**An Industry Oriented Mini Project Report**

**on**

**DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING USING PYTHON**

Submitted to the

**JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD**

In partial fulfilment of the requirement for the award of the degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

BY

**KAMLESH GADERIYA (17WJ1A05D5)**

**SANJAY KUMAR MEHTA (17WJ1A05G1)**

**Under the Esteemed Guidance Of**

**Dr. Rishi Sayal**

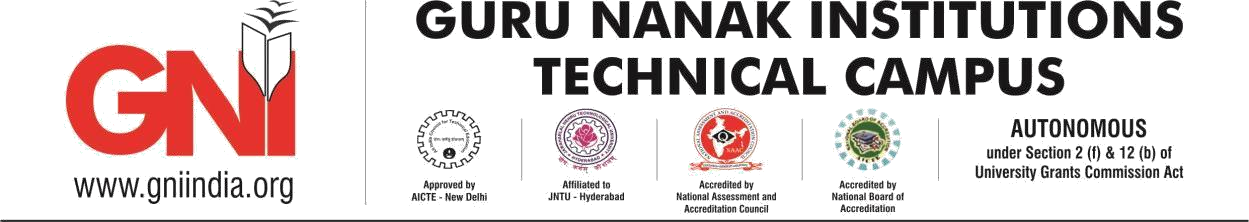
**Associate Director, CSE Dept.**



**GURU NANAK INSTITUTIONS TECHNICAL CAMPUS (AUTONOMOUS)**

**School of Engineering and Technology**

# Ibrahimpatnam R.R District 501506 2021



# Department of Computer Science & Engineering

**CERTIFICATE**

This is to certify that An Industry Oriented Mini Project Report entitled **“DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING PYTHON**” being submitted by **KAMLESH GADERIYA (17WJ1A05D5)** **SANJAY KUMAR MEHTA (17WJ1A05G1)** in partial fulfillment for the award of the Degree of **Bachelor of Technology** in **Computer Science & Engineering** during the academic year 2020-2021, is a record of bonafied record work carried out under our guidance and supervision at **Guru Nanak Institutions Technical Campus(Autonomous), JNTUH, Hyderabad**.

**INTERNAL GUIDE PROJECT CO-ORDINATOR HOD CSE**

## Dr. Rishi Sayal Mr. A.Ravi Dr. J.Rajeshwar

**EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

I wish to express our sincere thanks to **Dr. Rishi Sayal** Associate Director, GNITC for providing us the conductive environment for carrying through our academic schedules and Project with ease.

I have been truly blessed to have a wonderful adviser **Dr. J. Rajeshwar,** Professor & HODof CSE, GNITC for guiding us to explore the ramification of our work and we express our sincere gratitude towards him for leading me thought the completion of Project.

I would like to say our sincere thanks to **Mr. A. Ravi**, Associate Professor, Department of CSE, Project Coordinator, for providing seamless support and right suggestions are given in the development of the project.

I specially thank our internal guide **Dr. Rishi Sayal,** Associate Director, Department of CSE, for his suggestions and constant guidance in every stage of the project. Finally, we would like to thank our family members for their moral support and encouragement to achieve goals.

KAMLESH GADERIYA (17WJ1A05D5)

