**CONTENTS**

**1. INTRODUCTION 1**

[1.1 Existing System](#_Toc421353476) 1

[1.2 Proposed System 5](#_Toc421353477)

[1.3 Objectives 6](#_Toc421353478)

[1.4 Organization Of The Report 6](#_Toc421353479)

**2. LITERATURE SURVEY**

[2.1 Drowsiness Detection System For An Intelligent Vehicle 7](#_Toc421353481)

[2.2 Real Time Detection Of Drowsiness Of A Driver 8](#_Toc421353482)

**3.** [**SYSTEM REQUIREMENTS**](#_Toc421353483)

[3.1 Hardware Requirements 11](#_Toc421353484)

[3.2 Software Requirements 11](#_Toc421353485)

[3.3 Matlab 11](#_Toc421353486)

[3.4 Matlab Commands 12](#_Toc421353487)

[3.5 System Components 13](#_Toc421353488)

[3.5.1 Camera 13](#_Toc421353489)

**4.** [**ARCHITECTURE**](#_Toc421353490)

[4.1 Problem Definition 14](#_Toc421353491)

[4.2 Proposed Architecture 14](#_Toc421353492)

**5. DESIGN**

5.1 [Flowchart 16](#_Toc421353494)

[5.2 Data Flow Diagram 16](#_Toc421353495)

[5.3 Sequence Diagram 19](#_Toc421353496)

[5.4 Use Case Diagram 20](#_Toc421353497)

[5.5 Class Diagram 22](#_Toc421353498)

**6. SYSTEM IMPLEMENTATION**

[6.1 Segmentation Of Face 24](#_Toc421353500)

[6.1.1 Histogram 25](#_Toc421353501)

[6.1.2 Ycbcr Color Space 26](#_Toc421353502)

[6.1.3 Algorithm For Converting An RGB Image To Ycbcr Image 28](#_Toc421353503)

[6.2 Detection Of Eyes Condition 30](#_Toc421353504)

[6.2.1 Algorithm for Sobel edge detection 31](#_Toc421353505)

[6.2.2 Eye Template generation process 33](#_Toc421353506)

[6.3 Yawning Detection 35](#_Toc421353507)

[6.3.1 Algorithm for K-means clustering 38](#_Toc421353508)

[6.4 Coding 39](#_Toc421353509)

**7. IMPLEMENTATION**

[7.1 Software Testing 54](#_Toc421353511)

[7.2 Software Testing Types 54](#_Toc421353512)

[7.2.1 Black Box Testing 55](#_Toc421353513)

[7.2.2 White Box Testing 56](#_Toc421353514)

[7.3 A list of sanity Test Cases 56](#_Toc421353515)

**8. RESULTS**

[8.1 Result Analysis Table 59](#_Toc421353517)

[8.2 Final outcome 59](#_Toc421353518)

**9. CONCLUSION AND FUTURE WORK**

[9.1 Conclusion 62](#_Toc421353520)

[9.2 Future Enhancement 62](#_Toc421353521)

**10.BIBLIOGRAPHY**

**LIST OF FIGURES**

4.2.1 System Architecture 6

5.1 Flow Chart of Proposed System 17

5.2.1 Capturing Video Sequences 18 5.2.2 Monitoring Videos 19

5.2.3 Data Flow Diagram 20

5.3.1 Sequence Diagram for Proposed System 21

5.4.1 Use Case Diagram for Proposed System 22 5.5.1 Class Diagram for Proposed System 23

6.1.1.1 Digital Image 26

6.1.1.2 Histogram of an Image 26

6.1.2.1 Sample of Skin Image 28

6.1.3.1 Face Detection Process 29

6.2.2.1 Eye Template Generation Procedure 34

6.3.2 Yawning Detection 37

**LIST OF TABLES**

Table 6.2.2.1 Eye State and Features 34

Table 7.3 Test Cases for the Applications 57

**INTRODUCTION**

Each year hundreds of people lose their lives due to traffic accidents around the world. Unfortunately Iran ranks first in the world in terms of road fatalities and each year approximately thirty thousands of fellow countrymen lose their lives in these events. The role of human factor in accidents cannot be ruled out. According to national statistics in 90 to 95 percent of car accidents in Iran, human factor plays a pivotal role. In general, the driver fatigue accounts for 25 percent of accidents and approximately 60 percent of road accidents result in death or serious injury. In a study by the National Transportation Research Institute (NTSRB) in which 107 random car accidents had been selected, fatigue accounted for 58% of the all accidents. A main cause of fatigue is sleeplessness or insomnia. Drivers' drowsiness is a major contributing factor in severe road accidents that claims thousands of lives every year. According to accident statistics presented by Oklahoma Transportation Institute, which showed 22 percent of all accidents were due to driver’s drowsiness and fatigue, Bittner at al. (2000) proposed an electronic system to alarm drivers.

In recent years, the use of intelligent systems in cars has developed considerably. These systems use wireless sensor networks to monitor and transmit the condition of the car and the driver. Smart cars which use software techniques to control engine speed, steering, transmission, brake etc. has improved the quality of driving drastically. Ad hoc networks were the first systems to develop the automatic navigation in cars. A noticeable weakness of these systems is that their response to environmental changes is not real x`time. It is especially important in driving where time is a critical factor in driver's decision. On the other hand, another method to check the driver fatigue is monitoring the physical condition and facial expressions of the drivers, which wireless sensor networks are unable to process and transmit these information with adequate precision. Driver fatigue is a significant factor in a large number of vehicle accidents. Recent statistics estimate that annually 1,200 deaths and 76,000 injuries can be attributed to fatigue related

Crashes.

**LITERATURE SURVEY**

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then next steps are to determine which operating system and programming language can be used for developing the tool. Once the programmer starts building the tools the programmer need lot of external support. This support can be obtained from senior programmer, from books and from website. Before building the system the above consideration are taken into account for developing the proposed system.

## 2.1 Existing system

In the past few years, many researchers have been working on the development of safety systems using the different techniques. The most accurate techniques are based on physiological measures like brain waves, heart rate, pulse rate, respiration, etc. However, these techniques are intrusive since they require electrodes to be attached to the drivers, causing annoyance to them. A representative project in this line is the MIT Smart Car where several sensors (electrocardiogram, electromyogram, respiration, and skin conductance) are embedded in a car and visual information for sensor confirmation are used. In the advanced safety vehicle (ASV) project conducted by Toyota the driver must wear a wristband in order to measure his heart rate. Others techniques monitor eyes and gaze movements using a helmet or special contact lenses. These techniques, though less intrusive, are still not acceptable in practice.

## 2.1.1 Features

**Vehicle based measures:**

Deviation from the lane position, Steering wheel angle sensor.

**Physiological measures:**

Statistical & energy features derived from ECG.

## 2.1.2 Drawbacks

**Vehicle based measures:**

Too much dependent on the vehicle’s movements.

**Physiological measures:**

The measurement of raw physiological signals is always prone to noise and artifacts due to the movement that is involved with driving.

## 2.2 Problem Statement

To design a monitoring system using Matlab which processes the data to indicate the current driving aptitude of the driver and warning alarm is raised if driver is fatigue.

## 2.3 Proposed system

The proposed algorithm conducts the detection process by recording the video sequence of the drivers and image processing techniques. The system consists of four well-defined phases, namely the face detection, eye tracking, yawning detection and detection of head lowering.

The sequences of images from the camera are fed to the system. Initially, the system doesn’t know the initial position of the face. The system grabs the first image and tries to find the face region in the image using the skin color model. Due to unfavorable lighting conditions or initial head orientation of the driver, the localization might fail. So the system grabs another frame and repeats the same process until the face region is detected with certainty.

