DRIVER DROWSINESS MONITORING SYSTEM USING VISUAL BEHAVIOUR AND MACHINE LEARNING

A PROJECT REPORT
Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY
IN
INFORMATION TECHNOLOGY

Submitted By

RAPOLU PAVAN KUMAR
18BQ1A12D2

THOTA CHARAN
18BQ1A12G0

SATENNAPALLI SRINIVASA RAO
18BQ1A12D9

Under the Supervision of
Mrs. Sk. Mulla Almas
Assistant Prof.

DEPARTMENT OF INFORMATION TECHNOLOGY
VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY

Jawaharlal Nehru Technological University, Kakinada.

JUNE & 2022

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY: : NAMBUR

BONAFIDE CERTIFICATE

This is to certify that this project report **"Driver Drowsiness Detection using Visual Behavior and Machine Learning"** is the bonafide work of "**R. PAVAN KUMAR (18BQ1A12D2) T. CHARAN (18BQ1A12G0) S. SRINIVASA RAO (18BQ1A12D9)**", who carried out the project under the supervision of **"Mrs. Sk. Mulla Almas"** academic year 2021-22 towards partial fulfillment of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada. The results embodied in this report have not been submitted to any other University for the award of any degree.

Signature of the Head of the Department	Signature of the Supervisor
Dr. A. Kalavathi	**Mrs. Sk. Mulla Almas**
HEAD OF THE DEPARTMENT	**GUIDE**
Department of INFORMATION TECHNOLOGY	Assistant Prof. INFORMATION TECHNOLOGY

External Viva voce conducted on _____

Internal Examiner **External Examiner**

VASIREDDY VENKATADRI INSTITUTE OF TECHNOLOGY:: NAMBUR

CERTIFICATE OF AUTHENTICATION

I solemnly declare that this project report **"Driver Drowsiness Detection using Visual Behavior and Machine Learning"** is the bonafide work done totally by us, carried out under the supervision of **Sk. Mulla Almas**, towards partial fulfillment of the requirements of the Degree of Bachelor of Technology in Information Technology from Jawaharlal Nehru Technological University, Kakinada during the year 2021-22.

It is further certified that this work has NOT been submitted, either in part or in full, to any other department of Jawaharlal Nehru Technological University, or any other University, Institution or elsewhere, or for publication in any form.

Signature of the Student

DATE:

RAPOLU PAVAN KUMAR
18BQ1A12D2, 2021-22

THOTA CHARAN
18BQ1A12G0, 2021-22

SATENNAPALLI SRINIVASA RAO
18BQ1A12D9,2021-22

ACKNOWLEDGEMENT

We take this opportunity to express our deepest gratitude and appreciation to all those people who made this project work easier with words of encouragement, motivation, discipline, and faith by offering different places to look to expand my ideas and helped me towards the successful completion of this project work.

First and foremost, we express our deepest gratitude to **Mr. Vasireddy Vidya Sagar**, Chairman, Vasireddy Venkatadri Institute of Technology for providing necessary facilities throughout the Information Technology program.

We express our sincere thanks to **Dr. Y. Mallikarjuna Reddy**, Principal, Vasireddy Venkatadri Institute of Technology for his constant support and cooperation throughout the Information Technology program.

We express our sincere gratitude to **Dr. A. Kalavathi**, Professor & HOD, Information Technology, Vasireddy Venkatadri Institute of Technology for her constant encouragement, motivation and faith by offering different places to look to expand my ideas. We would like to express our sincere gratefulness to our guide **Sk. Mulla Almas**, MTech, and Project Coordinator, **Sirisha**, Associate Professor, further insightful advice, motivating suggestions, invaluable guidance, help and support in successful completion of this project.

We would like to take this opportunity to express our thanks to the teaching and non-teaching staff in Department of Information Technology,
VVIT for their invaluable help and support.

ABSTRACT

Drowsy driving is one of the major causes of road accidents and death. Hence, detection of driver's fatigue and its indication is an active research area. Most of the conventional methods are either vehicle-based, behavioral-based or physiological-based. Few methods are intrusive and distract the driver, some require expensive sensors and data handling.

Therefore, in this project, a low-cost, real-time driver's drowsiness detection system has been developed by us with acceptable accuracy. In the developed system, a webcam records the video and the driver's face is detected in each frame employing image processing techniques.

Facial landmarks on the detected face are pointed and subsequently the eye aspect ratio (EAR), mouth opening ratio (MOR) and nose length ratio (NLR) are computed and depending on their values, drowsiness is detected based on developed adaptive threshold.

Table of Contents

CHAPTER III SYSTEM DESIGN

CHAPTER IV SYSTEM IMPLEMEMTATION

CHAPTER V SYSTEM TESTING

CHAPTER VI CONCLUSION

CHAPTER VII Appendices

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1	SVM	Support Vector Machine
2	HOG	Histogram of Oriented Gradients
3	FLDA	Fisher ’ s linear discriminant analysis
4	EAR	Eye Aspect Ratio
5	MOR	Mouth Opening Ratio
6	NLR	Nose Length Ratio

CHAPTER I - INTRODUCTION

1.1 INTRODUCTION:

Driver drowsiness is one of the major causes of deaths occurring in road accidents. The truck drivers who drive for continuous long hours (especially at night), bus drivers of long-distance route or overnight buses are more susceptible to this problem. Driver drowsiness is an overcast nightmare to passengers in every country. Every year, a large number of injuries and deaths occur due to fatigue related road accidents. Various studies have suggested that around 20% of all road accidents are fatigue-related, upto 50% on certain roads. Hence, detection of driver's fatigue and its indication is an active area of research due to its immense practical applicability.

The development of drowsiness detection technologies is both an industrial and academic challenge. In the automotive industry, Volvo developed the Driver Alert Control which warns drivers suspected of drowsy driving by using a vehicle-mounted camera connected to its lane departure warning system (LDWS). Following a similar vein, an Attention Assist System has been developed and introduced by Mercedes-Benz that collects data drawn from a driver's driving patterns incessantly ascertains if the obtained information correlates with the steering movement and the driving circumstance at hand. The driver drowsiness detection system, supplied by Bosch, takes decisions based on data derived from the sensor stationed at the steering, the vehicles' driving velocity,turn signal use, and the camera mounted at the front of the car.

Notably, the use of these safety systems which detect drowsiness is not widespread and is uncommon among drivers because they are usually available in luxury vehicles. An increased embedding and connecting of smart devices equipped with sensors and mobile operating systems like Android, which has the largest installed operating system in cars, was shown by surveys in 2015. In addition, machine learning has made ground-breaking advances in recent years, especially in the area of deep learning. Thus, the use of these new technologies and methodologies can be an effective way to not only increase the efficiencies of the existing real-time driver drowsiness detection system but also provide a tool that can be widely used by drivers.

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.

Generally, the methods to detect drowsy drivers are classified in three types; vehicle based, behavioral based and physiological based. In vehicle-based method, a number of metrics like steering wheel movement, accelerator or brake pattern, vehicle speed, lateral acceleration, deviations from lane position etc. are monitored continuously. Detection of any abnormal change in these values is considered as driver drowsiness. This is a non-intrusive measurement as the sensors are not attached on the driver.

In behavioral based method, the visual behavior of driver i.e.., hand movement, leg movement etc., are analyzed to detect drowsiness. This is also a non-intrusive measurement as the sensors are not attached on the driver.

In physiological based method, the physiological signals like Electrocardiogram (ECG), Electro-oculogram (EOG), Electroencephalogram (EEG), heartbeat, pulse rate etc. are monitored and from these metrics, drowsiness or fatigue level is detected. This is intrusive measurement as the sensors are attached on the driver which will distract the driver. Depending on the sensors used in the system, system cost as well assize will increase. However, inclusion of more parameters/features will increase the accuracy of the system to a certain extent.

These are the motivating factors to develop a low-cost, real time driver's drowsiness detection system with acceptable accuracy. Hence, we developed a webcam-based system to detect driver's fatigue from the face image only using image processing and machine learning techniques to make the system low-cost as well as portable.

People in fatigue show some visual behaviors easily observable from changes in their facial features like eyes, head, mouth and face. Computer vision can be a natural and non-intrusive technique to monitor driver's vigilance. Face is the primary part of human

communication have been a research target in computer vision for a long time. The driver fatigue detection is considered as one of the most prospective commercial applications of automatic facial expression recognition. Automatic recognition (or analysis) of facial expression consists of three levels of tasks: face detection, facial expression information extraction, and expression classification. In these tasks, the information extraction is the main issue for the feature based facial expression recognition from an image sequence. It involves detection, identification and tracking facial feature points under different illuminations, face orientations and facial expressions.

Some common assumptions in previous face related works were: frontal facial views, constant illumination, and the fixed lighting source. Unfortunately, these assumptions are not realistic. In the application of real-world facial expression understanding, one has to consider at least three issues: capturing the full features in a variety of lighting conditions and head motion, multiple and non-rigid object tracking, and the self-occlusion of features.

The process of falling asleep at the wheel can be characterized by a gradual decline in alertness from a normal state due to monotonous driving conditions or other environmental factors; this diminished alertness leads to a state of fuzzy consciousness followed by the onset of fatigue. The critical issue that a fatigue detection system must address is the question of how to accurately and early detect fatigue at the initial stage. Possible non-intrusive techniques for detecting fatigue in drivers using computer vision are

- Methods based on eye and eyelid movements
- Methods based on head movement
- Methods based on mouth opening

The system uses Histogram Oriented Gradient (HOG) feature descriptor for face detection and facial points recognition. Then SVM is used to checkwhether detected object is face or non-face. It further monitors the Eye Aspect Ratio (EAR) , Mouth Opening Ratio (MOR) and Nose Length Ratio (NLR) of the driver up to a fixed number of frames to check the sleepiness, yawning and head bending.

1.2 Aim and Scope:

The major aim of this project is to develop a drowsiness detection system by monitoring the eyes; it is believed that the symptoms of driver fatigue can be detected early enough to avoid car accident. In such a case when drowsiness is detected, a warning signal is issued to alert the driver along with sending an SMS to the emergency contacts.

1.3 Goals and Objectives:

The objective is to build a detection system that identifies key attributes of drowsiness and triggers an alert when someone is drowsy before it is too late. In this project by monitoring visual behavior of a driver with webcam and machine learning SVM (support vector machine) algorithm we are detecting Drowsiness in a driver. This application will use inbuilt webcam to read images of a driver and then use OPENCV SVM algorithm extract facial features from the image and then check whether driver in image is blinking his eyes or yawning mouth or bending head for consecutive 10 frames and if this behavior is observed then application will send an alert to driver. We are using SVM pre-trained drowsiness model and using Euclidean distance function we are continuously predicting Eye Aspect Ratio, Mouth Opening Ratio and Nose Length Ratio. If these are closer to values of drowsy driver, then application will alert driver.

1.4 Description of the Existed System:

In the existing system it just detects the driver is drowsy or not based on only the one parameter that's our eyes. If eyes are closed then the existing system concludes that the driver is drowsy, which is a major drawback. Moreover, by seeing only eyes of a person we can't conclude the drowsiness.

