1. INTRODUCTION

1.1 INTRODUCTION:

Driver drowsiness driving is one of the main reasons for road accidents. In current survey it shows that out of 5 accidents one accident is due to drowsiness of the driver which is approximately 20% of road accidents and it increasing gradually in every year. The survey highlights the facts that total numbers of traffic deaths are excessive because of drowsiness of the driver. Driving a vehicle in a crowded road has become a nightmare because of the road conditions, poor weather conditions, haste to reach the destination and excess of traffic. Drowsiness of driver, drunk and drive are coming further major reasons for road accidents. Due to less conscious we can't take care of ours while driving. To provide security to driver, the vehicles are assisted with automated safety system that alerts driver by using alarm. All vehicles should be equipped with eye blink sensor and alcohol sensor sequentially to evade these types of accidents. The objective of the project is to by using IR sensor the eye blink is measured and controlled. The infrared rays are transmitted by IR transmitter into driver's eye. The eye reflects the transmitted infrared rays and these reflected rays are received by the IR receiver. If the eye is in closed status, the output of IR receiver is high. The IR receiver output is low, if the eye is in opening position. This informs that the eye is in opening or closing position. The alarm is indicated, if the output is given to logic circuit. This project is to decrease the accidents due to comatose through eye blink.

Driver drowsiness is a significant factor in the increasing number of accidents on today's roads and has been extensively accepted. This proof has been verified by many researchers that have demonstrated t ies between driver drowsiness and road accidents. Although it is hard to decide the exact number of accidents due to drowsiness, it is much likely to be underestimated.

2. SYSTEM ANALYSIS

2.1 ANALYSIS MODEL

The drowsiness alert system is used to detect the driver is in drowsy or not. It can be connected with any surveillance system installed at your premise. The authorities or admin can check the person through the system to confirm their identity. The accuracy rate of detecting a person with a face mask is 95-97% depending on the digital capabilities. The data has been transferred and stored automatically in the system to enable reports whenever you want. The study states that the reason for a mishap can be categorized as one of the accompanying primary classes: (1) human, (2) vehicular, and (3) surrounding factor. The driver's error represented 91% of the accidents. The other two classes of causative elements were referred to as 4% for the type of vehicle used and 5% for surrounding factors. Several measures are available for the measurement of drowsiness which includes the following:

1. Vehicle based measures.
2. Physiological measures.
3. Behavioral measures.

Vehicle Based Measures: Vehicle-based measures survey path position, which monitors the vehicle's position as it identifies with path markings, to determine driver weakness, and accumulate steering wheel movement information to characterize the fatigue from low level to high level. In many research project, researchers have used this method to detect fatigue, highlighting the continuous nature of this non-intrusive and cost-effective monitoring technique.

i. This is done sudden deviation of vehicle from lane position.
ii. Sudden movement of steering wheels.
iii. Pressure on acceleration paddles. For each measures threshold values are decided which when crossed indicated that driver is drowsy.

Advantages:

i. It is noninvasive in nature.

ii. Provides almost accurate result.

Disadvantages:

i. Vehicle based measures mostly affected by the geometry of road which sometimes unnecessarily activates the alarming system.

ii. The driving style of the current driver needs to be learned and modeled for the system to be efficient.

iii. The condition like micro sleeping which mostly happens in straight highways cannot be detected.

Physiological Measures: Physiological measures are the objective measures of the physical changes that occur in our body because of fatigue. These physiological changes can be simply measure by their respective instruments as follows:

i. ECG (electro cardiogram)

ii. EMG(electromyogram)

iii. EOG(electro occulogram)

iv. EEG(electroencephalogram)

- **Monitoring Heart Rate:** An ECG sensor can be installed in the steering wheel of a car to monitor a driver's pulse, which gives a sign of the driver's level of fatigue indirectly giving the state of drowsiness. Additionally the ECG sensor can be introduced in the back of the seat.
- **Monitoring Brain Waves:** Special caps embedded with electrodes measures the brain waves to identify fatigue in drivers and report results in real time. Then each brain waves can be classified accordingly to identify drowsiness.
- **Monitoring Muscle Fatigue:** As muscle fatigue is directly related to drowsiness. We know during fatigue the pressure on the steering wheel reduces and response of several

muscle drastically reduces hence it can be measured by installation of pressure sensors at steering wheel or by measuring the muscle response with applied stimuli to detect the fatigue.

- **Monitoring Eye Movements:** Invasive measurement of eye movement and eye closure can be done by using electro occulogram but it will be very uncomfortable for the driver to deal with.

Though this method gives the most accurate results regarding drowsiness. But it requires placement of several electrodes to be placed on head, chest and face which is not at all a convenient and annoying for a driver. Also they need to be very carefully placed on respective places for perfect result

Behavioral Measures: Certain behavioral changes take place during drowsing like

1. Yawning.
2. Amount of Eye Closure.
3. Eye Blinking.
4. Head Position.

2.2 PROPOSED SYSTEM:

Among all these four strategies, the most precise technique depends on human physiological measures. This procedure is executed in two ways: measuring changes in physiological signs, for example, brain waves, heart rate, and eye flickering; and measuring physical changes, for example, sagging posture, inclining of the driver's head and the open/shut conditions of the eyes. In spite of the fact that this procedure is most precise, it is not reasonable, since detecting electrodes would need to be put straightforward onto the driver's body, and thus be irritating and diverting to the driver. Also, long time driving would bring about sweat on the sensors, reducing their capacity to screen precisely. Hence this approach will be mostly focusing on amount of eye closure also called (PERCLOS) percentage of closure as it

provides the most accurate information on drowsiness. It is also non-intrusive in nature, hence does not affect the state of the driver and also the driver feels totally comfortable with this system. An environmental factor like road condition does not affect this system. The case of micro nap is also detected according the given threshold value. The development of this system includes face identification and tracking, detection and location of the human eye, human eye tracking, eye state detection, and driver fatigue testing. The key parts of the detection framework fused the detection and location of human eyes and driver fatigue testing. The improved technique for measuring the PERCLOS estimation of the driver was to compute the proportion of the eyes being open and shut with the aggregate number of frames for a given period This project involves controlling vehicle accidents and saving driver's life by alerting him. Whenever the driver's eye blinking period is more, we consider this condition under drowsiness. Immediate action will be taken and buzzer will blow up alerting the driver.

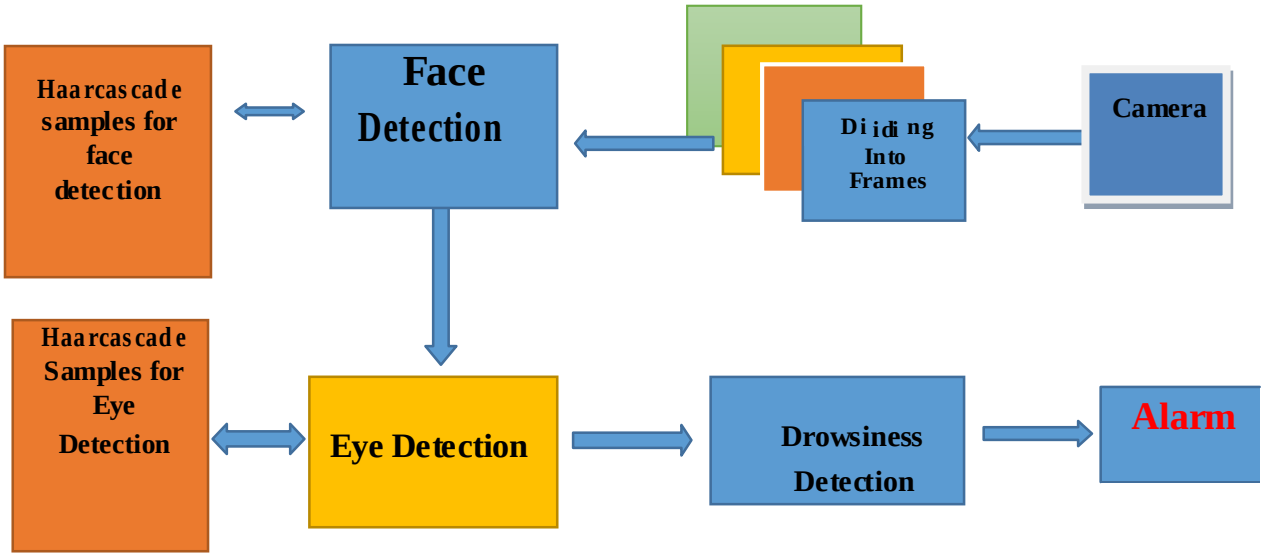

Fig 2.2.1: Flow chart showing entire process of drowsiness detection system

2.3 EXISTING SYSTEM:

The Existing System of driver drowsiness detection has following disadvantages:

i. Using of two cameras in the system one for monitoring the head movement and other for facial expressions.

ii. Aging of sensors and all these sensors are attached to the driver's body which may affect the driver.