**TABLE OF CONTENTS**

**CONTENTS………………………………………………………….. PAGE NO**

**ABSTRACT……………………………………………………………………………….i**

[**LIST OF FIGURES**](#_Toc8249)**………………………………………………………………………ii**

**LIST OF TABLES………………………………………………………………………..iii**

**CHAPTER 1. INTRODUCTION…………………………………………………...…...1**

1.1 OVERVIEW………………………………………………………...........1

1.2 OBJECTIVE…………………………………….……………………......1

1.3 EXISTING SYSTEM……………………………….……………….…...1

1.3.1 EXISTING SYSTEM DISADVANTAGES…....……………….2

1.3.2 LITERATURE SURVEY…………………………………..………......2

1.4 PROPOSED SYSTEM……………………………………………….….5

1.4.1 PROPOSED SYSTEM ADVANTAGES…………………….….5

**CHAPTER 2. PROJECT DESCRIPTION…………………………………………......6**

2.1 GENERAL…………………………………………………………….….6

2.2 PROBLEM DEFINATION………………………………….…………....6

2.3 METHODOLOGIES……………………………………….…………......6

2.4 TECHNIQUE OR ALGORITHM…………………………….…………..8

**CHAPTER 3. REQUIREMENTS………………………………………………….…....10**

3.1 GENERAL………………………………………………………….……...10

3.2 HARDWARE REQUIREMENTS…………………….….………..……..10

3.3 SOFTWARE REQUIREMENTS…………………………….……….…..10

3.4 FUNCTIONAL SPECIFICATION……………………….……………....11

3.5 NON-FUNCTIONAL SPECIFICATION……………….………………..11

**CHAPTER 4. SYSTEM DESIGN…………………………………………………......12**

4.1 GENERAL………………………………………………………………..12

4.1.1 USE CASE DIAGRAM…………………………………………...12

4.1.2 CLASS DIAGRAM……………………………………………….13

4.1.3 OBJECT DIAGRAM……………………………………………...14

4.1.4 STATE DIAGRAM………………………………………………..15

4.1.5 SEQUENCED DIAGRAM ……………………………………….16

4.1.6 COLLABARATION DIAGRAM……………………………….…17

4.1.7 ACTIVITY DIAGRAM…………………………………………...18

4.1.8 COMPONENT DIAGRAM……………………………………….19

4.1.9 DEPLOYMENT DIAGRAM………………………………………20

4.1.10 E-R DIAGRAM…………………………………………………...21

4.2 DATA FLOW DIAGRAM……………………………………………….22

4.2.1 SYSTEM ARCHETECTURE……………………………………...24

**CHAPTER 5. SOFTWARE SPECIFICATION…………………………………25**

5.1 PYTHON……………………………………………………………25

5.2 HISTORY OF PYTHON…………………………………………....25

5.2.1 IMPORTANCE OF PYTHON………………………………..25

5.2.2 FEATURES OF PYTHON……………………………………..26

5.2.3 LIBRARIES USED IN PYTHON……………………………...27

**CHAPTER 6. IMPLEMENTATION……………………………………………….28**

6.1 GENERAL………………………………………………………….....28

6.2 IMPLEMENTATION……………………………………………...….38

**CHAPTER 7. SNAPSHOTS………………………………………………………...40**

7.1 GENERAL…………………………………………………………….40

7.2 VARIOUS SNAPSHOTS……………………………………………..40

**CHAPTER 8. SOFTWARE TESTING……………………………………………....42**

8.1 GENERAL…………………………………………………………...….42

8.2 DEVELOPING METHODOLOGIES………………………………......42

8.3 TYPES OF TESTING……………………………………….…………..42

8.4 TEST CASES AND TEST RESULT……………………………………44

8.5 BUILD THE TEST PATH………………………………………………45

**CHAPTER 9. APPLICATIONS AND FUTURE ENHANCEMENT….……….…....46**

9.1 GENERAL………………………………………………………….…....46

9.2 FUTURE ENHANCEMENTS…………………………………….….…46

**CHAPTER 10. CONCLUSION & REFERENCE…………………………………….47**

10.1 CONCLUSION………………………………………………………...47

10.2 REFERENCES………………………………………………………...48

**ABSTRACT**

Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Hence, detection of driver’s fatigue and its indication is an active area of research due to its immense practical applicability. The basic drowsiness detection system has three blocks/modules; acquisition system, processing system and warning system. Here, the video of the driver’s frontal face is captured in acquisition system and transferred to the processing block where it is processed online to detect drowsiness. If drowsiness is detected, a warning or alarm is sent to the driver from the warning system. The CV system is based on a bright-pupil technique to facilitate eye detection and tracking. In this technique a near-infrared (NIR) light source is placed coaxially with the optical axis of the camera. The reflected light from the eye retina of the human subject is caught by a camera as a result of this alignment. In order to obtain this bright-pupil effect, a monocular complementary metal–oxide–semiconductor (CMOS) camera is used, with illumination from a ring of light-emitting diodes aligned coaxially with the camera lens and emitting NIR light at a wavelength of 880 nm. A video stream of frontal face images including eye regions with the bright pupil effect is continuously obtained and processed frame by frame.

# LIST OF FIGURES

# 

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 4.1.1 | Use case Diagram | 12 |
| 4.1.2 | Class Diagram | 13 |
| 4.1.3 | Object Diagram | 14 |
| 4.1.4 | State Diagram | 15 |
| 4.1.5 | Sequence Diagram | 16 |
| 4.1.6 | Collaboration Diagram | 17 |
| 4.1.7 | Activity Diagram | 18 |
| 4.1.8 | Component Diagram | 19 |
| 4.1.9 | Deployment Diagram | 20 |
| 4.1.10 | E-R Diagram | 21 |
| 4.2 | Data flow Diagram | 22 |
| 4.2.1 | System Architecture | 24 |

**SYMBOL TABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | NAME  Class A  Class B  Class B  Class A | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class B    Class B  Class A | It aggregates several classes into a single classes. |
| 5. | Aggregation | Interaction between the system and external environment |
| 5. | *Relation*  (uses) | *Uses* | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the process. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interaction between the system and external environment. |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15.  16. | Node  Data Process/State |  | Represents physical modules.  Represents a state or process. |
| 17. | External entity |  | Represent external entities ex-keyboard , sensors etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER 1**

**INTRODUCTION**

**1.1 GENERAL:**

With the development of new computing paradigm, cloud computing becomes the most notable one, which provides convenient, on-demand services from a shared pool of configurable computing resources. Therefore, an increasing number of companies and individuals prefer to outsource their data storage to cloud server. Despite the tremendous economic and technical advantages, unpredictable security and privacy concerns become the most prominent problem that hinders the widespread adoption of data storage in public cloud infrastructure. Encryption is a fundamental method to protect data privacy in remote storage. However, how to effectively execute keyword search for plaintext becomes difficult for encrypted data due to the unread ability of ciphertext. Searchable encryption provides mechanism to enable keyword search over encrypted data. For the file sharing system, such as multi-owner multiuser scenario, fine-grained search authorization is a desirable function for the data owners to share their private data with other authorized user. However, most of the available systems require the user to perform a large amount of complex bilinear pairing operations. These overwhelmed computations become a heavy burden for user’s terminal, which is especially serious for energy constrained devices. The outsourced decryption method allows user to recover the message with ultra-lightweight decryption. However, the cloud server might return wrong half-decrypted information as a result of malicious attack or system malfunction. Thus, it is an important issue to guarantee the correctness of outsourced decryption in public key encryption with keyword search (PEKS) system.

**1.2. OBJECTIVE**

In this system we are going to control the car as well as give sound alert to the driver using buzzer

So as to acquire this brilliant student impact, a web camera and Arduino uno is utilized, which connects to car controller to control the driving system automatically.

**1.3 EXISTING SYSTEM:**

The proposed driver drowsiness monitoring system has been depicted in Fig 1. At first, the video is recorded using a webcam. The camera will be positioned in front of the driver to capture the front face image. From the video, the frames are extracted to obtain 2-D images. Face is detected in the frames using histogram of oriented gradients (HOG) and linear support vector machine (SVM) for object detection. After detecting the face, facial landmarks like positions of eye, nose, and mouth are marked on the images. From the facial landmarks, eye aspect ratio, mouth opening ratio and position of the head are quantified and using these features and machine learning approach, a decision is obtained about the drowsiness of the driver. If drowsiness is detected, an alarm will be sent to the driver to alert him/her.

**1.3.1 EXISTING SYSTEM DISADVANTAGES:**

* No Security for the files.
* Wastage of Time.

**1.3.2 LITERATURE SURVEY**

**Title:** Intelligent Video-Based Drowsy Driver Detection System under Various Illuminations and Embedded Software Implementation

**Author:**Wei-Liang Ou, Ming-Ho Shih, Chien-Wei Chang, Xue-Han Yu, and Chih-Peng Fan

**Year:** 2015

**Description:**- An intelligent video-based drowsy driver detection system, which is unaffected by various illuminations, is developed in this study. Even if a driver wears glasses, the proposed system detects the drowsy conditions effectively. By a near-infrared-ray (NIR) camera, the proposed system is divided into two cascaded computational procedures: the driver eyes detection and the drowsy driver detection. The average open/closed eyes detection rates without/with glasses are 94% and 78%, respectively, and the accuracy of the drowsy status detection is up to 91%. By implementing on the FPGA-based embedded platform, the processing speed with the 640x480 format video is up to 16 frames per second (fps) after software optimizations.

# Title: Driver fatigue detection based on eye tracking and dynamic template matching

**Author:** W. B. Horng, C. Y. Chen, Y. Chang, C. H. Fan,

**Year:** 2004

**Description:** A vision-based real-time driver fatigue detection system is proposed for driving safely. The driver's face is located, from colour images captured in a car, by using the characteristic of skin colours. Then, edge detection is used to locate the regions of eyes. In addition to being used as the dynamic templates for eye tracking in the next frame, the obtained eyes' images are also used for fatigue detection in order to generate some warning alarms for driving safety. The system is tested on a Pentium III 550 CPU with 128 MB RAM. The experiment results seem quite encouraging and promising. The system can reach 20 frames per second for eye tracking, and the average correct rate for eye location and tracking can achieve 99.1% on four test videos. The correct rate for fatigue detection is l00%, but the average precision rate is 88.9% on the test videos.

