A Project Report

on

Driver Drowsiness Detection

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelors of Technology

GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under the Supervision of
Mr.V.Gokul Rajan (Assistant Professor)

Submitted By:
Mayank Jain (18021100039)
Muskan Prakash(18021011357)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATERNOIDA INDIA
December, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"Driver Drowsiness Detection"** in partial fulfillment of the requirements for the award of BACHELORS OF TECHNOLOGY submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of July 2021 to December 2021, under the supervision of Mr.V. GokulRajan(Assistant Professor), Department of Computer Science and Engineering, Galgotias University, Greater Noida.

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

Mayank Jain, 18021180039

Muskan Prakash, 18021011357

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr.V. Gokul Rajan(Assistant Professor)

CERTIFICATE

The Final Project Viva-Voce examination of Mayank Jain (18021180039) and Muskan Prakash (18021011357) has been held on ________________ and their work is recommended for the award of Bachelors of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator

Signature of Dean

Date: December, 2021
Place: Galgotias University, Greater Noida

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our guide Mr. V. Gokul Rajan, as well as our esteemed university who gave us the golden opportunity to do this wonderful project on the topic Driver Drowsiness Detection which also helped us in doing a lot of Research and We came to know about so many new things. We are really thankful to them.

Secondly, we would also like to thank our parents and friends who helped us a lot in finalizing this project within the limited time frame.

MAYANK JAIN (18SCSE1180040)
MUSKANPRAKASH(18SCSE10109)

ABSTRACT

Drowsiness and fatigue are one of the main causes leading to road accidents. Theycan be prevented by taking effort to get enough sleep before driving, drink coffee orenergydrink,orhavearestwhenthesignsofdrowsinessoccur.Thepopulardrowsiness detection method uses complex methods, such as EEG and ECG. Thismethodhashighaccuracyforitsmeasurementbutitneedstousecontactmeasurement and it has many limitations on driver fatigue and drowsiness monitor .Thus, it is not comfortable to be used in real time driving.

The designed system deals with detecting the face area of the image captured from the video. The purpose of using the face area so it can narrow down to detect eyes and mouth within the face area. Once the face is found, the eyes and mouth are found by creating the eye for left and right eye detection and also mouth detection. The parameters of the eyes and mouth detection are created within the face image.

This Project is based on the research conducted and the project made in the field of computer engineering to develop a system for driver drowsiness detection to preventaccidentsfromhappeningbecauseofdriverfatigueandsleepiness.Thereportproposed the results and solutions on the limitedimplementation of the various techniques that are introduced in the project. Whereas the implementation of the project gives the real-world idea of how the system works and what changes can be done in order to improve the utility of the overall system.

Keywords— Driverdrowsiness; eyedetection; yawndetection; blinkpattern; fatigu

TABLE OF CONTENTS

1

List of Figures

CHAPTER1

Introduction

I. PURPOSE

Driver exhaustion is a noteworthy factor in countless mishaps. Late measurements gauge that yearly 1,200 passing’s and 76,000 wounds can be credited to fatigue related crashes.

Humans have always invented machines and devised techniques to ease and protect their lives, for mundane activities like traveling to work, or for more interesting purposes like aircraft travel. With the advancement in technology, modes of transportation kept on advancing and our dependency on it started increasing exponentially. It has greatly affected our lives as we know it. Now, we can travel to places at a pace that even our grandparents wouldn’t have thought possible. In modern times, almost everyone in this world uses some sort of transportation every day. Some people are rich enough to have their own vehicles while others use public transportation. However, there are some rules and codes of conduct for those who drive irrespective of their social status. One of them is staying alert and active while driving.

Driver drowsiness and fatigue is a major factor which results into numerous vehicle accidents. Developing and maintaining technologies which can efficiently detect or prevent drowsiness at the wheel and alert the driver before am mishap is a major challenge in the field of accident prevention systems. Because of the dangerous that drowsiness can cause on the roads some methods need to be developed for preventing counteracting its effects.

With the advent of modern technology and real time scanning systems using cameras we can prevent major mishaps on the road by alerting car driver who is feeling drowsy through a drowsiness detection system.

The point of this undertaking is to build up a prototype drowsiness detection system .The spotlight will be put on planning a framework that will precisely monitor the open or shut condition of the driver's eyes continuously. By monitoring the eyes, it is believed that the symptoms of driver fatigue can be detected early enough to avoid a car accident. Detection of fatigue involves the observation of eye movements and blink patterns in a sequence of images of a face.

II. FACTORS CAUSING DROWSY DRIVER

Humans tend to ignore fatigue as a constraint of their under-performance. This can lead to many dangerous situations especially when drivers are responsible for their life as well as the life of other people on the road. Driver drowsiness is a dangerous mixture of driving while fatigued and sleepy.

Some of the factors that lead to fatigue are:

1)Driver fatigue generally happens when the driver has not slept well in the past 24 hours. An average human should sleep at least 7 hours every day for better health.

2)Other factors such as sleep disorders like insomnia, Sleep Apnea and Shift work sleep disorder (SWSD) which can occur because of irregular hours at work .

3)Too much work pressure can cause stress and anxiety which often leads to loss of sleep during normal hours.

4) According to a study, people who drive commercial vehicles especially trucks suffer from drowsy driving more regularly than non-transport drivers.

This is mainly because of increased demand for workers and higher pay for greater work hours at odd times of the day.

5)The human brain is trained to relate to sleep and night hours. However, this is the time when most transport vehicles are on the move. This leads to an increase in the number of cases of drowsy driving as the drivers fall asleep easily while driving at night compared to the day.

`PRODUCT SCOPE

There are many products out there that provide the measure of fatigue level in the drivers which are implemented in many vehicles. The driver drowsiness detectionsystemprovidesthesimilarfunctionalitybutwithbetterresultsandadditionalbenefits.Also,it alerts the user on reaching a certain saturation point of the drowsiness measure.

PROJECT FEATURES

Driver fatigue their time requirement is a fraction of the previously used methods. The algorithm starts with the detection of heads on colour pictures using deviations in colour and structure of the human face and that of the background. By normalizing the distance and position of the reference points, all faces should be transformed into the same size and position. For normalization, eyes serve as points of reference. Other OpenCV algorithm finds the eyes on any grayscale image by searching characteristic features of the eyes and eye sockets. Tests made on a standard database show that the algorithm works very fast and it is reliable.Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces)to train the classifier. Then we need to extract features from it. For this, OpenCV features shown in the below image are used. They are just like our convolutional kernel.

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.

There are many products out there that provide the measure of fatigue level in the drivers which are implemented in many vehicles. The driver drowsiness detection system provides the similar functionality but with better results and additional benefits. Also, it alerts the user on reaching a certain saturation point of the drowsiness measure.

FACTS & STATISTICS

Our current statistics reveal that just in 2015 in India alone, 148,707 people died due to car related accidents. Of these, at least 21 percent were caused due to fatigue causing drivers to make mistakes. This can be a relatively smaller number still, as among the multiple causes that can lead to an accident, the involvement of fatigue as a cause is generally grossly underestimated. Fatigue combined with bad infrastructure in developing countries like India is a recipe for disaster. 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. When there is an increased need for a job, the wages associated with it increases leading to more and more people adopting it. Such is the case for driving transport vehicles at night. Money motivates drivers to make unwise decisions like driving all night even with fatigue. This is mainly because the drivers are not themselves aware of the huge risk associated with driving when fatigued. Some countries have imposed restrictions on the number of hours a driver can drive at a stretch, but it is still not enough to solve this problem is its implementation.

Chapter2

Literature Survey

SYSTEM REVIEW

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

AccordingtotheSurveyonDriverFatigue-DrowsinessDetectionSystem,thedetection system includes the processes of face image extraction, yawning tendency,blink of eyes detection,eye are a extraction etc .There are many experiments done with OpenCV for android also which is available for cheap smart phones as well. Other experiments conducted have resulted in utmost accuracy when camera was placed at different locations. OpenCV is predominantly a technique for real time image processing which has free of cost implementations on latest computervision algorithms . It has all required computervision algorithms.

There are many experiments done with OpenCV for android also which is available for cheap smartphones as well. Other experiments conducted have resulted in utmost accuracy when camera was placed at different locations. OpenCV is predominantly a technique for real time image processing which has free of cost implementations on latest computer vision algorithms. It has all required computer vision algorithms.

2.1.1 Face and Eye Detection by CNN Algorithms

In this paper a novel approach to critical parts of face detection problems is given, based on analogic cellular neural network (CNN) algorithms. The proposed CNN algorithms find and help to normalize human faces effectively while cause for most accident related to the vehicle’s crashes. Driver fatigue their time requirement is a fraction of the previously used methods. The algorithm starts with the detection of heads on color pictures using deviations in color and structure of the human face and that of the background. By normalizing the distance and position of the reference points, all faces should be transformed into the same size and position. For normalization, eyes serve as points of reference. Other CNN International Journal of Computer Applications (0975 – 8887) Volume 181 – No. 25, November- 2018 39 algorithm finds the eyes on any grayscale image by searching characteristic features of the eyes and eye sockets. Tests made on a standard database show that the algorithm works very fast and it is reliable.

