

**COMPUTER VISION BASED DROWSINESS DETECTION
FOR MOTORIZED VEHICLES WITH
WEB PUSH NOTIFICATIONS**

Submitted in partial fulfilment of the requirements for the award of
Bachelor of Technology Degree in Information Technology

By

ABIRAMI R (Register no 38120006)

ARCHANA C (Register no 38120011)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING
SATHYABAMA INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600 119, TAMIL NADU**

MARCH 2022



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI 600 119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **ABIRAMI R (38120006)** and **ARCHANA C (38120011)** who carried out the project entitled “**COMPUTER VISION BASED DROWSINESS DETECTION FOR MOTORIZED VEHICLES WITH WEB PUSH NOTIFICATIONS**” under our supervision from November 2021 to April 2022.

INTERNAL GUIDE

V.Nirmal Rani M.TECH., Ph.D.,

HEAD OF THE DEPARTMENT

Dr. R. Subhashini M.E., Ph.D.,

Submitted for Viva Voce Examination held on -----

Internal Examiner

External Examiner

DECLARATION

I, **ABIRAMI R (Reg no:38120006)** and **ARCHANA C (Reg no:38120011)** hereby declare that the project report entitled “**COMPUTER VISION BASED DROWSINESS DETECTION FOR MOTORIZED VEHICLES WITH WEB PUSH NOTIFICATIONS**” done by me under the guidance of **MS. Nirmal Rani M.TECH., Ph.D.**, at “Sathyabama Institute of Science and Technology (Deemed to be University), Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119” is submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering/Technology degree in Information Technology.

DATE:

PLACE: CHENNAI

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey our thanks to **Dr. T. Sasikala M.E., Ph.D., Dean, School of Computing and Dr. R. Subhashini M.E., Ph.D., Head of the Department, Department of Information Technology** for providing us the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. Nirmal Rani M.TECH., Ph.D.**, for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all teaching and non-teaching staff members of the **DEPARTMENT OF INFORMATION TECHNOLOGY** who was helpful in many ways for the completion of the project.

ABSTRACT

Since the advent of high-speed motorized vehicles drowsy driving accidents have claimed the lives of millions of people across the globe. With more research happening in advanced high-speed engines, the risk of such accidents has increased ten-fold times. To avoid such accidents, this paper proposes a Computer Vision-based system drowsiness system for motorized vehicles with Web Push Notifications to notify the driver before any accident occurs. A real-time video system captures the face of the driver and a pre-trained machine learning model detects the eye boundaries from that real-time video stream. Then each eye is represented by 6 – coordinates (x,y) starting from the left corner of the eye and then working clockwise around the eye. The EAR (Ear Aspect Ratio) is calculated for 20 consecutive frames, which if less than a threshold sounds an alarm and sends an alert on your mobile device through a Web Push Notification. The alert when opened also shows some coffee shops near the driver's location to increase the driver's alertness.

TABLE OF CONTENTS

CHAPTER NO:	TITLE	PAGE NO
	ABSTRACT	V
	LIST OF FIGURES	Xi
	LIST OF ABBREVIATIONS	xv
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 MOTIVATION	1
	1.3 OUTLINE OF THE PROJECT	2
	1.3.1 PROBLEM STATEMENT	2
	1.3.2 OBJCETIVE OF THE PROJECT	2
2	LITERATURE SURVEY	3
3	AIM AND SCOPE OF THE PRESENT INVESTIGATION	5
	3.1 AIM	5
	3.2 SCOPE	5
	3.3 PROBLEM DEFENITION	5
	3.4 EXISTING SYSTEM	5
	3.5 PROPOSED SYSTEM	6
	3.5.1 ADVANTAGES OF PROPOSED SYSTEM	6
4	EXPERIMENTAL METHODS OR ALGORITHM USED	7
	4.1 ALGORITHM	7
	4.2 ALGORITHM DESIGN AND DEVELOPMENT	7
	4.2.1 VIOLA JONES ALGORITHM STAGES	7
	4.2.2 VIOLA-JONES FACE DETECTION ALGORITHM	8

4.2.3	CASCADE OF CLASSIFIERS	9
4.3	ANALYSIS AND TESTING	11
4.4	MODULES	11
4.4.1	FACE DETECTION	12
4.4.2	FACIAL MAPPING	12
4.4.3	EYE DETECTION	13
4.4.3.1	EYE ASPECT RATIO	14
4.4.4	DROWSINESS DETECTION	15
4.4.5	ALARM	16
4.5	MACHINE LEARNING	16
4.5.1	K-MEANS ALGORITHM	17
4.5.2	NORMAL OR NAÏVE BAYES CLASSIFIER ALGORITHM	17
4.5.3	DECISION TREES ALGORITHM	17
4.5.4	BOOSTING	17
4.5.5	RANDOM TREES ALGORITHM	18
4.5.6	FACE DETECTOR ALGORITHM	18
4.5.7	EXPECTATION MAXIMIZATION (EM) ALGORITHM	18
4.5.8	K-NEAREST NEIGHBOURS	18
4.5.9	NEURAL NETWORKS / MULTILAYER PERCEPTRON (MLP) ALGORITHM	19
4.5.10	SUPPORT VECTOR MACHINE (SVM) ALGORITHM	19
4.6	SYSTEM ARCHITECTURE	19
4.7	USE CASE DIAGRAM	20

5	SYSTEM REQUIREMENT AND DESIGN	21
5.1	REQUIREMENT	21
5.1.1	HARDWARE REQUIREMENT	21
5.1.2	SOFTWARE REQUIREMENT	21
5.2	SOFTWARE DESCRIPTION	21
5.2.1	THE MATLAB SYSTEM	21
5.2.1.1	DEVELOPMENT ENVIRONMENT	21
5.2.1.2	THE MATLAB MATHEMATICAL FUNCTION LIBRARY	21
5.2.1.3	THE MATLAB LANGUAGE	22
5.2.1.4	HANDLE GRAPHICS	22
5.2.1.5	THE MATLAB APPLICATION PROGRAM INTERFACE (API)	22
5.2.2	DEVELOPMENT ENVIRONMENT INTRODUCTION	22
5.2.2.1	STARTING AND QUITTING MATLAB	22
5.2.2.1.1	STARTING MATLAB	22
5.2.2.1.2	QUITTING MATLAB	23
5.2.3	MATLAB APPLICATION	23
5.2.3.1	MATLAB APPLICATION FUNCTIONALITY	24
5.2.3.1.1	CHECK THE DATABASE SERVER FOR ANY INPUT	24
5.2.3.1.2	READ THE IMAGE UPLOADED TO THE SERVER	24
5.2.3.1.3	IMAGE RESIZING	24
5.2.3.1.4	IMAGE CLASSIFICATION	24
5.3	IMAGE PROCESSING TOOL BOX	25

	5.3.1 IMAGE PROCESSING	25
6	RESULTS AND DISCUSSION	26
	6.1 EXPERIMENTAL RESULTS	26
	6.2 LIMITATIONS	29
7	CONCLUSION	30
	7.1 CONCLUSION	30
	7.2 FUTURE ENHANCEMENT	30
	REFERENCES	31
	APPENDIX	32
	SOURCE CODE	32

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO
Fig 4.2.2 (a)	HAAR FEATURE USED IN VIOLA-JONES	8
Fig 4.2.2 (b)	INTEGRAL IMAGE	9
Fig 4.2.3(a)	CASCADE OF CLASSIFIERS	10
Fig 4.2.3(b)	FEATURE FOR FACE DETECTION	11
Fig 4.4.2.(a)	FACIAL LANDMARKS BY DIP	12
Fig 4.4.2.(b)	FACIAL LANDMARKS SET WHICH IS DETECTED VIA DIB	13
Fig4.4.3.1 (a)	EAR	14
Fig4.4.3.1(b)	EYE ASPECT RATIO	15
Fig 4.4.4 (a)	BLOCK DIAGRAM OF DROWSINESS EVALUATOR MODULE FOR A VIDEO SEGMENT	16
Fig 4.6 (a)	SYSTEM ARCHITECTURE	19
Fig 4.7 (a)	USE CASE DIAGRAM	20
Fig 5.2.3.1(a)	MATLAB APPLICATION SNAPSHOT	24
Fig 6.1(a)	MATLAB SNAP	26
Fig 6.1(b)	COMMAND TO DETECT EYES	27
Fig 6.1(c)	COMMAND TO DETECT MOUTH	27
Fig 6.1(d)	CASCADE DETECTORS FOR FACE, EACH EYE AND MOUTH	27
Fig 6.1(e)	SAMPLE OUTPUT 1	28
Fig 6.1(f)	SAMPLE OUTPUT 2	28
Fig 6.1(g)	ALERT NOTIFICATION	28