## SYSTEM REQUIREMENTS

## Functional Requirements:

## System should alert the driver by making an alarming sound.

## Non Functional Requirements:

## System should be able to extract undistorted frames from the video in real time.

## System should detect eye iris and sclera if the eyes are open from the face top precisely and conclude whether the eyes are in open or closed state within each frame.

## System should not process the images when the car is not in motion which can be computed with the help of GPS.

## System should analyze the driver’s eye blinking rate accurately and its duration will be computed and the pattern will be noted. If the data in the pattern crosses a threshold limit, the system should alert the driver.

## Domain Requirements:

**DESIGN**

## 

## 4.1 Proposed Architecture

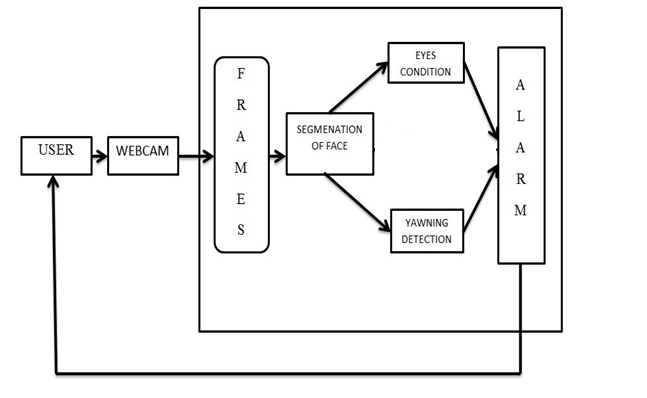
The general diagram of system has been shown in Fig. 4.1. As it can be seen, the image received from camera is sent to central processor to be processed and then it will operate considering condition of drivers' face.

Fig 4.1 System Architecture

**Description of Modules:**

**Segmentation of face:**

The face is segmented from the input image that is initially whatever the video that is recorded by the camera will be fragmented into the frames and this frames will be given as inputs for segmenting the face.

**Eyes condition:**

The position of the driver's eye is determined by using appropriate threshold. In this work, edge detection of the eyes region is considered.

**Yawning Detection:**

Among clustering methods used in segmentation of various parts of the image, the mean-based clustering was utilized for yawning detection. The objective function was to obtain then minimum distance between the classes, or basically between the image pixels.

**4.2 Flowchart**

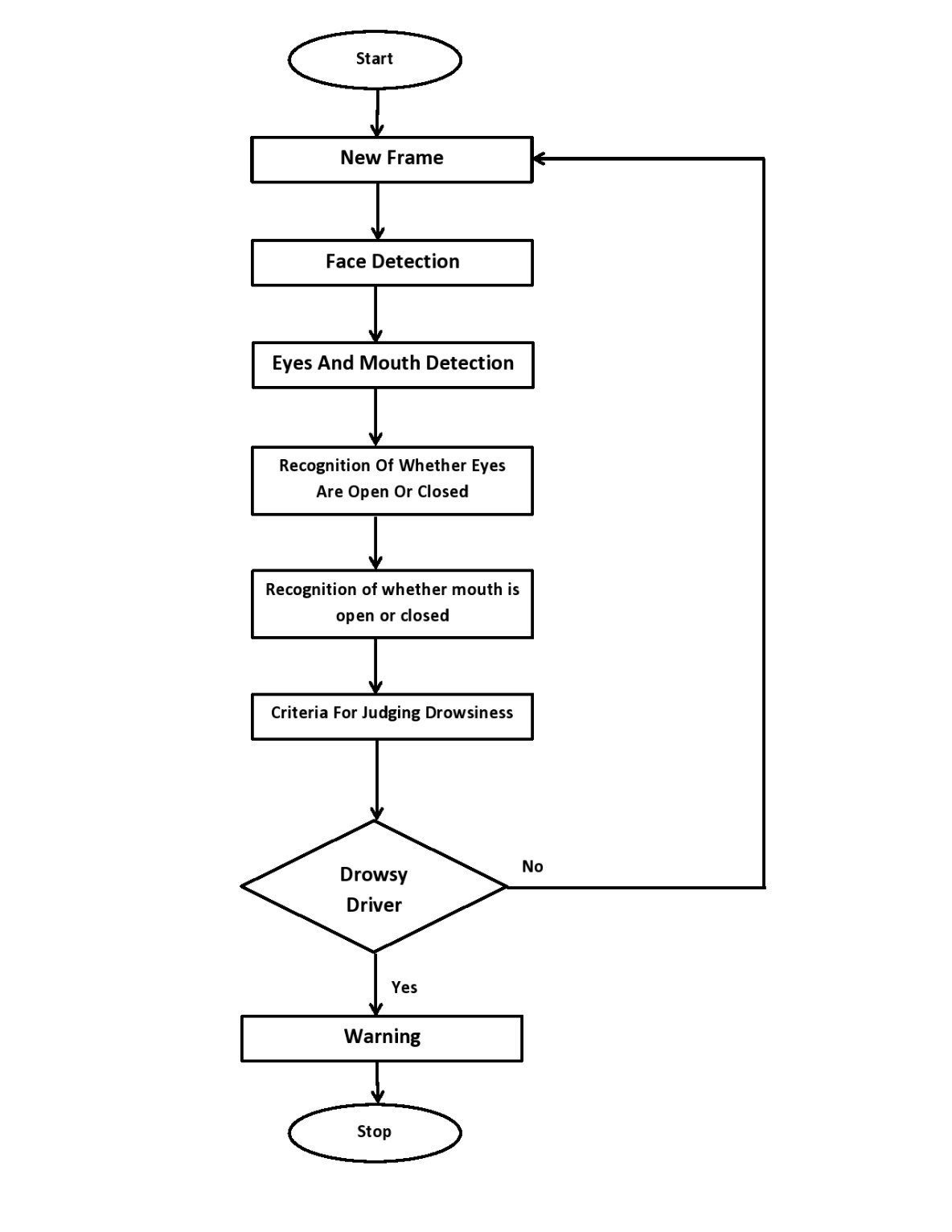


Fig 4.2 Flow Chart of Proposed System

## 4.3 Use case diagram

Use case diagram is a representation of a user's interaction with the system and depicting the specifications of a [use case](http://en.wikipedia.org/wiki/Use_Case). A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual [use case](http://en.wikipedia.org/wiki/Use_Case) and will often be accompanied by other types of diagrams as well.

A use case defines the interactions between external actors and the system under consideration to accomplish a goal. Actors must be able to make decisions, but need not be human: "An actor might be a person, a company or organization, a computer program or computer system hardware, software, or both. Actors are always [stakeholders](http://en.wikipedia.org/wiki/Project_stakeholder), but not all stakeholders are actors, since they "never interact directly with the system, even though they have the right to care how the system behaves”. For example, "the owners of the system, the company's board of directors, and regulatory bodies such as the Internal Revenue Service and the Department of Insurance" could all be stakeholders but are unlikely to be actors.

In our proposed system sequence diagram there are two actors i.e. the user and system were the camera monitors users face and system records the video and creates images for segmentation of face. Segmentation of face is done in order to extract only the eye and mouth region and discard the surrounding region which we are not interested in. Then the conditions for fatigue and non-fatigue are checked. If fatigue is detected then alarm is generated, if no fatigue is detected then no alarm is generated.



Fig 4.3 Use Case Diagram for Proposed System

## 4.4 Sequence diagram

A sequence diagram is an [interaction diagram](http://en.wikipedia.org/wiki/Interaction_diagram) that shows how processes operate with one another and in what order. It is a construct of a [Message Sequence Chart](http://en.wikipedia.org/wiki/Message_Sequence_Chart). A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.