The Hyundai Mobis system has been the most actively studied as a system for detecting and preventing drowsiness while driving domestic cars. This system prevents drowsy driving by recognizing the driver's face and detecting breathing conditions. As a function of the system, the beep sounds when the driver's eyes do not stare to the front. It is designed to prevent driving during drowsiness, and an alarm is activated by checking the blink time and speed (cycle) of the eyelid. However, it is still in the design phase and has not been developed. Figure 2.3.1 is a sleepiness prevention system under development for use by Hyundai Mobis. Surveillance cameras detect the driver's eyes, nose, and mouth and continuously provide statistical data.

Fig 2.3.1: Mobis by Hyundai.

Other domestic car manufacturers have developed sensors to detect gaps between cars. When the distance between the sensor and the vehicle in front is narrow, an alarm is provided to the driver and the brake is forced to operate. However, there is no drowsiness detection and prevention system in Korea that supports all of these factors. In Figure 2.3.2, the front sensor detects the lane of the vehicle

and sounds an alarm to warn the driver when the vehicle leaves the lane. This capability is currently being added to all vehicles.

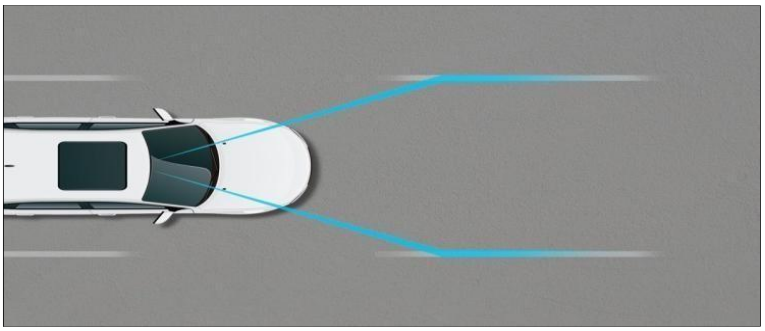

Fig 2.3.2: Camera sensors on the car

Figure 2.3.3 measures the tilt of the driver's face with a tilt and vibration sensor on the ear flap and generates a vibration when a certain tilt value is detected. Currently, no product has applied this Sustainability function. However, it could be applied to prevent drowsy driving in the future.

Fig 2.3.3: Inclination detection vibration sensor.

3. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- TECHNICAL FEASIBILITY
- OPERATIONAL FEASIBILITY
- ECONOMICAL FEASIBILITY

3.1 TECHINICAL FEASIBILITY

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc. A number of issues have to be considered while doing a technical analysis.

Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the Technologies that are to be required for the development of the new system.

Find out whether the organization currently possesses the required technologies

Is the required technology available with the organization?

If so is the capacity sufficient?

For instance –"Will the current printer be able to handle the new reports and forms required for the system?"

3.2 OPERATIONAL FEASIBILITY

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

- Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.
- Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.
- Have the user been involved in the planning and development of the project? Since the proposed system was to help reduce the hardships encountered

In the existing manual system, the new system was considered to be operational feasible.

3.3 ECONOMICAL FEASIBILITY

Economic feasibility attempts 2 weigh the costs of developing and implementing a new system, against the benefits that would accrue from having the new system in place. This feasibility study gives the top management the economic justification for the new system.

4. SOFTWARE REQUIREMENT SPECIFICATIONS

It is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based the agreement between customer and contractors. It may include the use cases of how user is going to interact with software system. The software requirement specification document consistent of all necessary requirements required for project development. To develop the software system we should have clear understanding of Software system

4.1 FUNCTIONAL REQUIREMENTS:

This requirement describes how a product must behave, what its features and functions. This requirements are product features or functions that developers must implement to enable users to accomplish their tasks .This requirements specifies criteria of behavior of system.

- Face scan by integrated camera
- Detection of eye bye OpenCV

4.2 PERFORMANCE:

Performance requirements define how well the software system accomplishes certain functions under specific conditions. Examples include the software's speed of response, throughput, execution time and storage capacity. The service levels comprising performance requirements are often based on supporting end-user tasks.

4.3 HARDWARE REQUIREMENTS:-

Computer hardware is the physical components that a computer system requires to function. It encompasses everything with a circuit board that operates within a PC or

laptop, including the motherboard, graphics card, CPU (Central Processing Unit), ventilation fans, webcam, power supply, and so on. To accomplish their tasks .This requirements specifies criteria of behavior of system.

- Pentium-IV (Processor).
- 512 MB Ram
- Embedded System with Webcam
- Hard disk 50 GB

4.4 SOFTWARE REQUIREMENTS: -

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. Opposite of hardware, which describes the physical aspects of a computer, software is a generic term used to refer to applications, scripts and programs that run on a device. Software can be thought of as the variable part of a computer, and hardware the invariable part.

- Operating System : Windows , Unix, MacOs
- Python
- Libraries of Python
  - OpenCV
  - Imutils
  - Dlib
  - Numpy
  - Scipy
  - Playsound

5. PROJECT DESIGN APPROACH

5.1 INTRODUCTION

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.

Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.

5.2 SYSTEM WORKFLOW

In this project, we'll discuss our driver drowsiness detector, detailing how our computer vision will be implemented. From there, we'll be using pre-trained shape predictor landmarks dat file to detect 68 face land marks. We'll then show you how to implement a python script to detect whether an eye is drowsy or not. We'll use pre-trained shape predictor dat file then calculate EAR ratio and view results. Then, proceed to implement drowsiness detector in real-time video streams.

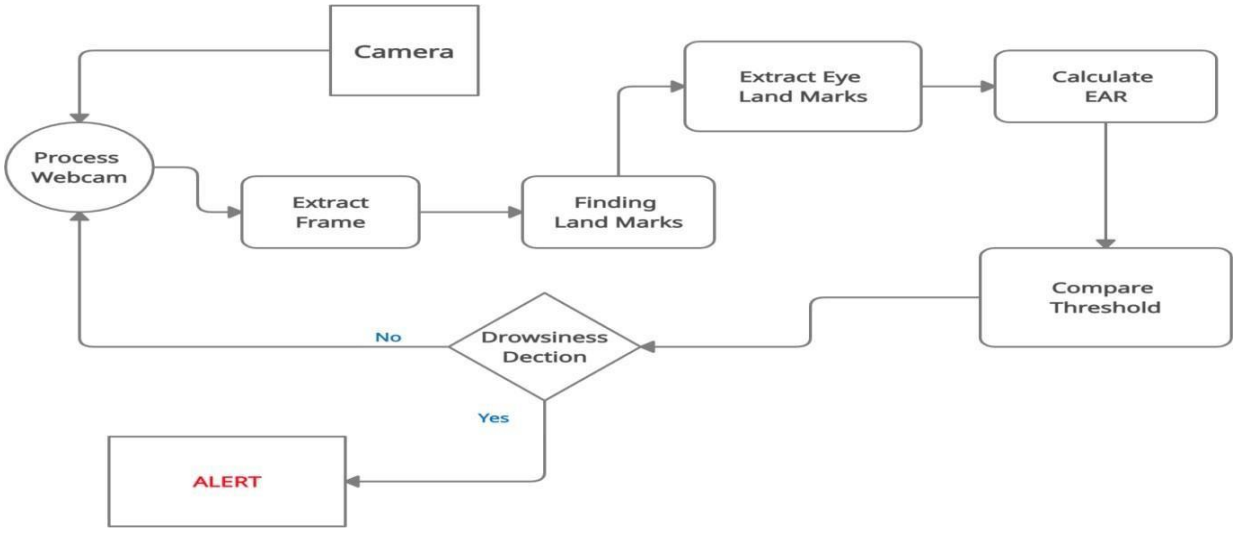

Fig 5.2.1: System workflow

5.3 E-R DIAGRAM

An Entity–relationship model (ER model) describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