**Title: Monitoring driver fatigue using facial analysis techniques**

**Author:** S. Singh, N. P. papa Nikolopoulos,

**Year:** 2002

**Description:** In this paper, we describe a non-intrusive vision-based system for the detection of driver fatigue. The system uses a color video camera that points directly rewards the driver's face and monitors the driver's eyes in order to detect micro-sleeps (short periods of sleep). The system deals with skin-color information in order to search for the face in the input space. After segmenting the pixels with skin like color, we perform blob processing in order to determine the exact position of the face. We reduce the search space by analysing the horizontal gradient map of the face, taking into account the knowledge that eye regions in the face present a great change in the horizontal intensity gradient. In order to find and track the location of the pupil, we use Gray scale model matching. We also use the same pattern recognition technique to determine whether the eye is open or closed. If the eyes remain closed for an abnormal period of time (5-6 sec), the system draws the conclusion that the person is falling asleep and issues a warning signal.

**Title:** Using Image Processing in the Proposed Drowsiness Detection System Design

**Author:** Mohsen Poursadeghiyan, Adel Mazloumi, Gebraeil Nasl Saraji, mohammad mehdi Baneshi

**Year:** 2018

**Description:** Drowsiness is one of the underlying causes of driving accidents, which contribute, to many road fatalities annually. Although numerous methods have been developed to detect the level of drowsiness, techniques based on image processing are quicker and more accurate in comparison with the other methods. The aim of this study was to use image-processing techniques to detect the levels of drowsiness in a driving simulator. Methods: This study was conducted on five suburban drivers using a driving simulator based on virtual reality laboratory of Khaje-Nasir Toosi University of Technology in 2015 Tehran, Iran. The facial expressions, as well as location of the eyes, were detected by Violla-Jones algorithm. Criteria for detecting drivers' levels of drowsiness by eyes tracking included eye blink duration blink frequency and PERCLOS that was used to confirm the results. Results: Eye closure duration and blink frequency have a direct ratio of drivers' levels of drowsiness. The mean of squares of errors for data trained by the network and data into the network for testing, were 0.0623 and 0.0700, respectively. Meanwhile, the percentage of accuracy of detecting system was 93. Conclusion: The results showed several dynamic changes of the eyes during the periods of drowsiness. The present study proposes a fast and accurate method for detecting the levels of drivers' drowsiness by considering the dynamic changes of the eyes.

**Title:** Drowsy Driver Identification Using Eye Blink detection

**Author:** R. Ahmad, and J. N. Borole,

**Year:** 2015

**Description:** As field of signal processing is widening in various security and surveillance applications, motivated the interest for implementing better application with less complications. A non-intrusive machine vision based concepts is used to simulate Drowsiness Detection System. The system is consisting of web camera which placed in a way that it records driver’s head movements in order to detect drowsiness. As drowsiness is detected, a signal is issued to alert the driver. The system deals with detecting face, eyes and mouth within the specific segment of the image. All the possible actions have been considered and output is generated accordingly. Drowsiness is determined by observing the eye blinking action of the driver. Other than drowsiness, driver’s attention while driving is also considered. The proposed algorithm is developed to minimize the complexity level from existing system while efficiency has given prime importance which was a main objective of the paper. The system is implemented using cascade object identifier from vision toolbox of MATLAB, which detects face, eyes, nose and mouth from the image which is captured from web camera. For this system Region of Interest is location of eyes and mouth which are determined and indicated by rectangle. Logic has been used here to identify whether eyes are open or closed unlike general methods. From mouth portion yawning is determined and considered. Project is simulated for on line and off line video with all possible situations of a driver. Results are formulated under different categories like normal driver, driver with glass under different light intensities. It is concluded that proposed system can also be utilized for other application. Results obtained from the proposed system provide efficient system analysis and overall good efficiency with some precautions by using simple flow of programming.

**1.4 PROPOSED SYSTEM**

The CV system can output five different measurements: pupil area, and the vector components gaze x, gaze y, head x, and head y as defined below. The resultant segmented region of the driver’s eye, with the associated NIR corneal reflection or ‘glint’ can be seen in Fig. 2(a). The pupil area is measured to obtain eye closure and number of blinks as functions of time, which is a direct indicator of drowsiness. In addition to eye closure, the attention of the driver as a function of time can be obtained by measuring the eye movement based on a vector between the centres of glint and pupil in the segmented image. The x and y components of this vector are called here gaze x and gaze y.

**1.4.1 PROPOSED SYSTEM ADVANTAGES**

* High Security.
* Less Time Consumption.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Here, we build a drowsiness detection system that will detect that a person’s eyes are closed for a few seconds. This system will alert the driver when drowsiness is detected. Drowsiness detection is a safety technology that can prevent accidents that are caused by drivers who fell asleep while driving.

**2.2 PROBLEM DEFINATION:**

Fatigueis a safety problem that has not yet been deeply tackled by any country in the world mainly because of its nature. Fatigue, in general , is very difficult to measure or observe unlike alcohol and drugs, which have clear key indicators and tests that are available easily. Probably, the best solutions to this problem are awareness about fatigue related accidents and promoting drivers to admit fatigue when needed. The former is hard and much more expensive to achieve, and the latter is not possible without the former as driving for long hours is very lucrative.

**2.3 METHODOLOGIES**

**2.3.1 MODULES NAME:**

This project having the following six modules:

* Live Streaming
* Frame Extraction
* Facial Landmark Indexes For Face Regions
* Measurements
* Classification

**1. ­­LIVE STREAMING:**

The video is recorded utilizing webcam and the casings are extricated and handled in a PC. Subsequent to removing the casings, picture preparing methods are applied on these 2D pictures. Directly, manufactured driver information has been created. The volunteers are approached to take a gander at the webcam with irregular eye squinting, eye shutting, yawning and head bowing. The video is caught for 30 minutes term.

**2. FRAME EXTRACTION:**

OpenCV library can be used to perform multiple operations on videos. Let’s try to do something interesting using CV2. Take a video as input and break the video into frame by frame and save those frame. Now, number of operations can be performed on these frames. Like reversing the video file or crop the video etc. For playing video in reverse mode, we need only to store the frames in a list and iterate reverse in the list of frames. Use reverse method of the list for reversing the order of frames in the list.

**3. FACIAL LANDMARK INDEXES FOR FACE REGIONS**

The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 [0-67] point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.

Examining the image, we can see that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

* The mouth can be accessed through points [48, 68].
* The right eyebrow through points [17, 22].
* The left eyebrow through points [22, 27].
* The right eye using [36, 42].
* The left eye with [42, 48].
* The nose using [27, 35].
* And the jaw via [0, 17].

**4. MEASUREMENTS**

**After detecting the facial landmarks, the features are computed as described below.**

**Eye aspect ratio (EAR):** From the eye corner points, the eye aspect ratio is calculated as the ratio of height and width of the eye.

**Mouth opening ratio (MOR):** Mouth opening ratio is a parameter to detect yawning during drowsiness

**Head Rotation Ratio (HRR):** Head Rotation Ratio is a value to detect the driver is rotating his head to right, left or bending.

