will be used afterwards for detections.

3.6.2 Mouth Detection

From the Dlib's facial landmarks, we can get the mouth coordinates as well. These facial landmarks can be seen in Figure 3. Numbers 49 to 68 define the mouth. In our case, we just need the outline of the mouth to correctly identify its shape and subsequently use those points to detect a yawn if there appears one. So, for this purpose, minimum 4 points are required i.e., left corner point, top-midpoint, right corner point, and bottom-midpoint. These points are identified by point

numbers 49, 52, 55, 58 respectively in the Dlib's 68 facial landmarks. Once we get these points through the Dlib's predictor object, we can easily identify the mouth.

3.6.3 Yawning Detection

Since we have the 4 required mouth points from mouth detection earlier, we can now move on to detect a yawn. The method pursued for this purpose is of getting ratio of mouth length to mouth height i.e., the horizontal line to the vertical line and check whether it is greater than a certain threshold. The left corner point and right corner point make up the horizontal line while top midpoint and bottom midpoint make up the vertical line. The coordinates of these points are given by the Dlib's predictor type object we created earlier and by passing the Grayscale frame into it as a parameter. Through this, we get all the landmark points of the face, store it in an object type variable, and then we can use the ones we need for our detections.

As we already have the points whose coordinates we need, we define and call a function with those point numbers (49, 52, 55, 58) and with the object type variable we stored the landmarks information in. In this function, we use the landmarks and each point one by one to get the x and y coordinates of that point. This is done for all 4 points one by one and storing the results inside variables to use them afterwards. To get the actual length of horizontal and vertical lines, `math.hypot()` function is used for left corner point's coordinates with right corner point's coordinates and top midpoint's coordinates with bottom midpoint's coordinates respectively. This function returns the Euclidean norm for given points and we save the resulting distance in variables. After that, we simply take the ratio of horizontal length to vertical length and return that ratio back to the function call.

Now since we have the mouth ratio, we just need to check whether it is greater than a certain threshold. The threshold was set as 1.65 upon trial and error and it detects a yawn perfectly. We can show the yawning message on the screen with `cv2.putText()` function.

3.6.4 Eyes Detection

The other important thing we need to do after face detection is detecting the eyes from the face. It is necessary for our drowsiness detection as it will be used for 2 things i.e., Eye blinks detection and PERCLOS estimation. Since we have already detected the face, our work in this part will be simple. We just need the landmarks from Dlib's predictor object as we got them before for yawning detection part. From figure 3 we know that landmark points 37 to 42 are for left eye and points 43 to 48 are for right eye. For subsequent work, we assigned these precise points of reference of both eyes to their respective variables.

3.6.5 Blinks Detection

There are 6 landmark points for each eye that we got during eyes detection i.e., left-most point, top-left point, top-right point, right-most point, bottom-right point, and bottom-left point. These points are described as p1, p2, p3, p4, p5, p6 respectively in figure 8 below.

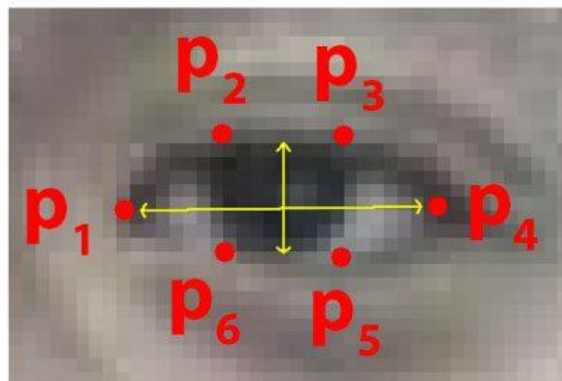


Figure 8: Eye landmarks [32]

For blinks detection, we need to get the Eye Aspect Ratio (EAR) for each eye. Before that, we need the Euclidean distance between p2 and p6 (say line A), p3 and p5 (say line B), p1 and p4 (say line C) to make two vertical distances and one horizontal distance respectively. EAR can be calculated by getting the ratio of Euclidean distances of (line A + line B) to (line C * 2).

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

Equation 1: Formula for calculating EAR [32]

We need the EAR for both eyes, so we define a function to calculate it through the above explained method. We pass the landmarks 37-42 for left eye and 43-48 for right eye to the function one by one to get the EAR for each eye. After calculating it for each eye, we add those ratios and divide it by 2 to get the Eye Ratio of both eyes simultaneously which can be used for blinks detection. If this Eye Ratio is less than 0.25 for 3 consecutive frames, we count a blink and counter a variable which keeps the record of how many blinks are detected up till a certain time. The counter can be also displayed on the screen with `cv2.putText()` function of OpenCV.

3.6.6 PERCLOS Estimation

PERCLOS is commonly known as Percentage Eye Closure. The ratio of closed-eye frames to total frames serves as its definition. We need to have both these values in order to calculate PERCLOS. From the previous part, we already have the blinks counter which is counting number of blinks and we know each blink consists of 3 consecutive frames. We need to calculate PERCLOS for each 10 seconds. To do that we need to have the total FPS of the recorded video which were saved in a variable (FPS) at the start before while loop. So now, we can set the number of frames at which we need our PERCLOS to be calculated i.e., 10 seconds. That makes our frames to be (FPS * 10) after 10 seconds. At each frame, we check whether the current number of frames are equal to that number of frames. If it is then we calculate PERCLOS by this formula i.e., (number of blinks * 3) / frames. To calculate the PERCLOS for those 10 seconds only which have just passed and not the whole time, we subtract previous value of PERCLOS from currently calculated value of PERCLOS to get Actual PERCLOS. Afterwards we will convert it into a percentage by multiplying the ratio by 100 and display it on the screen. The higher this percentage value is, the

drowsier the person will be. After the if statement is ended, we set previous PERCLOS to current PERCLOS so that we can calculate it again after 10 seconds.

3.6.7 Pose Estimation

For pose estimation, the MediaPipe library was used. The MediaPipe class “POSE” from this library is used to give all the key pose points that we saw in figure 4. We invoke this class by “mp.solution.pose” command and store it in a variable. Afterwards, we create a pose type object by calling the function Pose() for further use. For later use, we need to get the frame’s height and width by using “frame.shape” function and storing its first and second column only as its height and width respectively. For getting key pose points, we need to have the frame in RGB format, so we convert it into one by using cv2.cvtColor() function and giving in the frame and cv2.COLOR_BGR2RGB as arguments. Now we get the keypoints of a person’s pose by calling the function “process()” on pose type object with the current video frame as the function’s argument and store it in a variable (keypoints). After that, we convert the frame back to BGR format for further processing.