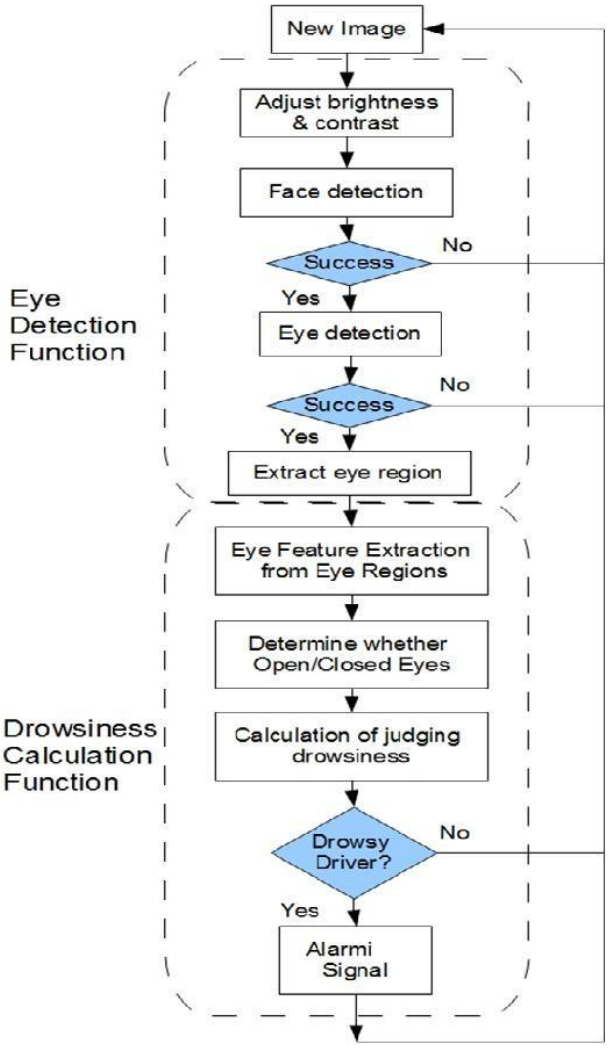

Fig 5.3.1: E-R Diagram

5.4 WORKFLOW

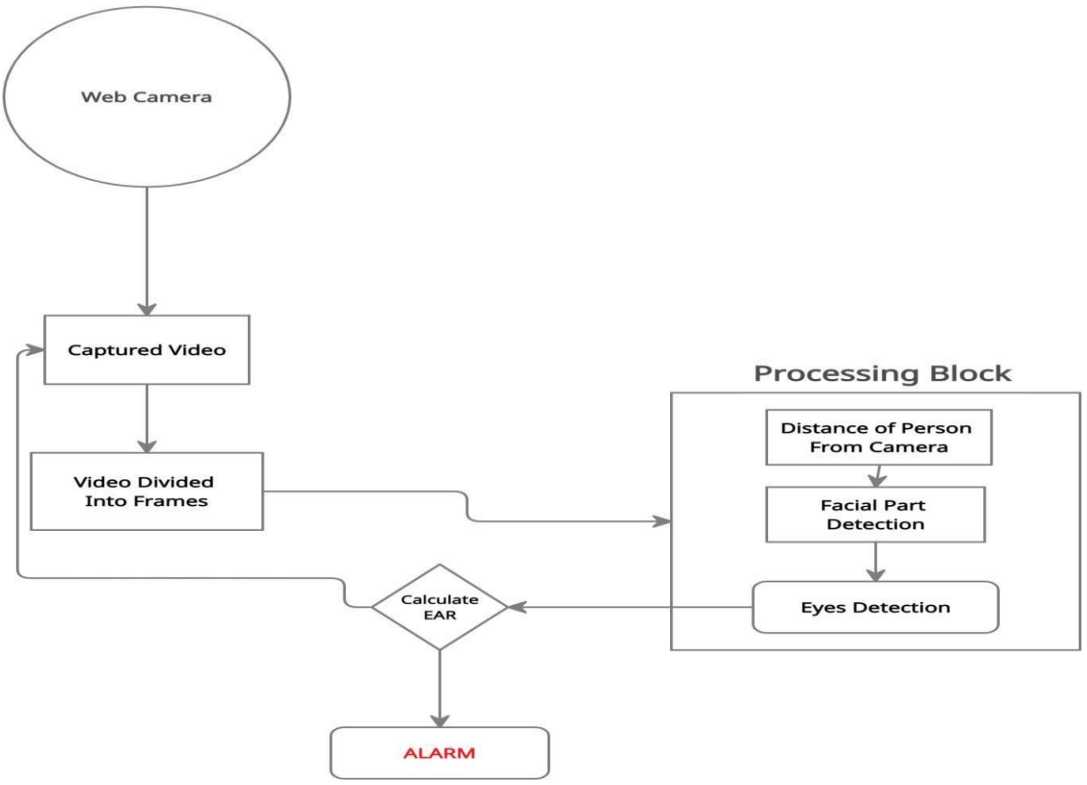

Fig 5.4.1: Workflow Diagram

5.5 DATA FLOW ANALYSIS

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

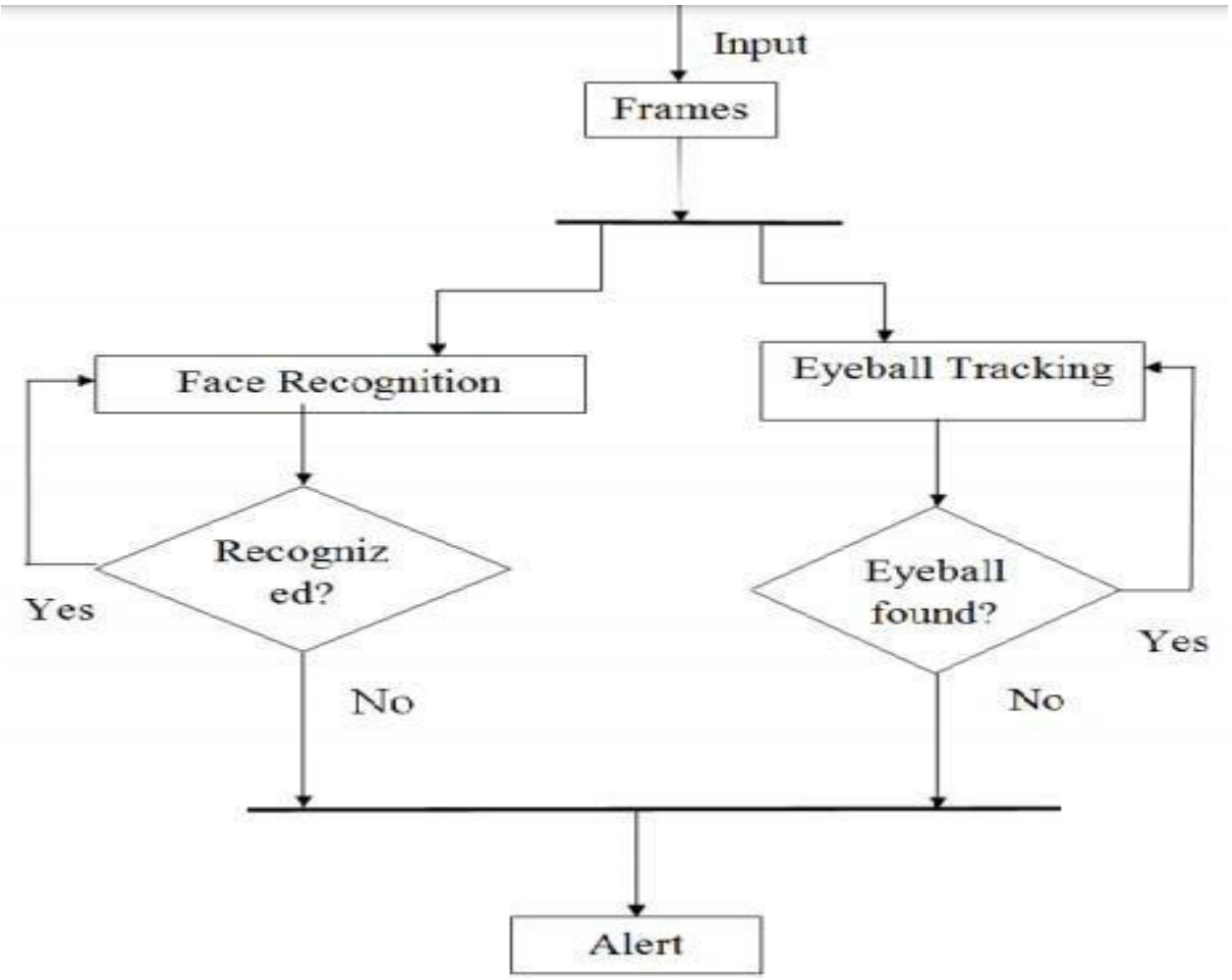

Fig 5.5.1: Data Flow Diagram

5.6 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business

modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

5.7 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. 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.

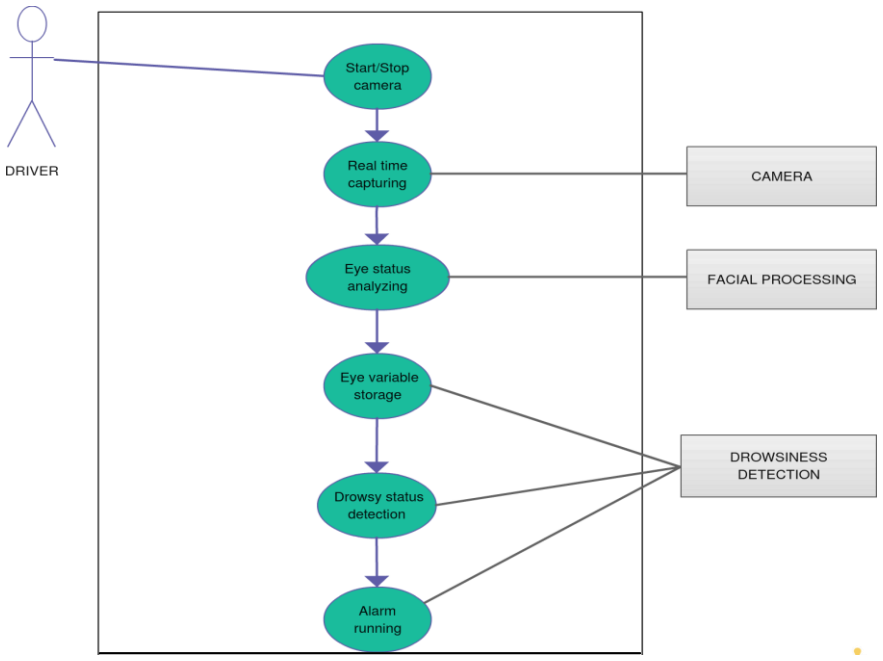

Fig 5.7.1: Usecase Diagram

