A
MAJOR PROJECT REPORT ON DRIVER'S DROWSINESSDETECTION

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

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING

P.VAISHNAV - 16P61A05J9

T.VIJAY KUMAR - 16P61A05K1

K.SRIKANTH - 17P65A0505

T.PRATYUSHA - 16P61A05N2

Under the guidance of

Mrs.Adilakshmi siripireddy

Assistant Professor

VIGNANA BHARATHI INSTITUTE OF TECHNOLOGY

(A UGC Autonomous Institution, Approved by AICTE, Affiliated to JNTUH, Kukatpally Accredited by National Board of Accreditation (NBA), National Assessment and Accreditation Council (NAAC)) Aushapur (V), Ghatkesar (M), Medchal(dist.).

2019 - 2020

VBIT

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Mini Project report titled DRIVER'S DROWSINESS DETECTION is being submitted by **P.Vaishnav** , **T.Vijay kumar ,K.Srikanth**,**T.Pratyusha** bearing roll numbers **16P61A05J9**,**16P61A05K1**,**17P65A0501**,**16P61A05N2** in B.Tech IV I semester in *COMPUTER SCIENCE and Engineering* is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Guide:

Mrs.Adilakshmi siripireddy
Assistant Professor, CSE

Head of the Department

Dr. K. Sreenivasa Rao

External Examinar

Date:

DECLARATION

We, **P.Vaishnav,T.Vijay kumar,K.Srikanth,T.Prathyusha** bearing hall ticket number 16P61A05J9**,** 16P61A05K2, 17P65A0505, 16P61A05N2**,** hereby declare that the project report entitled **"DRIVER'S DROWSINESS DETECTION** under the guidance of **Mrs. Adilakshmi siripireddy,** Department of Computer Science And Engineering, **Vignana Bharathi Institute of Technology**, Hyderabad, is submitted to Jawaharlal Nehru Technological University Hyderabad, Kukatpally, Hyderabad in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science And Engineering.

This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

P.Vaishnav(16P61A05J9)

T.Vijay kumar(16P61A05K1)

K.Srikanth(17P65A0505)

T.Prathyusha(16P61A05N2)

ACKNOWLEDGMENTS

Self-confidence, hard work, commitment and planning are essential to carry-out any task. Possessing these qualities is sheer waste, if an opportunity does not exist. So, we whole-heartedly thank **Mr. Dr. G. AMARENDER RAO**, Principal, and **Dr. K. Sreenivasa Rao**, Head of the Department, Computer Science and Engineering for their encouragement and support and guidance in carrying out the project.

We thank our Project Guide, **Mrs. Adilakshmi siripireddy** for providing us with an excellent project and guiding us in completing our project successfully.

We would also like to thank our project coordinator **Mrs. N. Yamini Devi** for his/her valuable advice and suggestions from time to time and for being a constant source of inspiration to us.

We would also like to express our sincere thanks to all the staff of Computer Science and Engineering, VBIT, for their kind cooperation and timely help during the course of our project. Finally, we would like to thank our parents and friends who have always stood by us whenever we were in need of them.

VBIT

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Mini Project report titled DRIVER'S DROWSINESS DETECTION is being submitted by **P.Vaishnav** , bearing roll number **16P61A05J9** in B.Tech IV I semester in *COMPUTER SCIENCE and Engineering* is a record bonafide work carried out by them. The results embodied in this report have not been submitted to any other University for the award of any degree.

Guide:

Mrs. Mrs.Adilakshmi siripireddy

Assistant Professor, CSE

Head of the Department

Dr. K. Sreenivasa Rao

ABSTRACT

The main idea behind this project is to develop a nonintrusive system which can detect fatigue of any human and can issue a timely warning. Drivers who do not take regular breaks when driving long distances run a high risk of becoming drowsy a state which they often fail to recognize early enough. According to the expert's studies show that around one quarter of all serious motorway accidents are attributable to sleepy drivers in need of a rest, meaning that drowsiness causes more road accidents than drink-driving. This system will monitor the driver eyes using a camera and by developing an algorithm we can detect symptoms of driver fatigue early enough to avoid the person from sleeping. So, this project will be helpful in detecting driver fatigue in advance and will give warning output in form of alarm and popups.

Moreover, the warning will be deactivated manually rather than automatically. For this purpose, a de- activation dialog will be generated which will contain some simple mathematical operation which when answered correctly will dismiss the warning. Moreover, if driver feels drowsy there is possibility of incorrect response to the dialog. We can judge this by plotting a graph in time domain. If all the three input variables show a possibility of fatigue at one moment, then a Warning signal is given in form of text and sound. This will directly give an indication of drowsiness/fatigue which can be further used as record of driver performance.

Keywords- Drowsiness, Supervised Learning, Unsupervised Learning, Machine Learning

.

Table of Contents

List of Figures

1. Introduction

Real Time Drowsiness behaviour which are related to fatigue are in the form of eye closing, head nodding or the brain activity. Hence, we can either measure change in physiological signals, such as brain waves, heart rate and eye blinking to monitor drowsiness or consider physical changes such as sagging posture, leaning of driver's head and open/closed state of eyes.

The former technique, while more accurate, is not realistic since highly sensitive electrodes would have to be attached directly on the driver' body and hence which can be annoying and distracting to the driver. In addition, long time working would result in perspiration on the sensors, diminishing their ability to monitor accurately. The second technique is to measure physical changes (i.e. open/closed eyes to detect fatigue) is well suited for real world conditions since it is non-intrusive by using a video camera to detect changes. In addition, micro sleeps that are short period of sleeps lasting 2 to 3 minutes are good indicators of fatigue. Thus, by continuously monitoring the eyes of the driver one can detect the sleepy state of driver and a timely warning is issued.

2. Review of Literature

In this section, we have discussed various methodologies that have been proposed by researchers for drowsiness detection and blink detection during the recent years.

Manu B.N in 2016, has proposed a method that detect the face using Haar feature-based cascade classifiers. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier that will detect the object. So along with the Haar feature-based classifiers, cascaded AdaBoost classifier is exploited to recognize the face region then the compensated image is segmented into numbers of rectangle areas, at any position and scale within the original image. Due to the difference of facial feature, Haarlike feature is efficient for real-time face detection. These can be calculated according to the difference of sum of pixel values within rectangle area and during the process the Adaboost algorithm will allow all the face samples and it will discard the non-face samples of images.

Amna Rahman in 2015, has proposed a method to detect the drowsiness by using Eye state detection with Eye blinking strategy. In this method first, the image is converted to gray scale and the corners are detected using Harris corner detection algorithm which will detect the corner at both side and at down curve of eye lid. After tracing the points then it will make a straight line between the upper two points and locates the mid-point by calculation of the line, and it connects the mid-point with the lower point. Now for each image it will perform the same procedure and it calculates the distance ‘d’ from the mid-point to the lower point to determine the eye state. Finally, the decision for the eye state is made based on distance ’d’ calculated. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”. They have also invoked intervals or time to know that the person is feeling drowsy or not. This is done by the average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second).

3. System Design

The system architecture of the proposed system is represented as follows,

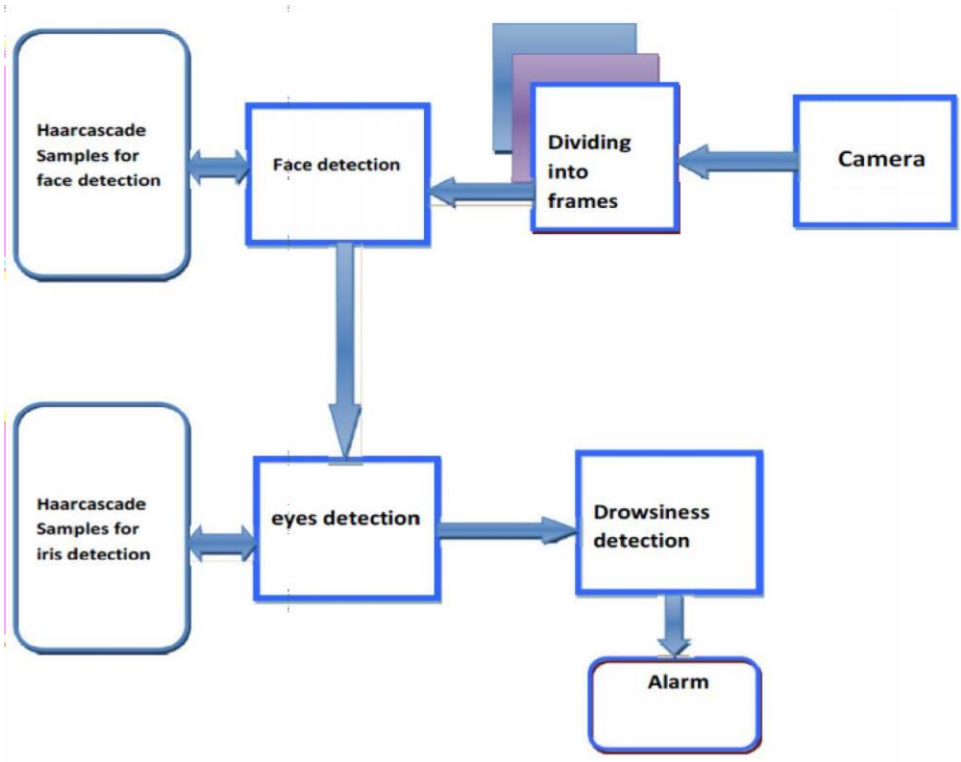


Fig 3.1. Proposed System Architecture

Fig 1. showcases the various important blocks in the proposed system and their high level interaction. It can be seen that the system consists of 5 distinct modules namely,(a) Video acquisition, (b) Dividing into frames, (c) Face detection, (d) Eye detection and (e) Drowsiness detection. In addition to these there are two external typically hardware components namely,

Camera for video acquisition and an audio alarm. The functionality of each these modules in the system can be described as follows:

Video acquisition: Video acquisition mainly involves obtaining the live video feed of the automobile driver. Video acquisition is achieved, by making use of a camera.

Dividing into frames: This module is used to take live video as its input and convert it into a series of frames/ images, which are then processed.

Face detection: The face detection function takes one frame at a time from the frames provided by the frame grabber, and in each and every frame it tries to detect the face of the automobile driver. This is achieved by making use of a set of pre-defined Haarcascade samples.

Eyes detection: Once the face detection function has detected the face of the automobile driver, the eyes detection function tries to detect the automobile driver's eyes. This is achieved by making use of a set of pre-defined Haarcascade samples.

Drowsiness detection: After detecting the eyes of the automobile driver , the drowsiness detection function detects if the automobile driver is drowsy or not , by taking into consideration the state of the eyes , that is , open or closed and the blink rate

As the proposed system makes use of OpenCV libraries, there is no necessary minimum resolution requirement on the camera. The schematic representation of the algorithm of the proposed system is depicted in Fig 2. In the proposed algorithm, first video acquisition is achieved by making use of an external camera placed in front of the automobile driver. The acquired video is then converted into a series of frames/images. The next step is to detect the automobile driver's face , in each and every frame extracted from the video.