LIST OF ABBREVIATIONS

ABBREVIATIONS	EXPANSION
DIP	DIGITAL IMAGE PROCESSING
ML	MACHINE LEARNING
DL	DEEP LEARNING
EAR	EYE ASPECT RATIO
CV	COMPUTER VISION
PERCLOS	PERCENTAGE OF EYE CLOSURE
ANN	ARTIFICIAL NEURAL NETWORKS
CNN	CONVOLUTIONAL NEURAL NETWORKS
SVM	SUPPORT VECTOR MACHINE
EM	EXPECTATION MAXIMIZATION
MLP	MULTILAYER PERCEPTRON
ECG	ELECTROCARDIOGRAPHY
EEG	ELECTROENCEPHALOGRAPHY
EOG	ELECTROOCULOGRAPHY

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Drowsy driving has become a serious issue in our society not only because it affects those who are driving, but also all other road users in danger. Accidents due to sleep deprivation is a common problem all around the world. According to a database study conducted by the Australian Transport Bureau in 2002 about fatality crashes it is found that 16.6% of fatal crashes were caused by sleep deprivation and based on the report submitted by Ministry of Transportation of Ontario in 2004, 25.5% of injury causing crashes and 17.8% of crashes with fatality related to sleep deprivation. All of the above statistics is an estimate because in several cases the drivers will not admit the real cause to the Police. The aim of this project is to develop a prototype for detecting drowsiness called drowsiness detection system. The focus will be placed on designing a system that will accurately monitor the driver's facial expressions in real-time. Our first challenge is how to define fatigue exactly and how to measure it. Despite the progress of science, there is still no precise definition for fatigue. Certainly, due to the lack of precise definition of fatigue, there is not any measurable criterion or tool. However, a precise definition for fatigue is not defined yet, but there is a relationship between fatigue and some symptoms including body temperature, eye movement, heart rate. One of the most important symptoms of fatigue appears in the eye. This project helps drivers to get prevented from drastic accidents and the loss of life and property can be saved.

1.2 MOTIVATION

- Driver's safety is an important factor in today's generation. Accidents occur mainly because of aging factor, fatigue, alcohol and distraction.
- The U.S. National Highway Traffic Safety Administration has reported that driving while drowsy is one of the reasons behind road accidents.

- Based on the Ontario Ministry of Transportation's Driver's Handbook, drivers' abilities are affected by drowsiness and fatigue long before they notice that they are getting tired.
- The above are therefore motivations to design and implement an assistive monitoring system in order to detect drivers' drowsiness and fatigue.

1.3 OUTLINE OF THE PROJECT

The project is all about detecting drowsiness for drivers. This project helps the driver to avoid accidents. We are using DIP technique for detecting the facial points. An alarm is been setup so that the driver wakes up. Alarm is issued when the driver closes his eye for a substantial amount of time. In this case, an alarm is issued and the driver wakes up.

1.3.1 PROBLEM STATEMENT

Designing a prototype Drowsiness Detection System which will focus on continuously and accurately monitoring the state of the driver's eyes in real time to check whether they are open or closed for more than a given period of time. Driver's inattention might be the result of a lack of alertness when driving due to driver drowsiness and distraction. Driver distraction occurs when an object or event draws a person's attention away from the driving task. Unlike driver distraction, driver drowsiness involves no triggering event but, instead, is characterized by a progressive withdrawal of attention from the road and traffic demands. Both driver drowsiness and distraction, however, might have the same effects, that is decreased driving performance, longer reaction time, and an increased risk of crash involvement.

1.3.2 OBJECTIVE OF THE PROJECT

Driver drowsiness detection is a car safety technology which spares the life of the driver by avoiding mishaps when the driver is getting languid. The framework works disregarding driver wearing displays and in different lighting conditions.

CHAPTER 2

LITERATURE REVIEW

According to the Survey on Driver Drowsiness Detection System, the detection system includes the processes of face image extraction, the human skin mapping, face part extraction, classification approaches has been used to reach the objective of the work.

Tuncer et al. [1] proposed an assistant system to track a lane, which will be activated for those drivers who are not able to perform a good job of lane keeping. Another method of face detection based on locating facial features is developed by Graf et al. [2].this method states that the morphological operations will be applied to find the areas that have high intensity with certain shapes. In the drowsiness detection method proposed by Pilutti et al, [3] driver assessment is determined in the context of a road departure warning and intervention system. This method states that the vehicle lateral

position is used as the input and steering wheel position as the output in order to develop a system that will be updated during driving. Craw et al. [4] proposed the following method: the frontal view face is detected based on template matching. The edges extracted from Sobel filtering will be grouped together to locate the face.

The same procedure has to be repeated for finding other facial features such as eyes, mouth and nose in the candidate face. Sigary, [5] proposed a method for hypo vigilance detection by processing of the eye region and without an explicit eye detection stage.

Tabrizi and Zoroofi [6] proposed a non-intrusive and simple way of fatigue detection by determining whether the eye is open or closed. The algorithm consist of three steps which were analyzed, such as determining eye regions by eye map and locating pupil center by the center of mass of the eye region image and the refining the pupil center and detecting the iris boundary. Omidyeganeh et al. [7] used a method of fatigue detection by applying the Structural Similarity Measure to find the eye location. In this method, the structural similarity value will be evaluated between -1 and 1. When images are the same, the max gained value will be 1 and when there are some differences, the result will be -1. Then the horizontal and vertical projection will be applied on the eye region to determine the amount of eye closure and align the detected eye region. M. Saradadevi and P.Bajaj,[8]proposed Driver Fatigue Detection Using Mouth and Yawning Analysis,

IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 6, pp. 183-188, June 2008.

The robust and reliable method of face detection based on the Viola-Jones theory has been used by Wang and Shi [9] in order to limit the mouth search area to the face region. The mouth region is located based on multi-threshold binarization in intensity space and by using the Gaussian model in RGB color space. The lip corner will be found by calculating the integral projection of the mouth in the vertical direction. The two lines which are running through the lower and upper lip boundaries that is resulting from the integral projection which represents the mouth openness. In this method, the yawning will be determined by finding the degree of mouth opening in terms of the aspect ratio of the mouth bounding rectangle. A huge mouth opening over a predefined threshold for a continuous number of frames means that the driver is in a state of drowsiness.

A. Cheng et. al. [10] described 'Driver Drowsiness Recognition Based on Computer Vision Technology'. They presented an nonintrusive drowsiness recognition method using eye-tracking and image processing. Robust eye detection algorithm was introduced to address the problems caused by changes in illumination and driver posture.

CHAPTER 3

AIM AND SCOPE OF THE PRESENT INVESTIGATION

3.1 AIM

The aim of our project is to design a system that will detect drowsiness and take necessary steps to avoid accidents. The driver drowsiness detection system, which is been implemented in this project aims that it can easily be available and can be used with different types of vehicles.

3.2 SCOPE:

- To automatically recognize the sleep and drowsiness of the driver and its detection.
- To analyze the focus of driver while driving.
- To determine concentration of drivers while driving.
- To detect whether the camera is focusing properly and the driver's face.
- To verify the drowsiness.

3.3 PROBLEM DEFINITION

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

3.4 EXISTING SYSTEM

In the driver face monitoring systems, useful symptoms for fatigue and distraction detection can be divided into three general categories:

- (i) symptoms related to the eye region;
- (ii) symptoms related to the mouth region;
- (iii) symptoms related to the head.

Eye is the most important area of the face where the symptoms of fatigue and distraction appear in it. Therefore, many of the driver face monitoring systems detect driver fatigue and distraction only based on the symptoms extracted from the eyes. The symptoms related to eye region include PERCLOS eyelid distance, eye blink speed, eye blink rate, and gaze direction. Yawning is one of the hypo vigilances symptoms related to the mouth region.

This symptom was extracted by detecting the open mouth. The existing system of driver drowsiness detection system has following disadvantages. There are mainly, using of two cameras in the system one for monitoring the head movement and the other one for facial expressions. The other disadvantage is aging of sensors and all these sensors are attached to the driver's body which may affect the driver.

3.5 PROPOSED SYSTEM

3.5.1 ADVANTAGES OF PROPOSED SYSTEM

- Efficiency while driving and safe driving.
- Decreased death rates.
- Reduced manual work.
- As the system is automated it doesn't require
- more resources like hand written record of
- Driver's safety, but the record is maintained in the database.
- The system has less hardware requirements in comparison to the other biometric system.
- As the system uses fewer resources therefore the cost of the system is less.
- The system also reduces the human effort.

CHAPTER 4

EXPERIMENTAL METHODS OR ALGORITHMS USED

4.1 ALGORITHM:

The overall algorithm is pretty straightforward one. First we have used a camera which is setup at desirable position in a car that looks for faces stream.

Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye- lids are down for a substantial amount of time.

