A Project Report

On

FACE MONITORING SYSTEM FOR SAFETY MEASURES

Submitted for the partial fulfilment of the requirement for the award of the Degree

of

BACHELOR OF TECHNOLOGY

In

Computer Science & Engineering

Submitted by:

TUSHIT AGARWAL (1606810342)

TAVISHI GUPTA (1606810336)

SPARSH GUPTA (1606810324)

SIDHARTH JAIN (1606810321)

Under the guidance of

Mr. Vishal Jaiswal

(ASSISTANT PROFESSOR, CSE DEPARTMENT)

Department of Computer Science and Engineering

MEERU T INSTITUTE OF ENGINEERING & TECHNOLOGY

Meerut – 250005

Approved by A.I.C.T.E

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Batch: 2016 - 2020)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Signature:
Name: Tushit Agarwal
Roll No.: 1606810342 Date:

Signature:
Name: Tavishi Gupta
Roll No.: 1606810336 Date:

Signature:
Name: Sparsh Gupta
Roll No.: 1606810324 Date:

Signature:
Name: Sidharth Jain
Roll No.: 1606810321 Date:

CERTIFICATE

*This is to certify that Project Report entitled **"FACE MONITORING SYSTEM FOR SAFETY MEASURES"** which is submitted by **TUSHIT AGARWAL, TAVISHI GUPTA, SPARSH GUPTA, SIDHARTH JAIN** in partial fulfilment of the requirement for the award of degree B. Tech. in Department of Computer Science and Engineering of Dr. A.P.J. Abdul Kalam Technical University, is a record of the candidates own work carried out by him under our supervision. The matter embodied in this thesis is original and has not been submitted for the award of any other degree.*

Head of Department	**Supervisor:**
Professor Pradeep Pant	Mr. Vishal Jaiswal
Computer Science & Engineering	Asst. Professor (CSE Dept.)

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to Assistant Professor **Mr. VISHAL JAISWAL**, Department of Computer Science and Engineering, Meerut Institute of Engineering & Technology, Meerut for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day.

We also take the opportunity to acknowledge the contribution of Professor **Pradeep Pant**, Head, Department of Computer Science and Engineering, Meerut Institute of Engineering & Technology, Meerut for his full support and assistance during the development of the project.

We also do not like to miss the opportunity to acknowledge the contribution of the Project Committee members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Signature:	Signature:
Name: Tushit Agarwal	Name: Tavishi Gupta
Roll No.: 1606810342	Roll No.: 1606810342
Date:	Date:
Signature:	Signature:
Name: Sparsh Gupta	Name: Sidharth Jain
Roll No.: 1606810324	Roll No.: 1606810321
Date:	Date:

TABLE OF CONTENTS

CHAPTER **PAGE NO.**

DECLARATION……………………………………………………………...............i
BONAFIDE CERTIFICATE……..……………………………………….…………ii
ACKNOWLEDGEMENTS…………………………………………………..……..…..iii
TABLE OF CONTENTS……………………………………………………………..iv-vi
ABSTRACT………………………….…………………………………..................vii
LIST OF PICTURES……………………………………………………………..viii

1. CHAPTER 1: INTRODUCTION…………………….……….…………………...01-03

1.1.Drowsiness…………..…...……………………………………….………………01

1.2.Driver Fatigue & Road Accidents….………………………………….…….……01

1.3.Solution Approach…………………………………………..………………...02

1.4.Objective…………………………………………………………….……………03

2. CHAPTER 2: SPECIFICATION ANALYSIS…………………………………...…04-05

2.1. Hardware Specification…….…...…………………………………….…..…..…04

2.2. Software Specification……………………………………………………………..04

2. **CHAPTER 3: PROBLEM DESCRIPTION………………………………..…..…06-09**

3.1.Overview…………………………………………..........................………...06

4. CHAPTER 4: LITERATURE OVERVIEW...…………………………..……….......10-14

4.1. Driver Drowsiness Detection…………..…..……...…………………….........10

4.1.1. Face Detection………………………………...………………………12

4.1.2. Eye State Analysis……………………….…………………………………13

4.2. Yawning Analysis……………………….………………………………….........14

5 CHAPTER 5: METHODOLOGY………………………………………..….…......15-31

5.1 Open CV……………………………………………………………………15

5.1.1 What is Open CV?..16

5.1.2 What is Computer Vision?...16

5.1.3 The origin of Open CV……………………………………………………..17

5.1.4 Open CV Structure and Content……………………………………………17

5.1.5 Why Open CV?...19

5.2.Python …………………………………………………..20

5.2.1 What is Python?...21

5.2.2 Uses of Python……………………………………………………………22

5.3. Algorithm Used………………………..…………………………………..24

5.3.1 Principal Component Analysis (PCA)……………………………….25

5.3.2 Image Acquisition…………………………………………………...…26

5.3.3 Dividing into Frames……………………………………………..……26

5.3.4 Face Detection………………………………………………………….26

5.3.5 Set Image Region of Interest…………………………………………..…28

5.3.6 Recognition of Face Region………………………………..……..……….28

5.3.7 Eye Detection……………………………………………………………..29

5.3.8 Digital Image Processing…………………………………………………29

6. CHAPTER 6: CODING……………………………………………...…………31-51

6.1. Libraries Used……..…………………..…………..…………………..……....31

6.2. GUI Window………………………………………………………………...…32

6.3. Main Code……………………………………………………………………36

7. CHAPTER 7: RESULT………………………...……………………………………52-63

7.1.Snapshots……………………………………………………………………..52

8. CHAPTER 7: CONCLUSION …………………………….…………………….....62

9. CHAPTER 8: FUTURE SCOPE………………………………………………………63

REFERENCES…………………………………………………………………………….64

ABSTRACT

A Face Monitoring System has been developed keeping in mind the safety measures for the drivers while driving, using a non-intrusive machine vision - based concepts. The system uses a small monochrome security camera that points directly towards the driver's face and monitors the driver's eyes in order to detect fatigue. In such a case when fatigue is detected, a warning signal is issued to alert the driver. This report describes how to find the eyes, and also how to determine if the eyes are open or closed. The algorithm developed is unique to any currently published papers, which was a primary objective of the project. The system deals with using information obtained for the binary version of the image to find the edges of the face, which narrows the area of where the eyes may exist. Once the face area is found, the eyes are found by computing the horizontal averages in the area. Taking into account the knowledge that eye regions in the face present great intensity changes, the eyes are located by finding the significant intensity changes in the face. Once the eyes are located, measuring the distances between the intensity changes in the eye area determine whether the eyes are open or closed. A large distance corresponds to eye closure. If the eyes are found closed for 20 consecutive frames, the system draws the conclusion that the driver is falling asleep and issues a warning signal. The system is also able to detect when the eyes cannot be found, and works under reasonable lighting conditions

LIST OF FIGURES

Figure No.	Name of Figure	Page No.
Figure 1	Open CV	15
Figure 2	Structure and content of Open CV	18
Figure 3	Image of Python	20
Figure 4	Algorithm used	23-24
Figure 5	Image of Principal Component Analysis	25
Figure 6	GUI Window	53
Figure 7	Image when user click on open camera button	54
Figure 8	Image when user click on stop camera button	55
Figure 9	Image when user click on start detection button	56
Figure 10	Image when user did yawning detection	57
Figure 11	Image when user is not in frame	58
Figure 12	Image when user did mobile detection	59
Figure 13	Image when user is distracted with some other object	60
Figure 14	Image when user click on stop detection button	61

CHAPTER-1

INTRODUCTION

1.1.Drowsiness

Drowsy means sleepy and having low energy . Drowsiness is position of near to sleep, a strong desire to sleep . Drowsiness is defined as a decreased level of awareness portrayed by sleepiness and trouble in staying alarm but the person awakes with simple excitement by stimuli .Drowsiness, also called excess sleepiness . It might be caused by an absence of rest, medicine, substance misuse, or a cerebral issue. It is mostly the result of fatigue which can be both mental and physical. Physical fatigue, or muscle weariness, is the temporary physical failure of a muscle to perform ideally. Mental fatigue is a temporary failure to keep up ideal psychological execution. The onset of mental exhaustion amid any intellectual action is progressive, and relies on an individual's psychological capacity, furthermore upon different elements, for example, lack of sleep and general well-being. Mental exhaustion has additionally been appeared to diminish physical performance. It can show as sleepiness, dormancy, or coordinated consideration weakness. Drowsiness may lead to forgetfulness or falling asleep at inappropriate times .

In the past years according to available data driver sleepiness has gotten to be one of the real reasons for street mishaps prompting demise and extreme physical injuries and loss of economy. A driver who falls asleep is in an edge of losing control over the vehicle prompting crash with other vehicle or stationary bodies. Keeping in mind to stop or reduce the number of accidents to a great extent the condition of sleepiness of the driver should be observed continuously.

1.2. Driver Fatigue & Road Accidents

Driver fatigue sometimes results in road accidents every year. It is not easy to estimate the exact amount of sleep related accidents but research presents that driver fatigue may be a contributing reason in upto 20% in road accidents. These types of accidents are about 50% more expected to result in death or serious hurt. They happen mainly at higher speed impacts.And the driver who has fallen asleep cannot brake. Drowsiness reduces response time which is a serious element of secure driving. It also reduces alertness, vigilance, and concentration so that the capacity to perform attention-based activities i.e. driving is impaired. The speed at which information is processed is also reduced by drowsiness. The quality of decision-making may also be affected. It is clear that drivers are awake when they are feeling sleepy, and so make a conscious decision about whether to continue driving or to stop for a rest. It may be that those who persist in driving underestimate the risk of actually falling asleep while driving. Or it may be that some likely to happen on long journeys on monotonous roads, such as motorways, between 2pm and 4pm especially after eating or taking an alcoholic drink, between 2am and6am, after having less sleep than normal, after drinking alcohol, driver takes medicines that cause drowsiness and after long working hours or on journeys home after long shifts, especially night shifts.

1.3.Solution Approach

Two kind of monitoring systems are named as vehicle oriented system and driver oriented system.

1.3.1Vehicle Oriented System

Drowsiness is detected by analysing the driver's behaving using information measured by sensors located in the vehicle, such as its position on the road, steering wheel movements, pressure on the driving pedals or the variability of the vehicle's speed. The main disadvantages of this approach is that driving behaviour may be very different from driver to driver. This makes it difficult to construct a ―correct driving‖ model that can be used to detect variations in driving behaviour. This model has to be learnt for each driver.

1.3.2Driver Oriented System

There are two types of driver oriented system which are named as- **(A)**

Instructive monitoring system based on biological indicators

Drowsiness is detected using physiological information. It is measured by sensors located on or around the driver. The physiological information is eye activity, cerebral activity, Yawns, facial expressions, or gaze direction. These systems are more reliable because physiological drowsiness signs are known and are similar to one driver or another driver. There is one drawback with placements of sensors on the driver's body. They may not bother driver while driving. Another drawback is that measurements may be difficult because the driver is constantly moving. This system is bulky to implement.

(B) Non-instructive monitoring system based on face analysis

The human face is dynamic and has a high degree of variability. Face detection is considered to be a difficult problem in computer vision research. As one of the most important features of the human face, human eyes play an important role in face recognition and facial expression analysis. In actuality, the eyes can be considered salient and relatively stable feature on the face in comparison with other facial features. Therefore, when detecting facial features, it is advantageous to detect eyes before the detection of other facial features. The position of other facial features can be estimated using the eye position. In addition, the size, the location and the image-plane rotation of face in the image can be normalized by only the position of both eyes.

1.4.Objective

Driver drowsiness detection is a car safety technology which helps to save the life of the driver by preventing accidents when the driver is getting drowsy. The main objective is to first design a system to detect driver's drowsiness by continuously monitoring retina of the eye.So that it can produce a alarm when driver is not awake. The system works in spite of driver wearing spectacles(tranceparent) and in various lighting conditions. To alert the drive on the detection of drowsiness by using buzzer or alarm. Traffic management can be maintained by reducing the accidents and cause less loss of life and property.

CHAPTER-2

SPECIFICATION ANALYSIS

2.1. Hardware Specification

System	**:** 2 Ghz or faster processor or Macbook
Hard Disk	**:** 16 GB for 32-bit OS 20 GB for 64-bit
Display	**:** 800x600
Ram	**:** 4 GB or more for both 32-bit & 64-bit

2.2. Software Specification

Open Source Computer Vision Library (Open CV) focuses on real-time vision for implementing projects and by providing computer vision and machine learning infrastructure. The library is written in C and C++ and runs under Linux, Windows and Mac OS X. Open CV was designed for computational efficiency and with a strong focus on realtime applications. The Open CV library contains over 500 functions that span many areas in vision, including factory product inspection, medical imaging, security, user interface, camera calibration, stereo vision, and robotics.

The basic structure of Open CV is as follows: The CV component contains the basic image processing and higher-level computer vision algorithms; ML is the machine learning library, which includes many statistical classifier and clustering tools. High GUI contains I/O routines and functions for storing and loading video and images, and CX Core contains the basic data structures and content.

Face detection in Open CV is done via a HaarClassifier, an object detection application based on a clever use of boosting. The Open CV distribution comes with a trained frontal face detector that works remarkably well. The HaarCassifier is a tree-based technique in Open CV which builds a boosted rejection cascade. The Viola-Jones detector implemented by Open CV is also referred to as the ”HaarClassifier” because it uses Haar features or, more precisely, Haar-like wavelets that consist of adding and subtracting rectangular image regions before thresholding the result.

The HaarClassifier that is included in Open CV is a supervised classifier. In this case we typically present histogram- and size-equalized image patches to the classifier, which are then labeled as containing (or not containing) the object of interest, which for this classifier is most commonly a face.

So, in brief, take multiple positive samples, that is, objects of interest, and negative samples, that is, images that do not contain those objects. Different features are extracted from samples and distinctive features are compressed into the statistical model parameters. It is easy to make an adjustment by adding new positive or negative samples.

