

Drowsiness Detection System For Drivers Using Image Processing Technique

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Technology degree in Information Technology

by

Gayathri K.S (Register no 37120027)

Iyer Priya Eswar Padaleeswaran (Register no 37120032)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC**

**JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600
119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with “A” grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of Gayathri K.S (37120027) and Iyer Priya Eswar Padaleeswaran(37120032) who carried out the project entitled “ DROWSINESS DETECTION SYSTEM FOR DRIVERS USING IMAGE PROCESSING TECHNIQUES” under my supervision from November 2020 to April 2021.

Internal Guide

Jithina Jose, M.E., (Ph.D)

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing , **Dr.R.Subhashini M.E., Ph.D. , Head** of the Department of Information Technology for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide Ms. Jithina Jose for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

DECLARATION

I **Gayathri K.S (Reg no:37120027)** and **Iyer Priya Eswar Padaleeswaran (Reg no:37120032)** hereby declare that the Project Report entitled **Drowsiness Detection System for Drivers using Image Processing Techniques**_done by me under the guidance of **Ms. Jithina Jose** is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in INFORMATION TECHNOLOGY.

DATE:

PLACE: CHENNAI.

SIGNATURE OF THECANDIDATE

ABSTRACT

Drowsy driving is highly contributing to a number of road accidents these days. Car crashes can be avoided by implementing a system with alarm to alert drowsy drivers to focus on the road and help them to stay focused. A system is developed to detect driver drowsiness and trigger them with an alarm to alert drivers to prevent accidents, and reduce loss of life and sufferings. Several Techniques have been studied and analysed to conclude the best technique with the highest accuracy to detect driver drowsiness. In this work, we are using a real-time system that utilizes computerized camera to automatically track and process driver's eye and yawn using Python, dlib and OpenCV. A camera will be utilized so that it concentrates towards monitoring the driver's eye and yawn. A trigger is issued to alert the driver. The eyes of the driver and yawn are discovered close for a specific measure of casing, the proposed framework accept that the driver is falling asleep and an alarm of caution is issued.

TABLE OF CONTENT

CHAPTER NO	TITLE NAME	PAGE NO
	ABSTRACT	I
	TABLE OF CONTENTS	II
	LIST OF ABBREVIATION	IV
	LIST OF FIGURES	V
1	INTRODUCTION	
	1.1 GENERAL	1
	1.2 MOTIVATION	1
	1.3 OUTLINE OF THE PROJECT	2
	1.4 PROBLEM STATEMENT	2
	1.5 OBJECTIVE OF THE PROJECT	2
2	AIM AND SCOPE	
	2.1 AIM	3
	2.2 PROBLEM DEFINITATION	3
	2.3 LITERATURE REVIEW	3
	2.4 SCOPE	5
3	ALGORITHM AND METHODS	
	3.1 ALGORITHM	6
	3.2 EXISTING SYSTEM	6
	3.3 PROPOSED SYSTEM	7
	3.3.1 ADVANTAGES OF PROPOSED SYSTEM	7
	3.4 MODULES	7
	3.4.1 FACE DETECTION	7
	3.4.2 FACIAL MAPPING	8
	3.4.3 EYE DETECTION	9
	3.4.3.1 EYE ASPECT RATIO(EAR)	10
	3.4.4 YAWN DETECTION	11

	3.4.5 ALARM	12
4	SYSTEM REQUIREMENTS AND DESIGN	
	4.1 REQUIREMENTS	13
	4.1.1 HARDWARE REQUIREMENTS	13
	4.1.2 SOFTWARE REQUIREMENTS	13
	4.2 SOFTWARE DESCRIPTION	13
	ROLE OF PYTHON	13
	NUMPY	14
	SCIPY	14
	IMULITIS	14
	DLIB	14
	ESPEAK	15
	4.3 DESIGN	15
	4.3.1 OPEN CV	15
	4.3.2 MACHINE LEARNING	15
	4.3.3 IMAGE PROCESSING	16
	4.4 SYSTEM ARCHITECTURE	19
	4.4.1 USE CASE DIAGRAM	20
5	RESULTS	
	5.1 SOFTWARE SETUP	21
	5.2 EXPERIMENTAL RESULTS	21
	5.3 LIMITATION	24
6	CONCLUSION	
	6.1 CONCLUSION	25
	6.2 FUTURE WORK	25
	REFERENCE	26
	APPENDIX	27
	SOURCE CODE	27

LIST OF ABBREVIATION

- | | | |
|----|------|------------------|
| 1. | Dlib | Digital Library |
| 2. | EAR | Eye Aspect Ratio |

LIST OF FIGURE

FIGURE NO	TITLE OF THE FIGURE	PAGE NO
3.1	Facial landmarks by opencv	8
3.2	Facial landmarks set which is detected via dlib	9
3.3	Eye aspect ratio	11
4.1	python	14
4.2	opencv	15
4.3	Architecture diagram	19
4.4	use case diagram	20
5.1	sample output 1	22
5.2	sample output 2	22
5.3	sample output 3	23
5.4	sample output 4	23

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Drowsy driving has become a serious issue in our society not only because it affects those who are driving, but also all other road users in danger. Accidents due to sleep deprivation is a common problem all around the world. According to a database study conducted by the Australian Transport Bureau in 2002 about fatality crashes it is found that 16.6% of fatal crashes were caused by sleep deprivation and based on the report submitted by Ministry of Transportation of Ontario in 2004, 25.5% of injury causing crashes and 17.8% of crashes with fatality related to sleep deprivation. All of the above statistics is an estimate because in several cases the drivers will not admit the real cause to the Police. The aim of this project is to develop a prototype for detecting drowsiness called drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the driver's facial expressions in real-time. Our first challenge is how to define fatigue exactly and how to measure it. Despite the progress of science, there is still no precise definition for fatigue. Certainly, due to the lack of precise definition of fatigue, there is not any measurable criterion or tool. However, a precise definition for fatigue is not defined yet, but there is a relationship between fatigue and some symptoms including body temperature, eye movement, heart rate. One of the most important symptoms of fatigue appears in the eye. This project helps drivers to get prevented from drastic accidents and the loss of life and property can be saved.