5.8 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

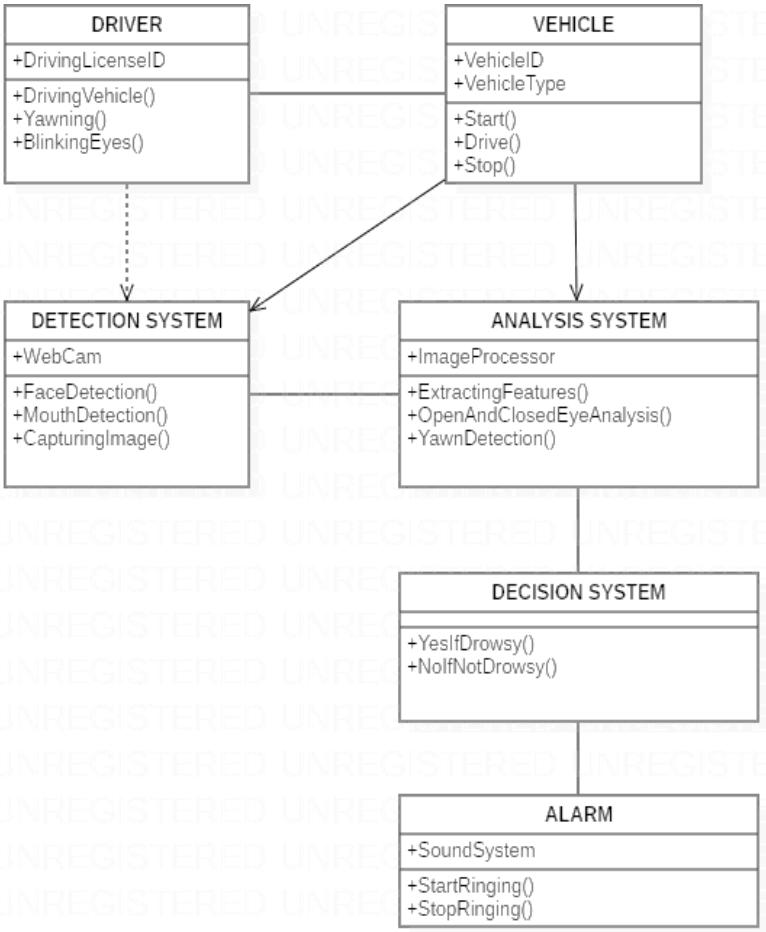

Fig 5.8.1: Class Diagram

5.9 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling 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. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

5.10 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling 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.

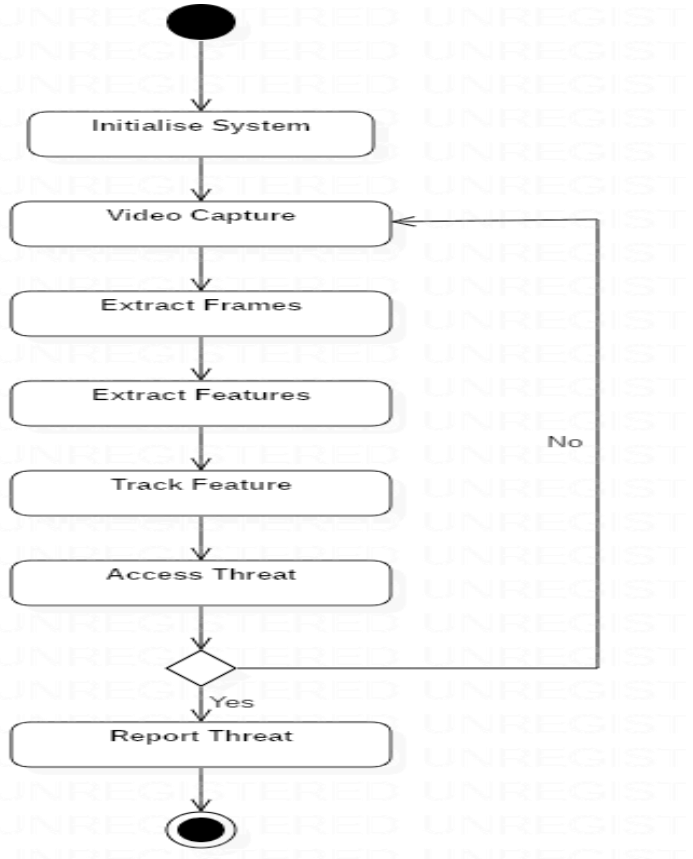

Fig 5.10.1: Activity Diagram

6. TECHNOLOGY DESCRIPTION

6.1 PYTHON PROGRAMMING LANGUAGE

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Below are some facts about Python Programming Language:

1. Python is currently the most widely used multi-purpose, high-level programming language.
2. Python allows programming in Object-Oriented and Procedural paradigms.
3. Python programs generally are smaller than other programming languages like Java. Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
4. Python language is being used by almost all tech-giant companies like – Google, Amazon, Face book, Instagram, Drop box, Uber… etc.
5. The biggest strength of Python is huge collection of standard library which can be used for the following:
    - Machine Learning
    - GUI Applications (like Kivy, Tkinter, PyQt etc)
    - Web frameworks like Django (used by YouTube, Instagram, Dropbox)
    - Image processing (like OpenCV, Pillow)
    - Web scraping (like Scrapy, BeautifulSoup, Selenium)
    - Test frameworks
    - Multimedia
    - Scientific computing and many more...

Advantages of Python

- Easy to Read, Learn and Write
- Improved Productivity
- Interpreted Language
- Dynamically Typed
- Free and Open-Source
- Vast Libraries Support
- Portability

Disadvantages of Python

- Slow Speed
- Not Memory Efficient
- Weak in Mobile Computing
- Database Access
- Runtime Errors

Python source files use the ".py" extension and are called "modules." With a Python module hello.py, the easiest way to run it is with the shell command "python hello.py Alice" which calls the Python interpreter to execute the code in hello.py, passing it the command line argument "Alice". See the official docs page on all the different options you have when running Python from the command-line.

Example

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

Output

This produces the following result −

```
Hello, Python!
```

6.2 OPEN CV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e., whatever operations one can do in Numpy can be combined with OpenCV. OpenCV is used for all sorts of image and video analysis, like facial recognition and detection, license plate reading, photo editing, advanced robotic vision, optical character recognition and a whole lot more.

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields −

- Image Processing − It focuses on image manipulation.
- Pattern Recognition − It explains various techniques to classify patterns.
- Photogrammetry − It is concerned with obtaining accurate measurements from images.

Computer Vision Vs Image Processing

Image processing deals with image-to-image transformation. The input and output of image processing are both images.

Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Applications of OpenCV

There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- Number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition
- OpenCV Functionality
- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)

CV2

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.cv2.imread () method loads an image from the specified file. If the image cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

ROI

A region of interest (often abbreviated ROI), are samples within a data set identified for a particular purpose. The concept of a ROI is commonly used in many application areas. For example, in medical imaging, the boundaries of a tumor may be defined on an image or in a volume, for the purpose of measuring its size.

NumPy

NumPy is a Python library and is written partially in Python, but most of the parts that require fast computation are written in C or C++. It used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. Num Py was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. Num Py stands for Numerical Python.

Why Use NumPy?

In Python we have lists that serve the purpose of arrays, but they are slow to process. Num Py aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in Num Py is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are made to be created as homogeneous arrays, considering the mathematical operations that can be performed on them. It would not be possible with heterogeneous data sets.

1. **NumPy uses much less memory to store data**

The NumPy arrays takes significantly less amount of memory as compared to python lists. It also provides a mechanism of specifying the data types of the contents, which allows further optimization of the code.

2. **Using NumPy for creating n-dimension arrays**

An n-dimension array is generally used for creating a matrix or tensors, again mainly for the mathematical calculation purpose. Compare to python list base n-dimension arrays, NumPy not only saves the memory usage, it provide a significant number of additional benefits which makes it easy to mathematical calculations.

3. **Mathematical operations on NumPy n-Dimension Arrays**

As stated earlier, NumPy is not only about efficient storing the data, it also makes it extremely easy to perform mathematical operations on it. Any actions on n-dimension arrays behaves exactly similar to mathematical operations.

4. **Finding Elements in NumPy array**

While working with data sets there will be times when we need to find specific data from the available data set. Though NumPy provides multiple functions for the same, three of them will be used more often than others. They are where, nonzero and count nonzero.

Where and nonzero functions returns the index associated with the True statement.

Dlib

DLib is an open source C++ library implementing a variety of machine learning algorithms, including classification, regression, clustering, data transformation, and structured prediction. And it is based on "Convolution Neural Network".

Understanding dlib's facial landmark detector

The pre-trained facial landmark detector inside the dlib library is used to estimate the location of 68 (x, y)-coordinates that map to facial structures on the face.

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

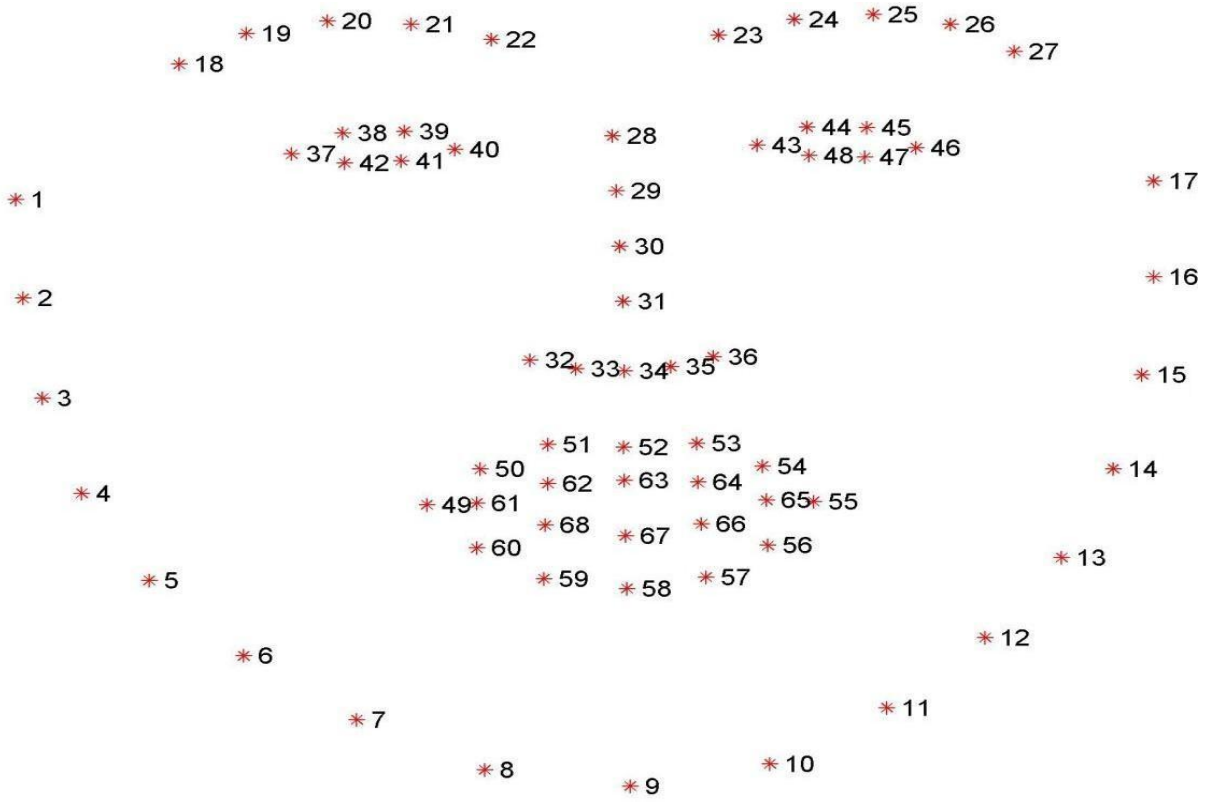

Fig 6.2.1: Visualizing the 68 facial landmark coordinates from the iBUG 300-W dataset

These annotations are part of the 68 point iBUG 300-W dataset which the dlib facial landmark predictor was trained on. It's important to note that other flavors of facial landmark detectors exist, including the 194 point model that can be trained on the HELEN dataset. Regardless of which dataset is used, the same dlib framework can be leveraged to train a shape predictor on the input training data.