The implementation of the drowsiness detector system includes machine learning algorithms which is viola jones algorithm. It is used to face detection only. It can be trained to detect a variety of object classes it was primarily by the problem of face detection.

4.2 Algorithm Design and Development

A few algorithms and technique has been used in the process of detecting face, eyes and mouth. The algorithm and technique used is Cascade Object Detector. The Cascade Object Detector uses the Viola-Jones algorithm to detect people's face, nose, eyes, mouth or upper body.

4.2.1 Viola jones algorithm stages :

- Selection of the Haar Feature for image.
- Creating the specified Integral Images.
- Adaboost Training for the image dataset.
- Cascading method for the Classifier.

4.2.2 Viola-Jones Face Detection Algorithm

Viola-Jones object detection framework can be used to detect a variety of object classes, but it is more focused on the detection of face and facial features. This algorithm uses the concept of rectangle features which involves the sums of pixels within the rectangular areas. From Figure 8, the sums of the pixels that lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles.

The value of a *two-rectangle feature*, which are represented by A and B is the difference between the sum of pixels within two rectangular regions. The regions have the same size and shape. They are also horizontally or vertically oriented and adjacent to each other. A *three-rectangle feature*, which is represented as C, computes the sum within two outside rectangles subtracted from the sum in a center rectangle. Finally, a *four-rectangle feature*, which is represented as D computes the difference between diagonal pairs of rectangles.

The classifiers use Haar features in order to encode facial features. Features used in are seen on Figure 4.2.2

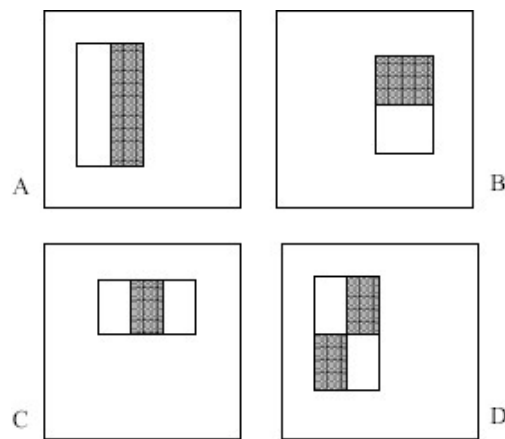


Figure4.2.2 (a): Haar Feature Used in Viola-Jones

For each feature, the value is the difference between the sum of the pixels in black regions and the sum of the pixels in white regions.

Rectangle features can be computed rapidly using a representation for the image which is called integral image. which is an intermediate representation. The integral image at location pixels (x, y) contains the sum of the upper left pixels of the original image, inclusively. As seen on Figure 4.2.2 .

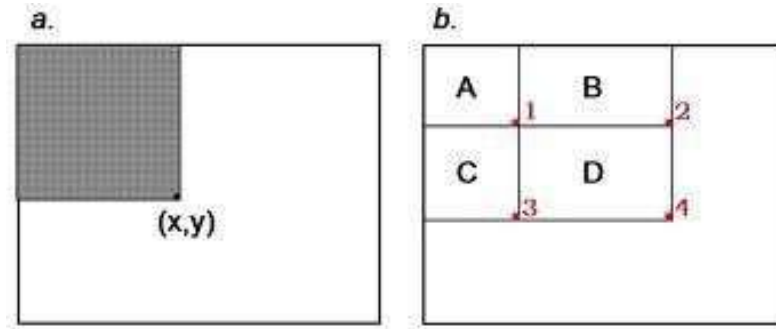


Figure 4.2.2 (b) : Integral Image

The value of the integral image at point (x, y) is the sum of all the pixels above and to the left. Based on the integral image, the sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is $A + B$, at location 3 is $A + C$, and at location 4 is $A + B + C + D$. Then, the sum within D is computed as $4 + 1 - 2 - 3$ that means the sum of the pixels within rectangle D is computed with four array references. Integral image provides the advantage of fast feature evaluation.

Adaboost is used to select a small set of features and train the classifiers. The learning algorithm for weak classifiers is designed to select the single rectangle feature best separating the positive and negative examples. For each feature, the weak classifier determines the optimal threshold.

4.2.3 Cascade of Classifiers

In a standard 24x24 pixel sub-window, there are a total of 45,396 possible features that are detected. This is a number that is too large and prohibitively expensive to be evaluated. In order to improve detection performance, features need to be added to the classifiers. However, this step directly increases the computation time and making the detection process much slower. Therefore, a cascade of classifiers is constructed to increase detection performance while radically reducing computation time.

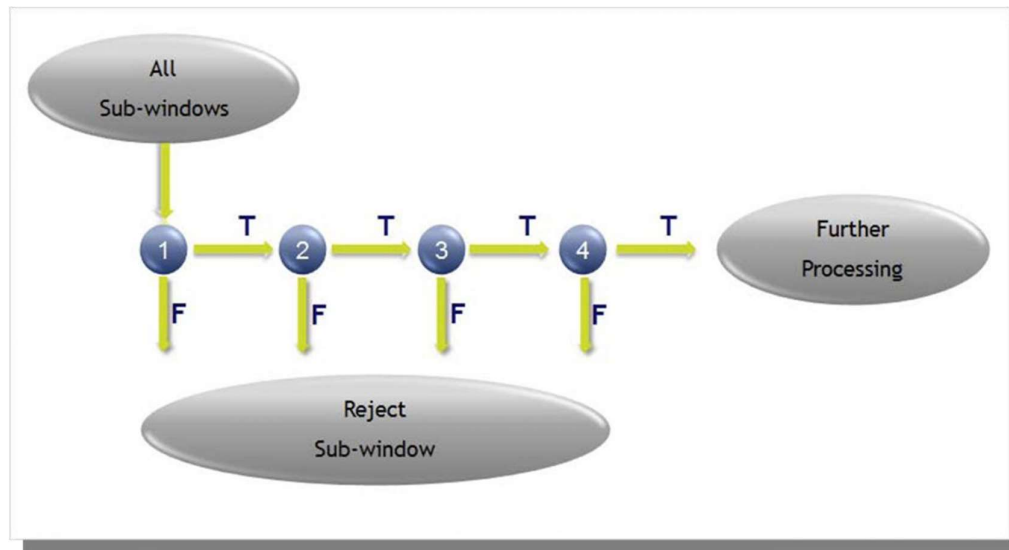


Figure 4.2.3(a) : Cascade of Classifiers

The evaluation of the strong classifiers generated by the learning process can be done quickly but it is not fast enough to run in real-time. Therefore, the strong classifiers are arranged in a cascade in order of complexity. Each successive classifier is trained only on selected samples which pass through the preceding classifiers. If at any stage in the cascade a classifier rejects the sub-window under inspection, no further processing is performed and it continues to search the next sub-window.

Weak classifiers rejecting non-face regions and accepting face regions, are trained in such a way that their false negative rate approaches 0. Then, they are combined in a form of a degenerate decision tree which is called “cascade”.

For the task of face detection, the features that are selected to detect the location of the face are shown in Figure 10. The two features are shown in the top row and are overlay on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. This is because the region of the eyes is often darker than the region of the nose and upper cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

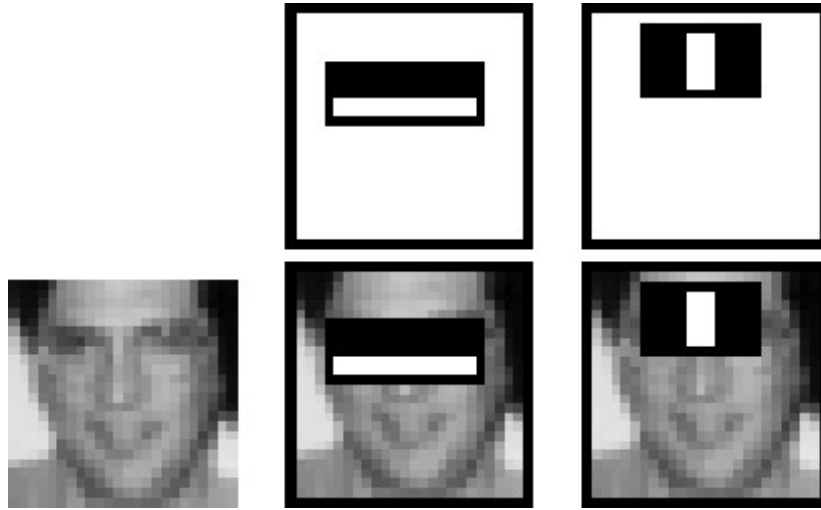


Figure 4.2.3(b): Feature for Face Detection

4.3 Analysis and Testing

In order to detect eye and mouth area, the face area need to be detected first. However, this step will reduce the performance and the speed of the system due to a large area of detection. The project objective is the detection of the drowsiness signs which is eyes and mouth area. Therefore, this project limited the area of detection to eye and mouth. This will enhance the performance of the system. Cascade Object Detector algorithm is being tested by using MATLAB® software to get the detection area for the system that will be develop. Testing need to be done to ensure it fulfils the required parameters needed.