2.1.2 Face Detection using Haar Cascades

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

2.1.3 Eye Detection Using Morphological and Color Image Processing

Eye detection is required in many applications like eye-gaze tracking, iris detection, video conferencing, auto-stereoscopic displays, face detection and face recognition. This paper proposes a novel technique for eye detection using color and morphological image processing. It is observed that eye regions in an image are characterized by low illumination, high density edges and high contrast as compared to other parts of the face. The method proposed is based on assumption that a frontal face image (full frontal) is available. Firstly, the skin region is detected using a color based training algorithm and six-sigma technique operated on RGB, HSV and NTSC scales. Further analysis involves morphological processing using boundary region detection and detection of light source reflection by an eye, commonly known as an eye dot. This gives a finite number of eye candidates from which noise is subsequently removed. This technique is found to be highly efficient and accurate for detecting eyes in frontal face images.

2.1.4 Algorithm for Eye Detection on Grey Intensity Face

This paper presents a robust eye detection algorithm for grey intensity images. The idea of our method is to combine the respective advantages of two existing techniques, feature based method and template-based method, and to overcome their shortcomings. Firstly, after the location of face region is detected, a feature-based method will be used to detect two rough regions of both eyes on the face. Then an accurate detection of iris centers will be continued by applying a template-based method in these two rough regions. Results of experiments to the faces without spectacles show that the proposed approach is not only robust but also quite efficient.

2.1.5 Real-Time Face Detection Using EdgeOrientation Matching

In this paper we describe our ongoing work on real-time face detection in grey level images using edge orientation information. We will show that edge orientation is a powerful local image feature to model objects like faces for detection purposes. We will present a simple and efficient method for template matching and object modelling based solely on edge orientation information. We also show how to obtain an optimal face model in the edge orientation domain from a set of training images. Unlike many approaches that model the grey level appearance of the face our approach is computationally very fast. It takes less than 0.08 seconds on a Pentium II 500MHz for a 320x240 image to be processed using a multi-resolution search with six resolution levels. We demonstrate the capability of our detection method on an image database of 17000 images taken from more than 2900 different people. The variations in head size, lighting and background are considerable. The obtained detection rate is more than 93% on that database.

TECHNOLOGY USED

a. 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.

b. JUPYTER LAB- Project Jupyter is a non-profit organization created to develop open-source software, open-standards, and services for interactive computing across dozens of programming languages.

c. IMAGE PROCESSING - In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images(Done usingOpenCV).

d. OpenCV - OpenCV is an opensource 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.

e. MACHINE LEARNING - Machine learning is the scientific study of algorithms and statistical models that computer systems use in order to perform a specific task effectively without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly told.

TOOLS AND IMAGE PROCESSING LIBRARIES

Following optimized tools and image processing libraries are used by author for implementation of presented algorithm. Open CV:OpenCV (Open-source Computer Vision) is the Swiss Army knife of computer vision.

It has a wide range of modules that can help us with a lot of computer vision problems. But perhaps the most useful part of OpenCV is its architecture and memory management. It provides you with a framework in which you can work with images and video in any way you want, using OpenCV's algorithms or your own, without worrying about allocating and reallocating memory for your images. Open CV libraries and functions are highly optimized and can be used for real time image and video processing. OPENCV's highly optimized image processing function are used by author for real time image processing of live video feed from camera.

DLib: Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems. It is used in both industry and academia in a wide range of domains including robotics, embedded devices, mobile phones, and large high performance computing environments. Dlib's open source licensing allows you to use it in any application, free of charge. Open Source Dib library is used by author for implementation of CNN(Neural Networks). Highly optimized Pre-learned facial shape predictor and detectors functions are used by author for detection of facial landmarks .Facial landmarks were further used for extracting eye

Requirement Analysis

- Python : Python is the basis of the program that we wrote. It utilizes many of the python libraries.
- Libraries :
  - Numpy: Pre-requisite for Dlib
  - Scipy: Used for calculating Euclidean distance between the eyelids.
  - Playsound: Used for sounding the alarm
  - Dlib: This program is used to find the frontal human face and estimate its pose using 68 face landmarks.
  - Imutils: Convenient functions written for Opencv.
  - Opencv: Used to get the video stream from the webcam, etc.
- OS : Program is tested on Windows 10 build 1903 and PopOS
- Laptop : Used to run our code.
- Webcam: Used to get the video feed.

Functional Requirements

A Functional prerequisite is described as one portion or an element of a product , in the entire methodology of programming building. A capacity or part is also depicted as the lead of a section and its yields, given a great deal of data sources. A useful prerequisite may be the figuring identified with specialized and subtleties or data control and getting ready or whatever other express usefulness that describes the target of a particular structure uses the useful necessities are found being utilized cases.

a. Recording the driver's steering behavior the moment the trip begins.

b. Then recognizes changes over the course of long trips

c. Determines the driver's level of fatigue.

Non- Functional Requirements

a. Image processing is done using the captured video.

b. Image is stored in a library called OpenCV.

c. Stored image undergo various algorithm and detects if the driver is fatigue and if fatigue raises an alarm.

CHAPTER 3
WORKING OF PROJECT

This section details the proposed approach to detect driver’s drowsiness that works on two levels .The process starts with capturing of live images from camera and is subsequently sent at local server. At the server’s side, Dlib library is employed to detect facial landmarks and a threshold value is used to detect whether driver is drowsy or not. These facial landmarks are then used to compute the EAR (Eye Aspect Ratio) and are returned back to the driver. In our context, the EAR value received at the application’s end would be compared with the threshold value taken as 0.25. If the EAR value is less than the threshold value, then this would indicate a state of fatigue. In case of Drowsiness, the driver and the passengers would be alerted by an alarm. The subsequent section details the working of each module.

1. Data Procurement

For using the application, the driver performs a registration if using the application for the first time. After performing a sign-up, the driver adds a ride by entering the source and destination of the ride. Likewise, an interface for the passengers is also provided where the passengers can connect with the ride, added by the driver. The driver then starts the ride. The proposed application then captures the real-time images of the driver. Images are captured every time the application receives a response from the server. The process goes on until the driver stops the ride. For testing the efficiency of the proposed approach, a data set of 50 volunteers was collected. Every participant was asked to blink their eyes intermittently while looking at camera for capturing EAR values. The logs of the results that were captured by the application were collected and analysed with the help of machine learning classifiers.

2. Facial Landmark Marking

To extract the facial landmarks of drivers, Dlib library was imported and deployed in our application. The library uses a pre-trained face detector, which is based on a modification to the histogram of oriented gradients and uses linear SVM (support vector machine) method for object detection. Actual facial landmark predictor was then initialized and facial landmarks captured by the application were used to calculate distance between points. These distances were used to compute EAR value. EAR is defined as the ratio of height and width of the eye and was computed using following equation. The numerator denotes the height of the eye and the denominator denotes the width of the eye and the details of the all the landmarks of eye are depicted by figure 1.

$$EAR = \frac{(|p2 - p6| + |p3 - p5|)}{2*|p1-p4|}$$

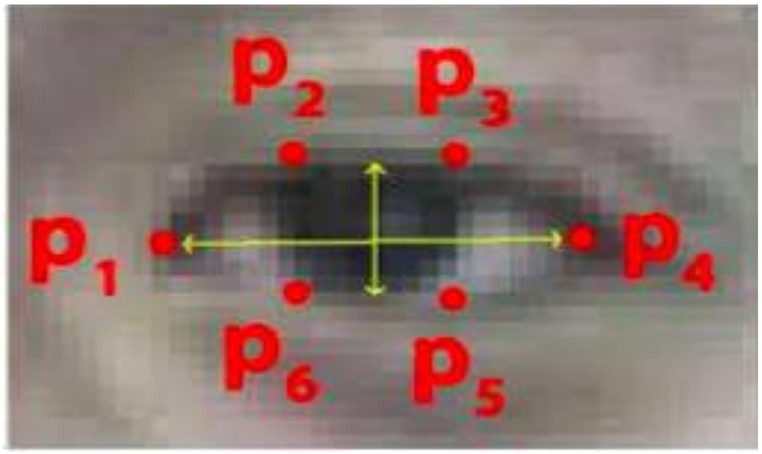


Fig 1. Landmarks of Eye in EAR

Referring equation, the numerator calculates the distance between the upper eyelid and the lower eyelid. The denominator represents the horizontal distance of the eye. When the eyes are open, the numerator value increases, thus increasing the EAR value, and when the eyes are closed the numerator value decreases, thus decreasing the EAR value. In this context, EAR values are used to detect driver’s drowsiness. EAR value of left and right eyes is calculated and then average is taken. In our drowsiness detector case, the Eye Aspect Ratiois monitored to check if the value falls below threshold value and also it does not increase again above the threshold value in the next frame.

The above condition implies that the person has closed his/her eyes and is in a drowsy state. On the contrary, if the EAR value increases again, it implies that the person has just blinked the eye and there is no case of drowsiness. Figure 2 depicts the block diagram of our proposed approach to detect driver's drowsiness. Figure 3 represents a snapshot of facial landmark points using Dlib library, which are used to compute EAR. Table 1 details the facial landmark points for left and right eye which were used for computation.