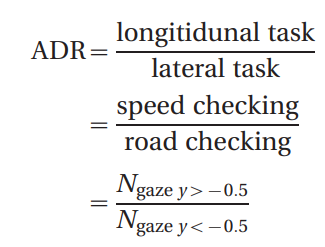
**5. CLASSIFICATION**

After computing all the three features, the next task is to detect drowsiness in the extracted frames. In the beginning, adaptive thresholding is considered for classification. Later, machine learning algorithms are used to classify the data.

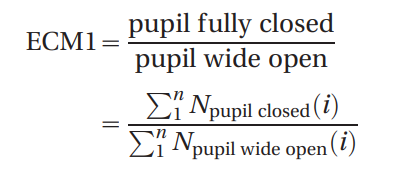
**2.4 TECHNIQUE USED OR ALGORITHM USED**

**2.3.1 Open-CV System**

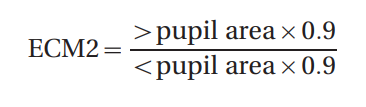
To utilize the full potential of the CV system’s output, three metrics are defined, based on the observations from histograms of measurements for six sessions for 30 subjects, based on a 180-session database of observations of the same phenomena; they are the attention division ratio (ADR) defined by

****

The first metric of eye closure (ECM1) defined by

****

And the second metric of eye closure (ECM2) defined by

****

The metrics are observed to be powerful in predicting the driver drowsiness level and they are believed to be more enriched than an eye closure metric alone (see Appendix 2).

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

In this chapter we have hardware and software requirements.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

**HARDWARE**

* PROCESSOR : PENTIUM IV 2.6 GHz, Intel Core 2 Duo.
* RAM : 512 MB DD RAM
* MONITOR : 15” COLOR
* HARD DISK : 40 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* + Front End : J2EE (JSP, SERVLET)
  + Back End : MY SQL 5.5
  + Operating System : Windows 7
  + IDE : Eclipse

**3.4 FUNCTIONAL SPECIFICATION:**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behaviour, and outputs. The essential sleepiness discovery framework has three squares/modules; procurement framework, preparing framework and cautioning framework. Here, the video of the driver's frontal face is caught in obtaining framework and moved to the handling square where it is prepared online to recognize tiredness. In the event that tiredness is recognized, an admonition or alert is send to the driver from the notice framework. The CV framework depends on a splendid understudy procedure to encourage eye identification and following. In this system we are going to control the car as well give sound alert to the driver using buzzer So as to acquire this brilliant student impact, a web camera and Arduino uno is utilized, which connects to car controller to control the driving system automatically.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**EFFICIENCY**

After detecting the facial landmarks, the features are computed as described below**.**

**Eye aspect ratio (EAR):** From the eye corner points, the eye aspect ratio is calculated as the ratio of height and width of the eye.

**Mouth opening ratio (MOR):** Mouth opening ratio is a parameter to detect yawning during drowsiness

**Head Rotation Ratio (HRR):** Head Rotation Ratio is a value to detect the driver is rotating his head to right, left or bending.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.1.1 USE CASE DIAGRAM:**



Figure No: 4.2 Use Case Diagram

**EXPLANATION:**

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.1.2 CLASS DIAGRAM :**

****

Figure No: 4.3 Class Diagram

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**4.1.3 OBJECT DIAGRAM:**



Figure No: 4.4 Object Diagram

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.1.4 STATE CHART DIAGRAM:**

****

Figure No: 4.5 State Chart Diagram

**EXPLANATION:**

State diagram are a loosely defined diagram to show workflow of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.1.5 SEQUENCE DIAGRAM:**



Figure No: 4.6 Sequence Diagram

**EXPLANATION:**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

**4.1.6 COLLABORATION DIAGRAM:**



Figure No: 4.7 Collaboration Diagram

**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). The concept is more than a decade old although it has been refined as modelling paradigms have evolved.

**4.1.7 ACTIVITY DIAGRAM:**



Figure No: 4.8 Activity Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.1.8 COMPONENT DIAGRAM:**



Figure No: 4.9 Component Diagram

**EXPLANATION:**

In the Unified Modelling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicate dependencies.

**4.1.9 DEPLOYMENT DIAGRAM:**



Figure No: 4.12 Deployment Diagram

**EXPLANATION:**

In the Unified Modelling Language, a deployment diagram depicts how are wired together to form larger deployment and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

**4.1.10 E-R DIAGRAM:**

Live Streaming

Frames

Faces

Classifying

EAR

MAR

HRR

Result

Result

Result

Figure No: 4.11 E-R Diagram

**EXPLANATION:**

Entity-Relationship Model (ERM) is an abstract and conceptual representation of data. Entity-relationship modelling is a database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database.

**4.2 DATA FLOW DIAGRAM:**

**LEVEL-0:**

Live Streaming

Classifying

Result

Figure No: 4.10.1 DFD Level 0 Diagram

**LEVEL-1:**

Live Streaming

Classifying

EAR

MAR

HRR

Result

Result

Result

Figure No: 4.10.2 DFD Level 1 Diagram

**LEVEL 2:**

Live Streaming

Frames

Faces

Classifying

EAR

MAR

HRR

Result

Result

Result

Figure No: 4.10.3 DFD Level 2 Diagram

**EXPLANATION**:

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.2.1 SYSTEM ARCHITECTURE:**

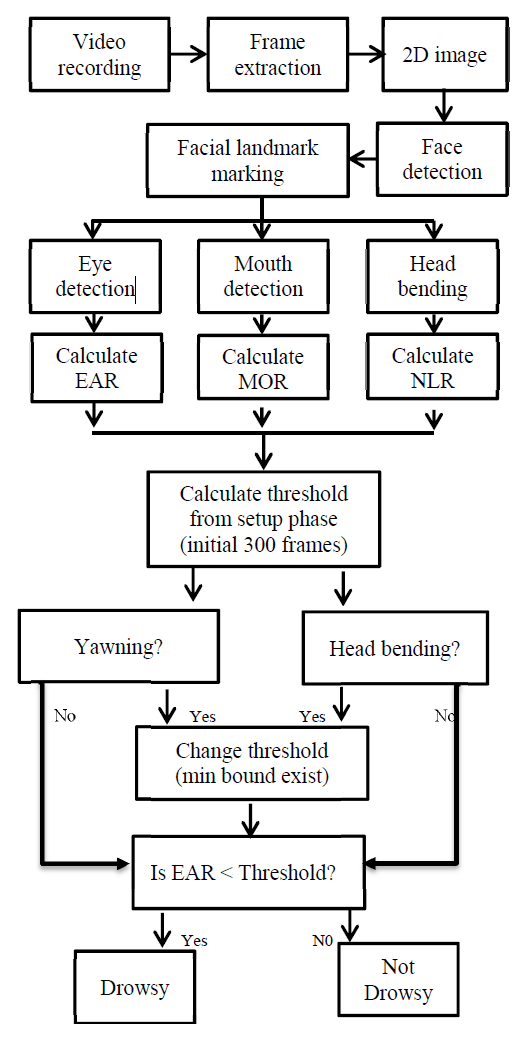


Figure No: 4.13 System Architecture Diagram

**EXPLANATION:**

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviours of the system. A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs).