1.2 MOTIVATION

- Driver's safety is an important factor in today's generation. Accidents occur mainly because of aging factor, fatigue, alcohol and distraction.
- The U.S. National Highway Traffic Safety Administration has reported that driving while drowsy is one of the reasons behind road accidents.
- Based on the Ontario Ministry of Transportation's Driver's Handbook, drivers' abilities are affected by drowsiness and fatigue long before they notice that they are getting tired.

- The above are therefore motivations to design and implement an assistive monitoring system in order to detect drivers' drowsiness and fatigue.

1.3 OUTLINE OF THE PROJECT

The project is all about detecting drowsiness for drivers. This project helps the driver to avoid accidents. We are using dlib facial landmark function for detecting the facial points. An alarm is been setup so that the driver wakes up. Alarm is issued when the driver closes his eye or yawn for a substantial amount of time. In this case, an alarm is issued and the driver wakes up. We are using a software called eSpeak. eSpeak is an open source software which helps in conversion from text-to-speech.

1.4 PROBLEM STATEMENT

Designing a prototype Drowsiness Detection System which will focus on continuously and accurately monitoring the state of the driver's eyes in real time to check whether they are open or closed for more than a given period of time. Driver's inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, that is decreased driving performance, longer reaction time, and an increased risk of crash involvement.

1.5 OBJECTIVE OF THE PROJECT

Driver drowsiness detection is a car safety technology which spares the life of the driver by avoiding mishaps when the driver is getting languid. The framework works disregarding driver wearing displays and in different lighting conditions. To caution the driver on the identification of laziness by utilizing ringer or alert. Traffic management can be maintained by reducing the accidents.

CHAPTER 2

AIM AND SCOPE

2.1 AIM

The aim of our project is to design a system that will detect drowsiness and take necessary steps to avoid accidents. The driver drowsiness detection system, which is been implemented in this project aims that it can easily be available and can be used with different types of vehicles.

2.2 PROBLEM DEFINITION

Fatigue is a safety problem that has not yet been deeply tackled by any country in the world mainly because of its nature. Fatigue, in general, is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue-related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative.

2.3 LITERATURE REVIEW

According to the Survey on Driver Fatigue-Drowsiness Detection System, the detection system includes the processes of face image extraction, yawning tendency, blink of eyes detection, eye area extraction etc.

Tuncer et al. [1] proposed an assistant system to track a lane, which will be activated for those drivers who are not able to perform a good job of lane keeping. Another method of face detection based on locating facial features is developed by Graf et al. [2].this method states that the morphological operations will be applied to find the areas that have high intensity with certain shapes. In the drowsiness detection method proposed by Pilutti et al, [3] driver assessment is determined in the context of a road departure warning and intervention system. This method states that the vehicle lateral

position is used as the input and steering wheel position as the output in order to develop a system that will be updated during driving. Craw et al. [4] proposed the following method: the frontal view face is detected based on template matching. The edges extracted from Sobel filtering will be grouped together to locate the face.

The same procedure has to be repeated for finding other facial features such as eyes, mouth and nose in the candidate face. Sigary, [5] proposed a method for hypo vigilance detection by processing of the eye region and without an explicit eye detection stage.

Tabrizi and Zoroofi [6] proposed a non-intrusive and simple way of fatigue detection by determining whether the eye is open or closed. The algorithm consist of three steps which were analyzed, such as determining eye regions by eye map and locating pupil center by the center of mass of the eye region image and the refining the pupil center and detecting the iris boundary. Omidyeganeh et al. [7] used a method of fatigue detection by applying the Structural Similarity Measure to find the eye location. In this method, the structural similarity value will be evaluated between -1 and 1. When images are the same, the max gained value will be 1 and when there are some differences, the result will be -1. Then the horizontal and vertical projection will be applied on the eye region to determine the amount of eye closure and align the detected eye region. M. Saradadevi and P.Bajaj,[8]proposed Driver Fatigue Detection Using Mouth and Yawning Analysis, IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 6, pp. 183-188, June 2008.

The robust and reliable method of face detection based on the Viola-Jones theory has been used by Wang and Shi [9] in order to limit the mouth search area to the face region. The mouth region is located based on multi-threshold binarization in intensity space and by using the Gaussian model in RGB color space. The lip corner will be found by calculating the integral projection of the mouth in the vertical direction. The two lines which are running through the lower and upper lip boundaries that is resulting from the integral projection which represents the mouth openness. In this method, the yawning will be determined by finding the degree of mouth opening in terms of the aspect ratio of the mouth bounding rectangle. A huge mouth opening over a predefined threshold for a continuous number of frames means that the driver is in a state of drowsiness.

A. Cheng et. al. [10] described 'Driver Drowsiness Recognition Based on Computer Vision Technology'. They presented an nonintrusive drowsiness recognition method using eye-tracking and image processing. Robust eye detection algorithm was introduced to address the problems caused by changes in illumination and driver posture.

2.4 SCOPE

- To automatically recognize the sleep and drowsiness of the driver and its detection.
- To analyze the focus of driver while driving.
- To determine concentration of drivers while driving.
- To detect whether the camera is focusing properly and the driver's face.
- To verify the drowsiness.

CHAPTER 3

ALGORITHM AND METHODS

3.1 ALGORITHM

The overall algorithm is pretty straightforward one. First we have used a camera which is setup at desirable position in a car that looks for faces stream.

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

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

On the off chance that the Eye Aspect Ratio demonstrates that the eyes and yawn are shut for a considerably long measure of time, the alert will sound noisy to wake the driver up. For the functionalities of the system and to make it work efficiently we have used OpenCv, dlib and Python.

The implementation of the drowsiness detector system includes machine learning algorithms which are in turn included in OpenCv ML algorithms. There are numerous ML algorithms but for our purpose we required only the face detector algorithm.

3.2 EXISTING SYSTEM

In the driver face monitoring systems, useful symptoms for fatigue and distraction detection can be divided into three general categories:

- (i) symptoms related to the eye region;
- (ii) symptoms related to the mouth region;
- (iii) symptoms related to the head.

