DRIVER DROWSINESS DETECTION SYSTEM [DRIVER ALERT]

A

Project Work

Submitted as Minor Project in Partial fulfillment for the award of Graduate Degree in Bachelor of Engineering in Computer Science & Engineering.

Submitted to

RAJIV GANDHI PROUDYOGIKI VISHWAVIDYALAYA BHOPAL (M.P)

Submitted By--

ANSHUL VERMA (0105CS191023) [TEAM LEADER]
HARSHIT SHRIVASTAVA (0105CS191048)
MANISH NATHRANI (0105CS191062)

Under the Guidance of -

PROF. GOLDI JARBAIS

(Department of Computer Science & Engineering)

Oriental Institute of Science & Technology, Bhopal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

JULY-DEC 2021

Oriental Institute of Science & Technology, Bhopal

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**DRIVER DROWSINESS DETECTION SYSTEM [DRIVER ALERT]**" being submitted by

ANSHUL VERMA (0105CS191023) [TEAM LEADER],

HARSHIT SHRIVASTAVA (01015CS191048) and

MANISH NATHRANI (0105CS191062) students of **5th** Semester, B. Tech in Computer Science & Engineering have done their work as MINOR PROJECT-I for Partial fulfillment of the B.Tech degree from RGPV, Bhopal (M.P.) is a record of Bonafede work carried out by them/him/her under our supervision.

Guide: Prof. Goldi Jarbais
Department of Computer Science&
Engineering

Head
Department of Computer Science &
Engineering

ACKNOWLEDGEMENT

We take the opportunity to express our cordial gratitude and deep sense of indebtedness to our **Guide** for the valuable guidance and inspiration throughout the project duration. We feel thankful to her for innovative ideas, which led to successful submission of this minor project work. We feel proud and fortune to work under such an outstanding mentor in the field of Via Store. She has always welcomed our problems and helped us to clear our doubts. We will always be grateful to her for providing us moral support and sufficient time.

We owe sincere thanks to **Director OIST**, for providing us with moral support and necessary help during our project work in the Department.

At the same time, we would also like to thank **HOD, CSE** and all other faculty members and all non-teaching staff of department of Computer Science &Engineering for their valuable co-operation.

We would also thank to our Institution, faculty members and staff without whom this project would have been a distant reality. We also extend our heartfelt thanks to our family and well-wishers.

ANSHUL VERMA
0105CS191023

HARSHIT SHRIVASTAVA
0105CS191048

MANISH NATHRANI
0105CS191062

APPROVAL CERTIFICATE

This is to certify that the project entitled Driver Drowsiness Detection System [Driver Alert], being submitted by -

ANSHUL VERMA (0105CS191023) [TEAM LEADER],

HARSHIT SHRIVASTAVA (0105CS191048) and

MANISH NATHRANI (0105CS191062)

students of 5th Semester, Bachelor of Technology in Computer Science and Engineering have done their work as Minor Project for Partial fulfillment of the degree from RGPV, Bhopal (M.P.).

Guide Name:

Date:

Signature:

ABSTRACT

Nowadays, more and more professions require long-term concentration. Drivers must keep a close eye on the road, so they can react to sudden events immediately. Driver fatigue often becomes a direct cause of many traffic accidents. Therefore, there is a need to develop the systems that will detect and notify a driver of her/him bad psychophysical condition, which could significantly reduce the number of fatigue-related car accidents. However, the development of such systems encounters many difficulties related to fast and proper recognition of a driver's fatigue symptoms. One of the technical possibilities to implement driver drowsiness detection systems is to use the vision-based approach. The technical aspects of using the vision system to detect a driver drowsiness are also discussed.

Drowsiness and Fatigue of drivers are amongst the significant causes of road accidents. Every year, they increase the amounts of deaths and fatalities injuries globally. In this paper, a module for Advanced Driver Assistance System (ADAS) is presented to reduce the number of accidents due to drivers fatigue and hence increase the transportation safety; this system deals with automatic driver drowsiness detection based on visual information and Artificial Intelligence. We propose an algorithm to locate, track, and analyze both the drivers face and eyes, a scientifically supported measure of drowsiness associated with slow eye closure.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 MACHINE LEARNING

Machine learning is Associate in drowsiness detection of computer science (AI) that gives systems the flexibility to mechanically learn and improve from expertise while not being expressively programmed. The most probable difference between Python and programming is largely in programming we have a tendency to use conditional statements to expressively tell the program to figure victimization those conditions whereas, Python mechanically learns and improves from expertise primarily the user ought to train the dataset and also the machine language learns mechanically by means independently. Python focuses on the event of pc programs which will access knowledge and use it learn for themselves. The process of learning begins with observations or knowledge, like examples, direct expertise, or instruction, so as to seem for patterns in knowledge and create higher choices in the future supported the examples that we offer. This type of process of learning is predominantly classified as supervised learning. The main objective of this type of learning is that the system already knows its output it's just the system structures itself towards the output in this type of learning also shows that the system learns even through feedback too. The essential point is to permit the PCs adapt naturally without human intercession or help and change activities likewise. Python algorithms are classified as:

1. Supervised ML Algorithms these algorithms can utilize what has been realized in the past to the new information being utilized by utilizing hailed models which will be useful in future scopes. Beginning with preparing the dataset while preparing the dataset the AI calculation sets up hindrances like fundamental capacities to make expectation of the yield esteems for future scopes by setting up these impediments the calculation consequently. takes in and develops itself from these obstructions. This sort of Learning needs an adequate measure of preparing to anticipate yields and to give yields to the particular new sources of info. This sort of learning calculation contrasts it's yield and the ideal yield and redresses itself and learns without anyone else by discover blunders and changing them.

2. Unsupervised Python Algorithms these algorithms square measure utterly completely different from supervised Python algorithms these algorithms square measure used once the dataset to coach is neither classified nor a flagged. So, the dataset ought to be cleansed before coaching it. knowledge clean-up is largely classifying and labelling the dataset. during this sort the system doesn't turn out the proper output however investigation {the knowledge info the information} set and attracts conclusion on the idea of the data clean up done before coaching the dataset and classifying and tired examples and describe the hidden structures from the dataset. unsupervised learning is that the coaching of machine mistreatment info that's neither classified nor labelled and permitting the rule to act on its info while not steerage. Here the task of machine is to cluster unsorted info per similarities, patterns and variations with none previous coaching of knowledge. the most distinction between supervised and unsupervised algorithms is that supervised learning is well-versed steerage whereas unsupervised learning isn't.

3. Semi-administered Learning it falls among regulated and solo learning this type of learning uses both labelled and unlabeled data. Typically, larger proportionate of un labelled data. Semi supervised learning requires both the types of data as when the labelled data requires more resources which are totally dependent on unlabelled data.

4. Reinforcement Learning is a type of dynamic learning this trains the algorithm using a system of reward and punishment it interacts with its environment to learn. This is inspired behavioral psychology it is similar to supervised learning but as in supervised learning it is explicitly told how to perform a task and what the desired outcome will be. As, in this type of learning it works through the problem on its own and finds a way to perform the task and get the desired.

1.2 PROBLEM STATEMENT

On road driver's fatigue and drowsiness is contributing more than 30%[1] of reported road accidents. Driver drowsiness can be estimated by monitoring biomedical signals, visual assessment of driver's bio-behavior from face images, by monitoring drivers performance or by combines all above techniques. Proposed algorithm is based on live monitoring of EAR (Eye aspect Ratio) by application of Image processing. HD live video is decomposed in continues frames and facial landmarks has been detected using pre trained Neural Network based Dlib functions. Dlib functions are trained using Cascading algorithm. Intel's Open

source Image processing libraries (OPEN CV) is used as primary Image processing tool. Python Language is used as main codding language.

1.3 PROJECT PURPOSE

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 blinking to monitor drowsiness or consider physical changes such as sagging 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.

Driver in-alertness is an important resulting from sleep deprivation or sleep disorders and is an important factor in the increasing number of the accidents on today's roads. Drowsy driver warning system can form the basis of the system to possibly reduce the accidents related to driver's drowsiness. In a year there were 824 fatalities recorded in NHTSA's FARS database that were drowsy driving-related. According to the results of the study presented at the International Symposium on Sleep Disorders, fatigue of drivers is responsible for30% of road accidents. Different techniques are used in driver-fatigue monitoring systems. These techniques are divided into four categories. The first category's includes intrusive techniques, which are mostly based on monitoring biomedical signals, and therefore require physical contact with the driver. The second category includes non-intrusive techniques based on visual assessment of driver's bio-behavior from face images. The third category includes methods based on driver's performance, which monitor vehicle behavior such as moving course, steering angle, speed, braking, etc. Finally, the fourth category combines techniques from the above mentioned three categories. The computer vision-based techniques from the second category are particularly effective, because the drowsiness can be detected by observing the facial features and visual bio-behavior such as head position, gaze, eye openness, eyelid movements, and mouth openness. Proposed algorithm is based on computer vision method. The main focus is on the detection of blinks by estimating the EAR (Eye aspect Ratio). This is achieved by monitoring the eyes of the driver throughout the entire video sequence. An IR camera will be used for capturing live video of driver eyes in all light conditions and frames will extracted for image processing scheme of video capturing.

1.4 PROJECT FEATURES

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 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.