As indicated in Figure 2, we start with discussing face detection which has 2 important functions (a) Identifying the region of interest, and (b) Detection of face from the above region using Haarcascade. To avoid processing the entire image, we mark the region of interest. By considering the region of interest it is possible to reduce the amount of processing required and also speeds up the processing, which is the primary goal of the proposed system.

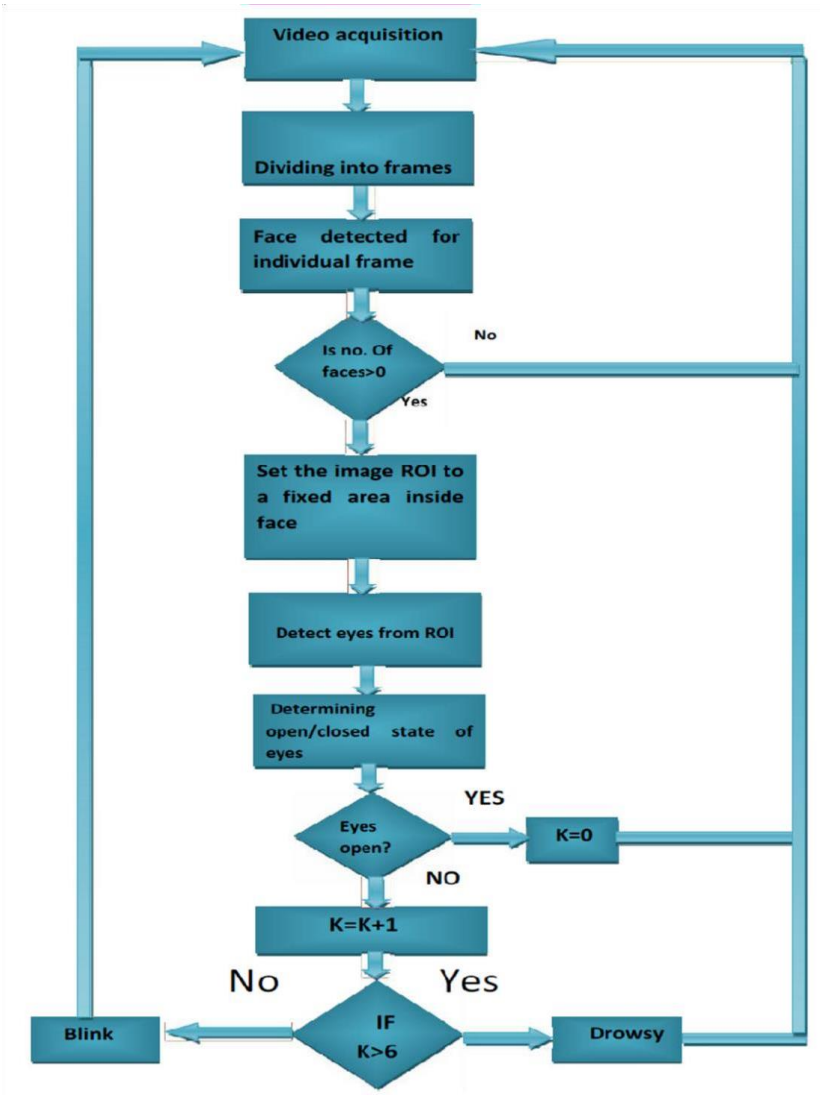


Fig 3.2. High Level System Flow

For detecting the face, since the camera is focused on the automobile driver, we can avoid processing the image at the corners thus reducing a significant amount of processing required. Once the region of interest is defined face has been detected, the region of interest is now the face, as the next step involves detecting eyes. To detect the eyes, instead of processing the entire face region , we mark a region of interest within the face region which further helps in achieving the primary goal of the proposed system. Next we make use of Haarcascade Xml file constructed for eye detection, and detect the eyes by processing only the region of interest . Once the eyes have been detected, the next step is to determine whether the eyes are in open/closed state, which is achieved by extracting and examining the pixel values from the eye region. If the eyes are detected to be open, no action is taken. But, if eyes are detected to be closed continuously for two seconds according to[6] , that is a particular number of frames depending on the frame rate, then it means that the automobile driver is feeling drowsy and a sound alarm is triggered. However, if the closed states of the eyes are not continuous, then it is declared as a blink.

4. Proposed Methodology

4.1) The different types of methodologies have been developed to find out drowsiness.

4.1.1) Physiological level approach: This technique is an intrusive method wherein electrodes are used to obtain pulse rate, heart rate and brain activity information. ECG is used to calculate the variations in heart rate and detect different conditions for drowsiness. The correlation between different signals such as ECG (electrocardiogram), EEG (electroencephalogram), and EMG (electromyogram) are made and then the output is generated whether the person is drowsy or not.

4.1.2) Behavioural based approach**:** In this technique eye blinking frequency, head pose, etc. of a person is monitored through a camera and the person is alerted if any of these drowsiness symptoms are detected.

4.2) The various technology that can be used are discussed as:

4.2.1) TensorFlow: It is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production.

TensorFlow computations are expressed as state full dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors".

4.2.2) Machine learning: Machine learning is the kind of programming which gives computers the capability to automatically learn from data without being explicitly programmed. This means in other words that these programs change their behaviour by learning from data. Python is clearly one of the best languages for machine learning. Python does contain special libraries for machine learning namely scipy, pandas and numpy which great for linear algebra and getting to know kernel methods of machine learning. The language is great to use when working with machine learning algorithms and has easy syntax relatively.

4.2.3) OpenCV: OpenCV stands for Open Source Computer Vision. It's an Open Source BSD licensed library that includes hundreds of advanced Computer Vision algorithms that are optimized to use hardware acceleration. OpenCV is commonly used for machine learning, image processing, image manipulation, and much more. OpenCV has a modular structure. There are shared and static libraries and a CV Namespace.

In short, OpenCV is used in our application to easily load bitmap files that contain landscaping pictures and perform a blend operation between two pictures so that one picture can be seen in

the background of another picture. This image manipulation is easily performed in a few lines of code using OpenCV versus other methods. OpenCV.org is a must if you want to explore and dive deeper into image processing and machine learning in general.

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

IMPLEMENTATION:

The implementation details of each the modules can be explained as follows:

Video Acquisition:

OpenCV provides extensive support for acquiring and processing live videos. It is also possible to choose whether the video has to be captured from the in-built webcam or an external camera by setting the right parameters. As mentioned earlier, OpenCV does not specify any minimum requirement on the camera, however OpenCV by default expects a particular resolution of the video that is being recorded, if the resolutions do not match, then an error is thrown. This error can be countered, by over riding the default value, which can be achieved, by manually specifying the resolution of the video being recorded.

Dividing into frames:

Once the video has been acquired, the next step is to divide it into a series of frames/images. This was initially done as a 2 step process. The first step is to grab a frame from the camera or a video file, in our case since the video is not stored, the frame is grabbed from the camera and once this is achieved, the next step is to retrieve the grabbed frame. While retrieving , the image/frame is first decompressed and then retrieved . However, the two step process took a lot of processing time as the grabbed frame had to be stored temporarily. To overcome this problem, we came up with a single step process, where a single function grabs a frame and returns it by decompressing.

Face detection:

Once the frames are successfully extracted the next step is to detect the face in each of these frames. This is achieved by making use of the Haarcascade file for face detection. The Haarcascade file contains a number of features of the face , such as height , width and thresholds of face colors., it is constructed by using a number of positive and negative samples. For face detection, we first load the cascade file. Then pass the acquired frame to an edge detection function, which detects all the possible objects of different sizes in the frame. To reduce the amount of processing, instead of detecting objects of all possible sizes, since the face of the automobile driver occupies a large part of the image, we can specify the edge detector to detect only objects of a particular size, this size is decided based on the Haarcascade file, wherein each Haarcascade file will be designed for a particular size. Now, the output the edge detector is stored in an array. Now, the output of the edge detector is then compared with the cascade file to identify the face in the frame. Since the cascade consists of both positive and negative samples, it is required to specify the number of failures on which an object detected should be classified as a negative sample. In our system, we set this value to 3, which helped in achieving both accuracy as well as less processing time. The output of this module is a frame with face detected in it.

Eye detection :

After detecting the face, the next step is to detect the eyes, this can be achieved by making use of the same technique used for face detection. However, to reduce the amount of processing, we mark the region of interest before trying to detect eyes. The region of interest is set by taking into account the following:

- The eyes are present only in the upper part of the face detected.
- The eyes are present a few pixels lower from the top edge of the face.

Once the region of interest is marked, the edge detection technique is applied only on the region of interest, thus reducing the amount of processing significantly. Now, we make use of the same technique as face detection for detecting the eyes by making use of Haarcascade Xml file for eyes detection. But, the output obtained was not very efficient, there were more than two objects classified as positive samples, indicating more than two eyes. To overcome this problem, the following steps are taken:

- Out of the detected objects, the object which has the highest surface area is obtained. This is considered as the first positive sample.
- Out of the detected objects, the object which has the highest surface area is obtained. This is considered as the first positive sample.

- A check is made to make sure that the two positive samples are not the same.
- Now, we check if the two positive samples have a minimum of 30 pixels from either of the edges.
- Next , we check if the two positive samples have a minimum of 20 pixels apart from each other.

After passing the above tests, we conclude that the two objects i.e positive sample 1 and positive sample 2 , are the eyes of the automobile driver.

Drowsiness Detection:

Once the eyes are detected, the next step is to determine if the eyes are in closed or open state. This is achieved by extracting the pixel values from the eye region. After extracting , we check if these pixel values are white, if they are white then it infers that the eyes are in the open state, if the pixel values are not white then it infers that the eyes are in the closed state .

This is done for each and every frame extracted. If the eyes are detected to be closed for two seconds or a certain number of consecutive frames depending on the frame rate, then the automobile driver is detected to be drowsy. If the eyes are detected to be closed in non consecutive frames, then We declare it as a blink.

If drowsiness is detected, a text message is displayed along with triggering an audio alarm. But, it was observed that the system was not able to run for an extended period of time , because the conversion of the acquired video from RGB to grayscale was occupying too much memory. To overcome this problem, instead of converting the video to grayscale, the RGB video only was used for processing. This conversion resulted in the following advantages,

- Better differentiation between colours, as it uses multichannel colours.
- Consumes very less memory.

- Capable of achieving blink detection, even when the automobile driver is wearing spectacles.

Hence there were two versions of the system that was implemented; the version 1.0 involves the conversion of the image to grayscale. Current version 2.0 makes use of the RGB video for processing.