Eye is the most important area of the face where the symptoms of fatigue and distraction appear in it. Therefore, many of the driver face monitoring systems detect driver fatigue and distraction only based on the symptoms extracted from the eyes. The symptoms related to eye region include PERCLOS eyelid distance, eye blink speed, eye blink rate, and gaze direction. Yawning is one of the hypo vigilance

symptoms related to the mouth region. This symptom was extracted by detecting the open mouth. The existing system of driver drowsiness detection system has following disadvantages. There are mainly, using of two cameras in the system one for monitoring the head movement and the other one for facial expressions. The other disadvantage is aging of sensors and all these sensors are attached to the driver's body which may affect the driver.

3.3 PROPOSED SYSTEM

3.3.1 ADVANTAGES OF PROPOSED SYSTEM

- Efficiency while driving and safe driving.
- Decreased death rates.
- Reduced manual work.
- As the system is automated it doesn't require
- more resources like hand written record of
- Driver's safety, but the record is maintained in the database.
- The system has less hardware requirements in comparison to the other biometric system.
- As the system uses fewer resources therefore the cost of the system is less.
- The system also reduces the human effort.
- No fatigue while driving.

3.4 MODULES

- Face detection
- Facial Mapping
- Eye detection
- Yawn detection
- Alarm

3.4.1 FACE DETECTION

For face detection itself, several approaches have been used in the related literature. Knowledge based methods try to encode human knowledge about the characteristics of a typical face, such as the relationships between facial features, and use them as a way to detect faces in an image. The goal of the feature invariant approaches is to find structural face features, such as eyebrows, eyes, nose, mouth

and hairline, which persist under various poses, viewpoints or lighting and use those features to detect faces. Such features are mostly extracted using edge detectors. The video is obtained from a camera focused on the driver's face. The Haar-based classifier contains several features such as heights, weights, face features, the threshold of face colours. It is constructed by using a lot of positive and negative samples. The cascade consists both positive and negative samples. Eyes and mouth features are extracted and parallel processing is preceded by successful driver's face detection.

3.4.2 FACIAL MAPPING

The algorithm is implemented using a Dlib python library that contains a landmark's facial detector with pre trained models. The 68-point shape predictor dataset was used to train the dlib facial landmark predictor and is the source of these marking.

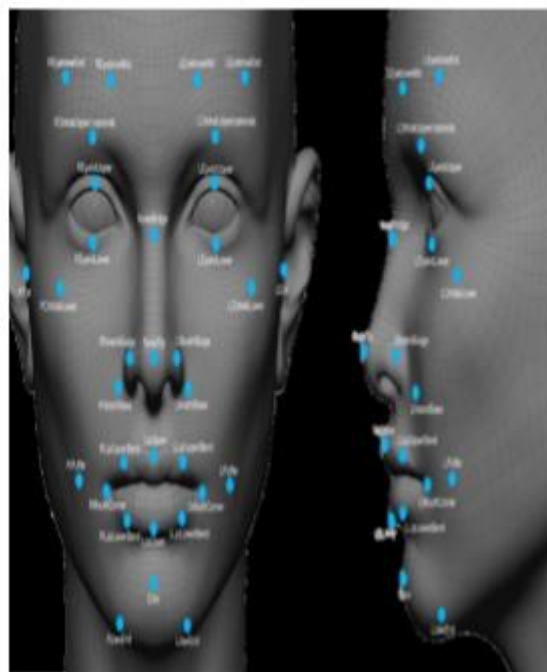


Fig.3.1 Facial landmarks by opencv

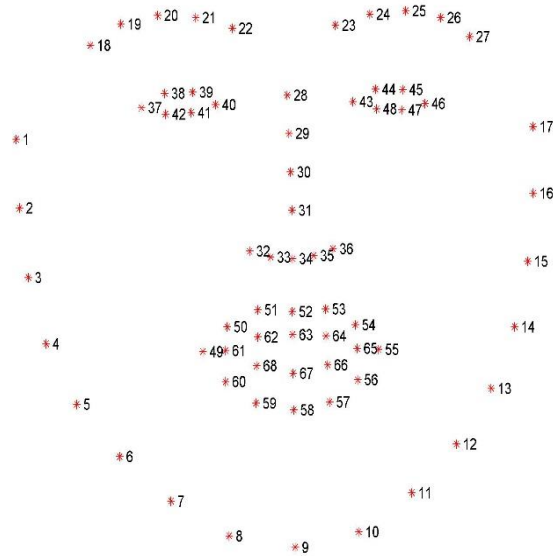


Fig 3.2 Facial landmarks set which is detected via dlib

3.4.3 EYE DETECTION

Eye tracking is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head. An eye tracker is a device for measuring eye positions and eye movement. Eye trackers are used in research on the visual system, in psychology, in psycholinguistics, marketing, as an input device for human-computer interaction, and in product design. Eye trackers are also being increasingly used for rehabilitative and assistive applications (related for instance to control of wheel chairs, robotic arms and prostheses). There are a number of methods for measuring eye movement. The most popular variant uses video images from which the eye position is extracted. We here are using eye tracking in detecting the drowsiness of the driver. Eye tracking is helping us to detect and sense the sleep of the driver, whether he is sleeping, wanting to sleep, getting exhausted while driving etc. We are first trying to generate a preview of the driver through the web camera of the laptop. The driver's preview image is being captured simultaneously by the web camera and the camera forms to give us a preview image. The camera now starts to record the video and automatically saves it in the backend so that it can be analyzed. After the images, videos, frames are being recorded and saved, the eyes are detected from them and then tracked.

3.4.3.1 EYE ASPECT RATIO (EAR)

The eye tracking involves to calculate the value for the eye aspect ratio. Eye aspect ratio (EAR) is a parameter that Determines eye state used to figure out if it is Open or closed. It can be calculated using facial landmarks plotted by the 68 facial landmark point plot provided by python's dlib library. Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye-lids are down for a substantial amount of time. Once we get the eye region, we calculate the Eye Aspect Ratio to find

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

out if the eye-lids are down for a substantial amount of time.

Where $p_1 \dots p_6$ are 2D facial landmark location.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only one set of horizontal points but two sets of vertical points.