1.5 MODULES DESCRIPTION

1.5.1 DATA COLLECTION

Data plays the most important role in this project. The data has to be genuine and trustworthy. Data has to be collected from the right sources and each record must have all the necessary fields to perform the analysis. It can either be procured from online sources or collected manually to attain more accuracy, but is a very tedious task. Once data is collected, it needs to be verified for relevance before finalizing as valid data for the project. The accuracy of the outcome is directly proportional to the size of the dataset. i.e. Bigger the size of the dataset, more is the accuracy. This is because the machine learning algorithms need a lot of data to work with, to identify patterns and learn on its own.

1.5.2 APPLYING ALGORITHMS

This module has two parts to it- identifying the algorithms and applying the algorithms. Identification plays a key role in this process and in the project as a whole. Identifying the right algorithm may be done through research and mostly trial and error. We pick the algorithms which yield the most accurate results and zero in on them. These algorithms are then applied to the data set using Spyder IDE. The identification, application and verification phase of the system go hand in hand, all three steps need to be executed for each algorithm to decide on which one is the most efficient.

CHAPTER 2

LITERATURE SURVEY

2.1 DATA MINING

Data mining is the process of sorting the large data sets to establish relationships and identification of patterns to solve real world problems through data analysis. Companies use data mining to turn the raw data into the information which can be useful for them. It mainly depends on computer processing, warehousing, and collection of effective data. Data mining is used in various different ways such as fraud detection, identify spam email, and database marketing.

The process of data mining is broken down into five steps. First it collects the data loads them into their own data warehouses. Second, they manage the data and store them either on the cloud or on in-house severs. Then, software applications sorts the data based on the user requirements and then, finally the user on the end displays the data in an easy manner like table or graph.
Various researches on this process have led to formation of different data mining algorithms. We can use these algorithms directly on a dataset to create some models and get the inferences from the dataset. Popular data mining algorithms are Nearest Neighbor (CNN), Naive Bayes, Random forest, Logistic regression, Decision tree, and Support vector machines.

1. Nearest neighbor (CNN) is a managed machine calculation which can be utilized to unravel a wide range of relapse and arrangement issues.

2. Naive Bayes calculation is a group of probabilistic calculations that exploits the likelihood hypothesis and Bayes hypothesis to anticipate the tag of a book.

3. Logistic relapse is a characterization calculation used to allot perception to set of discrete classes. A portion of the characterization issues are email spam, online misrepresentation identification, tumor harmful or kind. Calculated relapse essentially changes its yield utilizing the strategic sigmoid capacity to restore likelihood esteem.

4. Decision tree calculation goes under the class of directed learning. This calculation can be utilized to explain both the relapse and grouping issues. The choice tree utilizes tree portrayal to take care of the issues.

5. A support vector machine is an administered AI calculation which can be utilized for both relapse and grouping issues. In help vector machine, we plot every datum thing as a point in N-dimensional space (where n is the quantity of highlights) with the estimation of each component being a similar estimation of specific organizations.

6. Random timberland calculation is a technique that works by building numerous choice trees during preparing stage. The choice of greater part of the trees is picked by the arbitrary backwoods as an official conclusion. It utilizes include haphazardness and stowing when assembling an individual tree to attempt to make a woods of trees whose forecast is more exact and precise than that of any individual tree.

2.2 EXISTING SYSTEM

Images or the real time video is captured from the camera installed in front of the driver's face. This video is converted into number of frames. OpenCV face OpenCV-classifier is loaded. Each frame is compared with the pre-defined features of the OpenCV-classifiers. when the features are matched the face is detected and a rectangle is drawn around the face. Using feature extraction we estimate the position of the eyes. By comparing with the. OpenCV eye-OpenCV classifier, the eyes are detected and rectangles are drawn around left and right eye.

Manu B.N in 2016, has proposed a method that detect the face using OpenCV 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 using the fea with the
OpenCV 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, OpenCV 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). In-vehicle camera is commonly installed to realize the possible reasons of car accidents. Such a camera can also be used to detect the fatigue of the driver. Several studies related to the fatigue detection are described as follows. Sharma et al. [3] utilized the number of pixels in the eye image to determine the eye state, open or close. Hornget at. [4] established an edge map to locate the eyes locations and the eye state is determined based on the HSL color space of the eye image. Its accuracy is dependent on the location of the eyes. Sharma and Banga [3] converted the face image to YCbCr color space. The average and standard deviation of the pixel number in the binarization image is computed. Then, fuzzy rules [5] are used to determine the eye state. Liu et al. [6] and Tabrizi et al.

2.3 PROPOSED SYSTEMS

2.3.1 FACE AND EYE DETECTION BY OPENCV ALGORITHMS

In this paper a novel approach to critical parts of face detection problems is given, based on analogic cellular neural network (OpenCV) algorithms. The proposed OpenCV algorithms find and help to normalize human faces is, 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 point reference. Other OpenCV algorithm finds the eyes on any grayscale image by searching characteristic is features of the eyes and eye sockets. Tests made on a standard database show that the algorithm works very fast and it is reliable. In proposed method, first the image is acquired by the webcam for processing. The images of the driver are captured from the camera which is installed in front of the driver on the car dashboard. It will be passed to preprocessing which prepares the image for further processing by the system. Its main operations are to eliminate noises caused by the image acquisition subsystem and image enhancement using Histogram Equalization. Then we search and detect the faces in each individual frame. If no face is detected then another frame is acquired. If a face is detected, then a region of interest in marked within the face. This region of interest contains the eyes. Defining a region of interest significantly reduces the computational requirements of the system. After that the eyes are detected from the region of interest. If an eye is detected then there is no blink and the blink counter is set to „20‟. If the eyes are closed in a particular frame, then the blink counter is decremented and a blink is detected. When the eyes are closed for more than 4 frames then it is deducible that the driver is feeling drowsy. Hence drowsiness is detected and an alarm sounded. After that the whole process is repeated as long as the driver is driving the car. The overall flowchart for drowsiness detection system is shown in Figure 2.1

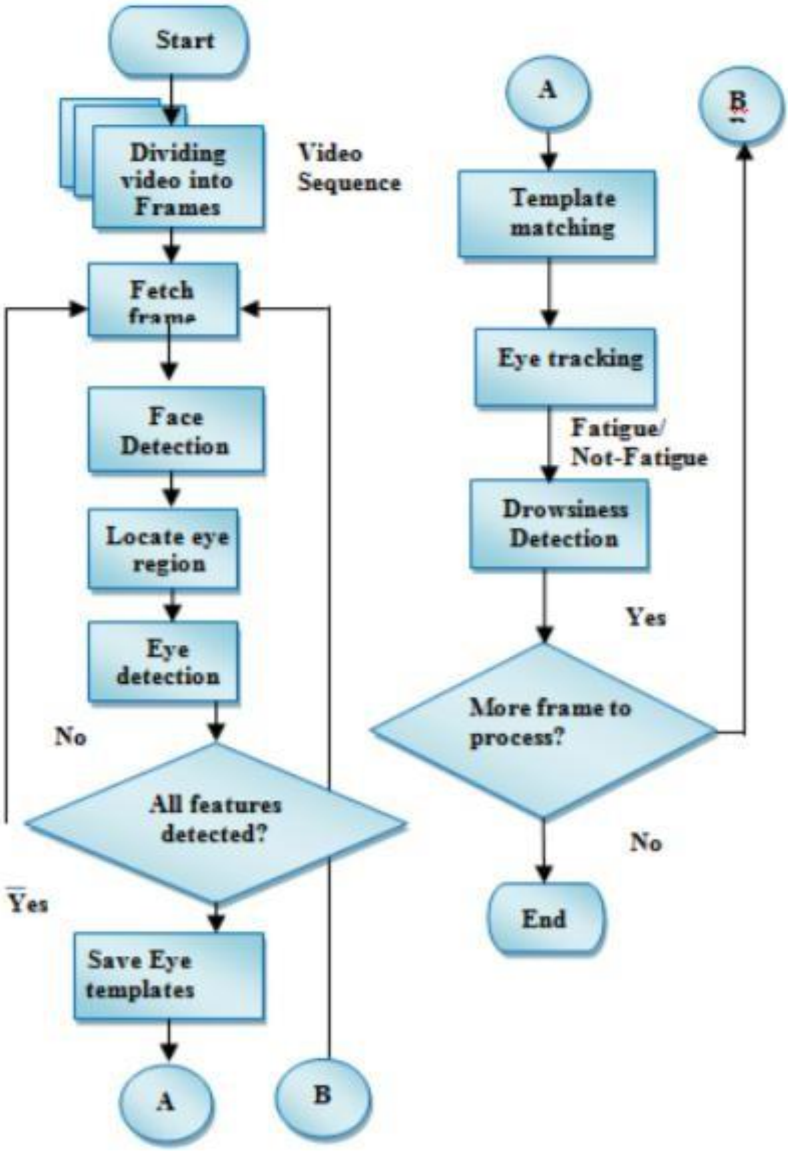


Figure 2.1. Flowchart of Driver Drowsiness Detection System

2.3.2 FACE DETECTION

Face detection is accomplished by the OpenCV algorithm proposed by Paul Viola and Michael Jones in 2001. Due to the complex background, it is not a good choice to locate or detect both the eyes in the original image, for this we will take much more time on searching the whole window with poor results. So firstly, we will locate the face, and reduce the range in which we will detect both the eyes. After doing this we can improve the tracking speed and correct rate, reduce the affect of the complex background. Besides, we propose a very simple but powerful method to reduce the computing complexity.

i) OpenCV-like features: The simple features used are suggestive approaches of OpenCV basis functions which have been used by Papageorgiouetal. A OpenCV-like feature considers affixed rectangular regions at a specific part in a detection window; each OpenCV like feature expressed by two or three jointed black and white rectangles shown in figure 2. The value of a OpenCV like feature is the difference between the sums of the pixel values within the black and white rectangular regions. These sums are used to find the difference between regions. Then the differences can be used to classify the sub region of an image. These differences are compared against learned threshold values to determine whether or not the object appears in the region.

