REAL TIME SLEEP/DROWSINESS DETECTION

Submitted in partial fulfillment of the requirements

of the degree of

Bachelor of Engineering

By

Roshan Shantaram Tavhare

(65)

Guide:

Dr. Varsha Shah

Computer Engineering Department

Rizvi College of Engineering

University of Mumbai

2018-2019

CERTIFICATE

This is to certify that the project entitled **"Real Time Sleep/Drowsiness Detection"** is a bonafide work of **"Roshan Shantaram Tavhare (65)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **"Bachelor of Engineering"** in **"Computer Engineering"**.

Dr. Varsha Shah
Guide

Prof. Shiburaj Pappu
Head of Department

Dr. Varsha Shah
Principal

Project Report Approval for B.E.

This Project report entitled **Real Time Sleep/Drowsiness Detection** by **Roshan Shantaram Tavhare (65)** is approved for the degree of Bachelor of Computer Engineering.

Examiners

1.----------

2.----------

Guide

1.----------

2.----------

Date: 22/04/2019

Place: Mumbai

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Roshan Shantaram Tavhare

Date: 22/04/2019

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 pop-ups.

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.

Index

Sr. No	Title	Page No
1.	Introduction	1
2.	Review and Literature	2
3.	Proposed Methodology and Algorithm	3
	3.1 Types of Methodologies	
	3.1.1 Physiological level approach	3
	3.1.2 Behavioral based approach	3
	3.2 Various Technologies Used	
	3.2.1 Tensor Flow	3
	3.2.2 Machine Learning	3
	3.2.3 Open CV	3
	3.2.4 Kivy	4
	3.3 System Description	
	3.3.1 Login/Sign Up	4
	3.3.2 Face Detection	5
	3.3.3 Eye Detection	6
	3.3.4 Recognition of Eye's State	8
	3.3.5 Eye State Determination	9
	3.3.6 Drowsiness Detection	9
4.	Modelling Diagrams	10
5.	Results and Discussions	13
6.	Conclusion	18
7.	References	19
	Appendix	20
	Acknowledgement	26

List of Figures

Sr. No	Title	Page No
3.1	Login interface	4
3.2	Sign Up interface	4
3.3	Flowchart and Algorithm	5
3.4	Five Harr like Features	5
3.5	Example of Harr like Features	5
3.6	Visualization of 68 Facial landmark coordinates	7
3.7	Detection of both the Eyes	7
3.8	Open and Close eyes with Landmarks	8
3.9	Eye’s Aspect Ratio for single blink	8
3.10	User Set-up Features interface	9
3.11	Drowsiness State	9
3.12	Console information	9
4.1	Flow chart of System	10
4.2	DFD level 0	10
4.3	DFD level 1	11
4.4	DFD level 2	11
4.5	Use-case Diagram	12
4.6	Sequence Diagram	12
5.1	Android App for Future Scope	17

Chapter 1
Introduction

Real Time Drowsiness behaviors 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

Chapter 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, 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 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).

Chapter 3

Proposed Methodology

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

3.1.1) 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.

3.1.2) Behavioral 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.

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

3.2.1) 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 stateful 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".

3.2.2) 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 behavior 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.

3.2.3) 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.

3.2.4) Kivy: Kivy is an open source Python library for developing mobile apps and other multitouch 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.

3.3) System Description:

3.3.1) 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, the same is shown in the fig .1 as follows

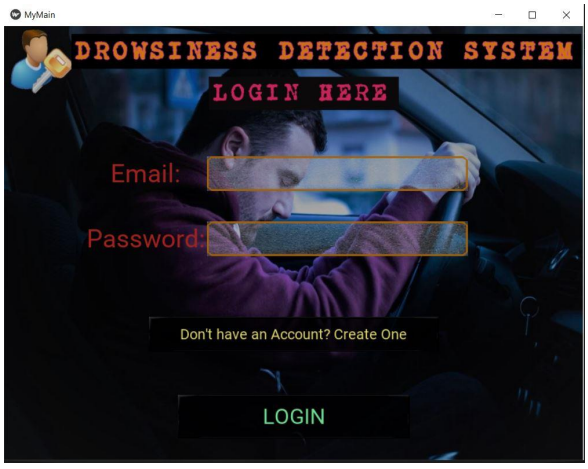


Fig.3.1

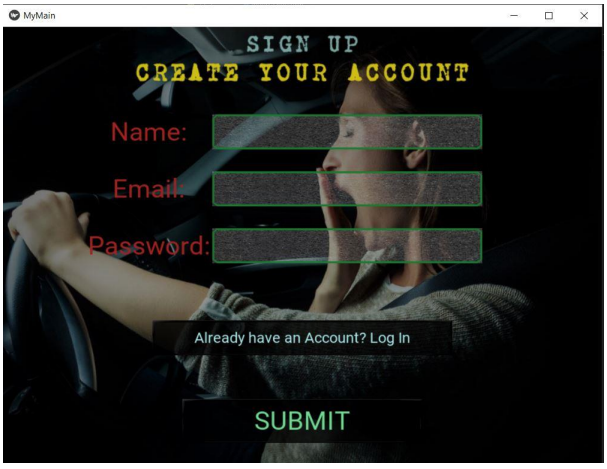


Fig.3.2

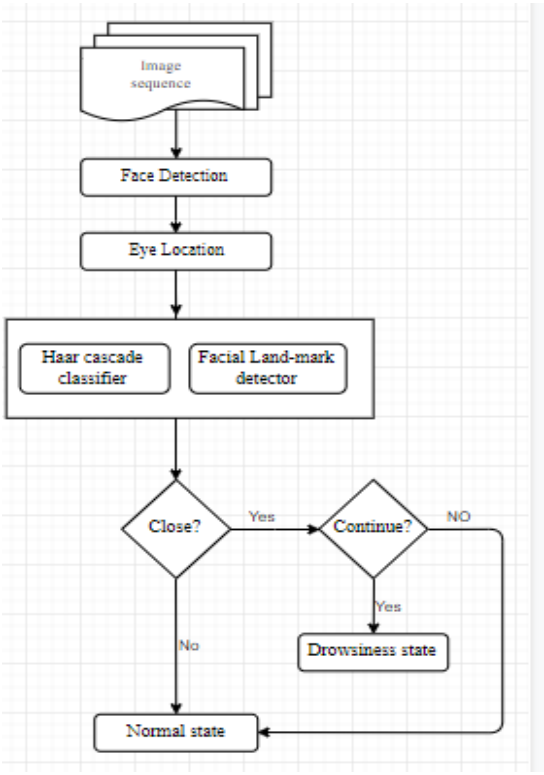


Fig.3.3**:** Flowchart of system

II. FLOWCHART AND ALGORITHM:

The various detection stages are discussed as:

3.3.2) 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. 3.4 represents five haar like features & example is shown in Fig.3.5

Fig. 3.4

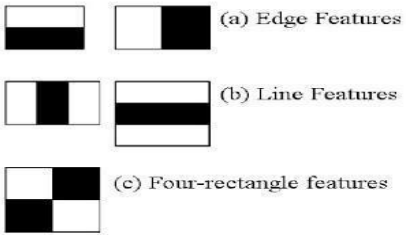


Fig. 3.5

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 nonface samples can be rejected

3.3.3) 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 labeled facial landmarks on an image. These images are manually labeled, 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:

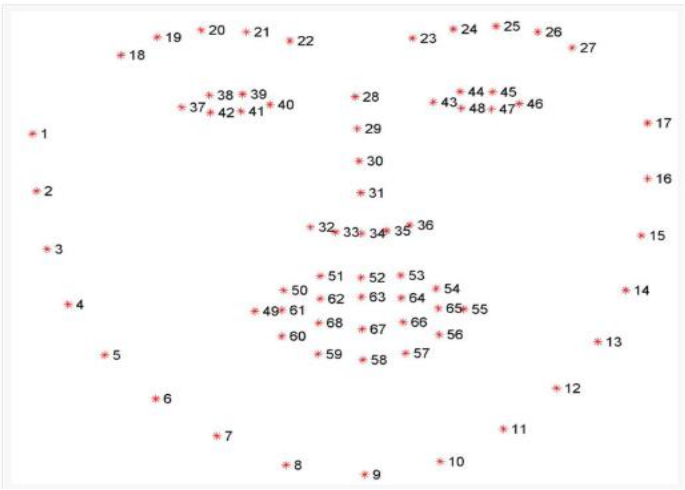


Fig.3.6: Visualizing the 68 facial landmark coordinates

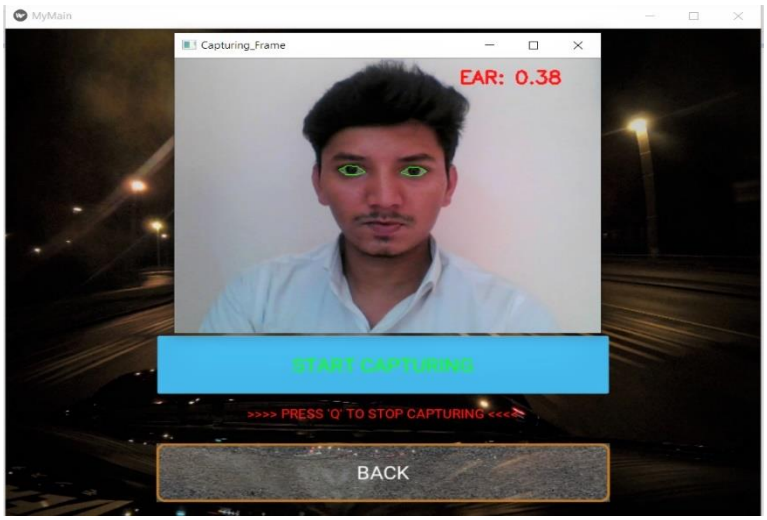


Fig.3.7: Detection of both the eyes

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 flavors 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.

3.3.4) 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 Aspected 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.

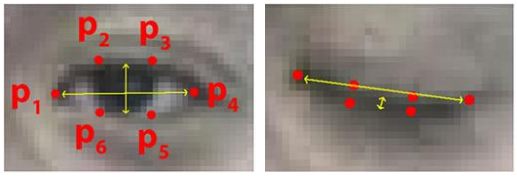


Fig.3.8

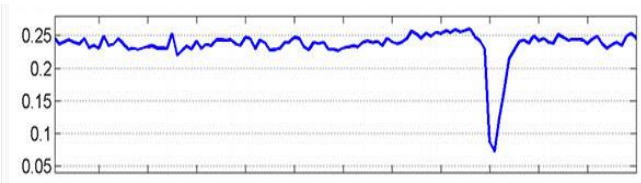


Fig.3.9 EAR for single blink

Fig 3.8: 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.

User flexibility Set up Features: This interface helps the user to adjust the EAR accordingly ranging from low to high percent. It also allows the user to set the timer sensitivity shown in the fig 3.10.

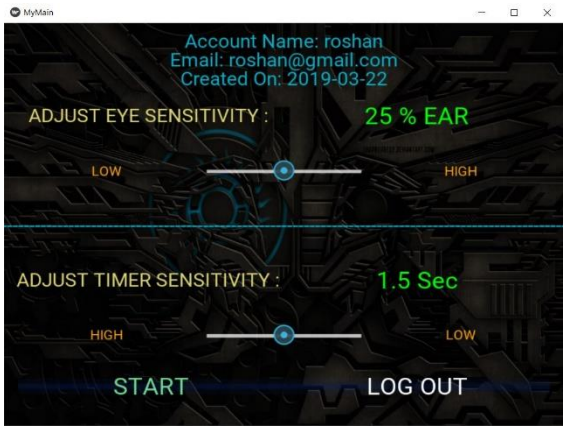


Fig. 3.10

3.3.5) 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”.

3.3.6) 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.

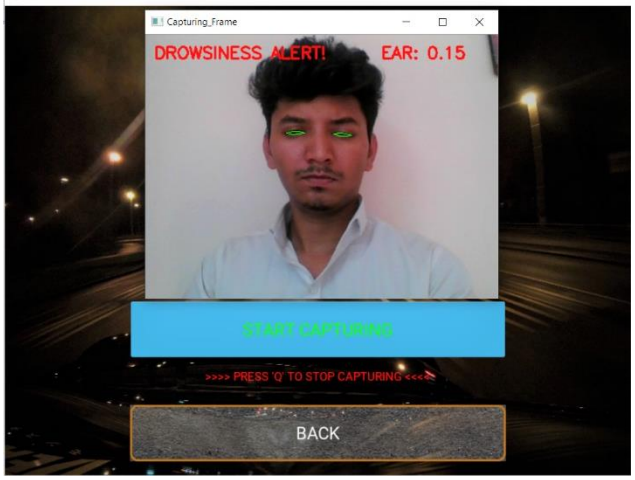
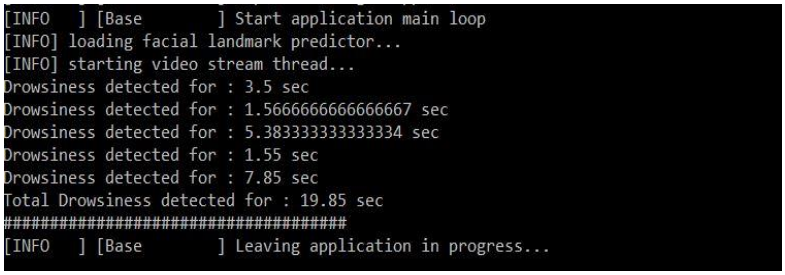


Fig.3.11 Drowsiness state

Fig .3.12 Console information



Chapter 4

Modelling diagram (project Work Flow)