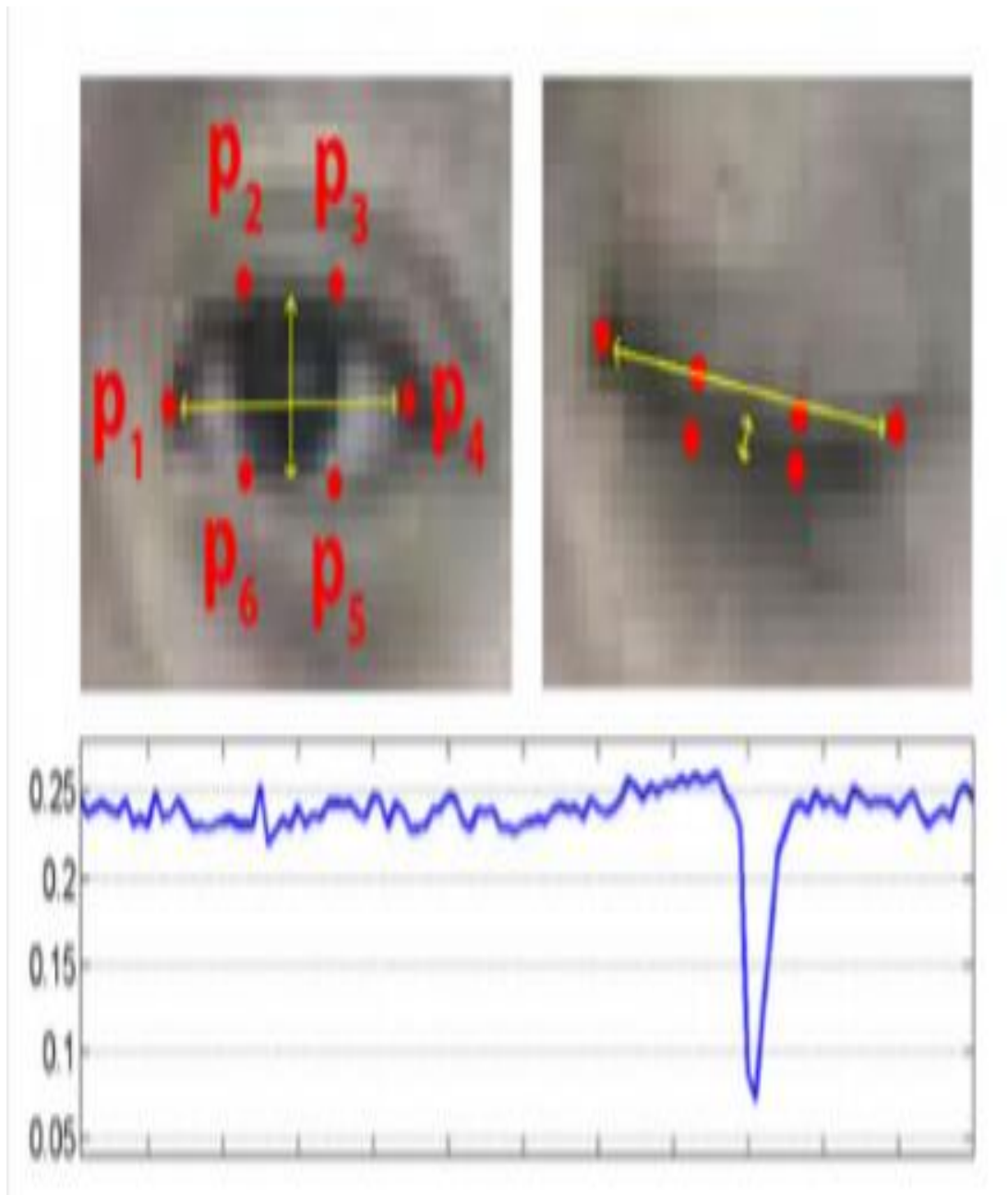


Fig 3.3 Eye aspect ratio

3.4.4 YAWN DETECTION

Yawning is characterized by a widely opened mouth. Yawning detection is performed in two main steps: In the first step, the yawn component will be detected in the face independent of the mouth location. This component is basically the hole in the face as the result of wide mouth opening. In the second step, the mouth location will be used to verify the validity of the detected component. Like the eye closure detection, the facial landmarks are used to detect an open mouth. Lip distance is the

parameter used to determine if the subject's mouth is open. the lip distance calculated from the frame is above lip distance threshold, the subject is determined to be yawning. An alarm is raised if the subject has yawned more than the set boundary value consecutively. Small openings that in reality are construed as a result of talking, eating are ignored.

3.4.5 ALARM

The alarm sound alerts the driver in case of drowsiness is detected. Instead of an alarm i.e buzzer we are giving here an text voice to alert the driver. For that we are using a module called eSpeak.

The eSpeak NG is a compact open source software text-to-speech synthesizer for linux, windows and other operating systems.

CHAPTER 4

SYSTEM REQUIREMENT AND DESIGN

4.1 REQUIREMENT

4.1.1 HARDWARE REQUIREMENT

- A system with minimum 8GB RAM.
- Camera
- Alarm

4.1.2 SOFTWARE REQUIREMENT

- Python version 3.8 installed in the system.
- Using pip command
 - Numpy
 - Scipy
 - Playsound
 - Dlib
 - Imutils
 - Opencv
 - eSpeak

4.2 SOFTWARE DESCRIPTION

ROLE OF PYTHON

Python is an interpreted, high-level, general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.



Fig 4.1:Python

NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.

SCIPY

SciPy in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands. SciPy is built on the Python NumPy extension.

IMUTILS

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

DLIB

It's a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face.

eSpeak

eSpeak is a compact open source software speech synthesizer for English and other languages, for Linux and Windows.

4.3 DESIGN

4.3.1 OPEN CV

OpenCV is an open source computer vision library accessible in python coding language to code for visionary capabilities of our smart pc. OpenCV was expected for computational capability and having a high focus on ongoing picture location and distinguishing proof. OpenCV is coded with streamlined C and can take work with multicore processors. If we need progressively programmed improvement utilizing Intel models [Intel], you can purchase Intel's Integrated Performance Primitives (IPP) libraries [IPP]. These comprise of low-level schedules in different algorithmic regions which are streamlined. OpenCV consequently utilizes the IPP library, at runtime if that library is introduced.



Fig 4.2: Opencv

4.3.2 IMAGE PROCESSING

In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images.