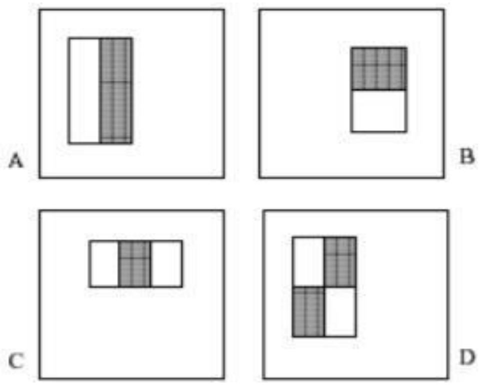


Figure 2.2: Rectangle features example: (A) and (B) shows two-rectangle features,
(C) Shows three-rectangle feature, and (D) shows four-rectangle feature

ii) Integral image The simple rectangular features of an image are calculated using an intermediate representation of an image, called the integral image as in (1) .The integral images are an array which consists of sums of the pixels‟ intensity values located directly to the left of a pixel and directly above the pixel at location (x,y) inclusive. Here, A[x,y] is the original image and Ai[x,y] is the integral image. Ai[x, y] = Σ A[x', y'] x'≤ x, y'≤ y (1) iii) AdaBoost Adaboost, nothing but "Adaptive Boosting ", It can be used with many other types of learning algorithms to improve their performance. Adaboost takes a number of positive and negative images features and training sets, The machine creates a set of weak classifiers of OpenCV-like features. It selects a set of weak classifiers to combine and that assigns lesser weights to good features whereas larger weights to poor features. This weighted combination gives strong classifier.

iii) Cascaded classifier

The cascade classifier consists of number of stages, where each stage is a collection of weak learners. The weak learners are simple classifiers known as decision stumps. Boosting is used to train the classifiers. It provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by the weak learners.

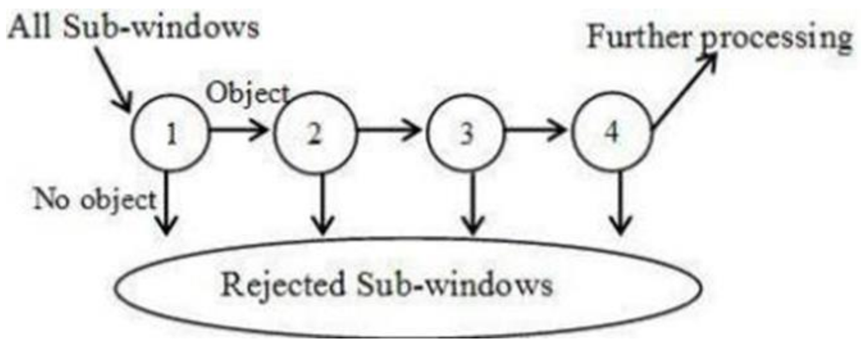


Figure 2.3: Cascade of Classifiers

Each stage of the classifier shows the region defined by the current location of the sliding window as either positive or negative. Positive indicates an object was found and negative indicates no object. If the label is negative, the classification of this region is complete, and the detector shifts the window to the next location. If the label is positive, the classifier passes the region to the next stage. The detector reports an object found at the current window location when the final stage classifies the region as positive. It is used to eliminate less likely regions quickly so that no more processing is needed. Hence, the speed of overall algorithm is increased.

2.4 SYSTEM DESCRIPTION

2.4.1 EYE DETECTION

Images or the real time video is captured from the camera installed in front of the driver's face. This video is converted into number of frames. OpenCV face OpenCV-classifier is loaded. Each frame is compared with the pre-defined features of the OpenCV-classifiers. When the features are matched the face is detected and a rectangle is drawn around the face. Using feature extraction. we estimate the position of the eyes. By comparing with the OpenCV eye-OpenCV classifier, the eyes are detected and rectangles are drawn around left and right. Eye. 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
- Nose7

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 Sullivn.
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:

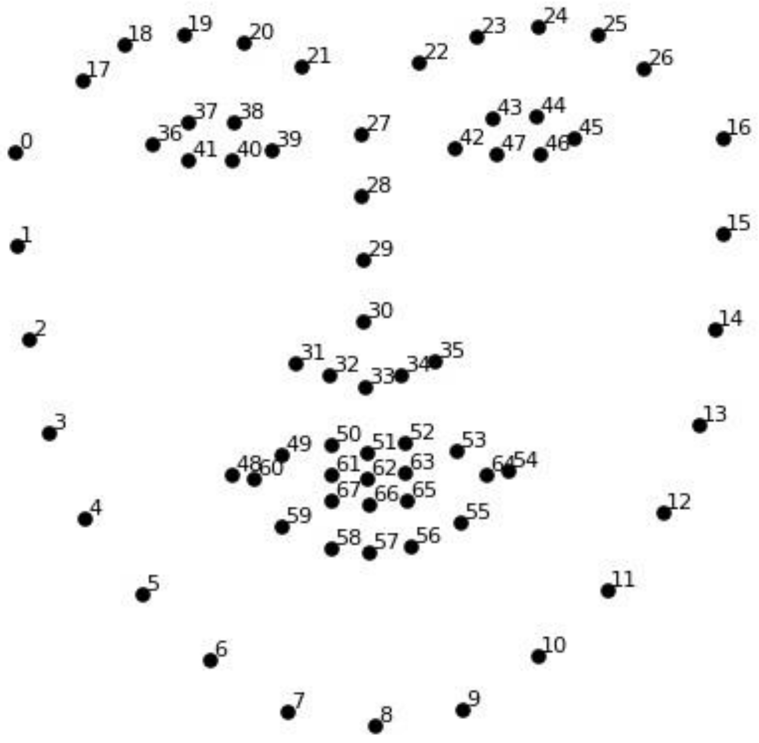


Fig.2.4: Visualizing the 68 facial landmark coordinates

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 official landmark detectors exist, including the 194-point model that can be trained on the HELEN dataset. Regardless of which in dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data.

2.4.2 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 nonblinking patterns.

2.4.3 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 = [p2 − p6] + [p3 − p5] / 2 * [p1 − p4]

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.

2.4.4 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”. 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.

2.4.5 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.

- **Template Matching**

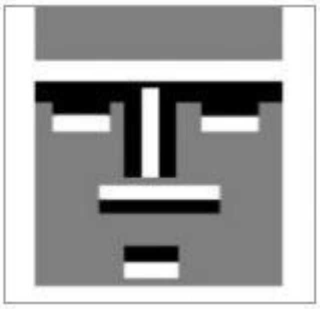
  Template matching is basically the two-dimensional cross correlation of a grayscale image with a grayscale template; hence it can be used to estimate the degree of similarity between the two images. Template matching is sensitive to variation of poses. Template matching is necessary for the desired accuracy in analyzing the user's blinking since it allows the user some freedom to move around slightly. Template matching is a technique in digital image processing for finding small parts of an image which match a template image. The normalized correlation coefficient is used to accomplish the tracking

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

- **Behavioral based approach:**
  In this technique eye blinking frequency, head pose, etc. of a person is monitored through a camera and the person is also any of these drowsiness symptoms are detected.

2.4.6 FACE DETECTION

For the face Detection it uses OpenCV 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, OpenCV 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. 3.2 represents five OpenCV like features shown in Fig 3.3.



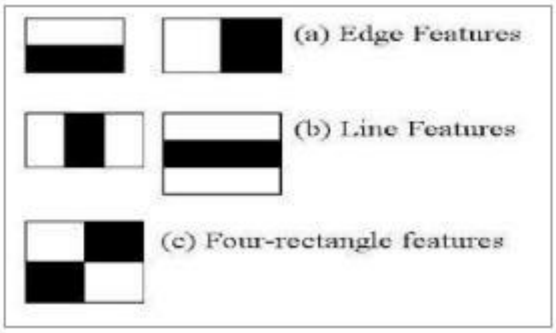


Fig 3.2 Fig 3.3

2.5 TECHNOLOGY USED

2.5.1 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 the machine learning applications such as neural networks. It is used for both research and production. TensorFlow computations are expressed as tensor.

2.5.2 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.

2.5.3 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, 4 image processing, image proc. 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 meths.

2.5.4 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 coordinates.

Python: Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping in the presf complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability. Python language is used by author due to his cross platform compatibility as main coding language for algorithm. Open CV and Dlib libraries are integrated in python interpreter for using readymade optimized functions.

2.6 SOFTWARE DESCRIPTION

2.6.1 PYTHON

Python is a significant level programming language used on a very basic level for generally helpful programming. It was made in the year 1991 by Guido van Rossum. It has a structure sanity that underlines generally on points of view, for instance, code intelligibility and strikingly uses significant whitespace. It is a viable programming language and is commonly used over the globe for both little and tremendous scope programming.