1.5 Description of the Proposed System:

In the proposed system we are using SVM pre-trained drowsiness model and using Euclidean distance function we are continuously predicting Eye Aspect Ratio, Mouth Opening Ratio and Nose Length Ratio. If these are closer to values of drowsy driver, then application will alert driver by playing an alarm sound and moreover it sends the message to the emergency contacts such that we can eradicate the happening of accident in advance.

1.6 Feasibility Study:

A feasibility study is a preliminary study which investigates the information of prospective users and determines the resources requirements, costs, benefits and feasibility of proposed system. A feasibility study takes into account various constraints within which the system should be implemented and operated. In this stage, the resource needed for the implementation such as computing equipment, manpower and costs are estimated. The estimated are compared with available resources and a cost benefit analysis of the system is made. The feasibility analysis activity involves the analysis of the problem and collection of all relevant information relating to the project. The main objectives of the feasibility study are to determine whether the project would be feasible in terms of economic feasibility, technical feasibility and operational feasibility and schedule feasibility or not. It is to make sure that the input data which are required for the project are available. Thus, we evaluated the feasibility of the system.

1.6.1 Technical Feasibility:

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system

- Does the necessary technology exist to do what is suggested?
- Do the proposed equipment have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

1.6.2 Operational Feasibility:

The project is operationally feasible as the user having basic knowledge about computer and Internet. It is based on operational performance of the model depending on the training dataset.

- Is there sufficient support for the management from the users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the user that will undermine the possible application benefits?

1.6.3 Economical Feasibility:

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. A simple economic analysis which gives the actual comparison of costs and benefits are much more meaningful in this case. In addition, this proves to be useful point of reference to compare actual costs as the project progresses.

CHAPTER II - REQUIREMENT SPECIFICATION

2.1 Description of the Problem:

The main goal of this project is to develop a non-intrusive system for vehicles that can find the driver's tiredness and concern a warning with time. Because there are a great number of traffic accidents due to fatigue of the drivers, this system aspires to avoid many crashes on roads, thus saving money and minimizing personal suffering. The developed system continually monitors the driver's mouth, eye, and head through the real-time camera which is focused at the driver's face. The changes in mouth and eyes are analyzed and then processed to find the tiredness of the drivers and also to send alarm. This approach is simple and less complex as no training is required compared to the existing approaches. Three possible cases such as eye closure, yawing, and head tilt are considered for fatigue detection of the driver. Therefore, this approach helps to anticipate the fatigue of the driver and also gives a warning output in the form of alarm.

2.2 Proposed Solution:

The critical issue that a fatigue detection system must address is the question of how to accurately and early detect fatigue at the initial stage. Possible non intrusive techniques for detecting fatigue in drivers using computer vision, capturing video, extract frames, detect face in each frame and marking facial landmarks.

The system uses Histogram Oriented Gradient (HOG) feature descriptor for face detection and facial points recognition. Then SVM is used to check whether detected object is face or non-face. It further monitors the Eye Aspect Ratio (EAR) , Mouth Opening Ratio (MOR) and Nose Length Ratio (NLR) of the driver up to a fixed number of frames to check the sleepiness, yawning and head bending. If any of these is observed then drowsiness is identified and an alarm is sent.

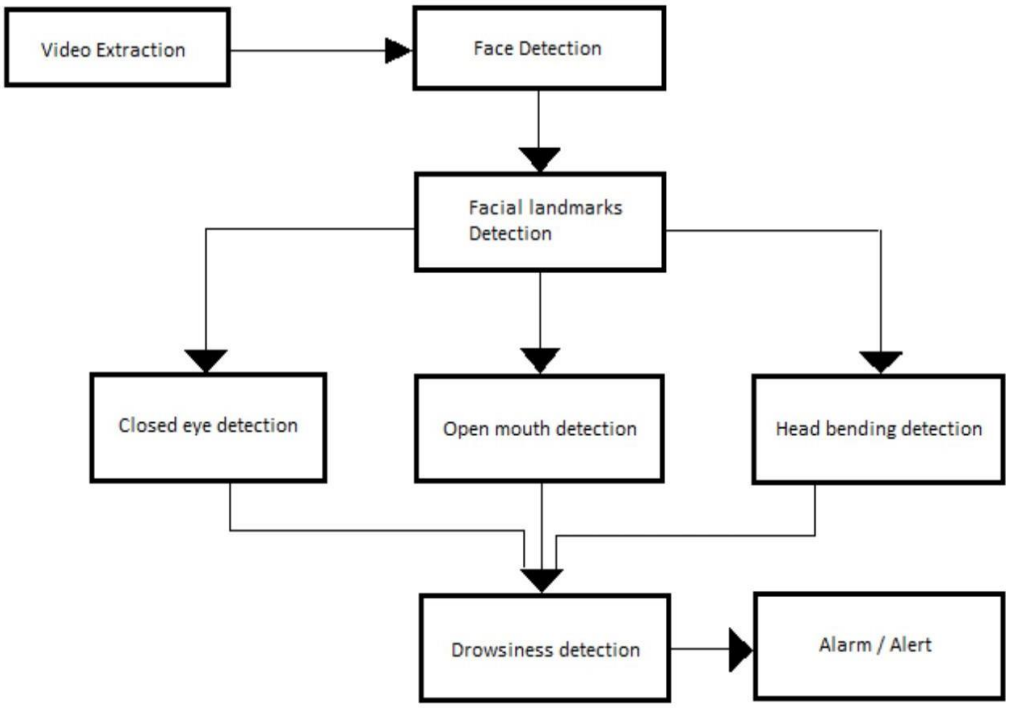


Fig 2.2.1 Proposed Solution

2.3 System Requirements:

2.3.1 External Interfaces:

As the project was developed only in the local system, so there is no requirement of connection to the external world. The system will accept the input from the user and produces the output within the system itself.

2.3.2 User Interfaces:

The user interface that is required for the project is a well-organized user's device camera to record the video of the user.

2.3.3 Hardware Interfaces:

1. **Processor** : Intel CORE i3 processor with minimum 2.9 GHz speed.
2. **Display** : LCD/LED Color.
3. **Accessories** : Web Cam, Keyboard & Mouse.
4. **RAM** : Minimum 4 GB (Recommended 8GB).
5. **Hard Disk** : Minimum 500 GB

2.3.4 Software Interfaces:

1. **Operating System:** Windows10, Linux.
2. **Python IDLE** (Latest)

2.3.5 Communication Interfaces:

2.3.5.1 Functional Requirements:

1. System should be able to monitor the driver by capturing video and take the video as input.
2. Each frame of video generates facial landmarks and different assessment metrics are calculated for each frame.
3. System should be able to identify the whether driver is drowsy or not correctly.
4. System should alert the driver when the driver is said to have drowsiness.

2.3.5.2 Non-Functional Requirements:

Reliability **:** The system is highly reliable.

Accessibility **:** It can be easily accessible i.e. click & run.

Efficiency **:** Resource consumption for given load is quite low.

Robustness **:** Our system is not capable to cope with errors during execution.

Fault Tolerance : Our system is not fault tolerant due to insufficient hardware.

2.3.6 Modules:

1.Face Detection:

The face is recognized initially by using the user's camera and the data is passed to next module.

2.Eyes Detection:

The faces are then used to detect the posture and position of the eyes of the face and these outcomes are passed to extract the feature values.

3.Mouth Detection:

Further the lips of the user face is identified and the outcomes are used to extract the features of the lips.

2.4 System Analysis Methods:

2.4.1 Use Case Diagram:

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actors and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior. Actors are not part of the system. Actors represent anyone or anything that interacts with (input to or receive output from) the system. Use-case diagrams can be used during analysis to capture the system requirements and to understand how the system should work. During the design phase, you can use use-case diagrams to specify the behavior of the system as implemented. Use case is a sequence of transactions performed by a system that yields a measurable result of values for a particular actor. The use cases are all the ways the system may be used.

Use case is a list of actions or event steps, typically defining the interactions between a role (known as an actor) and a system, to achieve a goal. In case of the use case diagram developer and the end user are the actors. Use casesare best discovered by examining the actors and defining what the actor will be able to do with the system. Since all the needs of a system typically cannot be covered in one use case, it is usual to have a collection of use cases. Together this use case collection specifies all the ways the system. An include relationship is a stereotyped relationship that connects a base use case to an inclusion use case.

An include relationship specifies how behavior in the inclusion use case is used by the base use case. An extend relationship is a stereotyped relationship that specifies how the functionality of one use case can be inserted into thefunctionality of another use case. Extend relationships between uses cases are modeled as dependencies by using the Extend stereotype. The different use cases are import dataset for importing datasets, preprocessing, model building, validation and prediction.

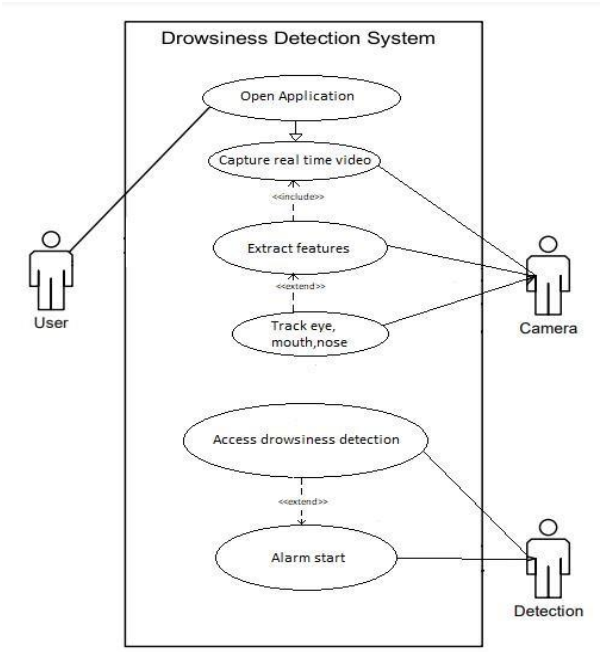


Fig 2.4.2.1 – Use Case Diagram

2.4.2 Activity Diagram:

An activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose ofActivity diagram is to provide a view of flows and what is going on inside a use case or among several classes. Activity diagrams contain activities, transitions between the activities, decision points, and synchronization bars. An activity represents the performance of some behavior in the workflow. In the UML, activities are represented as rectangles with rounded edges, transitions are drawn as directed arrows, decision points are shown as diamonds, and synchronization bars are drawn as thick horizontal or vertical bars as shown in the following. The activity icon appears as a rectangle with rounded ends with a name and a component for actions.

Swim lanes may be used to partition an activity diagram. This typically is done to show what person or organization is responsible for the activitiescontained in the swim lane. Swim lanes are helpful when modeling a business workflow because they can represent organizational units or roles within a business model. Swim lanes are very

similar to an object because they provide a way to tell who is performing a certain role. Swim lanes only appear on activity diagrams. When a swim lane is dragged onto an activity diagram, it becomes a swim lane view. Swim lanes appear as small icons in the browse while swim lane views appear between the thin, vertical lines with a header that can be renamed and relocated.