4.4 MODULES

- Face detection
- Facial Mapping
- Eye detection
- Drowsiness detection
- Alarm

4.4.1 FACE DETECTION

For face detection itself, several approaches have been used in the related literature. Knowledge based methods try to encode human knowledge about the characteristics of a typical face, such as the relationships between facial features, and use them as a way to detect faces in an image. The goal of the feature invariant approaches is to find structural face features, such as eyebrows, eyes, nose, mouth and hairline, which persist under various poses, viewpoints or lighting and use those features to detect faces. Such features are mostly extracted using edge detectors. The video is obtained from a camera focused on the driver's face. The Haar-based classifier contains several features such as heights, weights, face features, the threshold of face colours. It is constructed by using a lot of positive and negative samples. The cascade consists both positive and negative samples. Eyes and mouth features are extracted and parallel processing is preceded by successful driver's face detection.

4.4.2 FACIAL MAPPING

The algorithm is implemented using a DIP that contains a landmark's facial detector with pre trained models. The 68-point shape predictor dataset was used to train the DIP facial landmark predictor and is the source of these marking.

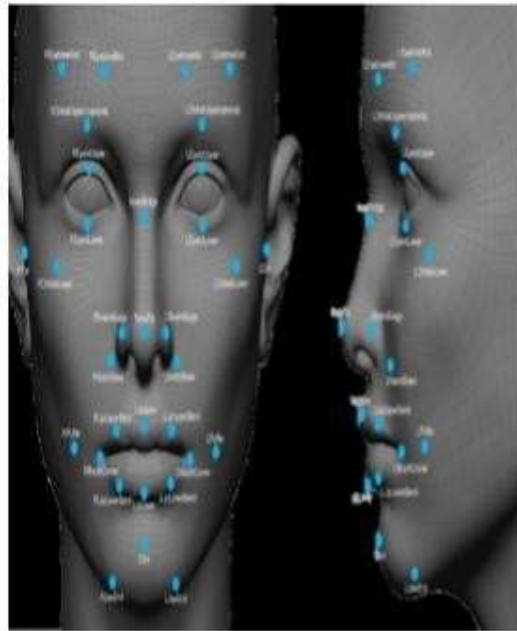


Fig.4.4.2.(a) Facial landmarks by DIP

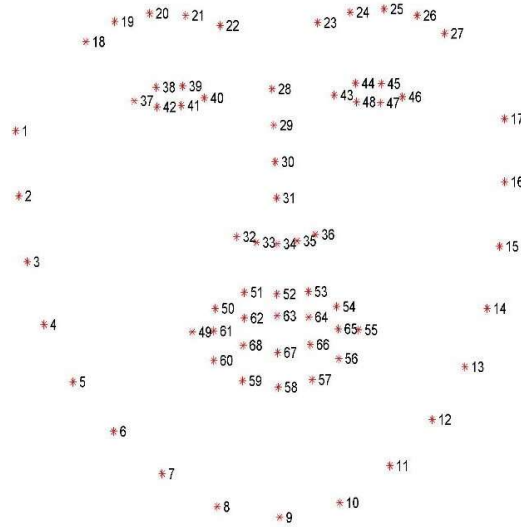


Fig 4.4.2.(b) Facial landmarks set which is detected via dib

4.4.3 EYE DETECTION

Eye tracking is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head. An eye tracker is a device for measuring eye positions and eye movement. Eye trackers are used in research on the visual system, in psychology, in psycholinguistics, marketing, as an input device for human-computer interaction, and in product design. Eye trackers are also being increasingly used for rehabilitative and assistive applications (related for instance to control of wheel chairs, robotic arms and prostheses). There are a number of methods for measuring eye movement. The most popular variant uses video images from which the eye position is extracted. We here are using eye tracking in detecting the drowsiness of the driver. Eye tracking is helping us to detect and sense the sleep of the driver, whether he is sleeping, wanting to sleep, getting exhausted while driving etc. We are first trying to generate a preview of the driver through the web camera of the laptop. The driver's preview image is being captured simultaneously by the web camera and the camera forms to give us a

preview image. The camera now starts to record the video and automatically saves it in the backend so that it can be analyzed. After the images, videos, frames are being recorded and saved, the eyes are detected from them and then tracked.

4.4.3.1 EYE ASPECT RATIO (EAR)

The eye tracking involves to calculate the value for the eye aspect ratio. Eye aspect ratio (EAR) is a parameter that Determines eye state used to figure out if it is Open or closed. It can be calculated using facial landmarks plotted by the 68 facial landmark point plot provided by python's dlib library. Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye-lids are down for a substantial amount of time. Once we get the eye region, we calculate the Eye Aspect Ratio to find

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

Figure 4.4.3.1 (a) EAR

out if the eye-lids are down for a substantial amount of time. Where

$p_1 \dots p_6$ are 2D facial landmark location.

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.

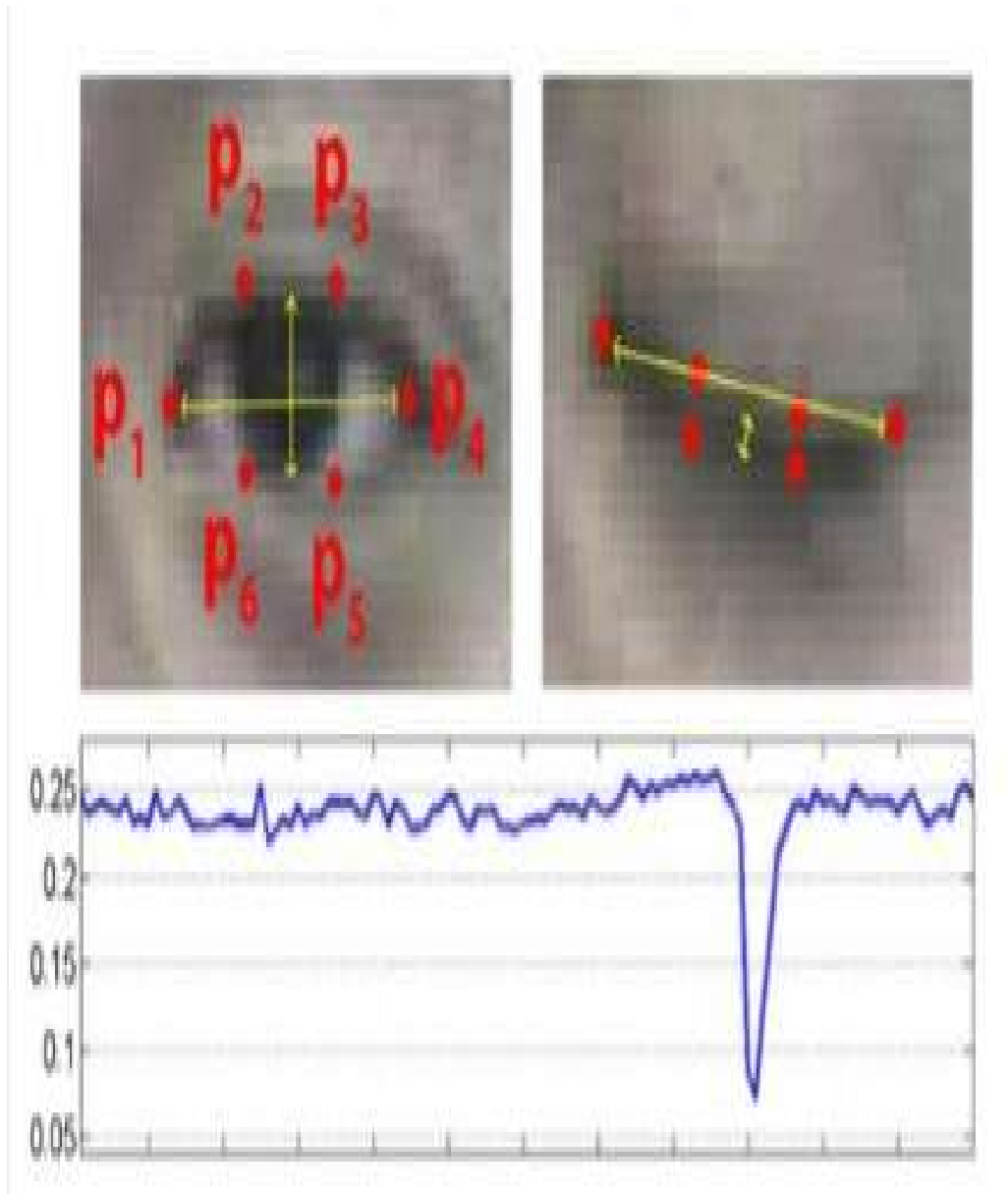


Fig 4.4.3.1(b) Eye aspect ratio

4.4.4 DROWSINESS DETECTION

Drowsiness is characterized by a widely opened mouth. Yawning detection is performed in two main steps: In the first step, the yawn component will be detected in the face independent of the mouth location. This component is basically the hole in the face as the result of wide mouth opening. In the second step, the mouth location will be used to verify the validity of the detected component. Like the eye closure detection, the facial landmarks are

used to detect an open mouth. Lip distance is the parameter used to determine if the subject's mouth is open. the lip distance calculated from the frame is above lip distance threshold, the subject is determined to be drowsiness. An alarm is raised if the subject have drowsy more than the set boundary value consecutively. Small openings that in reality are construed as a result of talking, eating are ignored.