Python has motorized memory the board. It gives different programming perfect models some of which are object-arranged, fundamental, utilitarian and procedural. It in like manner has a broad and sweeping standard library. Favorable circumstances of Python which will help us in this task are:

- Python can run on any Operating System as it is stage free.
- Python grammar is straightforward and simple to recollect. Size and intricacy of the code is decreased in python when contrasted with other programming dialects.
- The prototyping process is accelerated because of the utilization of a compiler over a mediator.
- Python underpins every one of the 3-proceduralprogramming, object-situated programming just as useful programming and is in this manner considered extremely proficient.

Some of the striking utilizations of Python are:

- Helps in server-side programming in Web improvement
- Used tremendously in programming advancement where various complex issues should be settled
- Used in insights
- Used for System scripting
- Python bolsters database network and can likewise peruse from, write to and annex documents.
- It can be utilized in taking care of huge information and furthermore to determine complex scientific issues.
- It is additionally utilized for programming prototyping in the product advancement process

2.6.2 SPYDER/ANACONDA

Spyder is a Python Integrated Development Environment (IDE) for programming in the Python programming language. It is an open source, cross-stage IDE accessible on Anaconda. Spyder has a voluminous arrangement of inbuilt libraries that can be utilized. Not many of them are:

- NumPy
- SciPy
- Matplotlib
- Pandas
- IPython
- SymPy
- Cython

Spyder was made by Pierre Raybaut in 2009 and post 2012 Spyder has been kept up and unendingly improved by a gathering of intelligent Python engineers and a submitted network for the equal.

- PyFlakes
- Pylint
- Rope

For GUI, Spyder utilizes Qt. It is intended to utilize both of the PyQt or PySide Python ties. A slim reflection layer created by the Spyder venture and later

embraced by various different bundles and gives the opportunity to utilize either backend is called QtPy. It is a python system used to create basic frontend GUIs for python activities and projects created in Python.
As a cross-stage IDE it is accessible through:

- Anaconda
- Windows,
- macOS through MacPorts, and on significant Linux conveyances, for example,
- Debian
- Arch Linux,
- Gentoo Linux
- Fedora,
- openSUSE
- Ubuntu

Spyder can be extensible with first-gathering and outsider modules and has the game plan for instinctive gadgets for data audit and embeds Python-unequivocal code quality confirmation and thoughtfulness instruments.

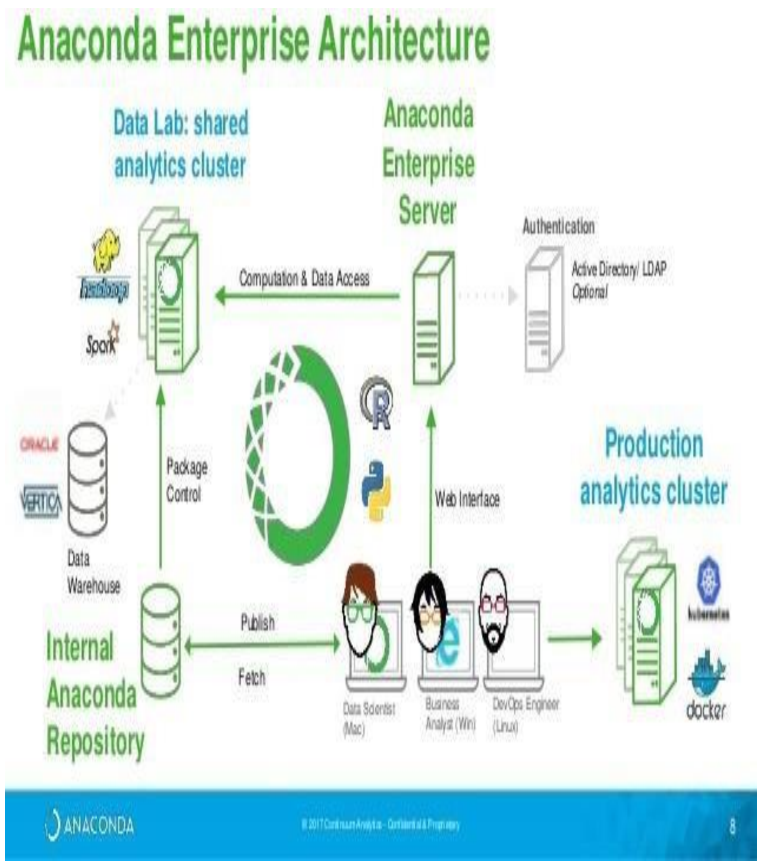

Anaconda Enterprise Architecture
Data Lab: shared analytics cluster
Anaconda Enterprise Server
Authentication
Active Directory/ LDAP Optional
Computation & Data Access
Spark
Package Control
Web Interface
Production analytics cluster
Data Warehouse
Internal Anaconda Repository
Publish
Fetch
Data Scientist (Mac)
Business Analyst (Win)
DevOps Engineer (Linux)
docker
ANACONDA

Fig 2.5 Anaconda Architecture

The salient features of Spyder IDE are:

- Comprises of an editor with efficient syntax highlighting, introspection and code completion on typing a part of it

- It facilitates use of parallel Python consoles on the IDE

- It has an illustrious Help Pane which facilitates in retrieving and rendering rich text documentation on functions, classes and methods either manually or automatically.

- It has the ability to explore and edit the variables from a GUI

- For the purpose of step-by-step execution, a debugger linked to IPdb is used.

- It enables static code analysis with the use of Pylint

- It has a run-time Profiler in order to mark the code

- It has an efficient project support that allows easy and parallel work on multiple development projects simultaneously
- It has an in-built file explorer which can be used for browsing and interacting with the filesystem and managing projects
- It has a search option which can be used to search for anything over a vast filesystem with a single regular expression
- It has a provision for online help that allows users to search and view Python and its various packages documentation inside the IDE
- It has an internal console which allows for introspection and control over it's own operation
- And last but not the least, it consists of a history log, which can come in handy when we have to record every user command which was entered in each console.

2.6.3 PANDAS

The dataset which is gotten will be in csv design. i.e. they will be comma-separated qualities. This dataset must be sustained into Pandas to perform prescient investigation. Pandas is utilized alongside python programming language and is accordingly an ideal fit for this task. It gives data structures and activities to recovering and controlling numerical tables and time arrangement. It is an open source programming, made allowed to use by people in general. The name Pandas is gotten from the econometrics term "panel data" which is utilized for data sets that incorporate readings over numerous timespans for a similar entity.

The prominent features of Pandas are:

- It provides support for Integrated indexing of Data Frame objects.
- It has numerous tools that facilitate in reading and writing of data between different file formats.
- Data alignment and missing data, if any is taken care of.
- Reduction in the size of datasets based on certain criteria is simplified. Indexingandsubsettingoflargedatasetstohaveareferenceismadeavailable. Insertion and deletion of columns is made easy.
- Has a “group by” engine which provides split-apply-combine operations on the data sets.
- Merging and joining of datasets is supported.
- It facilitates hierarchical axis indexing in order to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: It includes a myriad of functionalities such as frequency conversion, moving window statistics, moving window linear regressions, date shifting and lagging.
- It provides data filtration for the datasets.

2.6.4 SKLEARN

Scikit-learn is an open-source software machine learning library for the Python programming language. It encompasses numerous classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to collaborate with the Python numerical and scientific libraries NumPy and SciPy. The scikit-learn which was initially called as scikits. Learn was a “Google Summer of Code” project by David Cournapeau. The name originates from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 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.

3.2 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.

3.3 HARDWARE REQUIREMENTS

Processor	Intel or high level
RAM	2 GB
Space on disk	Min 2 GB
Device	Any device that has internet access
Execution space	Min 100mb

The system basically consists of webcam, microcontroller ARM7, buzzer, vibrator, LED‟s, relay. Microcontroller is used to perform various function and operations is ARM7 i.e. LPC2148. This paper presents a non-intrusive approach for drowsiness detection, based on computer vision. The web camera is installed in a car and it is able to work under real operation conditions.

3.4 SOFTWARE REQUIREMENTS

The implementation is done by using OpenCV which is an open-source image processing library for C#.NET. OpenCV is a cross platform .Net wrapper to the OpenCV image processing library. It permits OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, Iron, Python etc. To detect human facial feature, Intel developed an Open source library used for the implementation of computer vision related programs called OpenCV(Open source computer vision). . This creates a classifier given a set of positive and negative samples. OpenCV is an open source computer vision library. It is designed for computational efficiency and with a strong focus on real time applications. It helps to build vision applications quickly and easily. OpenCV satisfies the low processing power and high-speed requirements of our application.

CHAPTER 4

DESIGN

Drivers face is continuously monitored using a video camera. In order to detect the drowsiness the first step is to detect the face using the series of frame shots taken by the camera. Then the location of the eyes is detected and retina of the eye is continuously monitored. The captured image is sent to the Raspberry Pi board for image processing. The raspberry Pi converts the received image to digital signal using Open CV. The digital signal is transmitted from transmitter to the receiver. Both the transmitter and the receiver are paired up. The signal is then passed to the LPC2148, the microcontroller. If the signal crosses the threshold of two seconds, then the alarm beeps and the speed of the vehicle is automatically reduced. 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. The raspberry Pi converts the received image to digital signal using Open CV. The digital signal is transmitted from transmitter to the receiver. Both the transmitter and the receiver are paired up. The signal is then passed to the LPC2148, the microcontroller. If the signal crosses the threshold of two seconds, then the alarm beeps and the speed of the vehicle is automatically reduced.

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.