CHAPTER-3

PROBLEM DESCRIPTION

3.1.Overview

Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,000 deaths and 72,000 injuries can be attributed to fatigue related crashes. There has been ample development in the field of safety in case of accidents in the form of air bags, etc. However, the development of technologies for detecting and preventing drowsiness of the driver is a major challenge in the field of accident avoidance systems. Because of the hazard that drowsiness presents on the road, methods need to be developed for counteracting its aff ects. The aim of this project is to develop a prototype drowsiness detection system. This system will primarily measure and record the real time features of the driver or the driving pattern and continuously evaluate them on the basis of the levels predetermined to indicate drowsiness. The driver may show signs of fatigue in numerous ways. The project surveys diff erent methods of eye detection and checking whether the detected eyes are open or closed. On the basis of the number of times the eyes are found to be closed, it can be determined whether the person is drowsy or not. Another method of drowsiness detection is the evaluation of the features of the mouth. If the driver yawns, showing signs of drowsiness, it can be used to trigger the alarm system.

Drowsiness detection in the non intrusive form , may be done efficiently by increasing the number of parameters on which the driver is being monitored. This will lead to a complex algorithm which can be easily flexible to detect drowsiness. Such a driver drowsiness detection system has tremendous scope in the car industry. There are millions of cars being manufactured every year and it is a universally accepted fact that drowsiness among drivers is an important accident factor. Hence, car manufacturing companies are the biggest market for such a system. Mercedes Benz has introduced a fatigue detection system in certain models of high end cars. However, the proposed prototype that this project aims at creating shall be far more inexpensive as compared to that model, thus creating a bigger market for this product. Indian roads are prone to fatigue related accidents essentially among truck drivers, long distance bus drivers and BPO employees during the late hours. Introduction of such a system in long distance vehicles and in public transport would reduce the number of accidents.

Several studies have investigated the relationship between driver fatigue and crash risk and have attempted to quantify the risk increase.

The data collected in the 100 Car Naturalistic Driving Study shows that driving while fatigued increases a driver’s risk of involvement in a crash or near-crash by nearly four times. Studies of professional drivers (bus, lorry, truck) show that it takes around 9 or 10 hours of driving, or 11 hours of work, before crash risk starts to rise. Hamelin found that after 11 hours of work span the crash risk doubles. The eff ect of task duration is practically always entangled with the effects of the time of day and sometimes also with the length of time awake and previous lack of sleep. The duration of a trip may be of lesser importance compared to these other factors - many fatigue-related accidents occur after driving for only a few hours. Short trips can also end up in fatigue-related crashes because time of day and long and irregular working hours are stronger predictors of fatigue than time spent driving.
The association of non-medical (lifestyle) determinants of fatigue with crash has not been the subject of thorough research. There is still a lack of knowledge concerning the contribution of increasing total hours of work, and shift schedules to driver fatigue. Whereas research into fatigue and sleep in truck drivers has led to awareness of these problems and some modification of work conditions, occupationally induced fatigue in potentially much larger numbers of commuters has received little attention. The Federal Motor Carrier Safety Administration (FMCSA), the trucking industry, highway safety advocates, and transportation researchers have all identified driver drowsiness as a high priority commercial vehicle safety issue. Drowsiness affects mental alertness, decreasing an individual’s ability to operate a vehicle safely and increasing the risk of human error that could lead to fatalities and injuries. Furthermore, it has been shown to slow reaction time, decreases awareness, and impairs judgment. Long hours behind

the wheel in monotonous driving environments make truck drivers particularly prone to drowsy-driving crashes.

Successfully addressing the issue of driver drowsiness in the commercial motor vehicle industry is a formidable and multi-faceted challenge. Operational requirements are diverse, and factors such as work schedules, duty times, rest periods, recovery opportunities, and response to customer needs can vary widely. In addition, the interaction of the principal physiological factors that underlie the formation of sleepiness, namely the homeostatic drive for sleep and circadian rhythms, are complex. While these challenges preclude a single, simple solution to the problem, there is reason to believe that driver drowsiness can nevertheless be effectively managed, thus resulting in a significant reduction in related risk and improved safety. Addressing the need for a reduction in crashes related to driver drowsiness in transportation will require some innovative concepts and evolving methodologies. Invehicle technological approaches, both available and emerging, have great potential as relevant and eff ective tools to address fatigue. Within any comprehensive and eff ective fatigue management program, an on-board device that monitors driver state in real time may have real value as safety net. Sleepy drivers exhibit certain observable behaviour, including eye gaze, eyelid movement, pupil movement, head movement, and facial expression. Non-invasive techniques are currently being employed to assess a driver's alertness level through the visual observation of his/her physical condition using a remote camera and state-of-the-art technologies in computer vision. Recent progress in machine vision research and advances in computer hardware technologies have made it possible to measure head pose, eye gaze, and eyelid movement accurately and in real time using video cameras. The alertness monitoring technologies monitor - usually on-line and in real time - biobehavioural aspects of the operator; for example, eye gaze, eye closure, pupil occlusion, head position and movement, brain wave activity, and heart rate. To be practical and useful as driver warning systems, these devices must acquire, interpret, and feed-back information to the operator in real world driving environments. As such, there exists a need and, thus, ongoing efforts are underway, to validate operator-based, on-board fatigue monitoring technologies in a real-world naturalistic driving environment.

In view of the need for non intrusive relatively inexpensive modules for fatigue detection in vehicles, this project has been formulated to monitor the driver’s actions and reactions, check for fatigue and on detection of fatigue or drowsiness, stimulate the appropriate plan of action, which could be an alarm or deceleration of the vehicle, etc.

CHAPTER-4

LITERATURE OVERVIEW

4.1 Drowsiness Detection

The study states that the reason for a mishap can be categorized as one of the accompanying primary classes: (1) human, (2) vehicular, and (3) surrounding factor. The driver's error represented 91% of the accidents. The other two classes of causative elements were referred to as 4% for the type of vehicle used and 5% for surrounding factors. Several measures are available for the measurement of drowsiness which includes the following:

1. Vehicle based measures.
2. Physiological measures.
3. Behavioral measures

1. **Vehicle based measures**.

Vehicle-based measures survey path position, which monitors the vehicle's position as it identifies with path markings, to determine driver weakness, and accumulate steering wheel movement information to characterize the fatigue from low level to high level. In many research project, researchers have used this method to detect fatigue, highlighting the continuous nature of this non-intrusive and cost-effective monitoring technique.

This is done by:

1. Sudden deviation of vehicle from lane position.
2. Sudden movement of steering wheels.
3. Pressure on acceleration paddles.

For each measures threshold values are decided which when crossed indicated that driver is drowsy.

2. **Physiological measures**.

Physiological measures are the objective measures of the physical changes that occur in our body because of fatigue. These physiological changes can be simply measure by their respective instruments as follows: ECG (electro cardiogram) EMG (electromyogram) EOG (electroculogram) EEG (electroencephalogram)

Monitoring Heart Rate: An ECG sensor can be installed in the steering wheel of a car to monitor a driver's pulse, which gives a sign of the driver's level of fatigue indirectly giving the state of drowsiness. Additionally the ECG sensor can be introduced in the back of the seat.

Monitoring Brain Waves: Special caps embedded with electrodes measures the brain waves to identify fatigue in drivers and report results in real time. Then each brain waves can be classified accordingly to identify drowsiness.

Monitoring muscle fatigue: As muscle fatigue is directly related to drowsiness. We know during fatigue the pressure on the steering wheel reduces and response of several muscle drastically reduces hence it can be measured by installation of pressure sensors at steering wheel or by measuring the muscle response with applied stimuli to detect the fatigue.

Monitoring eye movements: Invasive measurement of eye movement and eye closure can be done by using electro oscillogram but it will be very uncomfortable for the driver to deal with.

Though this method gives the most accurate results regarding drowsiness. But it requires placement of several electrodes to be placed on head, chest and face which is not at all a convenient and annoying for a driver. Also, they need to be very carefully placed on respective places for perfect result.

Behavioural measures. Certain behavioural changes take place during drowsing like

1. Yawning

2. Amount of eye closure

3. Eye blinking

4. Head position

Utilizing a web camera introduced inside the automobile we can get the picture of the driver. Despite the fact that the camera creates a video clip, we have to apply the developed algorithm on each edge of the video stream. This paper is only focused on the applying the proposed mechanism only on single frame. The used camera is a low cost web camera with a frame rate of 30 fps .

We are dealing with real time situation where video is recorded and has to be processed. But the processing or application of algorithm can be done only on an image. Hence the captured video has to be divided into frames for analysing. In this stage we detect the region containing the face of the driver. A specified algorithm is for detection of face in every frame. By face detection we means that locating the face in a frame or in other words finding location of facial characters through a type of technology with the use of computer. The frame may be any random frame. Only facial related structures or features are detected and all others types of objects like buildings, tree, bodies are ignored.

After successful detection of face eye needs to be detected for further processing.

4.1.1 Face Detection

We know that face is also a type of object. So we can consider detection of face as a particular case of object detection. In this type of object type of class detection, we try to know where the objects in the interest image are located and what is their size which may belongs to a particular class. The work of algorithm that is made for face detection is mostly concentrated on finding the front side of the face. But the algorithm that are developed recently focus on more general cases. For our case it may be face in the tilted position or any other portion of the faces and also it finds the possibility of multiple faces. Which means the rotation axis with respect to the present observer from the reference of face in a particular. Or even if there is vertical rotation plane then also it is able to solve the purpose. In new type of algorithm, it is considered that the picture or video is a variable which means that different condition in them like hue contrast may change its variance. The amount of light may also affect. Also the position of the input may vary the output. Many calculations actualize the face-detection assignment as a two way pattern-differentiation task. It means the contextual features present in the interest image is repeatedly change into features and this results in preparing the respective classifier on the reference faces which decides if the specified area is a face or any other objects. If we obtain a positive response for the detecting a face then the process goes for next stage continuation otherwise the algorithm is designed in such manner to go for capturing of image till any hint of face is found. The main algorithm used for this process is Viola Jones algorithm. For getting particular output the utilization of cascade part of open CV is made. Cascade file of Open CV contains 24 stages and has got 2913 weak classifiers. Its window starts with size of 24 x 24 pixels. Set up for the starting scale has to be made 1.0 and the step size of each scale was set to 1.1 and the position step size Δ was set to 1.0. The total number of scales used is 32 resulting in a total of more than 1.8 million possible detection window which is huge. Training of cascade was done by Open Cv hence it is easy to use.

4.1.2. Eye State Analysis

Poor contrast of eyes generally creates a lot of problems in its detection. After successful detection of face eye needs to be detected for further processing. In our method eye is the decision parameter for finding the state of driver. Though detection of eye does not look complex but the actual process is quite hectic. In this case it performs the detection of eye in the specified region with the use of feature detection. Generally, Eigen approach is used for this process. It is a time taking process. When eye detection is done then the result is matched with the reference or threshold value for deciding the state of the driver. Eye detection is divided into two categories: eye contour detection and eye position detection. Basically, eyes are detected based on the assumption that they are darker than other part of the face. Hence Haar Features of similar type can be moved throughout the upper part of the face to match with the feature of eye leading to location of eye. We consider as potential eye areas, the nonskin locales inside face district. Clearly, eyes ought to be inside a face area and eyes are not distinguished as skin by the skin identifier. In this way, we need to discover eye-simple sets among a decreased number of potential eye regions. In recent years several eye detection methods have been developed. Deformable template is one of the popular methods in identifying the human eye. In this method, a model of eye is designed first and then eye position is obtained by recursive method. But this method strongly depends on initial position of the eye which should be near the actual position of eye. In the template matching aspect, the proposed algorithm is based on eigen features and neural networks for the extraction of eyes using rectangular fitting from grey-level face images. This method does not need a large set of training images in its advantage and does by eigen features and sliding window. But this algorithm fails if the user uses glasses or having beard. We know that using Haar features in AdaBoost results in increasing computational efficiency and accuracy than other methods for face detection. But Haar feature has a limitation i.e. discriminant capability. Although the Haar features vary with different patterns, sizes and positions, they can only represent the regular rectangular shapes. But for our case of eye detection eye and iris is of round shape. Hence eyes can be represented by learning discriminate features to characterize eye patterns. So an approach towards probabilistic classifier to separate eyes and non-eyes are much better option for better accuracy and for robustness.

4.2 Yawning Analysis

Yawning can be detected from the extent of openness of the mouth. If ratio of mouth height to width is above 0.5, the user is yawning and if more than 6 frames detect yawning, the system points fatigue.

Problems :-

• Performance of the IR illuminator is good at night, but reduces on bright days.

• Yawning cannot be detected when the driver puts a hand on his face.