4.3) System Description:

4.3.1) Login/ Sign up: The first step in the system is the simple and flexible credential interface where the user/driver must enter the login credentials, or he/she can Sign up in order to be a part of the system.

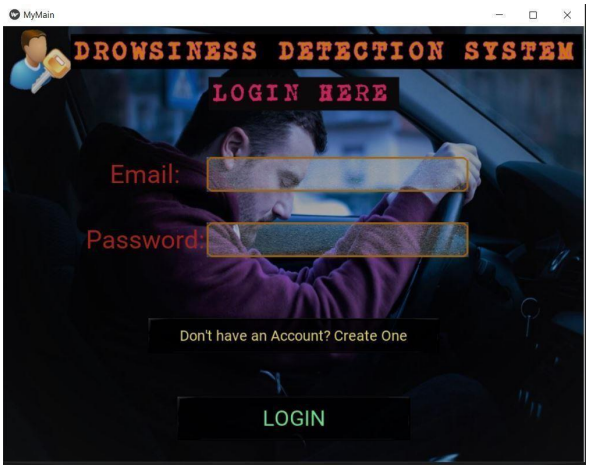


Fig.4.1

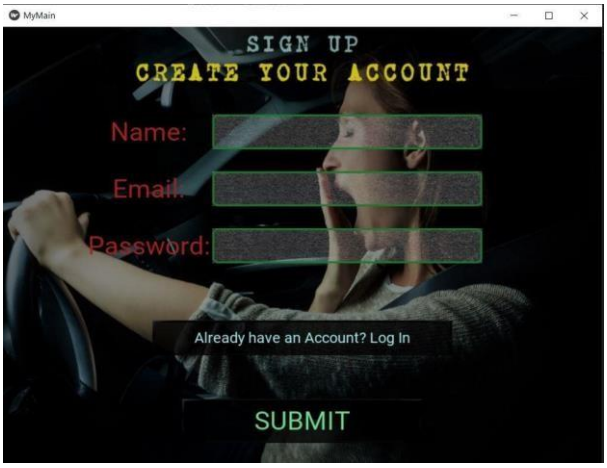


Fig.4.2

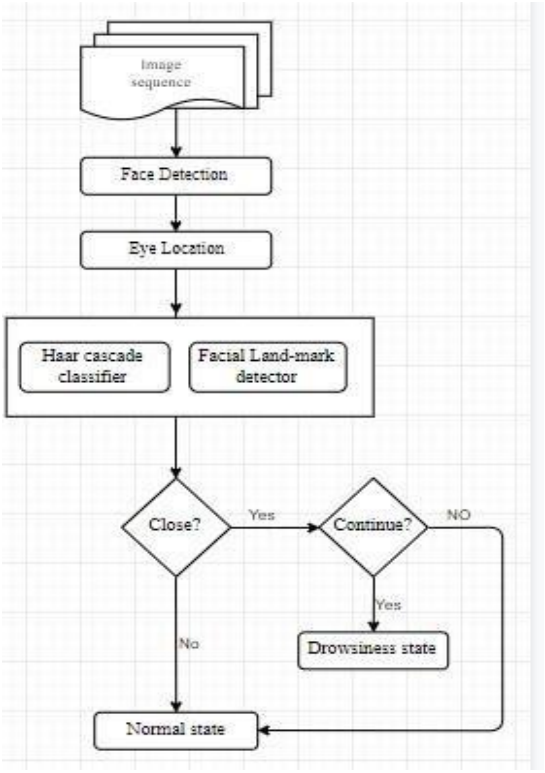


Fig.4.3**:** Flowchart of system

II. FLOWCHART AND ALGORITHM:

The various detection stages are discussed as:

4.3.2) Face Detection: For the face Detection it uses 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. Fig. 4.4 represents five Haar like features & example is shown in Fig.4.5

Fig. 4.4

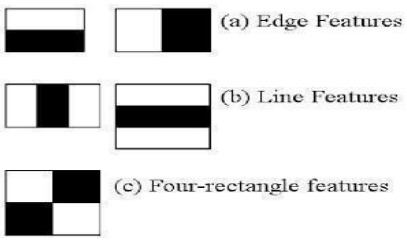


Fig. 4.5

A cascaded AdaBoost classifier with the Haar-like features is exploited to find out the face region. First, the compensated image is segmented into numbers of rectangle areas, at any position and scale within the original image. Due to the difference of facial feature, Haar-like feature is efficient for real-time face detection. These can be calculated according to the difference of sum of pixel values within rectangle areas. The features can be represented by the different composition of the black region and white region. A cascaded AdaBoost classifier is a strong classifier which is a combination of several weak classifiers. Each weak classifier is trained by Adaboost algorithm. If a candidate sample passes through the cascaded AdaBoost classifier, the face region can be found. Almost all of face samples can pass through and non-face samples can be rejected

4.3.3) Eye detection: In the system we have used facial landmark prediction for eye detection Facial landmarks are used to localize and represent salient regions of the face, such as:

- Eyes
- Eyebrows
- Nose
- Mouth
- Jawline

Facial landmarks have been successfully applied to face alignment, head pose estimation, face swapping, blink detection and much more. In the context of facial landmarks, our goal is detecting important facial structures on the face using shape prediction methods. Detecting facial landmarks is therefore a twostep process:

- Localize the face in the image.
- Detect the key facial structures on the face ROI.

Localize the face in the image: The face image is localized by Haar feature-based cascade classifiers which was discussed in the first step of our algorithm i.e. face detection.
Detect the key facial structures on the face ROI: There are a variety of facial landmark detectors, but all methods essentially try to localize and label the following facial regions:

- Mouth
- Right eyebrow
- Left eyebrow
- Right eye
- Left eye
- Nose

The facial landmark detector included in the Dlib library is an implementation of the One Millisecond Face Alignment with an Ensemble of Regression Trees paper by Kazemi and Sullivan (2014).

This method starts by using:

1. A training set of labelled facial landmarks on an image. These images are manually labelled, specifying specific (x, y)-coordinates of regions surrounding each facial structure.
2. Priors, of more specifically, the probability on distance between pairs of input pixels. The pre-trained facial landmark detector inside the Dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

The indexes of the 68 coordinates can be visualized on the image below:

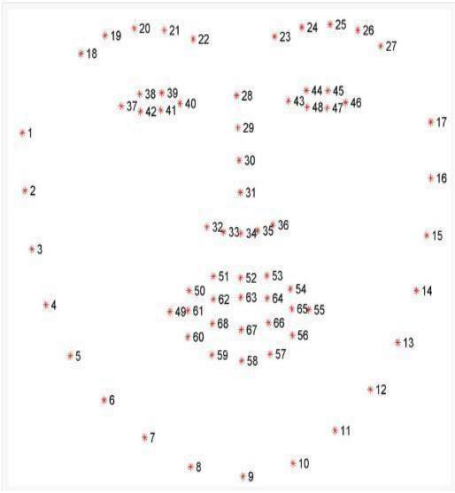


Fig.4.6: Visualizing the 68 facial landmark coordinates

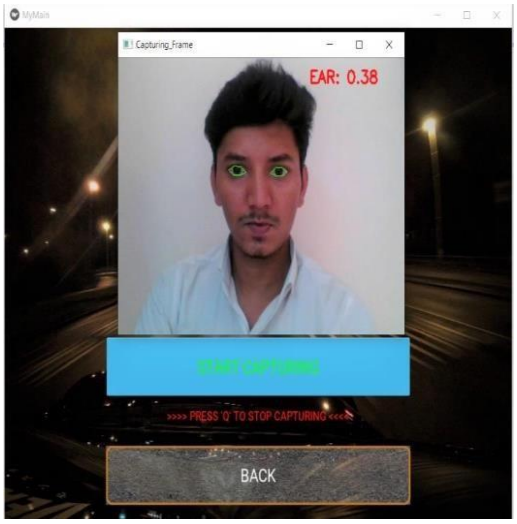


Fig.4.7: Detection of both the eyes

We can detect and access both the eye region by the following facial landmark index show below

- The right eye using [36, 42].
- The left eye with [42, 48].

These annotations are part of the 68 point iBUG 300-W dataset which the Dlib facial landmark predictor was trained on. It's important to note that other flavours of facial landmark detectors exist, including the 194 point model that can be trained on the HELEN dataset.

Regardless of which dataset is used, the same Dlib framework can be leveraged to train a shape predictor on the input training data.

4.3.4) Recognition of Eye's State:

The eye area can be estimated from optical flow, by sparse tracking or by frame-to-frame intensity differencing and adaptive thresholding. And Finally, a decision is made whether the eyes are or are not covered by eyelids. A different approach is to infer the state of the eye opening from a single image, as e.g. by correlation matching with open and closed eye templates, a heuristic horizontal or vertical image intensity projection over the eye region, a parametric model fitting to find the eyelids, or active shape models. A major drawback of the previous approaches is that they usually implicitly impose too strong requirements on the setup, in the sense of a relative face-camera pose (head orientation), image resolution, illumination, motion dynamics, etc. Especially the heuristic methods that use raw image intensity are likely to be very sensitive despite their real-time performance.

Therefore, we propose a simple but efficient algorithm to detect eye blinks by using a recent facial landmark detector. A single scalar quantity that reflects a level of the eye opening is derived from the landmarks. Finally, having a per-frame sequence of the eye-opening estimates, the eye blinks are found by an SVM classifier that is trained on examples of blinking and non-blinking patterns.

Eye Aspect Ratio Calculation:

For every video frame, the eye landmarks are detected. The eye aspect ratio (EAR) between height and width of the eye is computed.

$$EAR = \frac{||p2 - p6|| + ||p3 - p5||}{2||p1 - p4||} \quad (1)$$

where p1, . . ., p6 are the 2D landmark locations, depicted in Fig. 1. The EAR is mostly constant when an eye is open and is getting close to zero while closing an eye. It is partially person and head pose insensitive. Aspect ratio of the open eye has a small variance among individuals, and it is fully invariant to a uniform scaling of the image and in-plane rotation of the face. Since eye blinking is performed by both eyes synchronously, the EAR of both eyes is averaged.

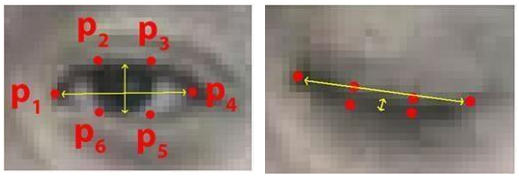


Fig.4.8

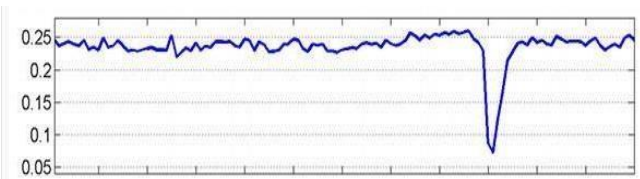


Fig.4.9 EAR for single blink

Fig 4.8: Open and closed eyes with landmarks p(i) automatically detected. The eye aspect ratio EAR in Eq. (1) plotted for several frames of a video sequence.