4.1 SYSTEM ARCHITECTURE

4.1.1 CLASS DIAGRAM

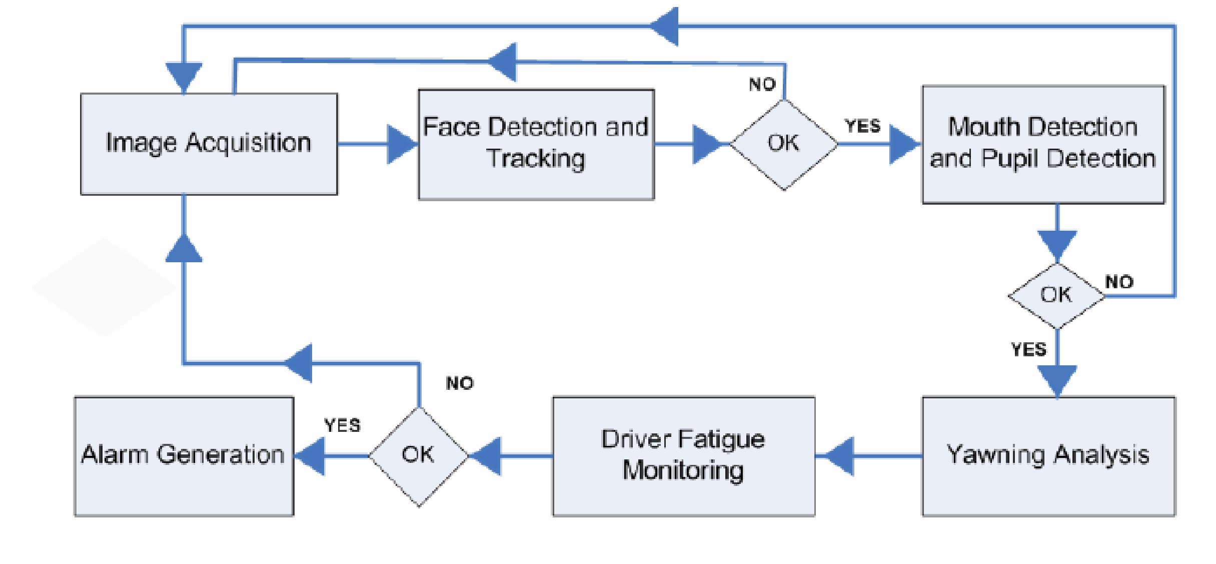


Image Acquisition
Face Detection and Tracking
NO
OK
YES
Mouth Detection and Pupil Detection
OK
NO
YES
Yawning Analysis
Driver Fatigue Monitoring
OK
NO
YES
Alarm Generation

Fig 4.1 Class Diagram

This viewpoint contains the static structure of system with entities, interaction among entities and software patterns that applied to the static structure of the system. Figure shows associated class diagram which contains information.

1. **Main Activity :**
   This class is the starting point of our project. It starts all the services. Methods getBatteryPercentage(): Gets the percentage of battery level. SpeakOut(): Triggers Text-To-Speech.

2. **STTService :**
   This class is responsible for Speech-To-Text. It is supposed to work at background all the time and receive voice commands. Attributes serviceState: Stores the current state of service`: WAITING,LISTENING, PROCESSING Methods getNumberWithName(String name): Gets the phone number of according to the name from contacts. calling(String phoneNumber):Calls the number.

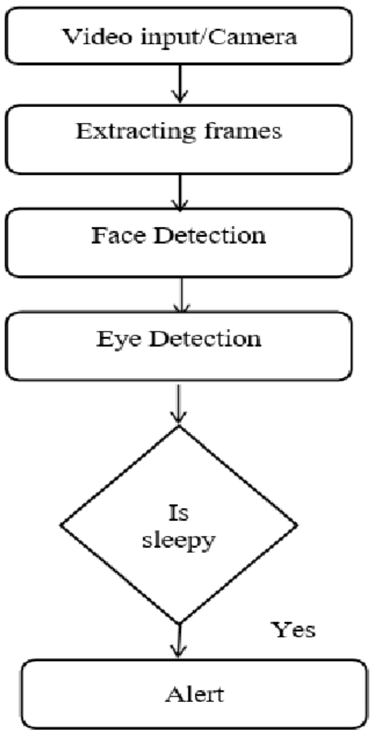
3. **ShakeEventListener :**
   This class implemented for detecting system crashes.

4. **RecognitionListener :**
   This class implemented for speech recognition as background.

5. **NLServic:**
   This class implemented for notification handling.

6. **DrowsinessDetection :**
   This class implements driver drowsiness detection.

4.1.2 STATE DIAGRAM



State Diagram

1. Idle

   When the application starts, it waits for starter command. After it receives the starter command, it goes the listening state.

2. Listening

In this state, the application waits for a valid command. If one of the supported command is received, it goes to one of the states to perform a task according to the received command.

3. Message

If the user says “message” in the listening state, the application enters the message state. When an existing contact name is provided, the application enters the feedback state. If the contact name is not found, the application goes to idle state.

4. Record

Video Whenever user gives the command “record” in the listening state, the application enters the Video Record state. After the recording starts, the application goes to idle state. If the user stops video recording, video is saved to the hard drive.

5. Eye Detection

When the users says “detect” in the listening state, the application enters the Eye Detection state and starts detection process. If the user closes eye detection, the application goes to the idle state.

4.1.3 SEQUENCE DIAGRAM

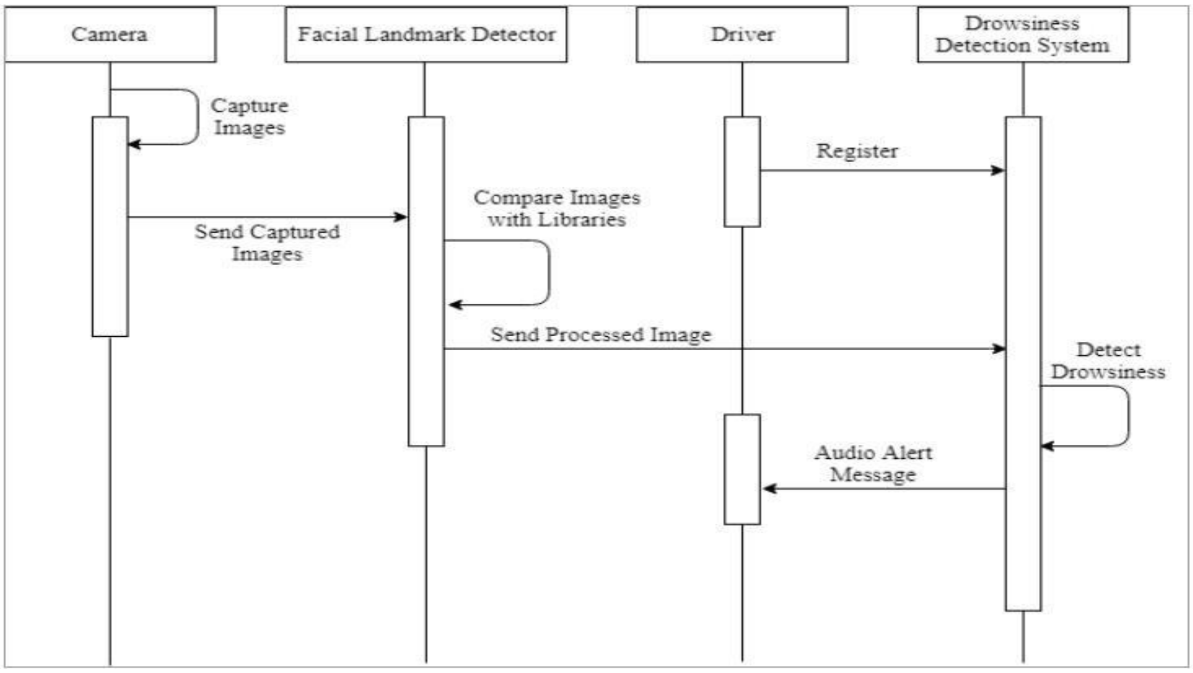


Fig 4.3 Sequence Diagram

1. **Camera :**
   Captures the video of the driver when the vehicle is in motion and sends the captured video to the storage unit.

2. **Facial Landmark Detector :**
   Detects the drivers facial changes and reports then.

3. **Driver :**
   The person who is driving the vehicle.

4. **Drowsiness Detection System:**
   The Image capturing mechanism where sequence of images is processed using various algorithms to detect whether the driver is fatigue or not.