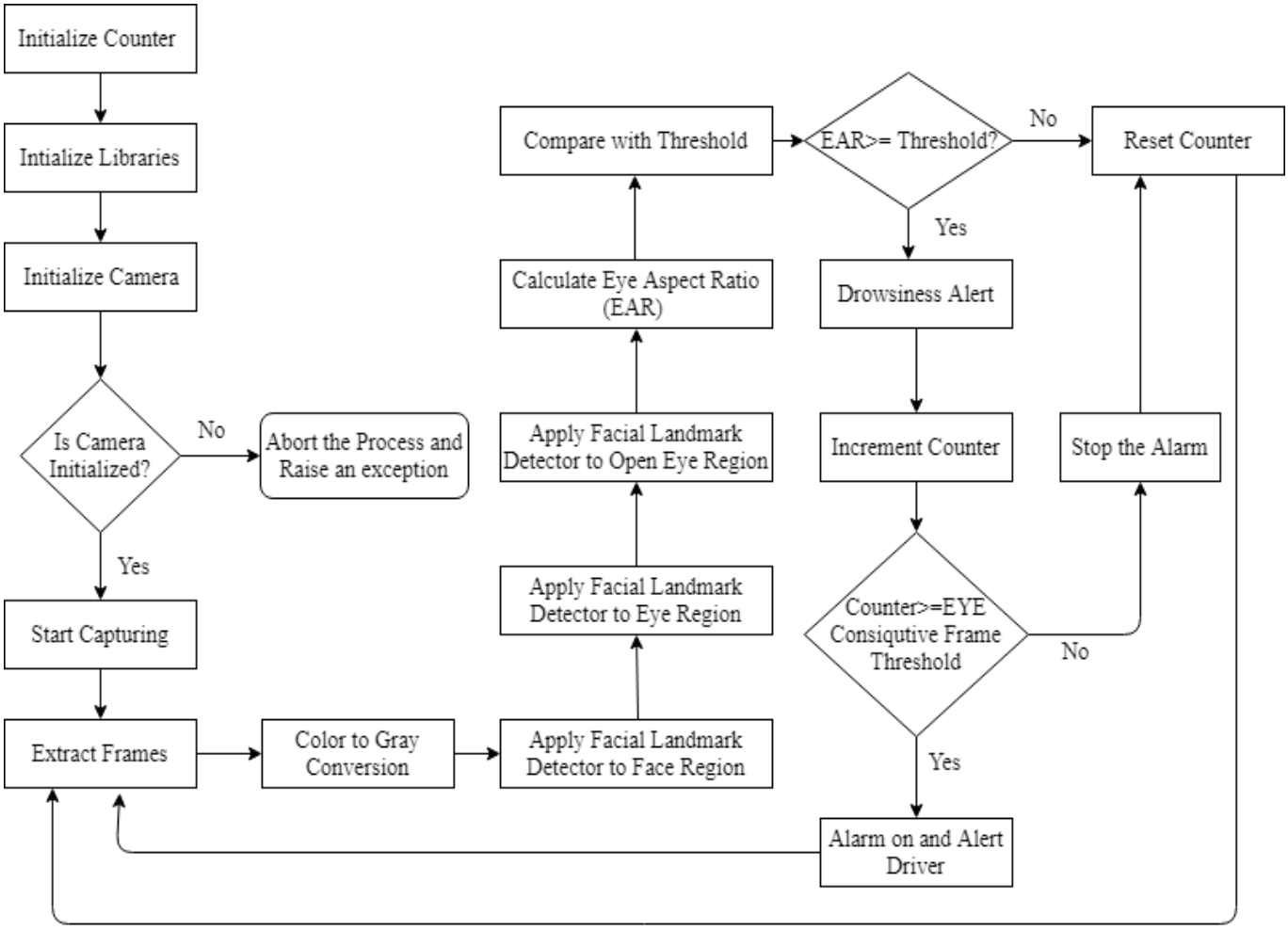


Fig. 4.1 Flow chart of System

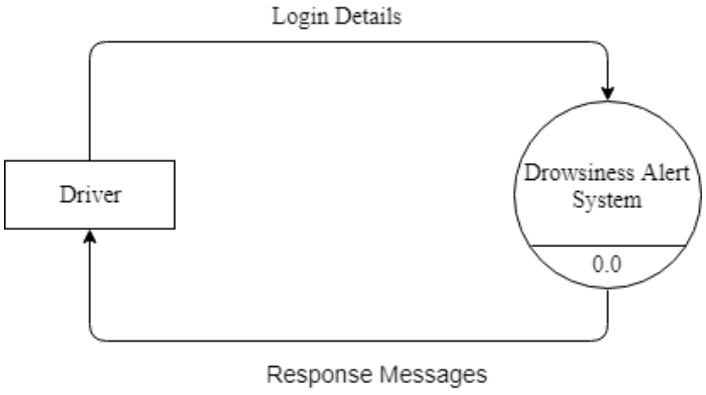


Fig.4.2 DFD level 0

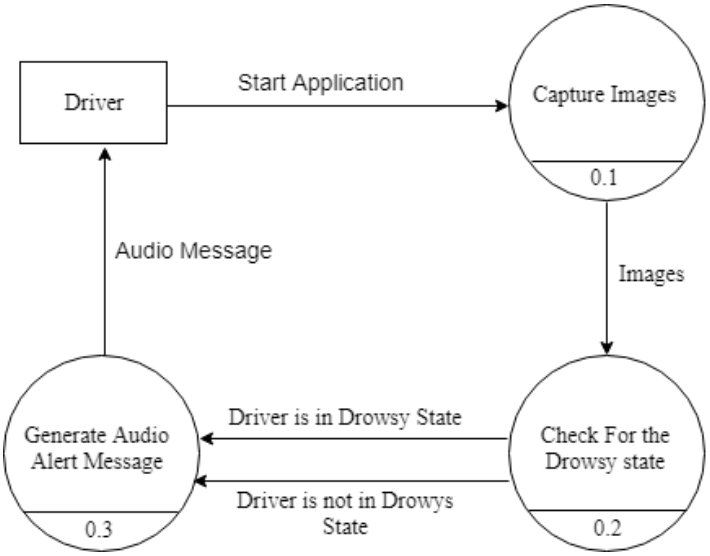


Fig 4.3 DFD level 1

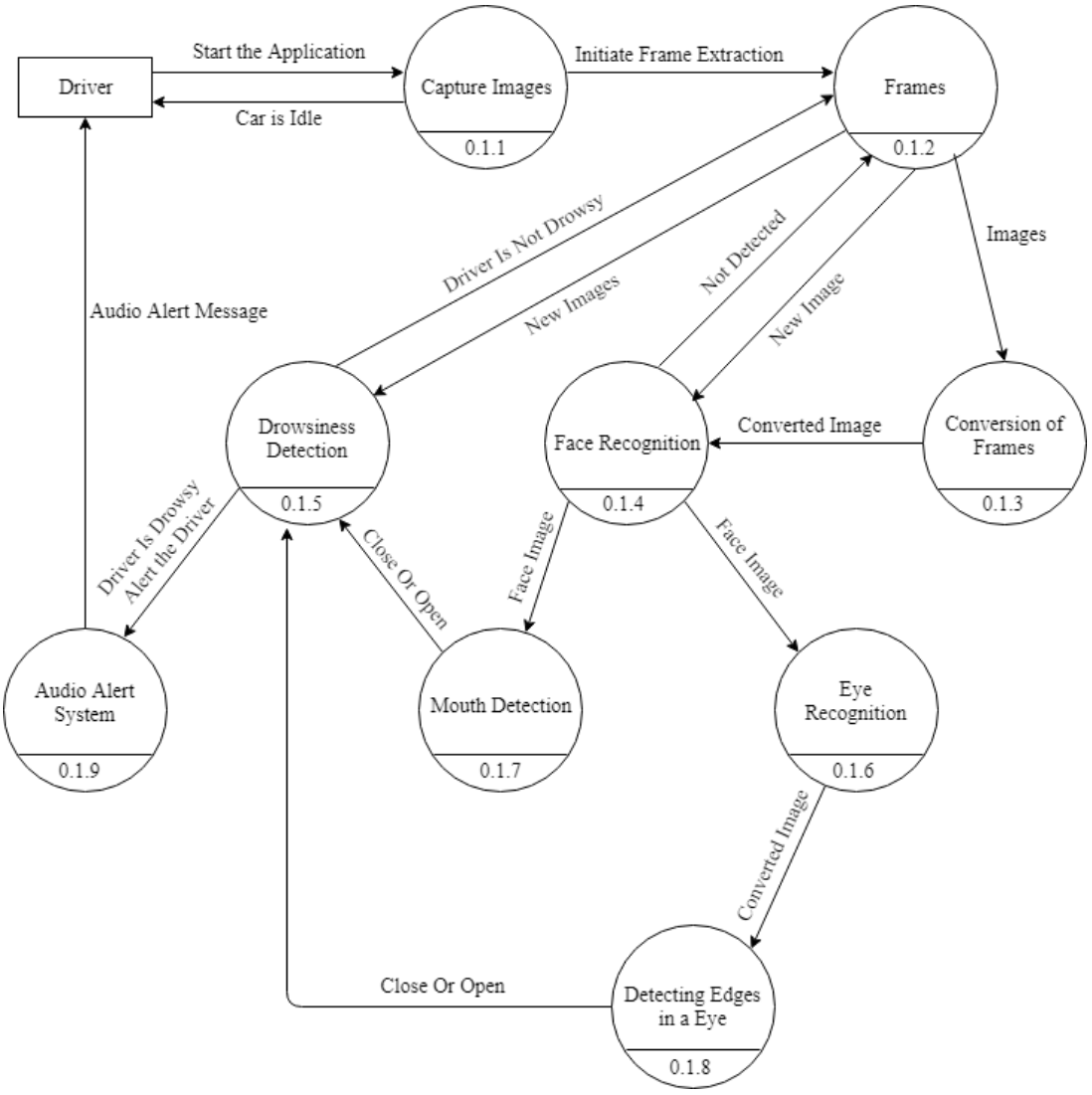


Fig 4.4 DFD level 2