To get to each landmark and its coordinates, we need to go through a path which contains this information. This full path containing every single landmark’s information can be given as **“pose.process(frame).pose_landmarks.landmark[mp.solutions.pose.PoseLandmark.<Specific_Landmark>].coordinate”**. As this path is complicated and difficult to understand, we save some parts of it in different variables for our easiness to get to this path. The “pose.process(frame)” part was saved in variable “keypoints” before. Using this, we get variable **“lm”** that will contain the “keypoints.pose_landmarks” part which will have the list of the pose landmarks. The part of “mp.solution.pose” was already saved in a variable “mp_pose” in the beginning when we called the POSE class. Another variable called **“lmPose”** is created that uses the “mp_pose.PoseLandmark” path which will have each specific landmark in it, and which can be

called by name i.e., Left Shoulder, Right Shoulder etc. So now the path becomes very simple i.e., to get the left shoulder's x coordinate, our full path can be written as **"lm.landmark[lmPose.LEFT_SHOULDER].x"**. To get the actual pixel value of this landmark's x coordinate we need to multiply this with width of the frame which was stored earlier. Similarly, to get the actual pixel value of this landmark's y coordinate, we need to multiply this with height of the frame which was also stored earlier. If we do not find landmarks list in the frame i.e., "lm" value is "none", we skip that frame so that the processing doesn't stop or give an error. By this method, we get the x and y coordinates of Left Shoulder, Right Shoulder and Nose to detect the pose from and store them in their respective variables.

Finding the angle between the nose and the left and right shoulders is necessary because it is the main predictive factor in pose identification which will give us the driving head pose of the driver. We do this angle measurement by keeping following figure in mind.

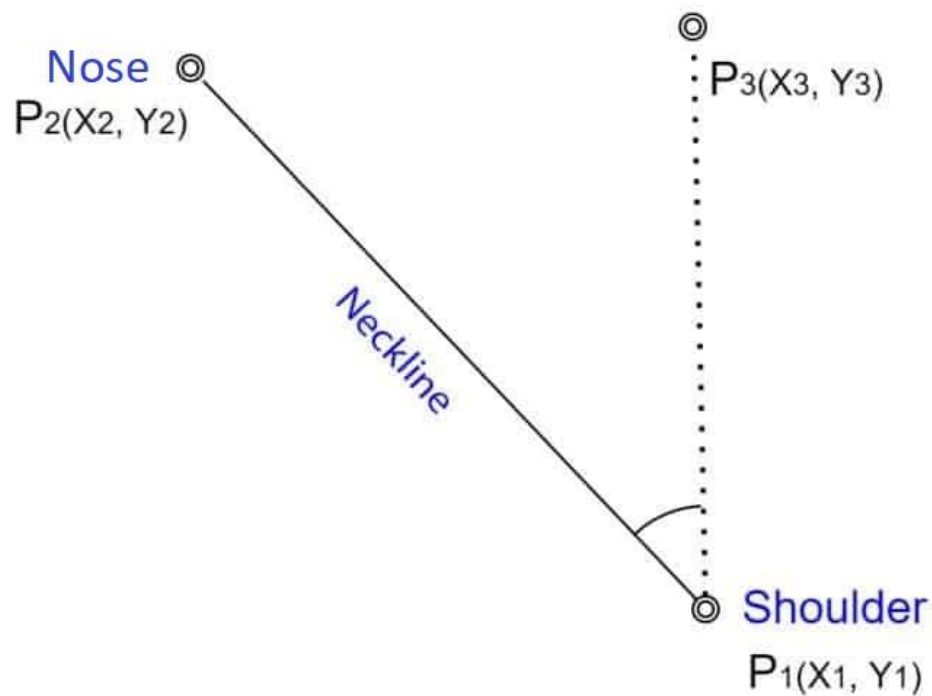


Figure 9: Nose Shoulder Angle Measurement [33]

Here, P1(x1, y1) stands for the shoulder, P2(x2, y2) for the eye, and P3(x3, y3) for the neckline (any point on the vertical axis passing through P1). Evidently, P3's x-coordinate matches P1's exactly. Let's assume y3 = 0 for the sake of simplicity since y3 is true for all y.

To determine the interior angle of three points, we use the vector method. Given by, is the angle between the two vectors P12 and P13:

$$\theta = \arccos\left(\frac{\vec{P}_{12} \cdot \vec{P}_{13}}{|\vec{P}_{12}| \cdot |\vec{P}_{13}|}\right)$$

Solving for θ we get,

$$\theta = \arccos\left(\frac{y_1^2 - y_1 \cdot y_2}{y_1 \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}\right)$$

Equation 2: Angle Between 2 Vectors [33]

This example is given taking Shoulder as the pivot point. The same is also true if we take the Nose as the pivot and calculate its angle with both Shoulders. It is more appropriate to do so because upon facial analysis, the direction of the Nose determines whether a driver's pose is correct or incorrect.

We need to make a function using equation 2 to calculate the angles between different keypoints. The function will take 4 values as arguments i.e., x1, y1, x2, y2 on which we will calculate angles. After calculating the theta angles (which are in radian), we need to convert those angles into degrees. Now we check whether they are exceeding or preceding a certain value. Upon trial and error, it is concluded that when the angle of Nose with respect to Left or Right Shoulder exceeds 142 degrees or precedes 127 degrees, the pose of the driver is incorrect.

3.6.8 Drowsiness Alert

After doing all the detections, we need to come up with a way to give a drowsiness alert to the driver. To do that, we have to have certain conditions which fulfill the requirements of drowsy state. In this study, we developed the following criteria to determine if a driver is currently experiencing or is headed toward drowsiness:

1. Mouth is open i.e., the driver's yawn is 3 seconds long.
2. Eyes are continuously closed for 3 seconds at least.
3. The PERCLOS percentage reaches 30% or more.
4. Pose is continuously incorrect for 3 seconds.

If any one of these conditions is met, we proceed to declare that the driver is drowsy and hence show up a message on the screen of the alert. This is done by OpenCV's `cv2.putText()` function through which we were already printing out the messages on the screen.