4.3.5) Eye State Determination:

Finally, the decision for the eye state is made based on EAR calculated in the previous step. If the distance is zero or is close to zero, the eye state is classified as “closed” otherwise the eye state is identified as “open”.

4.3.6) Drowsiness Detection:

The last step of the algorithm is to determine the person’s condition based on a pre-set condition for drowsiness. The average blink duration of a person is 100-400 milliseconds (i.e. 0.1-0.4 of a second). Hence if a person is drowsy his eye closure must be beyond this interval. We set a time frame of 5 seconds. If the eyes remain closed for five or more seconds, drowsiness is detected and alert pop regarding this is triggered.

1

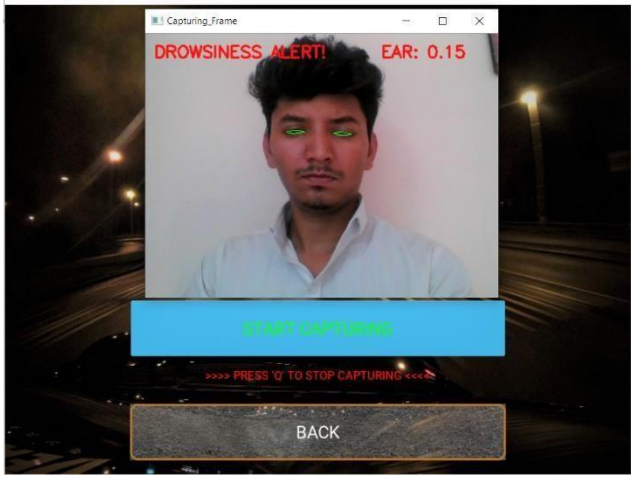


Fig.4.10 Drowsiness state

Fig .4.11 Console information

```
[INFO   ] [Base        ] Start application main loop
[INFO] loading facial landmark predictor...
[INFO] starting video stream thread...
Drowsiness detected for : 3.5 sec
Drowsiness detected for : 1.5666666666666667 sec
Drowsiness detected for : 5.383333333333334 sec
Drowsiness detected for : 1.55 sec
Drowsiness detected for : 7.85 sec
Total Drowsiness detected for : 19.85 sec
####################################
[INFO   ] [Base        ] Leaving application in progress...
```

5.**Testing**

The tests were conducted in various conditions including:

1. Different lightining conditions.
2. Drivers posture and position of the automobile drivers face.
3. Drivers with spectacles.

5.1 Test case 1: When there is ambient light:

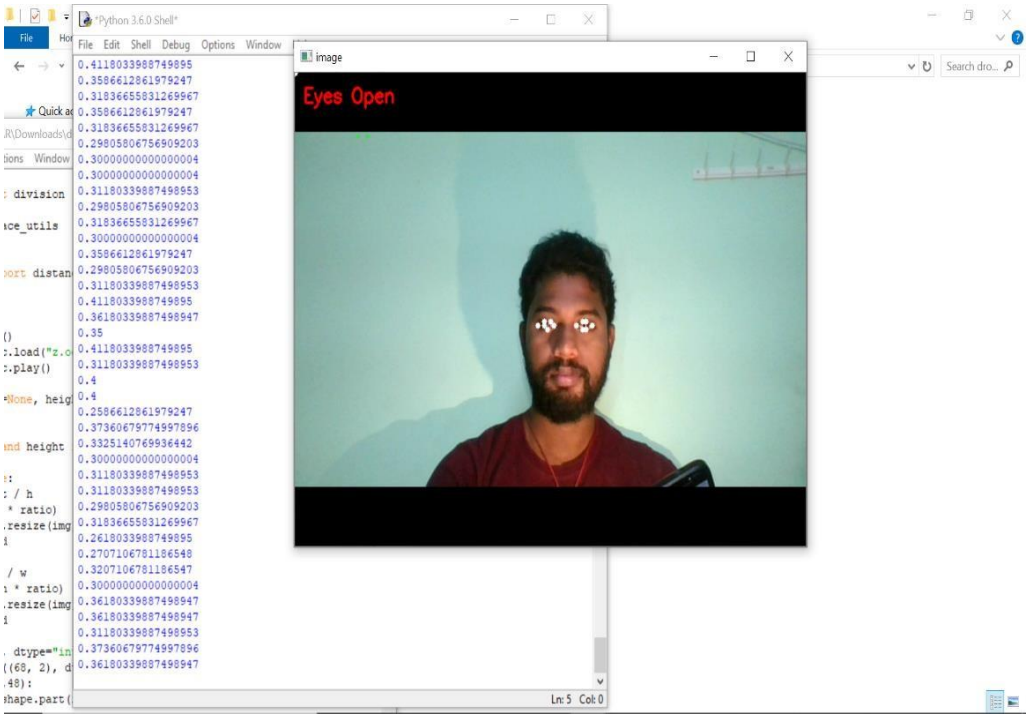


Fig 5.1. Test Scenario #1 – Ambient lighting

Result: As shown in Fig 5.1, when there is ambient amount of light, the automobile driver's face and eyes are successfully detected.

5.2 Test case 2: Position of the automobile drivers face:

Centre Positioned:

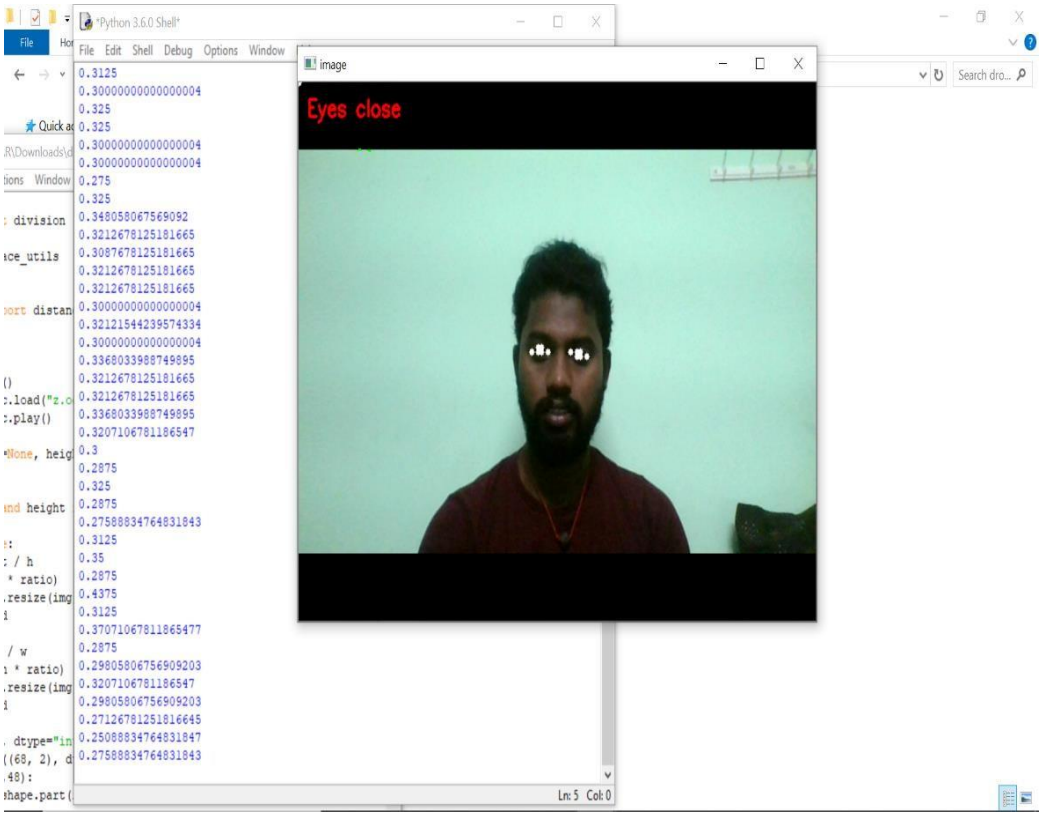


Fig 5.2. Test Scenario Sample#2 driver face in center of frame

RESULT : As shown in Fig 5.2,When the automobile driver's face is positioned at the Centre, the face, eyes , eye blinks, and drowsiness was successfully detected.

Right Positioned :

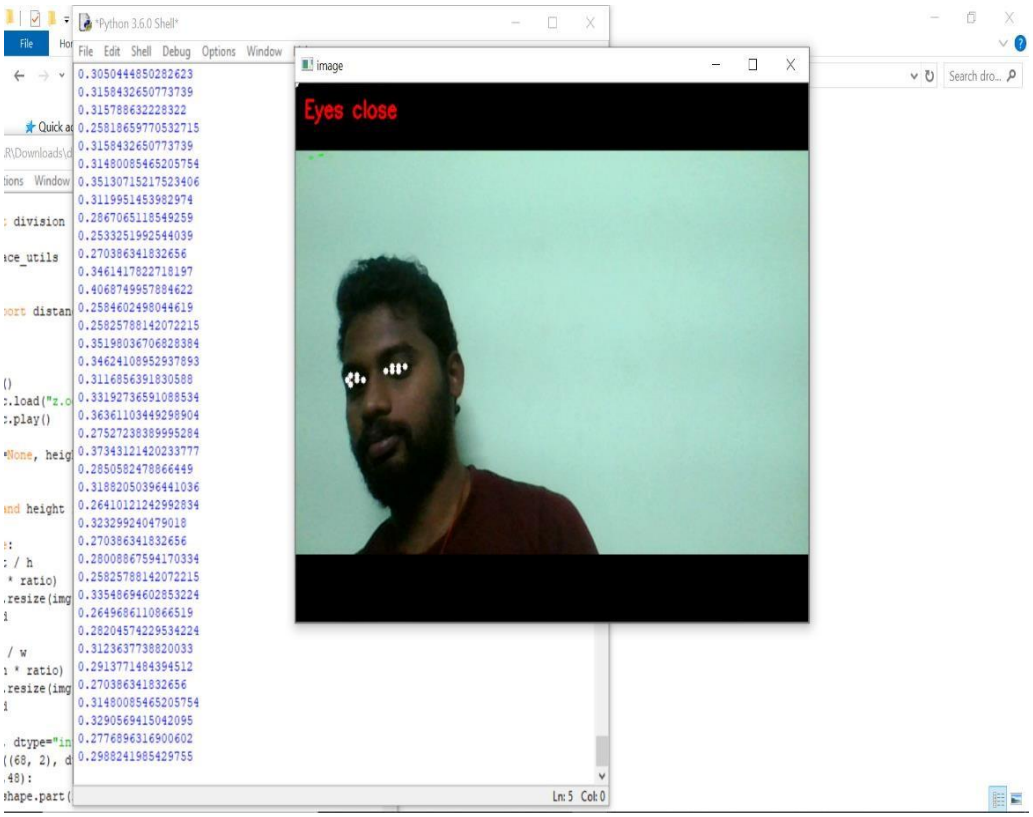


Fig 5.3. Test Scenario Sample#3 – driver face to right of frame

RESULT: As shown in Fig 5.3,When the automobile driver's face is positioned at the Right, the face, eyes , eye blinks, and drowsiness was successfully detected.