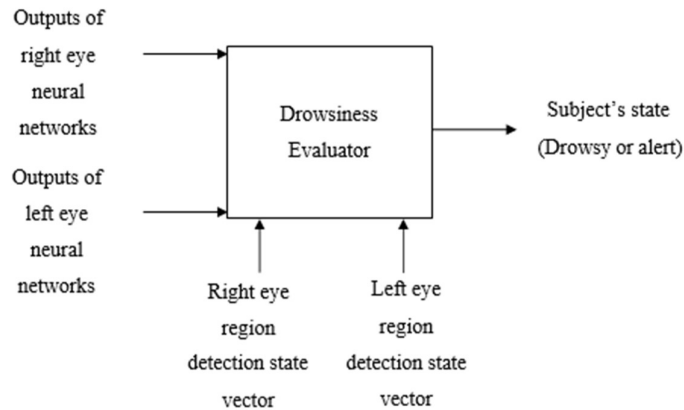


Figure 4.4.4 (a) : Block diagram of Drowsiness Evaluator module for a video segment

4.4.5 ALARM

The alarm sound alerts the driver in case of drowsiness is detected. Instead of an alarm i.e buzzer we are giving here an text voice to alert the driver. For that we are using a module called web push notifications.

Web Push Notifications are platform independent for linux , windows and other operating Systems.

4.5 MACHINE LEARNING

Machine learning (ML) is the study of computer algorithms that improve automatically through experience. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning involves

computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

4.5.1 K-means Algorithm

It is an unsupervised clustering algorithm which signifies a distribution of data w.r.t. K centers, K being chosen by the coder. The difference between K-means and expectation maximization is that in K-means the centers aren't Gaussian. Also the clusters formed look somewhat like soap bubbles, as centers compete to occupy the closest data points. All these cluster areas are usually used as a form of sparse histogram bin for representing the data.

4.5.2 Normal or Naïve Bayes classifier algorithm

It is a generative classifier where features are often assumed to be of Gaussian distribution and also statistically independent from one another.

4.5.3 Decision trees algorithm:

It is a partially discriminative classifier. The tree we talk about just finds a singular data feature and determines a threshold value of the current node which best divides the data into different classes. The data is broken into parts and the procedure is recursively repeated through the left as well as the right branches of the decision tree. Even if it is not the top performer, it's usually the first thing we try as it is fast and has a very high functionality.

4.5.4 Boosting:

It is a discriminative group of classifiers. In boosting, the final classification decision is made by taking into account the combined weighted classification decisions of the group of classifiers. We learn in training the group of classifiers one after the other. Each classifier present in the group is called a weak classifier. These weak classifiers are usually composed of single-variable decision trees known as —stumps. Learning its classification decisions from the given data and also learning a weight for its vote based on its accuracy on the data are

things the decision tree learns during training. While each classifier is trained one after the other, the data points are re-weighted to make more attention be paid to the data points in which errors were made. This continues until the net error over the entire data set, obtained from the combined weighted vote of all the decision trees present, falls below a certain threshold. This algorithm is usually effective when a very large quantity of training data is available.

4.5.5 Random trees algorithm

It is a discriminative woods of a great deal of choice trees, every one of which is worked down to a maximal part profundity. At the season of adapting, each hub of each tree is permitted a decision of part factors, however just from an arbitrarily produced subset of the considerable number of information highlights. This makes sure that all the trees become statistically independent and a decision maker. In the run mode, all the trees get an unweighted vote. Random trees are usually quite effective. They can also perform regression by taking the average of the output numbers from every tree.

4.5.6 Face detector algorithm

It is an object detection application. It is based on a smart use of boosting. A trained frontal face detector is available with the OpenCV distribution. This works remarkably well. We can train the algorithm for other objects by using the software provided. This works wonderfully for rigid objects.

4.5.7 Expectation maximization (EM) algorithm

It is used for clustering. It is a generative unsupervised algorithm It fits N multidimensional Gaussians to the data, N being chosen by the user. It can act as an efficient way for representing a more complex distribution using only a few parameters (i.e. means and variances). Usually used in segmentation, it can be compared with K- means.

4.5.8 K-nearest neighbours

a lot of It is one of the simplest discriminative classifier. The training data is simply stored using labels. Then, a test data point is classified in accordance to the majority vote of the K nearest data points. K-nearest neighbours is probably the simplest algorithm we can use .It is usually effective but it can be slow. It also requires memory.

4.5.9 Neural networks / Multilayer perceptron (MLP) algorithm

It is a discriminative algorithm which almost always contains hidden units in between the output and the input nodes for better representation of the input signal. It is slow to train,

however it is quite fast to run. It remains the best performer for applications like letter recognition.

4.5.10 Support vector machine (SVM) algorithm

It is a discriminative classifier that is likewise equipped for doing relapse. Here, a separation work in the middle of two information focuses is characterized in a higher- dimensional space. (Anticipating information onto higher measurements helps in making the information more probable for direct division.) Support vector machine (SVM) gets the hang of isolating hyperplanes which maximally separate every one of the classes in the higher measurement. This will in general be the best when there is restricted information. Be that as it may, when huge informational indexes are accessible, boosting or arbitrary trees are liked.

4.6 SYSTEM ARCHITECTURE

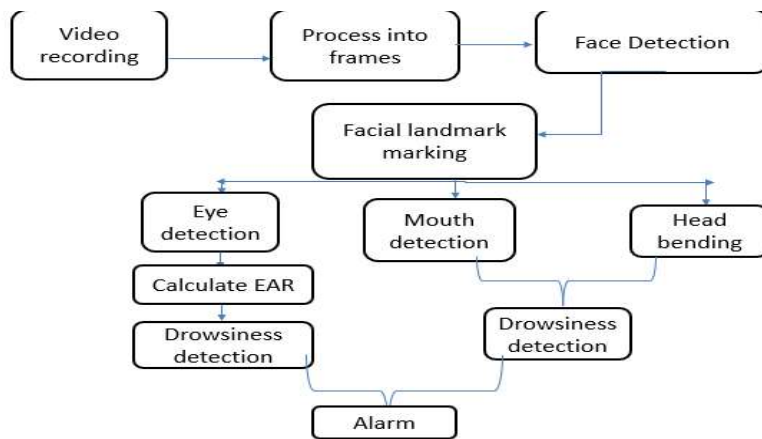


Fig 4.6 (a): system architecture

The diagram depicts how the full function of the drowsiness detection is carried out.

It describes the importance of each step that is required to complete the detection of drowsiness. We can see how first the brightness and contrast level of the camera is adjusted. Then the face is detected.

If it is successfully previewed then only the further step is taken. The eye detection takes place.

The decision for proper eye detection is taken and then the eye region is focused and extracted.

The eyes are determined whether they are closed or opened.

The drowsiness is calculated through the data being stored and saved. The drowsiness is judged now.

If the driver is drowsy then alarm rings loud and alerts the driver.

This is how the detection for driver drowsiness system works and it alerts the driver and the accidents is prevented from being occurred.