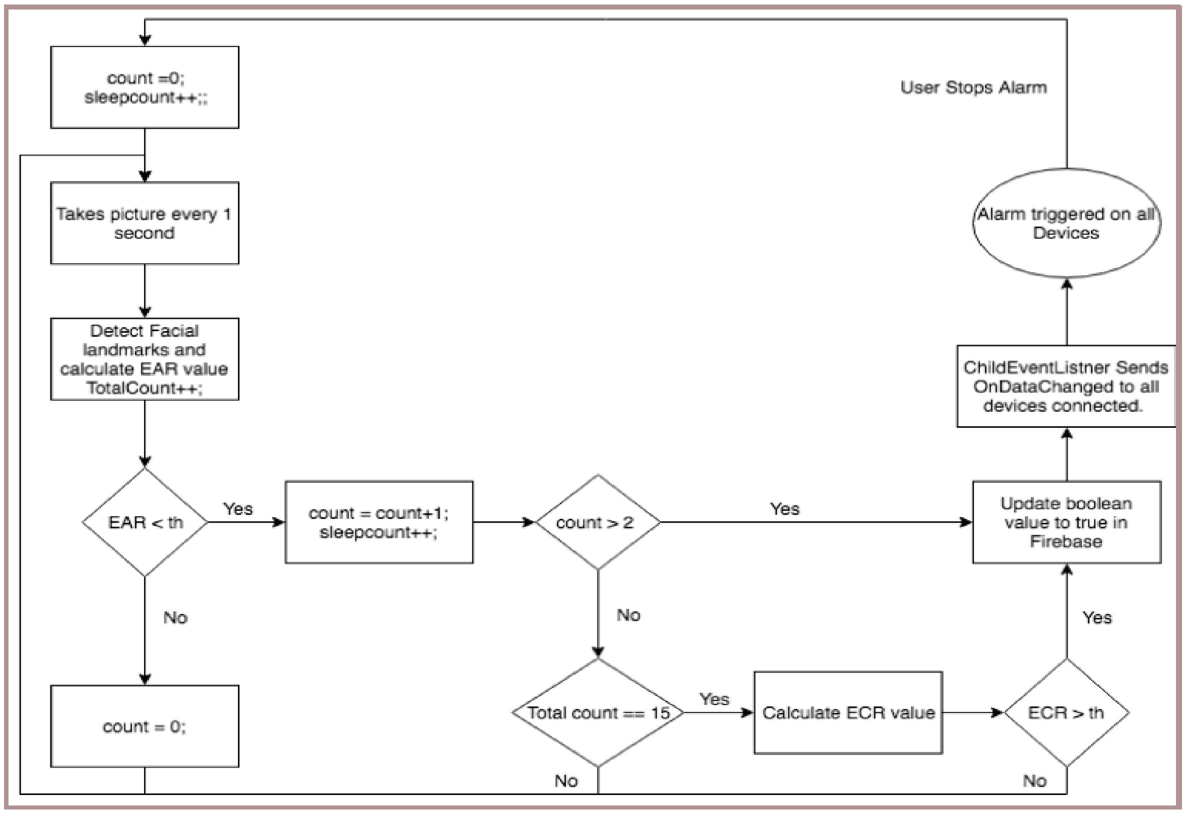

count =0;
sleepcount++;;
User Stops Alarm
Takes picture every 1 second
Alarm triggered on all Devices
Detect Facial landmarks and calculate EAR value TotalCount++;
ChildEventListner Sends OnDataChanged to all devices connected.
EAR < th
Yes
count = count+1; sleepcount++;
count > 2
Yes
Update boolean value to true in Firebase
No
No
Yes
count = 0;
Total count == 15
Yes
Calculate ECR value
ECR > th
No
No

Fig. 2 - Block Diagram of Proposed Drowsiness Detection Algorithm

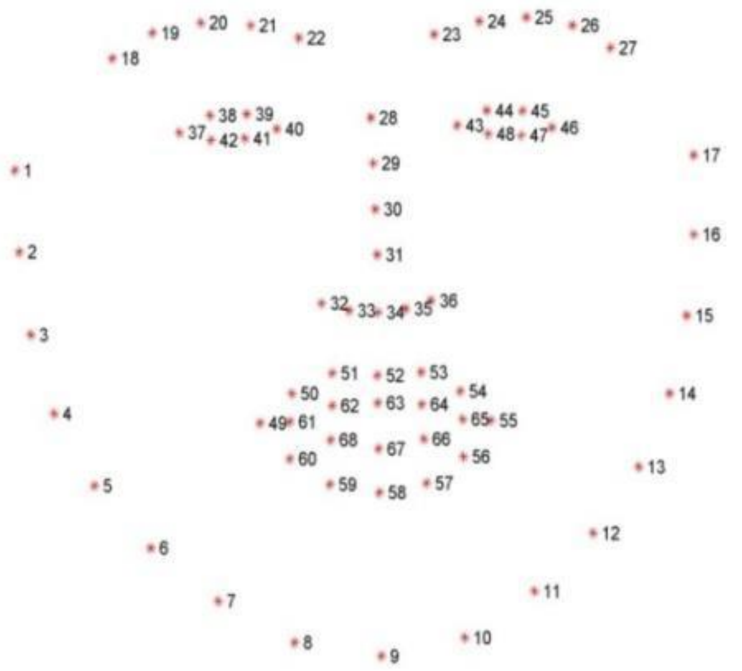


Fig. 3 –Facial Landmark Points according to Dlib library

3. Classification

After capturing the facial landmark points, EAR value computed by the server is now received at the android device of the driver and compared with the threshold value which was earlier set to be 0.25. If the value is less than the threshold then the counter value is incremented, else the counter value is set back to zero. If the counter value reaches to three, an alarm is triggered in the android device. In addition, another variable (Sleep Counter) is maintained which counts the number of times the EAR value is less than threshold value. Variable (Total Counter) stores the total count of responses from the server side and is used to calculate the ECR (Eye Closure Ratio). It is defined as the ratio of Sleep Counter and Total Counter value and was computed using equation.

$$\text{ECR} = SleepCounter/TotalCounter$$

In our context, the value of ECR was calculated for every 15 consecutive frames (captured from camera). As soon as the frame number reaches to 16, the value of total counter becomes one and sleep counter becomes zero. Whenever the ECR value exceeds the threshold value which is set to 0.5, then a Siren is buzzed to indicate drowsy state of the driver.

CHAPTER 4
SYSTEM DESIGN

This projected is designed to achieve certain goals which are:

- 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.
- The system works in spite of driver wearing spectacles and in various lighting conditions.
- To alert the driver on the detection of drowsiness by using buzzer or alarm.
- Speed of the vehicle can be reduced.
- Traffic management can be maintained by reducing the accidents.

The drowsiness detection and correction system developed is capable of detecting drowsiness in a rapid manner. The system which can differentiate normal eye blink and drowsiness which can prevent the driver from entering the state of sleepiness while driving. The system works well even in case of drivers wearing spectacles and under low light conditions also. During the monitoring, the system is able to decide if the eyes are opened or closed. When the eyes have been closed for about two seconds, the alarm beeps to alert the driver and the speed of the vehicle is reduced. By doing this many accidents will reduced and provides safe life to the driver and vehicle safety. A system for driver safety and car security is presented only in the luxurious costly cars. Using drowsiness detection system, driver safety can be implemented in normal cars also. Once the eye aspect ratio calculated, algorithm can threshold it to determine if a person is blinking the eye aspect ratio will remain approximately constant when the eyes are open and then will rapidly approach zero during a blink, then increase again as the eye opens. The duration of blink further provide estimation of microsleep . The proposed algorithm has been tested on personal car driver for testing purposes. For authentic results, the camera position was focused on the driver’s face. Further, the algorithm has been tested in day time driving and Night time using IR camera. The results are discussed in Result section and found satisfactory. The proposed algorithm focused solely on using the eye aspect ratio as a quantitative metric to determine if a person has blinked in a video.

Testing

Software testing is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools.

Importance of Testing

1.Software testing is really required to point out the defects and errors that were made during the development phases. Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

2. It’s essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained.

If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization.
Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, if proper software testing may also prevent monetary losses.

3.It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. (Know more about Software Quality) As explained in the previous point, delivering good quality product on time builds the customers confidence in the team and the organization.

4.Testing is necessary in order to provide the facilities to the customers like the delivery of the high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results. High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.

5.Testing is required for an effective performance of software application or product.

6.It’s important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development. Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application. If defects related to requirements or design are detected late in the life cyle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.