4 Results and Discussion

In this section, we will talk about the results obtained from our experiments.

4.1 Yawning

After we detected the face, we had to detect yawning if there was any. So just to see if the system is detecting the yawns correctly, the subject deliberately yawned.



Figure 10: Subject Yawning

In figure 10, a yawn is detected meaning the ratio of horizontal length to vertical length is greater than 1.65 threshold.

However, this yawning detection accuracy goes down with the time of day depending upon environmental lighting conditions. To visualize this, figure 11 shows the plot of detected yawning accuracy with respect to 4 different day times. i.e., Day, Noon, Evening, Night.

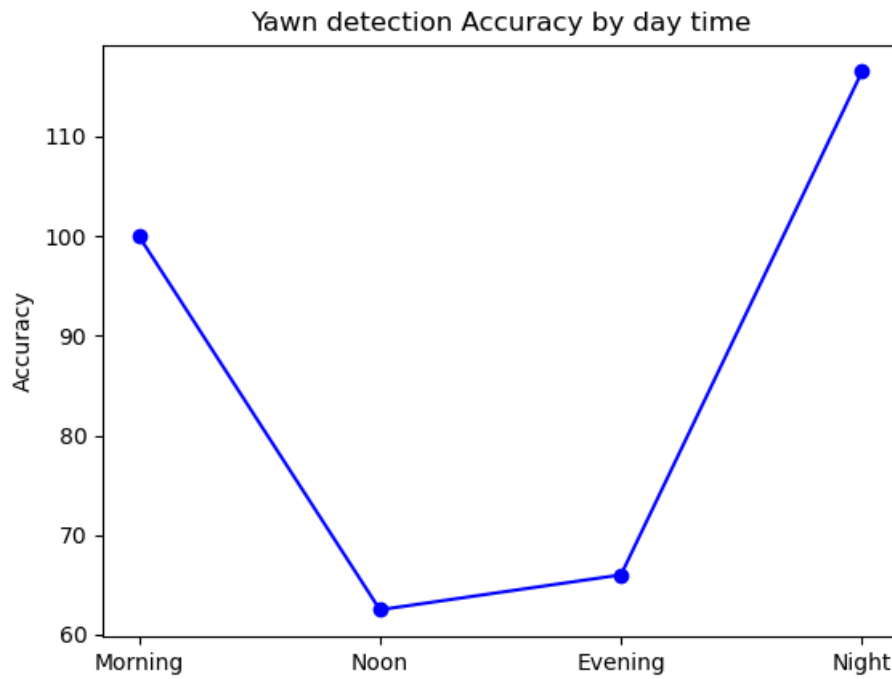


Figure 11: Yawn Detection Accuracy by Day time

This accuracy graph shows that the number of false positives increases in low light thus predicting more yawn than the actual yawns.

The experimental videos were of 2 types with respect to the recording camera angle. In 1st type, the camera was placed on the dashboard just behind the left of steering wheel (Infront) while in 2nd type, the camera was placed on the lap of the driver (On-lap). The accuracy for both these is given in figure 12.

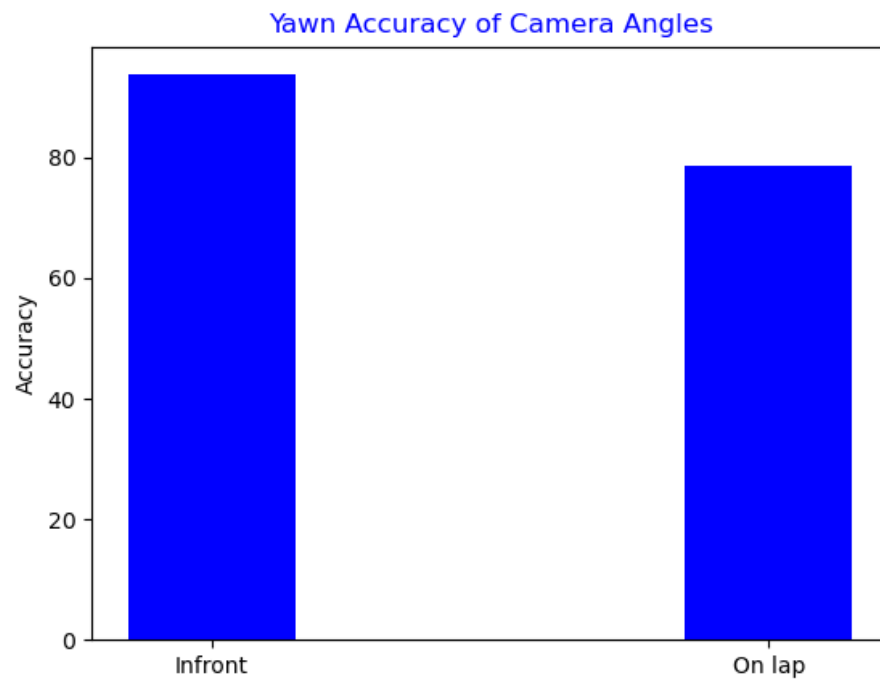


Figure 12: Yawn Detection Accuracy by camera angle

We can see that the Infront type has more accuracy than the On-lap type. So, for yawning, camera should be placed in front of the driver just behind the left of the steering wheel.

4.2 Blinking

Afterwards, we needed to detect eye blinks of the subject. In figure 10, it is shown that our code is detecting a blink of a subject successfully thus confirming that the EAR is less than 0.25 for 3 consecutive frames.



Figure 13: Subject Blinking

The same case as yawning is seen here. The no. of detected blinks increases with the low light environment. i.e., False positive rate increases. This can be viewed in figure 14.