Left positioned:

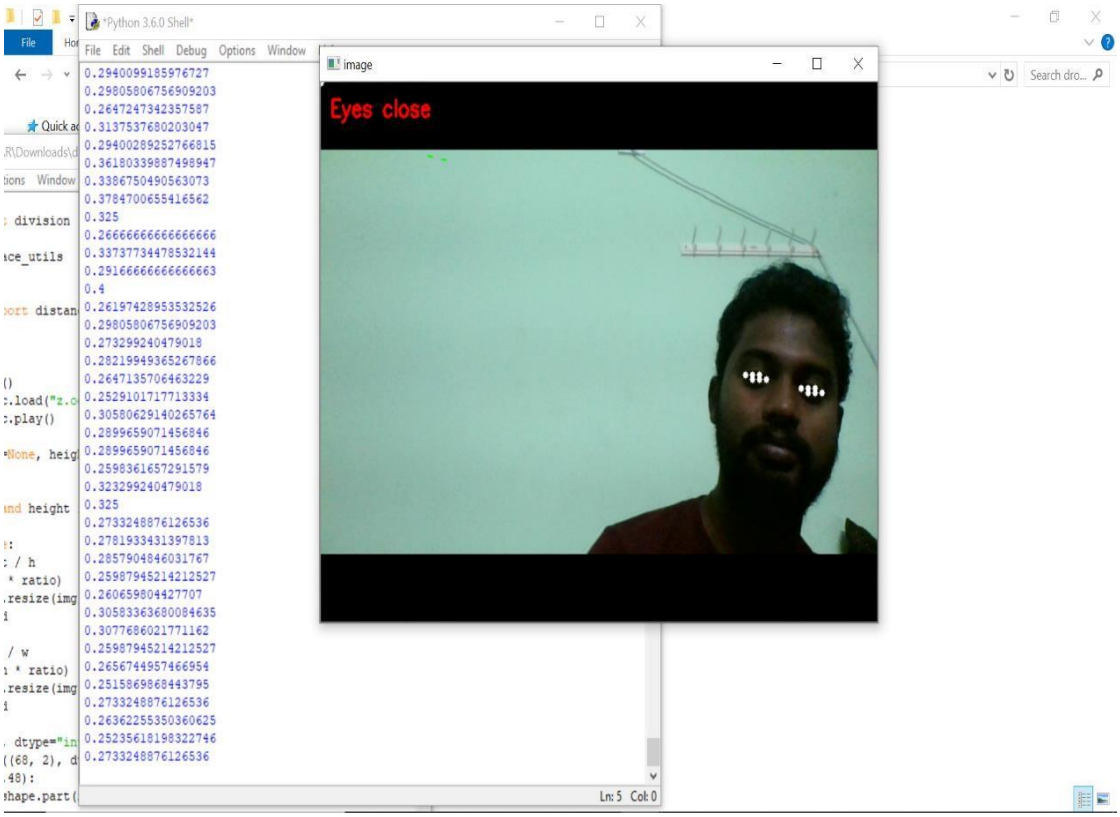


Fig 5.4 Test Scenario Sample#4- driver face to left of frame

RESULT : As shown in screen snapshot in Fig 5.4,when the automobile driver's face is positioned at the Left, the face, eyes , eye blinks, and drowsiness was successfully detected.

Test case 3: When the automobile driver is wearing spectacles:

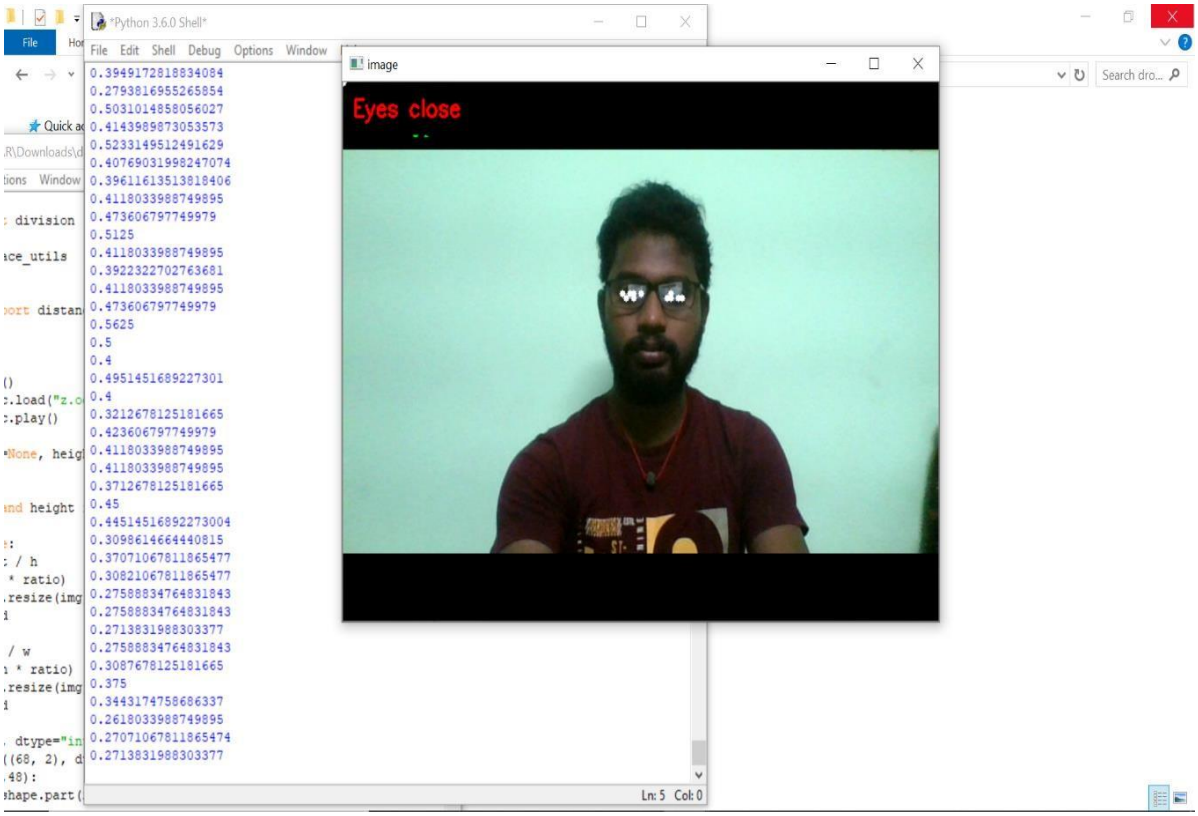


Fig 5.5. Test Scenario Sample#5 – Driver with spectacles

RESULT : As shown in screen snapshot in Fig 5.5,When the automobile driver is wearing spectacles, the face, eyes , eye blinks, and drowsiness was successfully detected.

Test case 4: When the automobile driver's head s tilted:

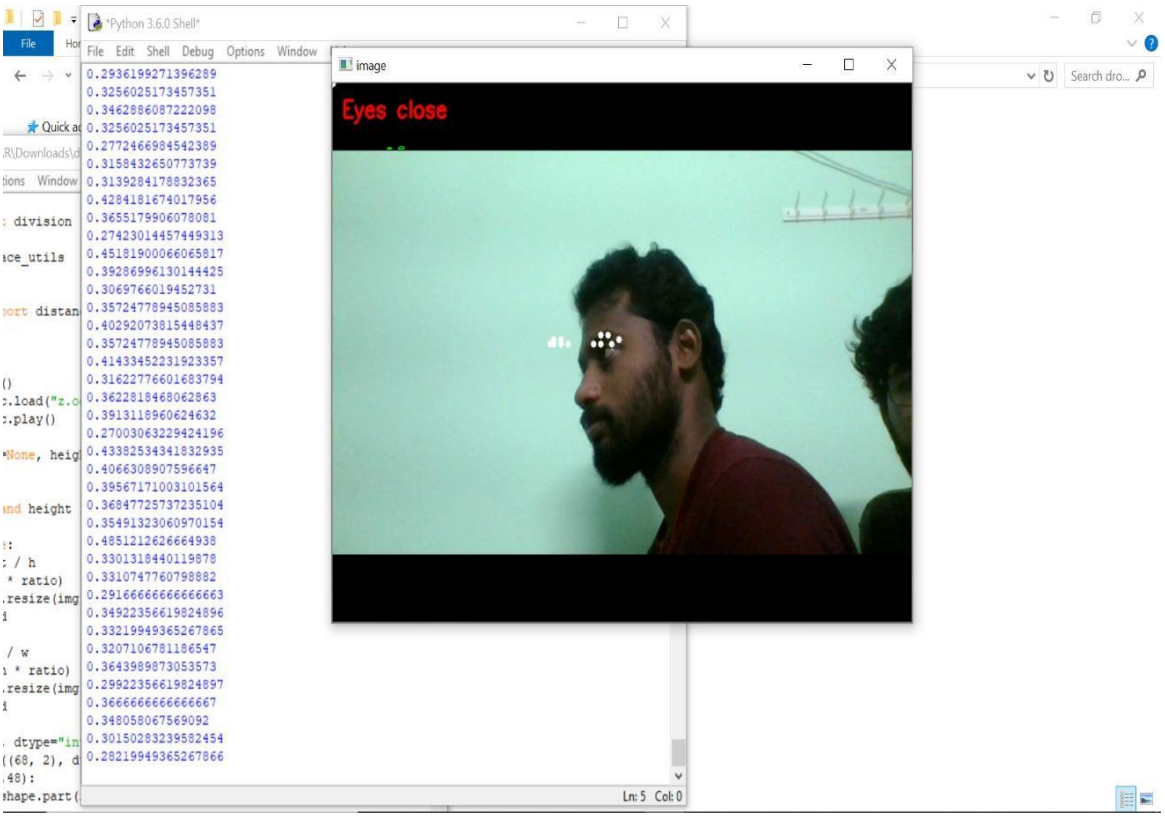


Fig 5.6. Test Scenario Sample#6 – Various posture

RESULT : As shown in screen snapshot in Fig 5.6,when the automobile driver's face is tilted for more than 30 degrees from vertical plane, it was observed that the detection of face and eyes failed.

The system was extensively tested even in real world scenarios, this was achieved by placing the camera on the dashboard of the car , focusing on the automobile driver. It was found that the system gave positive output unless there was any direct light falling on the camera.