An activity represents the performance of some behavior in the work flow. In the UML, activities are represented as rectangles with rounded edges, transitions are drawn as directed arrows, decision points are shownas diamonds, and synchronization bars are drawn as thick horizontal or vertical bars as shown in the following. The activity icon appears as a rectangle with rounded ends with a name and a component for actions. An activity diagram is a variation of a special case of a state machine, in which the states are activities representing the performance of operations and the transitions are triggered by the completion of the operations. The purpose of Activity diagram is to provide a view of flows and what is going on inside a use case or among several classes.

Activity diagrams can be regarded as a form of a structured flowchart combined with a traditional data flow diagram. Typical flowchart techniques lack con- structs for expressing concurrency. However, the join and split symbols in activity diagrams only resolve this for simple cases; the meaning of the model is not clear when they are arbitrarily combined with decisions or loops.

Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- ellipses represent actions.
- diamonds represent decisions.
- bars represent the start (split) or end (join) of concurrent activities.
- a black circle represents the start (initial node) of the work flow.
- an encircled black circle represents the end (final node).

Arrows run from the start towards the end and represent the order in which activities happen.

The work flow in this case begins from importing the dataset by the developer and then preprocessing the data, feature extraction, model building, validating that model by generating a histogram for lbp descriptor and finally predicting the test sample class label.

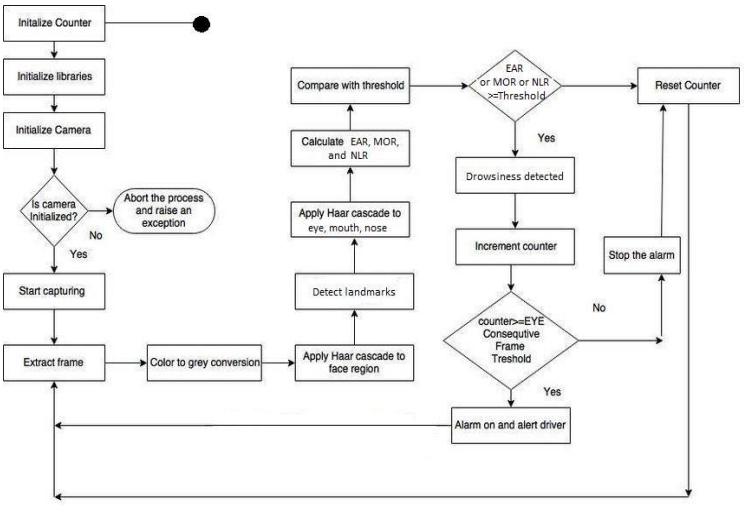


Fig 2.4.4 – Activity Diagram

2.5 System Design Methods:

2.5.1 Class Diagram:

Class diagrams contain icons representing classes, interfaces, and their relationships. You can create one or more class diagrams to represent the classes atthe top level of the current model; such class diagrams are themselves contained by the top level of the current model. You can also create one or more class diagrams to represent classes contained by each package in your model; such class diagrams are themselves contained by the package enclosing the classes they represent; the icons representing logical packages and classes in class diagrams.

1. Class diagrams are created to provide a picture or view of some or all of the classes in the model.

2. The main class diagram in the logical view of the model is typically a picture of the packages in the' system. Each package also has its own main class diagram, which typically displays the public classes of the package.

A Class diagram is a picture for describing generic descriptions of possible systems. Class diagrams and collaboration diagrams are alternate representations of object models.

A Class is a description of a group of objects with common properties (attributes) common behavior (operations), common relationships to other objects, and common semantics. Thus, a class is a template to create objects. Each object is an instance of some class and objects cannot be instances of more than one class.

In the UML, classes are represented as compartmentalized rectangles.

1. The top compartment contains the name of the class.
2. The middle compartment contains the structure of the class (attributes).
3. The bottom compartment contains the behavior of the class(operations).

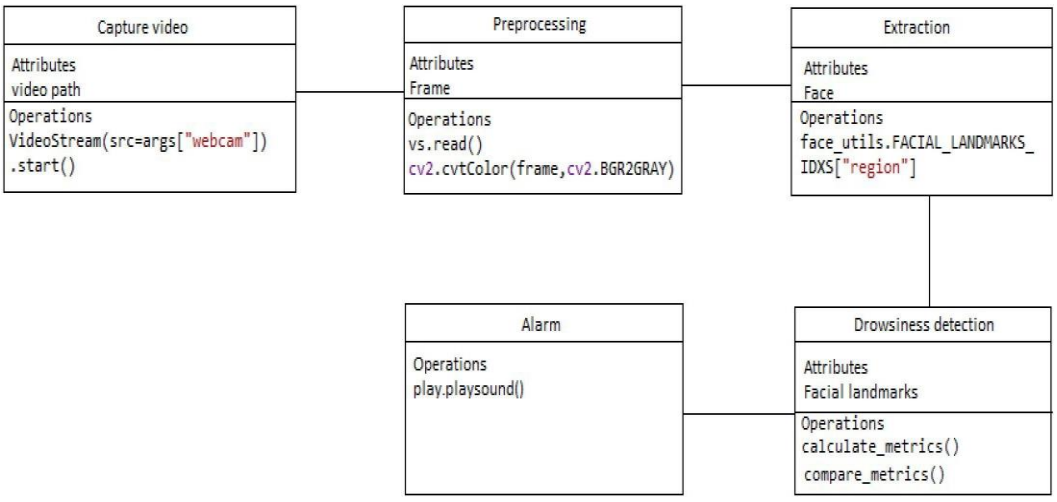


Fig 2.5.2.1 – Class Diagram

2.5.2 Sequence Diagram:

A sequence diagram is an 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 thesequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called as event diagrams.

A sequence diagram shows, as parallel vertical lines (lifelines), different pro- cesses or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifelineis that of an object, it demonstrates a role. Leaving the instance name blank can represent anonymous and unnamed instances.

Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages. If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response. Asynchronous calls are present in multithreaded applications, event-driven applications and in message-oriented middleware. Activation boxes, or method- call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message.

Objects calling methods on themselves use messages and add new activation boxes on top of any others to indicate a further level of processing. If an object is destroyed, an X is drawn on bottom of the lifeline, and the dashed line ceases to be drawn below it. It should be the result of a message, either from the object itself, or

another.

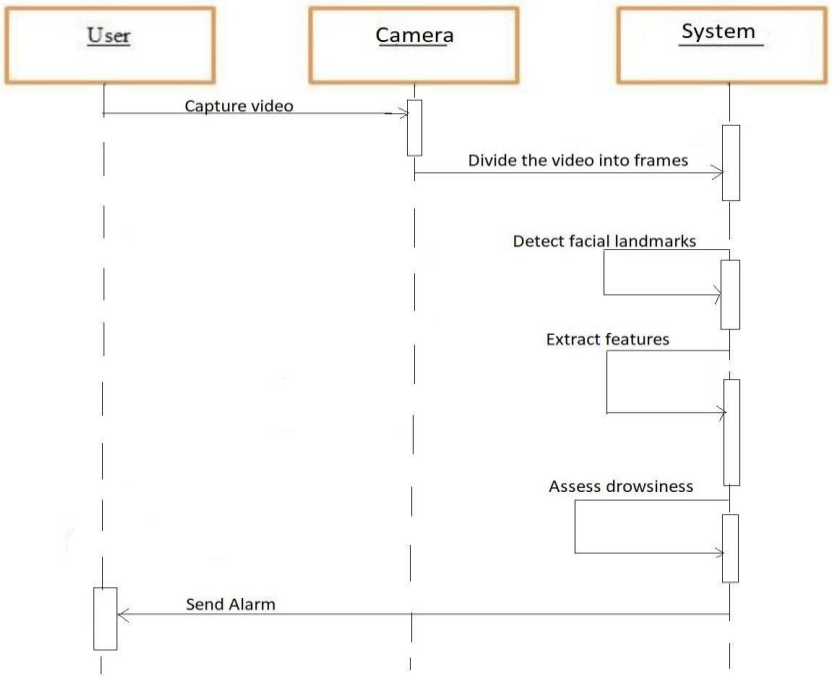


Fig 2.5.3 - Sequence Diagram

CHAPTER III SYSTEM DESIGN

3.1 Introduction:

3.1.1 Design Overview:

It is well known that when fully connected network is used, the light intensity of each pixel in the input image is regarded as the value of the corresponding neuron in the input layer. For example, for a 28*28-pixel image, the number of input neurons is 784, leading to the high dimension of network parameters, which is difficult to train. In addition, the fully connected network architecture does not consider the spatial structure of images, and treats input pixels that are far away or close to each other on the exact same basis, making it difficult for the network to learn spatial structure information. The CNN adopts an architecture suitable for image characteristics, which makes the network model more consistent with data structure characteristics and is conducive to the training of deep and multi-layer networks.

3.2 System Architectural Design:

The architecture of a system describes its major components, their relationships (structures), and how they interact with each other. Software architecture and design includes several contributory factors such as Business strategy, quality attributes, human dynamics, design, and IT environment.

Software Architecture

Architecture serves as a **blueprint for a system**. It provides an abstraction to manage the system complexity and establish a communication and coordination mechanism among components. It defines a **structured solution** to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security.

Goals of Architecture

The primary goal of the architecture is to identify requirements that affect the structure of the application. A well-laid architecture reduces the business risks associated with building a technical solution and builds a bridge between business and technical requirements.

3.2.1 Chosen System Architecture:

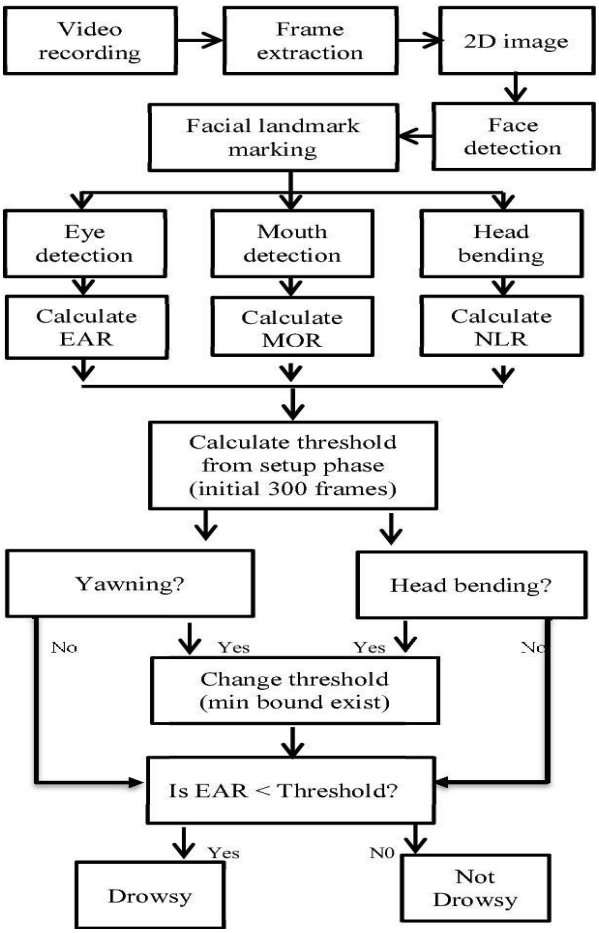


Fig 3.2.1.1 - Architecture

3.2.2 System Interface Descriptions:

Training Stage:

In this stage nearly 70% of the dataset is used to train the model. The training can be done by developing programming model and by passing the data to it.