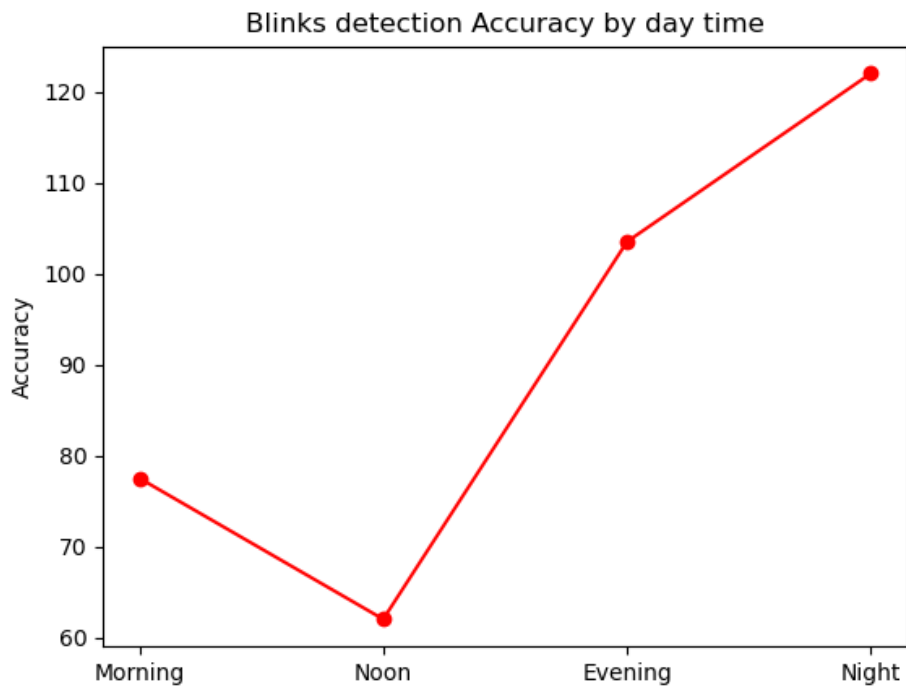


Figure 14: Blinks Detection Accuracy by Day time

The camera angle plot of blinks is given in figure 15.

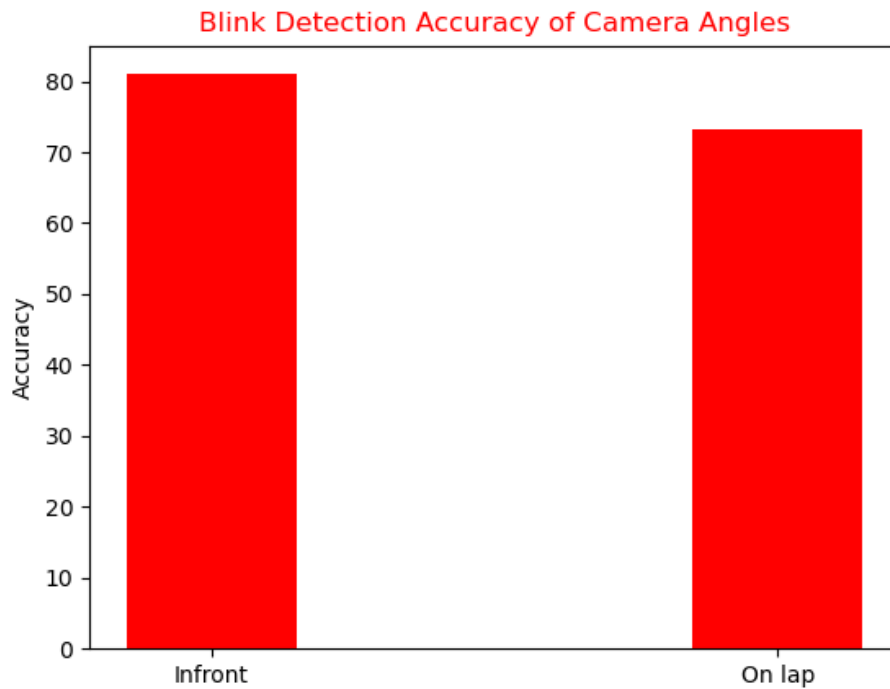


Figure 15: Blinks Detection Accuracy by Camera Angle

Same as yawning, the camera should be placed in front of the driver for accurate blinks detection.

4.3 PERCLOS

The PERCLOS was defined to be as percentage of time eyes were closed in 10 seconds. As soon as the subject starts blinking too much, this percentage goes higher showing the subject becoming drowsy. To put this into perspective, PERCLOS at different time frames are shown in figures below.



Figure 16: PERCLOS at the start of video, after 20 seconds, 30 seconds, 40 seconds

As can be seen in figure 12, PERCLOS is increasing after every 10 seconds in first 3 images while it reduces during the time period of 30 and 40 seconds. This happens because there were a smaller number of blinks during that period. Thus, concluding PERCLOS depends on number of blinks.

4.4 Pose



Figure 17: Subject's Pose while driving

The pose detection via MediaPipe can be seen in figure 17 above. The landmarks of Right Shoulder, Left Shoulder and Nose are connected together. The angle between Nose and Right Shoulder and angle between Nose and Left Shoulder are calculated and displayed.

This pose estimation accuracy also follows the trend and goes higher than 100% due to false positives during night-time. Its graph is shown below.

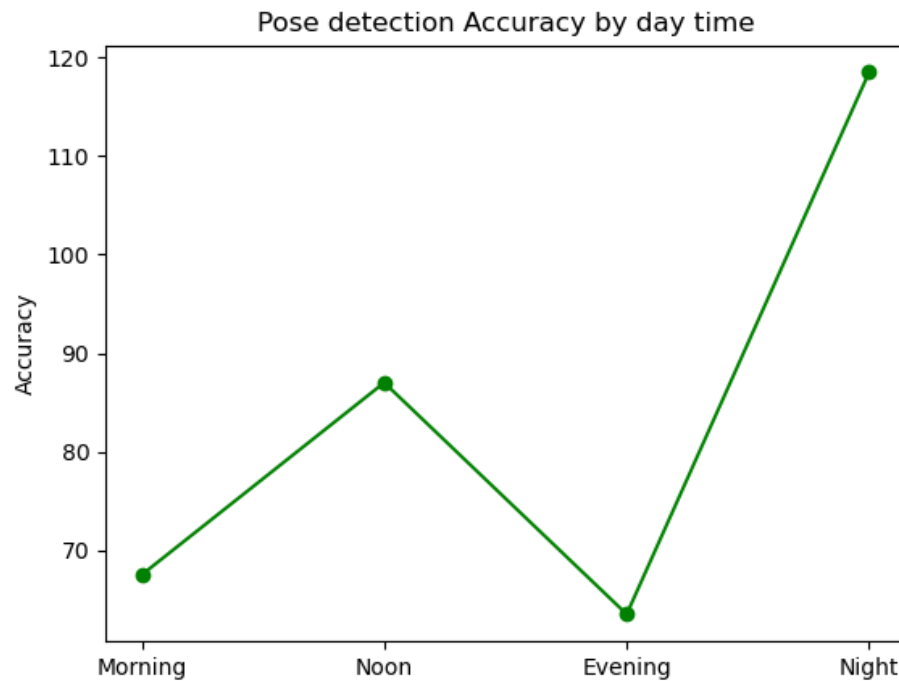


Figure 18: Pose Detection Accuracy by Day time

Similarly, the detection accuracy with respect to camera angles is maximum when the camera is Infront and it decreases when camera is on-lap. Graph is given below in figure 19.