6. Project Work flow (diagrams)

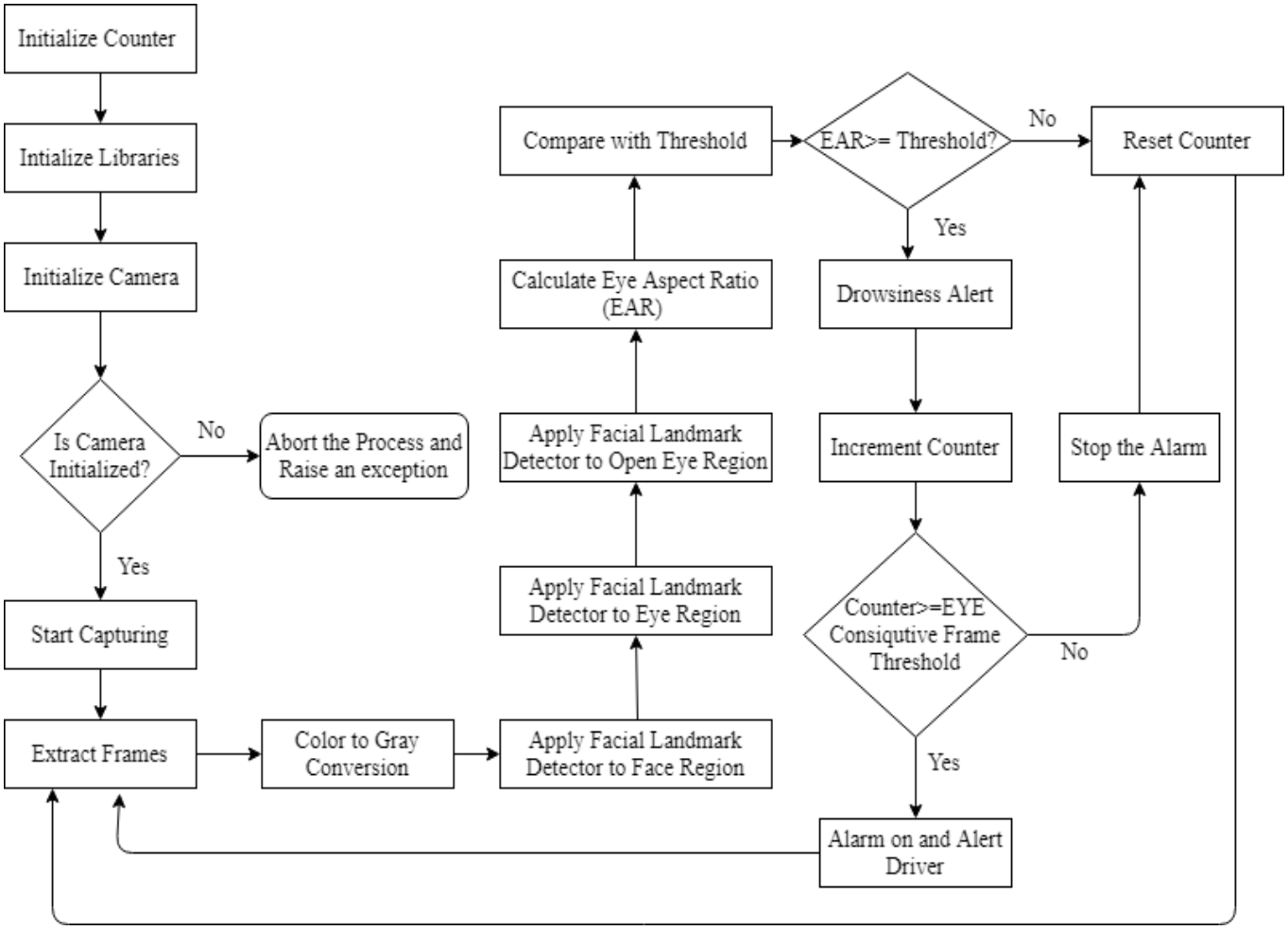


Fig. 6.1 Flow chart of System

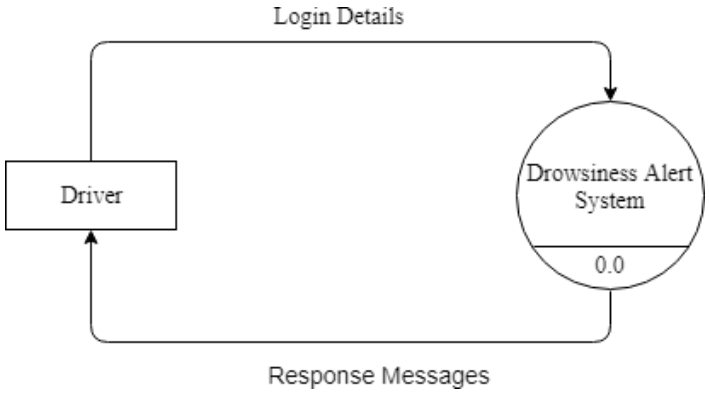


Fig.6.2 DFD level 0

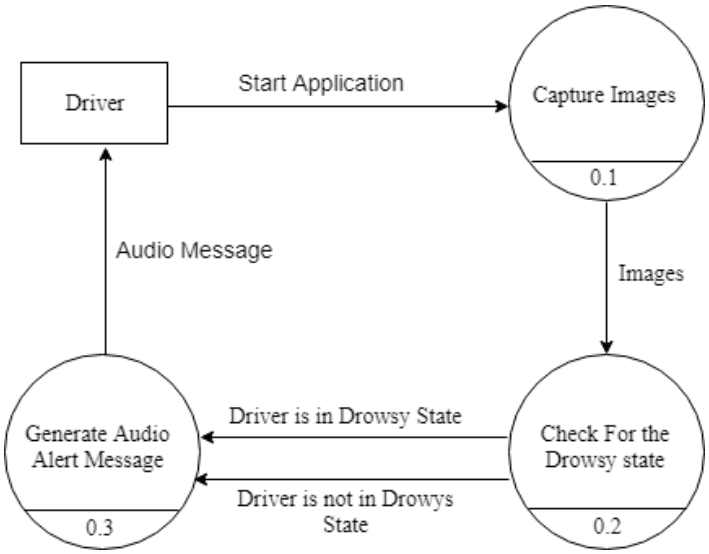


Fig 6.3 DFD level 1