Testing Stage:

In this stage nearly 30% of the dataset images are used to test the trained model or in application level the user's camera is used to test the model.

Camera:

The user's camera is the main interface to use the application and it observes the scenario and in live the face of user is recorded and passed on to the model.

3.3 Detailed Description of the Components:

Pre-Trained Model:

It is the model that was developed after training using the training dataset.

Fine Tuning:

Fine-tuning is a common technique for transfer learning. The target model copies all model designs with their parameters from the source model except the output layer, and fine-tunes these parameters based on the target dataset. In contrast, the output layer of the target model needs to be trained from scratch.

Crop Face:

In this process, from the scenario captured from the user's camera device the face of the user is identified and only the face is taken into consideration.

3.4 User Interface Design:

3.4.1 Description of the user interface:

The system user interface is the IDLE where the main logic of the application resides and the user can access the application through the tkinter application.

3.4.3 Objects and Actions:

Objects	Actions
Camera	Capture the face of the user
Trained Model	To accept the new images and predict the emotion.

CHAPTER IV - SYSTEM IMPLEMENTATION

4.1 Tools and Technologies Used:

The Library & Packages:

OpenCV:

OpenCV (Open-Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Ego motion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Support vector machine (SVM)
- Deep neural networks (DNN)

NumPy:

NumPy is an acronym for "Numeric Python" or "Numerical Python". It is an open-source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines. Furthermore, NumPy enriches the programming language Python with powerful data structures for efficient computation of multi-dimensional arrays and matrices. The implementation is even aiming at huge matrices and arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) function
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier Transform, and random number capabilities.

NumPy Array:

A **NumPy array** is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the **array**; the shape of an **array** is a tuple of integers giving the size ofthe **array** along each dimension.

Python IDLE:

IDLE is Python’s Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and macOS
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

Tkinter:

Tkinter allows you to develop desktop applications. It's a very good tool for GUI programming in Python. Tkinter is pronounced as tea-kay-inter. Tkinter is the Python interface to Tk, which is the GUI toolkit for Tcl/Tk.

Tcl (pronounced as tickle) is a scripting language often used in testing, prototyping, and GUI development. Tk is an open-source, cross-platform widget toolkit used by many different programming languages to build GUI programs.

Tkinter is a good choice because of the following reasons:

- Easy to learn.
- Use very little code to make a functional desktop application.
- Layered design.
- Portable across all operating systems including Windows, macOS, and Linux.
- Pre-installed with the standard Python library

SciPy:

The SciPy is an open-source scientific library of Python that is distributed under a BSD license. It is used to solve the complex scientific and mathematical problems. It is built on top of the NumPy extension, which means if we import the SciPy, there is no need to import NumPy. The **SciPy** is pronounced as **Sigh pi**, and it depends on the NumPy, including the appropriate and fast N-dimension array manipulation.

It provides many user-friendly and effective numerical functions for numerical integration and optimization.

The **SciPy** library supports **integration, gradient optimization, special functions, ordinary differential equation solvers, parallel programming tools**, and many more. We can say that **SciPy** implementation exists in every complex numerical computation.

imutils:

imutils is a package based on OpenCV, which can call the OpenCV interface more simply. It can easily realize a series of operations such as image translation, rotation, scaling, skeletonization and so on.

argparse:

The argparse module in Python helps create a program in a command-line-environment in a way that appears not only easy to code but also improves interaction. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid argument.

playsound:

playsound is a “pure Python, cross platform, single function module with no dependencies for playing sounds.” With this module, you can play a sound file with a single line of code: from playsound import playsound playsound('myfile.wav’)

time:

Python's time module offers a wide variety of time-related features and is often handy when building your backend with Python. Using this library, you can fetch the current time and date in standard formats.

dlib:

dlib is a toolkit for making real world machine learning and data analysis applications in C++. While the library is originally written in C++, it has good, easy to use Python bindings. It can be majorly used for face detection and facial landmark detection.

To implement dlib module in python we are incorporating a software known as CMAKE.

CMake:

CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native make files and workspaces that can be used in the compiler environment of your choice.

LITERATURE STUDY:

As per various literature surveys it is found that for implementing this project four basic stepsare required to be performed.

i. Preprocessing
ii. Face registration
iii. Facial feature extraction
iv. Emotion classification

Description about all these processes is given below-

Preprocessing:

Preprocessing is a common name for operations with images at the lowest level of abstraction both input and output are intensity images. Most preprocessing steps that are implemented are –

a. Reduce the noise

b. Convert The Image To Binary/Grayscale.

c. Pixel Brightness Transformation.

Fig 4.1.1 Preprocessing

Face Registration:

Face Registration is a computer technology being used in a variety of applications that identifies human faces in digital images. In this face registration step, faces are first located in the image using some set of landmark points called “face localization” or “face detection”. These detected faces are then geometrically normalized to match some template image in a process called “face registration”.

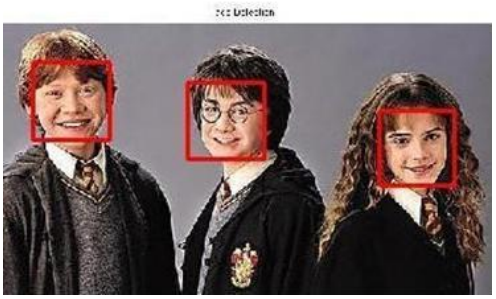


Fig 4.1.2 – Face Detection

Facial Feature Extraction:

Facial Features extraction is an important step in face recognition and is defined as the process of locating specific regions, points, landmarks, or curves/contours in a given 2-D image or a 3D range image. In this feature extraction step, a numerical feature vector is generated from the resulting registered image. Common features that can be extracted are-

a. Lips
b. Eyes
c. Nose tip

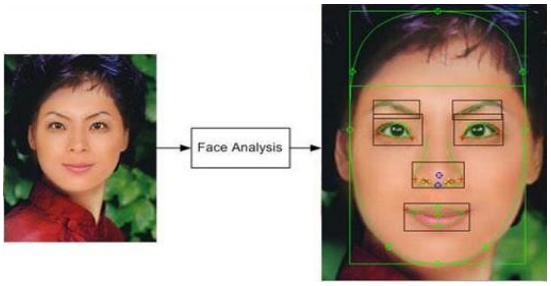


Fig 4.1.3 – Feature Extraction

4.2 Implementation:

Modules to be implemented are:

A. Data Acquisition

B. Face Detection

C. Facial landmark marking

D. Feature Extraction Classification

E. Classification

A. Data Acquisition

The video is recorded using webcam and the frames are extracted and processed in a laptop. After extracting the frames, image processingtechniques are applied on these 2D images. Presently, synthetic driver data has been generated. The volunteers are asked to look at the webcam with intermittent eye blinking, eye closing, yawning and head bending.

B. Face Detection

After extracting the frames, first the human faces are detected in each frame. Numerous online face detection algorithms are there. Human faces are detected using histogram of oriented gradients (HOG) and linear SVM method. In this method, positive samples of descriptors are computed on them. Subsequently, negative samples (samples that do not contain the required object to be detected i.e., human face here) of same size are taken and HOG descriptors are calculated. Usually, the number of negative samples is very greater than number of positive samples. After obtaining the features for both the classes, a linear SVM is trained for the classification task. To improve the accuracy of SVM, hard negative mining is used. In this method, after training, the classifier is tested on the labeled data and the false positive sample feature values are used again for training for the test image, the fixed size window is translated over the image and the classifier computes the output for each window location. Finally, the maximum value output is considered as the detected face

and a bounding box is drawn around the face. This non-maximum suppression step removes the redundant and overlapping bounding boxes

C. Facial Landmark marking

After detecting the face, the next task is to find the locations of different facial features like the corners of the eyes and mouth, the tipoff the nose and so on. Prior to that, the face images should be normalized in order to reduce the effect of distance from the camera, non-uniform illumination and varying image resolution. Therefore, the face image is resized to a width of 500 pixels and converted to gray- scale image.

After image normalization, ensemble of regression trees is used to estimate the landmark positions on face from a sparse subset of pixel intensities. In this method, the sum of square error loss is optimized using gradient boosting learning. Different priors are used to find different structures. Using this method, the boundary points of eyes, mouth and the central line of the nose are marked and the number of points for eye, mouth and nose are given in below table.

Parts	Landmark Points
Mouth	[13-24]
Right eye	[1-6]
Left eye	[7-12]
Nose	[25-28]

Fig: 4.2.C.1 Facial Landmark Points

The facial landmarks are shown in below Figure. The red points arethe detected landmarks for further processing.

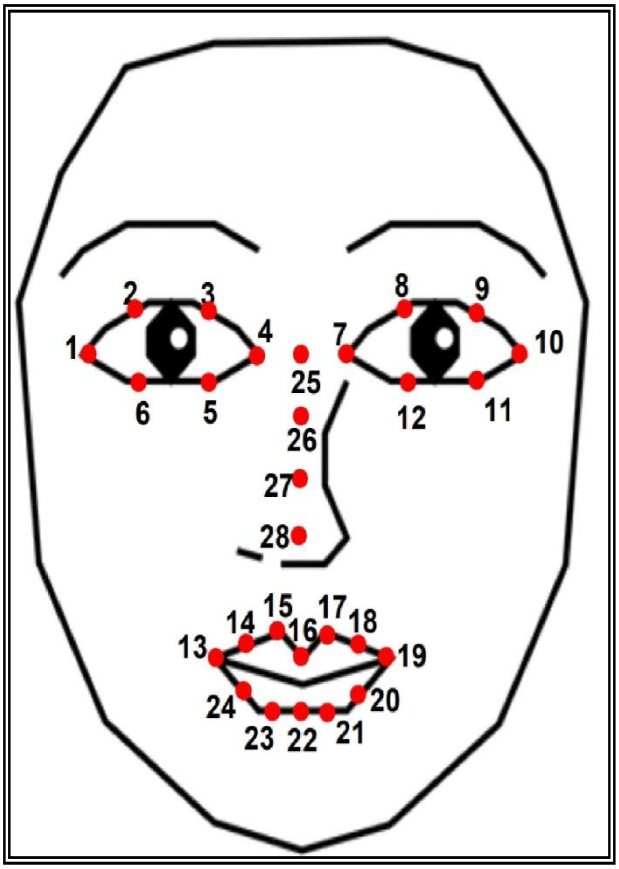


Fig 4.1.4 – 68 Facial Landmarks

D. Feature Extraction

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 iscalculated as the ratio of height and width of the eye as given by,

$$EAR = \frac{(p_2 - p_6) + (p_3 - p_5)}{2(p_4 - p_1)}$$

where p_i represents point marked as i in facial landmark and $(p_i - p_j)$ is the distance between points marked as i and j. Therefore, when the eyes are fully open, EAR is high value and as the eyes are closed, EAR value goes towards zero. Thus, monotonically decreasing EAR values indicate gradually closing eyes and it's almost zero for completely closed eyes (eye blink). Consequently, EAR values indicate the drowsiness of the driver as eye blinks occur due to drowsiness.