Imutils

Imutils are a series of convenience functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, and displaying Matplotlib images easier with OpenCV and both Python 2.7 and Python 3.

Scipy

SciPy, pronounced as Sigh Pi, is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations. The SciPy library depends

on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. NumPy and SciPy are easy to use, but powerful enough to depend on by some of the world's leading scientists and engineers.

Eye Aspect Ratio

Each eye is represented by 6 (x, y)-coordinates, starting at the left-corner of the eye (as if you were looking at the person), and then working clockwise around the remainder of the region:

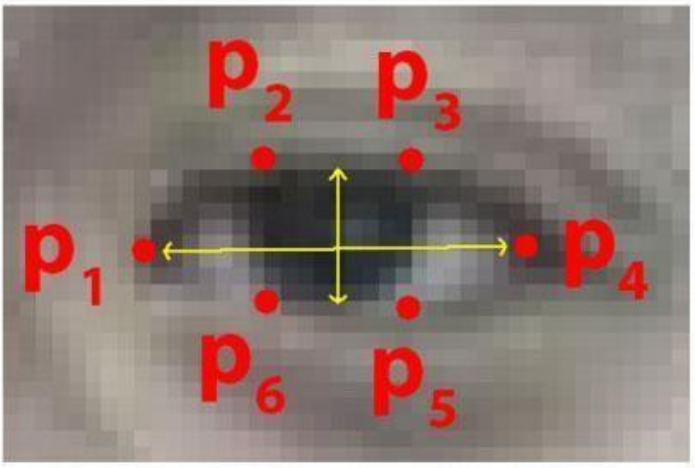

Fig 6.2.2: The 6 facial landmarks associated with the eye.

Based on this image, we should take away on key point. There is a relation between the width and the height of these coordinates.

Based on the work by Soukupová and Čech in their 2016 paper, Real-Time Eye Blink Detection using Facial Landmarks, we can then derive an equation that reflects this relation called the eye aspect ratio (EAR)

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

Fig 6.2.3: The eye aspect ratio equation.

Where p1, …, p6 are 2D facial landmark locations.

The numerator of this equation computes the distance between the vertical eye landmarks while the denominator computes the distance between horizontal eye landmarks, weighting the denominator appropriately since there is only one set of horizontal points but two sets of vertical points.

Using this simple equation, we can avoid image processing techniques and simply rely on the ratio of eye landmark distances to determine if a person is blinking.

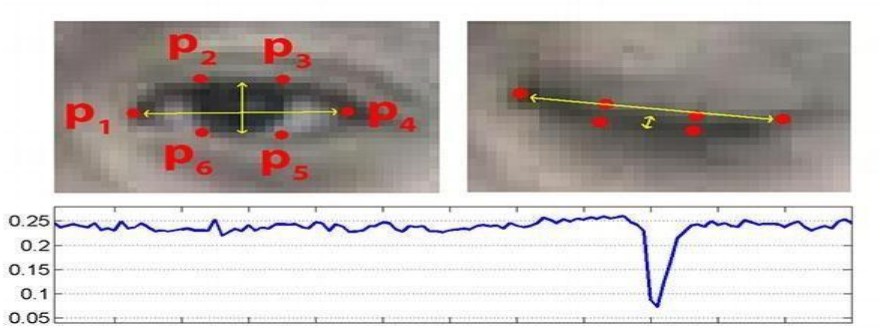

Fig 6.2.4: Top-left: A visualization of eye landmarks when then the eye is open. Top-right: Eye landmarks when the eye is closed. Bottom: Plotting the eye aspect ratio over time. The dip in the eye aspect ratio indicates a blink

On the top-left we have an eye that is fully open — the eye aspect ratio here would be large(r) and relatively constant over time. However, once the person blinks (top-right) the eye aspect ratio decreases dramatically, approaching zero. The bottom figure plots a graph of the eye aspect ratio over time for a video clip. As we can see, the eye aspect ratio is constant, then rapidly drops close to zero, then increases again, indicating a single blink has taken place.

Playsound

The playsound module is a cross platform module that can play audio files. This doesn't have any dependencies, simply install with pip in your virtualenv and run.

Argparse

The argparse module makes it easy to write user-friendly command-line interfaces. It parses the defined arguments from the sys.argv. The argparse module also automatically generates help and usage messages and issues errors when users give the program invalid arguments.

6.3 SAMPLE CODE

```
from scipy.spatial import distance as dist
from imutils.video import VideoStream
from imutils import face_utils
from threading import Thread
import numpy as np
import playsound
import argparse
import imutils
import time
import dlib
import cv2

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

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

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


# 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 for to set off  the
# alarm
EYE_AR_THRESH = 0.3
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(args["shape_predictor"])

# 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 = VideoStream(src=args["webcam"]).start()
time.sleep(1.0)

# 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 = 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)
    # 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
                                    if args["alarm"] != "":
                                            t = Thread(target=sound_alarm,
                                                    args=(args["alarm"],))
                                            t.deamon = True
                                            t.start()

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

```
                # otherwise, the eye aspect ratio is not below the blink
                # threshold, so reset the counter and alarm
                else:
                        COUNTER = 0
                        ALARM_ON = False
        # draw the computed eye aspect ratio on the frame to help
                # with debugging and setting the correct eye aspect ratio
                # thresholds and frame counters
                cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

        # show the frame
        cv2.imshow("Frame", frame)
        key = cv2.waitKey(1) & 0xFF

        # if the `q` key was pressed, break from the loop
        if key == ord("q"):
                break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