CHAPTER-5

METHODOLOGY

5.1. Open CV

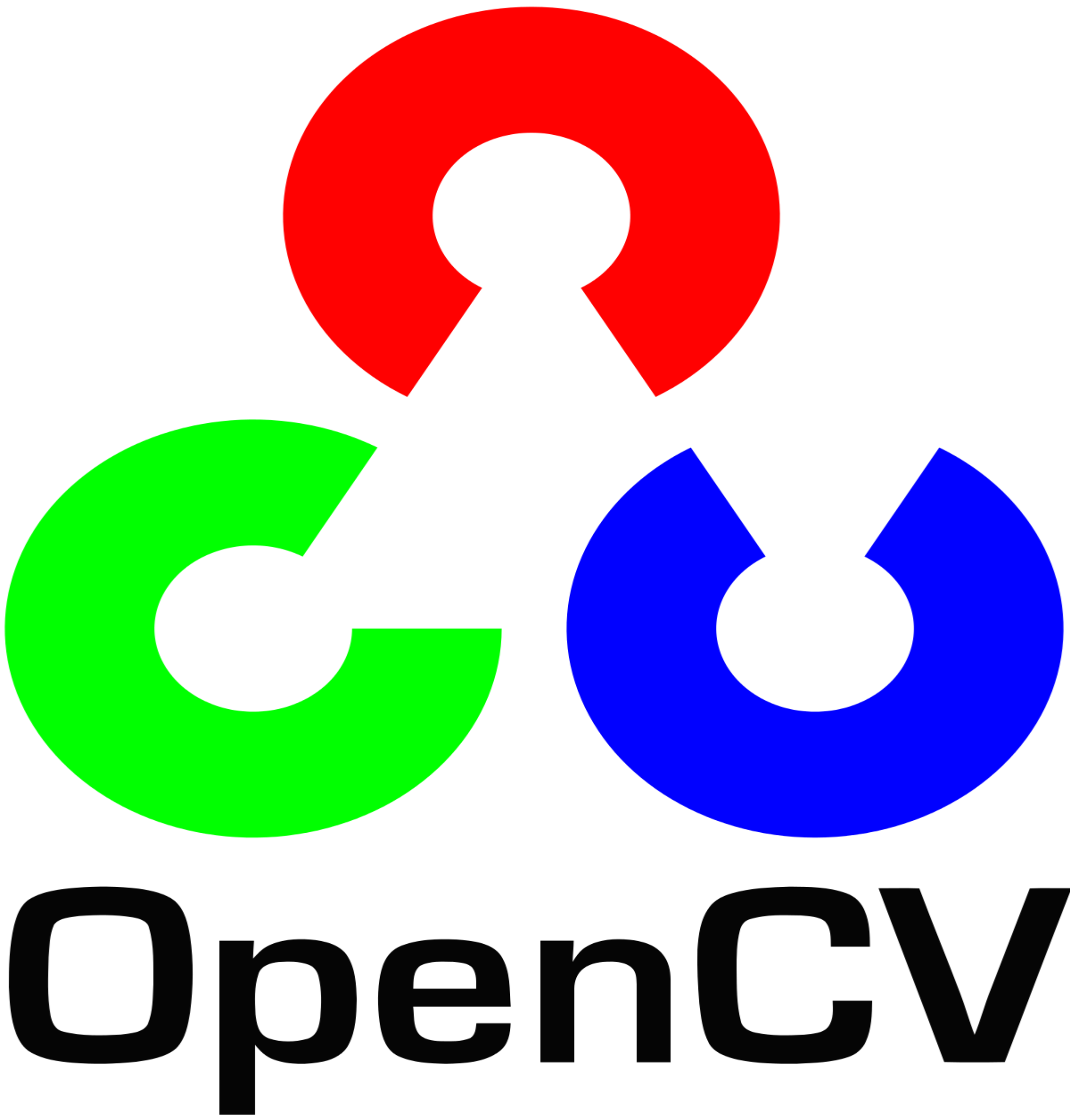


Fig 1: Open CV

5.1.1. What Is Open CV?

Open CV is an open source computer vision library available. Open CV was designed for computational efficiency and having a high focus on real-time image detection. Open CV is coded with optimized C and can take work with multicore processors. If we desire more automatic optimization using Intel architectures [Intel], you can buy Intel’s Integrated Performance Primitives (IPP) libraries. These consist of low-level routines in various algorithmic areas which are optimized. Open CV automatically uses the IPP library, at runtime if that library is installed.

One of Open CVs goals is to provide a simple-to-use computer vision infrastructure which helps people to build highly sophisticated vision applications fast. The Open CV library, containing over 500 functions, spans many areas in vision. Because computer vision and machine learning oft en goes hand-in-hand, Open CV also has a complete, general-purpose, Machine Learning Library (MLL). This sub library is focused on statistical pattern recognition and clustering. The MLL is very useful for the vision functions that are the basis of Open CV’s usefulness, but is general enough to be used for any machine learning problem.

5.1.2 What is Computer Vision?

Computer vision is the transforming of data from a still, or video camera into either a representation or a new decision. All such transformations are performed to achieve a particular goal. Computer vision is concerned with the theory behind artificial systems that extract information from images. A computer obtains a grid of numbers from a camera or from the disk, and that’s that. Usually, there is no built in pattern recognition or automatic control of focus and aperture, no cross-associations with years of experience. For the most part, vision systems are still fairly naïve.

5.1.3 The Origin of Open CV

Open CV came out of an Intel Research initiative meant to advance CPU-intensive applications. Toward this end, Intel launched various projects that included real-time ray tracing and also 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. One of the programmers working for Intel at the time was visiting universities. He noticed that a few top university groups, like the MIT Media Lab, used to have well-developed as well as internally open computer vision infrastructures—code that was passed from one student to another and which gave each subsequent student a valuable foundation while developing his own vision application. Instead of having to reinvent the basic functions from beginning, a new student may start by adding to that which came before.

5.1.4 Open CV Structure and Content

Open CV can be broadly structured into five primary components, four of which are shown in the figure. The CV component contains mainly the basic image processing and higherlevel computer vision algorithms; MLL the machine learning library includes many statistical classifiers as well as clustering tools. High GUI component contains I/O routines with functions for storing, loading video & images, while CX Core contains all the basic data structures and content. Open CV was designed to be portable. It was originally written to compile across Borland C++, MSVC++, and the Intel compilers. This meant that the C and C++ code had to be fairly standard in order to make cross-platform support easier. Figure shows the platforms on which Open CV is known to run. Support for32-bitIntel architecture (IA32) on Windows is the most mature, followed by Linux on the same architecture. Mac OS X portability became a priority only after Apple started using Intel processors. (The OS X port isn't as mature as the Windows or Linux versions, but this is changing rapidly.) These are followed by64-bitsupport on extended memory (EM64T) and the64-bitIntel architecture (IA64). The least mature portability is on Sun hardware and other operating systems.

If an architecture or OS doesn't appear in Figure , doesn't mean there are no Open CV ports to it. Open CV has been ported to almost every commercial system, from PowerPC Macs to robotic dogs. Open CV runs well on AMD's line of processors, and even the further optimizations available in IPP will take advantage of multimedia extensions (MMX) in AMD processors that incorporate this technology.

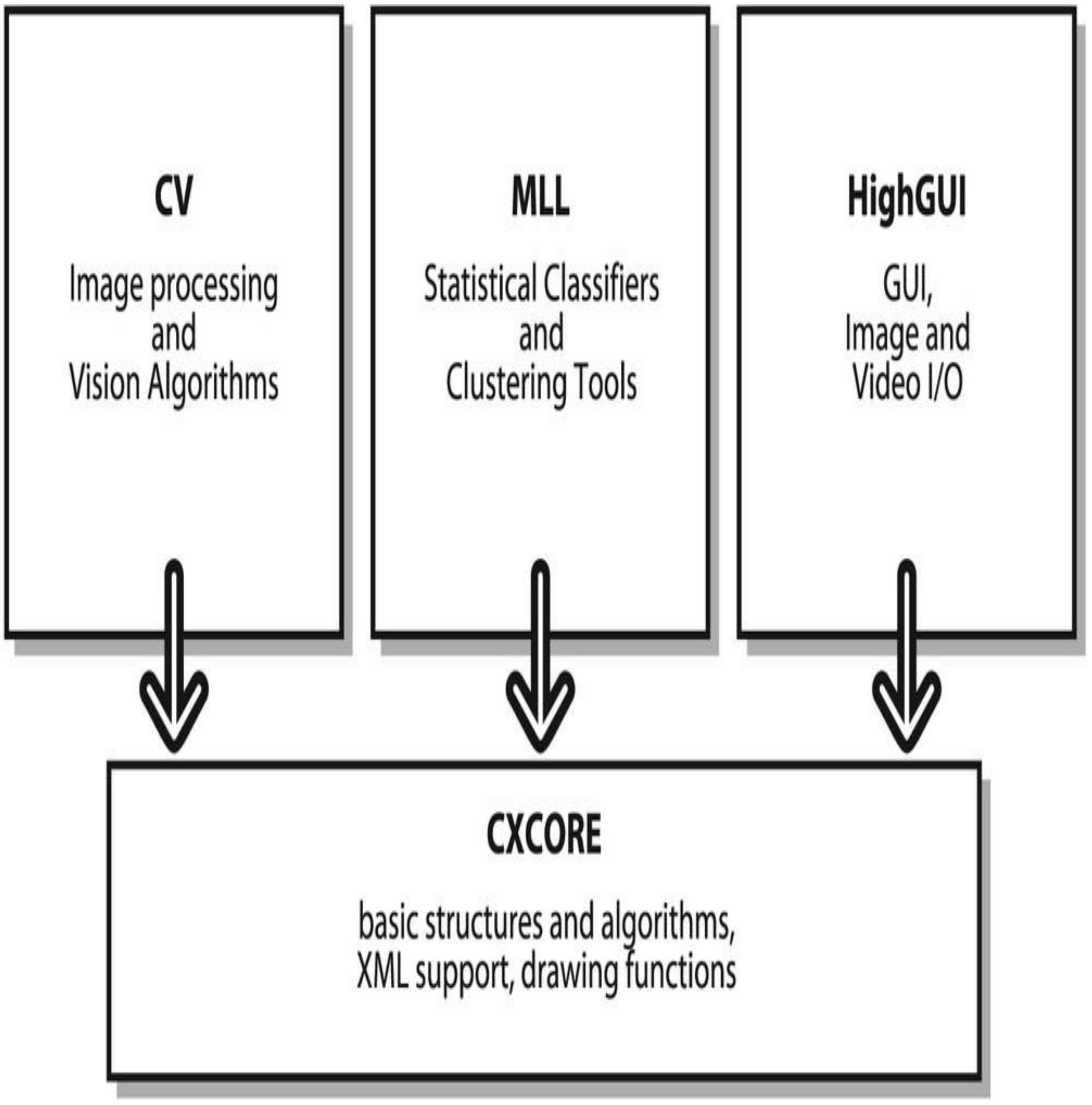

CV
Image processing and Vision Algorithms
MLL
Statistical Classifiers and Clustering Tools
HighGUI
GUI, Image and Video I/O
CXCORE
basic structures and algorithms, XML support, drawing functions

Fig 2:-Structure and Content of Open CV

5.1.5 Why Open CV?

Specific

Open CV was designed for image processing. Every function and data structure has been designed with an Image Processing application in mind. Meanwhile, Matlab, is quite generic. You can get almost everything in the world by means of toolboxes. It may be financial toolboxes or specialized DNA toolboxes.

Speedy

Matlab is just way too slow. Matlab itself was built upon Java. Also Java was built upon C. So when we run a Matlab program, our computer gets busy trying to interpret and compile all that complicated Matlab code. Then it is turned into Java, and finally executes the code.

If we use C/C++, we don’t waste all that time. We directly provide machine language code to the computer, and it gets executed. So ultimately we get more image processing, and not more interpreting.

After doing some real time image processing with both Matlab and Open CV, we usually got very low speeds, a maximum of about 4-5 frames being processed per second with Matlab. With Open CV however, we get actual real time processing at around 30 frames being processed per second.

Sure we pay the price for speed – a more cryptic language to deal with, but it’s definitely worth it. We can do a lot more, like perform some really complex mathematics on images using C and still get away with good enough speeds for your application.

Efficient

Matlab uses just way too much system resources. With Open CV, we can get away with as little as 10mb RAM for a real-time application. Although with today’s computers, the RAM factor isn’t a big thing to be worried about. However, our drowsiness detection system is to be used inside a car in a way that is non-intrusive and small; so a low processing requirement is vital.

Thus we can see how Open CV is a better choice than Matlab for a real-time drowsiness detection system.

5.2 Python

Fig 3: Image of Python

5.2.1 What is Python?

Python is an object-oriented, high-level programming language with integrated dynamic semantics primarily for web and app development. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python was developed by Guido van Rossum, and it is free software. Python is also free in other important ways, for example you are free to copy it as many times as you like, and free to study the source code, and make changes to it. There is a world wide movement behind the idea of free software, initiated in 1983 by Richard Stallman.

Python is relatively simple, so it's easy to learn since it requires a unique syntax that focuses on readability. It bears some similarities to Fortran, one of the earliest programming languages, but it is much more powerful than Fortran. Developers can read and translate Python code much easier than other languages. It is an interpreted language, which means that the written code is not actually translated to a computer-readable format at runtime. Whereas, most programming languages do this conversion before the program is even run. This type of language is also referred to as a "scripting language" because it was initially meant to be used for trivial projects.

Python allows you to use variables without declaring them, and it relies on indentation as a control structure. You are not forced to define classes in Python unlike java but you are free to do so when convenient. In turn, this reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers.

The document focuses on learning python for the purpose of doing mathematical calculations. Hence , Python is a good choice for mathematical calculations, since we can write code quickly, test is easy and its syntax is similar to the way mathematical ideas are expressed in the mathematical ideas. Additionally, Python supports the use of modules and packages, which means that programs can be designed in a modular style and code can be reused across a variety of projects. Once you've developed a module or package you need, it can be scaled for use in other projects, and it's easy to import or export these modules.

One of the most promising benefits of Python is that both the standard library and the interpreter are available free of charge, in both binary and source form. There is no exclusivity either, as Python and all the necessary tools are available on all major platforms. Therefore, it is an enticing option for developers who don't want to worry about paying high development costs.

If this description of Python over your head, don't worry. You'll understand it soon enough. What you need to take away from this section is that Python is a programming language used to develop software on the web and in app form, including mobile. It's relatively easy to learn, and the necessary tools are available to all free of charge.

5.2.2. Uses of Python

Simple

Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English. The pseudo-code nature of python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

Easy to Learn

As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

Free and Open Source

Python is an example of FLOSS (Free Libre and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs . FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good. It has been created and is constantly improved by a community who just want to see a better python.

High-Level Language

When you write programs in Python, you need to bother about the low-level details such as managing the memory used by your program, etc.