Mouth opening ratio (MOR):

Mouth opening ratio is a parameter to detect yawning during drowsiness. Similar to EAR, it is calculated as

$$MOR = \frac{(p_{15} - p_{23}) + (p_{16} - p_{22}) + (p_{17} - p_{21})}{3(p_{19} - p_{13})}$$

As defined, it increases rapidly when mouth opens due to yawning and remains at that high value for a while due to yawn (indicating that the mouth is open) and again decreases rapidly towards zero. As yawn is one ofthe characteristics of drowsiness, MOR gives a measure regarding driver drowsiness.

Head Bending:

Due to drowsiness, usually driver's head tilts (forward or backward) with respect to vertical axis. So, from the head bending angle,driver drowsiness can be detected. As the projected length of nose on the camera focal plane is proportional to this bending, it can be used as a measure of head bending. In normal condition, our nose makes an acute angle with respect to focal plane of the camera. This angle increases as the head moves vertically up and decreases on moving down. Therefore, the ratio of nose length to an average nose length while awake is a measure of head bending and if the value is greater or less than a particular range, it indicates head bending as well as drowsiness. From the facial landmarks, the noselength is calculated and it is defined as

$$NLR = \frac{nose\ length(p_{28} - p_{25})}{average\ nose\ length}$$

The average nose length is computed during the setup phase of the experiment.

E. 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. For computing the threshold values for each feature, it is assumed that initially the driver is in complete awake state. This is called setup phase. In the setup phase, the EAR values for first seventy-five (for 10s at 30 fps) frames are recorded. Out of these seventy-five initial frames containing face, average of 75 values is considered as thehard threshold for EAR.

The higher values are considered so that no eye closing instances will be present. If the test value is less than this threshold, then eye closing (i.e., drowsiness) is detected. As the size of eye can vary from person to person, this initial setup for each person will reduce this effect. Similarly, for calculating threshold of MOR, since the mouth may not be open to its maximum in initial frames (setup phase) so the threshold is taken experimentally from the

observations. If the test value is greater than this threshold then yawn (i.e., drowsiness) is detected.

Head bending feature is used to find the angle made by head with respect to vertical axis in terms of ratio of projected nose lengths. Normally, NLR has values from 0.9 to 1.1 for normal upright position of head and it increases or decreases when head bends down or up in the state of drowsiness. The average nose length is computed as the average of the nose lengths in the setup phase assuming that no head bending is there. After computing the threshold values, the system is used for testing.

The system detects the drowsiness if in a test frame drowsiness is detected for at least one feature. To make this thresholding more realistic, the decision for each frame depends on the last 15 frames. If at least 10 frames (out of those 15) satisfy drowsiness conditions for at least one feature, then the system gives drowsiness detection indication and the alarm. To make this thresholding adaptive, another single threshold value is computed which initially depends on EAR threshold value. The average of EAR values is computed as the average of 75 values out of 75 frames in the setup phase. Then offset is determined heuristically and the threshold is obtained as offsetsubtracted from the average value.

Driver safety is at risk when EAR is below this threshold. This EAR threshold value increases slightly with each yawning and head bending upto a certain limit. As each yawning and head bending is distributed over multiple frames, so yawning and head bending of consecutive frames are considered as single yawn and head bending and added once in the adaptive threshold. In a test frame, if EAR value is less than this adaptive threshold value, then drowsiness is detected and an alarm is given to the driver. Sometimes it may happen that when the head is too low due to bending, the system is unable to detect the face. In such situation, previous three frames are considered and if head bending was detected in the three frames, drowsiness alert will be shown.

4.2 ALGORITHMS

A. HAAR FEATURES

The face detection method uses Haar features for face detection. Haar features are extracted by using a set of rectangular black and white windows. The black color has a weight of -1 and the white region has weight 0. The windows are firstapplied to the image and corresponding values are multiplied with the pixel intensities. Then these values are added together, and the Haar feature corresponding to the window used is obtained. But all the Haar features extracted are not required for successful detection of faces. Hence the most important features that can be used for face detection are taken.

B. CASCADE CLASSIFIERS

Once the Haar features are obtained then individual classifiers are built based on the values of each Haar feature. These individual classifiers are then arranged into a cascade classifier. A cascaded classifier is combination of several classifiers arranged in the different stages cascaded on after one another. The number of classifiers in each stage and their threshold values are determined by the boosting algorithm during the training of the classifiers with labeled face images. The cascade classifier used here has 22 stages and a total of 2135 features.

C. HOG FEATURES

The HOG features method was developed by Dalal and Trigs in 2005. It is a feature descriptor that is used for various object detection applications in the field of computer vision. The key idea of the HOG features is to group gradient magnitudes into bins in a histogram based on its orientation. HOG features are extracted from the eye images obtained in the previous section. For that the image

is first resizes into 24x24 pixels. Then the image is divided into 4 blocks of size 16x16

pixels with each block overlapping half of the region covered by the preceding block. Each block has 4 cells of size 8x8 pixels. Next the gradients are computed for each pixel inside the cells using Sobel filters. These gradients magnitudes are then plotted in a histogram which has magnitudes on the y-axis and orientations in the x-axis. The x-axis is divided into 9 bins, each bin having a width of 20 degrees. The gradient magnitudes are then arranged into these 9-bin histograms based on their orientation. The value of each of these 9 bins corresponds to the feature values extracted from each cell of the image. Thus each cell of 64 pixels is represented using 9 feature values. Then these feature values of all the cells inside each block are concatenated to obtain the final feature descriptor.

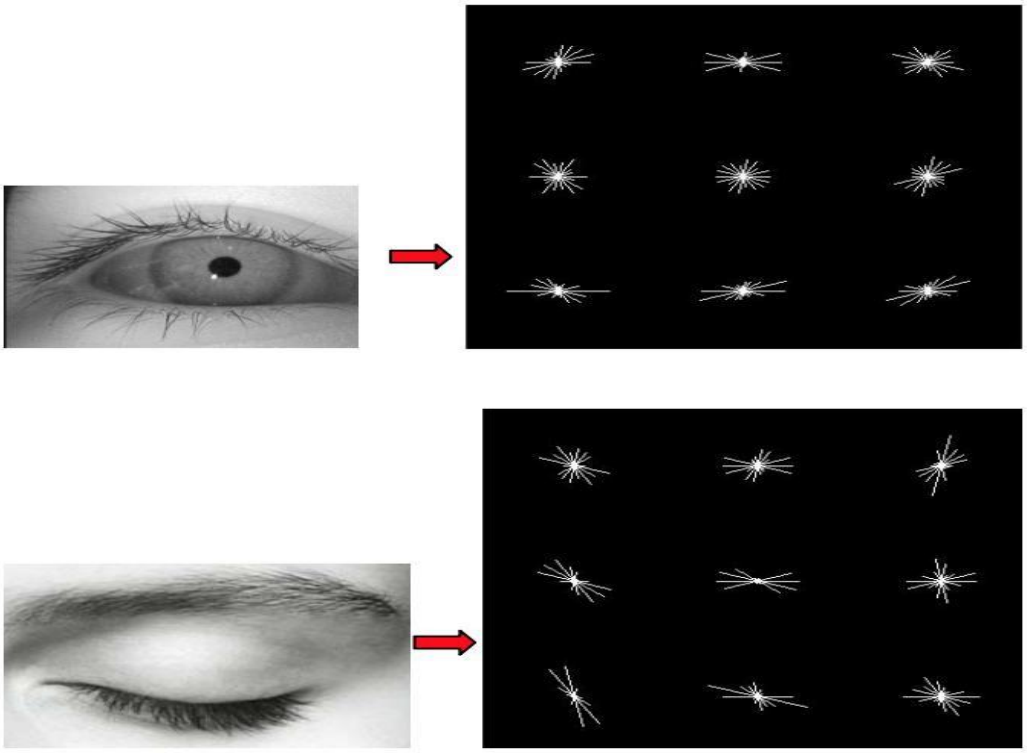


Fig 4.2.1 - HOG features extracted from open and closed eyes

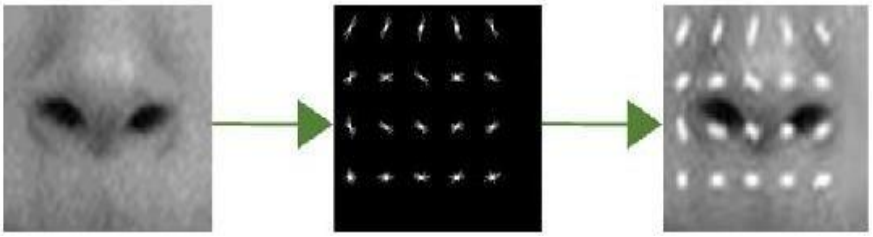


Fig. 4.2.2 HOG features extracted from nose

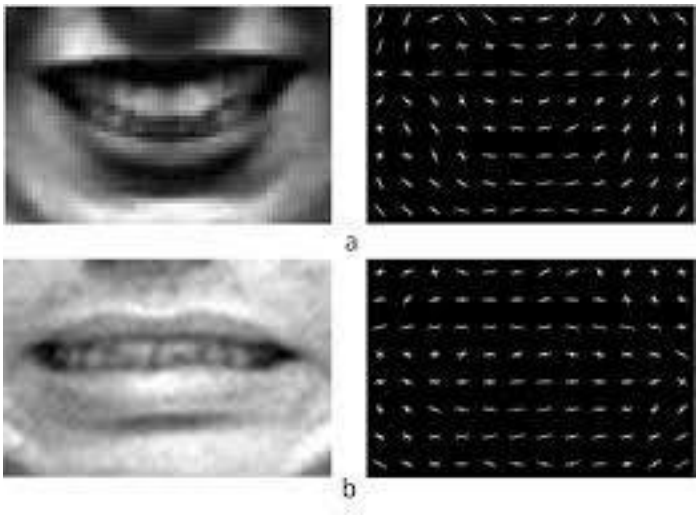


Fig. 4.2.3 HOG features extracted from mouth

D. SVM CLASSIFIER

Support Vector Machines were initially developed by Vapnik and his team. It was later improved by other researchers. It is a statistical learning model used commonly for classification problems. Let the data points of the eye images be represented by: (h_1,y_1), (h_2,y_2) (h_n,y_n) were hi represent the HOG feature vector
representing the nth eye image and y_n represent the class of the nth eye image. y can have two values 0 or a 1. 0 represents the closed eye image and 1 represents the open eye image. The basic idea of the HOG features is to find a hyper plane with the maximum margin that separates the two classes. In case of linearly separable data the hyper plane in terms of support vectors is given by:

$$f(h) = \sum_{i=1}^{n} \alpha_i y_i h_i(.) h(i) + b$$

where y_i denotes the class of the data point hi and h(i) represents the support vector machines. This is a Lagrange optimization problem and it is solved using Lagrange multipliers α_i (i = 1,...l). In case the data is not linearly separable then it is first mapped into another feature space where it is linearly separable using Kernel function. Then the equation becomes:

$$f(h) = \sum_{i=1}^{n} \alpha_i y_i V(h_i, h(i)) + b$$

where V represents the Kernel function. In the presented system the SVM classifier uses a Gaussian Radial Basis function for mapping the data into another feature space. The classifiers are trained using 40 eye images each of open and closed eyes. Once the classifier is trained then the test images are classified applied to the classifier for classification.