7.It’s required to stay in the business.Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application. In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

SYSTEM ARCHITECTURE

3.1 USECASEDIAGRAM

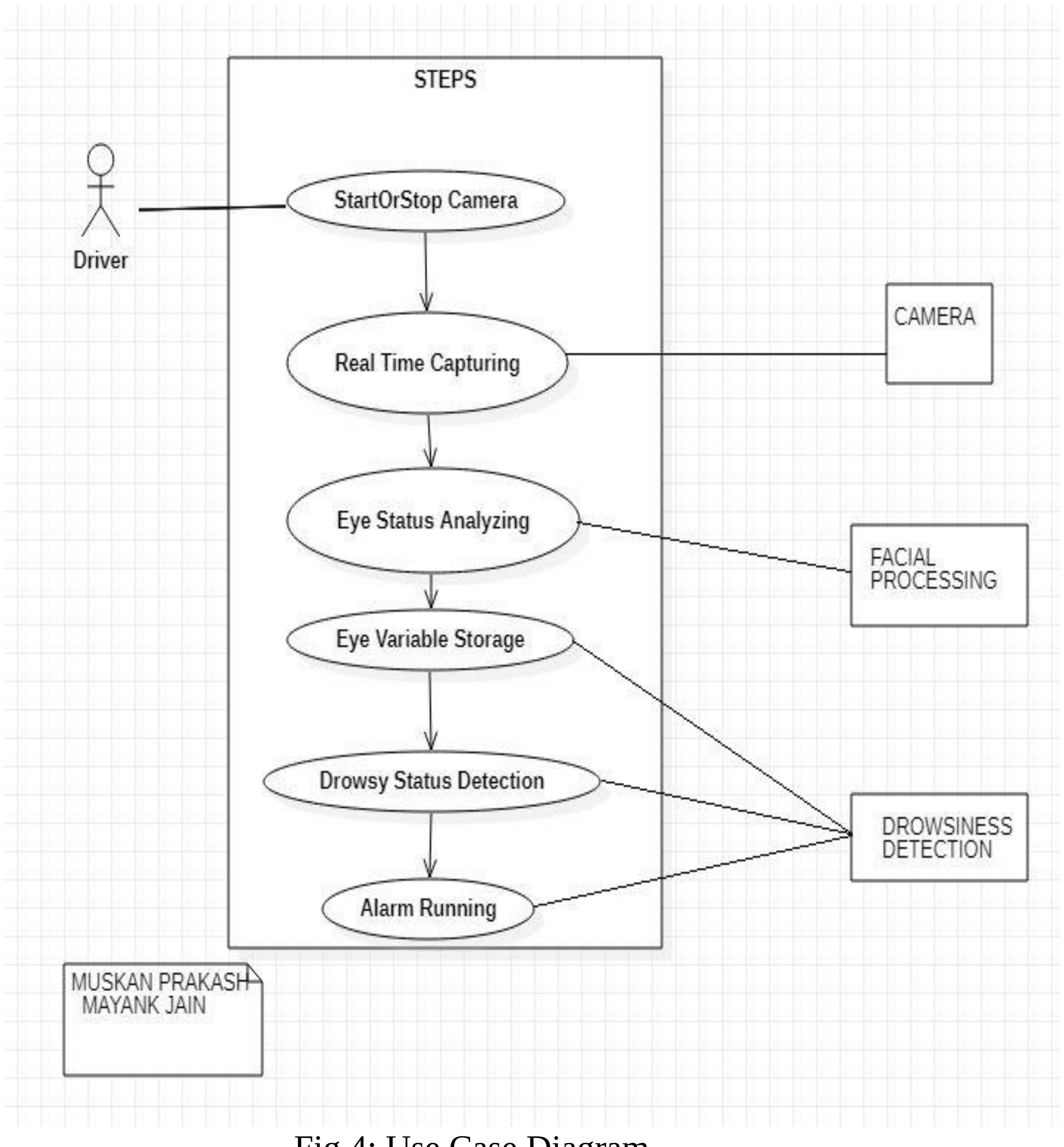