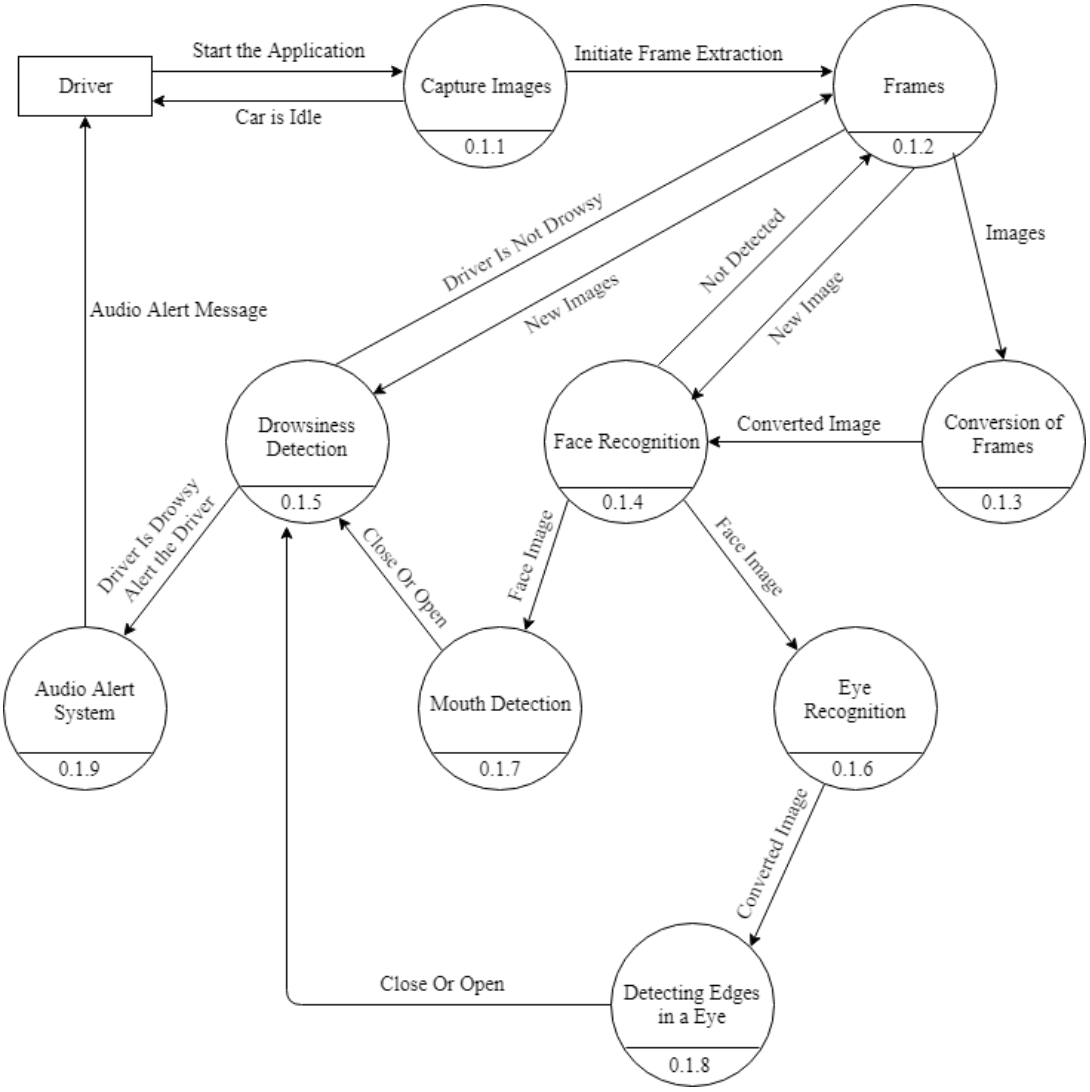


Fig 6.4 DFD level 2

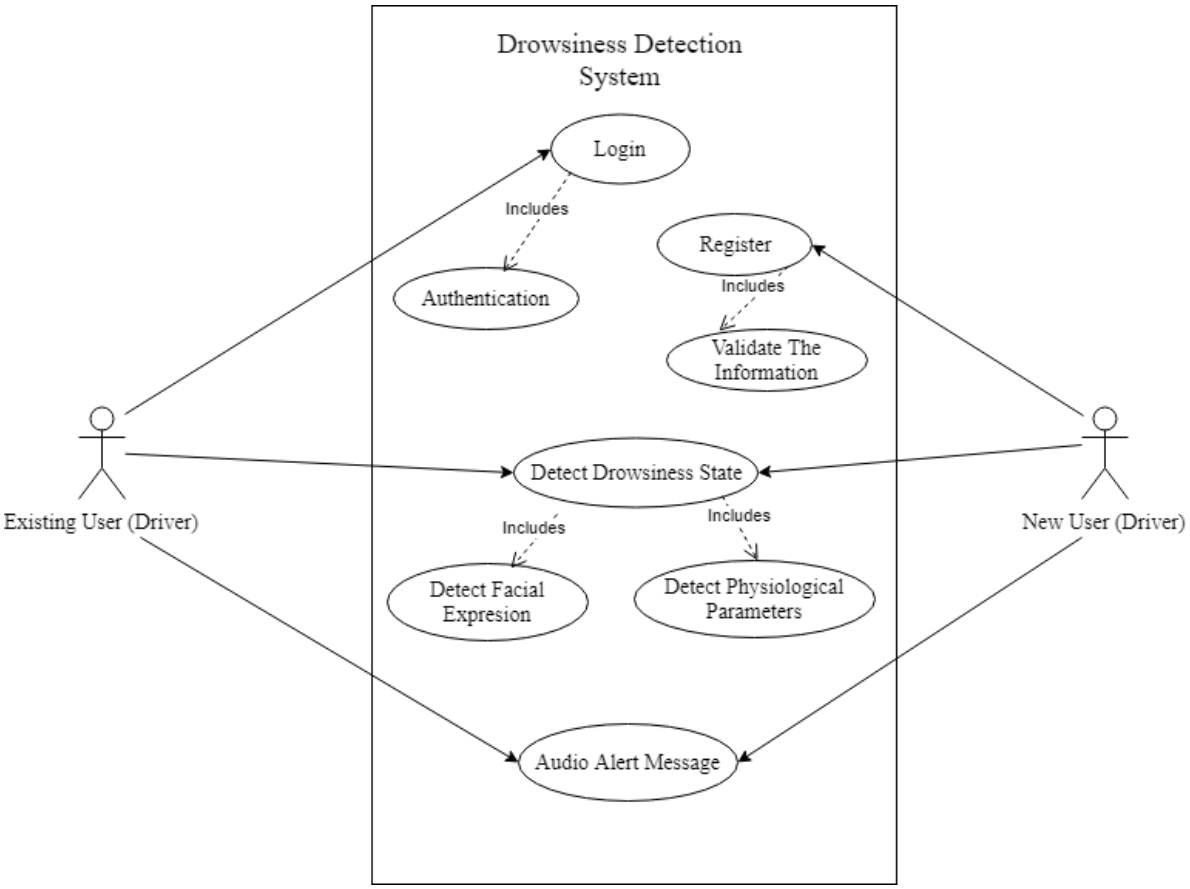


Fig 6.5 use-case Diagram

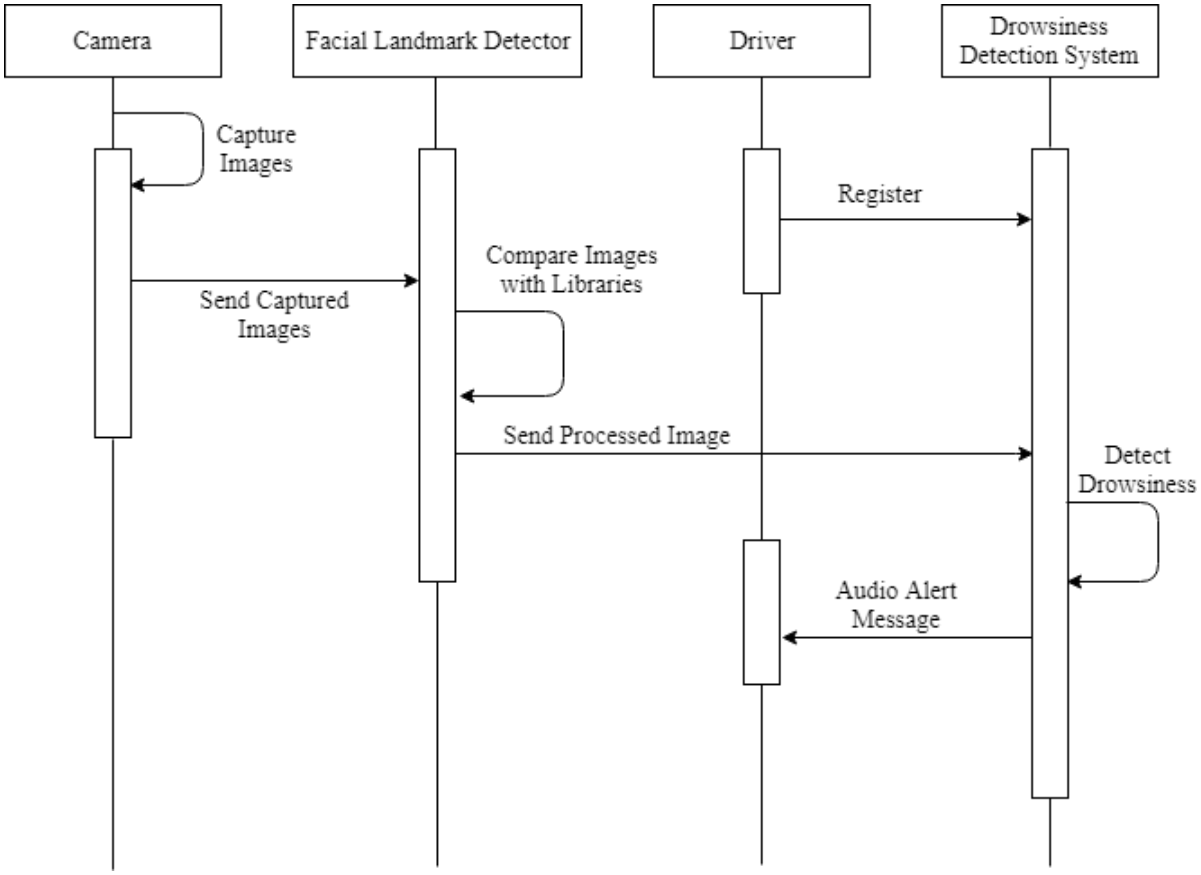
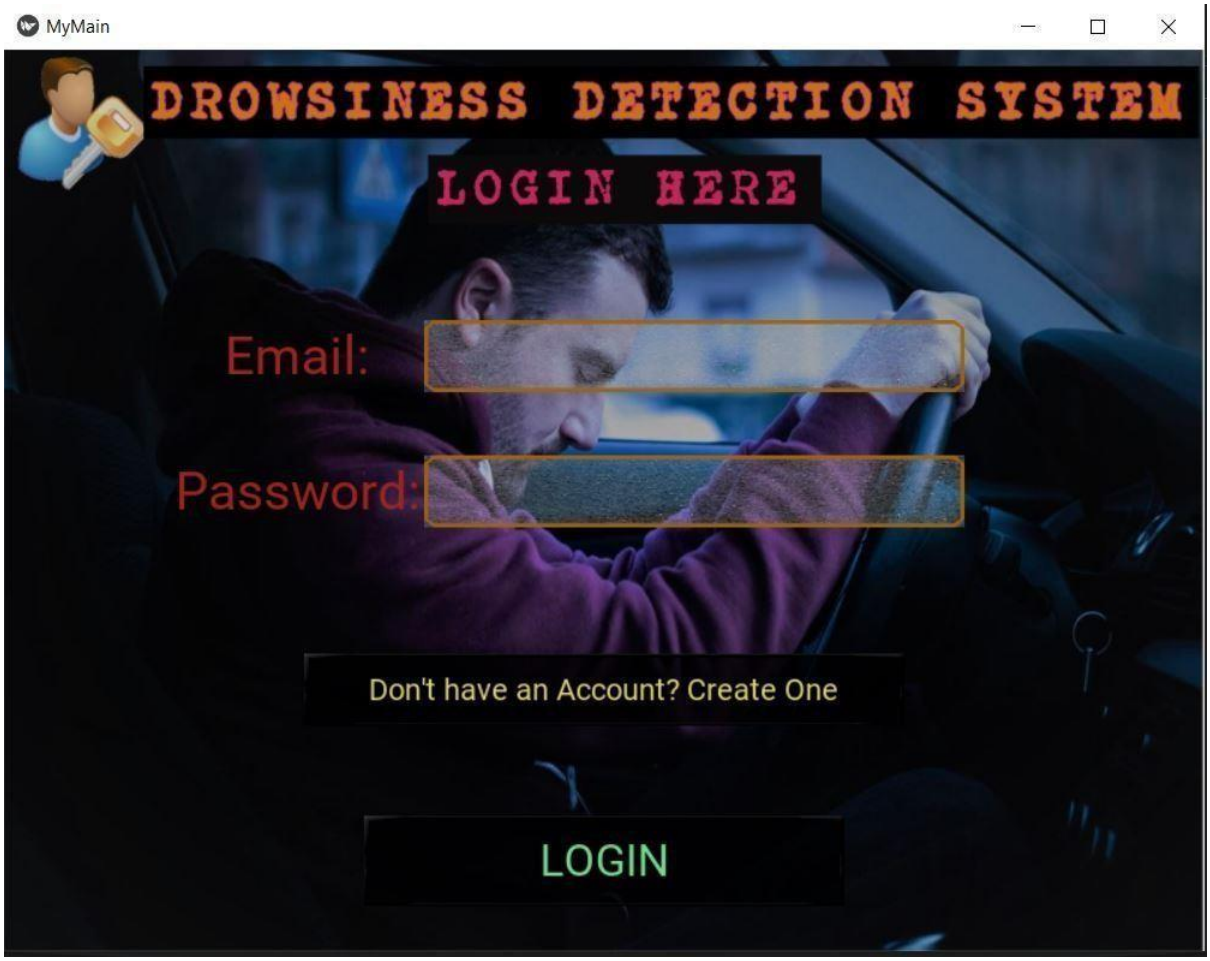