4.3.3 MACHINE LEARNING

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

CLASSIFIER

Classification is the process of predicting the class of given data points. Classes are sometimes called as targets/ labels or categories. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

K-means Algorithm

It is an unsupervised clustering algorithm which signifies a distribution of data w.r.t. K centers, K being chosen by the coder. The difference between K-means and expectation maximization is that in K-means the centers aren't Gaussian. Also the clusters formed look somewhat like soap bubbles, as centers compete to occupy the closest data points. All these cluster areas are usually used as a form of sparse histogram bin for representing the data.

Normal or Naïve Bayes classifier algorithm

It is a generative classifier where features are often assumed to be of Gaussian distribution and also statistically independent from one another. This assumption is usually false. That's why it's usually known as a —naïve Bayes classifier. That said, this method usually works surprisingly well.

Decision trees algorithm

It is a partially discriminative classifier. The tree we talk about just finds a singular data feature and determines a threshold value of the current node which best divides the data into different classes. The data is broken into parts and the procedure is recursively repeated through the left as well as the right branches of the decision tree. Even if it is not the top performer, it's usually the first thing we try as it is fast and has a very high functionality.

Boosting

It is a discriminative group of classifiers. In boosting, the final classification decision is made by taking into account the combined weighted classification decisions of the group of classifiers. We learn in training the group of classifiers one after the other. Each classifier present in the group is called a weak classifier. These weak classifiers are usually composed of single-variable decision trees known as —stumps. Learning its classification decisions from the given data and also learning a weight for its vote based on its accuracy on the data are things the decision tree learns during training. While each classifier is trained one after the other, the data points are re-weighted to make more attention be paid to the data points in which errors were made. This continues until the net error over the entire data set, obtained from the combined weighted vote of all the decision trees present, falls below a certain threshold. This algorithm is usually effective when a very large quantity of training data is available. [4]

Random trees algorithm

It is a discriminative woods of a great deal of choice trees, every one of which is worked down to a maximal part profundity. At the season of adapting, each hub of each tree is permitted a decision of part factors, however just from an arbitrarily produced subset of the considerable number of information highlights. This makes sure that all the trees become statistically independent and a decision maker. In the run mode, all the trees get an unweighted vote. Random trees are usually quite effective. They can *also perform regression by taking the average of the output numbers from every tree.*

Face detector algorithm

It is an object detection application. It is based on a smart use of boosting. A trained frontal face detector is available with the OpenCV distribution. This works remarkably well. We can train the algorithm for other objects by using the software provided. This works wonderfully for rigid objects.

Expectation maximization (EM) algorithm

It is used for clustering. It is a generative unsupervised algorithm. It fits N multidimensional Gaussians to the data, N being chosen by the user. It can act as an efficient way for representing a more complex distribution using only a few parameters (i.e. means and variances). Usually used in segmentation, it can be compared with K-means.

K-nearest neighbors

It is one of the simplest discriminative classifiers. The training data is simply stored using labels. Then, a test data point is classified in accordance to the majority vote of the K nearest data points. K-nearest neighbours is probably the simplest algorithm we can use. It is usually effective but it can be slow. It also requires a lot of memory.

Neural networks / Multilayer perceptron (MLP) algorithm

It is a discriminative algorithm which almost always contains hidden units in between the output and the input nodes for better representation of the input signal. It is slow to train, however it is quite fast to run. It remains the best performer for applications like letter recognition.

Support vector machine (SVM) algorithm

It is a discriminative classifier that is likewise equipped for doing relapse. Here, a separation work in the middle of two information focuses is characterized in a higher-dimensional space. (Anticipating information onto higher measurements helps in making the information more probable for direct division.) Support vector machine (SVM) gets the hang of isolating hyperplanes which maximally separate every one of the classes in the higher measurement. This will in general be the best when there is

restricted information. Be that as it may, when huge informational indexes are accessible, boosting or arbitrary trees are liked.

4.4 SYSTEM ARCHITECTURE

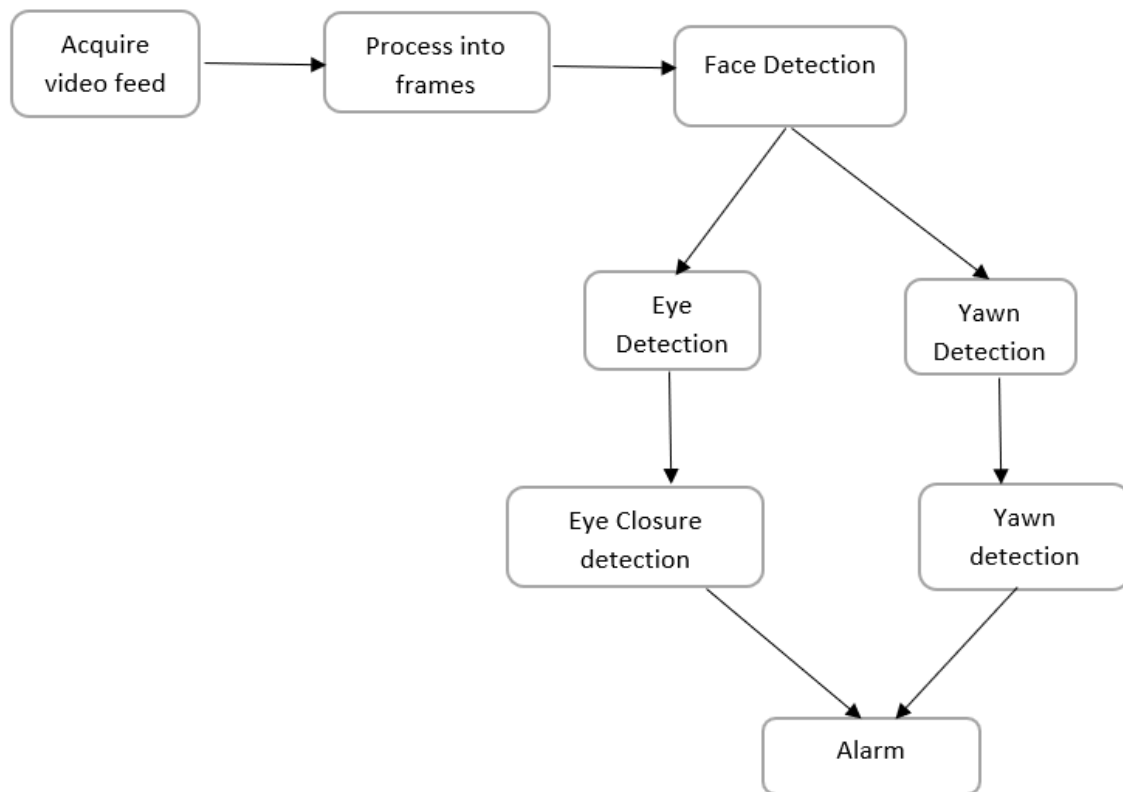


Fig 4.3 Architecture diagram

The diagram depicts how the full function of the drowsiness detection is carried out.