Fig 4: Use Case Diagram

ACTIVITYDIAGRAM

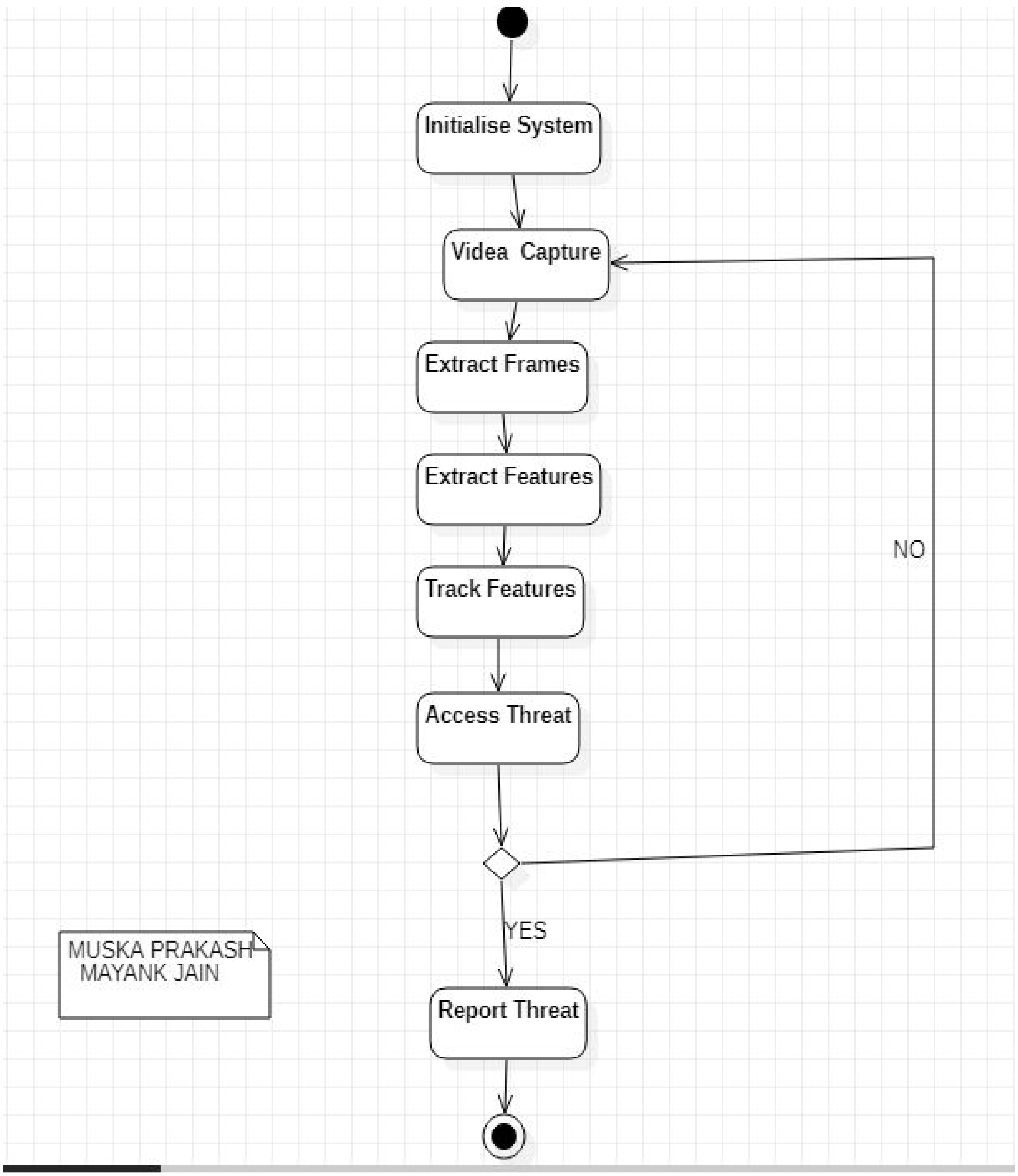


fig 5: Activity Diagram

CLASSDIAGRAM

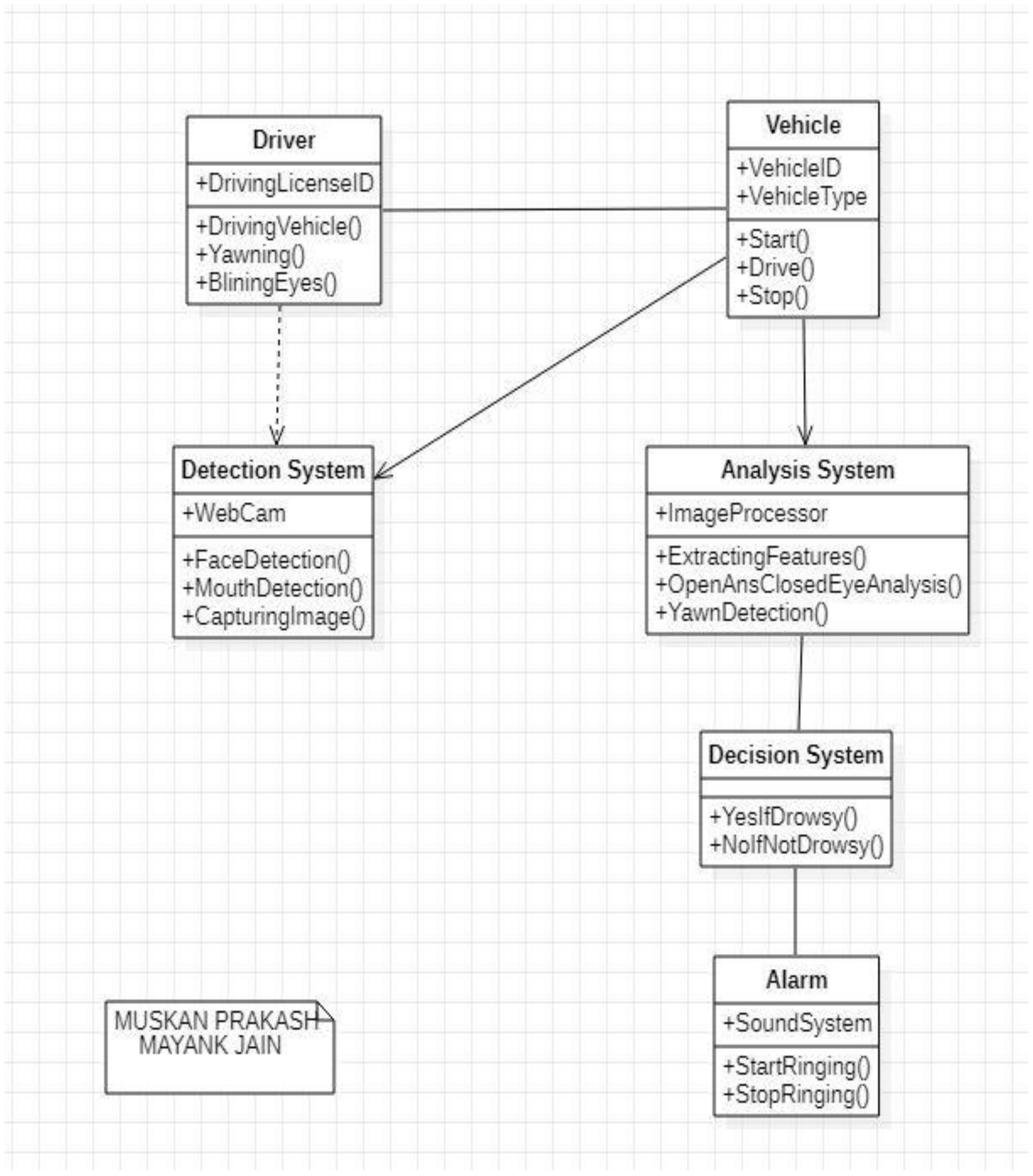
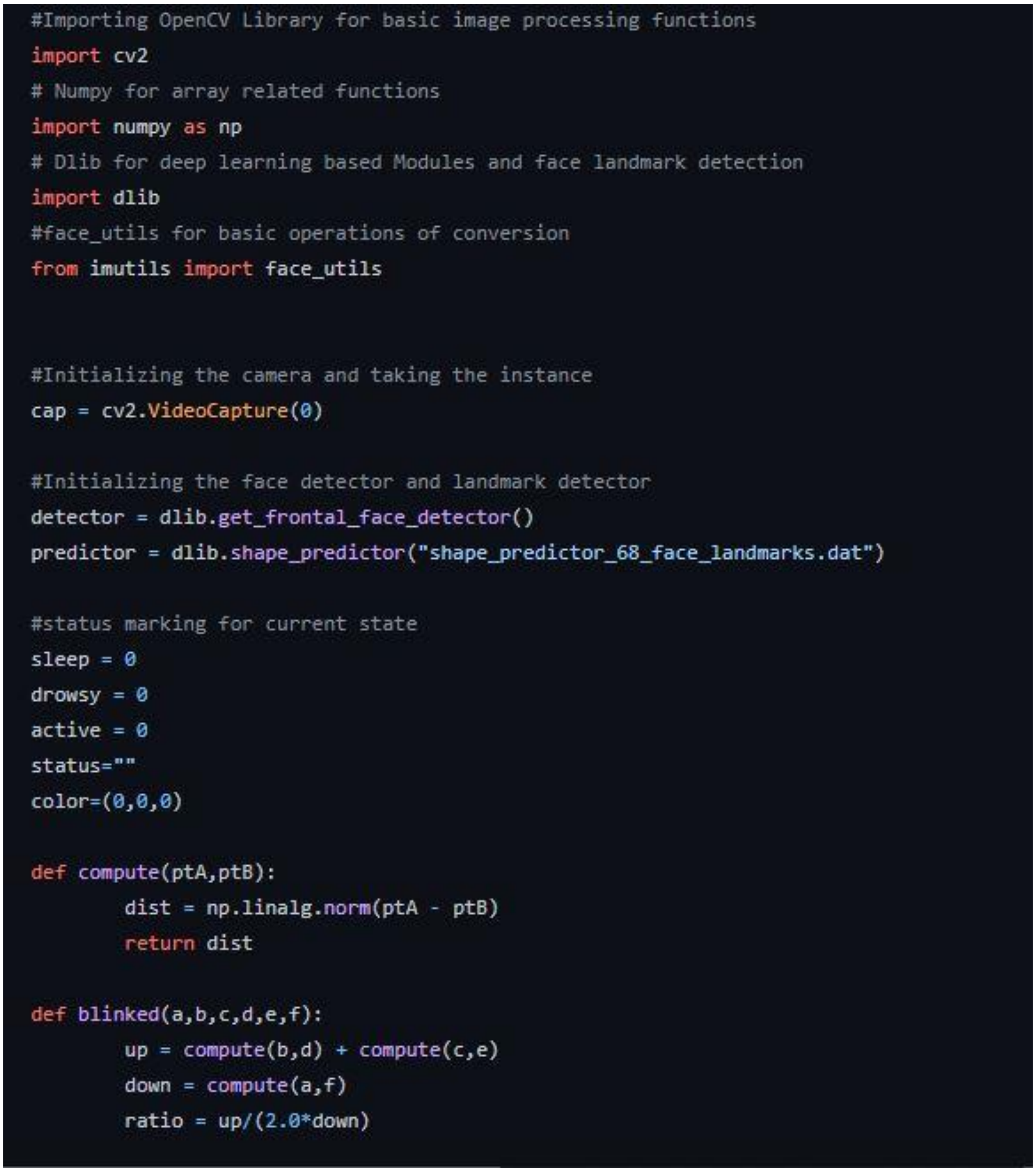