Object Oriented

Python supports procedure-oriented programming as well as object-oriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In object-oriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP (Object Oriented Programming), especially when compared to big languages like C++ or Java.

Extensive Libraries

The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI
(graphical user interfaces), and other system-dependent stuff.

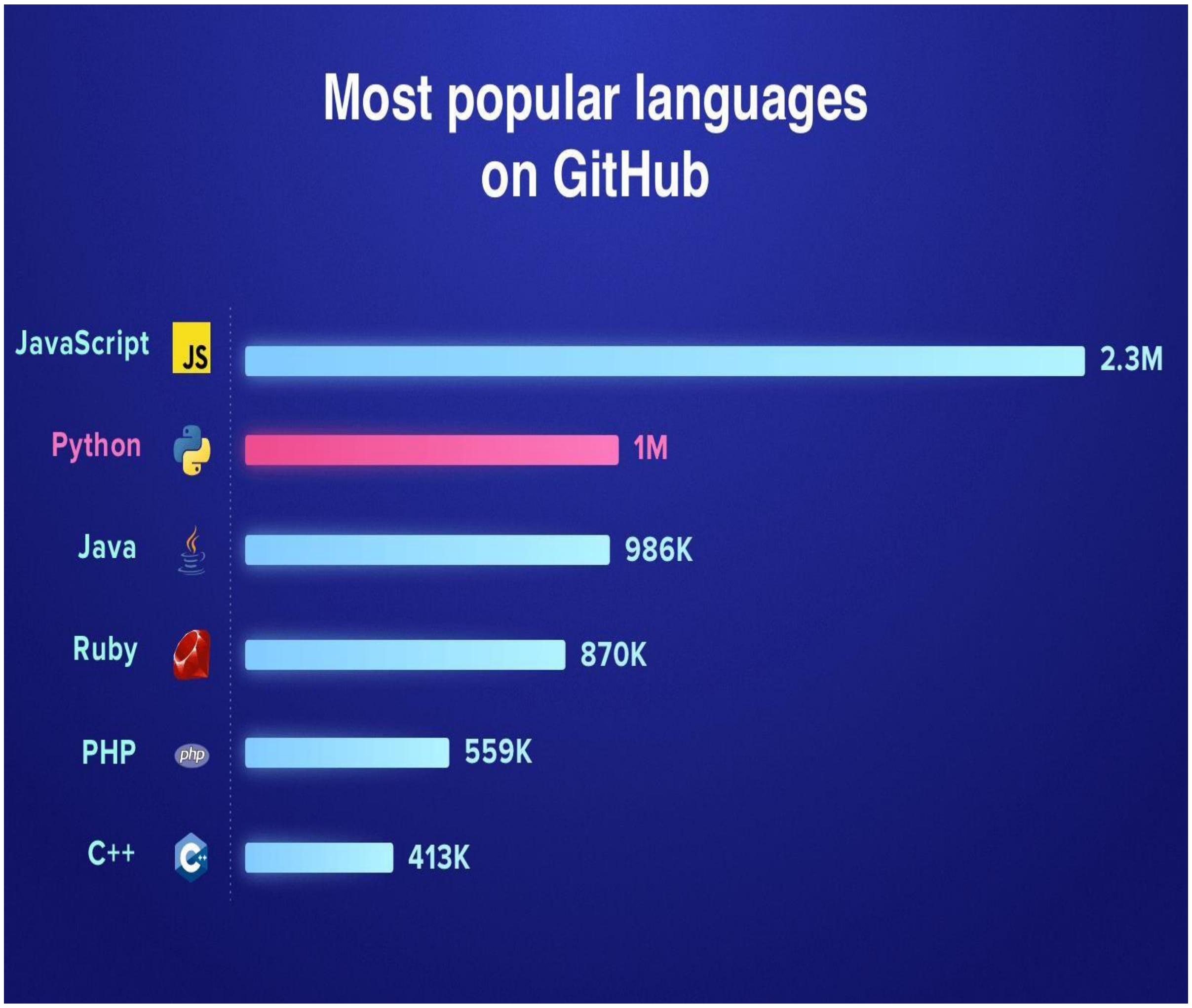
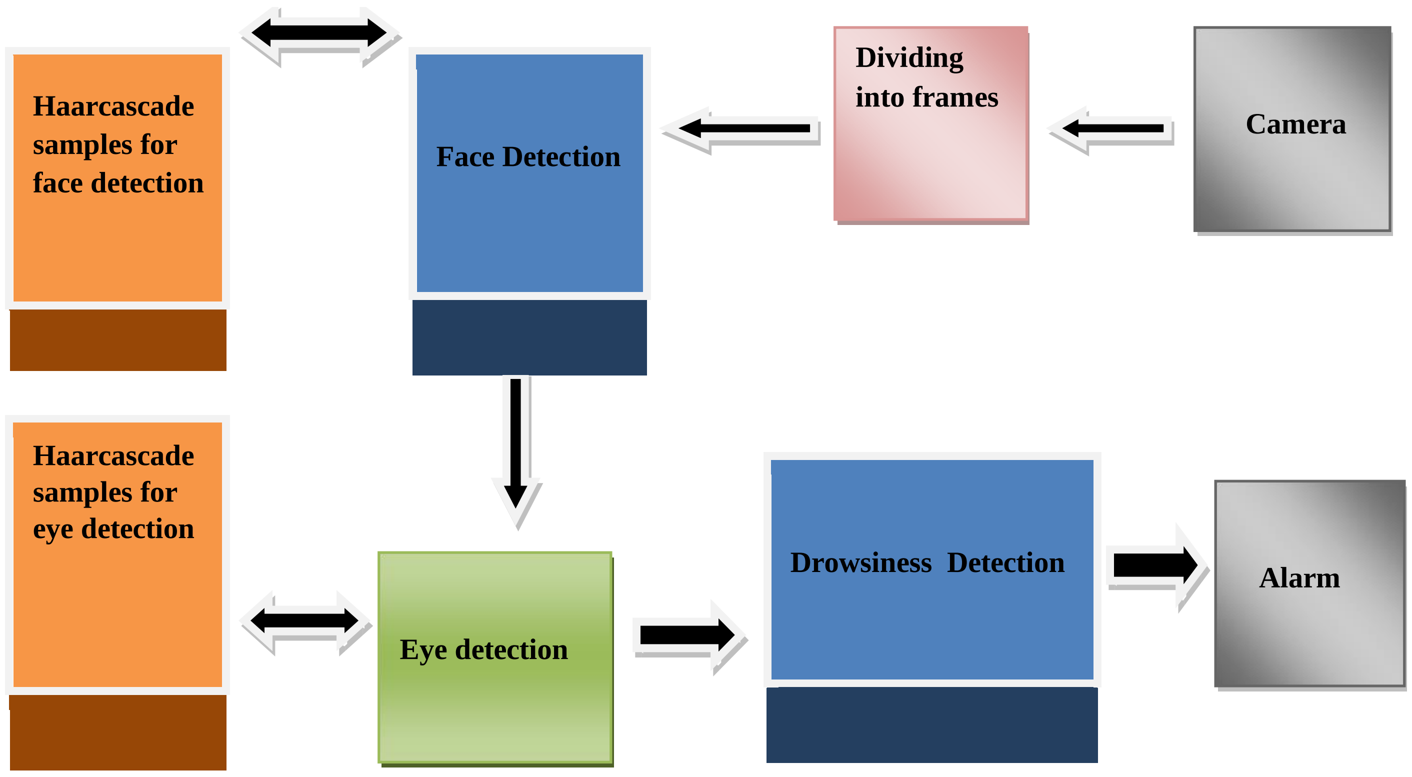


Fig 4:- Representing the usage of Python on GitHub (1 million and counting)

5.3 Algorithm Used



Drowsiness of a person can be measured by the extended period of time for which his/her eyes are in closed state. In our system, primary attention is given to the faster detection and processing of data. The number of frames for which eyes are closed is monitored. If the number of frames exceeds a certain value, then a warning message is generated on the display showing that the driver is feeling drowsy.

In our algorithm, first the image is acquired by the camera for processing. Then we use the Haarcascade file face.xml to search and detect the faces in each individual frame. If no face is detected then another frame is acquired. If a face is detected , then a region of interest is marked within the face. This region of interest contains the eyes. Defining a region of interest significantly reduces the computational requirements of the system. After that the eyes are detected from the region of interest by using Haarcascade_eye.xml.

5.3.1 Principal Component Analysis (PCA)

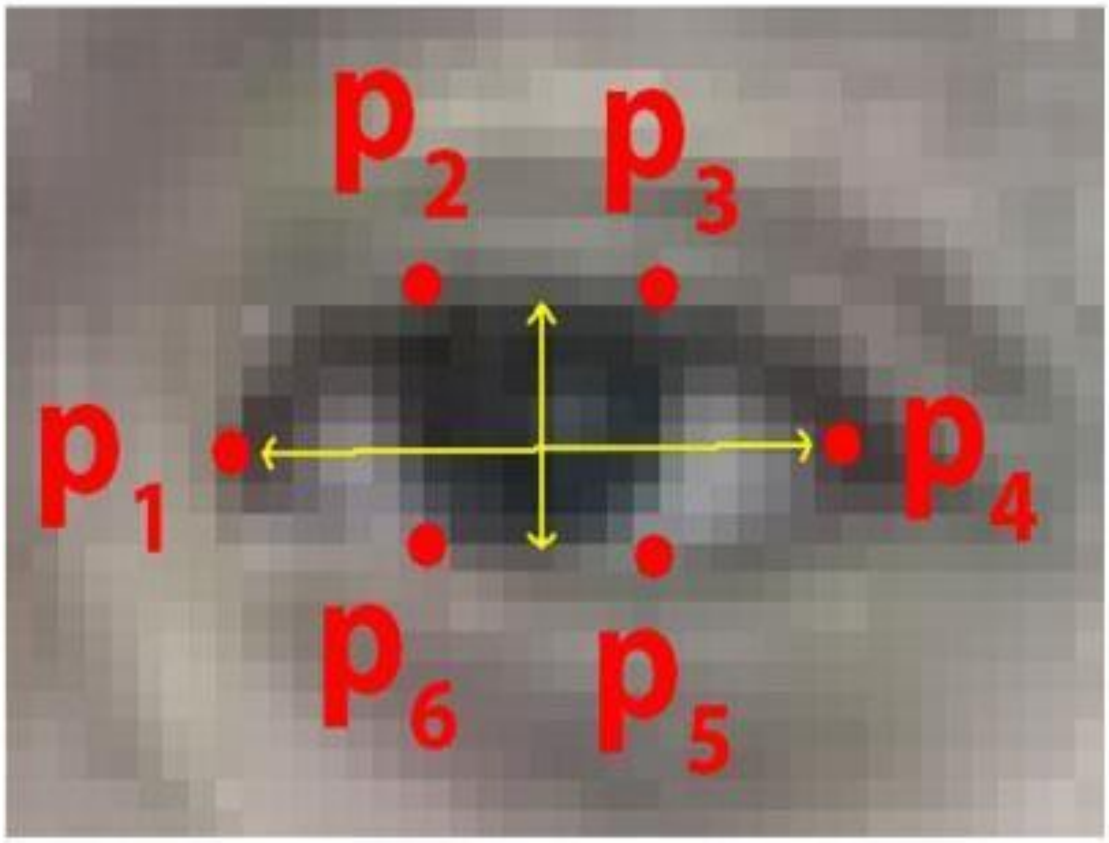


Fig 5: Image of PCA

Principal component analysis (PCA) was invented in 1901 by Karl Pearson. If the resulted data is repeated again and again or has redundancy the PCA helps in reducing this redundancy. PCA basically removes the variables to reduce redundancy. So after reduction of variables we will get less variables named as Principal Components. Principal components will generally represent all the variables present in the obtained variable. But it only reduction of variables does not solve the purpose. Main Problem appears when we try to achieve face recognition in a more and high dimensional space. The main objective of PCA is to decrease the no of dimension as well as retain more and more possible variation in the given data set. But we know that reduction in dimension results in information loss as information are directly linked with dimension. Hence we can overcome the problem of data loss by choosing the best principal components as main principal components determines the low dimension. Though use of PCA has many advantages but mostly it is used for eigen face approach. In eigen face approach the reduction of size of the data base is achieved for recognizing the test images. The obtained images are stored in the data base in vector form which are also called feature vectors. And these are found out from set of Eigen face obtained by projecting it over trained image. So basically PCA is used for Eigen face approach for the reduction of dimensionality with our causing the loss of data.

5.3.2 Image Acquisition

The function cvCaptureFromCAM allocates and initialized the CvCapture structure for reading a video stream from the camera. CvCapture* cvCaptureFromCAM(int index); Index of the camera to be used. If there is only one camera or it does not matter what camera to use , -1 may be passed.

cvSetCaptureProperty Sets camera properties For example

cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_WIDTH, 280);

cvSetCaptureProperty(capture, CV_CAP_PROP_FRAME_HEIGHT, 220);

The function cvQueryFrame() grabs a frame from a camera or video file, decompresses it and returns it. This function is just a combination of Grab Frame and Retrieve Frame, but in one call. The returned image should not be released or modified by the user. In the event of an error, the return value may be NULL.

5.3.3 Dividing into Frames:

We are dealing with real time situation where video is recorded and has to be processed. But the processing or application of algorithm can be done only on an image. Hence the captured video has to be divided into frames for analysing.

5.3.4 Face detection:-

In this stage we detect the region containing the face of the driver. A specified algorithm is for detection of face in every frame. By face detection we means that locating the face in a frame or in other words finding location of facial characters through a type of technology with the use of computer. The frame may be any random frame. Only facial related structures or features are detected and all others types of objects like buildings, tree, bodies are ignored.