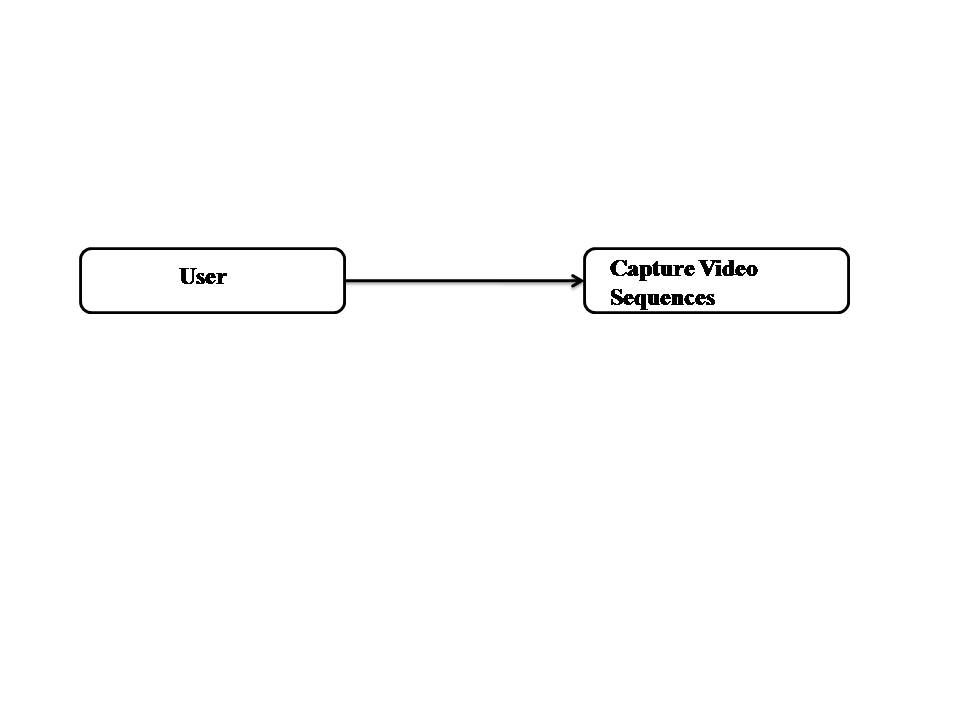
## Fig 4.4 Sequence Diagram of Proposed System

## 4.5 Data Flow Diagram

A data flow diagram (DFD) is a graphical representation of the flow of data through information system, modeling its process aspects. They can also be used for visualization of data processing.

A DFD shows what kinds of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel (which is shown on a [flowchart](http://en.wikipedia.org/wiki/Flowchart)).

**Level 0:** In this level, the person's face is filmed by a camera in the first step by receiving15fps video sequence. Four different video sequences captures image frames.

****

**Fig 4.5.1 Recording driver’s face frames**

**Level 1:**

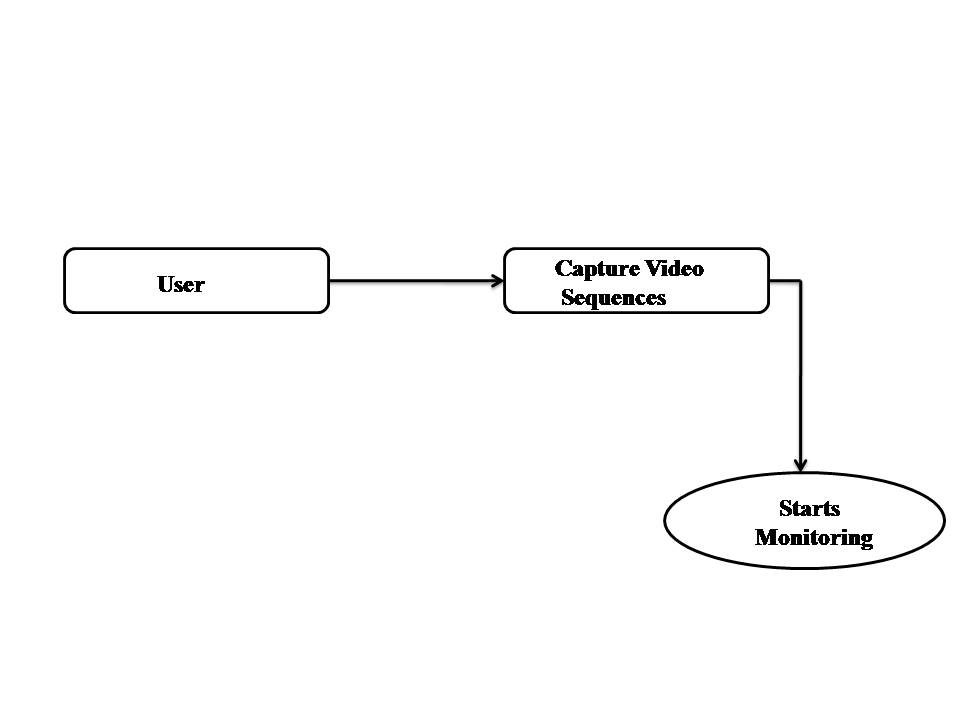
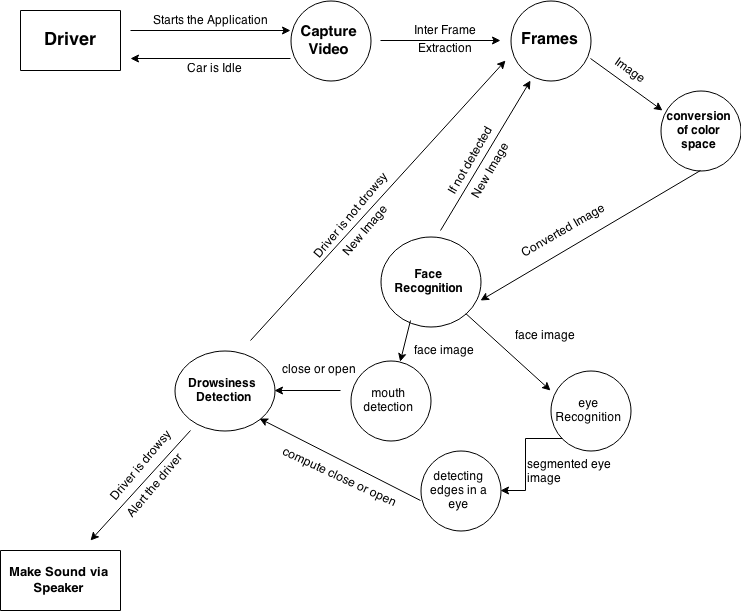
****

Fig 4.5.2 Detection of various facial features of driver

After the video is captured , the position of the eyes and mouth are detected. The eye movements and head movements are tracked.

**Level 2:**

Here, after the monitoring, if the eye condition of the driver is not proper or, he is yawning or his head is lowered due to drowsiness or fatigueness, an alarm is generated to alert the driver. The camera is positioned so that we can monitor the driver’s head and especially the eyes. We want to measure the movement of the head, eyes and eyelids with the aid of the camera. The camera is connected directly to the processor’s video input, and it must meet certain technical requirements including for example adequate resolution, frame rate.

Fig 4.5.3 Data Flow Diagram of the system

## 

# SYSTEM IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned into a working system. Thus it can be considered as the critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective.

The implement stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve change over and evaluation of change over methods.

Here the proposed system includes four modules they are as follows

* Segmentation of face
* Detection of eyes condition
* Yawning detection

## 5.1 Segmentation of face

This is the very first module in which the face is segmented from the input image that is initially whatever the video that is recorded by the camera will be fragmented into the frames and then into the image, this image will be given as input for segmenting the face.