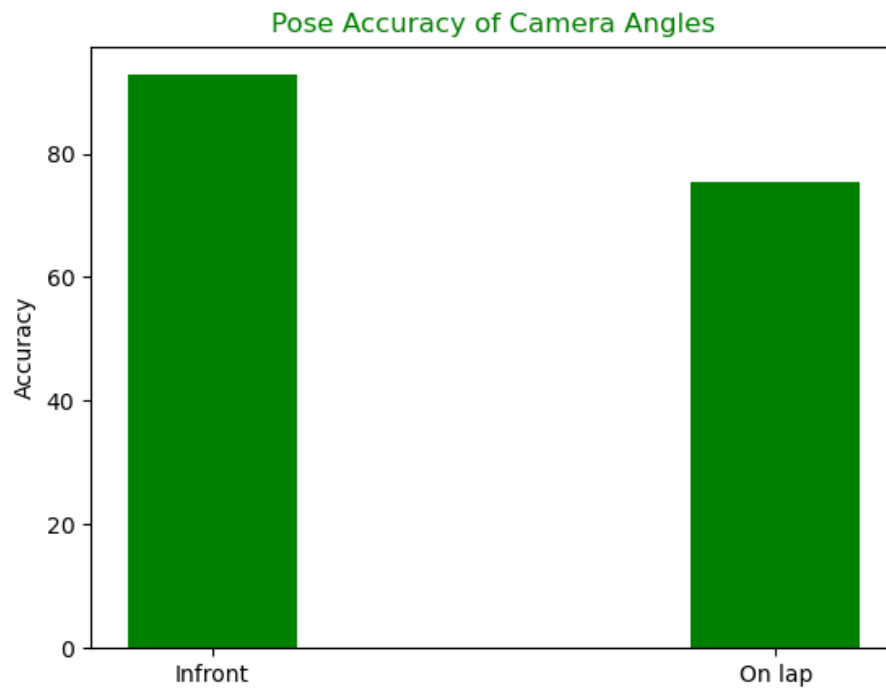


Figure 19: Pose Detection Accuracy by Camera Angle

4.5 Drowsiness Alert

Drowsiness is detected based on 4 conditions as explained earlier. If any one of those conditions come true, we can give a drowsy alert. Figure 20 shows how the alert is given when the driver's pose is incorrect for 3 seconds.

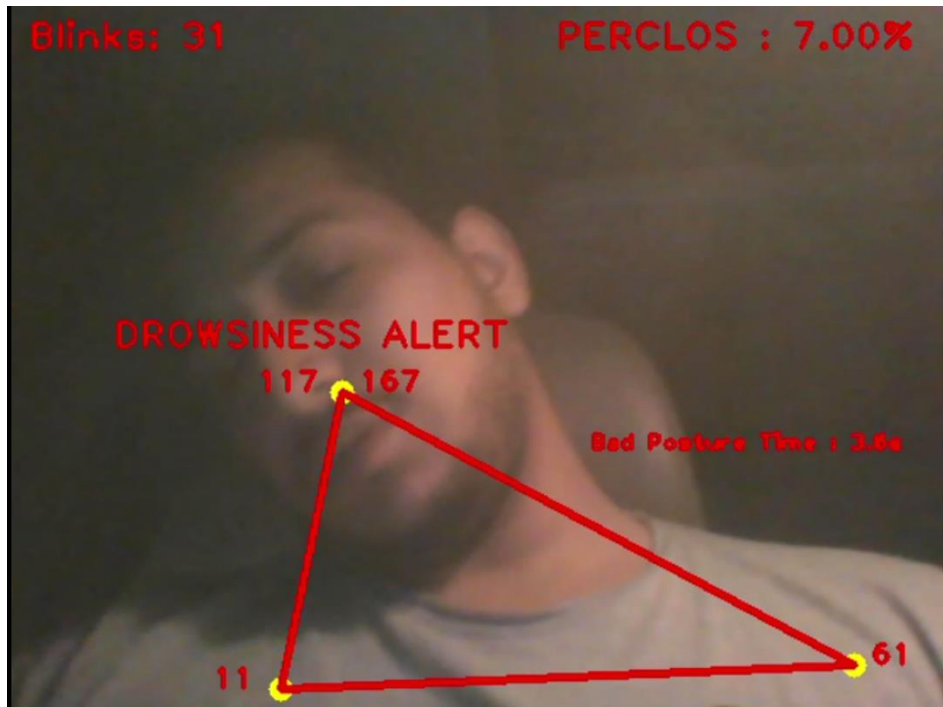


Figure 20: Drowsiness Alert Based on Incorrect Pose

The accuracy graph is also given with respect to different day time in figure 21.