We know that face is also a type of object. So we can consider detection of face as a particular case of object detection. In this type of object type of class detection, we try to know where the objects in the interest image are located and what is their size which may belongs to a particular class. The work of algorithm that is made for face detection is mostly concentrated on finding the front side of the face. But the algorithms that are developed recently focus on more general cases. For our case it may be face in the tilted position or any other portion of the faces and also it finds the possibility of multiple faces. Which means the rotation axis with respect to the present observer from the reference of face in a particular? Or even if there is vertical rotation plane then also it is able to solve the purpose. In new type of algorithm it is considered that the picture or video is a variable which means that different condition in them like hue contrast may change its variance. The amount of light may also affect. Also the position of the input may vary the output. Many calculations actualize the face-detection assignment as a two way pattern-differentiation task.

Cascade is loaded by:

```
cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );
```

storage is allocated:

```
CvMemStorage* storage = cvCreateMemStorage(0);

CvSeq* cvHaarDetectObjects(  const CvArr* image,

CvHaarClassifierCascade* cascade,  CvMemStorage*

storage,

double scale_factor = 1.1,  int

min_neighbors = 3,

int flags = 0,

CvSize min_size = cvSize(40,40)

);

 cvRectangle(  img

, cvPoint(face->x, face->y),  cvPoint(

face->x + face->width,  face->y

+ face->height

), CV_RGB(0, 255, 0),

1, 8, 0
```

)

The above function draws a rectangle in the image for the corresponding corner points .the other parameter are for drawing colour and thickness and type of lines in the rectangle.

5.3.5 Set Image Region of interest:-

```
cvSetImageROI(
img, /* the source image */  cvRect(  face->x, /* x = start
from leftmost */  face->y + (face->height)/5, /* y = a few
pixels from the top */  face->width, /* width = same width
with the face */
(face->height)/3 /* height = 1/2 of face height */
)
```

It sets the ROI for the corresponding image. We have taken from the left most point in the face to a few pixels from the top to half of face height. Width of the region is same as that of the face. The rectangular region is now used to get eyes.

5.3.6 Recognition of face region

To detect the region of face after minimizing the background extra portion from the image, labelling method is used. In the labelling method, components are connected in 2-D binary image. This function will return a matrix of the same size as binary image. It contains labels for the connected objects in binary image. it can have a value of either 4 where 4 specifies 4-connected objects or 8 where 8 specifies 8-connected objects. If the argument is omitted, it defaults to 8. The elements of matrix are integer values greater than or equal to 0. The pixels labeled 0 are the background. The pixels labelled 1 make up one object; the pixels labelled 2 make up a second object and so forth.

After that it is required to measure the properties of image regions. The region properties function is used. This function measures a set of properties for each labelled region in the label matrix. Positive integer elements of matrix correspond to different regions. For example, the set of elements of matrix equal to 1 corresponds to region 1; the set of elements of matrix equal to 2 corresponds to region 2; and so on. The return value is a structure array of length matrix. The fields of the structure array denote different measurements for each region, as specified by properties. Properties can be a comma-separated list of strings, a cell array containing strings, the single string 'all', or the string 'basic'. There are set of valid property strings. Property strings are case insensitive. This rectangle box is used to store different operated images. And these operated images are three kind of. First operated image which is stored as matrix in rectangle box is cropped cb-cr image. Second image is gray image which is converted from color image. Third image is histogram equalization image which is stored as matrix in rectangle box.

5.3.7 Eye Detection:-

In our method eye is the decision parameter for finding the state of driver. Though detection of eye may be easier to locate, but it’s really quite complicated. At this point it performs the detection of eye in the required particular region with the use of detection of several features. Generally Eigen approach is used for this process. It is a time taking process. When eye detection is done then the result is matched with the reference or threshold value for deciding the state of the driver.

Poor contrast of eyes generally creates lots of problems in its detection. After successful detection of face eye needs to be detected for further processing. In our method eye is the decision parameter for finding the state of driver. Though detection of eye does not look complex but the actual process is quite hectic. In this case it performs the detection of eye in the specified region with the use of feature detection. Generally Eigen approach is used for this process. It is a time taking process. When eye detection is done then the result is matched with the reference or threshold value for deciding the state of the driver.

5.3.8 Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. In the field of digital signal processing, there are many advantages of digital image processing with compare to analog image processing. There is a much wider range of algorithms to be applied to the input data. It can avoid problems like the build-up of noise and signal deformation during processing. Since images are defined over two dimensions, even applied in the case of more dimensions also, digital image processing may be modelled in the form of multidimensional systems.

Digital image processing permits the use of much more difficult algorithms. Digital image processing can offer both more complicated performance at simple tasks, and the implementation of methods which would be impossible by analog means. Digital image processing is the only practical technology for classification, pattern recognition, projection, feature extraction and multi-scale signal analysis.

Some techniques which are used in digital image processing names as pixlation, principal components analysis, linear filtering, anisotropic diffusion, wavelets, neural networks, independent component analysis, hidden markov model, self-organizing maps and partial differential equation. There is necessary to discuss on Digital Image processing because throughout whole methodology this technique only is used.

An image may be defined as a two-dimensional function, f(x,y), where x and y are spatial coordinates, and the amplitude of f at any pair of coordinates (x,y) is called the intensity or gray level of the image at that point. When x.y, and the amplitude values of f are all finite, discrete quantities, the image is called as a digital image. Digital image processing refers to

processing digital images using a digital computer. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels, and pixels. Pixel is the term used most widely to denote the elements of a digital image.

Vision is the most advanced of among all senses, so it is obvious that images participate the single most vital role in human perception. But unlike humans, who are limited to the visual band of electromagnetic spectrum, imaging machines cover almost the entire electromagnetic spectrum. It has range from gamma to radio waves. They can operate also on images generated by sources that humans do not usually relate with images. These include electron microscopy, ultrasound, and computer-generated images. So, it is easily seen that the digital image processing encompasses a large, huge and wide varied field of applications.

CHAPTER-6

CODING

6.1 Libraries Used

tensorflow==1.4.0

PyQt5

opencv-python

numpy

cmake

dlib

imutils

tensorflow-object_detection-api

pillow==7.1.1

matplotlib==3.2.1

lxml==4.5.0

kiwisolver==1.2.0

urllib3==1.25.3

pywin32

markupsafe==1.1.1

decorator==4.2.2

virtualenv

os

tarfile

Pyparsing == 2.3.1

6.2 GUI Code

```
from PyQt5 import QtCore, QtGui, QtWidgets

class      Ui_MainWindow(object):

def setupUi(self, MainWindow):

        MainWindow.setObjectName("MainWindow")

MainWindow.resize(644, 488)        self.centralwidget =

QtWidgets.QWidget(MainWindow)

self.centralwidget.setObjectName("centralwidget")

self.horizontalLayout = QtWidgets.QHBoxLayout(self.centralwidget)

self.horizontalLayout.setObjectName("horizontalLayout")

self.imgLabel = QtWidgets.QLabel(self.centralwidget)

        sizePolicy          =          QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Expanding,

QtWidgets.QSizePolicy.Preferred)        sizePolicy.setHorizontalStretch(0)

sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.imgLabel.sizePolicy().hasHeightForWidth())

self.imgLabel.setSizePolicy(sizePolicy)

self.imgLabel.setFrameShape(QtWidgets.QFrame.Box)

self.imgLabel.setAlignment(QtCore.Qt.AlignCenter)

self.imgLabel.setObjectName("imgLabel") self.horizontalLayout.addWidget(self.imgLabel)

self.verticalLayout = QtWidgets.QVBoxLayout()
self.verticalLayout.setObjectName("verticalLayout")        self.openCameraButton =
QtWidgets.QPushButton(self.centralwidget)
        sizePolicy          =         QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Maximum)
        sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.openCameraButton.sizePolicy().hasHeightForWidth())
self.openCameraButton.setSizePolicy(sizePolicy)        font = QtGui.QFont()
font.setPointSize(10)        self.openCameraButton.setFont(font)
self.openCameraButton.setObjectName("openCameraButton")
self.verticalLayout.addWidget(self.openCameraButton)        self.stopCameraButton =
QtWidgets.QPushButton(self.centralwidget)        font = QtGui.QFont()
font.setPointSize(10)        self.stopCameraButton.setFont(font)
self.stopCameraButton.setObjectName("stopCameraButton")
self.verticalLayout.addWidget(self.stopCameraButton)
        self.startDetectionButton = QtWidgets.QPushButton(self.centralwidget)
        sizePolicy          =         QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Fixed)        sizePolicy.setHorizontalStretch(0)
sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.startDetectionButton.sizePolicy().hasHeightForWidth())
self.startDetectionButton.setSizePolicy(sizePolicy)        font = QtGui.QFont()
```

```
font.setPointSize(10)        self.startDetectionButton.setFont(font)

self.startDetectionButton.setObjectName("startDetectionButton")

self.verticalLayout.addWidget(self.startDetectionButton)        self.stopDetectionButton =

QtWidgets.QPushButton(self.centralwidget)        font = QtGui.QFont()

font.setPointSize(10)        self.stopDetectionButton.setFont(font)

self.stopDetectionButton.setObjectName("stopDetectionButton")

self.verticalLayout.addWidget(self.stopDetectionButton)        self.exitButton =

QtWidgets.QPushButton(self.centralwidget)

        sizePolicy            =            QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Minimum,
QtWidgets.QSizePolicy.Maximum)

        sizePolicy.setHorizontalStretch(0) sizePolicy.setVerticalStretch(0)

sizePolicy.setHeightForWidth(self.exitButton.sizePolicy().hasHeightForWidth())

self.exitButton.setSizePolicy(sizePolicy)        font = QtGui.QFont()

font.setPointSize(10)        self.exitButton.setFont(font)

self.exitButton.setObjectName("exitButton")

self.verticalLayout.addWidget(self.exitButton)

        spacerItem = QtWidgets.QSpacerItem(20, 40, QtWidgets.QSizePolicy.Minimum,

QtWidgets.QSizePolicy.Expanding)        self.verticalLayout.addItem(spacerItem)

self.horizontalLayout.addLayout(self.verticalLayout)

MainWindow.setCentralWidget(self.centralwidget)        self.statusbar =

QtWidgets.QStatusBar(MainWindow)        self.statusbar.setObjectName("statusbar")

        MainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(MainWindow)

        QtCore.QMetaObject.connectSlotsByName(MainWindow)


    def retranslateUi(self, MainWindow):

        _translate = QtCore.QCoreApplication.translate

        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))

self.imgLabel.setText(_translate("MainWindow", "Live Camera Feed"))

self.openCameraButton.setText(_translate("MainWindow", "Open Camera"))

self.stopCameraButton.setText(_translate("MainWindow", "Stop Camera"))

self.startDetectionButton.setText(_translate("MainWindow", "Start Detection"))

self.stopDetectionButton.setText(_translate("MainWindow", "Stop Detection"))

self.exitButton.setText(_translate("MainWindow", "Exit"))




if __name__ == "__main__":

    import sys     app =

QtWidgets.QApplication(sys.argv)

MainWindow = QtWidgets.QMainWindow()

ui = Ui_MainWindow()

ui.setupUi(MainWindow)

MainWindow.show()     sys.exit(app.exec_())
```

This code helps us to create a graphical window on the screen which we open as soon as our program gets compiled successfully and starts to run.