It describes the importance of each step that is required to complete the detection of drowsiness.

We can see how first the brightness and contrast level of the camera is adjusted.

Then the face is detected.

If it is successfully previewed then only the further step is taken.

The eye detection takes place.

The decision for proper eye detection is taken and then the eye region is focused and extracted.

The eyes are determined whether they are closed or opened.

The drowsiness is calculated through the data being stored and saved.

The drowsiness is judged now.

If the driver is drowsy then alarm rings loud and alerts the driver.

This is how the detection for driver drowsiness system works and it alerts the driver and the accidents is prevented from being occurred.

4.4.1 USE CASE DIAGRAM

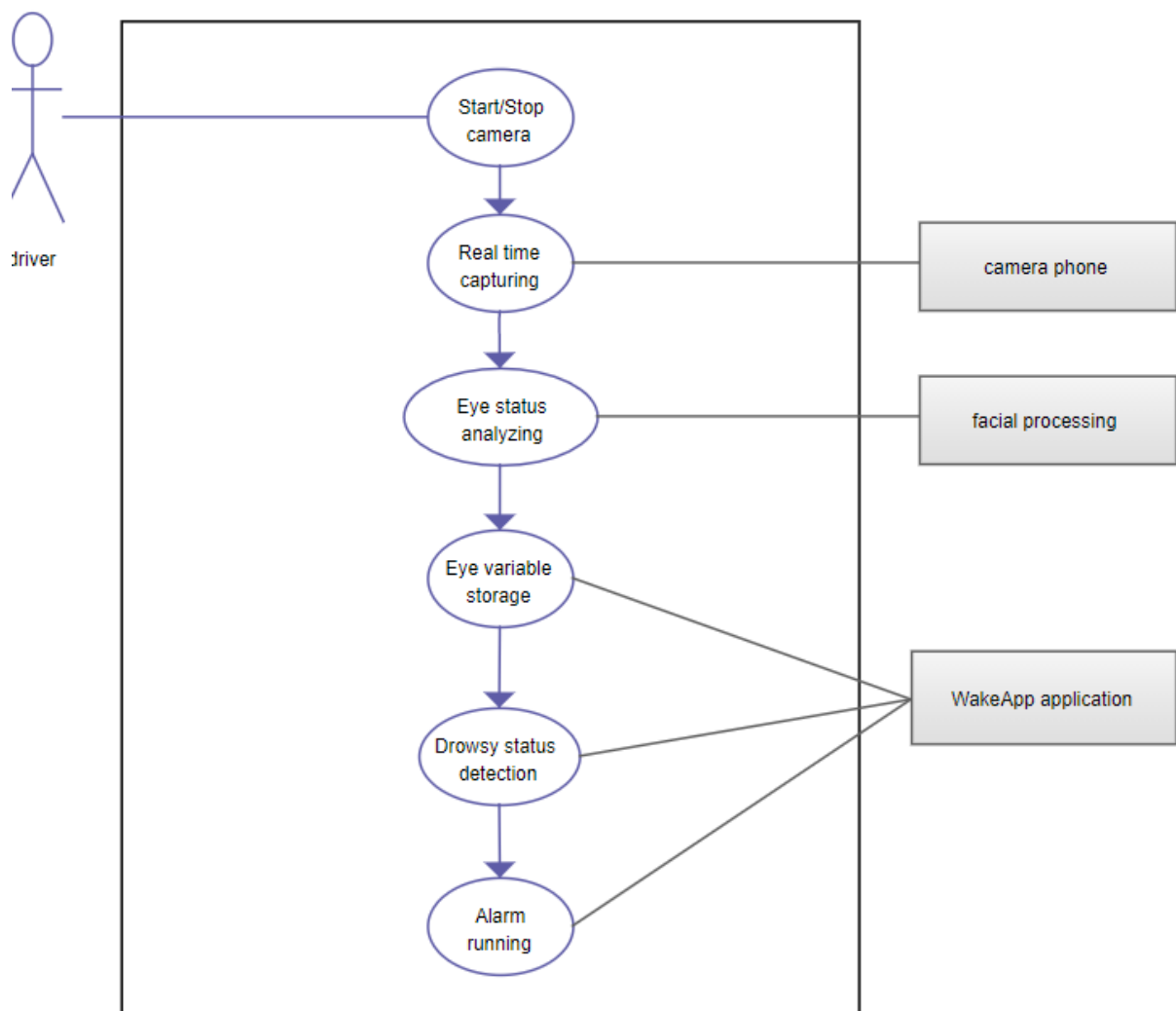


Fig 4.4 use case diagram

CHAPTER 5

RESULTS AND DISCUSSION

5.1 SOFTWARE SETUP

For the execution of the project we need to have a software called Python with version 3.8. After the installation of the software we need to have certain library function such as dlib, numpy, scipy, imulits, pip commands, playsound, opencv.

5.2 EXPERIMENTAL RESULTS

After doing literature survey, different techniques have been found for detecting driver drowsiness and they use different types of data as input for their algorithm. First, we have used a camera which is setup at desirable position in a car that looks for faces stream. The face of the driver will be found in the video in such a way that it should not affect the performance of accurate face detection in terms of varying lightning conditions. If face gets detected, the facial landmark detection task is applied and region of eyes and mouth is extracted. Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye-lids are down for a substantial amount of time and we are detecting the yawn. For the functionalities of the system and to make it work efficiently we have used OpenCv, dlib and Python. If the driver is drowsy then the driver is alerted with an alarm sound.