4.7 USE CASE DIAGRAM

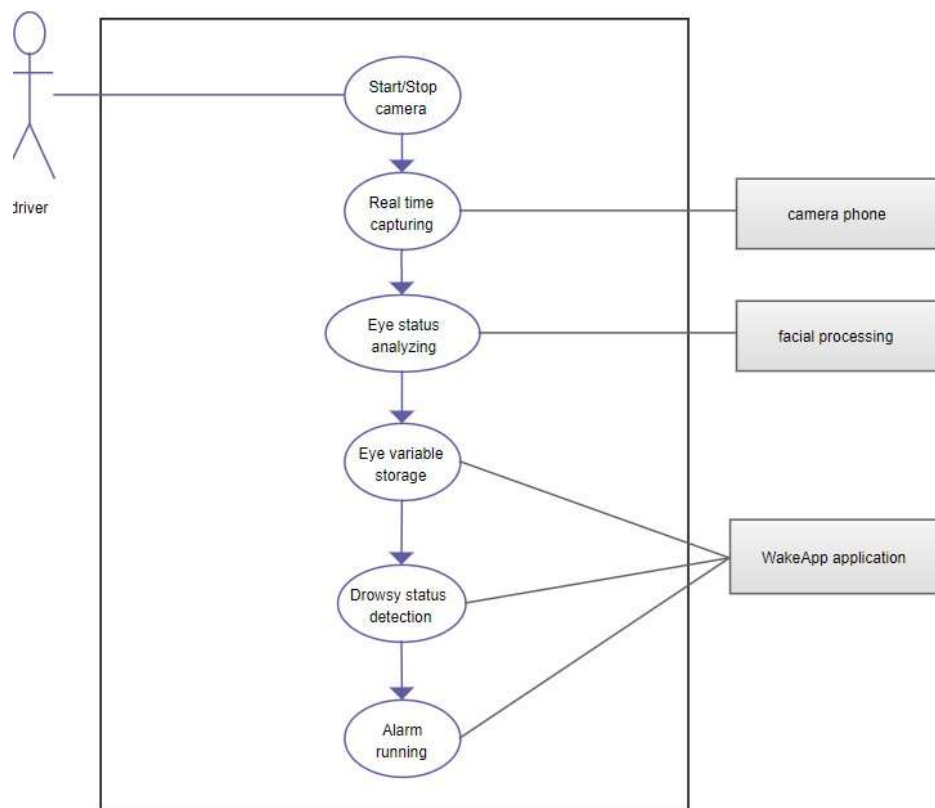


fig 4.7 (a): use case diagram

CHAPTER 5

SYSTEM REQUIREMENT AND DESIGN

5.1 REQUIREMENT

5.1.1 HARDWARE REQUIREMENT

- A system with minimum 8GB RAM.
- Camera
- Alarm
- Node MCU
- UART Protocol

5.1.2 SOFTWARE REQUIREMENT

- MATLAB 2021a
- Image processing tool box

5.2 SOFTWARE DESCRIPTION

5.2.1 The MATLAB System

The MATLAB system consists of five main parts:

5.2.1.1 Development Environment:

This is the set of tools and facilities that help you use MATLAB functions and files. Many of these tools are graphical user interfaces. It includes the MATLAB desktop and Command Window, a command history, and browsers for viewing help, the workspace, files, and the search path.

5.2.1.2 The MATLAB Mathematical Function Library:

This is a vast collection of computational algorithms ranging from elementary functions like sum, sine, cosine, and complex arithmetic, to more sophisticated functions like matrix inverse, matrix eigenvalues, Bessel functions.

5.2.1.3 The MATLAB Language:

This is a high-level matrix/array language with control flow statements, functions, data structures, input/output, and object-oriented programming features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create complete large and complex application programs.

5.2.1.4 Handle Graphics:

This is the MATLAB graphics system. It includes high-level commands for two-dimensional and three-dimensional data visualization, image processing, animation, and presentation graphics. It also includes low-level commands that allow you to fully customize the appearance of graphics as well as to build complete graphical user interfaces on your MATLAB applications.

5.2.1.5 The MATLAB Application Program Interface (API):

This is a library that allows you to write C and FORTRAN programs that interact with MATLAB. It includes facilities for calling routines from MATLAB (dynamic linking), calling MATLAB as a computational engine, and for reading and writing MAT-files.

5.2.2 DEVELOPMENT ENVIRONMENT Introduction

This chapter provides a brief introduction to starting and quitting MATLAB, and the tools and functions that help you to work with MATLAB variables and files. For more information about the topics covered here, see the corresponding topics under Development Environment in the MATLAB documentation, which is available online as well as in print.

5.2.2.1 Starting and Quitting MATLAB

5.2.2.1.1 Starting MATLAB:

On a Microsoft Windows platform, to start MATLAB, double-click the MATLAB shortcut icon on your Windows desktop.

On a UNIX platform, to start MATLAB, type matlab at the operating system prompt. After starting MATLAB, the MATLAB desktop opens - see MATLAB Desktop. You can change the directory in which MATLAB starts, define startup options including running a script upon startup, and reduce startup time in some situations.

5.2.2.1.2 Quitting MATLAB:

To end your MATLAB session, select Exit MATLAB from the File menu in the desktop, or type quit in the Command Window. To execute specified functions each time MATLAB quits, such as saving the workspace, you can create and run a finish.m script.

5.2.3 MATLAB Application

The MATLAB application is consisted of a several files, each one is responsible of a certain

functionality, and it's divided into three parts:

- **Learning Model:** This part is related to the learning model training and evaluation, and is executed once a new model to be trained and it contains loading the data using Image Datastore object and feed it to the model to be trained as explained in section.
- **Server Connection:** This part is related to establishment of the connection between the MATLAB and the server, and is consist of two parts the first one for establishing the connection, called once at the beginning and is used to set the host address, the port used, username, the password, java database connector path and string, then use the JDBC driver to connect to the database server.
- **Preprocessing and image manipulation:** This part includes several preprocessing functions which is applied to the data before the training or for the data to be classified, and contains functions for.

5.2.3.1 MATLAB application functionality

5.2.3.1.1 Check the database server for any input :

The MATLAB application is running always on the server side and this function is listening to any changes in the database server, any addition in the database server means that the android application wrote a new input, then this function triggers the MATLAB application to process the new input.

5.2.3.1.2 Read the image uploaded to the server :

When the MATLAB discovers a new input from the database server, this function read the image path from the database and use that path to load the image into the application to perform a classification task.

5.2.3.1.3 Image resizing:

When the image is uploaded to the MATLAB application, this function changes the size of the image to a certain size that is configured in the model to be suitable for the classification.

5.2.3.1.4 Image classification:

This task is to load a pre-trained model and use it to classify the input data into a suitable class of skin diseases, this function is the main function of the application.

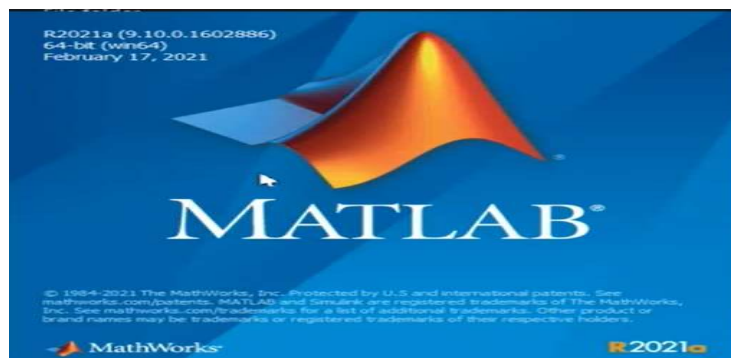


Fig 5.2.3.1(a): MATLAB APPLICATION SNAPSHOT

5.3 IMAGE PROCESSING TOOL BOX:

5.3.1 IMAGE PROCESSING

In computer science, digital image processing is the use of computer algorithms to perform image processing on digital images. Image Processing Toolbox™ provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization, and algorithm development. You can perform image segmentation, image enhancement, noise reduction, geometric transformations, and image registration using deep learning and traditional image processing techniques. The toolbox supports processing of 2D, 3D, and arbitrarily large images.

Image Processing Toolbox apps let you automate common image processing workflows. You can interactively segment image data, compare image registration techniques, and batch-process large datasets. Visualization functions and apps let you explore images, 3D volumes, and videos; adjust contrast; create histograms; and manipulate regions of interest (ROIs).

You can accelerate your algorithms by running them on multicore processors and GPUs. Many toolbox functions support C/C++ code generation for desktop prototyping and embedded vision system deployment.

The Image Processing Toolbox provides a set of functions and applications for image processing, analysis, and visualization. There are many functions available for image analysis, image segmentation, image enhancement, noise reduction, geometric transformations, and image registration.

RESULTS AND DISCUSSION

6.1 EXPERIMENTAL RESULTS

After doing literature survey, different techniques have been found for detecting driver drowsiness and they use different types of data as input for their algorithm. First, we have used a camera which is setup at desirable position in a car that looks for faces stream. The face of the driver will be found in the video in such a way that it should not affect the performance of accurate face detection in terms of varying lightning conditions. If face gets detected, the facial landmark detection task is applied and region of eyes and mouth is extracted. Once we get the eye region, we calculate the Eye Aspect Ratio to find out if the eye-lids are down for a substantial amount of time and we are detecting the yawn. If the driver is drowsy then the driver is alerted with an alarm sound.