4.3 DATA SOURCES USED

For experimental analysis, a pre-trained HOG + Linear SVM objectdetector was employed mainly for the task of face detection. This pre-trained object detector is shape_predictor_68_face_landmarks.dat. This model creates predictor object. The video of the subject under test is captured using the web camera. The video captured is stored as a collection of frames (images). Each of these frames are extracted and processed separately. Each frame is sent to the pre- trained object detector to detect face. Packages like tkinter, scipy, imutils, numpy, dlib, cv2, argparse, playsound are used.

4.3 Screen Shots:

```
from tkinter import *
import tkinter
from scipy.spatial import distance as dist
from imutils import face_utils
from imutils.video import VideoStream
import imutils
import numpy as np
import argparse
import time
import dlib
import cv2
import playsound as play
from Mouth_Opening_Ratio import MOR
from Ear_Aspect_Ratio import EAR
from Nose_Length_Ratio import NLR
from Send_Sms import sendSms

main = tkinter.Tk()
main.title("Driver Drowsiness Monitoring")
main.geometry("500x400")

def startMonitoring():
    NOSE_AVERAGE = 1
    EYE_AR_THRESHOLD =0
    pathlabel.config(text="Webcam Connected Successfully")
    webcamera = cv2.VideoCapture(0)
    #vs = VideoStream(src=args["webcam"]).start()
    ap = argparse.ArgumentParser()
    ap.add_argument("-w", "--webcam", type=int, default=0,help="index of webcam on system")
    args = vars(ap.parse_args())
    vs = VideoStream(src=args["webcam"]).start()
    svm_predictor_path = 'C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/SVMclassifier.dat'
    svm_detector = dlib.get_frontal_face_detector()
    svm_predictor = dlib.shape_predictor(svm_predictor_path)
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    (sNos,eNos) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
    print("SetUp loading...")
    print("Please keep ur HEAD STRAIGHT toward's camera")
    time.sleep(5)
    print("Taking values please wait...")
for i in range(75):
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = svm_detector(gray, 0)
    for rect in rects:
        point1 = 0.0
        ear_sum=0.0
        shape = svm_predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        nose = shape[sNos:eNos]
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = EAR(leftEye)
        rightEAR = EAR(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        ear_sum+=ear
        EYE_AR_THRESHOLD=ear_sum
        point = dist.euclidean(nose[3], nose[0])
        point1 += point
        NOSE_AVERAGE = point1
        #print("Avrg: ", NOSE_AVERAGE)
NOSE_AVERAGE= NOSE_AVERAGE/75
print("EYE_AVERAGE: ",EYE_AR_THRESHOLD)
EYE_AR_CONSEC_FRAMES = 10
EYE_AR_THRESH = 0.25
NOSE_LENGTH_UP = 0.7
NOSE_LENGTH_DOWN = 1.1
MOU_AR_THRESH = 0.75
COUNTER1 = 0
COUNTER = 0
yawnStatus = False
yawns = 0
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
```

Importing the required modules and loading the 68 landmarks dataset and finding the left eye, right eye, nose values of the user.

```
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    prev_yawn_status = yawnStatus
    rects = svm_detector(gray, 0)
    for rect in rects:
        shape = svm_predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        mouth = shape[mStart:mEnd]
        nose = shape[sNos:eNos]
        leftEAR = EAR(leftEye)
        rightEAR = EAR(rightEye)
        mouEAR = MOR(mouth)
        noseNLR = NLR(nose,NOSE_AVERAGE)
        ear = (leftEAR + rightEAR) / 2.0
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        mouthHull = cv2.convexHull(mouth)
        noseHull = cv2.convexHull(nose)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 255), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 255), 1)
        cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [noseHull], -1, (0, 255, 0), 1)
        if noseNLR>NOSE_LENGTH_DOWN or noseNLR<NOSE_LENGTH_UP:
            #print("Head Bending")
            COUNTER1 = COUNTER1+1
            cv2.putText(frame, "NLR: {:.2f}".format(noseNLR), (480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if COUNTER1>EYE_AR_CONSEC_FRAMES:
                play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
                sendSms()
                cv2.putText(frame, "DROWSINESS ALERT!", (10, 130),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.putText(frame, "Head Bending!", (10, 50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                #cv2.putText(frame, "NLR: {:.2f}".format(noseNLR), (480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                cv2.putText(frame, "Head Bending!", (10, 50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        else:
            COUNTER1 = 0
            cv2.putText(frame, "NLR: {:.2f}".format(noseNLR), (480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        if ear < EYE_AR_THRESH:
            COUNTER += 1
            cv2.putText(frame, "Eyes Closed ", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                cv2.putText(frame, "DROWSINESS ALERT!", (10, 50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
                sendSms()
        else:
            COUNTER = 0
            #print("EYE_AVERAGE: ",EYE_AR_THRESHOLD)
            cv2.putText(frame, "Eyes Open ", (10, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
        cv2.putText(frame, "EAR: {:.2f}".format(ear), (480, 30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        if mouEAR > MOU_AR_THRESH:
            cv2.putText(frame, "Yawning, DROWSINESS ALERT! ", (10, 70),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
            sendSms()
            yawnStatus = True
            output_text = "Yawn Count: " + str(yawns + 1)
            cv2.putText(frame, output_text, (10,100),cv2.FONT_HERSHEY_SIMPLEX, 0.7,(255,0,0),2)
        else:
            yawnStatus = False
        if prev_yawn_status == True and yawnStatus == False:
            yawns+=1
        cv2.putText(frame, "MOR: {:.2f}".format(mouEAR), (480, 60),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        #cv2.putText(frame,"Visual Behaviour & Machine Learning Drowsiness Detection @ Drowsiness",(370,470),cv2.FONT_HERSHEY_COMPLEX,0.6,(153,51,102),1)
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
cv2.destroyAllWindows()
vs.stop()
webcamera.release()
```

Finding the MOR, NLR,EAR values and depending on the threshold values we play the alarm sound and send sms

```
font = ('times', 16, 'bold')
title = Label(main, text='Driver Drowsiness Monitoring System using Visual\nBehaviour and Machine Learning',anchor=W, justify=LEFT)
title.config(bg='black', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)


font1 = ('times', 14, 'bold')
upload = Button(main, text="Start Behaviour Monitoring Using Webcam", command=startMonitoring)
upload.place(x=50,y=200)
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='DarkOrange1', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50,y=250)


main.config(bg='chocolate1')
main.mainloop()
```

Designing the tkinter window for monitoring

CHAPTER V - SYSTEM TESTING:

5.1 Introduction:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.

In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behavior of a component or a system when it is tested.

System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.

5.1.1 System Overview:

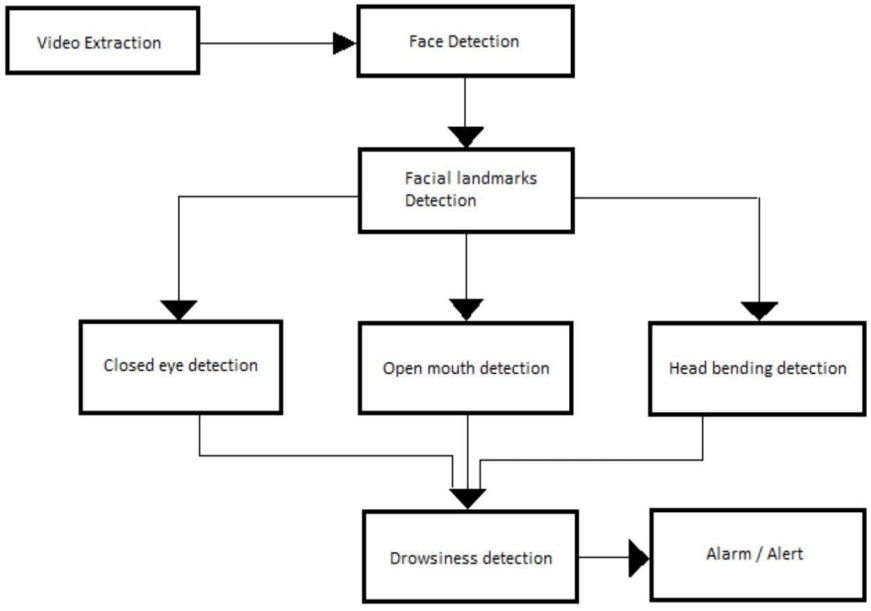


Fig 5.1.1.1 System Overview

5.1.2 Testing Process:

Testing is a disciplined process of finding and debugging defects to produce defect-free software. During testing, a test plan is prepared that specifies the name of the module to be tested, reference modules, date and time, location, name of the tester, testing tools, etc. Software engineers design test cases while writing the source codes, Testers may also involve in test case design.

Test cases include the input, output, and the conditions. Software tester runs the program using test cases according to the test plan and observes the test results. We can validate the test rest cases also.