4.1.3 DATA FLOW DIAGRAM

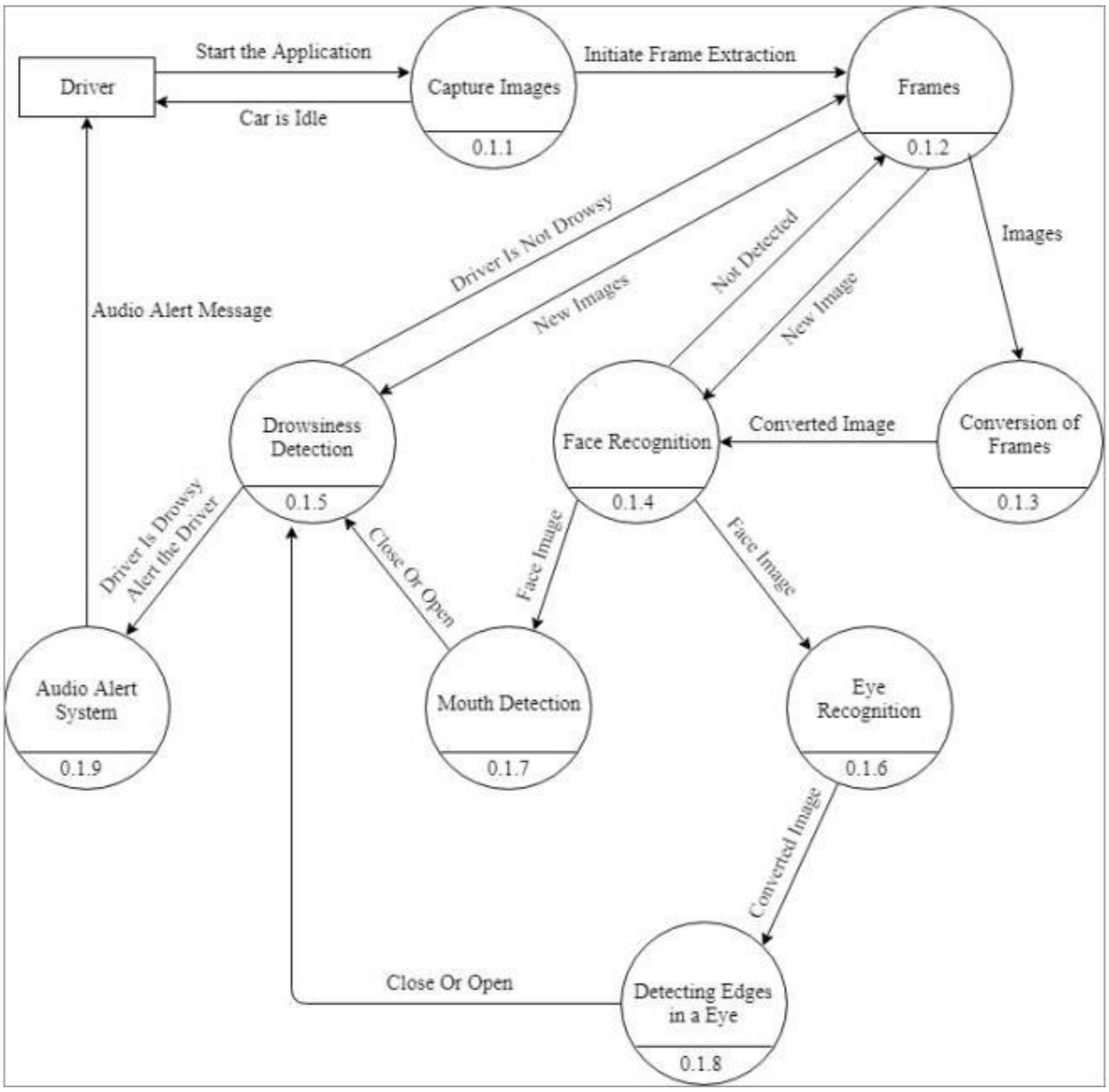


Fig 4.4 Data Flow Diagram

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.

4.2 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 smaller red 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.

CHAPTER 5

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.

5.1 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. 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.

5.2 TYPES OF TESTING

5.2.1. FUNCTIONAL TESTING:

Functional Testing is a type of software testing whereby the system is tested against the functional requirements/specifications. Functions (or features) are tested by feeding them input and examining the output. Functional testing ensures that the requirements are properly satisfied by the application. This type of testing is not concerned with how processing occurs, but rather, with the results of processing. It simulates actual system usage but does not make any system structure assumptions. During functional testing, Black Box Testing technique is used in which the internal logic of the system being tested is not known to the

tester. Functional testing is normally performed during the levels of System Testing and Acceptance Testing.
Typically, functional testing involves the following steps:

- Identify functions that the software is expected to perform.
- Create input data based on the function's specifications.
- Determine the output based on the function's specifications.
- Execute the test case.
- Compare the actual and expected outputs.

Functional testing is more effective when the test conditions are created directly from user/business requirements. When test conditions are created from the system documentation (system requirements/ design documents), the defects in that documentation will not be detected through testing and this may be the cause of end-users' wrath when they finally use the software.

5.2.2. NON-FUNCTIONAL TESTING

NON-FUNCTIONAL TESTING is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing. An excellent example of a non-functional test would be to check how many people can simultaneously login into a software. Non-functional testing is equally important as functional testing and affects client satisfaction. The Non-Functional requirements were also not given proper attention in the earlier test cycles. However, this has changed now. Non-functional tests are now most important as they consider all the application performance and security issues these days. This testing has a greater impact on applications when it comes to the performance of the application under high user traffic. This testing ensures that your application is stable and is able to handle load under extreme conditions.

5.2.3. REGRESSION TESTING

REGRESSION TESTING is a type of software testing that intends to ensure that changes (enhancements or defect fixes) to the software have not adversely affected it. The likelihood of any code change impacting functionalities that are not directly associated with the code is always there and it is essential that regression testing is conducted to make sure that fixing one thing has not broken another thing.

During regression testing, new test cases are not created but previously created test cases are re-executed. Regression [noun] literally means the act of going back to a previous place or state; return or reversion. In an ideal case, a full regression test is desirable but oftentimes there are time/resource constraints. In such cases, it is essential to do an impact analysis of the changes to identify areas of the software that have the highest probability of being affected by the change and that have the highest impact to users in case of malfunction and focus testing around those areas. Due to the scale and importance of regression testing, more and more companies and projects are adopting regression test automation tools.

5.2.4. PERFORMANCE TESTING

Performance testing is the process of determining the speed, responsiveness and stability of a computer, network, software program or device under a workload.

Performance testing can involve quantitative tests done in a lab, or occur in the production environment in limited scenarios. Typical parameters include processing speed, data transfer rate, network bandwidth and throughput, workload efficiency and reliability. For example, an organization can measure the response time of a program when a user requests an action or the number of millions of instructions per second (MIPS) at which a mainframe function.

CHAPTER 6

SOURCE CODE

1. Approach **A**

Using 68 Frontal Face Landmarks

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
#Pygame for playing the Alarming Sound
from pygame import mixer
#os for dealing with the Directory
import os

#Initilizing the Sound
mixer.init()
sound = mixer.Sound('alarm.wav')
path = os.getcwd()
```

```
#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)
ratio = 0
score = 0
thicc = 0

#Function for Computing Distance between two Points
def compute(ptA,ptB):
    dist = np.linalg.norm(ptA - ptB)
    return dist

#Blinked Function to calculate the Ratio ofthe Individual Eyes
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>0.18 and ratio<=0.25):
        return 1
    else:
        return 0



#While Loop Running for each Frame
while True:
    face_frame, frame = cap.read()
    height,width = frame.shape[:2]

    #Converting the Image to Grayscale
    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])

        #Now judge what to do for the eye blinks
        if(left_blink==0 or right_blink==0):
            sleep+=1
            drowsy=0
            active=0
            score += 1
            if(sleep>6):
                status="SLEEPING !!!"
                color = (0,0,255)
                cv2.putText(frame, "Take a Break...", (10,height-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,255,255),2)

        elif(left_blink==1 or right_blink==1):
            sleep=0
            active=0
            drowsy+=1
            score += 1
            if(drowsy>4):
                status="DROWSY !"
                color = (255,0,0)
        else:
            drowsy=0
            sleep=0
            active+=1
            score -= 1
            if(active>9):
                status="ACTIVE"
                color = (0,255,0)

        #To put the Text on the Top Right of the Frame
        cv2.putText(frame,                         status,                         (width-200,50),
    cv2.FONT_HERSHEY_SIMPLEX, 1.2, color,3)



        #For creating circles on the Face Landmarks
        for n in range(0, 68):
            (x,y) = landmarks[n]
            cv2.circle(face_frame, (x, y), 1, (255, 255, 255), -1)



    #For Dealing with the Scores
    if(score<0):
        score = 0
    cv2.putText(frame,'Score:'+str(score),(10,40),
cv2.FONT_HERSHEY_SIMPLEX, 1,(255,255,255),1,cv2.LINE_AA)
    if(score>12):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass

        #For Creating a Red Border during the detection of Sleepy State
        if(thicc<16):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)



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

2. Approach **B**

Using the concept of Model Training and CNN

❖ Libraries Used –

- import numpy as np # linear algebra
- import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
- import os
- import cv2
- from sklearn.preprocessing import LabelBinarizer
- from sklearn.model_selection import train_test_split
- from sklearn.metrics import classification_report
- from tensorflow.keras.layers import Input, Lambda, Dense, Flatten, Conv2D, MaxPooling2D, Dropout
- from tensorflow.keras.models import Model
- from tensorflow.keras.models import Sequential
- from keras.preprocessing.image import ImageDataGenerator
- import tensorflow as tf
- from keras.models import load_model
- import kerasfrom pygame import mixer
- import time

❖ Functions Used –

- os.listdir("C:\\Users\\hp\\Desktop\\DriverDrowsinessDetectionSystem\\Drowsiness Detection aa\\Dataset\\train")
- plt.imread("C:\\Users\\hp\\Desktop\\DriverDrowsinessDetectionSystem\\Drowsiness Detection aa\\Dataset\\train\\Open\\_57.jpg")
- os.path.join(direc, category)
- categories.index(category)
- os.listdir(path_link)
- cv2.imread(os.path.join(path_link,image), cv2.IMREAD_COLOR)
- cv2.CascadeClassifier(face_cas_path)
- face_cascade.detectMultiScale(image_array, 1.3, 5)
- cv2.r/ectcv2.resize(roi_color, (IMG_SIZE, IMG_SIZE))angle(image_array, (x, y), (x+w, y+h), (0, 255, 0), 2)
- yaw_no.append([resized_array, class_num1])
- yaw_no.extend(data)
- X = np.array(X)
- X = X.reshape(-1, 145, 145, 3)