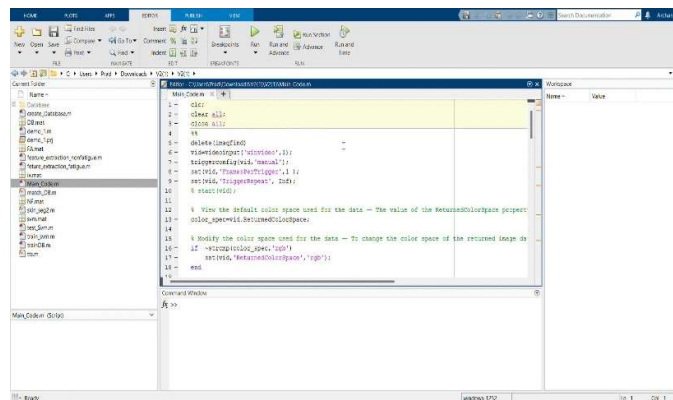


Fig 6.1(a): Matlab snap

```

1 % Create a cascade detector object.
2 faceDetector = vision.CascadeObjectDetector();
3
4 % Create a cascade detector to detect Left eye.
5 LeftEyeDetector = vision.CascadeObjectDetector('LeftEye');
6
7 % Create a cascade detector to detect Right eye.
8 RightEyeDetector = vision.CascadeObjectDetector('RightEye');
9
10 % Read a video frame and run the detector.
11 videoFileReader = vision.VideoFileReader('video5.mp4');
12 videoFrame = step(videoFileReader);
13 bbox = step(faceDetector, videoFrame);
14
15 cgface=[bbox(1,1)+bbox(1,3)/2, bbox(1,2)+bbox(1,4)/2];
16
17 videoFileReader = vision.VideoFileReader('video5.mp4');
18 videoFrame = step(videoFileReader);
19 bbox_1 = step(LeftEyeDetector, videoFrame);
20
21 % Remove outlier left eye
22 [n1,nc]=size(bbox_1);
23 for i=1:n1
24     cgtmp=[bbox_1(i,1)+bbox_1(i,3)/2, bbox_1(i,2)+bbox_1(i,4)/2];
25     if ( cgtmp(1,1)<cgface(1,1) && (cgtmp(1,2)<cgface(1,2) ) )
26         CoordLeftEye=bbox_1(i,1);
27     end
28 end

```

```

1 % Create a cascade detector object.
2 Detector = vision.CascadeObjectDetector('Mouth');
3
4 % Read a video frame and run the detector.
5 videoFileReader = vision.VideoFileReader('Video2.mp4');
6 videoFrame = step(videoFileReader);
7 bbox = step(Detector, videoFrame);
8
9 % Draw the returned bounding box around the detected face.
10 videoOut = insertObjectAnnotation(videoFrame, 'rectangle', bbox, 'Mouth');
11 figure, imshow(videoOut), title('Detected Mouth');

```

The screenshot shows the MATLAB R2014a environment. The top toolbar includes tabs for HOME, PLOTS, APPS, EDITOR, PUBLISH, and VIEW. The EDITOR tab is active, displaying a script named 'FINALcode.m'. The script's content is as follows:

```

1 clear; close all; clc;
2
3 %% Creation of the detectors for Face, Eyes (left and right) and Mouth
4
5 % Create a cascade detector object to detect Face.
6 FaceDetector = vision.CascadeObjectDetector();
7
8 % Create a cascade detector to detect Left eye.
9 LeftEyeDetector = vision.CascadeObjectDetector('LeftEye');
10
11 % Create a cascade detector to detect Right eye.
12 RightEyeDetector = vision.CascadeObjectDetector('RightEye');
13
14 % Create a cascade detector to detect Mouth.
15 MouthDetector = vision.CascadeObjectDetector('Mouth');
16
17 % Create an edge detector to detect if the eye is open or closed
18 EdgeDetector = vision.EdgeDetector('BinaryImageOutputFormat', false, 'GradientComponentOutputFormat', 'nbpdx');
19
20 % Create a line process
21 t=linespace(-nbpdx,nbpdx,2*nbpdx);
22 sig=1.5;
23 hexp=(-(t.^2)/(2*sig^2))./(sqrt(2*pi)*sig);
24 figure, plot(h);
25 y=(-(2*pi)/(2*sig^2)).*exp(-(t.^2)/(2*sig^2))./(sqrt(2*pi)*sig);
26 figure, plot(y);
27 Hx=h'*y;
28 Hy=y'*x;
29
30

```

The script is being executed, as indicated by the 'Run' button in the toolbar and the 'Run' status in the bottom right corner. The Command Window shows the execution progress.

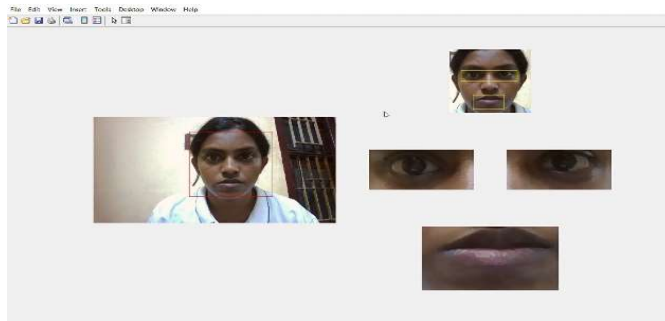


Fig 6.1(e): sample output 1

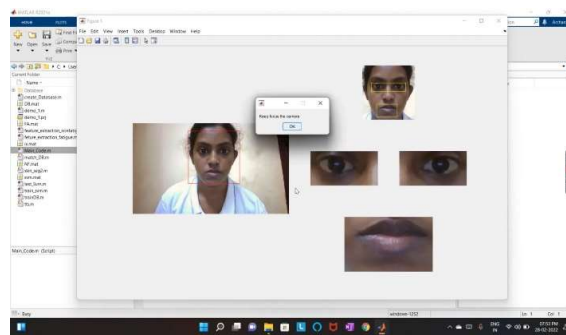


Fig 6.1(f): sample output 2

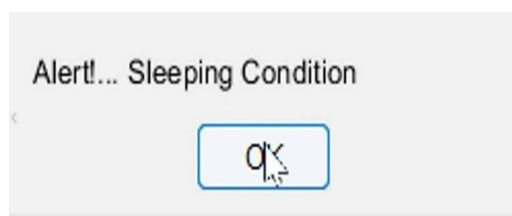


Fig 6.1(g): Alert Notification

6.2 LIMITATIONS

1. Dependence on ambient light

With poor lighting conditions even though face is easily detected, sometimes the system is unable to detect the eyes. So it gives an erroneous result which must be taken care of. In real time scenario infrared backlights should be used to avoid poor lighting conditions.

2. Hardware requirements

Our system was run in a PC with a configuration of 1.6GHz and 1GB RAM Pentium dual core processor. Though the system runs fine on higher configurations, when a system has an inferior configuration, the system may not be smooth and drowsiness detection will be slow. 46 The problem was resolved by using dedicated hardware in real time applications, so there are no issues of frame buffering or slower detection.

3. Poor detection with spectacles

When the driver wears glasses the system fails to detect eyes which is the most significant drawback of our system. This issue has not yet been resolved and is a challenge for almost all eye detection systems designed so far.

4. Problem with multiple faces

If more than one face is detected by the webcam, then our system gives an erroneous result. This problem is not important as we want to detect the drowsiness of a single driver.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Hence driver drowsiness detection is a very important part for the driver while he drives the car. So every driver should have a drowsiness detection system in his vehicle so that he can be almost safe and secure from road accidents, deaths from road accidents. A sleep of even a second can take the life of the driver. Then why to take any risk? This risk can be prevented when the driver uses the drowsiness detection system. The truck drivers, car passengers, taxi drivers, all can be aware about this system. They can manage to get help through it. Also they must be tracked on every route where they are driving to prevent any mishappening. The fatigue detection system in drivers has thus been simulated in Open CV. The real time system has been successfully created to detect the face and hence the eyes and mouth of the driver to check whether he is blinking or yawning to acquire information about his level of alertness. The alarm that has been implemented is a web push notification software which converts text to speech to alert the driver. To realize the drowsiness detection on the real roads, the head posture is also a good choice in the future work.

7.2 FUTURE ENHANCEMENT

Some drivers cover their mouth while yawning or they have different signs of sleepiness like eye closure or falling head; in this case, future work may consist of combining the detection of different fatigue signs.

When the driver wears glasses the system may not detect eyes which is the most noteworthy disadvantage of these systems. This issue has not yet been settled in near future it can be sorted out.