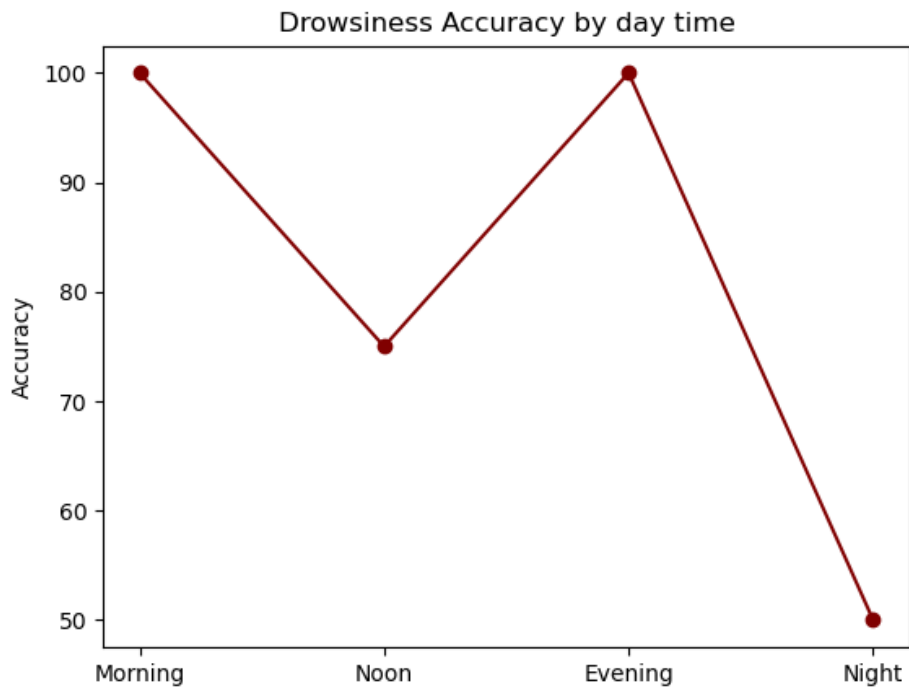


Figure 21: Drowsiness Detection Accuracy by Day time

The drowsiness accuracy goes down with regards to low lighting environments as can be shown in the graph.

The accuracy graph with respect to camera angles is given in figure 22.

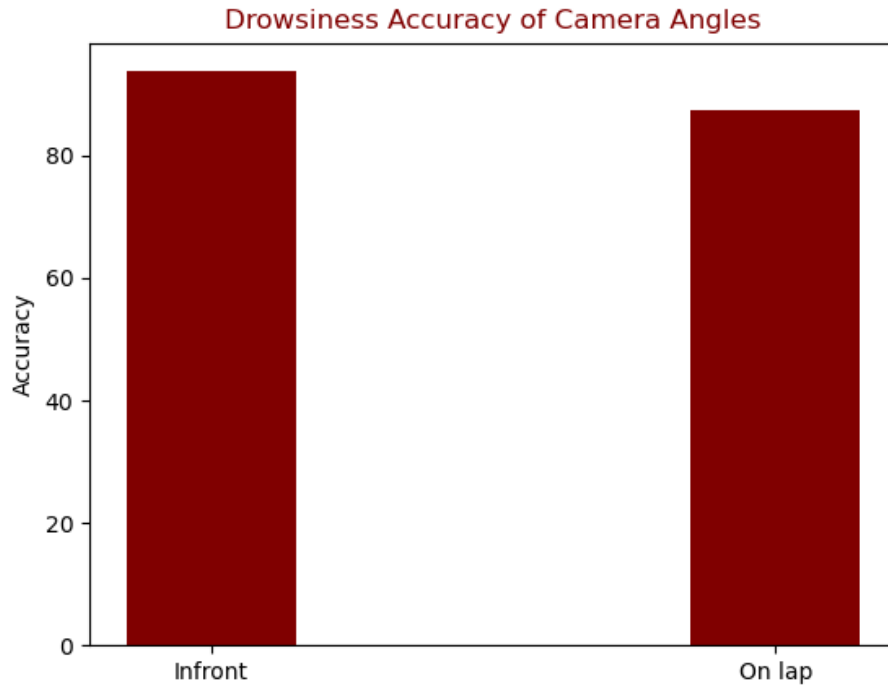


Figure 22: Drowsiness Detection Accuracy by Camera Angle

The drowsiness detection accuracy is the most when the camera is placed Infront of the driver.

4.6 Real-time Detection

One of the advantages of this system is that it can detect and predict all the parameters in real-time thus making this very useful. From the detections, it will also give an alert of drowsiness in real-time if drowsiness occurs. Figure 23 shows the real-time processing happening.

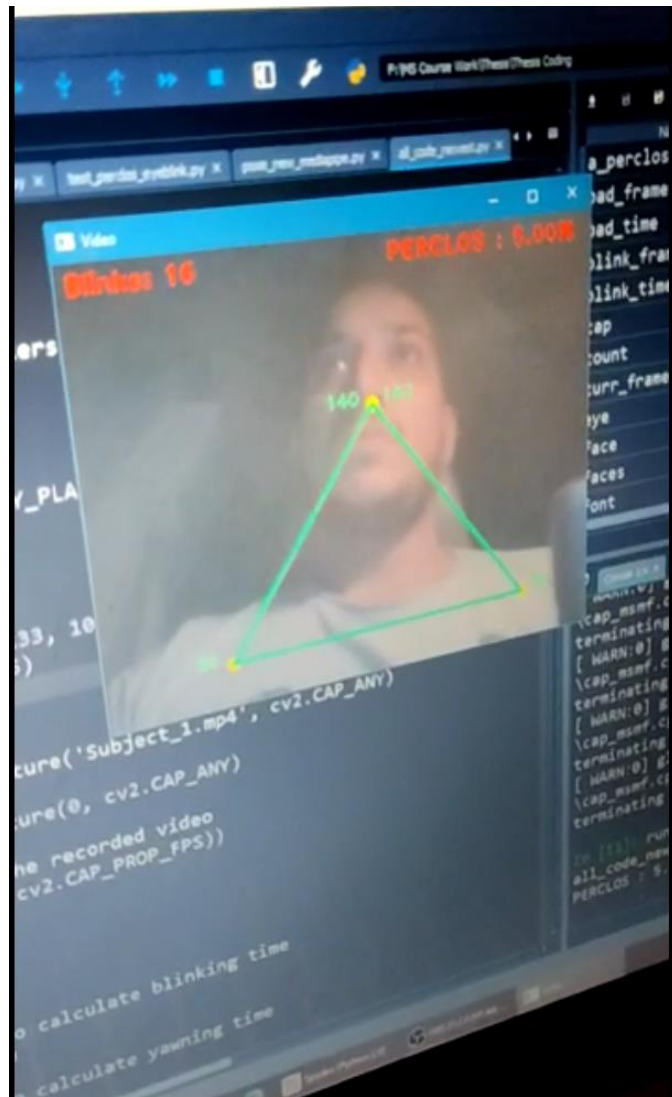


Figure 23: Real-time Processing

The processing is happening via laptop's webcam while the screenshot is taken by mobile phone to show real-time detection.