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 PYTHON**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## **5.2 HISTORY OF PYTHON**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### **5.2.1 IMPORTANCE OF PYTHON**

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### **5.2.2 FEATURES OF PYTHON**

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.
* Apart from the above-mentioned features, Python has a big list of good features, few are listed below
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.2.3 Libraries used in python:**

* NumPy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as data frames.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure 5.2.3 : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

**Detect\_Drowsiness.py**

# USAGE

# python detect\_drowsiness.py --shape-predictor shape\_predictor\_68\_face\_landmarks.dat

# python detect\_drowsiness.py --shape-predictor shape\_predictor\_68\_face\_landmarks.dat --alarm alarm.wav

# import the necessary packages

from scipy.spatial import distance as dist

from imutils import face\_utils

from threading import Thread

import numpy as np

import dlib

import cv2

def sound\_alarm(path):

    # play an alarm sound

        #playsound(path)

    return ""

def get\_landmarks(im):

    rects = detector(im, 1)

    if len(rects) > 1:

        return "error"

    if len(rects) == 0:

        return "error"

    return np.matrix([[p.x, p.y] for p in predictor(im, rects[0]).parts()])

def annotate\_landmarks(im, landmarks):

    im = im.copy()

    for idx, point in enumerate(landmarks):

        pos = (point[0, 0], point[0, 1])

        cv2.putText(im, str(idx), pos,

                    fontFace=cv2.FONT\_HERSHEY\_SCRIPT\_SIMPLEX,

                    fontScale=0.4,

                    color=(0, 0, 255))

        cv2.circle(im, pos, 3, color=(0, 255, 255))

    return im

def top\_lip(landmarks):

    top\_lip\_pts = []

    for i in range(50,53):

        top\_lip\_pts.append(landmarks[i])

    for i in range(61,64):

        top\_lip\_pts.append(landmarks[i])

    top\_lip\_all\_pts = np.squeeze(np.asarray(top\_lip\_pts))

    top\_lip\_mean = np.mean(top\_lip\_all\_pts, axis=0)

    return int(top\_lip\_mean[1])

def bottom\_lip(landmarks):

    bottom\_lip\_pts = []

    for i in range(65,68):

        bottom\_lip\_pts.append(landmarks[i])

    for i in range(56,59):

        bottom\_lip\_pts.append(landmarks[i])

    bottom\_lip\_all\_pts = np.squeeze(np.asarray(bottom\_lip\_pts))

    bottom\_lip\_mean = np.mean(bottom\_lip\_all\_pts, axis=0)

    return int(bottom\_lip\_mean[1])

def mouth\_open(image):

    landmarks = get\_landmarks(image)

    if landmarks == "error":

        return image, 0

    image\_with\_landmarks = annotate\_landmarks(image, landmarks)

    top\_lip\_center = top\_lip(landmarks)

    bottom\_lip\_center = bottom\_lip(landmarks)

    lip\_distance = abs(top\_lip\_center - bottom\_lip\_center)

    return image\_with\_landmarks, lip\_distance

    #cv2.imshow('Result', image\_with\_landmarks)

    #cv2.imwrite('image\_with\_landmarks.jpg',image\_with\_landmarks)

    #cv2.waitKey(0)

    #cv2.destroyAllWindows()

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

EYE\_AR\_THRESH = 0.25

EYE\_AR\_CONSEC\_FRAMES = 48

# initialize the frame counter as well as a boolean used to

# indicate if the alarm is going off

COUNTER = 0

ALARM\_ON = False

# 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("shape\_predictor\_68\_face\_landmarks.dat")

# 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"]

# start the video stream thread

print("[INFO] starting video stream thread...")

#vs = cv2.VideoCapture('test.mov')

vs = cv2.VideoCapture(0)

#time.sleep(1.0)

read=1

yawns = 0

yawn\_status = False

# loop over frames from the video stream

while read:

    # grab the frame from the threaded video file stream, resize

    # it, and convert it to grayscale

    # channels)

    read,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)

    prev\_yawn\_status = yawn\_status

    # 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

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

        # visualize each of the eyes

        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 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

                    t = Thread(target=sound\_alarm('alarm.mav'))

                    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

        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),

            cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

        top\_lip\_center = top\_lip(shape)

        bottom\_lip\_center = bottom\_lip(shape)

        lip\_distance = abs(top\_lip\_center - bottom\_lip\_center)

        if lip\_distance > 25:

            yawn\_status = True

            cv2.putText(frame, "Subject is Yawning", (50,450),

                    cv2.FONT\_HERSHEY\_COMPLEX, 1,(0,0,255),2)

            output\_text = " Yawn Count: " + str(yawns + 1)

            cv2.putText(frame, output\_text, (50,50),

                    cv2.FONT\_HERSHEY\_COMPLEX, 1,(0,255,127),2)

        else:

            yawn\_status = False

        if prev\_yawn\_status == True and yawn\_status == False:

            yawns += 1

    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()

**Detect.py**

# -\*- coding: utf-8 -\*-

 # -\*- coding: utf-8 -\*-

# import the necessary packages

from scipy.spatial import distance as dist

from imutils.video import FileVideoStream

from imutils.video import VideoStream

from imutils import face\_utils

import numpy as np

import argparse

import imutils

import time

import dlib

import cv2

# define two constants, one for the eye aspect ratio to indicate

# blink and then a second constant for the number of consecutive

# frames the eye must be below the threshold

EYE\_AR\_THRESH = 0.2

EYE\_AR\_CONSEC\_FRAMES = 48

# initialize the frame counters and the total number of blinks

COUNTER = 0

TOTAL = 0

def rect\_to\_bb(rect):

    x=rect.left;()

    y=rect.top();

    w=rect.right()-x;

    h=rect.bottom()-y;

def crop\_center(img,cropx,cropy):

    y,x = img.shape

    startx = x//2-(cropx//2)

    starty = y//2-(cropy//2)

    return img[starty:starty+cropy,startx:startx+cropx]

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

# 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("shape\_predictor\_68\_face\_landmarks.dat")

# grab the indexes of the facial landmarks for the left and

# right eye, respectively

# start the video stream thread

print("[INFO] starting video stream thread...")

vs = cv2.VideoCapture('test.mov')

fileStream = True

print('d')

# vs = VideoStream(src=0).start()

# vs = VideoStream(usePiCamera=True).start()

# fileStream = False

#time.sleep(1.0)

# loop over frames from the video stream

while True:

    # if this is a file video stream, then we need to check if

    # there any more frames left in the buffer to process

    print('a')

    (lStart, lEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["left\_eye"]

    (rStart, rEnd) = face\_utils.FACIAL\_LANDMARKS\_IDXS["right\_eye"]

    print('b')

    # grab the frame from the threaded video file stream, resize

    # it, and convert it to grayscale

    # channels)

    ret,frame = vs.read()

    print('c')

    frame = imutils.resize(frame, width=450)