Fig 6: Class Diagram

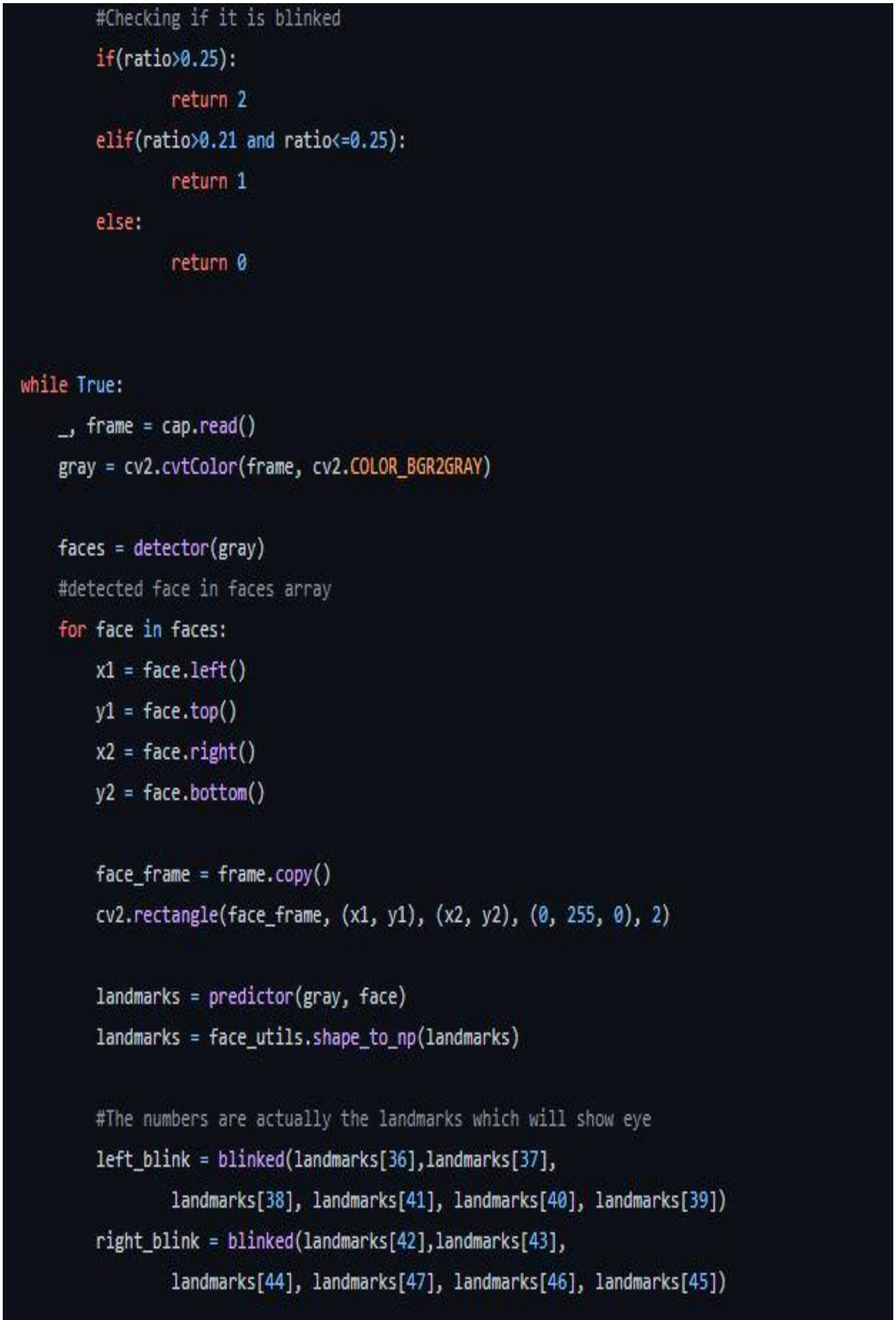
EXPECTED OUTCOME

We have used Open CV as a platform to develop a code for eye detection in real time. The code is then implemented on system installed with Open CV software. To detect human eyes, face has to be detected initially. This is done by OpenCV face haar cascade classifier. Once the face is detected, the location of the eyes is estimated and eye detection is done using eye Haar-cascade classifier. Hence using the open CV, face and eyes are detected accurately and displayed on the monitor as shown in the. The larger yellow square indicates the face while smallerred squares indicate the eyes. Once face and eyes are detected, it is checking status of eyes i.e. open or closed state of the eyes. If both eyes remain closed for successive frames, it indicates that the driver is drowsy and gives the warning signal .A webcam is a video camera that feeds its image in real time to a computer or computer network. Unlike an IP camera (which uses a direct connection using Ethernet or Wi-Fi), a webcam is generally connected by a USB cable, FireWire cable, or similar cable. Their most popular use is the establishment of video links, permitting computers to act as videophones. The common use as a video camera for the World Wide Web gave the webcam its name. Other popular uses include security surveillance, computer vision, video broadcasting, and for recording social videos. A fatigue detection system based on the above method was implemented by using Visual C++. At first, we fix a camera on a car in front of the driver. Then we capture some videos from 8 drivers in normal conditions. The whole input image format is 320×240 and they are in RGB color space. We have also found that the optimum distance from camera which obtained about 30cm-50cm that is very suitable for our method.The average accuracy of our combination method is 90.873%. Thus our eye detection method is robust and irrelevant with different sizes and more accurate. According to obtained results, our system can determine the eye states with a high rate of correct decision.

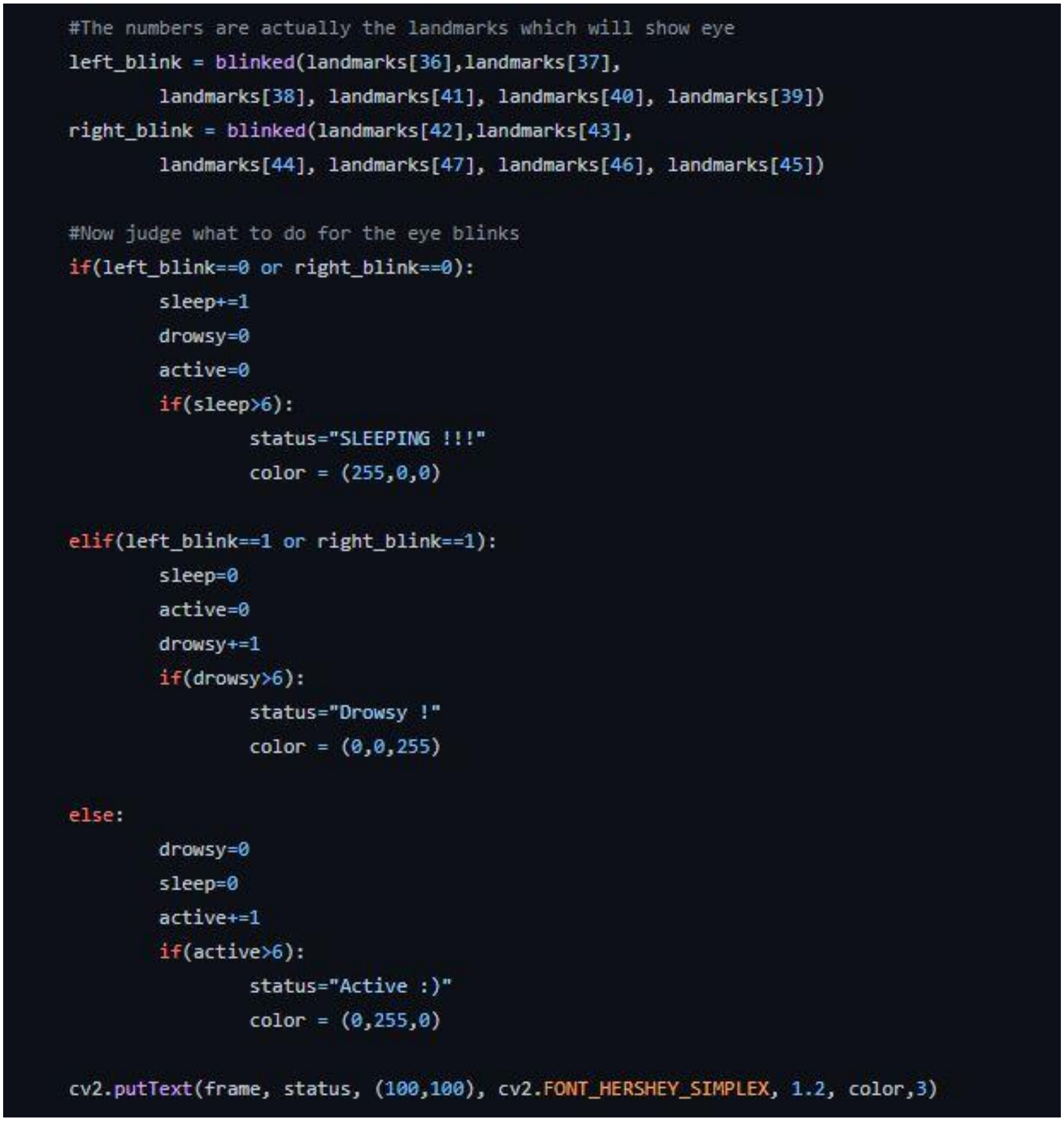
SOURCSE CODE:



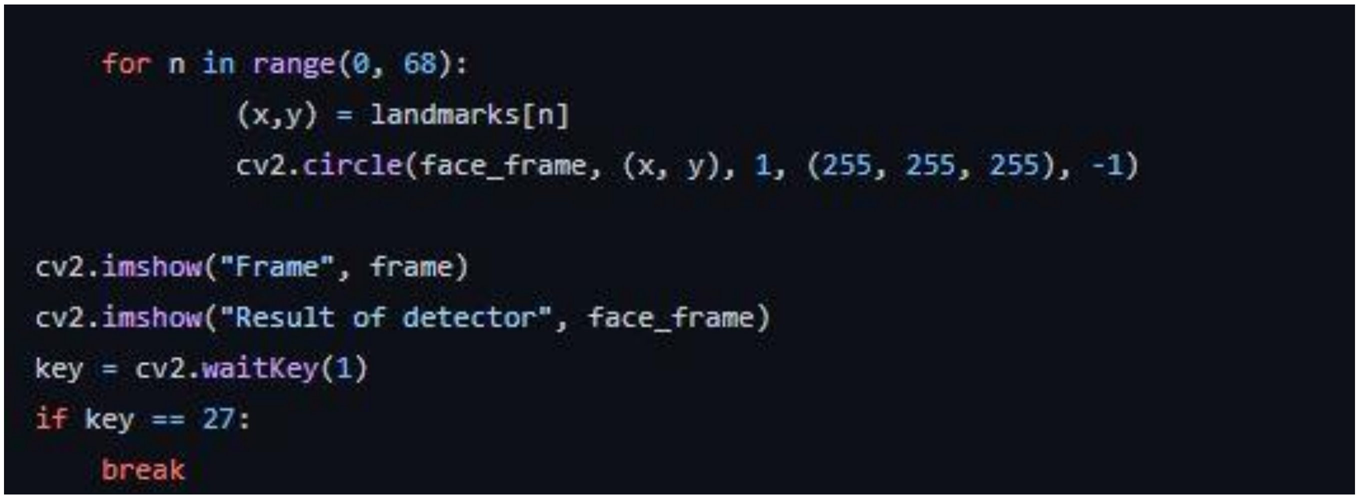
```
#Importing OpenCV Library for basic image processing functions
import cv2
# Numpy for array related functions
import numpy as np
# Dlib for deep learning based Modules and face landmark detection
import dlib
#face_utils for basic operations of conversion
from imutils import face_utils


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

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

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

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

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



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


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

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

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

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

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



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

#Now judge what to do for the eye blinks
if(left_blink==0 or right_blink==0):
        sleep+=1
        drowsy=0
        active=0
        if(sleep>6):
                status="SLEEPING !!!"
                color = (255,0,0)

elif(left_blink==1 or right_blink==1):
        sleep=0
        active=0
        drowsy+=1
        if(drowsy>6):
                status="Drowsy !"
                color = (0,0,255)

else:
        drowsy=0
        sleep=0
        active+=1
        if(active>6):
                status="Active :)"
                color = (0,255,0)

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


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

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

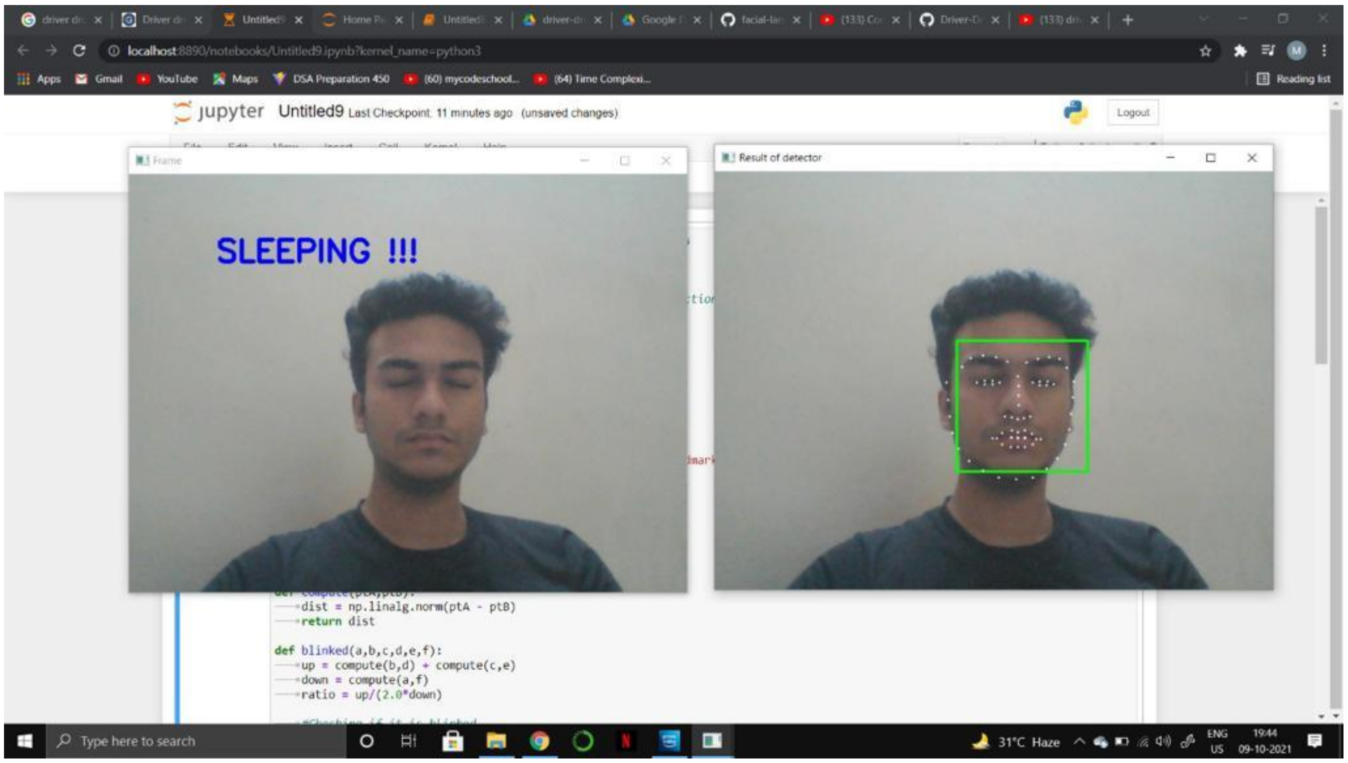
OUTPUT SCREENSHOT

Logic of project

The project includes direct working with the 68 facial landmark detector and also the face detector of the Dlib library. The 68 facial landmark detector is a robustly trained efficient detector which detects the points on the human face using which we determine whether the eyes are open or they are closed.

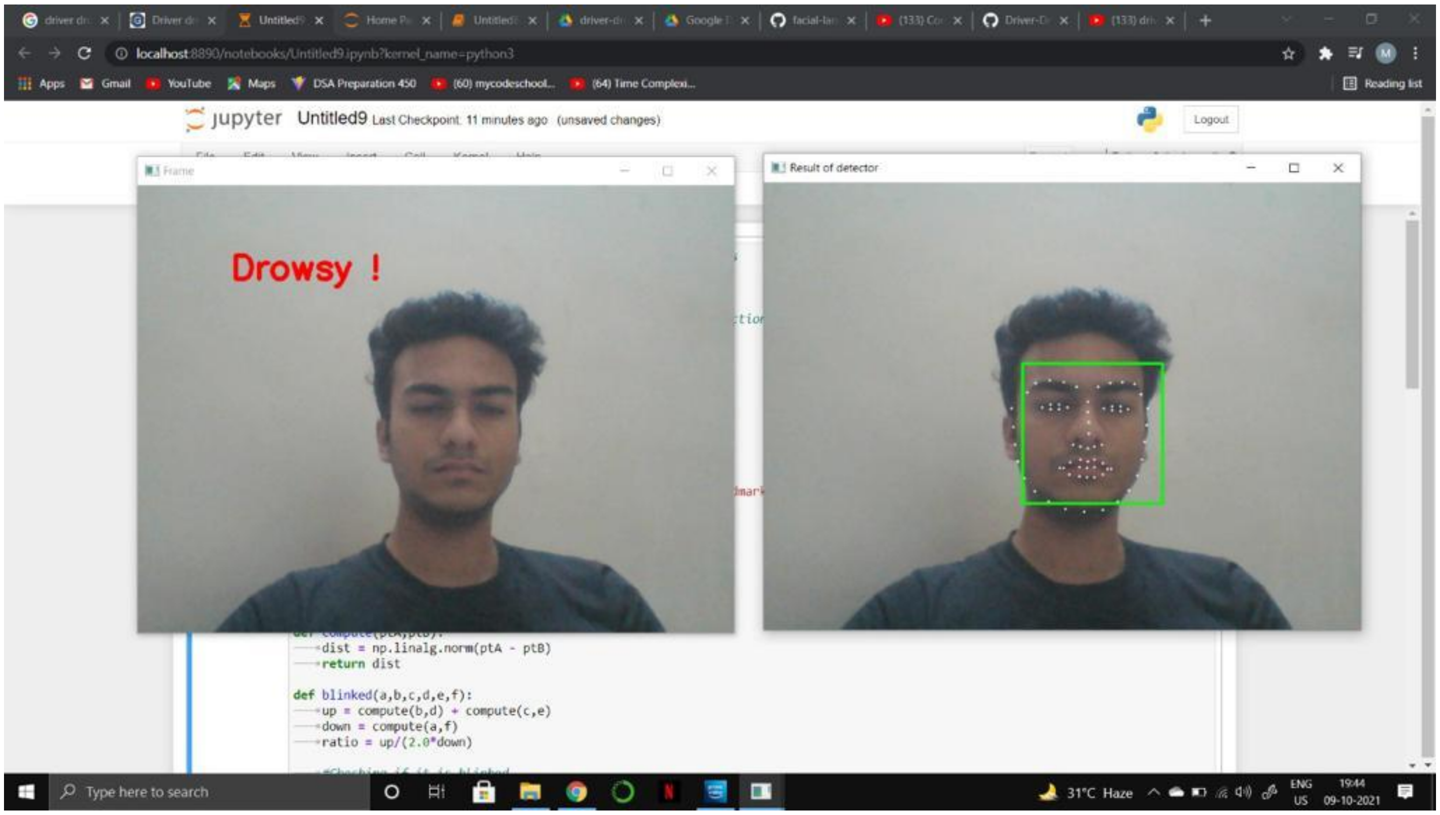
The working of the project

As you can see the above screenshot where the landmarks are detected using the detector.
Now we are taking the ratio which is described as 'Sum of distances of vertical landmarks divided by twice the distance between horizontal landmarks'.
Now this ratio is totally dependent on your system which you may configure accordingly for the thresholds of sleeping, drowsy, active.


jupyter Untitled9 Last Checkpoint: 11 minutes ago (unsaved changes)
Logout
Frame
Result of detector
SLEEPING !!!

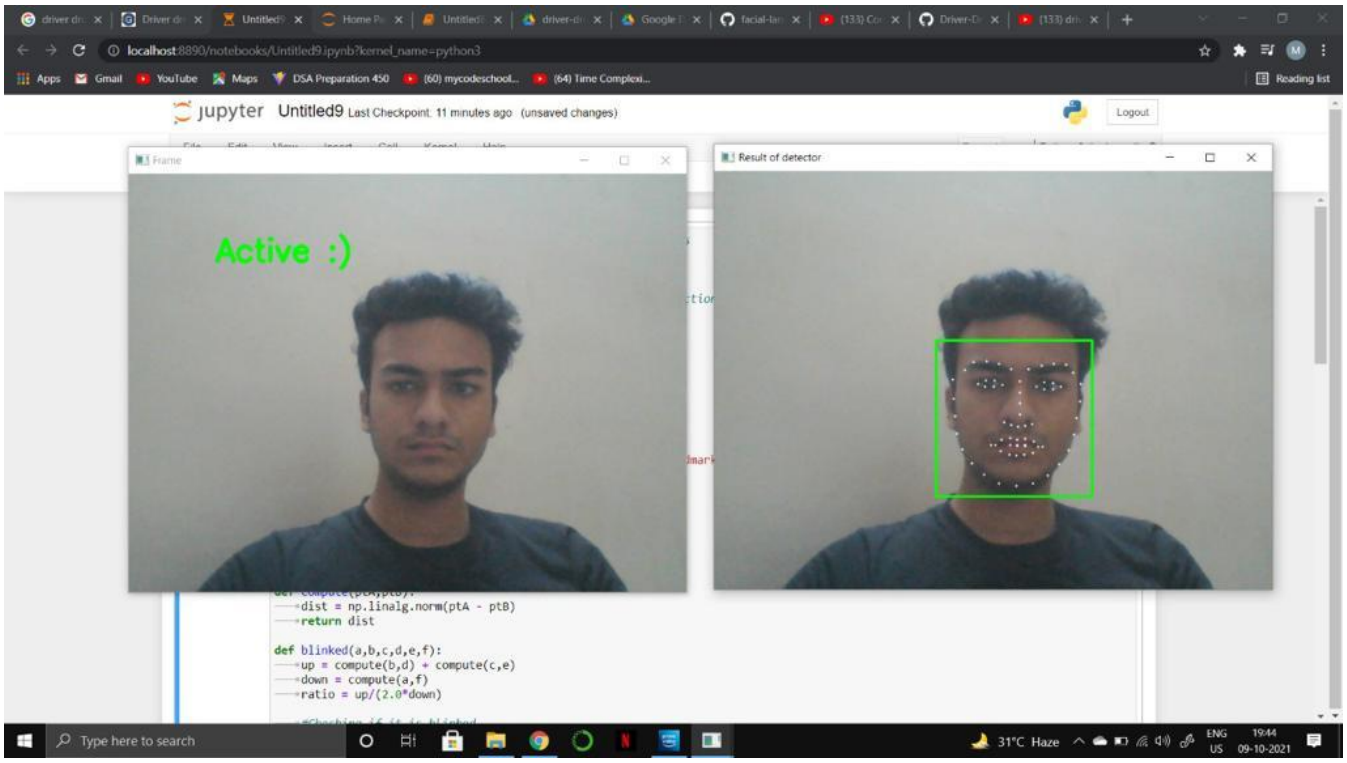
```
    dist = np.linalg.norm(ptA - ptB)
    return dist