6.3 Main Code

```
import cv2 import sys import

numpy as np from PyQt5

import QtCore from

PyQt5.QtCore import * from

PyQt5.QtGui import * from

PyQt5.QtWidgets import *

import MainWindow_gui

import dlib from imutils

import face_utils import

tensorflow as tf


import tarfile


import os from object_detection.utils import label_map_util

from object_detection.utils import visualization_utils as vis_util

class FaceSecurityML(QMainWindow,

MainWindow_gui.Ui_MainWindow):
    # serialWrite = QtCore.pyqtSignal(object)

    def __init__(self):
        super(FaceSecurityML, self).__init__()

        self.setupUi(self)

        # What model to download.
        MODEL_NAME = 'ssd_mobilenet_v1_coco_2017_11_17'
        MODEL_FILE = MODEL_NAME + '.tar.gz'
        DOWNLOAD_BASE = 'http://download.tensorflow.org/models/object_detection/'

        # Path to frozen detection graph. This is the actual model that is used for the object
detection.
        PATH_TO_CKPT = MODEL_NAME + '/frozen_inference_graph.pb'

        # List of the strings that is used to add correct label for each box.
        PATH_TO_LABELS = os.path.join('data', 'mscoco_label_map.pbtxt')

        NUM_CLASSES = 90
        # opener = urllib.request.URLopener()
        # opener.retrieve(DOWNLOAD_BASE + MODEL_FILE, MODEL_FILE)

        tar_file    =    tarfile.open(MODEL_FILE)

for file in tar_file.getmembers():

            file_name    =    os.path.basename(file.name)

if 'frozen_inference_graph.pb' in file_name:

                tar_file.extract(file, os.getcwd())

self.detection_graph = tf.Graph()          with

self.detection_graph.as_default():

            od_graph_def = tf.GraphDef()             with

tf.gfile.GFile(PATH_TO_CKPT, 'rb') as fid:                serialized_graph

= fid.read()                 od_graph_def.ParseFromString(serialized_graph)

tf.import_graph_def(od_graph_def, name='')         label_map =

label_map_util.load_labelmap(PATH_TO_LABELS)

        categories            =          label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NUM_CLASSES,

                                                            use_display_name=True)

        self.category_index        =        label_map_util.create_category_index(categories)

self.sess = tf.Session(graph=self.detection_graph)


        self.openCameraButton.clicked.connect(self.openCameraClicked)

self.stopCameraButton.clicked.connect(self.stopCameraClicked)

self.startDetectionButton.clicked.connect(self.startAllDetection)
self.stopDetectionButton.clicked.connect(self.stopAllDetection)
self.exitButton.clicked.connect(self.exitClicked)


        print("[INFO] loading facial landmark predictor...")         self.faceCascade =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')         self.predictor =
dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')


        # grab the indexes of the facial landmarks for the left and
        # right eye, respectively


        (self.lStart, self.lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
        (self.rStart, self.rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
        (self.mStart, self.mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]


        self.EYE_AR_THRESH = 0.3
        self.EYE_AR_CONSEC_FRAMES = 20
        self.COUNTER = 0
self.YAWN_COUNTER = 0
self.FACE_COUNTER = 0
self.ALARM_ON = False
self.start_detection_Flag = False
self.frequency = 3500         self.duration
= 1000
        #             self.speak             =             wincl.Dispatch("SAPI.SpVoice")
self.systemStatusFlag = False


    @pyqtSlot(bool)     def
updateSystemStatus(self, status):
self.systemStatusFlag = status


    @pyqtSlot()     def
startAllDetection(self):
self.start_detection_Flag = True
        # self.SerialThread.start()


    @pyqtSlot()
    def stopAllDetection(self):
        self.start_detection_Flag = False
        # if (self.SerialThread.isRunning()):
        #     self.SerialThread.terminate()


    def blinkDetector(self, img):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

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

        # import the necessary packages

        # Definite input and output Tensors for detection_graph         image_tensor =
self.detection_graph.get_tensor_by_name('image_tensor:0')         # Each box
represents a part of the image where a particular object was detected.
detection_boxes = self.detection_graph.get_tensor_by_name('detection_boxes:0')

        # Each score represent how level of confidence for each of the objects.

        # Score is shown on the result image, together with the class label.

        detection_scores = self.detection_graph.get_tensor_by_name('detection_scores:0')
detection_classes = self.detection_graph.get_tensor_by_name('detection_classes:0')
num_detections = self.detection_graph.get_tensor_by_name('num_detections:0')


        # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
image_np_expanded = np.expand_dims(img, axis=0)


        # Actual detection.

        (boxes, scores, classes, num) = self.sess.run(

           [detection_boxes,       detection_scores,       detection_classes,       num_detections],
feed_dict={image_tensor: image_np_expanded})

        # print(classes)
        # cat_util.


        # Visualization of the results of a detection.


        list_output = vis_util.getlabels_on_image_array(
            img,            np.squeeze(boxes),
np.squeeze(classes).astype(np.int32),
np.squeeze(scores),
self.category_index,
use_normalized_coordinates=True,
line_thickness=8)
        print(list_output)
        if  list_output.__contains__('cell  phone')  or  list_output.__contains__('toothbrush')  or
list_output.__contains__(
                'remote') or list_output.__contains__('remote'):
            cv2.putText(img, "Be Alert!!", (20, 50), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,
0, 255), 2)
            print('\a')


        # else:
```

```
        # self.DisplayImage(img, 1)


        if len(rects) is not 0:
            self.FACE_COUNTER = 0


            for (x, y, w, h) in rects:
                rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))
shape = self.predictor(gray, rect)                shape =
face_utils.shape_to_np(shape)


                leftEye = shape[self.lStart:self.lEnd]
                rightEye = shape[self.rStart:self.rEnd]
leftEAR = self.eye_aspect_ratio(leftEye)
rightEAR = self.eye_aspect_ratio(rightEye)


                mouthshape = shape[self.mStart:self.mEnd]                mouthOpenDistance =
self.euclidean_dist(mouthshape[18], mouthshape[14])                #
print(mouthOpenDistance)


                ear = (leftEAR + rightEAR) / 2.0
            leftEyeHull = cv2.convexHull(leftEye)
rightEyeHull = cv2.convexHull(rightEye)
cv2.drawContours(img, [leftEyeHull], -1, (0, 255, 0), 1)
cv2.drawContours(img, [rightEyeHull], -1, (0, 255, 0), 1)

            if      ear      <      self.EYE_AR_THRESH:
self.COUNTER += 1
               # print('Counter:',self.COUNTER)

               # if the eyes were closed for a sufficient number of
               # frames, then sound the alarm
               if self.COUNTER >= self.EYE_AR_CONSEC_FRAMES:
                  # if the alarm is not on, turn it on
if not self.ALARM_ON:
self.ALARM_ON = True
print("ALARM_ON")
                     # os.system("beep -f 555 -l 460")
                     print('\a')
                     # winsound.Beep(self.frequency, self.duration)
                     # self.sendData('a')
```

```
                        # draw an alarm on the frame                         cv2.putText(img,
"DROWSINESS ALERT!", (10, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

                        self.DisplayImage(img, 1)

                    # otherwise, the eye aspect ratio is not below the blink
                    # threshold, so reset the counter and alarm
                else:
                    self.COUNTER = 0
self.ALARM_ON = False                      #
self.sendData('x')


                if mouthOpenDistance > 6:
                    self.YAWN_COUNTER += 1

                    if self.YAWN_COUNTER >= 15:
                        print('Driver      is      Yawning      !')
print('\a')
                        if                not                self.ALARM_ON:
self.ALARM_ON = True
```

```
                        # print("ALARM_ON")

                        # winsound.Beep(self.frequency, self.duration)


                        # self.speak.Speak("you are feeling sleepy, please refrain from driving and
refresh yourself")

                        # self.sendData('a')


                        # draw an alarm on the frame

cv2.putText(img, "Yawning ALERT!", (100, 30),

cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

self.DisplayImage(img, 1)



else:

                 self.YAWN_COUNTER              =              0

self.ALARM_ON = False

                 # self.sendData('x')


                 # draw the computed eye aspect ratio on the frame to help

                 # with debugging and setting the correct eye aspect ratio
```

```
                    # thresholds and frame counters                 cv2.putText(img,
"EAR: {:.3f}".format(ear), (500, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


                self.DisplayImage(img, 1)


else:
            self.FACE_COUNTER += 1


            if        self.FACE_COUNTER        >=        15:
print('Driver Not AWAKE !')
                # self.speak.Speak("Wake UP")
                #                self.sendData('s')
print('\a')
                cv2.putText(img, "Driver Not AWAKE !", (100, 30),
cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)                self.DisplayImage(img, 1)


    def parse_Serial_data(self):
        pass


    def euclidean_dist(self, ptA, ptB):
        # compute and return the euclidean distance between the two
        # points                    return
np.linalg.norm(ptA - ptB)

    def eye_aspect_ratio(self, eye):
        # compute the euclidean distances between the two sets of
        # vertical eye landmarks (x, y)-coordinates

A = self.euclidean_dist(eye[1], eye[5])
B = self.euclidean_dist(eye[2], eye[4])

        # compute the euclidean distance between the horizontal
        # eye landmark (x, y)-coordinates
C = self.euclidean_dist(eye[0], eye[3])

        # compute the eye aspect ratio
ear = (A + B) / (2.0 * C)

        # return the eye aspect ratio
return ear

    @pyqtSlot()                 def
openCameraClicked(self):
        self.capture = cv2.VideoCapture(0)
self.capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 600)
self.capture.set(cv2.CAP_PROP_FRAME_WIDTH, 800)

        self.timer = QTimer(self)
self.timer.timeout.connect(self.update_frame)        self.timer.start(0.1)

    def update_frame(self):
        ret,       self.image       =       self.capture.read()
self.image = cv2.flip(self.image, 1)

        if                (self.start_detection_Flag                ==                True):
self.blinkDetector(self.image)
        else:
            self.DisplayImage(self.image, 1)

    @pyqtSlot()     def
stopCameraClicked(self):
self.timer.stop()
self.capture.release()
```

```
    def DisplayImage(self, img, window=1):

qformat = QImage.Format_Indexed8

if len(img.shape) == 3:             if

(img.shape[2]) == 4:

                qformat = QImage.Format_RGBA8888

            else:

                qformat   =   QImage.Format_RGB888   outImg   =   QImage(img,   img.shape[1],
img.shape[0], img.strides[0], qformat)



        outImg = outImg.rgbSwapped()



        if window == 1:

            self.imgLabel.setPixmap(QPixmap.fromImage(outImg))

self.imgLabel.setAlignment(QtCore.Qt.AlignCenter | QtCore.Qt.AlignVCenter)

self.imgLabel.setScaledContents(True)



    @pyqtSlot()         def

exitClicked(self):

        # self.SerialThread.Close_Serial()

        QApplication.instance().quit()
```

```
app = QApplication(sys.argv)

window = FaceSecurityML()

window.show() app.exec_()
```

This is the main code of our program which will detect the eyes and on the basis of the state of the eyes it will notify that whether the driver is feeling drowsy or not.

CHAPTER 7

RESULT

As soon as we start or run the project, a GUI window will open on the screen in front of the user so that the user can interact with that window and did his, her work. The window consists of 5 buttons, each of them having a different and unique functionality.

The project is about to detect the drowsiness of the driver by detecting the eyes of the driver and that basis we will judge that whether the driver is feeling drowsy or not. If the driver eyes are closed than an alarm is blown so that the driver and rest of the members in the car can know about that the accidents can be prevented. The alarm is blown if the driver eyes are closed, driver is not in camera frame, driver is yawning and if he is using the mobile phone then in all these cases the alarm will be blown.

7.1 Snapshots

There are some snapshots of the user using the software which tell us that how the software will interact with the user and what will be its functionality in different cases.

Snapshots with their description are described below:

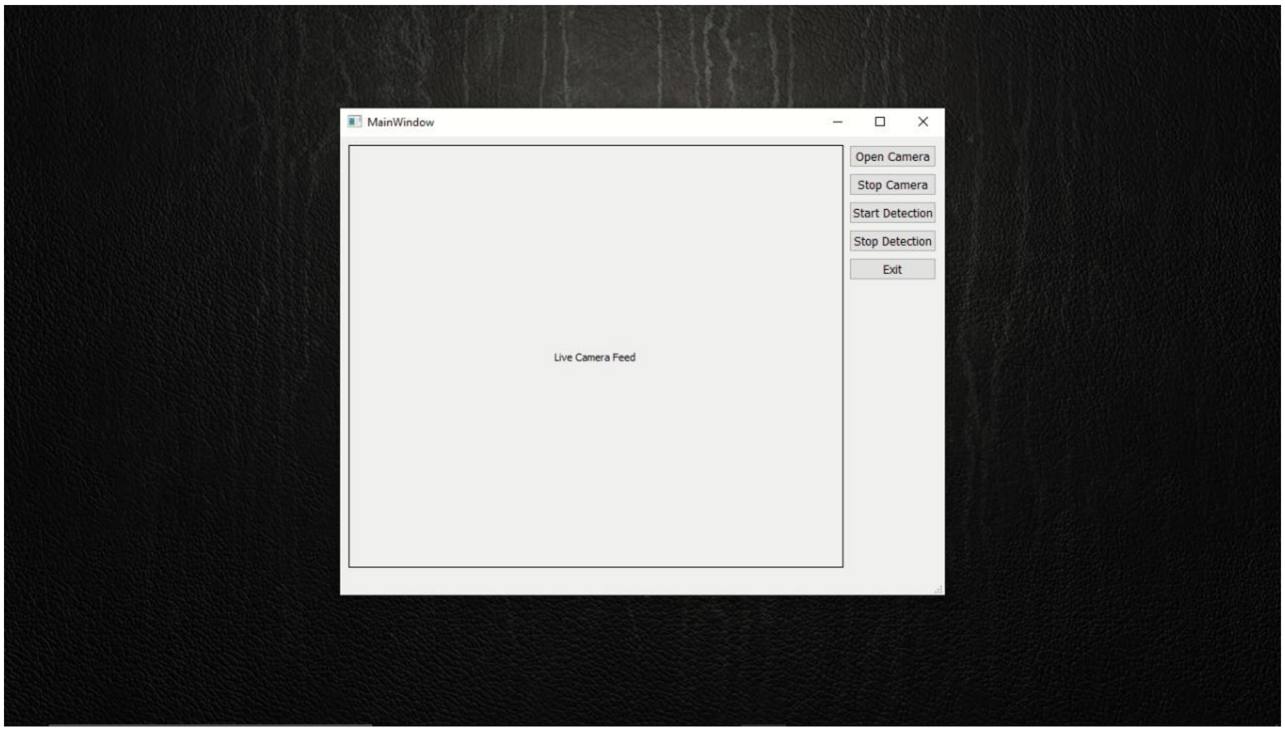


Fig 6: GUI Window

This is the first screen that will be opened on the main screen as soon as our code gets executed. This contains a GUI screen which has 5 buttons on it, the working of each button can be easily understood by the names only. To open the camera, to close the camera, to start detection, to stop detection and to exit or close the program execution. This is the working of the 5 buttons available on the screen.