7. OUTPUT SCREENS

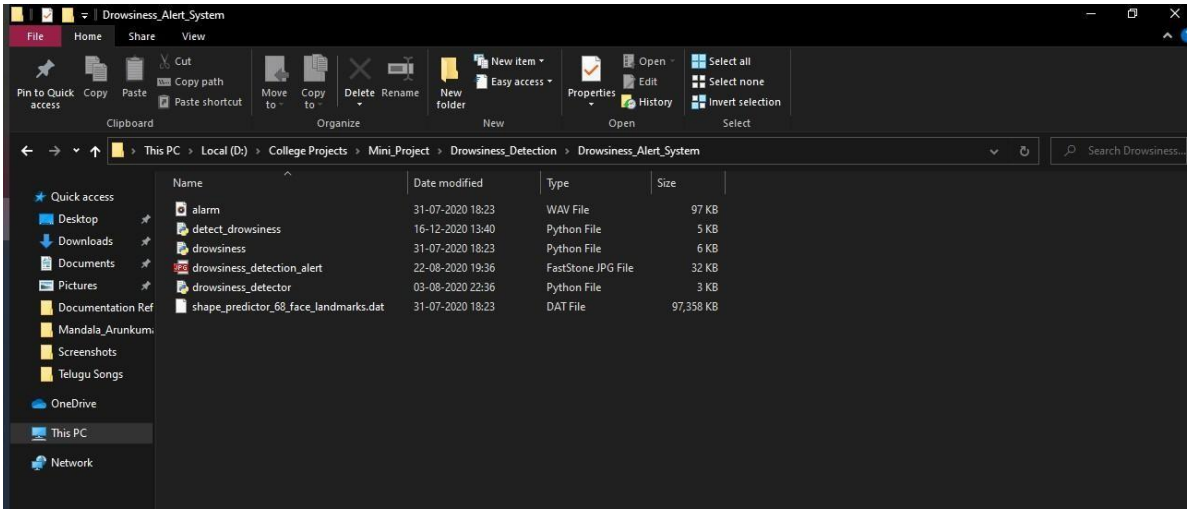

Fig 7.1: Drowsiness_Alert_System File

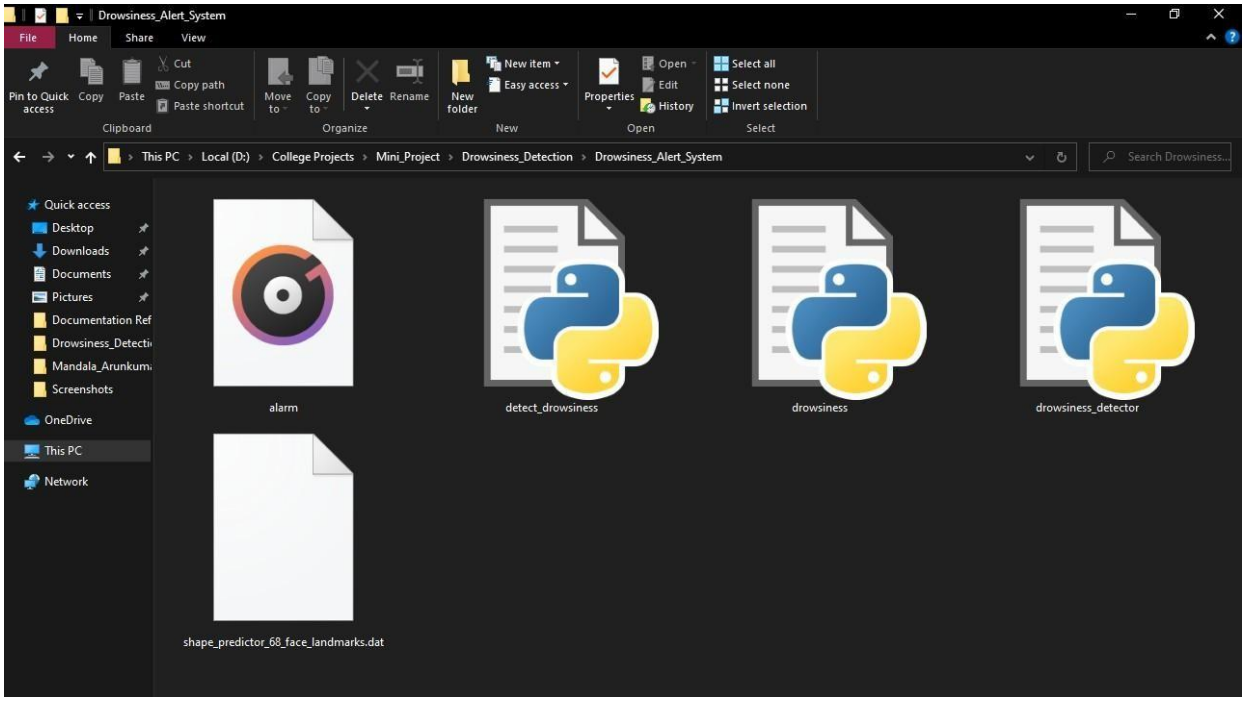

Fig7.2: Dlib Shape_predictor_68_face_landmarks.dat File

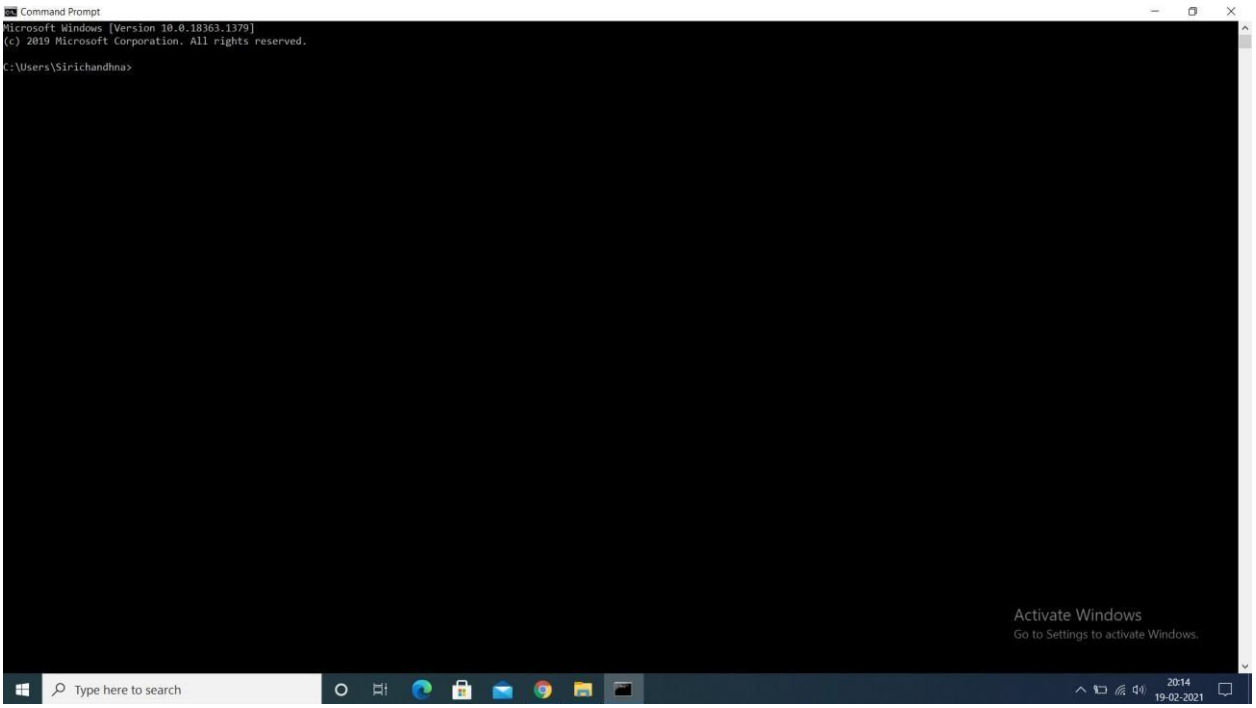

Fig7.3: Opening Command Prompt

Non-Drowsy Person

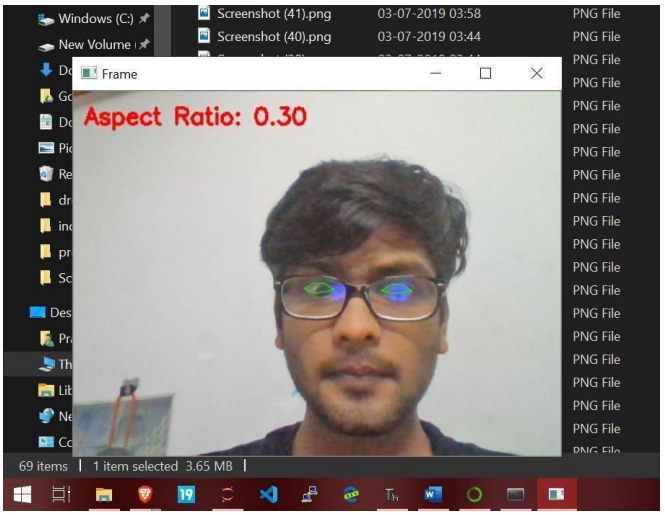

Fig 7.4: Non Drowsy State

Drowsy Person

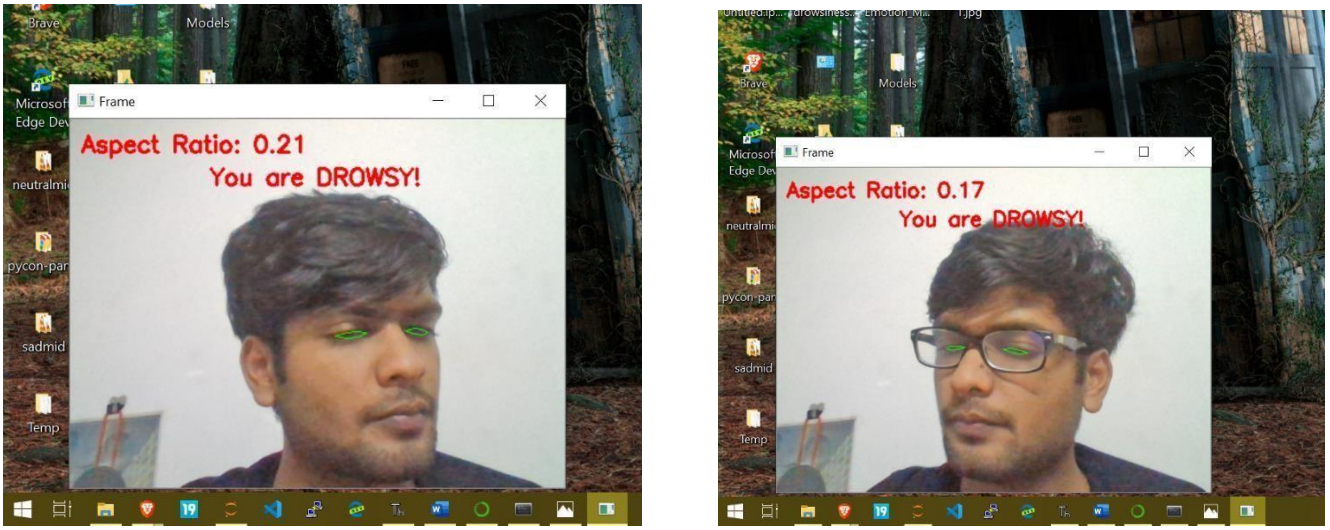

Fig 7.5: Different aspect ratio at drowsy state

8.TESTING

8.1 INTRODUCTION

Testing is a process of executing a program with the indent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding.

System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

A good test case is one that has a high probability of finding an as undiscovered error. A successful test is one that uncovers an as undiscovered error.

Testing Objectives :

- Testing is a process of executing a program with the intent of finding an error
- A good test case is one that has a probability of finding an as yet undiscovered error
- A successful test is one that uncovers an undiscovered error

Testing Principles:

- All tests should be traceable to end user requirements
- Tests should be planned long before testing begins
- Testing should begin on a small scale and progress towards testing in large
- Exhaustive testing is not possible
- To be most effective testing should be conducted by a independent third party

The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used. They are

8.2 Test Cases

INPUT	**Eyes Detection Accuracy**	**Drowsiness Accuracy**
Sample 1	100%	87.5 %
Sample 2	95%	100%
Sample 3	80%	62.5%
Sample 4	100%	87.5%
Sample 5	100%	100%
TOTAL	95%	87.5%

8.3 SOFTWARE REQUIREMENTS

Testing Methods:

There are two major type of testing. They are:

1) White box Testing
2) Black box Testing

White box Testing

White box sometimes called “Glass Box Testing” is a test case design uses the control structure of the procedural design test case.

Using white box testing methods, the following tests were made on the system.

All independent paths within a module have been exercised once. In our system, ensuring that case was selected and executed checked all case structures. The bugs that were prevailing in some part of the code where fixed.

a) All logical decisions were checked for the truth and falsity of the values.

Black Box testing

Black Box testing is functional requirement of the software. This black box testing enables the software engineering to derive a set of input conditions that will fully exercise all functional requirements for a program. Black box testing is not an alternative to white box testing rather it is complementary approach that is likely to uncover a different class of errors that White box methods like...

1) Interface errors
2) Performance indata structure
3) Performance errors
4) Initializing and terminating errors

8.4 UNIT TESTING

Unit testing is a software verification method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. In procedural programming a unit may be an individual function or procedure. Ideally, each test case is independent from the others: substitutes like method stubs, objects, fakes and test harnesses can be used to assist testing a module in isolation.

8.5 INTEGRATION TESTING

This testing is sometimes called integration and testing. Integration testing is the phase in software testing in which individual software are combined and tested as a group. It occurs after unit and before system testing. Integrated testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates and delivers as its output the integrated system ready for system testing.

Validation testing:

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in a manner that can reasonably expected by a customer. After validation test has been conducted, one of the following two possible conditions exists. The functions or performance characteristics confirm to specification and are accepted.

User acceptance testing

User acceptance of a system is a key factor of any system. The system under consideration is tested for acceptance by constantly keeping in touch.

9.FUTURE ENHANCEMENTS

This framework can be stretched out further to have abundant security highlights, for example, just a certain no of individuals can have specialist get to or work the vehicle. If there should be an occurrence of an endeavor to robbery, the vehicle's motor don't begin or an alarm sounds. A picture of the burglar is taken in an attempted theft &sent to the owner of the vehicle who can register a case against the thief of the vehicle. Moving forward, there are a few things we can do to further improve our results and fine-tune the models. First, we need to incorporate distance between the facial landmarks to account for any movement by the subject in the video. Realistically the participants will not be static on the screen and we believe sudden movements by the participant may signal drowsiness or waking up from micro-sleep. Second, we want to update parameters with our more complex models (NNs, ensembles, etc.) in order to achieve better results. Third and finally, we would like to collect our own training data from a larger sample of participants (more data!!!) while including new distinct signals of drowsiness like sudden head movement, hand movement, or even tracking eye movements.

10. CONCLUSION

In this project, we are developing a machine learning model for drowsiness alert system using python and opencv. We are developing the drowsiness alert system to identify whether the driver is feeling drowsy or not. If the driver is drowsy it detects eyes blinking using webcam and generate buzzer to alert the driver from drowsy. Captured video was divided into frames and each frames were analyzed. Successful detection of face followed by detection of eye. If closure of eye for successive frames were detected then it is classified as drowsy condition else it is regarded as normal blink and the loop of capturing image and analyzing the state of driver is carried out again and again. In this implementation during the drowsy state the eye is not surrounded by square or it is not detected and corresponding message is shown. If the driver is not drowsy then eye is identified by a square and it gives alarm when the driver is in drowsy.

11.BIBLIOGRAPHY

- ✓ Ameratunga.S , Bailey.J, Connor.J, Civil.I, Dunn.R , Jackson.R , Norton.R, and Robinson.E, ―Driver sleepiness and risk of serious injury to car occupants: Population control study. British Medical Journal, vol. 324, 2002, pp. 1125–1129.

- ✓ Bronte.S, Bergasa.L, Delgado.B, Garcia.I, Hernandez.N and Sevillano.M, ―Vision- based drowsiness detector for a realistic driving simulator, ‖ in IEEE Intelligent Transportations Systems Conference (ITSC), 2010.

- ✓ Distante.A, D'Orazio.T, Guaragnella.C and Leo.M, ―A visualapproach for driver inattention detection, Pattern Recogn., vol. 40, no. 8, 2007, pp. 2341–2355.

- ✓ Bradski.G, Kaehler.A, -Learning OpenCV, O'Reilly, 2008.

- ✓ Igarashi.K ,Itou.K, Itakura.F, Miyajima.C, Ozawa.T, Takeda.K and Wakita.T, ―Driver identification using driving behavior signals,IEICE - Trans. Inf. Syst., vol. E89- D, 2006.

- ✓ Nakano.T, Suzuki.M, Yamamoto.N, Yamamoto.O and Yamamoto.S, ―Measurement of driver's consciousness by image processing a method for presuming driver's drowsiness by eye- blinks coping with individual differences. Systems, Man and Cybernetics, vol. 4, 2006.