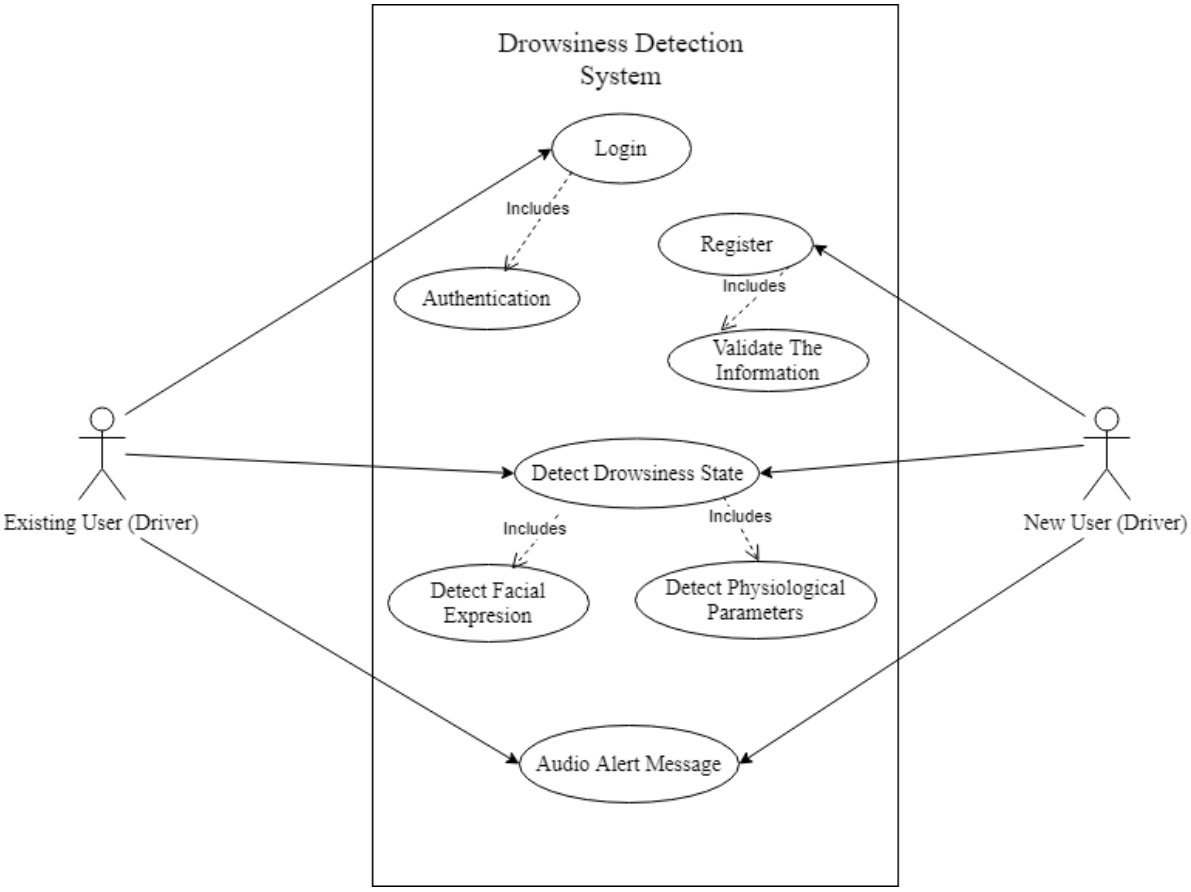


Fig 4.5 use-case Diagram

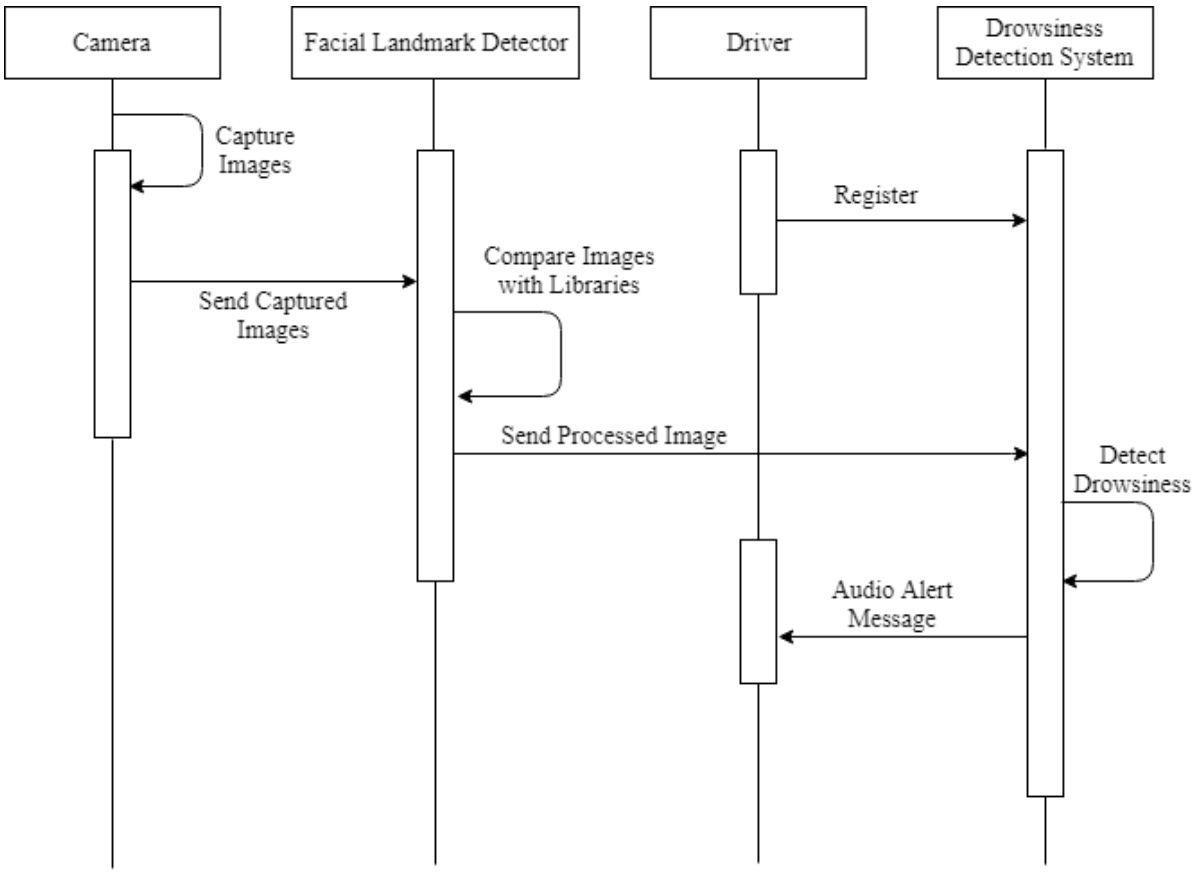


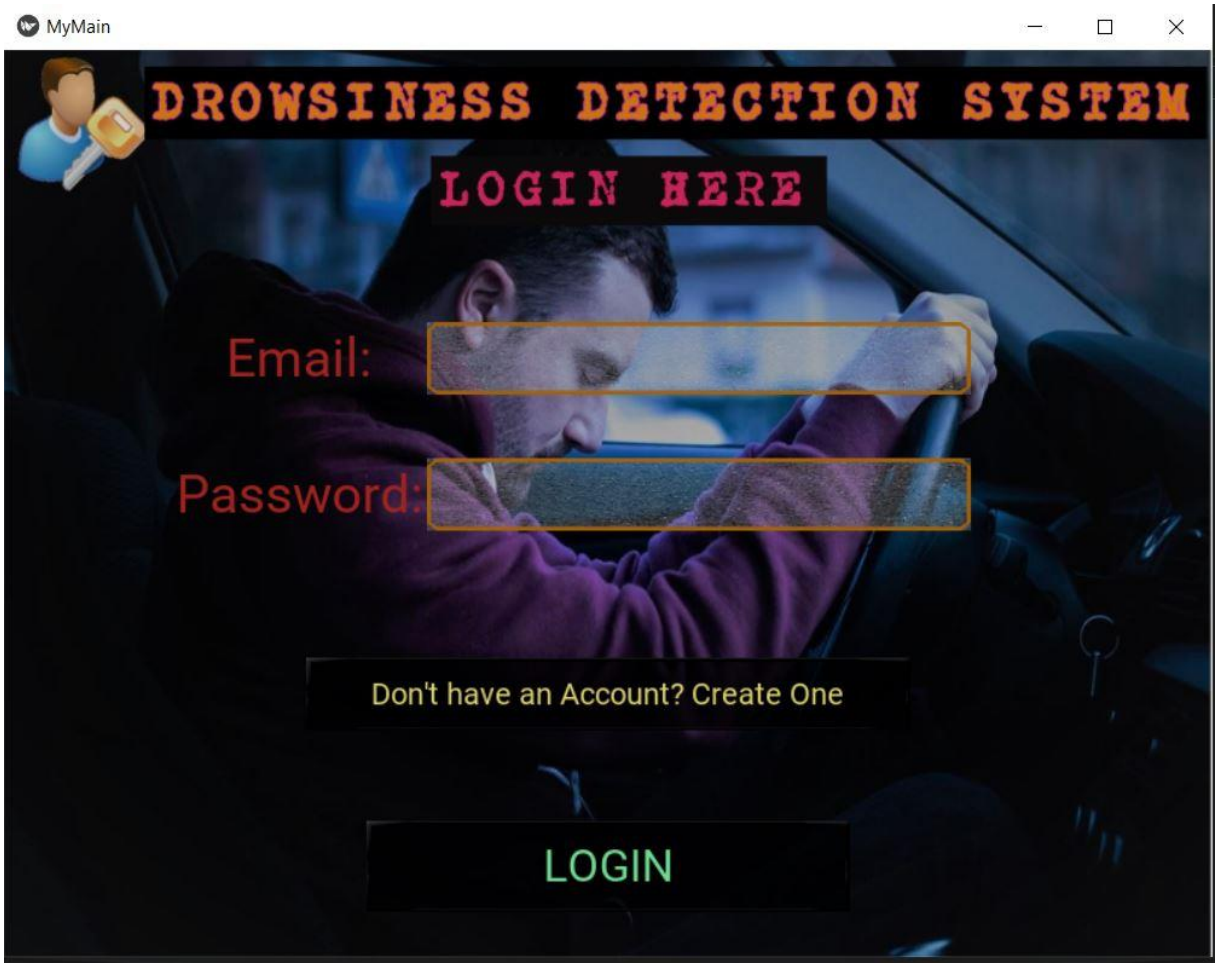
Fig 4.6 Sequence Diagram

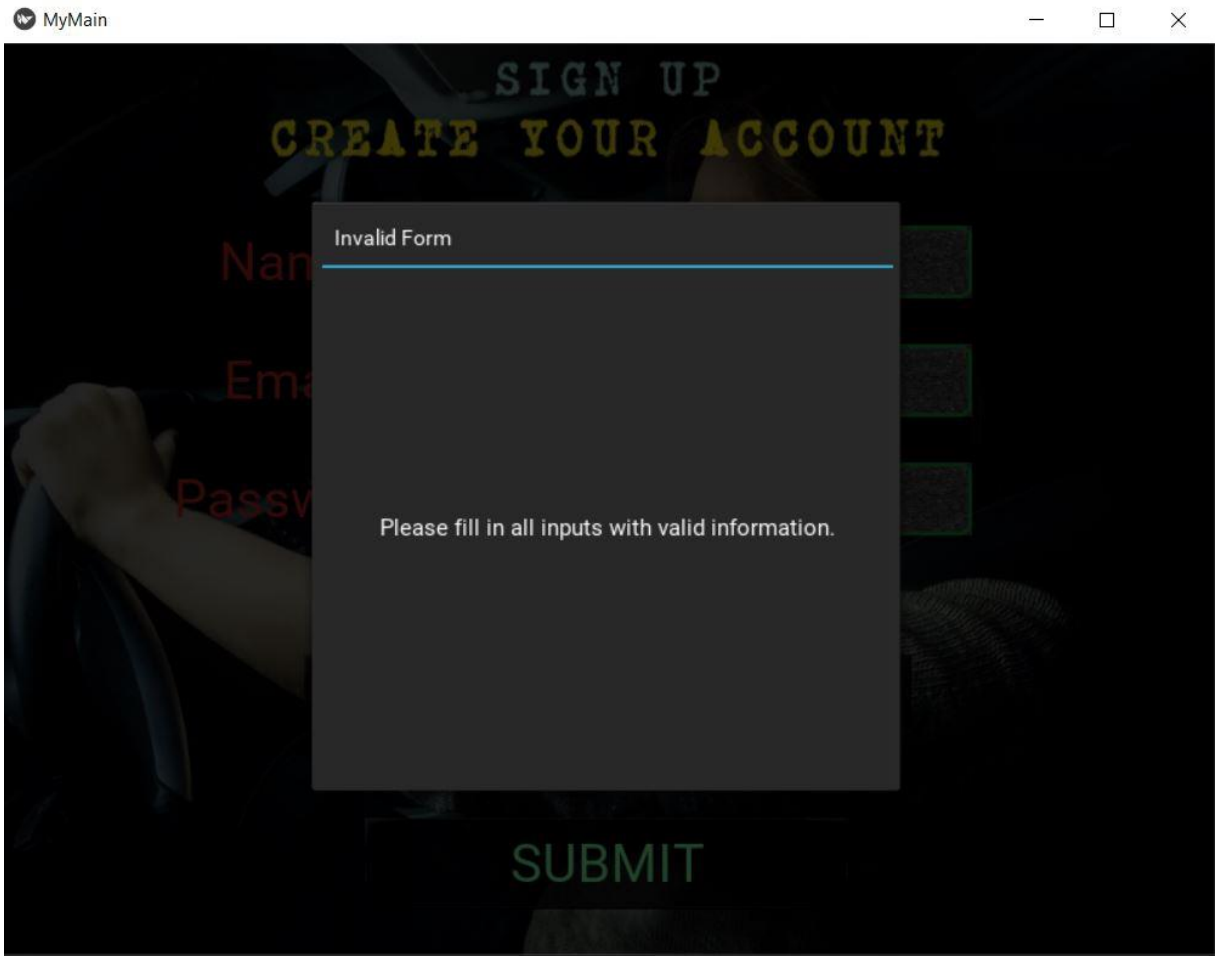
Chapter 5