The partial segmentation of the image by selecting the appropriate threshold is based on dividing the image into the background and foreground classes. Thresholding is primarily concerned with selecting an appropriate threshold according to image histogram. That is, the value of thresholding or border as the brightness intensity is considered as the basis of the division and the brightness intensities greater and less than threshold is equal to 1 and zero respectively.

The purpose of face detection is to minimize the error rate in identifying facial expressions. The importance of this part is to measure the position of the eyes, the mouth and the head.

### 5.1.1 Histogram

A histogram is a graphical representation of the [distribution](http://en.wikipedia.org/wiki/Frequency_distribution) of data.

There are two types of histogram they are as follows

* Image histogram
* Color histogram

Image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value. Image histograms are present on many modern digital cameras. Horizontal axis of the graph represents the tonal variations, while the vertical axis represents the number of pixels in that particular tone. In the field of computer vision image histograms can be useful tool for thresholding. This threshold value can be used for edge detection, image segmentation and co-occurrence matrix.

Representing an histogram of an image in matlab

A=imread(‘sample.jpg’);

hist(A);

Here imread() reads a grayscale or color image from the file specified by the string filename. If the file is not in the current folder, or in a folder on the MATLABpath, specify the full pathname. Basically here we are converting an image to histogram only for getting the threshold values in order to separate the foreground and background class.

|  |  |
| --- | --- |
| C:\Users\Mai\Documents\Bluetooth Folder\Capture.PNG  Fig 6.1.1.1 Digital Image | C:\Users\Mai\Documents\Bluetooth Folder\1.PNG  Fig 6.1.1.2 Histogram Of An Image |

Here the below figure shows the image in RGB color space and its histogram. Here we are finding histogram of the image in order to find the threshold value of the image.Using this threshold value we are classifying foreground and background image.

### 5.1.2 YCbCr Color Space

Initially the camera will record the video of a person who is driving so that all the images in that recorded video will be in the RGB color space that includes driver face along

with surrounding area in the vehicle.

The purpose of face detection is to minimize the error rate in identifying facial expressions. The importance of this part is to measure the position of the eyes, the mouth and the head.

YCbCr, Y′CbCr, or Y Pb/Cb Pr/Cr, also written as YCBCR or Y′CBCR, is a family of [color space](http://en.wikipedia.org/wiki/Color_space)  used as a part of the [color image pipeline](http://en.wikipedia.org/wiki/Color_image_pipeline) in [video](http://en.wikipedia.org/wiki/Video) and [digital photography](http://en.wikipedia.org/wiki/Digital_photography) systems. Y′ is the [luma](http://en.wikipedia.org/wiki/Luma_(video)) component and CB and CR are the blue-difference and red-difference chroma components. Y′ (with prime) is distinguished from Y, which is [luminance](http://en.wikipedia.org/wiki/Luminance_(relative)), meaning that light intensity is nonlinearly encoded based on [gamma corrected](http://en.wikipedia.org/wiki/Gamma_correction) [RGB](http://en.wikipedia.org/wiki/RGB) primaries.

Y′CbCr is not an [absolute color space](http://en.wikipedia.org/wiki/Absolute_color_space); rather, it is a way of encoding [RGB](http://en.wikipedia.org/wiki/RGB) information. The actual color displayed depends on the actual [RGB](http://en.wikipedia.org/wiki/RGB) primaries used to display the signal. Therefore a value expressed as Y′CbCr is predictable only if standard RGB primary chromaticities are used.

The first step in the face detection algorithm is using skin segmentation to reject as much non- image based on skin color converting the RGB picture to YCbCr space or to HSV space. An YCbCr space segments the image into a luminosity component and color components. The main advantage of converting the image to the YCbCr domain is that influence of luminosity can be removed during our image processing. In the RGB domain, each component of the picture (red, green and blue) has a different brightness. However, in the YCbCr domain all information about the brightness is given by the Y-component, since the Cb (blue) and Cr (red) components are independent from the luminosity.

There are many ways of segmenting indication on whether a pixel is part of the skin or not. Background and faces can be distinguished by applying maximum and minimum threshold values for both Cb and Cr components.

Considering there is no standard color-image database which can assess the faces in images, we set up the color-image database with 500 skin images. There are some sample images that are shown in the following figure.

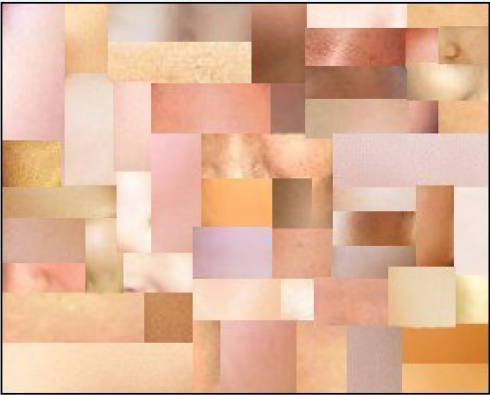


Fig 5.1.2 Sample of Skin Images

### 5.1.3 Steps for converting an RGB image to YCbCr image

Formula used for convert an RGB pixel to YCbCr pixel is as follows

Y=0.299R+0.5879G+0.114B

Cb=-0.169R-0.331G+0.5B

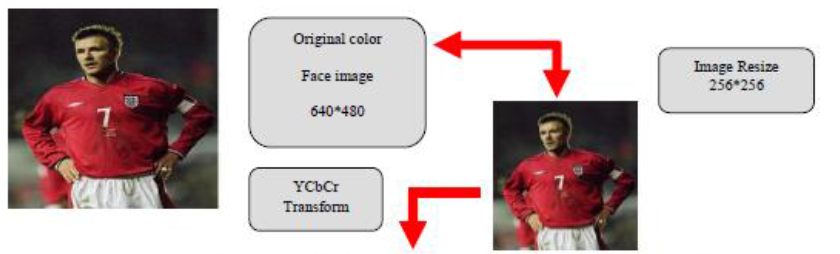
Cr=0.5R-0.419G-0.081B

Step 1: Reading an input image

RGB= imread('sample.jpg');

Step 2: Converting an RGB image to YCbCr image

YCBCR = rgb2ycbcr(RGB)





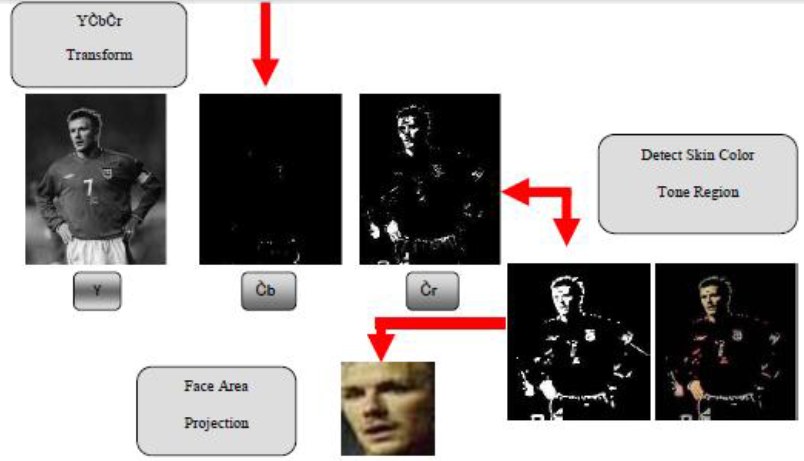


Fig 5.1.3 Face Detection Process

## 5.2 Detection of Eyes condition

Important factor which helps detect driver fatigue is the state of eyes, i.e. whether they are open or closed. In the state of fatigue, eyelid muscles subconsciously attempt to accelerate the process of going to sleep. Using this property, determining whether eyes are open or closed is done by relying on the difference of brightness intensity of the pupil in the image and its symmetry. Locating the position of the eye in the frame taken from the drivers face is difficult. The position of the eye can be identified by drawing on geometry properties and symmetry. Edging is concerned with locating the position of areas or pixels where the brightness intensity has considerably increased. One of the effective operators for edge detection is the Sobel operator.