4.7 Limitations

The limitations of this work are that in low lighting conditions, it doesn't work accurately. Also another limitation is that when the camera is placed on the side of the driver, it doesn't detect any blink, yawn and also it doesn't correctly estimates the pose. Figures 24 & 25 both show these cases.

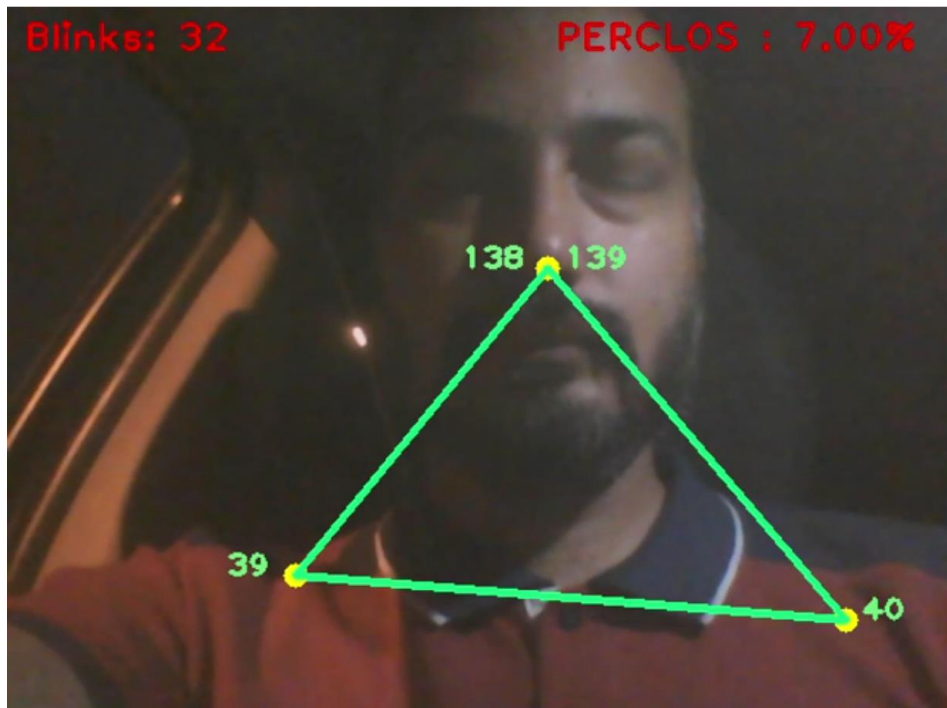


Figure 24: Night-time Detection



Figure 25: Side Pose Detection

In figure 24, blink is not detected while in figure 25, the blinks, yawn are not detected and also the angle is incorrectly estimated.

5. Conclusion

In this research, we saw how to get different features of a driver to tell us about if the driver is feeling drowsy. We used behavioral features such as Eye blinking, Yawning, PERCLOS and Pose. They were all thoroughly seen and calculated. We got to the conclusion that these 4 features together can give accurate results of driver drowsiness during daytime when the lighting conditions are bright. Also, we determined that the camera angle should be placed in front of driver on the dashboard just on the left side of behind the steering wheel.

6. Future Work

Following work is recommended for future work: -

- Use of infrared camera to compensate for low lighting conditions.
- Different facial expressions of the driver can be used to identify whether the driver is feeling relaxed, fatigued, or drowsy.
- A dataset in relaxed environments for driving can be gathered to give more accurate drowsy dataset to work on.
- Physiological and behavioral-based assessments can be made together to get even more accurate results.

7. References

- [1] M. Doudou, A. Bouabdallah, and V. Berge-Cherfaoui, "Driver Drowsiness Measurement Technologies: Current Research, Market Solutions, and Challenges," *Int. J. Intell. Transp. Syst. Res.*, vol. 18, no. 2, pp. 297–319, 2020.
- [2] A. Picot, A. Caplier, and S. Charbonnier, "Comparison between EOG and high frame rate camera for drowsiness detection," *2009 Work. Appl. Comput. Vision, WACV 2009*, 2009.
- [3] M. J. Flores, A. De La Escalera, and J. M. Armingol, "Driver drowsiness warning system using visual information for both diurnal and nocturnal illumination conditions," *EURASIP J. Adv. Signal Process.*, vol. 2010, 2010.
- [4] J. Singh, Hardeep., Bhatia, J. S., & Kaur, "Eye Tracking Based Driver Drowsiness Monitoring and Warning System," *Int. J. Tech. Res. Appl.*, vol. 3, no. 3, pp. 190–194, 2011.
- [5] W. Shen, H. Sun, E. Cheng, Q. Zhu, Q. Li, and W. Shen, "Effective driver fatigue Monitoring through pupil detection and yawing analysis in low light level environments," *Int. J. Digit. Content Technol. its Appl.*, vol. 6, no. 17, pp. 372–383, 2012.
- [6] R. O. Mbouna, S. G. Kong, and M. G. Chun, "Visual analysis of eye state and head pose for driver alertness monitoring," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1462–1469, 2013.
- [7] E. Tadesse, W. Sheng, and M. Liu, "Driver Drowsiness Detection through HMM based Dynamic Modeling," *2014 IEEE Int. Conf. Robot. Autom.*, 2014.
- [8] G. J. Al-Anizy, M. J. Nordin, and M. M. Razooq, "Automatic Driver Drowsiness Detection Using Haar Algorithm and Support Vector Machine Techniques," *Asian J. Appl. Sci.*, 2015.
- [9] O. Khunpisuth, T. Chotchinasri, V. Koschakosai, and N. Hnoohom, "Driver Drowsiness Detection Using Eye-Closeness Detection," *Proc. - 12th Int. Conf. Signal Image Technol. Internet-Based Syst. SITIS 2016*, pp. 661–668, 2017.
- [10] M. Babaeian, N. Bhardwaj, B. Esquivel, and M. Mozumdar, "Real time driver drowsiness detection using a logistic-regression-based machine learning algorithm," *2016 IEEE Green Energy Syst. Conf. IGSEC 2016*, 2016.
- [11] X.-P. Huynh, S.-M. Park, and Y.-G. Kim, "Detection of Driver Drowsiness Using 3D Deep Neural Network and Semi-Supervised Gradient Boosting Machine," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10116 LNCS, p. v, 2017.