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

Fig8. Sleeping


Jupyter Untitled9 Last Checkpoint: 11 minutes ago (unsaved changes)
Logout
Frame
Result of detector
Drowsy !

```
    dist = np.linalg.norm(ptA - ptB)
    return dist

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

Fig9. Drowsiness detected


Active :)
Frame
Result of detector
jupyter Untitled9 Last Checkpoint: 11 minutes ago (unsaved changes)
Logout

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

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

    #Checking if it is blinked
```

Fig 10. Active driver

CHAPTER 5
RESULT & DISCUSSIONS

In present paper, author demonstrated how to build a blink and drowsiness detector using OpenCV, Python, and Dlib open source Libraries by measuring EAR. The first step in building a blink detector is to perform facial landmark detection to localize the eyes in a given frame from a video stream. The eye aspect ratio for each eye can be calculated using Euclidian distance functions of OPEN CV , which is a singular value, relating the distances between the vertical eye landmark points to the distances between the horizontal landmark points. Once the eye aspect ratio calculated, algorithm can threshold it to determine if a person is blinking the eye aspect ratio will remain approximately constant when the eyes are open and then will rapidly approach zero during a blink, then increase again as the eye opens. The duration of blink further provide estimation of microsleep. The proposed algorithm has been tested on personal car driver for testing purposes. For authentic results, the camera position was focused on the driver's face. Further, the algorithm has been tested in day time driving and Night time using IR camera. The results are discussed in Result section and found satisfactory. The proposed algorithm focused solely on using the eye aspect ratio as a quantitative metric to determine if a person has blinked in a video stream. However, due to noise in a video stream, subpar facial landmark detections, or fast changes in viewing angle, a simple threshold on the eye aspect ratio could produce a false-positive detection, reporting that a blink had taken place when in reality the person had not blinked. To make our blink detector more robust to these challenges further following improvements can be implemented Computing the eye aspect ratio for the Nth frame, along with the eye aspect ratios for N – 6 and N + 6 frames, then concatenating these eye aspect ratios to form a 13 dimensional feature vector. Training a Support Vector Machine (SVM) on these feature vectors. The combination of the temporal-based feature vector and SVM classifier helps reduce false-positive blink detections and improves the overall accuracy of the blink detector. Two-way analysis has been performed in our work. Our first phase includes the results obtained by the android application when the driver faces the camera. Data is collected from this phase, is further used in the second phase where detailed analysis of the results has been performed using machine learning classifiers to test the effectiveness of the proposed approach. Classifiers that were employed for empirical analysis were Naive Bayes, Support Vector Machine and Random Forest(K. Das and R. N. Behera,2017). To evaluate the performance of the classifiers, we compared the results obtained based on standard performance metrics. Naive Bayes Classifier is used to identify objects by applying Bayes Algorithm. Random Forest Classifier is an ensemble algorithm which generates a set of uncorrelated decision trees by randomly selecting the subset of training set and then aggregates them to arrive at a conclusion. SVM (Support Vector Machine) is a discriminative classifier that finds out a line that demarcates the classes. Table 2 enumerates the results obtained by employing different classifiers.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE

6.1 **CONCLUSION**

Research on drowsy driving detection algorithm is one of the most important methods to reduce traffic accidents. As we know, there are significant individual differences, especially the size of eyes, between different person. It is crucial to take the individual differences into consideration when study on the algorithm based on computer vision.

It completely meets the objectives and requirements of the system. The framework has achieved an unfaltering state where all the bugs have been disposed of. The framework cognizant clients who are familiar with the framework and comprehend its focal points and the fact that it takes care of the issue of stressing out for individuals having fatigue-related issues to inform them about the drowsiness level while driving.

In this work, a real time system that monitors and detects the loss of attention of drivers of vehicles is proposed. The face of the driver has been detected by capturing facial landmarks and warning is given to the driver to avoid real time crashes. Non-intrusive methods have been preferred over intrusive methods to prevent the driver from being distracted due to the sensors attached on his body. The proposed approach uses Eye Aspect Ratio and Eye Closure Ratio with adaptive thresholding to detect driver's drowsiness in real-time. This is useful in situations when the drivers are used to strenuous workload and drive continuously for long distances. The proposed system works with the collected data sets under different conditions.

6.2 **FUTURE SCOPE**

The model can be improved incrementally by using other parameters like blink rate, yawning, state of the car, etc. If all these parameters are used it can improve the accuracy by a lot. We plan to further work on the project by adding a sensor to track the heart rate in order to prevent accidents caused due to sudden heart attacks to drivers.

Same model and techniques can be used for various other uses like Netflix and other streaming services can detect when the user is asleep and stop the video accordingly. It can also be used in application that prevents user from sleeping.

The future work can include integration of the proposed system with globally used applications like Uber and Ola. The system, if integrated, can reduce the number of casualties and injuries that happen regularly due to these drowsy states of the drivers. This experiment can run as a part of pilot plan i.e., for a few days/months in different regions of the world where such incidents occur regularly. Thus, our proposed approach also gives the same accuracy for the people wearing spectacles. Accuracy of our proposed system improves with the increase in brightness of the surrounding environment. The work can be extended for different types users such as bike riders or in different domains like railways, airlines etc.

REFRENCES

1. “Road Accidents in India 2016,” 2016.

2. S. Sangle, B. Rathore, R. Rathod, A. Yadav, and A. Yadav, “Real Time Drowsiness Detection System,” pp. 87–92, 2018.

3. V. Varghese, A. Shenoy, S. Ks, and K. P. Remya, “Ear Based Driver Drowsiness Detection System,” vol. 2018, pp. 93–96, 2018.

4. A. Kumar and R. Patra, “Driver drowsiness monitoring system using visual behaviour and machine learning,” *ISCAIE 2018 - 2018 IEEE Symposium on Computer Applications and Industrial Electronics*, pp. 339–344, 2018.

5. *T. Hwang, M. Kim, S. Hong, and K. S. Park, “Driver drowsiness detection using the in-ear EEG,” Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS, vol. 2016–Octob, pp. 4646–4649, 2016.*

6. S. Junawane, S. Jagtap, P. Deshpande, and L. Soni, “Driver Drowsiness Detection Techniques: A Survey,” vol. 6, no. 11, pp. 2015–2017, 2017.

7. R. Jabbar, K. Al-Khalifa, M. Kharbeche, W. Alhajyaseen, M. Jafari, and S. Jiang, “Real-time Driver Drowsiness Detection for Android Application Using Deep Neural Networks Techniques,” *Procedia Computer Science*, vol. 130, pp. 400–407, 2018.

8. T. Soukupova and J. Cech, “Real-time eye blink detection using facial landmarks,” *Computer Vision Winter Workshop (CVWW)*, 2016.

9. I. García, S. Bronte, L. M. Bergasa, J. Almazán, and J. Yebes, “Vision-based drowsiness detector for real driving conditions,” *IEEE Intelligent Vehicles Symposium, Proceedings*, pp. 618–623, 2012.

10. S. S. Nagargoje and D. S. Shilvant, “Drowsiness Detection System for Car Assisted Driver Using Image Processing,” *International Journal of Electrical and Electronics Research ISSN*, vol. 3, no. 4, pp. 175–179, 2015.