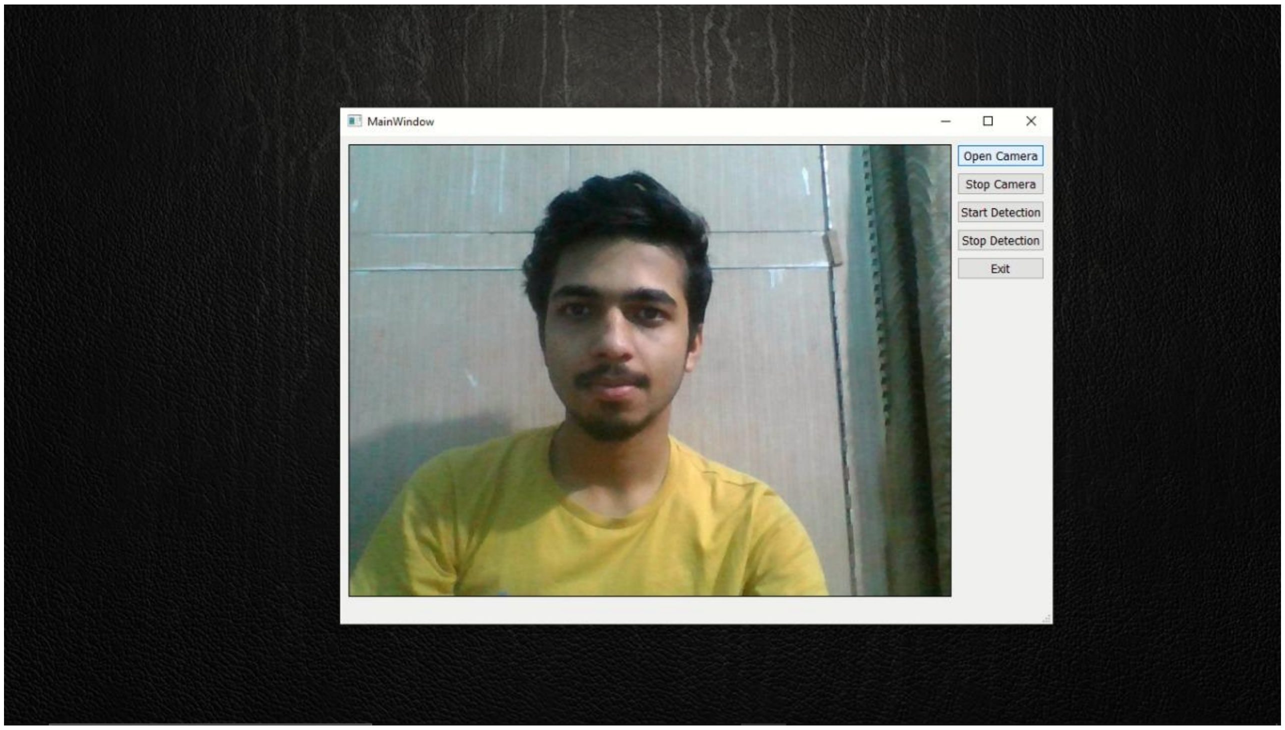
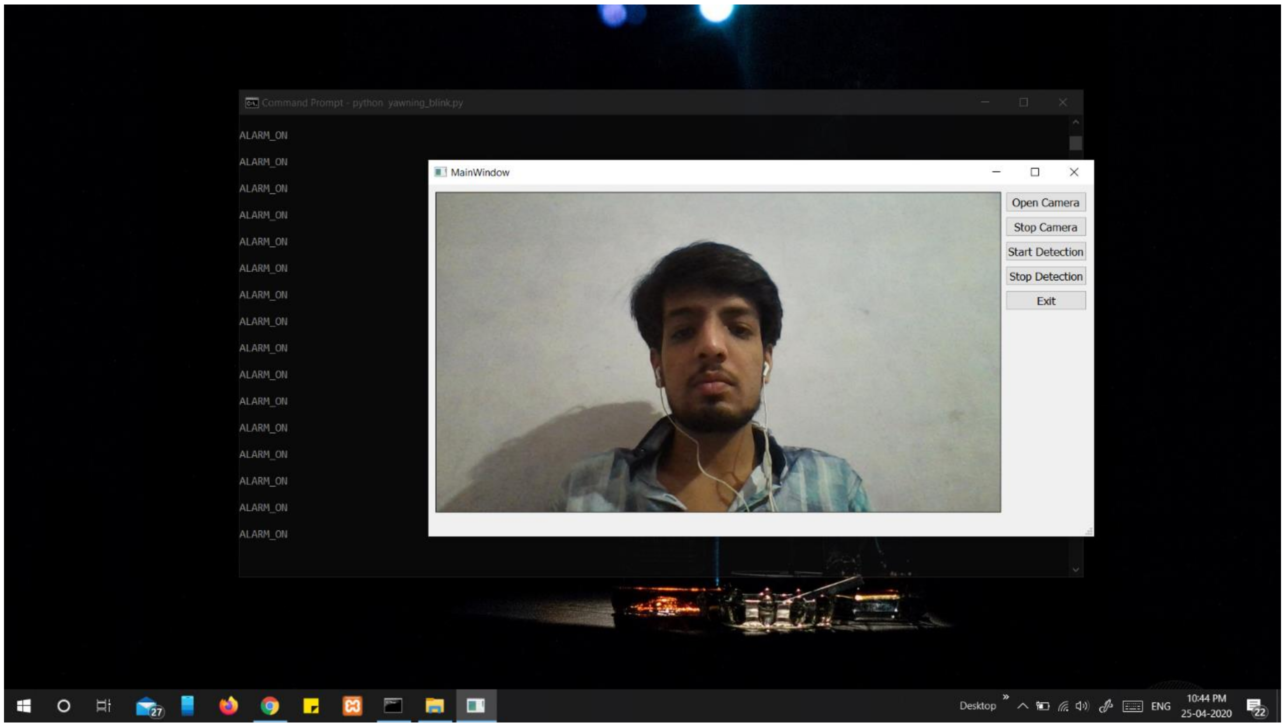


Fig 7: Image when user click on open camera button

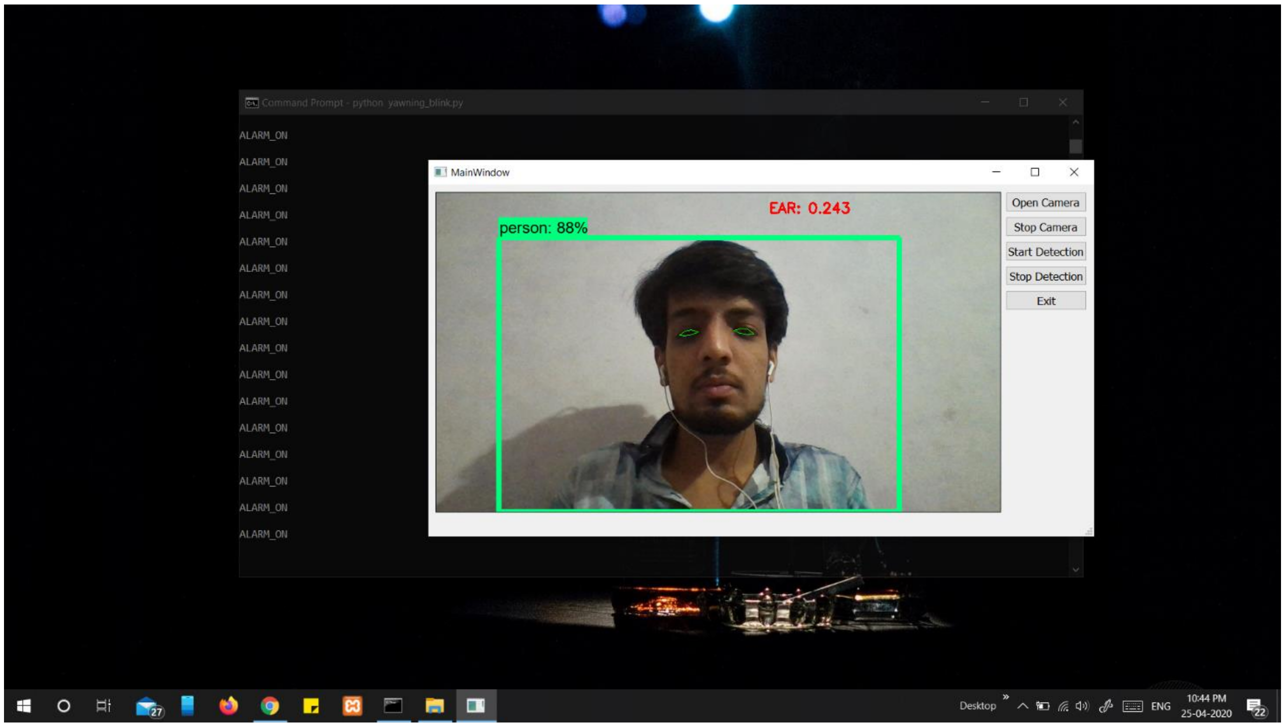
This is the image when the user click on the open camera button the camera of the system will get open and starts to detect the person who so ever is within its range of the system i.e. it will behave as a normal camera to capture the images. It will detect all the gestures of the person means if he is moving his face and eyes here and there then the camera will act accordingly.



```
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
```

Fig 8: Image when user click on stop camera button

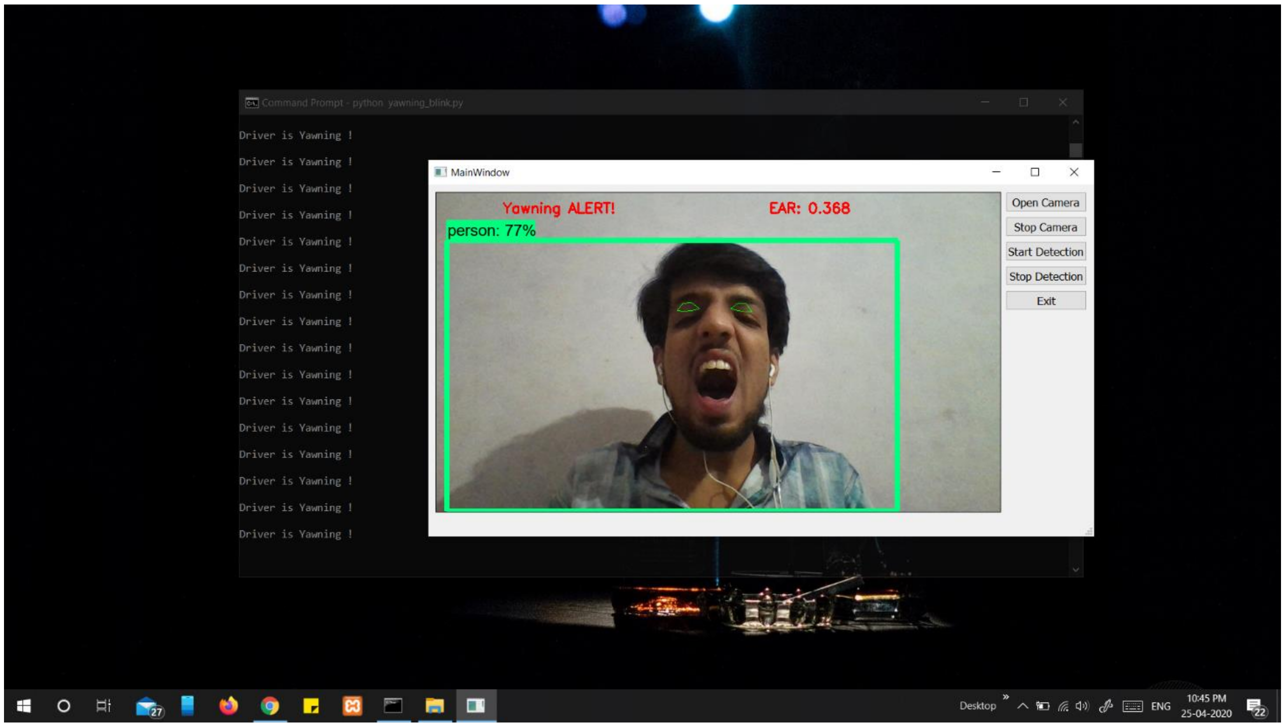
This is the image when the user click on the stop camera button the camera will get closed as soon as the button is clicked and the image of the user will be as it is display on the screen. It will continuously keep that same image on the screen and in the same manner until the user again click on the open camera button.



```
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
```

Fig 9: Image when user click on start detection button

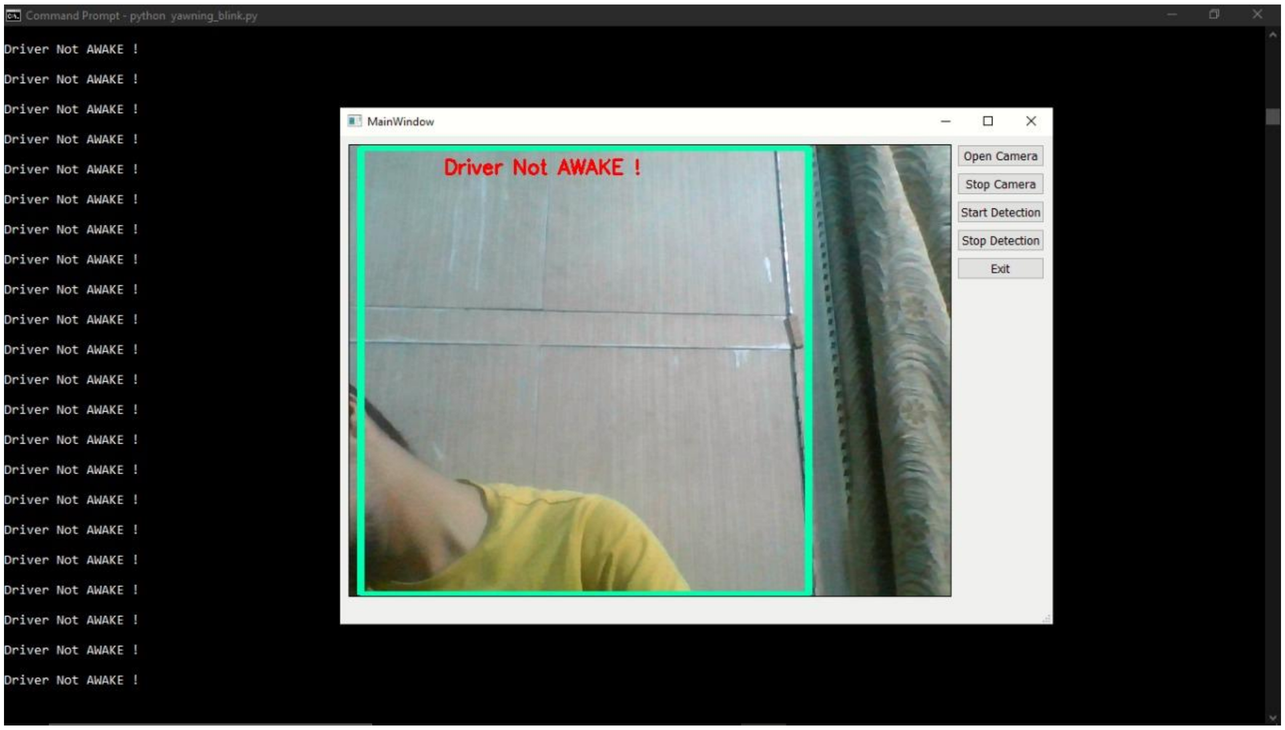
This is the image when the user click on the start detection button then it starts to detect the eyes position by scanning the eyes continuously and on the basis of that we recognize that whether the driver is feeling drowsy or not. If the driver eyes are found to be closed for successive number of frames then we can say that the driver is drowsy else the driver is not drowsy or awake,



```
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
Driver is Yawning !
```