- [12] A. Chellappa, M. S. Reddy, R. Ezhilarasie, S. Kanimozhi Suguna, and A. Umamakeswari, "Fatigue detection using Raspberry Pi 3," *Int. J. Eng. Technol.*, vol. 7, no. 2, pp. 29–32, 2018.
- [13] E. E. Galarza, F. D. Egas, F. M. Silva, P. M. Velasco, and E. D. Galarza, "Real time driver drowsiness detection based on driver's face image behavior using a system of human computer interaction implemented in a smartphone," *Adv. Intell. Syst. Comput.*, vol. 721, no. Icits, pp. 563–572, 2018.
- [14] W. Tipprasert, T. Charoenpong, C. Chianrabutra, and C. Sukjamsri, "A Method of Driver's Eyes Closure and Yawning Detection for Drowsiness Analysis by Infrared Camera," *2019 1st Int. Symp. Instrumentation, Control. Artif. Intell. Robot. ICA-SYMP 2019*, pp. 61–64, 2019.
- [15] U. Lahoti, R. Joshi, N. Vyas, K. Deshpande, and S. Jain, "Trends in Computer Drowsiness Detection System for Online Courses," *Int. J. Adv. Trends Comput. Sci. Eng.*, vol. 9, no. 2, pp. 1930–1934, 2020.
- [16] J. Gwak, A. Hirao, and M. Shino, "An investigation of early detection of driver drowsiness using ensemble machine learning based on hybrid sensing," *Appl. Sci.*, vol. 10, no. 8, 2020.
- [17] D. Kanade, A. Patil, V. Bang, M. Jayale, D. Dodal, and R. Katkamvar, "Driver Alertness Detection Using Opencv in Python," *Int. J. Eng. Appl. Sci. Technol.*, vol. 04, no. 06, pp. 99–102, 2019.
- [18] W. Mahdi, B. Akrouf, R. Alroobaea, and A. Alsufyani, "Automated drowsiness detection through facial features analysis," *Comput. y Sist.*, vol. 23, no. 2, pp. 511–521, 2019.
- [19] M. A. Puspasari, H. Iridiastadi, I. Z. Sutralaksana, and A. Sjafruddin, "Ocular indicators as fatigue detection instruments for Indonesian drivers," *Ind. Eng. Manag. Syst.*, vol. 18, no. 4, pp. 748–760, 2019.
- [20] B. Rajkumarsingh and D. Totah, "Drowsiness Detection using Android Application and Mobile Vision Face API," *R&D J.*, vol. 37, no. February, pp. 26–34, 2021.
- [21] R. R. Reddy and D. P. Pulicherla, "Driver drowsiness detection using MATLAB," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2, pp. 843–844, 2019.
- [22] B. Bakker *et al.*, "A Multi-Stage, Multi-Feature Machine Learning Approach to Detect Driver Sleepiness in Naturalistic Road Driving Conditions," *IEEE Trans. Intell. Transp. Syst.*, 2021.
- [23] A. K. Biswal, D. Singh, B. K. Pattanayak, D. Samanta, and M. H. Yang, "IoT-based smart alert system for drowsy driver detection," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021.
- [24] W. Deng and R. Wu, "Real-Time Driver-Drowsiness Detection System Using Facial Features,"

IEEE Access, vol. 7, pp. 118727–118738, 2019.

- [25] R. Jabbar, M. Shinoy, M. Kharbeche, K. Al-Khalifa, M. Krichen, and K. Barkaoui, “Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application,” *2020 IEEE Int. Conf. Informatics, IoT, Enabling Technol. ICloT 2020*, pp. 237–242, 2020.
- [26] A. L. A. Ramos, J. C. Erandio, E. M. Enteria, N. Del Carmen, L. J. Enriquez, and D. H. Mangilaya, “Driver Drowsiness Detection Based on Eye Movement and Yawning Using Facial Landmark Analysis,” *Int. J. Simul. Syst. Sci. Technol.*, pp. 1–8, 2019.
- [27] J. Y. Wong and P. Y. Lau, “Real-Time Driver Alert System Using,” *ECTI Trans. Electr. Eng. Electron. Commun.*, vol. 17, no. 2, pp. 193–203, 2019.
- [28] Z. Zhao, N. Zhou, L. Zhang, H. Yan, Y. Xu, and Z. Zhang, “Driver Fatigue Detection Based on Convolutional Neural Networks Using EM-CNN,” *Comput. Intell. Neurosci.*, vol. 2020, no. 3, 2020.
- [29] M. Kumar, “Driver drowsiness detection techniques: A review,” *Adv. Math. Sci. J.*, vol. 9, no. 6, pp. 3933–3938, 2020.
- [30] “Dlib 68 points Face landmark Detection with OpenCV and Python - Studytonight.” [Online]. Available: <https://www.studytonight.com/post/dlib-68-points-face-landmark-detection-with-opencv-and-python>. [Accessed: 12-Dec-2021].
- [31] “Pose - mediapipe.” [Online]. Available: <https://google.github.io/mediapipe/solutions/pose.html>. [Accessed: 12-May-2022].
- [32] “Mouse Cursor Control Using Facial Movements.” [Online]. Available: <https://pythonawesome.com/mouse-cursor-control-using-facial-movements/>. [Accessed: 12-Dec-2021].
- [33] “Building a Body Posture Analysis System using MediaPipe.” [Online]. Available: <https://learnopencv.com/building-a-body-posture-analysis-system-using-mediapipe/>. [Accessed: 12-May-2022].

Drowsiness Detection

ORIGINALITY REPORT

11%
SIMILARITY INDEX

8%
INTERNET SOURCES

5%
PUBLICATIONS

5%
STUDENT PAPERS

Handwritten signature
Seni... of Mechanical and
Manu... Engineering
(S...), Islamabad

PRIMARY SOURCES

1 Submitted to Higher Education Commission Pakistan
Student Paper **2%**

2 link.springer.com
Internet Source **1%**

3 Submitted to School of Business and Management ITB
Student Paper **<1%**

4 www.paideumajournal.com
Internet Source **<1%**

5 www.scielo.org.za
Internet Source **<1%**

6 Ramandeep Kaur, Ankit Guleria. "Digital Eye Strain Detection System Based on SVM", 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021
Publication **<1%**

7 www.fruct.org
Internet Source **<1%**

www.mdpi.com