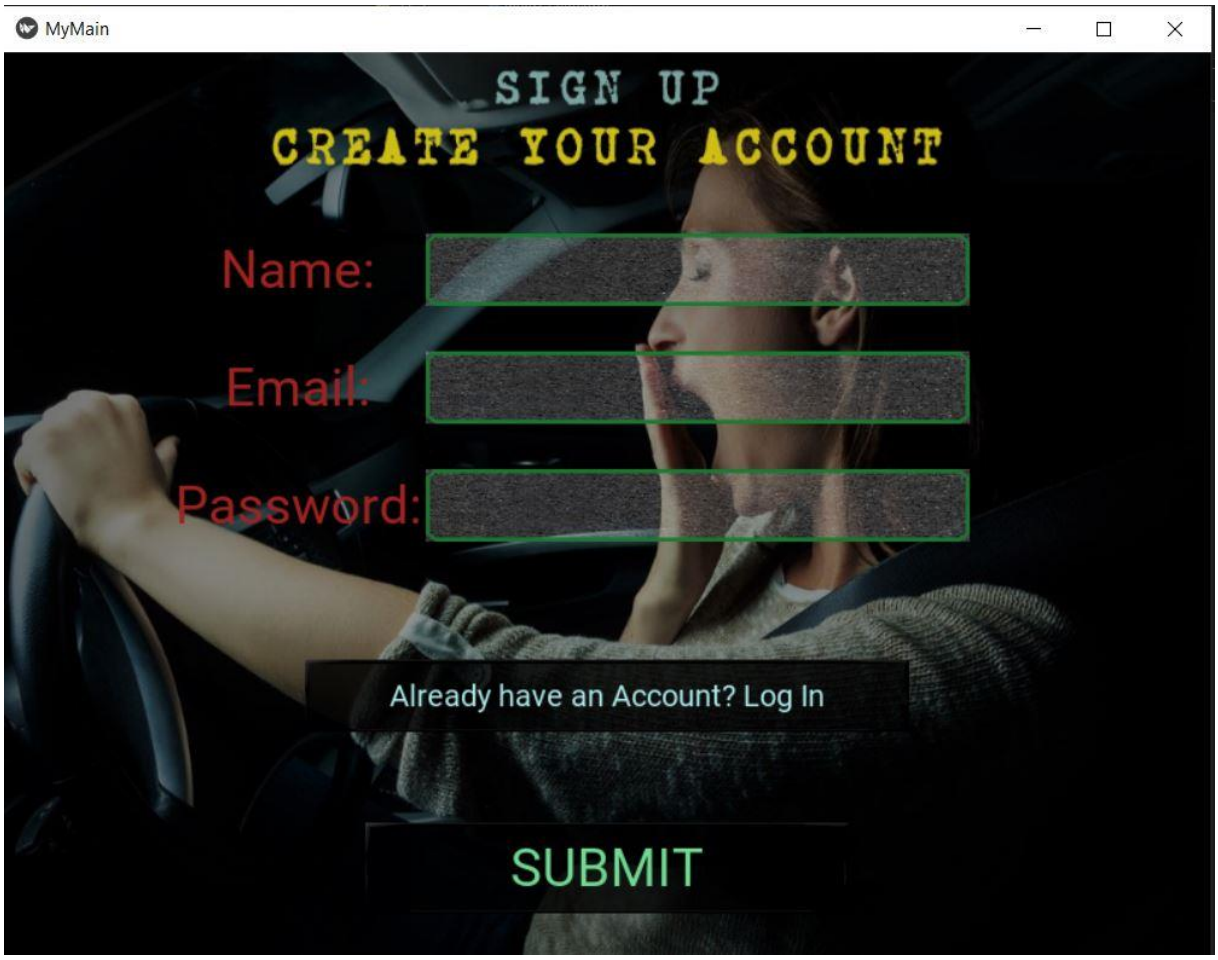
Results and Discussions

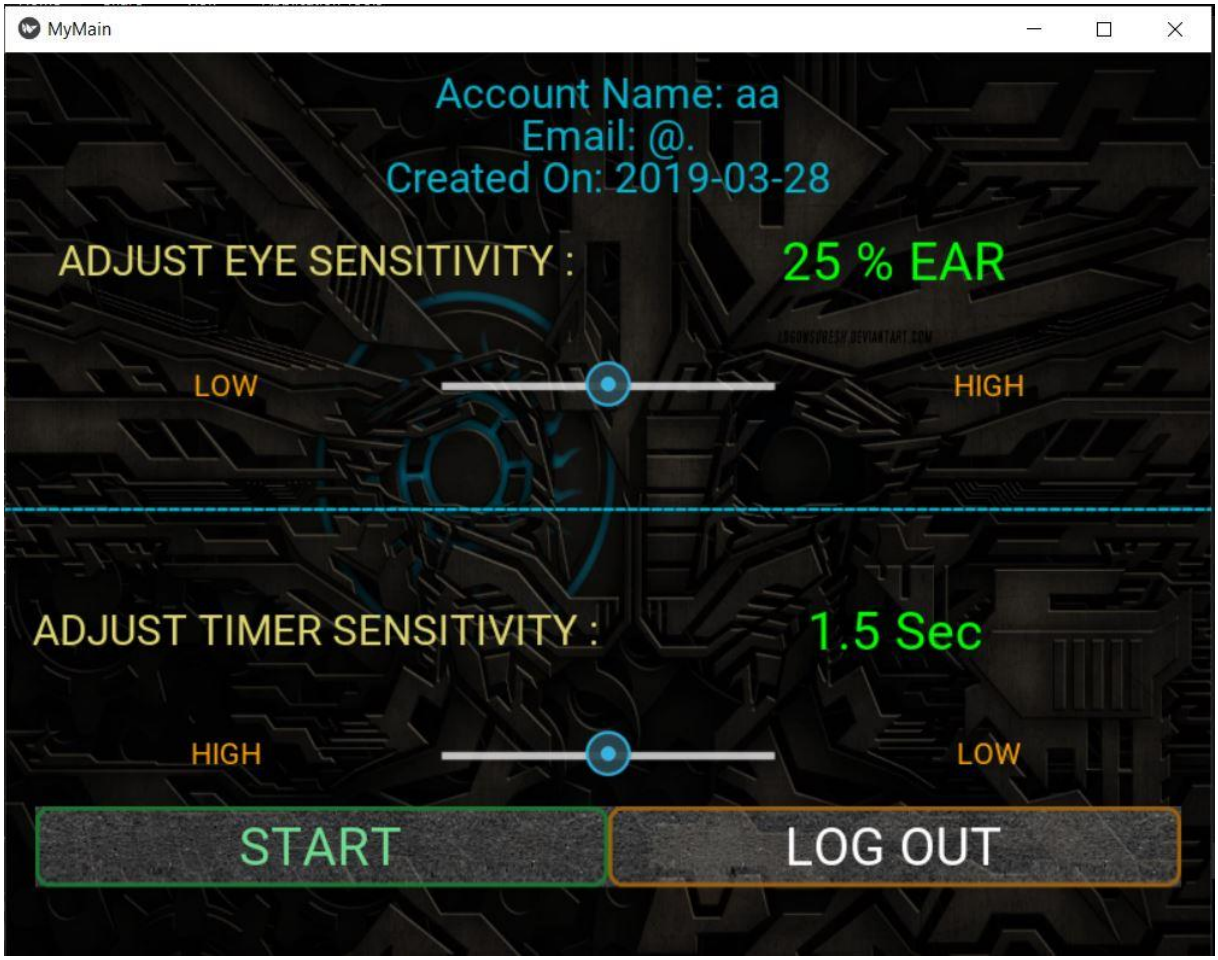
Implementation of drowsiness detection with Python and OpenCV was done which includes the following steps: Successful runtime capturing of video with camera. Captured video was divided into frames and each frame were analyzed. Successful detection of face followed by detection of eye. If closure of eye 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 analyzing 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.