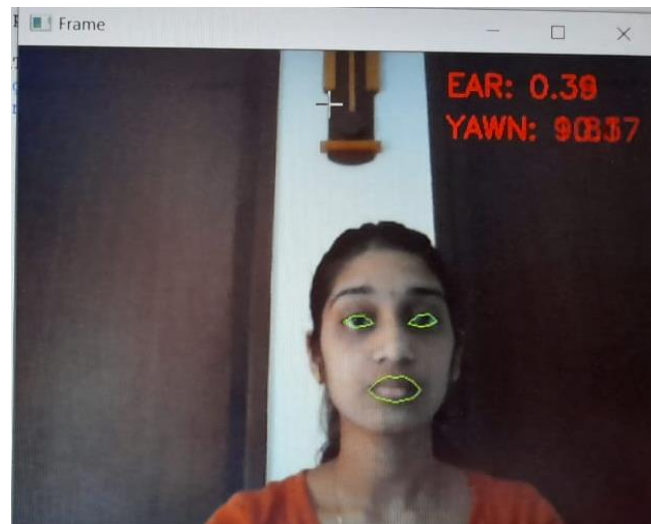


Fig 5.1 output Sample 1

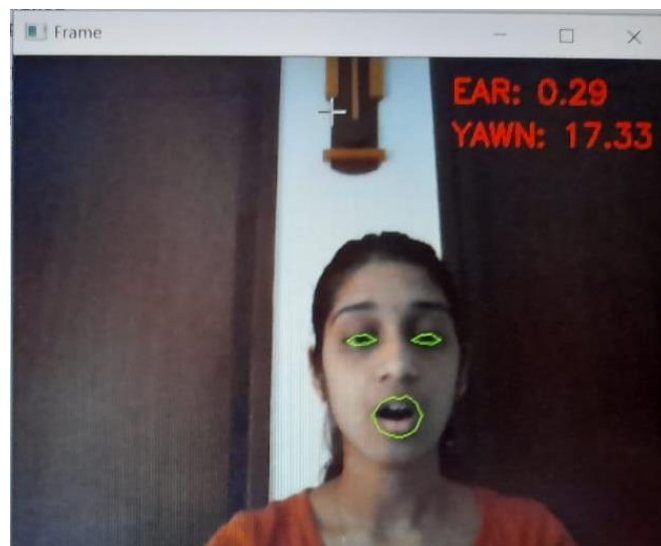


Fig 5.2 output Sample 2

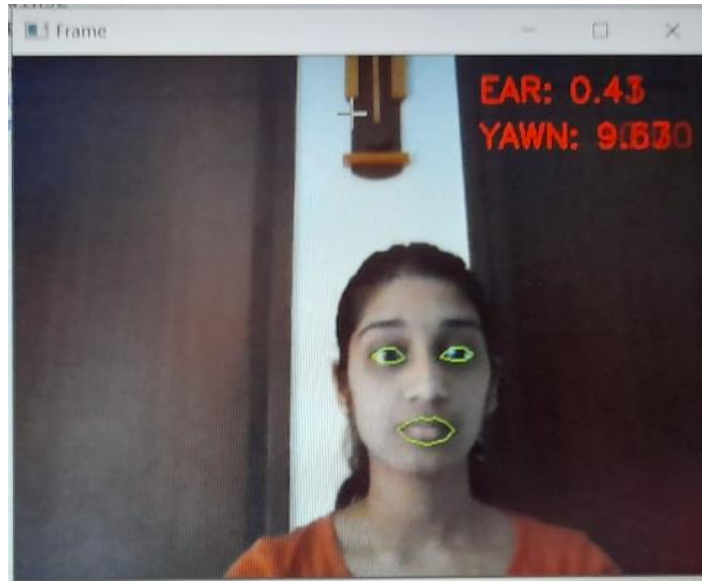


Fig 5.3:sample output 3

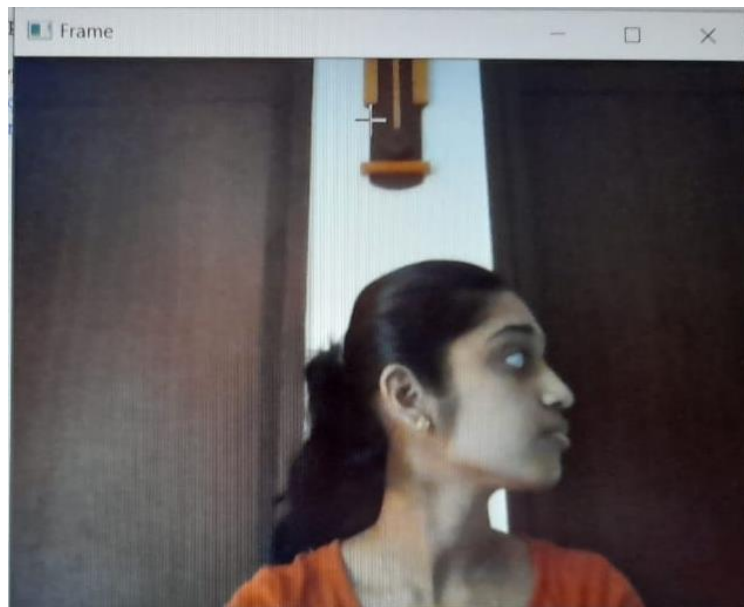


Fig 5.4 sample output 4

5.3 LIMITATION

1. Dependence on ambient light

With poor lighting conditions even though face is easily detected, sometimes the system is unable to detect the eyes. So it gives an erroneous result which must be taken care of. In real time scenario infrared backlights should be used to avoid poor lighting conditions.

2. Hardware requirements

Our system was run in a PC with a configuration of 1.6GHz and 1GB RAM Pentium dual core processor. Though the system runs fine on higher configurations, when a system has an inferior configuration, the system may not be smooth and drowsiness detection will be slow. 46 The problem was resolved by using dedicated hardware in real time applications, so there are no issues of frame buffering or slower detection.

3. Poor detection with spectacles

When the driver wears glasses the system fails to detect eyes which is the most significant drawback of our system. This issue has not yet been resolved and is a challenge for almost all eye detection systems designed so far.

4. Problem with multiple faces

If more than one face is detected by the webcam, then our system gives an erroneous result. This problem is not important as we want to detect the drowsiness of a single driver.