Fig 10: Image when user did yawning analysis

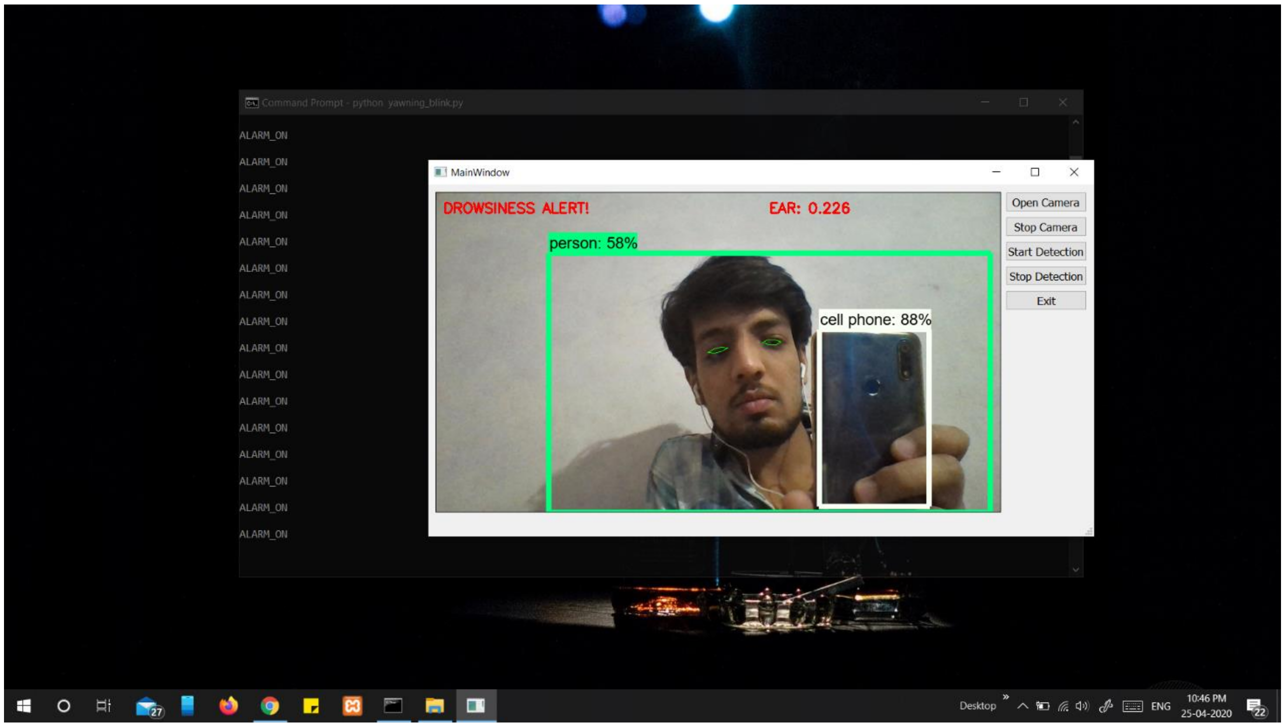
This is the image when the user click on the start detection button and the system starts to detect. This is the image when the driver is driving and yawning , so after some number of yawning we can make an alert that the driver is feeling drowsy by blowing a alarm or buzzer, so that the driver can be awake and the accident can be prevented or avoided.



```
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
Driver Not AWAKE !
```

Fig 11: Image when user is not in frame

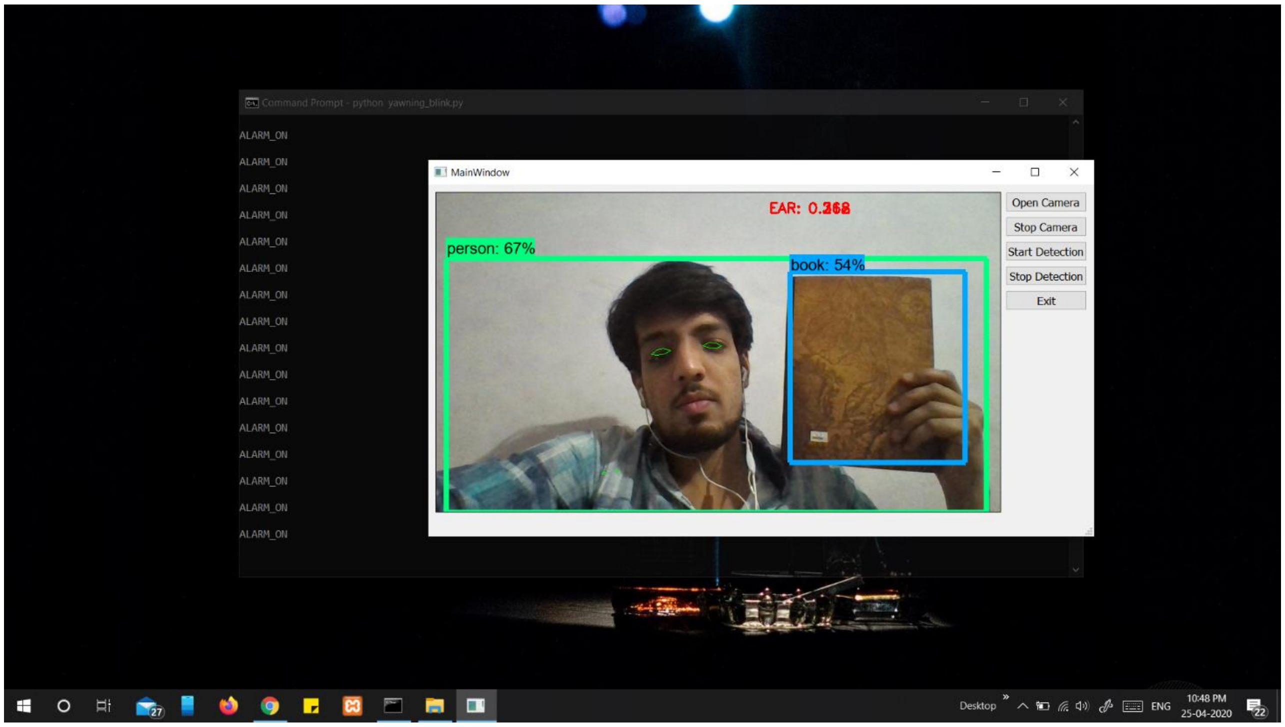
This is the image when the user click on the start detection button and the system starts to detect. This is the image when the driver is not in frame i.e. the driver is not in the range of the camera. We can conclude from it that the driver might had fallen asleep so he had bended towards the side of the gate side or towards the non-driver side, so accident can occur so in order to avoid the accident we will blow the alarm or buzzer so that the driver will be awake and the accident can be prevented.



```
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
```

Fig 12: Image when user did mobile detection

This is the image when the user click on the start detection button and the system starts to detect. This is the image when the system had started detection but is not able to scan the eyes of the driver and the driver is using his mobile phone and the system has detected the phone than also the system will blow an alarm or buzzer so that the driver got awake that his attention is not in driving car but in something else. So, in order to prevent the accident, he should focus on driving the car and not on anything else.



```
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
ALARM_ON
```

Fig.13: Image when user is distracted with some other objects

This is the image when the user click on the start detection button and the system starts to detect. This is the image when the system had started detection but is not able to scan the eyes of the driver and the driver is busy with some other objects and the system has detected the presence of other object. In that case also, the system will blow an alarm or buzzer so that the driver got awake that his attention is not in driving car but in something else. So, in order to prevent the accident, he should focus on driving the car and not on anything else.

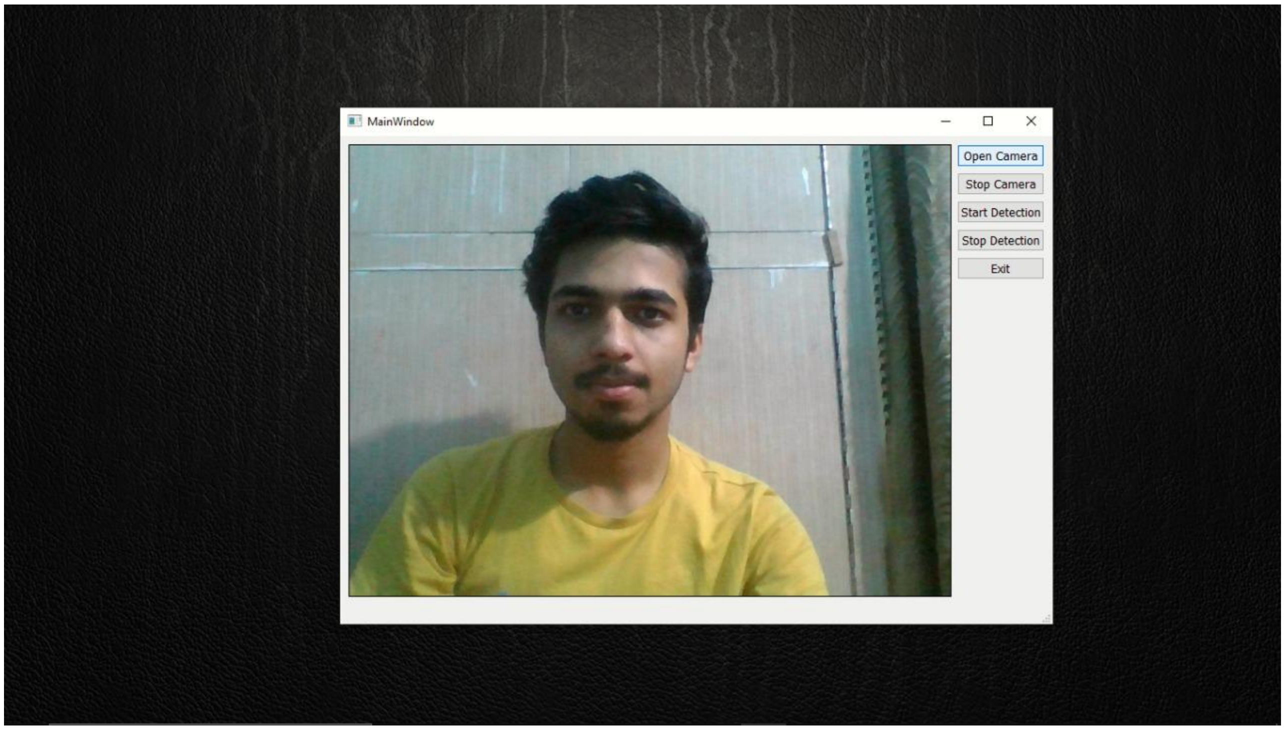


Fig 14: Image when user click on stop detection button

This is the image when the user click on the stop detection button and the system stops to detect. This is the image when the system had stopped detection and is not able to scan the eyes of the driver and working like a simple camera, to just show your face.

CHAPTER 7

CONCLUSION

This is study that uses the coordinates of eyes and mouth for the detection of drowsiness of the driver and also to check that whether the driver is within the camera scope or not and to check that driver is using is using the mobile phone or not while driving.

In this the capture image of the driver is divided into frames and then they are analysed. Successful detection is followed by the detection of eyes and mouth. If closure of eyes for successive number of frames were detected then it is detected as a drowsy condition else if it is regarded as a normal blink of eye and the loop of capturing image and analysing the state of driver is carried out again and again. If the eyes are detected to be closed for more than 20 consecutive frames, the alarm/buzzer sets on or the alarm/buzzer is sets on when the driver face can't be detected or not in the range of the camera or either he is using his mobile phone while driving and his phone has been detected in the camera.

The system has been tried and tested in different lighting conditions and with different people with varied facial characteristics especially the eyes. It has been experimentally found that absolute accuracy is achieved when the lighting conditions are bright and favourable. The biggest drawback experienced till now is the presence of beard or sunglasses and spectacles which are coloured but works completely good with transparent on the driver's face. This inferences with the detection of eyes and mouth may lead to false triggering.

This system is a real time system and checks the state of the state of the driver all through the journey that he is drowsy or not. In order to have efficient usage of the system we are not saving the images we are just capturing the images using them and then discarding them after their use has been finished and starts dealing with another image or other image.

This reduces the memory requirement, our expense and makes the system much faster.

CHAPTER 8

FUTURE SCOPE

The future scope for this project includes increasing the speed of operation of the system and hence increase the accuracy rate. Further, this concept can be extended to provide an inexpensive solution for commercial vehicles.

The difficulties faced due to bad lighting that may occur while driving during night time is a potent problem that needs to be taken care of. Bearded men and people wearing coloured sunglasses while driving too should be able to use this system accurately. This is a drawback that needs to be mended as future scope.

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

Since, it brings the vehicle speed down to a controllable limit, the chances of accident occurrence is greatly reduced which is quite helpful for avoiding crashes caused by the drowsiness of the driver related cases.

REFERENCES

1. https://en.wikipedia.org/wiki/Driver_drowsiness_detection
2. https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/
3. Alexander Colic, Oge Marques, Borko Furht, "Driver Drowsiness Detection: Systems and Solutions" – 2014th Edition
4. Stack Overflow - https://stackoverflow.com/questions
5. GitHub - https://github.com/tensorflow/tensorflow
6. Adrian Kaehler and Gary Bradski, "Learning OpenCv".
7. You Tube - https://www.youtube.com/channel/UC0rqucBdTuFTjJiefW5t-IQ