The position of the driver's eye is determined by using appropriate threshold. These two areas are separated using edge detection and in accordance with as well as the symmetrical properties of the eye, the gravity center of the eye is determined. Finally the pupil is identified. If eyes are open then it is treated as the normal state during which the alarm is not set off. If eyes are closed then it is treated as the fatigue state during which the alarm is set on.

Edge detection is the process of localizing pixel intensity transitions. The edge detection has been used by object recognition, target tracking, segmentation, and etc. Therefore, the edge detection is one of the most important parts of image processing. There mainly exist several edge detection methods like Sobel .These methods have been proposed for detecting transitions in images. Early methods determined the best gradient operator to detect sharp intensity variations. Commonly used method for detecting edges is to apply derivative operators on images. Computing the gradient in several directions and combining the result of each gradient. The value of the gradient magnitude and orientation is estimated using two differentiation masks.

In this work, Sobel which is an edge detection method is considered. Because of the simplicity and common uses, this method is preferred by the others methods in this work. The Sobel edge detector uses two masks, one vertical and one horizontal. These masks are generally used 3×3 matrices. Especially, the matrices which have 3×3 dimensions are used in matlab (edge.m). The masks of the Sobel edge detection are extended to 5×5 dimensions are constructed in this work. A matlab function, called as Sobel5×5 is developed by using these new matrices toolboxes.

### 5.2.1 Algorithm for Sobel edge detection

Step 1: Accept the input image.

imread(input image)

Reads a rgb color image from the file specified by the string filename. If the file is not in the current folder, or in a folder on the MATLAB path, specify the full pathname

Step 2: Specifies the Sobel method and the dimension of image

BW = edge(I,'sobel')

Sobel method takes input image as its input, and returns a binary image BW of the same size with 1's where the function finds edges in I and 0's elsewhere. And this syntax also specifies the dimension of the image to be 1 dimension.

Step 3: Reading the BW image

imread(BW)

It reads a grayscale image from the file specified by the string filename. If the file is not in the

current folder or in a folder on the MATLAB path, specify the full pathname.

Step 4: Specifies the threshold for the Sobel method

BW = edge(I,'s]’;obel',thresh)

It specifies the threshold for the Sobel method. Edge ignores all edges that are not stronger than thresh. If you do not specify thresh, or if thresh is empty ([]), edge chooses the value automatically. Threshold values ranging from 365 to 535

Step 5: Specifies the direction of detection for the Sobel method

BW = edge (I,'sobel', thresh, direction)

It specifies the direction of detection for the Sobel method. Direction is a string specifying whether to look for 'horizontal' or 'vertical' edges i.e.; gv and gh or 'both' (the default).

Step 6: Returns the threshold value.

[BW, thresh] = edge (I,'sobel',...)

Provides an optional string input. String 'no thinning' speeds up the operation of the algorithm by skipping the additional edge thinning stage. By default, or when 'thinning' string is specified, the algorithm applies edge thinning.

Step 7: Two masks are used for having the maximum edge at vertical and horizontal level i.e; gv and gh.

Mask along horizontal direction i.e. gh

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A11 | A12 | A13 | …… | A1K | + | M11 | M12 | M13 | **→** | B11 | B12 | B13 | ….. | B1K |
| A21 | A22 | A23 | …… | A2K | M21 | M22 | M23 | B21 | **B22** | B23 | ….. | B2K |
| A31 | A32 | A33 | …… | A3K | M31 | M32 | M33 | B31 | B32 | B33 | ….. | B3K |
| : | : | : | : | : |  | | | | | : | : | : | : | : |

Input image Mask gh output image

B22=(A11\*M11)+(A12\*M12)+(A13\*M13)+(A21\*M21)+(A22\*M22)+(A23\*M23)+(A31\*M3) +(A32\*M32)+(A33\*M33)

Mask along horizontal direction i.e. gv

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A11 | A12 | A13 | …… | A1K | + | M11 | M21 | M31 | **→** | B11 | B12 | B13 | ….. | B1K |
| A21 | A22 | A23 | …… | A2K | M12 | M22 | M32 | B21 | **B22** | B23 | ….. | B2K |
| A31 | A32 | A33 | …… | A3K | M13 | M23 | M33 | B31 | B32 | B33 | ….. | B3K |
| : | : | : | : | : |  | | | | | : | : | : | : | : |

Input image Mask gv output image

Mask along vertical direction i.e. gv

B22=(A11\*M11)+(A12\*M21)+(A13\*M31)+(A21\*M12)+(A22\*M22)+(A23\*M32)+(A31\*M1)+(A32\*M23)+(A33\*M33)

Step 8: Returns vertical and horizontal edge responses to Sobel operators.

[BW,thresh,gv,gh] = edge(I,'sobel',...)

Returns the threshold value of eyes along vertical and horizontal value and the status of the eyes are determined

Step 9: Template matching is done based on status of eyes given from step8.

Step 10: Finally if the fatigue is detected then the alarm is generated to the user.

### 5.2.2 Eye Template generation process

After successful face and facial feature detection, eye state can be determined in every frame using the correlation coefficient template matching method. By considering the diversity to the nearby pixels and the similarity to the eye pixels sufficiently, the specific region of the eyes can be obtained. After this step, Sobel edge detection method is used again to detect the eyes’ precise boundaries. The method has been adopted frequently in the eye detection which mentioned above. Our method starts from left and right side, to find eyes, therefore we can detect the eyes separately. We segment the eyes from the image and use them to generate an eye template, by this means we obtain a rather stable eye template for the status analyzing and reduce the influence of light reflections. The eye template is generated as follows.

C:\Users\Mai\Desktop\facebook\12.PNG**BINARY PROCESSING**

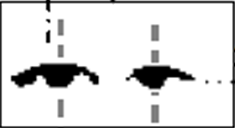
**And Operation**C:\Users\Mai\Desktop\facebook\17.PNG

**Or Operation**C:\Users\Mai\Desktop\facebook\16.PNG



**Eye template**





**Image alignment**

Fig 5.2.2 Eye Template Generation Procedure

To distinguish the driver’s status the eyes’ states should be recognized ahead. There are two factors which can affect the size of the eyes in the frames. On the one hand, human eyes are always different in size. On the other hand, the distance between driver and the camera is the other reason. So we normalize the eye template to a fixed size of 12×30 before feature extraction. For each eye template, eye area, average height of pupil, width to height ratio are the most important features to judge eye’s status which is shown in below table.

**Table 5.2.2 Eye States And Features**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Eye Region | Area(pixel) | Eye Template | Average Height | Ratio |
| Full open | G:\Final img\t1.PNG | 200 | G:\Final img\t4.PNG | 7.6 | 2.8750 |
| Half open | G:\Final img\t2.PNG | 155 | G:\Final img\t5.PNG | 6.8 | 3.0000 |
| Closed | G:\Final img\t3.PNG | 114 | G:\Final img\t6.PNG | 6.0 | 3.1667 |

The eye states can be divided into two types: full open, half open and closed. From the table above we can see that eyes of different states present different features. By analyzing the driver’s eyes states changes while driving we discovered two principles which can indicate driver drowsiness. Firstly, if a driver’s eyes keep closed over 4 consecutive frames it is believed that the driver is drowsy. Secondly, fatigue can be confirmed if a driver’s eyes only change between half open and closed over 8 consecutive frames. The different eye states of full open and half open sometimes cannot be well distinguished which has caused more false judgments and the fast movement of drivers head can result in the driver’s eyes tracking failure. Before the system is put into use we trained it in advance to get different states parameters for the driver aiming at improving the accuracy of the driver’s status analysis.