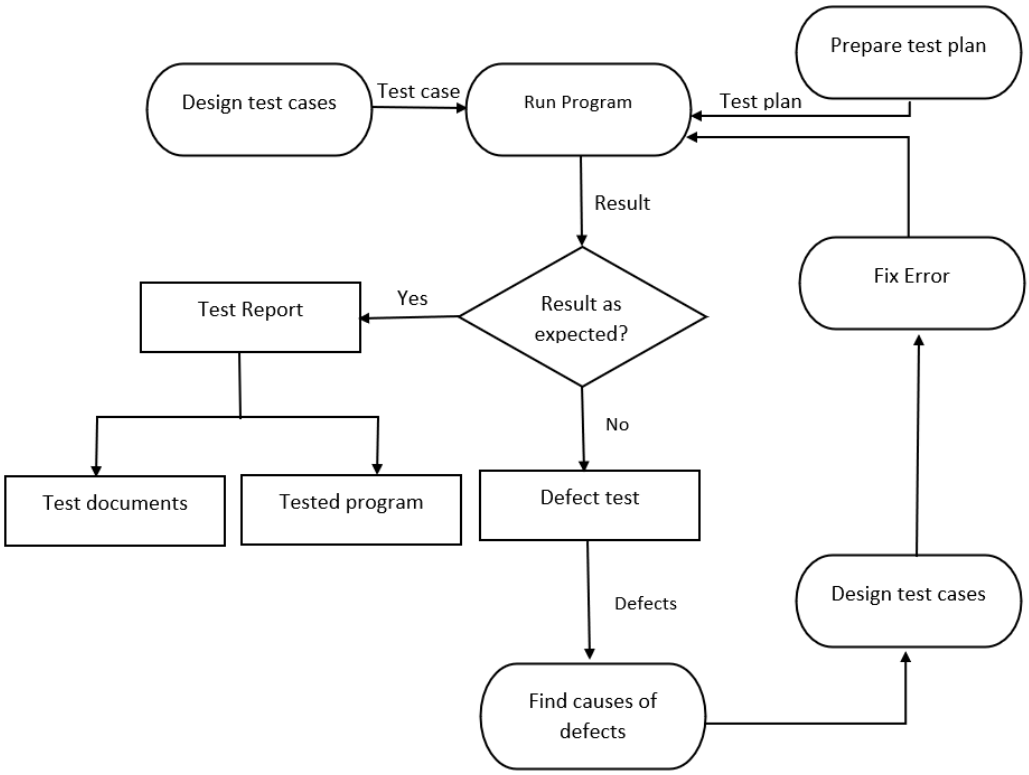


Fig 5.1.2.1 -Testing Process

Unit Testing:

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

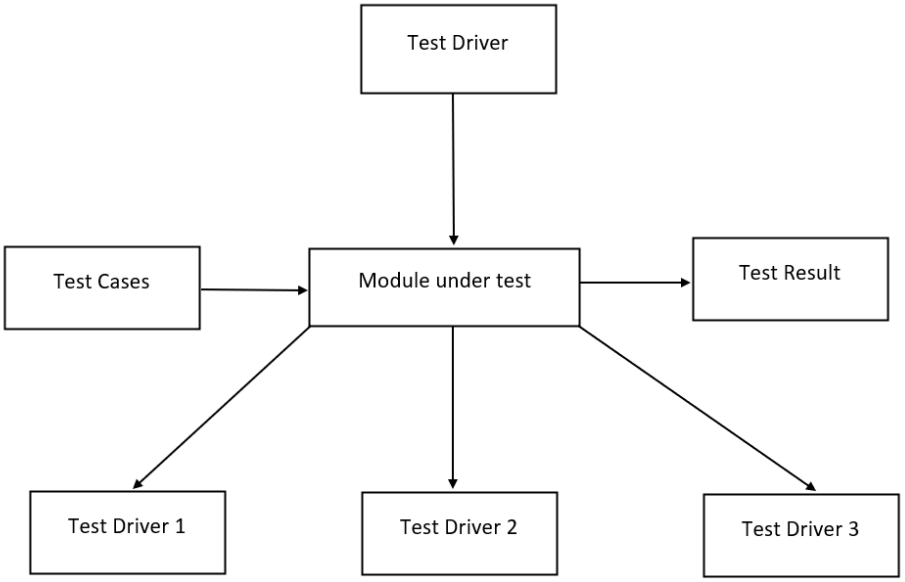


Fig 5.1.2.2 – Unit Testing

Integration Testing:

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process isto take unit tested modules and builds a program structure that has been dictatedby design.

User Acceptance Testing:

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

Output Testing:

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways - one is on screen and another in printed format.

5.2 Test Plan:

5.2.1 Features to be Tested:

1.Access of the Camera:

The test needs to be performed whether user camera is accessible or not. If not, the OpenCV module need to be checked and corresponding snippet of the code need to unit tested to test whether issue resolves.

2. Face Recognition:

When testing the system based on the user mimics the system must recognize the images of the user accessing the system.

3.Model Usage:

The pre-trained model needs to be accessed by the system to accept the new inputs and to predict the emotions of the user accessing the system.

5.2.2 Features not to be tested:

1.Algorithm Performance:

The used CNN algorithm performance need not be tested as the trained data has got the accuracy more than 90%.

5.2.3 Testing tools and Environment:

The system is tested in the usual python integrated development learning environment. The IDLE is used to unit test the code and the whole system.

5.3 Test Cases:

Test Case 1:

Test Case id	Test Scenario	Test Case	Pre-Condition	Test Steps	Expected Result	Post Conditions	Actual Result	Test Status (P/F)
1	Capture the video of driver	Import the frames of video	Run Application	Select a frame, Find face and detect landmarks	Eyes closed	Driver closed eyes in a frame	Displayed Eyes closed	P

Table 5.3.1 – Test Case 1

Test Case 2:

Test Case id	Test Scenario	Test Case	Pre Condition	Test Steps	Expected Result	Post Condition	Actual Result	Test Status (P/F)
2	Capture the video of driver	Import the frames of video	Run Application	Select a frame, Find face and detect landmarks	Yawning	Driver opened mouth in a frame	Displayed Yawning	P

Table 5.3.2 – Test Case 2

Test Case 3:

Test Caseid	Test Scenario	Test Case	Pre Condition	Test Steps	Expected Result	Post Condition	Actual Result	Test Status (P/F)
1	Capture the video of driver	Import the frames of video	Run Application	Select a frame, Find face and Detect landmarks	Head bending	Driver bent head in a frame	Displayed Head bending	P

Table 5.3.3 – Test Case 3

Test Case 4:

Test Case id	Test Scenario	Test Case	Pre Condition	Test Steps	Expected Result	Post Condition	Actual Result	Test Status (P/F)
1	Capture the video of driver	Import the frames of video	Run Application	Select a frame, Find face and detect landmarks	Drowsiness Alert and ring alarm	Driver closed eyes or is yawning or bent head for more than 10 previous consecutive frames	Displayed Drowsiness Alert and alarm is sent	P

Table 5.3.4 – Test Case 4

CHAPTER VI - CONCLUSION:

6.1 CONCLUSION:

In this project, a low-cost, real-time driver drowsiness monitoring system has been developed based on visual behavior and machine learning. The system uses the Haar based cascade classifier and HOG-SVM combination object detector for face detection. Here, visual behavior 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. The presented system performs well under the normal lighting conditions and normal resolutions. The method is non-intrusive and hence user friendly. It doesn't need any special hardware other than a normal web camera. This makes the system suitable to be implemented in desktop computers, mobile devices and so on. This method can be used in wide variety of applications like driver alertness measurement, liveliness detection, concentration measurement,measure of attentiveness etc.

6.2 FUTURE ENHANCEMENTS:

The future works may focus on the utilization of outer factors such as vehicle states, sleeping hours, weather conditions, mechanical data, etc, for fatigue measurement. Driver drowsiness pose a major threat to highway safety, and the problem is particularly severe for commercial motor vehicle operators. Twenty-four-hour operations, high annual mileage, exposure to challenging environmental conditions, and demanding work schedules all contribute to this serious safety issue. Monitoring the driver's state of drowsiness and vigilance and providing feedback on their condition so that they can take appropriate action is one crucial step in a series of preventive measures necessary to address this problem. Currently there is not adjustment in zoom or direction of the camera during operation. Future work may be to automatically zoom in on the eyes once they are localized.

CHAPTER VII APPENDICES:

7.1 Sample Code Snippets:

DrowsinessDetector.py:

```
from tkinter import *
from Ear_Aspect_Ratio import EAR
import tkinter
from scipy.spatial import distance as dist
from imutils import face_utils
from imutils.video import VideoStream
import imutils
import numpy as np
import argparse
import time
import dlib
import cv2
import playsound as play
from Mouth_Opening_Ratio import MOR
from Nose_Length_Ratio import NLR
from Send_Sms import sendSms

main = tkinter.Tk()
main.title("Driver Drowsiness Monitoring")
main.geometry("500x400")


def startMonitoring():
    NOSE_AVERAGE = 1
    EYE_AR_THRESHOLD =0
    pathlabel.config(text="Webcam Connected Successfully")
    webcamera = cv2.VideoCapture(0)
    #vs = VideoStream(src=args["webcam"]).start()
    ap = argparse.ArgumentParser()
    ap.add_argument("-w", "--webcam", type=int, default=0,help="index of webcam
on system")
    args = vars(ap.parse_args())
    vs = VideoStream(src=args["webcam"]).start()
    svm_predictor_path =
'C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/SVMclassifier.dat'
    svm_detector = dlib.get_frontal_face_detector()
    svm_predictor = dlib.shape_predictor(svm_predictor_path)
    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    (sNos,eNos) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
    print("SetUp loading...")
```

```
print("Please keep ur HEAD STRAIGHT toward's camera")
time.sleep(5)
print("Taking values please wait...")
for i in range(75):
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = svm_detector(gray, 0)
    for rect in rects:
        point1 = 0.0
        ear_sum=0.0
        shape = svm_predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        nose = shape[sNos:eNos]
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]
        leftEAR = EAR(leftEye)
        rightEAR = EAR(rightEye)
        ear = (leftEAR + rightEAR) / 2.0
        ear_sum+=ear
        EYE_AR_THRESHOLD=ear_sum
        point = dist.euclidean(nose[3], nose[0])
        point1 += point
        NOSE_AVERAGE = point1
        #print("Avrg: ", NOSE_AVERAGE)
NOSE_AVERAGE= NOSE_AVERAGE/75
print("EYE_AVERAGE: ",EYE_AR_THRESHOLD)
EYE_AR_CONSEC_FRAMES = 10
EYE_AR_THRESH = 0.25
NOSE_LENGTH_UP = 0.7
NOSE_LENGTH_DOWN = 1.1
MOU_AR_THRESH = 0.75
COUNTER1 = 0
COUNTER = 0
yawnStatus = False
yawns = 0
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]

while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    prev_yawn_status = yawnStatus
    rects = svm_detector(gray, 0)
    for rect in rects:
        shape = svm_predictor(gray, rect)
        shape = face_utils.shape_to_np(shape)
        leftEye = shape[lStart:lEnd]
```

```
        rightEye = shape[rStart:rEnd]
        mouth = shape[mStart:mEnd]
        nose = shape[sNos:eNos]
        leftEAR = EAR(leftEye)
        rightEAR = EAR(rightEye)
        mouEAR = MOR(mouth)
        noseNLR = NLR(nose,NOSE_AVERAGE)
        ear = (leftEAR + rightEAR) / 2.0
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        mouthHull = cv2.convexHull(mouth)
        noseHull = cv2.convexHull(nose)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 255), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 255), 1)
        cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [noseHull], -1, (0, 255, 0), 1)
        if noseNLR>NOSE_LENGTH_DOWN or noseNLR<NOSE_LENGTH_UP:
            #print("Head Bending")
            COUNTER1 = COUNTER1+1
            cv2.putText(frame, "NLR: {:.2f}".format(noseNLR),
(480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if COUNTER1>EYE_AR_CONSEC_FRAMES:

play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
                sendSms()
                cv2.putText(frame, "DROWSINESS ALERT!", (10,
130),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                cv2.putText(frame, "Head Bending!", (10,
50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                #cv2.putText(frame, "NLR: {:.2f}".format(noseNLR),
(480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            else:
                cv2.putText(frame, "Head Bending!", (10,
50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        else:
            COUNTER1 = 0
            cv2.putText(frame, "NLR: {:.2f}".format(noseNLR),
(480,90),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        if ear < EYE_AR_THRESH:
            COUNTER += 1
            cv2.putText(frame, "Eyes Closed ", (10,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if COUNTER >= EYE_AR_CONSEC_FRAMES:
                cv2.putText(frame, "DROWSINESS ALERT!", (10,
50),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
                sendSms()
```

```
            else:
                COUNTER = 0
                #print("EYE_AVERAGE: ",EYE_AR_THRESHOLD)
                cv2.putText(frame, "Eyes Open ", (10,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (480,
30),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            if mouEAR > MOU_AR_THRESH:
                cv2.putText(frame, "Yawning, DROWSINESS ALERT! ", (10,
70),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

play.playsound("C:/Users/Pavan/Desktop/Project/Files/DriverDrowsiness/MP3.wav")
                sendSms()
                yawnStatus = True
                output_text = "Yawn Count: " + str(yawns + 1)
                cv2.putText(frame, output_text,
(10,100),cv2.FONT_HERSHEY_SIMPLEX, 0.7,(255,0,0),2)
            else:
                yawnStatus = False
            if prev_yawn_status == True and yawnStatus == False:
                yawns+=1
            cv2.putText(frame, "MOR: {:.2f}".format(mouEAR), (480,
60),cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            #cv2.putText(frame,"Visual Behaviour & Machine Learning Drowsiness
Detection @
Drowsiness",(370,470),cv2.FONT_HERSHEY_COMPLEX,0.6,(153,51,102),1)
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF
        if key == ord("q"):
            break
    cv2.destroyAllWindows()
    vs.stop()
    webcamera.release()



font = ('times', 16, 'bold')
title = Label(main, text='Driver Drowsiness Monitoring System using
Visual\nBehaviour and Machine Learning',anchor=W, justify=LEFT)
title.config(bg='black', fg='white')
title.config(font=font)
title.config(height=3, width=120)
title.place(x=0,y=5)

font1 = ('times', 14, 'bold')
upload = Button(main, text="Start Behaviour Monitoring Using Webcam",
command=startMonitoring)
upload.place(x=50,y=200)
```

```
upload.config(font=font1)

pathlabel = Label(main)
pathlabel.config(bg='DarkOrange1', fg='white')
pathlabel.config(font=font1)
pathlabel.place(x=50,y=250)

main.config(bg='chocolate1')
main.mainloop()
```