- label_bin = LabelBinarizer()
- y = label_bin.fit_transform(y)
- X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=seed, test_size=test_size)
- train_generator = ImageDataGenerator(rescale=1/255, zoom_range=0.2, horizontal_flip=True, rotation_range=30)
- test_generator = ImageDataGenerator(rescale=1/255)
- train_generator = train_generator.flow(np.array(X_train), y_train, shuffle=False)
- test_generator = test_generator.flow(np.array(X_test), y_test, shuffle=False)

Model Training and Validation

- model = Sequential()
- model.add(Conv2D(256, (3, 3), activation="relu", input_shape=X_train.shape[1:]))
- model.add(MaxPooling2D(2, 2))
- model.add(Conv2D(128, (3, 3), activation="relu"))
- model.add(MaxPooling2D(2, 2))
- model.add(Conv2D(64, (3, 3), activation="relu"))
- model.add(MaxPooling2D(2, 2))
- model.add(Conv2D(32, (3, 3), activation="relu"))
- model.add(MaxPooling2D(2, 2))
- model.add(Flatten())
- model.add(Dropout(0.5))
- model.add(Dense(64, activation="relu"))
- model.add(Dense(4, activation="softmax"))
- model.compile(loss="categorical_crossentropy",metrics=["accuracy"],optimizer="adam")
- model.summary()
- history = model.fit(train_generator, epochs=10, validation_data=test_generator, shuffle=True,validation_steps=len(test_generator))
- model.save('C:\\Users\\hp\\Desktop\\DriverDrowsinessDetectionSystem \\Drowsiness Detection aa\\Models\\cnnCat2_new_1.h5', overwrite=False)
- epochs = range(len(accuracy))
- plt.plot(epochs, accuracy, "b", label="trainning accuracy")
- plt.plot(epochs, val_accuracy, "r", label="validation accuracy")

- plt.legend()
- plt.show()
- model = load_model('Models/cnnCat2_new.h5')
- model.predict(X_test)
- Prediction = np.argmax(model.predict(X_test), axis=-1)
- classification_report(np.argmax(y_test,axis=1),prediction,target_names= labels_new)

Prediction on the basis of the Model

- mixer.init()
- sound = mixer.Sound('alarm.wav')
- face = cv2.CascadeClassifier('haar cascade files\haarcascade_frontalface_alt.xml')
- leye = cv2.CascadeClassifier('haar cascade files\haarcascade.xml')
- reye = cv2.CascadeClassifier('haar cascade files\haarcascade - Copy.xml')
- model = load_model('Models/cnnCat2_new.h5')
- path = os.getcwd()
- cap = cv2.VideoCapture(0)
- ret, frame = cap.read()
- height,width = frame.shape[:2]
- faces=face.detectMultiScale(frame,minNeighbors=5,scaleFactor=1.1,minSize=(25,25))
- left_eye = leye.detectMultiScale(frame)
- right_eye = reye.detectMultiScale(frame)
- cv2.rectangle(frame, (0,height-100) ,(155,height) , (0,0,0) , thickness=cv2.FILLED)
- face_new = frame[y:y+h,x:x+w]
- face_new = cv2.resize(face_new,(145,145))
- face_new = face_new/255
- face_new = face_new.reshape(145,145,-1)
- face_new = np.expand_dims(face_new,axis=0)
- fpred = (model.predict(face_new) > 0.5).astype("int32")
- if((fpred[0][0]==1).any()):
  ```
       lbl='Yawn'
     if((fpred[0][1]==1).any()):
       lbl='No Yawn'
     break
  ```
- if(((rpred[0][2]==1).any()) and ((lpred[0][2]==1).any())):
  ```
     score = score + 1
  ```

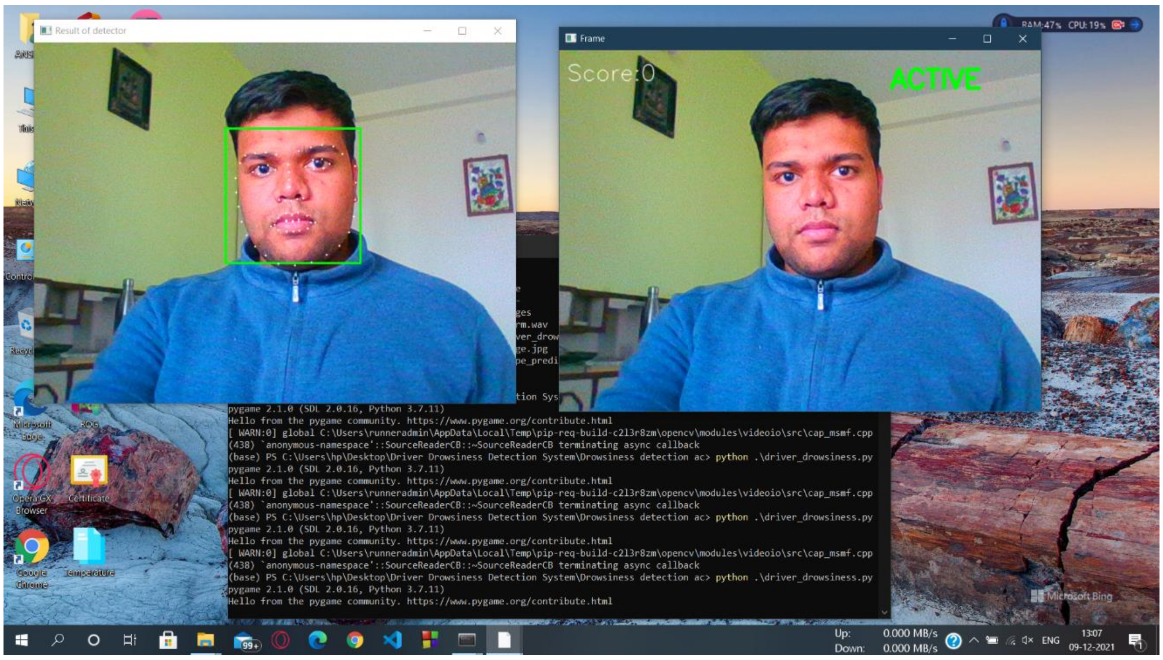
```
  cv2.putText(frame,"Closed",(10,height-
60),font,1,(255,255,255),1,cv2.LINE_AA)
➢ if(score>2):
        #person is feeling sleepy so we beep the alarm
        cv2.imwrite(os.path.join(path,'image.jpg'),frame)
        try:
            sound.play()

        except:  # isplaying = False
            pass
        if(thicc<8):
            thicc= thicc+2
        else:
            thicc=thicc-2
            if(thicc<2):
                thicc=2
        cv2.rectangle(frame,(0,0),(width,height),(0,0,255),thicc)
    cv2.imshow('frame',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
➢ cap.release()
➢ cv2.destroyAllWindows()
```