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    # detect faces in the grayscale frame

    rects = detector(gray, 0)

    # 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]

        x=leftEye[0][0]

        y=leftEye[1][1]

        w=leftEye[3][0]-leftEye[0][0]

        h=leftEye[4][1]-leftEye[1][1]

        ax=(x+w)/2

        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

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

        # visualize each of the eyes

        leftEyeHull = cv2.convexHull(leftEye)

        rightEyeHull = cv2.convexHull(rightEye)

        # 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

        # otherwise, the eye aspect ratio is not below the blink

        # threshold

        else:

            # if the eyes were closed for a sufficient number of

            # then increment the total number of blinks

            if COUNTER >= EYE\_AR\_CONSEC\_FRAMES:

                TOTAL += 1

                #cv2.rectangle(frame, (x,y),(x+w,y+h), (0, 255, 0), 1)

                #cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

                eye=frame[y:y+h,x:x+w]

                eye=cv2.resize(eye,(300,150),interpolation=cv2.INTER\_AREA)

                eye\_gray=cv2.cvtColor(eye,cv2.COLOR\_BGR2GRAY)

                eye\_blur=cv2.GaussianBlur(eye\_gray,(7,7),0)

                #\_,thre=cv2.threshold(eye\_blur,127,255,cv2.THRESH\_BINARY\_INV)

                th, thre = cv2.threshold(eye\_gray, 100, 255,cv2.THRESH\_BINARY\_INV)

                cv2.imshow("Threshold", thre)

                #count,\_=cv2.findContours(eye\_blur,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

                contours, \_ = cv2.findContours(thre, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

                contours = sorted(contours, key=lambda x: cv2.contourArea(x), reverse=True)

                for cnt in contours:

                    (x,y,w,h)=cv2.boundingRect(cnt)

                    cv2.rectangle(eye,(x,y),(x+w,y+h),(0,0,255),3)

                    px=(x+w)/2

                    py=(y+h)/2

                    if px<ax:

                        print("left")

                        break;

                    else:

                        print("right")

                        break;

                cv2.imshow("EyeBlur", eye\_blur) #Burred

                cv2.imshow("Eye", eye)

                    # reset the eye frame counter

            COUNTER = 0

        # draw the total number of blinks on the frame along with

        # the computed eye aspect ratio for the frame

        cv2.putText(frame, "Blinks: {}".format(COUNTER), (10, 30),

            cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

        cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),

            cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255), 2)

        break;

    # 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()

**HeadBending.py**

# -\*- coding: utf-8 -\*-

# import the necessary packages

from imutils import face\_utils

import numpy as np

import argparse

import imutils

import dlib

import cv2

detector = dlib.get\_frontal\_face\_detector()

predictor = dlib.shape\_predictor('shape\_predictor\_68\_face\_landmarks.dat')

# load the input image, resize it, and convert it to grayscale

vs = cv2.VideoStream(src='test.mov').start()

# loop over frames from the video stream

while True:

    # grab the frame from the threaded video file stream, resize

    # it, and convert it to grayscale

    # channels)

    frame,image = vs.read()

    image = imutils.resize(image, width=500)

    gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

    # detect faces in the grayscale image

    rects = detector(gray, 1)

    # loop over the face detections

    for (i, rect) in enumerate(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)

        points = face\_utils.FACIAL\_LANDMARKS\_IDXS["nose"]

        print(points)

        # show the output image with the face detections + facial landmarks

    cv2.imshow("Output", image)

    cv2.waitKey(0)

**6.2 IMPLEMENTATION:**

* In our program we used Dlib, a pre-trained program trained on the HELEN dataset to detect human faces using the pre-defined 68 landmarks.

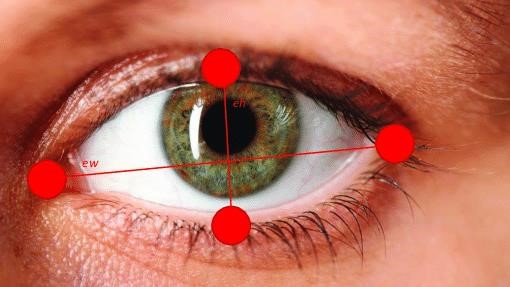


Landmarked Image of a person by Dlib



HELEN Dataset Samples

* After passing our video feed to the dlib frame by frame, we are able to detect left eye and right eye features of the face.
* Now, we drew contours around it using OpenCV.
* Using Scipy’s Euclidean function, we calculated sum of both eyes’ aspect ratio which is the sum of 2 distinct vertical distances between the eyelids divided by its horizontal distance.



Eyes with horizontal and vertical distance marked for Eye Aspect Ratio calculation.

* Now we check if the aspect ratio value is less than 0.25 (0.25 was chosen as a base case after some tests). If it is less an alarm is sounded and user is warned.

**CHAPTER 7**

**SNAPSHOTS**

**7.1 General:**

This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**7.2 VARIOUS SNAPSHOTS**