REFERENCES

1. O. Tuncer, L. E. Karsl Gil, "Vision based lane keeping assistance control triggered by a driver inattention monitor", IEEE International Conference on Systems Man and Cybernetics (SMC), Istanbul, 10-13 Oct. 2010.
2. H. Graf, and E. Cosatto, "Locating Faces and Facial Parts", in Proc. First Intl Workshop Automatic Face and Gesture Recognition, 1995.
3. T. Pilutti, A. Ulsoy, "Identification of driver state for lane-keeping tasks", IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans, vol. 29, no. 5, pp. 486-502, 1999
4. I. Craw, J. Lishman, "Automatic Extraction of Face Features", Pattern Recognition Letters, vol. 5, pp. 183-187, 1987.
5. M. Sigari, "Driver Hypo-vigilance Detection Based on Eyelid Behaviour", in Seventh International Conference on Advances in Pattern Recognition, 2009.
6. P. Tabrizi and R. Zoroofi, "Open or Closed Eye Analysis for Drowsiness Detection", First Workshops on Image Processing Theory, Tools and Applications, 2008.
7. M. Omidyeganeh, A. Javadtalab and S. Shirmohammadi, "Intelligent driver drowsiness detection through fusion of yawning and eye closure", in IEEE International Conference on Virtual Environments Human-Computer Interfaces and Measurement Systems (VECIMS), 2011.
8. M. Saradadevi and P. Bajaj, "Driver Fatigue Detection Using Mouth and Yawning Analysis" IJCSNS International Journal of Computer Science and Network Security, vol. 8, no. 6, pp. 183-188, June 2008.
9. T. Wang and P. Shi, "Yawning detection for determining driver drowsiness", in Proceedings of 2005 IEEE International Workshop on VLSI Design and Video Technology, 2005.
10. Zhang, Wei; Cheng, Bo; Lin, Yingzi, "Driver drowsiness recognition based on computer vision technology." Published in: Tsinghua Science and Technology (Volume: 17, Issue: 3) Page(s):354 -362 Date of Publication: June 2012.

APPENDIX

SOURCE CODE:

```
clc;
clear all;
close all;
%%
delete(imaqfind)
vid=videoinput('winvideo',1);
triggerconfig(vid,'manual');
set(vid,'FramesPerTrigger',1 );
set(vid,'TriggerRepeat', Inf);
% start(vid);

% View the default color space used for the data — The value of the
ReturnedColorSpace property indicates the color space of the image data.
color_spec=vid.ReturnedColorSpace;

% Modify the color space used for the data — To change the color space of the
returned image data, set the value of the ReturnedColorSpace property.
if ~strcmp(color_spec,'rgb')
    set(vid,'ReturnedColorSpace','rgb');
end

start(vid)

% Create a detector object
```

```

faceDetector = vision.CascadeObjectDetector;
faceDetectorLeye = vision.CascadeObjectDetector('EyePairBig');
faceDetectorM = vision.CascadeObjectDetector('Mouth');

load ix

for ii = 1:100
    trigger(vid);
    im=getdata(vid,1); % Get the frame in im
    imshow(im)

    subplot(3,4,[1 2 5 6 9 10]);
    imshow(im)

    % Detect faces
    bbox = step(faceDetector, im);

    if ~isempty(bbox);
        bbox = bbox(1,:);

        % Plot box
        rectangle('Position',bbox,'edgecolor','r');

        lc = imcrop(im,bbox);
        subplot(3,4,[3 4]);
        imshow(lc)

        bboxeye = step(faceDetectorLeye, lc);

        if ~isempty(bboxeye);
            bboxeye = bboxeye(1,:);

```

```

Eeye = imcrop(Ic,bboxeye);

% Plot box
rectangle('Position',bboxeye,'edgecolor','y');

else
    disp('Eyes not detected')
end

if isempty(bboxeye)
    continue;
end

Ic(1:(bboxeye(2)+2*bboxeye(4)),:,:) = 0;

% Detect Mouth
bboxM = step(faceDetectorM, Ic);

if ~isempty(bboxM);
    bboxMtemp = bboxM;

    if ~isempty(bboxMtemp)

        bboxM = bboxMtemp(1,:);
        Emouth = imcrop(Ic,bboxM);

        % Plot box
        rectangle('Position',bboxM,'edgecolor','y');

    else
        disp('Mouth not detected')
    end
end

```

```

        end
    else
        disp('Mouth not detected')
    end

    [nre nce k ] = size(Eeye);

    % Divide into two parts
    Leye = Eeye(:,1:round(nce/2),:);
    Reye = Eeye(:,round(nce/2+1):end,:);

    subplot(3,4,7)
    imshow(Leye);
    subplot(3,4,8)
    imshow(Reye);
    subplot(3,4,[11,12]);
    imshow(Emouth);

    Leye = rgb2gray(Leye);
    Reye = rgb2gray(Reye);
    Emouth = rgb2gray(Emouth);

    imwrite(Leye,['Database/LE/' num2str(ix) '.bmp']);
    imwrite(Reye,['Database/RE/' num2str(ix) '.bmp']);
    imwrite(Emouth,['Database/M/' num2str(ix) '.bmp']);
    ix = ix+1;
    save ix ix
    pause(0.00005)
end

```

end

clc;

clear all;

close all;

%%

delete(imaqfind)

vid=videoinput('winvideo',1);

triggerconfig(vid,'manual');

set(vid,'FramesPerTrigger',1);

set(vid,'TriggerRepeat', Inf);

% start(vid);

% View the default color space used for the data — The value of the
ReturnedColorSpace property indicates the color space of the image data.

color_spec=vid.ReturnedColorSpace;

% Modify the color space used for the data — To change the color space of the
returned image data, set the value of the ReturnedColorSpace property.

if ~strcmp(color_spec,'rgb')

 set(vid,'ReturnedColorSpace','rgb');

end

start(vid)

% Create a detector object

faceDetector = vision.CascadeObjectDetector;

faceDetectorLeye = vision.CascadeObjectDetector('EyePairBig');

```
faceDetectorM = vision.CascadeObjectDetector('Mouth');
```

```
for ii = 1:500
```

```
    trigger(vid);
```

```
    im=getdata(vid,1); % Get the frame in im
```

```
    imshow(im)
```

```
    subplot(3,4,[1 2 5 6 9 10]);
```

```
    imshow(im)
```

```
    % Detect faces
```

```
    bbox = step(faceDetector, im);
```

```
    if ~isempty(bbox);
```

```
        bbox = bbox(1,:);
```

```
    % Plot box
```

```
    rectangle('Position',bbox,'edgecolor','r');
```

```
    lc = imcrop(im,bbox);
```

```
    subplot(3,4,[3 4]);
```

```
    imshow(lc)
```

```
    bboxeye = step(faceDetectorLeye, lc);
```

```
    if ~isempty(bboxeye);
```

```
        bboxeye = bboxeye(1,:);
```

```
        Eeye = imcrop(lc,bboxeye);
```

```

    % Plot box
    rectangle('Position',bboxeye,'edgecolor','y');
else
    msgbox('Alert!... Sleeping Condition')
end

if isempty(bboxeye)
    continue;
end
lc(1:bboxeye(2)+2*bboxeye(4),:,:) = 0;

% Detect Mouth
bboxM = step(faceDetectorM, lc);

if ~isempty(bboxM);
    bboxMtemp = bboxM;

    if ~isempty(bboxMtemp)

        bboxM = bboxMtemp(1,:);
        Emouth = imcrop(lc,bboxM);

        % Plot box
        rectangle('Position',bboxM,'edgecolor','y');
    else
        msgbox('50% Sleeping Condition')
    end
end
else

```

```

        msgbox('Keep focus the camera')
    end

    [nre nce k ] = size(Eeye);

    % Divide into two parts
    Leye = Eeye(:,1:round(nce/2),:);
    Reye = Eeye(:,round(nce/2+1):end,:);

%     Emouth
    subplot(3,4,7)
    imshow(Leye);
    subplot(3,4,8)
    imshow(Reye);
    subplot(3,4,[11,12]);
    imshow(Emouth);
    pause(0.00005)
end
end
clc;
clear all;
close all;
%%
cl = {'open','close'};
% Initialise empty cell structure
DBL = cell(1,1);
DBR = cell(1,1);

```

```

DBM = cell(1,1);
dim = [30 60;
      30 60
      40 65];
% Left eye
disp('Training left eye');
for ii = 1:2
    disp(cl{ii})
    fpath = ['Database/LE/' cl{ii} '/*.bmp'];
    D = dir(fpath);    for kk = 1:length(D)
    impath = ['Database/LE/' cl{ii} '/' D(kk).name];

    I = imread(impath);
    % Resize to standard size
    Is = imresize(I,[dim(1,1) dim(1,2)]);
%    [nr nc] = size(I);
    % Save to database
    DBL{ii,kk} = Is;
    imshow(Is);
    pause(0.01)
    end
end
% Right Eye
disp('Training Right eye');
for ii = 1:2
    disp(cl{ii})

```

```
fpath = ['Database/RE/' cl{ii} '/*.bmp'];  
pause(0.01)  
end  
end
```

THANK YOU