## 5.3 Yawning Detection

K-means clustering is a partitioning method that treats observations in your data as objects having locations and distances from each other. It partitions the objects into K mutually exclusive clusters, such that objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible. Each cluster is characterized by its centroid, or center point. The function K-means performs K-Means clustering, using an iterative algorithm that assigns objects to clusters so that the sum of distances from each object to its cluster centroid, over all clusters, is a minimum.

With K-means clustering, you must specify the number of clusters that you want to create. We can choose from five different distance measures, depending on the kind of data you are clustering. Each cluster in the partition is defined by its member objects and by its centroid, or center. The centroid for each cluster is the point to which the sum of distances from all objects in that cluster is minimized. K-means computes cluster centroid differently for each distance measure, to minimize the sum with respect to the measure that you specify.

K-means uses an iterative algorithm that minimizes the sum of distances from each object to its cluster centroid, over all clusters. This algorithm moves objects between clusters until the sum cannot be decreased further. The result is a set of clusters that are as compact and well-separated as possible. You can control the details of the minimization using several optional input parameters to K-means, including ones for the initial values of the cluster centroid, and for the maximum number of iterations. First, load the data and call K-means with the desired number of clusters set to 2, and using squared Euclidean distance. To get an idea of how well-separated the resulting clusters are, you can make a silhouette plot. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters. The centroids of each cluster are plotted using circled X's. Three of the points from the lower cluster, plotted with triangles, are very close to points from the upper cluster, plotted with squares. But, in fact, because the upper cluster is so spread out, those three points are closer to the centroid of the lower cluster than to that of the upper cluster, even though they are separated from the bulk of the points in their own cluster by a gap. Because K-means clustering only considers distances, and not densities, this kind of result can occur.

In this paper the other sign of fatigue during driving, which is manifested in a person's face, is frequent yawning that is due to body reflexes when a person is exhausted and about to fall asleep. Various systems have been proposed for measuring yawning some of which are slow and time consuming while others are not very accurate in separating the mouth area at the time of yawning.

An efficient technique is needed that is able to display the changes in face configuration and detect the yawning. Among clustering methods used in segmentation of various parts of the image, the mean-based clustering or K-means was utilized. The objective function was to obtain then minimum distance between the classes, or basically between the image pixels.



**Fig 5.3.1 *Normal Mouth Detection***

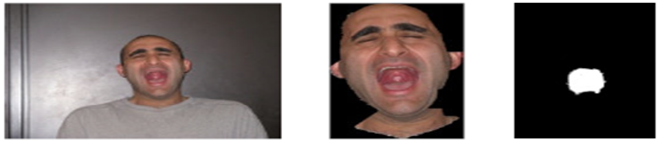


Fig 5.3.2 Yawning Detection

K-means uses a two-phase iterative algorithm to minimize the sum of point-to-centroid distances, summed over all K-clusters:

1. The first phase uses batch updates, where in each iteration consists of reassigning points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroids. This phase occasionally does not converge to solution that is a local minimum, that is, a partition of the data where moving any single point to a different cluster increases the total sum of distances. This is more likely for small data sets. The batch phase is fast, but potentially only approximates a solution as a starting point for the second phase.
2. The second phase uses online updates, where points are individually reassigned if doing so will reduce the sum of distances, and cluster centroid are recomputed after each reassignment. For each of the iterations during the second phase consists of one pass though all the points. The second phase will converge to a local minimum, although there may be other local minima with lower total sum of distances. The problem of finding the global minimum can only be solved in general by an exhaustive (or clever, or lucky) choice of starting points, but using several replicates with random starting points typically results in a solution that is a global minimum.

Once the object gets partitioned into clusters each cluster in the partition is defined by its member objects and by its centroid, or center. The centroid for each cluster is the point to which the sum of distances from all objects in that cluster is minimized. K-means computes cluster centroids differently for each distance measure, to minimize the sum with respect to the measure that you specify. Classification of image pixels is based on the brightness intensity. The frequency of brightness intensity of images (histogram) has been used to select the appropriate number of secretions. Based on this classification, pixels can be divided into a maximum of 255 clusters. Here, the number of proposed clusters has been 5 to 10. Thus, classification of image by K-means algorithm can help select proper threshold and NOT brightness intensity acquired from clustering level of yawning detection in input image of frames. Finally, a large part of the segmented area in the image demonstrates the position of mouth and detects the yawning.

### 5.3.1 Algorithm for K-means clustering

The steps for K-means clustering as given below:

Step 1: Partitions the points in the n-by-p data matrix X into k clusters.

IDX = kmeans(X,k)

Input image is converted into n-by-p data matrix X. The X data matrix is converted into

K-clusters. K-means clustering is a method of vector quantization.

Step 2: Returns the k cluster centroid locations in the k-by-p matrix C.

[IDX,C] = kmeans(X,k)

Where each iteration consists of reassigning points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroid.

Step 3: returns the within-cluster sums of point-to-centroid distances in the 1-by-k vector sumd.

[IDX,C,sumd] = kmeans(X,k)

This phase occasionally does not converge to solution that is a local minimum, that is, a partition of the data where moving any single point to a different cluster increases the total sum of distances. This is more likely for small data sets.

Step 4: Returns distances from each point to every centroid in the n-by-k matrix D.

[IDX,C,sumd,D] = kmeans(X,k)

Points are individually reassigned if doing so will reduce the sum of distances, and cluster centroids are recomputed after each reassignment. Each iteration during the second phase consists of one pass though all the points. The second phase will converge to a local minimum, although there may be other local minima with lower total sum of distances.

Step 5: Enables you to specify optional parameter/value pairs to control the iterative algorithm used by K-means.

[...] =kmeans(...,param1,val1,param2,val2,...)

## 5.4 CODING

clc;

clear all;

close all;

%%

load DB

load svm

cl = {'open','close'};

dim = [30 60;

30 60

40 65];

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');

tic

% Initialise vector

LC = 0; % Left eye closer

RC = 0; % Right eye closer

MC = 0; % Mouth closer

TF = 0; % Total frames

TC = 0; % Total closure

Feature = [];

c1p = 1;

species = 'Non-Fatigue';

for ii = 1:600

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');

S = skin\_seg2(im);

% Segment skin region

bw3 = cat(3,S,S,S);

% Multiply with original image and show the output

Iss = double(im).\*bw3;

Ic = imcrop(im,bbox);

Ic1 = imcrop(Iss,bbox);

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

imshow(uint8(Ic1))

bboxeye = step(faceDetectorLeye, Ic);

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')

continue;

end

else

disp('Mouth not detected')

continue;

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(edge(rgb2gray(Leye),'sobel'));

subplot(3,4,8)

imshow(edge(rgb2gray(Reye),'sobel'));

Emouth3 = Emouth;

Leye = rgb2gray(Leye);

Reye = rgb2gray(Reye);

Emouth = rgb2gray(Emouth);

% K means clustering

X = Emouth(:);

[nr1 nc1 ] = size(Emouth);

cid = kmeans(double(X),2,'emptyaction','drop');

kout = reshape(cid,nr1,nc1);

subplot(3,4,[11,12]);

% Segment

Ism = zeros(nr1,nc1,3);

% Ism(:,:,3) = 255;