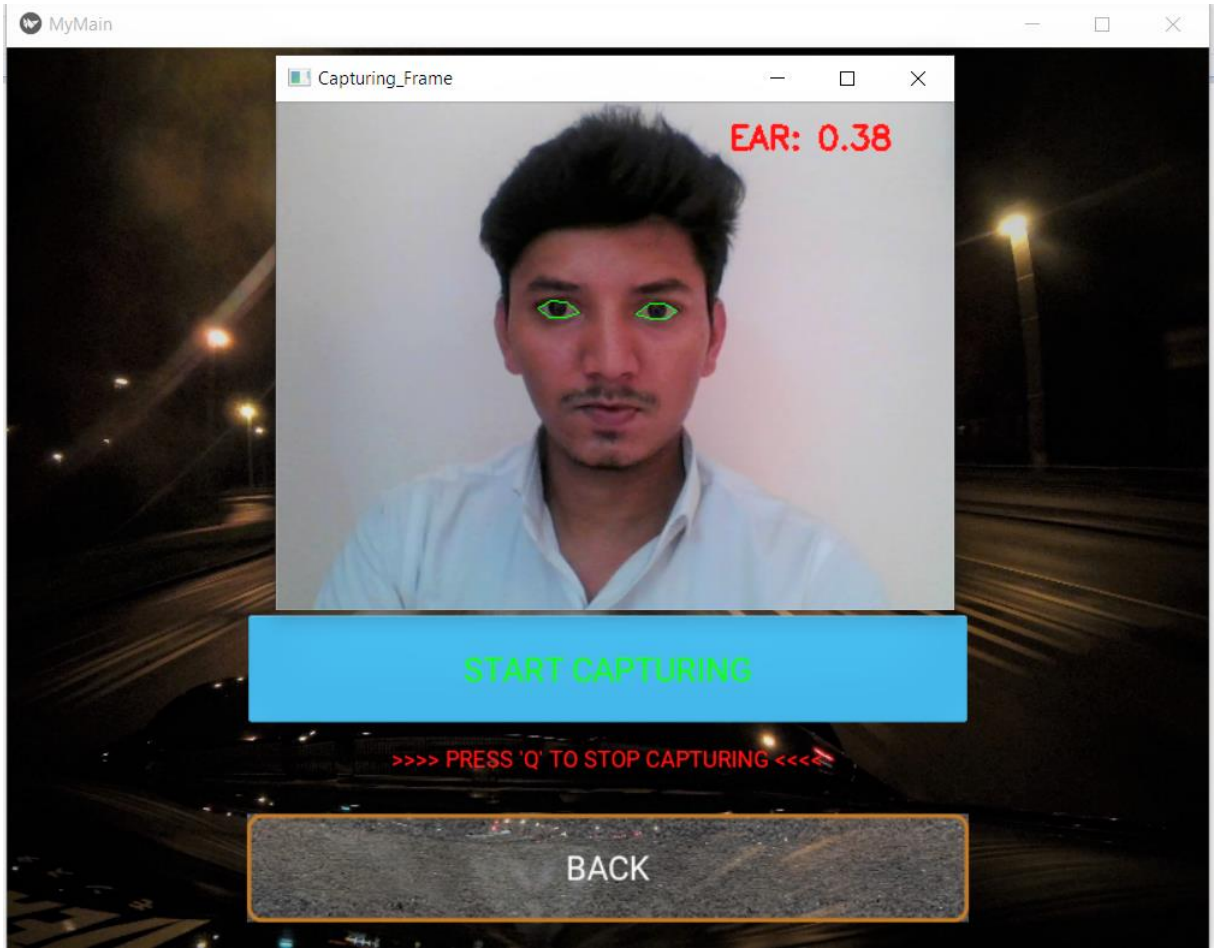
Screenshots of the system shown below:

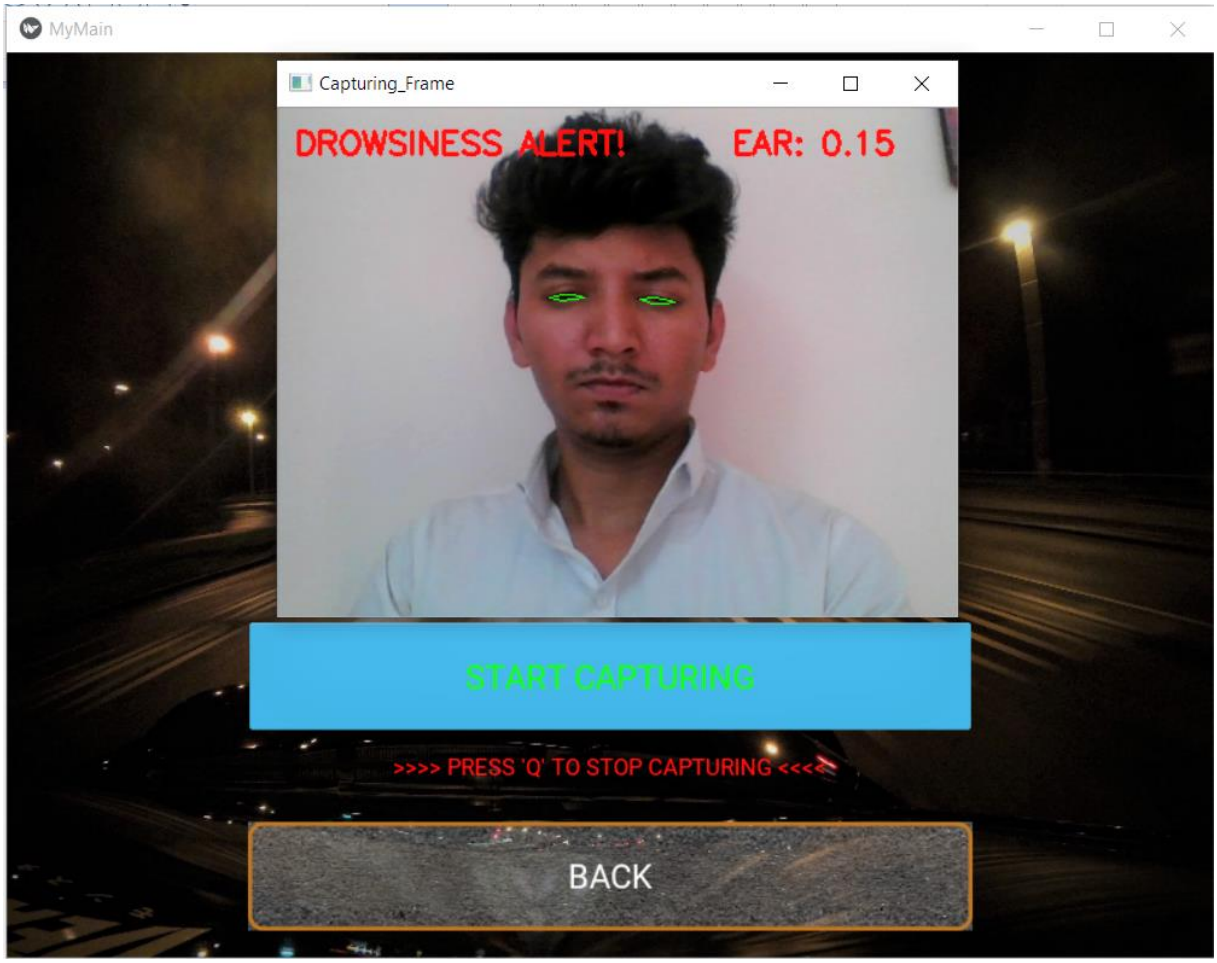



MyMain
SIGN UP
CREATE YOUR ACCOUNT
Invalid Form
Please fill in all inputs with valid information.
SUBMIT


MyMain
SIGN UP
CREATE YOUR ACCOUNT
Name:
Email:
Password:
Already have an Account? Log In
SUBMIT


MyMain
Account Name: aa
Email: @.
Created On: 2019-03-28
ADJUST EYE SENSITIVITY :
25 % EAR
LOW
HIGH
ADJUST TIMER SENSITIVITY :
1.5 Sec
HIGH
LOW
START
LOG OUT


MyMain
Capturing_Frame
EAR: 0.38
START CAPTURING
>>>> PRESS 'Q' TO STOP CAPTURING <<<<
BACK




```
[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...
```

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

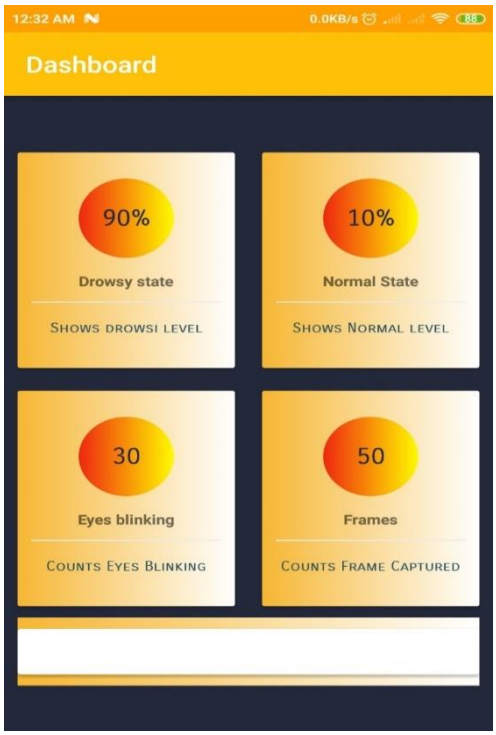


Fig 5.1 Android Application for Future scope

Chapter 6

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-the-wild.

Limitations:

Use of spectacles: In case the user uses spectacle 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.

Chapter 7

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-Human-Eyes- 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-Of-HumanEyesUsing-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/

Appendix

1.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 playsound
        import argparse
        import imutils
        import time
        import dlib
        import cv2
```

Sound Alarm

```
        def sound_alarm(path):
# play an alarm sound
        playsound.playsound(path)
```

eye_aspect_ratio function

```
        def eye_aspect_ratio(eye):
```

```
# compute the euclidean distances between the two sets of
# vertical eye landmarks (x, y)-coordinates
        A = dist.euclidean(eye[1], eye[5])
        B = dist.euclidean(eye[2], eye[4])

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

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

# return the eye aspect ratio
        return ear
```

Parsing command Line Argument

```
        # construct the argument parse and parse the arguments
        ap = argparse.ArgumentParser()
        ap.add_argument("-p", "--shape-predictor", required=True,
        help="path to facial landmark predictor")
        ap.add_argument("-a", "--alarm", type=str, default="",
        help="path alarm .WAV file")
        ap.add_argument("-w", "--webcam", type=int, default=0,
        help="index of webcam on system")
        args = vars(ap.parse_args())
```

Defining EYE_AR_THRESH

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

Facial landmark predictor

```
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
        print("[INFO] loading facial landmark predictor...")
        detector = dlib.get_frontal_face_detector()
        predictor = dlib.shape_predictor(args["shape_predictor"])
```

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"]
```

Instantiate VideoStrem

```
# start the video stream thread
        print("[INFO] starting video stream thread...")
        vs = VideoStream(src=args["webcam"]).start()
        time.sleep(1.0)
 # loop over frames from the video stream
    while True:
        frame = vs.read()
        frame = imutils.resize(frame, width=450)
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
 # detect faces in the grayscale frame
       rects = detector(gray, 0)
```

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

```
# loop over the face detections
    for rect in rects:
# determine the facial landmarks for the face region, then
# convert the facial landmark (x, y)-coordinates to a NumPy
# array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)


# extract the left and right eye coordinates, then use the
# coordinates to compute the eye aspect ratio for both
eyes
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
# average the eye aspect ratio together for both eyes
     ear = (leftEAR + rightEAR) / 2.0
```

Visualize each of the eye regions

```
# compute the convex hull for the left and right eye, then
   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:

```
# check to see if the eye aspect ratio is below the blink
# threshold, and if so, increment the blink frame counter
        if ear < EYE_AR_THRESH:
        COUNTER += 1

# if the eyes were closed for a sufficient number of
# then sound the alarm
        if COUNTER >= EYE_AR_CONSEC_FRAMES:
# if the alarm is not on, turn it on
        if not ALARM_ON:
        ALARM_ON = True

# check to see if an alarm file was supplied,
# and if so, start a thread to have the alarm
# sound played in the background
        if args["alarm"] != "":
        t = Thread(target=sound_alarm,
        args=(args["alarm"],))
        t.deamon = True
        t.start()

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

# otherwise, the eye aspect ratio is not below the blink
# threshold, so reset the counter and alarm
        else:
        COUNTER = 0
        ALARM_ON = False
```

Displaying the output frame:

```
# draw the computed eye aspect ratio on the frame to help
# with debugging and setting the correct eye aspect ratio
# thresholds and frame counters
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
 # show the frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
 # if the `q` key was pressed, break from the loop
        if key == ord("q"):
        break
 # do a bit of cleanup
        cv2.destroyAllWindows()
        vs.stop()
```

Acknowledgements

I am profoundly grateful to **Dr. Varsha Shah** for his expert guidance and continuous encouragement throughout to see that this project rights its target.

I would like to express deepest appreciation towards Dr. Varsha Shah, Principal RCOE, Mumbai and Prof.Shiburaj Pappu HOD Computer Department whose invaluable guidance supported me in this project.

At last I must express my sincere heartfelt gratitude to all the staff members of Computer Engineering Department who helped us directly or indirectly during this course of work.

Roshan Tavhare