CHAPTER 6

CONCLUSION

6.1 CONCLUSION

Hence driver drowsiness detection is a very important part for the driver while he drives the car. So every driver should have a drowsiness detection system in his vehicle so that he can be almost safe and secure from road accidents, deaths from road accidents. A sleep of even a second can take the life of the driver. Then why to take any risk? This risk can be prevented when the driver uses the drowsiness detection system. The truck drivers, car passengers, taxi drivers, all can be aware about this system. They can manage to get help through it. Also they must be tracked on every route where they are driving to prevent any mishappening. The fatigue detection system in drivers has thus been simulated in Open CV. The real time system has been successfully created to detect the face and hence the eyes and mouth of the driver to check whether he is blinking or yawning to acquire information about his level of alertness. The alarm that has been implemented is a eSpeak software which converts text to speech to alert the driver. To realize the fatigue detection on the real roads, the head posture is also a good choice in the future work.

6.2 FUTURE ENHANCEMENT

Some drivers cover their mouth while yawning or they have different signs of sleepiness like eye closure or falling head; in this case, future work may consist of combining the detection of different fatigue signs.

When the driver wears glasses the system may not detect eyes which is the most noteworthy disadvantage of these systems. This issue has not yet been settled in near future it can be sorted out.

REFERENCES

1. O. Tuncer, L. E. Karsl Gil, “Vision based lane keeping assistance control triggered by a driver inattention monitor”, IEEE International Conference on Systems Man and Cybernetics (SMC), Istanbul, 10-13 Oct. 2010.
2. H. Graf, and E. Cosatto, “Locating Faces and Facial Parts”, in Proc. First Intl Workshop Automatic Face and Gesture Recognition, 1995.
3. T. Pilutti, A. Ulsoy, “Identification of driver state for lane-keeping tasks”, IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 29, no. 5, pp. 486-502, 1999
4. I. Craw, J. Lishman, “Automatic Extraction of Face Features”, Pattern Recognition Letters, vol. 5, pp. 183-187, 1987.
5. M. Sigari, “Driver Hypo-vigilance Detection Based on Eyelid Behaviour”, in Seventh International Conference on Advances in Pattern Recognition, 2009.
6. P. Tabrizi and R. Zoroofi, “Open or Closed Eye Analysis for Drowsiness Detection”, First Workshops on Image Processing Theory, Tools and Applications, 2008.
7. M. Omidyeganeh, A. Javadtalab and S. Shirmohammadi, “Intelligent driver drowsiness detection through fusion of yawning and eye closure”, in IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011.
8. M. Saradadevi and P. Bajaj, “Driver Fatigue Detection Using Mouth and Yawning Analysis” IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 6, pp. 183-188, June 2008.
9. T. Wang and P. Shi, “Yawning detection for determining driver drowsiness”, in Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005.
10. Zhang, Wei; Cheng, Bo; Lin, Yingzi, “Driver drowsiness recognition based on computer vision technology.” Published in: Tsinghua Science and Technology (Volume: 17, Issue: 3) Page(s):354 -362 Date of Publication: June 2012.

APPENDIX

SOURCE CODE

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
import os
import subprocess

def alarm(msg):
    global alarm_status
    global alarm_status2
    global saying
    while alarm_status:
        print('call')
        s = 'espeak "' + msg + '"'
        subprocess.call(s)
    if alarm_status2:
        print('call')
        saying = True
        s = 'espeak "' + msg + '"'
```

```

subprocess.call(s)

saying = False

def eye_aspect_ratio(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear = (A + B) / (2.0 * C)
    return ear

def final_ear(shape):
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
    return (ear, leftEye, rightEye)

def lip_distance(shape):
    top_lip = shape[50:53]
    top_lip = np.concatenate((top_lip, shape[61:64]))
    low_lip = shape[56:59]
    low_lip = np.concatenate((low_lip, shape[65:68]))
    top_mean = np.mean(top_lip, axis=0)
    low_mean = np.mean(low_lip, axis=0)
    distance = abs(top_mean[1] - low_mean[1])
    return distance

ap = argparse.ArgumentParser()

```

```

ap.add_argument("-w", "--webcam", type=int, default=0,
                help="index of webcam on system")

args = vars(ap.parse_args())

EYE_AR_THRESH = 0.3

EYE_AR_CONSEC_FRAMES = 30

YAWN_THRESH = 20

alarm_status = False

alarm_status2 = False

saying = False

COUNTER = 0

print("-> Loading the predictor and detector...")

detector = dlib.get_frontal_face_detector()

#detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml") #Faster
#but less accurate

predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

print("-> Starting Video Stream")

vs = VideoStream(src=args["webcam"]).start()

#vs= VideoStream(usePiCamera=True).start()    //For Raspberry Pi

time.sleep(1.0)

while True:

    frame =vs.read()

    frame = imutils.resize(frame, width=450)

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

    rects = detector(gray, 0)

    #rects = detector.detectMultiScale(gray, scaleFactor=1.1,
    #                                #minNeighbors=5, minSize=(30, 30),
    #                                #flags=cv2.CASCADE_SCALE_IMAGE)

```

```

for rect in rects:
    #for (x, y, w, h) in rects:
        #rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))

        shape = predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        eye = final_eye(shape)
        ear = eye[0]

        leftEye = eye [1]
        rightEye = eye[2]

        distance = lip_distance(shape)
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

        lip = shape[48:60]
        cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

        if ear < EYE_AR_THRESH:
            COUNTER += 1

            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                if alarm_status == False:
                    alarm_status = True

                    t = Thread(target=alarm, args=('wake up sir',))

                    t.daemon = True

                    t.start()

                cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        else:

```

```

COUNTER = 0

alarm_status = False

if (distance > YAWN_THRESH):

    cv2.putText(frame, "Yawn Alert", (10, 30),
                 cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    if alarm_status2 == False and saying == False:

        alarm_status2 = True

        t = Thread(target=alarm, args=('take some fresh air sir',))

        t.daemon = True

        t.start()

    else:

        alarm_status2 = False

cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

cv2.imshow("Frame", frame)

key = cv2.waitKey(1) & 0xFF

if key == ord("q"):

    break

cv2.destroyAllWindows()

vs.stop()

```