% Ism(:,:,3) = 125;

Ism(:,:,3) = 255;

bwm = kout-1;

bwm3 = cat(3,bwm,bwm,bwm);

Ism(logical(bwm3)) = Emouth3(logical(bwm3));

imshow(uint8(Ism));

% Template matching using correlation coefficient

% Left eye

% Resize to standard size

Leye = imresize(Leye,[dim(1,1) dim(1,2)]);

c1 =match\_DB(Leye,DBL);

subplot(3,4,7)

title(cl{c1})

% Right eye

% Resize to standard size

Reye = imresize(Reye,[dim(2,1) dim(2,2)]);

c2 = match\_DB(Reye,DBR);

subplot(3,4,8)

title(cl{c2})

% Mouth

% Resize to standard size

Emouth = imresize(Emouth,[dim(3,1) dim(3,2)]);

c3 = match\_DB(Emouth,DBM);

subplot(3,4,[11,12]);

title(cl{c3})

if c1 == 2

LC = LC+1;

if c1p == 1

TC = TC+1;

end

end

if c2==2

RC = RC+1;

end

if c3 == 1

MC = MC + 1;

end

TF = TF + 1; % Total frames

toc

if toc>8

Feature = [LC/TF RC/TF MC/TF TC]

species = svmclassify(svmStruct,Feature);

tic

% Initialise vector

LC = 0; % Left eye closer

RC = 0; % Right eye closer

MC = 0; % Mouth closer

TF = 0; % Total frames

TC = 0; % Total closure

end

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

if strcmpi(species,'Fatigue')

text(20,20,species,'fontsize',14,'color','r','Fontweight','bold')

beep;

else

text(20,20,species,'fontsize',14,'color','g','Fontweight','bold')

end

c1p = c1;

pause(0.00005)

end

end

**FACE SEGMENTATION**

%%

vid=videoinput('winvideo',1);

triggerconfig(vid,'manual');

set(vid,'FramesPerTrigger',1 );

set(vid,'TriggerRepeat', Inf);

% 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');

Ic = imcrop(im,bbox);

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

imshow(Ic)

bboxeye = step(faceDetectorLeye, Ic);

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

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,:);

% 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

**EYE DETECTION**

bboxeye = step(faceDetectorLeye, Ic);

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;

**Sobel Edge Detection**

% Divide into two parts

Leye = Eeye(:,1:round(nce/2),:);

Reye = Eeye(:,round(nce/2+1):end,:);

subplot(3,4,7)

imshow(edge(rgb2gray(Leye),'sobel'));

subplot(3,4,8)

imshow(edge(rgb2gray(Reye),'sobel'));

**FEATURE EXTRACTION**

load DB

cl = {'open','close'};

dim = [30 60;

30 60

40 65];

delete(imaqfind)

vid=videoinput('winvideo',1);

triggerconfig(vid,'manual');

set(vid,'FramesPerTrigger',1 );

set(vid,'TriggerRepeat', Inf);

% 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');

tic

% Initialise vector

LC = 0; % Left eye closer

RC = 0; % Right eye closer

MC = 0; % Mouth closer

TF = 0; % Total frames

TC = 0; % Total closure

Feature = [];

c1p = 1;

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');

Ic = imcrop(im,bbox);

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

imshow(Ic)

bboxeye = step(faceDetectorLeye, Ic);

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')

continue;

end

else

disp('Mouth not detected')

continue;

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);

% Template matching using correlation coefficient

% Left eye

% Resize to standard size

Leye = imresize(Leye,[dim(1,1) dim(1,2)]);

c1 =match\_DB(Leye,DBL);

subplot(3,4,7)

title(cl{c1})

% Right eye

% Resize to standard size

Reye = imresize(Reye,[dim(2,1) dim(2,2)]);

c2 = match\_DB(Reye,DBR);

subplot(3,4,8)

title(cl{c2})

% Mouth

% Resize to standard size

Emouth = imresize(Emouth,[dim(3,1) dim(3,2)]);

c3 = match\_DB(Emouth,DBM);

subplot(3,4,[11,12]);

title(cl{c3})

if c1 == 2

LC = LC+1;

if c1p == 1

TC = TC+1;

end

end

if c2==2

RC = RC+1;

end

if c3 == 1

MC = MC + 1;

end

TF = TF + 1; % Total frames

if toc>10

Feature = [Feature;LC/TF RC/TF MC/TF TC]

tic

% Initialise vector

LC = 0; % Left eye closer

RC = 0; % Right eye closer

MC = 0; % Mouth closer

TF = 0; % Total frames

TC = 0; % Total closure

end

c1p = c1;

pause(0.00005)

end

end

save FA Feature

**Yawning Detection**

% K means clustering

X = Emouth(:);

[nr1 nc1 ] = size(Emouth);

cid = kmeans(double(X),2,'emptyaction','drop');

kout = reshape(cid,nr1,nc1);

subplot(3,4,[11,12]);

% Segment

Ism = zeros(nr1,nc1,3);

% Ism(:,:,3) = 255;

% Ism(:,:,3) = 125;

Ism(:,:,3) = 255;

bwm = kout-1;

bwm3 = cat(3,bwm,bwm,bwm);

Ism(logical(bwm3)) = Emouth3(logical(bwm3));

imshow(uint8(Ism));

# TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover faults or defects in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user exceptions and does not fail in an unacceptable manner.

A good test case is the one that has a high probability of finding an as yet undiscovered error. A successful test is the one that uncovers a undiscovered error. Testing may be carried out during the implementation phase to verify that the software behaves as intended by its designers and after implementation is complete.

## 6.1 Software Testing

A primary purpose of testing is to detect software failures so that defects may be discovered and corrected. Testing cannot establish that a product functions properly under all conditions but can only establish that it does not function properly under specific conditions. The scope of software testing often includes examination of code as well as execution of that code in various environments and conditions as well as examining the aspects of code: does it do what it is supposed to do and do what it needs to do. In the current culture of software development, a testing organization may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

## 6.2 Software Testing Types

Software testing life cycle is the process that explains the flow of the tests that are to be carried on each step of software testing of the product. The V-model i.e. Verification and validation model is a perfect model which is used in the improvement of the software project. This model contains software development life cycle on one side and software testing life cycle on the other hand side. A checklist for software tester sets a baseline that guides him to carry on the day-to-day activities.

### 6.2.1 Black Box Testing

It explains the process of giving the input to the system and checking the output, without considering how the system generates the output. It is also called as Behaviour Testing.

**Functional Testing**

In this type of testing, the software is tested for the functional requirements. This checks whether the application is behaving according to the specification.

**Performance Testing**

This type of testing checks whether the system is performing properly, according to the user’s requirements. Performance testing depends upon the Load and Stress Testing that is internally or externally applied to the system.

**Integration Testing**

Integration Testing is the phase in [software testing](http://en.wikipedia.org/wiki/Software_testing) in which individual software modules are combined and tested as a group. This mostly focuses in the design and construction of the software architecture. The purpose of integration testing is to verify functional, performance, and reliability [requirements](http://en.wikipedia.org/wiki/Requirement) placed on major design items.

Integration testing is further classified into Bottom-up Integration and Top-Down Integration testing.

**Bottom-up Integration Testing**

It is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested. All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will be formed and can be used for integration testing.

**Top-Down Integration testing**

It is an approach to integrated testing where the top down integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

**System Testing**

System testing is the testing conducted on a complete, integrated system, to evaluate the system’s compliance with the specified requirements. This type of software testing validates that the system meets its functional and non-functional requirements and also intended to test beyond the bounds defined in the software/hardware requirement specifications.