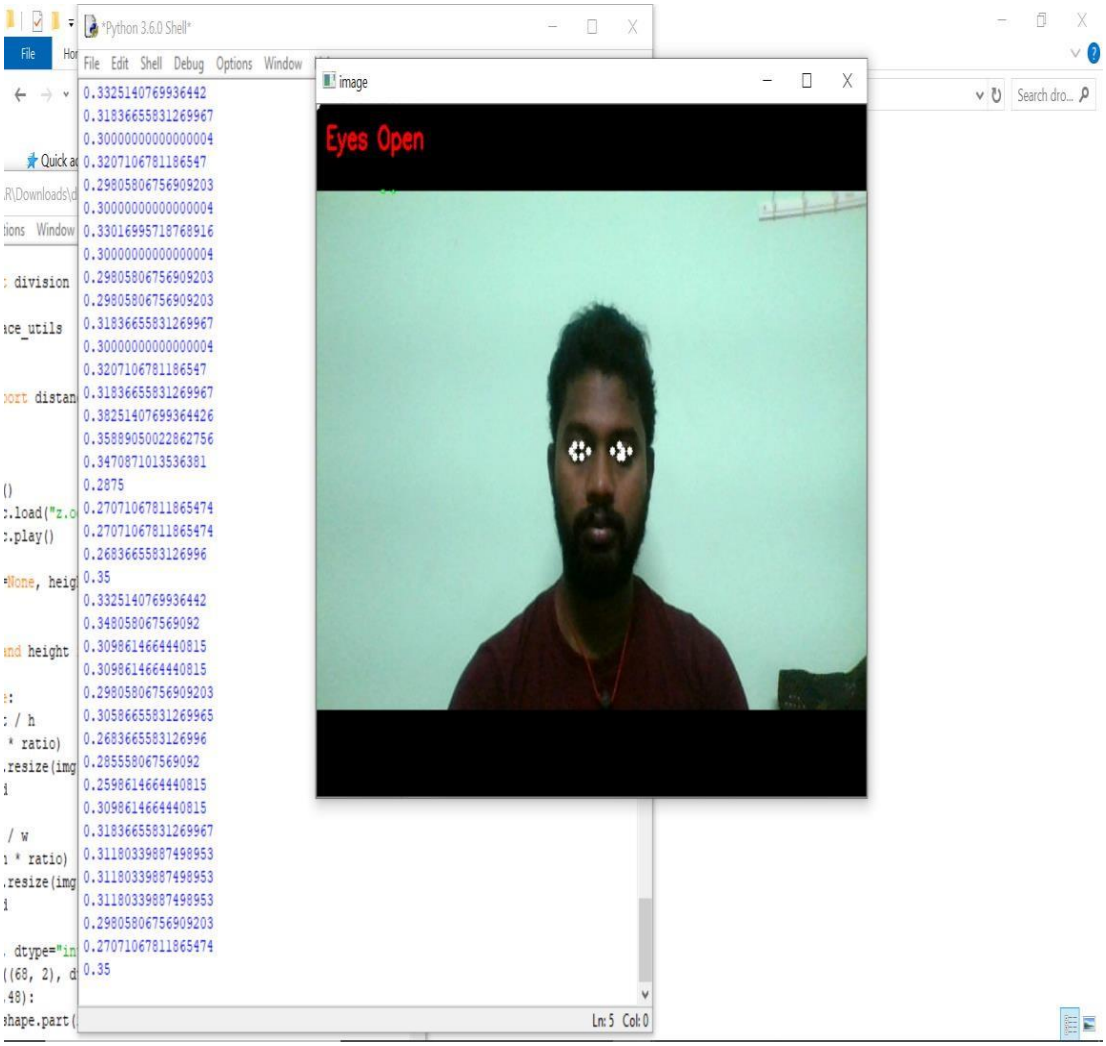
Fig 6.6 Sequence Diagram

7. Results and Discussions

Implementation of drowsiness detection with Python and OpenCV was done which includes the following steps: Successful runtime capturing of video with camera. Captured video was divided into frames and each frame were analysed. Successful detection of face followed by detection of eye. If closure of eye for successive frames were detected, then it is classified as drowsy condition else it is regarded as normal blink and the loop of capturing image and analysing the state of driver is carried out again and again. In this implementation during the drowsy state the eye is not surrounded by circle or it is not detected, and corresponding message is shown.

Screenshots of the system shown below:





Future Work:

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

We can also provide the user with an Android application which will provide with the information of his/her drowsiness level during any journey. The user will know Normal state, Drowsy State, the number of times blinked the eyes according to the number of frames captures. Which can be shown in fig 6.1

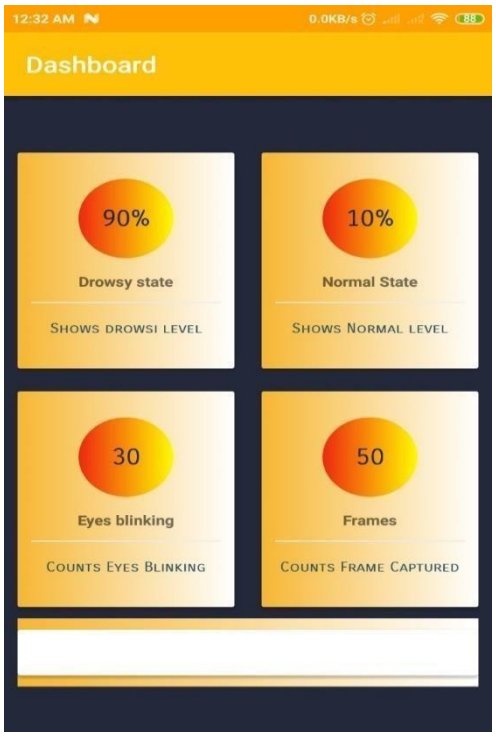


Fig 7.1 Android Application for Future scope

8. Conclusions

A real-time eye blink detection algorithm was presented. We quantitatively demonstrated that Haar feature-based cascade classifiers and regression-based facial landmark detectors are precise enough to reliably estimate the positive images of face and a level of eye openness. While they are robust to low image quality (low image resolution in a large extent) and in-thewild.

The primary goal of this project is to develop a real time drowsiness monitoring system in automobiles. We developed a simple system consisting of 5 modules namely(a) video acquisition, (b) dividing into frames, (c) face detection, (d) eye detection, and (e)drowsiness detection . Each of these components can be implemented independently thus providing a way to structure them based on the requirements.

Four features that make our system different from existing ones are:

a. Focus on the driver, which is a direct way of detecting the drowsiness
b. A real-time system that detects face,iris, blink, and driver drowsiness
c. A completely non-intrusive system, and
d. Cost effective

Limitations:

Use of spectacles: In case the user uses spectacle (sunglasses) then it is difficult to detect the state of the eye. As it hugely depends on light hence reflection of spectacles may give the output for a closed eye as opened eye. Hence for this purpose the closeness of eye to the camera is required to avoid light.

Multiple face problem: If multiple face arises in the window then the camera may detect more number of faces undesired output may appear. Because of different condition of different faces. So, we need to make sure that only the driver face come within the range of the camera. Also, the speed of detection reduces because of operation on multiple faces.

Future Enhancement:

The system at this stage is a “Proof of Concept” for a much substantial endeavour. This will serve as a first step towards a distinguished technology that can bring about an evolution aimed at ace development. The developed system has special emphasis on real-time monitoring with flexibility, adaptability and enhancements as the foremost requirements.

Future enhancements are always meant to be items that require more planning, budget and staffing to have them implemented. There following are couple of recommended areas for future enhancements:

- **Standalone product:** It can be implemented as a standalone product, which can be installed in an automobile for monitoring the automobile driver.
- **Smart phone application:** It can be implemented as a smart phone application, which can be installed on smart phones. And the automobile driver can start the application after placing it at a position where the camera is focused on the driver.

9. Appendix

Coding

Importing our required Python packages.

detect_drowsiness.py

```
import  the necessary packages
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np  import dlib  import cv2
```

Sound Alarm

```
  def start_sound():
        pygame.mixer.init()
pygame.mixer.music.load("z.ogg" )
 pygame.mixer.music.play()
# z.ogg is the music file
```

eye_aspect_ratio function def

```
resize(img,                 width=None,
height=None,
interpolation=cv2.INTER_AREA):
    global ratio
    w, h = img.shape
```

```
    if width is None and height is None:
        return img
    elif width is None:
        ratio = height / h
        width = int(w * ratio)
        resized = cv2.resize(img, (height, width), interpolation)
        return resized
    else:
        ratio = width / w
        height = int(h * ratio)
        resized = cv2.resize(img, (height, width), interpolation)
        return resized
# return the eye aspect ratio return ear
def shape_to_np(shape, dtype="int"):
    coords = np.zeros((68, 2), dtype=dtype)
    for i in range(36,48):
        coords[i] = (shape.part(i).x, shape.part(i).y)
    return coords
def eye_aspect_ratio(eye):
# calculate eucilean dist.
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
ear = (A + B) / (2.0 * C)
```

Defining EYE_AR_THRESH

```
EYE_AR_THRESH = 0.3
EYE_AR_CONSEC_FRAMES = 48
COUNTER = 0
ALARM_ON = False
```

Facial landmark predictor

```
predictor_path = 'shape_predictor_68_face_landmarks.dat_2'


detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(predictor_path)
```

Extracting the eye regions

```
# grab the indexes of the facial landmarks for the left and
# right eye, respectively
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

Facial landmark detection to localize each of the important regions of the face:

```
# determine the facial landmarks for the face region, then
# convert the facial landmark (x, y)-coordinates to a NumPy


# extract the left and right eye coordinates, then use the
#coordinates to compute the eye aspect ratio for both eyes
shape = predictor(frame_resized, d)
    shape = shape_to_np(shape)
    leftEye= shape[lStart:lEnd]
```

```
        rightEye=  shape[rStart:rEnd]
        leftEAR= eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
```

Visualize each of the eye regions

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

Check to see if the person in our video stream is starting to show symptoms of drowsiness:

```
if ear>.25:
            print (ear)
            total=0
            alarm=False
            cv2.putText(frame, "Eyes Open ",
(10,30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        else:
            total+=1
```

```
            if total>20:

                if not alarm:

                    alarm=True

                    d=threading.Thread(target=start_sound)

                    d.setDaemon(True)

                    d.start()

                    cv2.putText(frame, "drowsiness detect" ,(250,
30),cv2.FONT_HERSHEY_SIMPLEX, 1.7, (0, 0, 0), 4)

            cv2.putText(frame, "Eyes close".format(total), (10,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        for (x, y) in shape:

            cv2.circle(frame, (int(x/ratio), int(y/ratio)), 3, (255,
255, 255), -1)
```

Displaying the output frame:

```
cv2.imshow("image", frame)
# if the `q` key was pressed, break from the loop if
        if cv2.waitKey(1) & 0xFF == ord('q'):
    cv2.destroyAllWindows()
    camera.release()
    break
```

10. References

IEEE standard

Journal Paper,

[1] Facial Features Monitoring for Real Time Drowsiness Detection by Manu B.N, 2016 12th International Conference on Innovations in Information Technology (IIT) [Pg. 78-81] https://ieeexplore.ieee.org/document/7880030

[2] Real Time Drowsiness Detection using Eye Blink Monitoring by Amna Rahman Department of Software Engineering Fatima Jinnah Women University 2015 National Software Engineering Conference (NSEC 2015) https://ieeexplore.ieee.org/document/7396336

[3] Implementation of the Driver Drowsiness Detection System by K. Srijayathi International Journal of Science, Engineering and Technology Research (IJSETR) Volume 2, Issue 9, September 2013 **Names of Websites referred**

https://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-Of-HumanEyes- Using-a https://realpython.com/face-recognition-with-python/

https://www.pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/

https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

https://www.pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/

https://www.codeproject.com/Articles/26897/TrackEye-Real-Time-Tracking-OfHumanEyesUsing-a https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html

https://www.learnopencv.com/training-better-haar-lbp-cascade-eye-detector-opencv/