Send_Sms.py:

```
import requests
def sendSms():
   url = "https://www.fast2sms.com/dev/bulkV2"
   payload = "sender_id=FSTSMS&message=Driver is drowsy please
alert!!!&language=english&route=p&numbers=7013017884"
   headers = {
   'authorization':
"L3V98nqRvjweK74TbWdNotkc2uhCDHzSYFrXUlZM5iamAIyP0fV12z4MaHOA
ouDpg8xSLkPKBqcvZhiR",
   'Content-Type': "application/x-www-form-urlencoded",
   'Cache-Control': "no-cache",
   }
   response = requests.request("POST", url, data=payload, headers=headers)
   print(response.text)
```

Nose_Length_Ratio.py:

```
from scipy.spatial import distance as dist
def NLR(drivernose,NOSE_AVERAGE):
   #NOSE_AVERAGE =1.2
   point = dist.euclidean(drivernose[3], drivernose[0])
   nose_np = point / NOSE
_AVERAGE
   return nose_np/75
```

Mouth_Opening_Ratio.py:

```
from scipy.spatial import distance as dist
def MOR(drivermouth):
    # compute the euclidean distances between the horizontal
    point   = dist.euclidean(drivermouth[0], drivermouth[6])
    # compute the euclidean distances between the vertical
    point1  = dist.euclidean(drivermouth[2], drivermouth[10])
    point2  = dist.euclidean(drivermouth[4], drivermouth[8])
    # taking average
    Ypoint   = (point1+point2)/2.0
    # compute mouth aspect ratio
    mouth_aspect_ratio = Ypoint/point
    return mouth_aspect_ratio
```

Ear_Aspect_Ratio.py:

```
from scipy.spatial import distance as dist
def EAR(drivereye):
    point1 = dist.euclidean(drivereye[1], drivereye[5])
    point2 = dist.euclidean(drivereye[2], drivereye[4])
    # compute the euclidean distance between the horizontal
    distance = dist.euclidean(drivereye[0], drivereye[3])
    # compute the eye aspect ratio
    ear_aspect_ratio = (point1 + point2) / (2.0 * distance)
    return ear_aspect_ratio
```

Certificate of Paper Publication:

RAPOLU PAVAN KUMAR – 18BQ1A12D2

THOTA CHARAN – 18BQ1A12G0

CERTIFICATE
OF PUBLICATION

IJIRCCE

International Journal of Innovative Research in Computer and Communication Engineering

Website: www.ijircce.com Email: ijircce@gmail.com

This is hereby Awarding this Certificate to

THOTA CHARAN

UG Student, IV Year Dept. of I.T., Vasireddy Venkatadri Institute of Technology, Guntur, India

Published a paper entitled

Driver Drowsiness Monitoring System using Visual Behavior and Machine Learning

in IJIRCCE, Volume 10, Issue 5, May 2022

Impact Factor 8.165

e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER

INNO SPACE SJIF Scientific Journal Impact Factor

Editor-in-Chief

SATTENAPALLI SRINIVASARAO – 18BQ1A12D9

CERTIFICATE
OF PUBLICATION

IJIRCCE

International Journal of Innovative Research in Computer and Communication Engineering

Website: www.ijircce.com Email: ijircce@gmail.com

This is hereby Awarding this Certificate to

SATTENAPALLI SRINIVASARAO

UG Student, IV Year Dept. of I.T., Vasireddy Venkatadri Institute of Technology, Guntur, India

Published a paper entitled

Driver Drowsiness Monitoring System using Visual Behavior and Machine Learning

in IJIRCCE, Volume 10, Issue 5, May 2022

Impact Factor 8.165

e-ISSN: 2320-9801
p-ISSN: 2320-9798

ISSN INTERNATIONAL STANDARD SERIAL NUMBER

INNO SPACE SJIF Scientific Journal Impact Factor

P. Kumar
Editor-in-Chief

7.2 Bibliography:

[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. papanikolopoulos, "Monitoring Driver Fatigue using Facial Analysis Techniques", IEEE Conference on IntelligentTransportation 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,

[6] B.M. Abadi, A. R. Foroshani, A. Niknezhad, "The Steps ofProposed 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.

[7] 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.

[8] 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.

[9] 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.

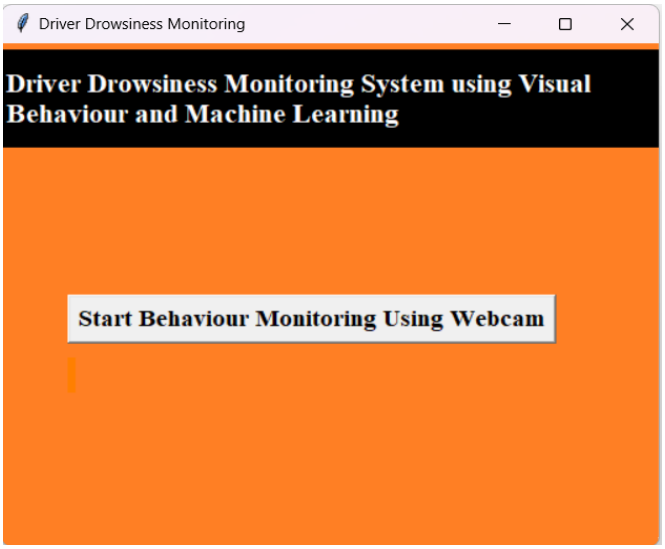
[10] K. T. Chui, K. F. Tsang, H. R. Chi, B. W. K. Ling, and C. K. Wu, "An accurate ECG based transportation safety drowsiness detection scheme," IEEE Transactions on Industrial Informatics, vol. 12, no. 4, pp. 1438– 1452, Aug. 2016.

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

[12] 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.

[13] Richard O. Duda, Peter E. Hart, David G. Stork, "Pattern Classification", Wiley student edition.

RESULT:



Output Sample 1:

Fig: Eyes Open

Output Sample 2:

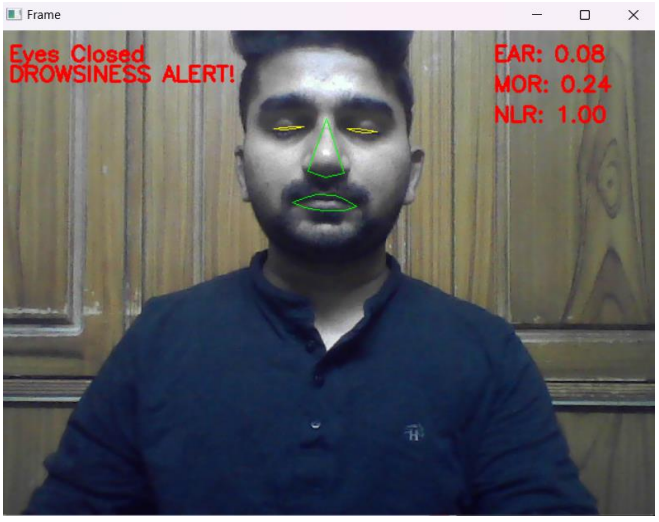


Fig: Eyes Closed

Output Sample 3:

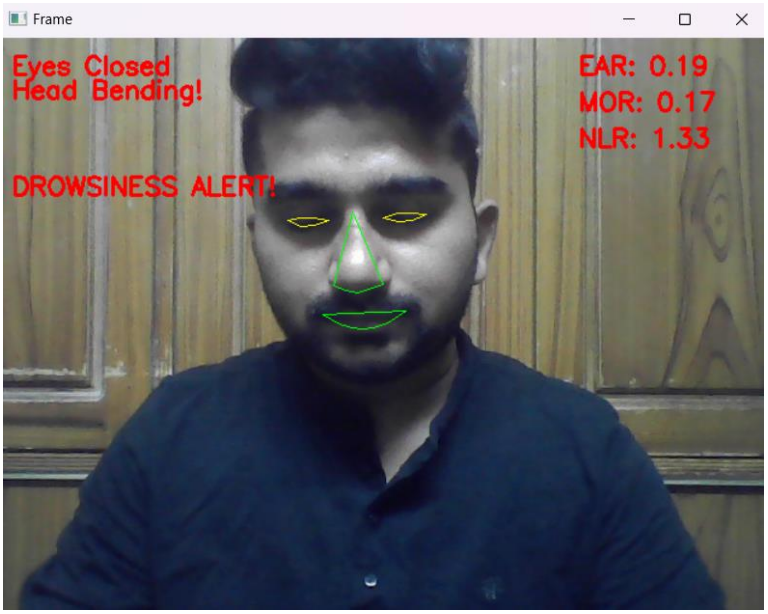


Fig: Head Bending

Output Sample 4:

Fig : Yawning