CHAPTER 7

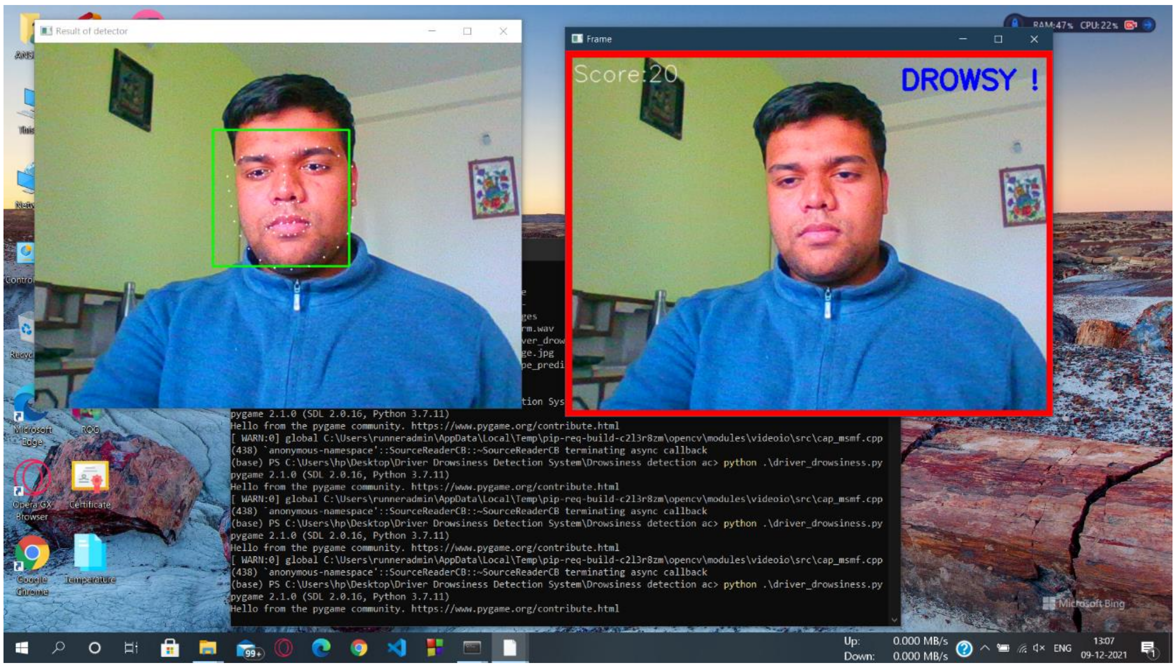
OUTPUT SCREENSHOTS

1. Active State



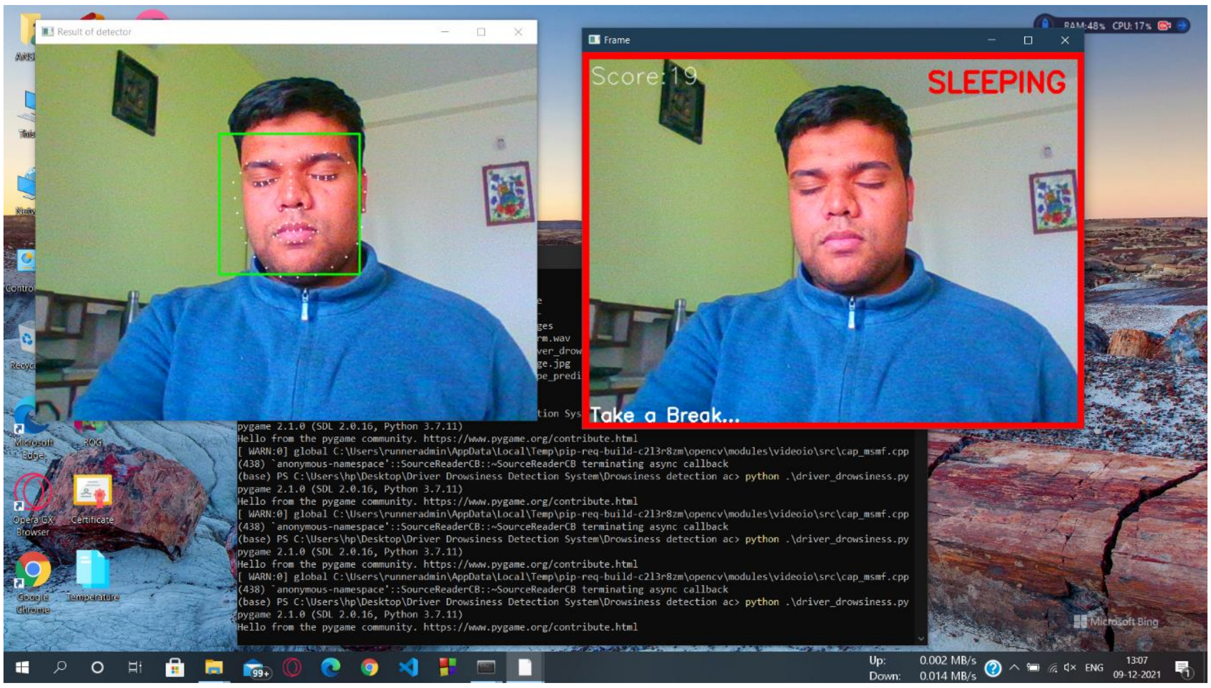
```
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

2. Drowsy State



```
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

3. Sleeping State



```
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
[ WARN:0] global C:\Users\runneradmin\AppData\Local\Temp\pip-req-build-c2l3r8zm\opencv\modules\videoio\src\cap_msmf.cpp
(438) `anonymous-namespace'::SourceReaderCB::~SourceReaderCB terminating async callback
(base) PS C:\Users\hp\Desktop\Driver Drowsiness Detection System\Drowsiness detection ac> python .\driver_drowsiness.py
pygame 2.1.0 (SDL 2.0.16, Python 3.7.11)
Hello from the pygame community. https://www.pygame.org/contribute.html
```

CHAPTER 8

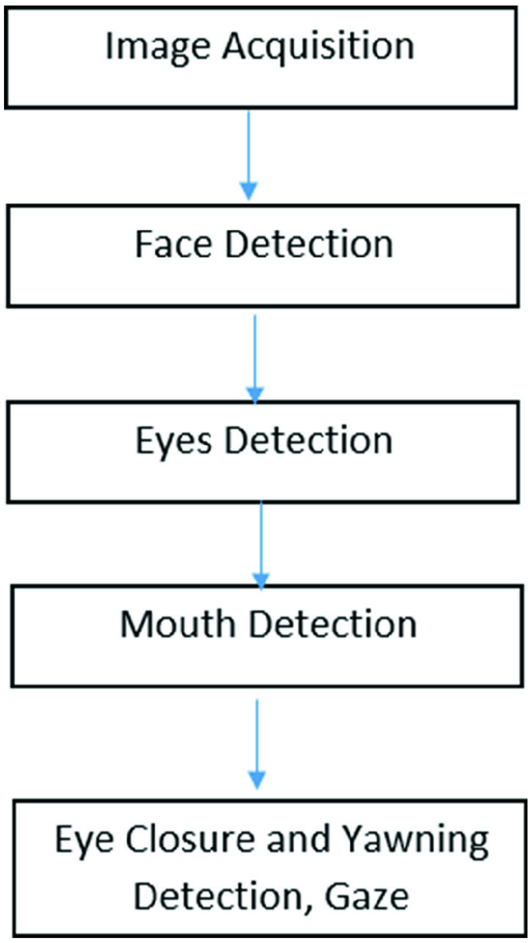
CONCLUSION

This project tells how to build a drowsiness detector using OpenCV, Python, and Dlib opensource Libraries. 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 N^{th} 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.

FUTURE REFERENCES

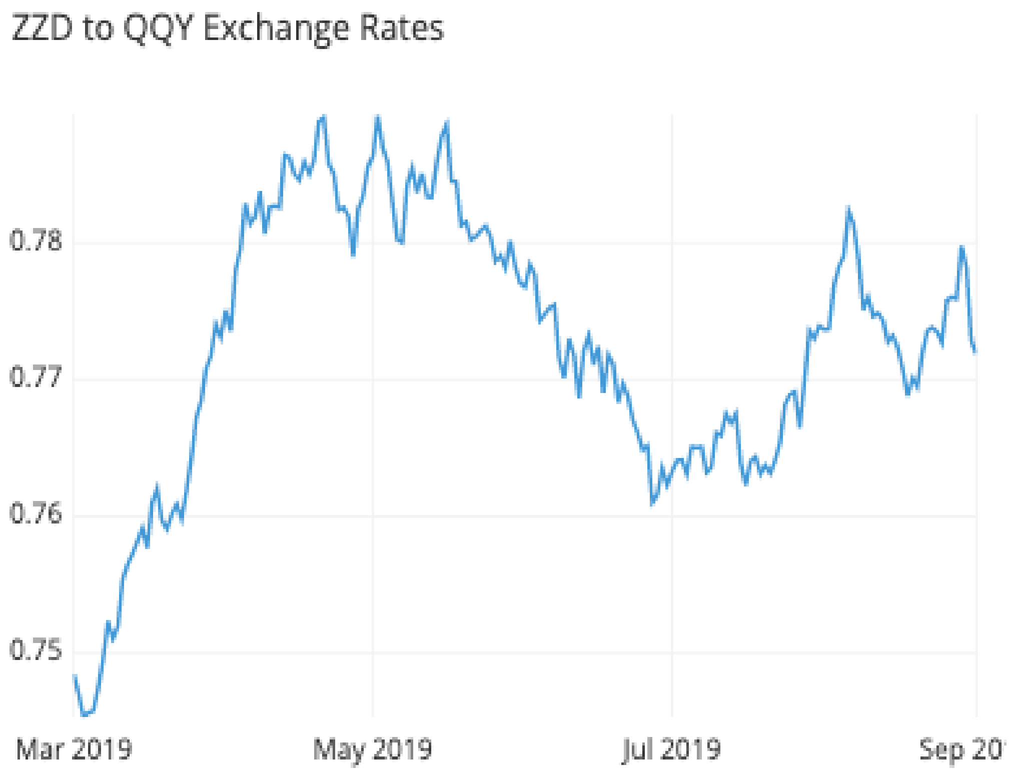
1) Yawning Detection :-

Yawn Detection is all about detecting yawn (open one’s mouth wide and inhale deeply due to tiredness or boredom) using OpenCV and Dlib. It can be used in various major applications like Self Driving Cars, Driver’s Fatigue detection, Driver’s Drowsiness detection, Driver’s consciousness detection etc.



2) Calculating score of Driver :-

On the basis of drowsiness state and sleepy state, we calculate score of driver which indicates how efficient is the driver and on the basis of this score we design a graph of his performance at each hour.



BIBLIOGRAPHY

[1] Association for Safe InternationalRoad Travel (ASIRT), Road Crash Statistics.http://asirt.org/initiatives/informingroadusers/road-safety-facts/road-cras h-statistics, 2016

[2] https://en.wikipedia.org/wiki/Microslee

[3] Journal of VLSI Signal Processing 23, 497–511 (1999) c °1999 Kluwer Academic Publishers. Manufactured in The Netherlands.

[4] https://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_de tection.html

[5] Eye Detection Using Morphological and Color Image Processing Tanmay Rajpath Aka, Rajnesh Kumar and Eric Schwartz

[6] A Robust Algorithm for Eye Detection on Grey Intensity Face without Spectacles- JCS&T Vol. 5 No. 3

[7] Frobel Kebbuck: Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001, Halmstad, Sweden, June 2001. Proceedings, Springer. ISBN 3-540-42216-1.

Thank You