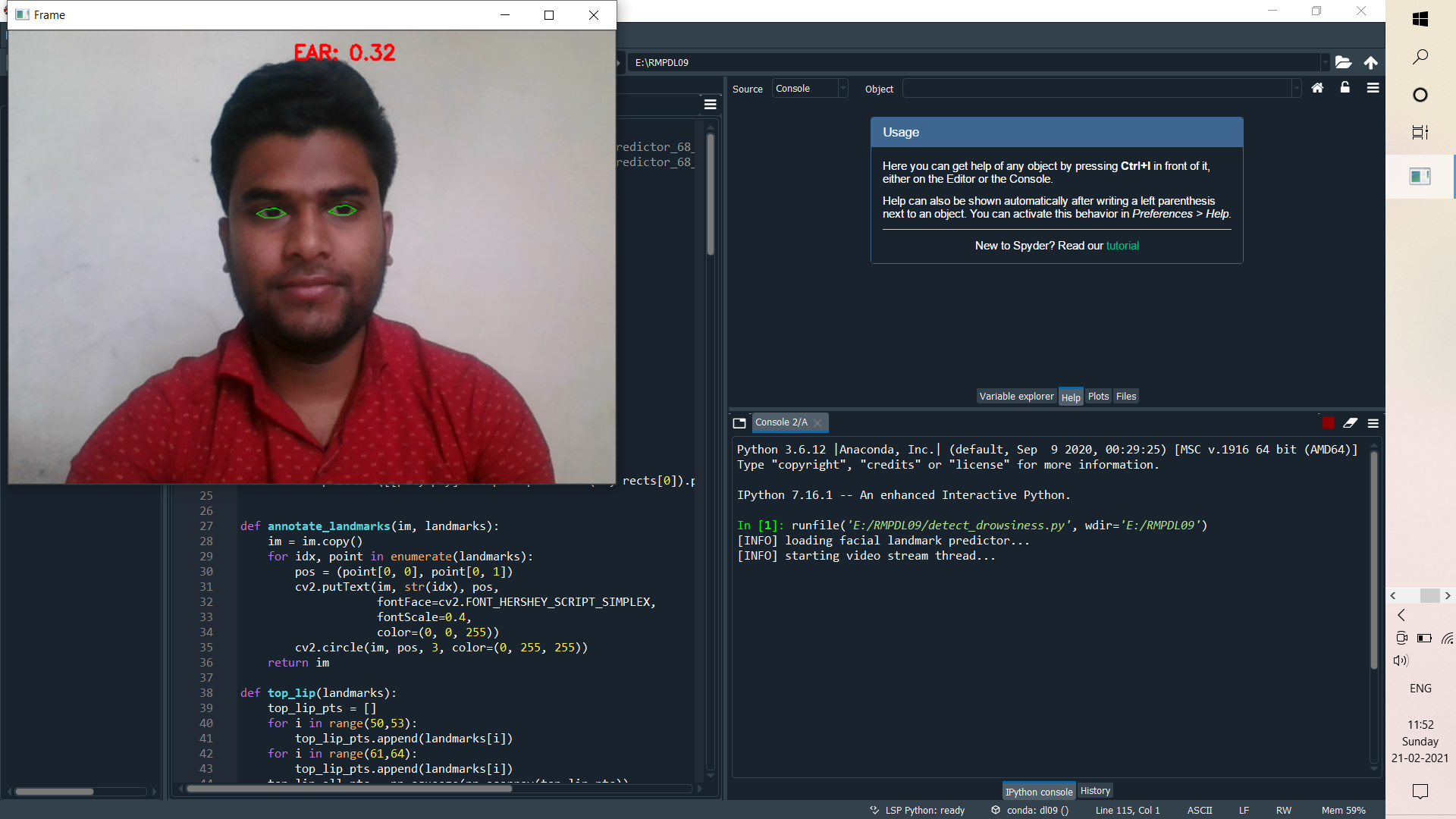
****

Figure No: 7.2.1 Detection of Drowsiness

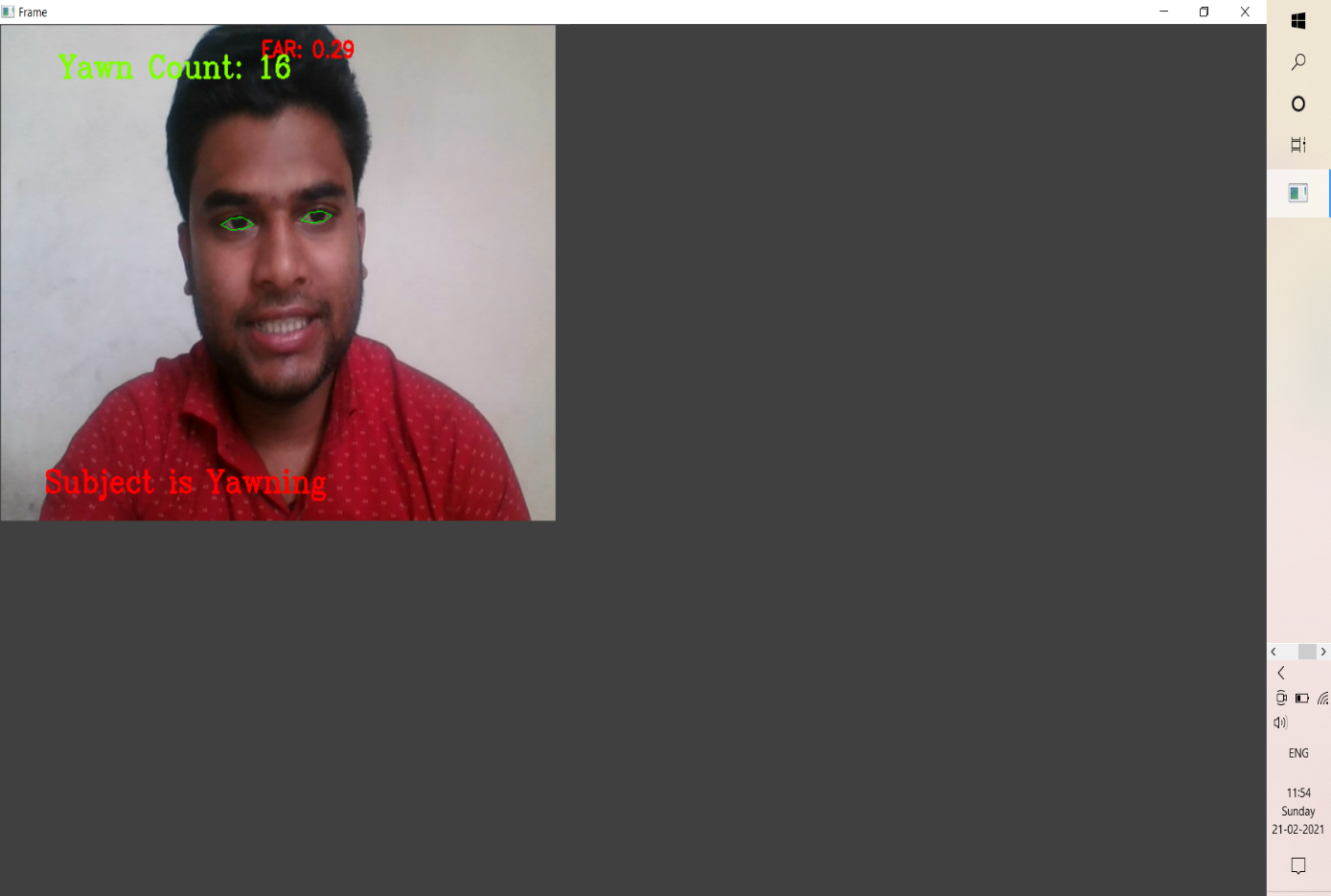
****

Figure No: 7.2.2 Yawning identification

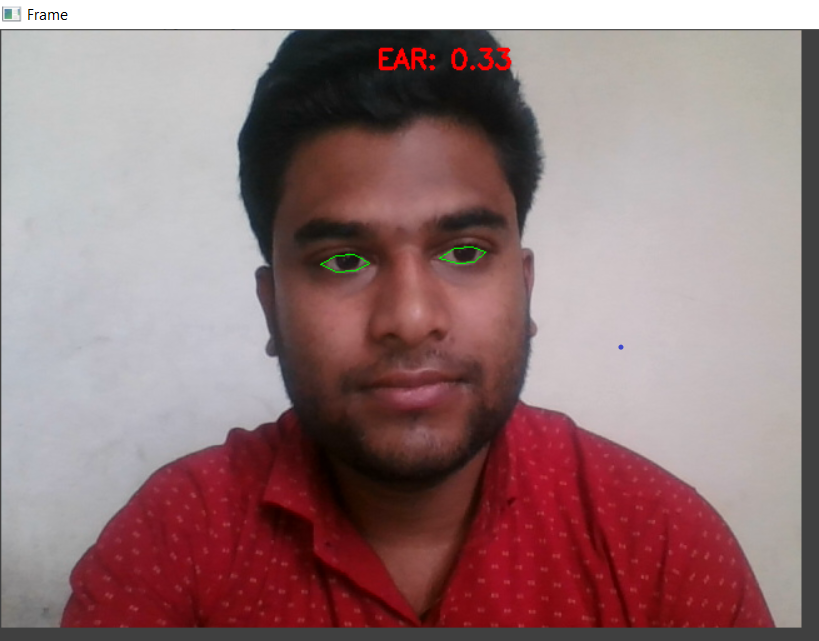


Figure No: 7.2.3 Detecting Eye Aspect Ratio

**EXPLANATION:**

In above figure it shows that the person Yawning, Yawning Count: 16 and Ear Count: 0.29 of that person.

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 TYPES OF TESTS**

**8.3.1 Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**8.3.2 Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is cantered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**8.3.3 System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

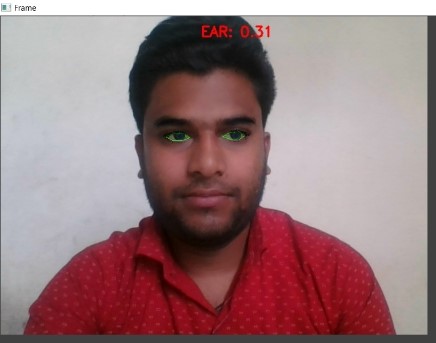
**8.3.6 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**8.4 TEST CASES AND TEST RESULTS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test ID | Test Case Title | Test Condition | System Behaviour | Expected Result |
| T01 | NSGY | Straight Face, Good Light, With Glasses | Non Drowsy | Non Drowsy |
| T02 | YTGN | Tilted Face, Good Light, No Glasses | Drowsy | Drowsy |
| T03 | YTGY | Tilted Face, Good Light, With Glasses | Drowsy | Drowsy |

**Note: Testing is performed manually**

**  **

Test Id: T01 Test Id: T02 Test Id: T03

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node.
* The Route add operation is done only when there is a Route request in need.
* The Status of Nodes information is done automatically in the Cache Updation process.

**8.5 BUILD THE TEST PLAN:**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Scenario | Test Case | Test Data | Expected Result | Success/Failure |
| Check with valid data | Provide Valid Video | Provide Image Frames | EAR Value / Yawning Status / Drowsiness Alert | Success |
| Check with invalid data | Provide invalid  Video | Provide Image Frames | No Alert and No Predictions | Success |

**CHAPTER 9**

**APPLICATION**

**9.1 GENERAL**

In addition to eye closure, the attention of the driver as a function of time can be obtained by measuring the eye movement based on a vector between the centres of glint and pupil in the segmented image. The x and y components of this vector are called here gaze x and gaze y.

**9.2 FUTURE ENHANCEMENT**

The sensitivity of FLDA and SVM is 0.896 and 0.956 respectively whereas the specificity is 1 for both. As FLDA and SVM give better accuracy, work will be carried out to implement them in the developed system to do the classification (i.e., drowsiness0 detection) online. Also, the system will be implemented in hardware to make it portable for car system and pilot study on drivers will be carried out to validate the developed system.

**CHAPTER 10**

**CONCLUSION & REFERENCE**

**10.1 CONCLUSION**

In this paper, a low cost, real time driver drowsiness monitoring system has been proposed based on visual behaviour and machine learning. Here, visual behaviour features like eye aspect ratio, mouth opening ratio and nose length ratio are computed from the streaming video, captured by a webcam. An adaptive thresholding technique has been developed to detect driver drowsiness in real time. The developed system works accurately with the generated synthetic data. Subsequently, the feature values are stored and machine learning algorithms have been used for classification. Bayesian classifier, FLDA and SVM have been explored here. It has been observed that FLDA and SVM outperform Bayesian classifier.

**10.2 REFERENCE:**

[1] W. L. Ou, M. H. Shih, C. W. Chang, X. H. Yu, C. P. Fan, "Intelligent Video-Based Drowsy Driver Detection System under Various Illuminations and Embedded Software Implementation", 2015 international Conf. on Consumer Electronics - Taiwan, 2015.

[2] W. B. Horng, C. Y. Chen, Y. Chang, C. H. Fan, “Driver Fatigue Detection based on Eye Tracking and Dynamic Template Matching”, IEEE International Conference on Networking,, Sensing and Control, Taipei, Taiwan, March 21-23, 2004.

[3] S. Singh, N. P. papa Nikolopoulos, “Monitoring Driver Fatigue using Facial Analysis Techniques”, IEEE Conference on Intelligent Transportation System, pp 314-318.

[4] B. Alshaqaqi, A. S. Baquhaizel, M. E. A. Ouis, M. Bouumehed, A. Ouamri, M. Keche, “Driver Drowsiness Detection System”, IEEE International Workshop on Systems, Signal Processing and their Applications, 2013.

[5] M. Karchani, A. Mazloumi, G. N. Saraji, A. Nahvi, K. S. Haghighi, B. M. Abadi, A. R. Foroshani, A. Niknezhad, “The Steps of Proposed Drowsiness Detection System Design based on Image Processing in Simulator Driving”, International Research Journal of Applied and Basic Sciences, vol. 9(6), pp 878-887, 2015.

[6] R. Ahmad, and J. N. Borole, “Drowsy Driver Identification Using Eye Blink Detection,” IJISET - International Journal of Computer Science and Information Technologies, vol. 6, no. 1, pp. 270-274, Jan. 2015.

[7] A. Abas, J. Mellor, and X. Chen, “Non-intrusive drowsiness detection by employing Support Vector Machine,” 2014 20th International Conference on Automation and Computing (ICAC), Bedfordshire, UK, 2014, pp. 188- 193.

[8] A. Sengupta, A. Dasgupta, A. Chaudhuri, A. George, A. Routray, R. Guha; "A Multimodal System for Assessing Alertness Levels Due to Cognitive Loading", IEEE Trans. on Neural Systems and Rehabilitation Engg., vol. 25 (7), pp 1037-1046, 2017.

[9] K. T. Chui, K. F. Tsang, H. R. Chi, B. W. K. Ling, and C. K. Wu, “An accurate ECG based transportation safety drowsiness detection scheme,” IEEE Transactions on Industrial Informatics, vol. 12, no. 4, pp. 1438- 1452, Aug. 2016.

[10] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection”, IEEE conf. on CVPR, 2005.

[11] V. Kazemi and J. Sullivan; "One millisecond face alignment with an ensemble of regression trees", IEEE Conf. on Computer Vision and Pattern Recognition, 23-28 June, 2014, Columbus, OH, USA.

[12] Richard O. Duda, Peter E. Hart, David G. Stork, “Pattern Classification”, Wiley student edition.

[13] Dataset: https://sites.google.com/site/invedrifac/344