### 6.2.2 White Box Testing

It is the process of giving the input to the system and checking, how the system processes the input, to generate the output. It is mandatory for a tester to have the knowledge of the source code.

**Unit Testing**

This type of testing is done at the developer’s site to check whether a particular piece/unit of code is working fine. Unit testing deals with testing the unit as a whole.

## 6.3 A list of sanity Test Cases

Prior to any testing it’s mandatory to write test cases, so that it helps in

1. Giving an approach, description, pre-conditions to achieve the expected result.
2. Test cases are reusable, helps in regression testing while releasing various versions.

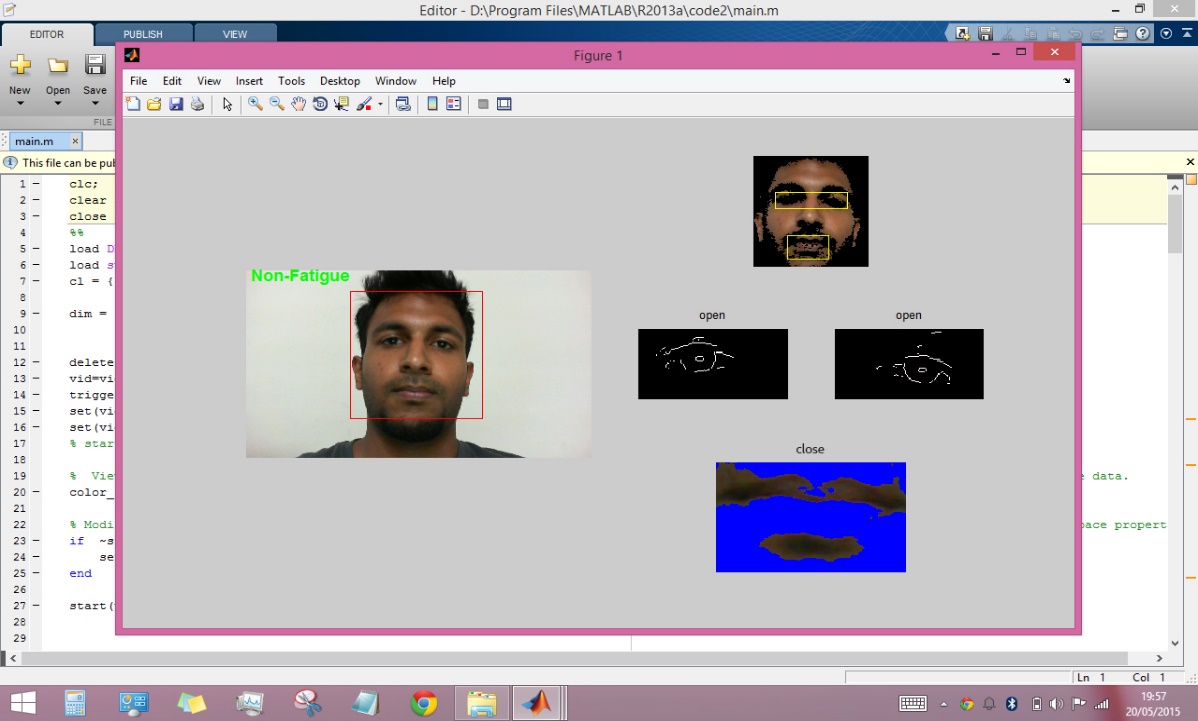
**Table 6.3 Test Cases for the Applications**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test case ID** | **Test Case** | **Input** | **Expected output** | **Action** | **Result** |
| **TID1** | Detecting face | Input Frame | Face detected | Segmentation of face | Pass |
| **TID2** | Detecting eyes | Segmentation of face | Eyes detected | Edges of the eyes | Pass |
| **TID3** | Detection of mouth | Segmentation of face | Detected mouth | Clustered mouth with large hole | Pass |
| **TID4** | Driver’s fatigue condition | Eyes open and Mouth closed | Non Fatigue | No Alarm | Pass |
| **TID5** | Driver’s fatigue condition | Eyes closed and Mouth closed | Fatigue | Alarm  generated | Pass |
| **TID6** | Driver’s fatigue condition | Eyes closed and Mouth opened | Fatigue | Alarm  Generated | Pass |
| **TID7** | Driver’s fatigue condition | Eyes opened and Mouth opened | Fatigue | Alarm  Generated | Pass |

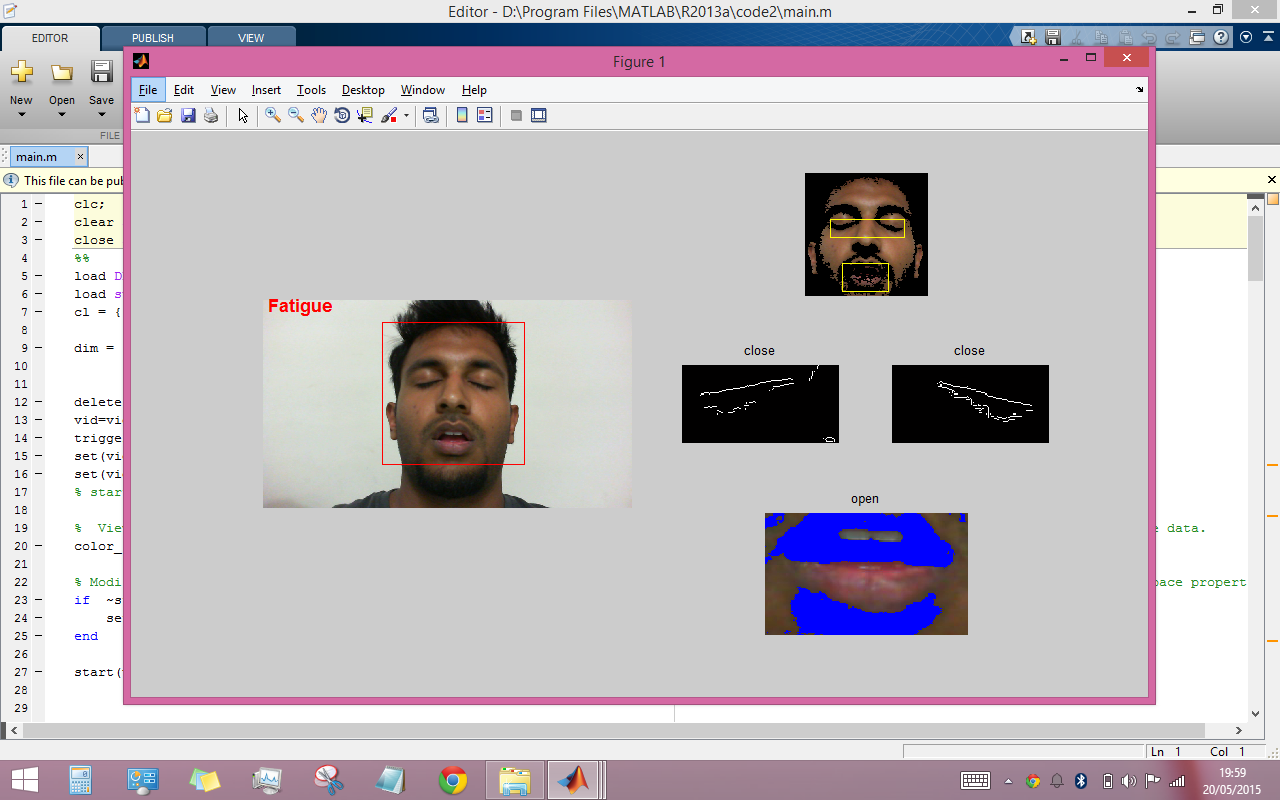
## 6.4 Final outcome

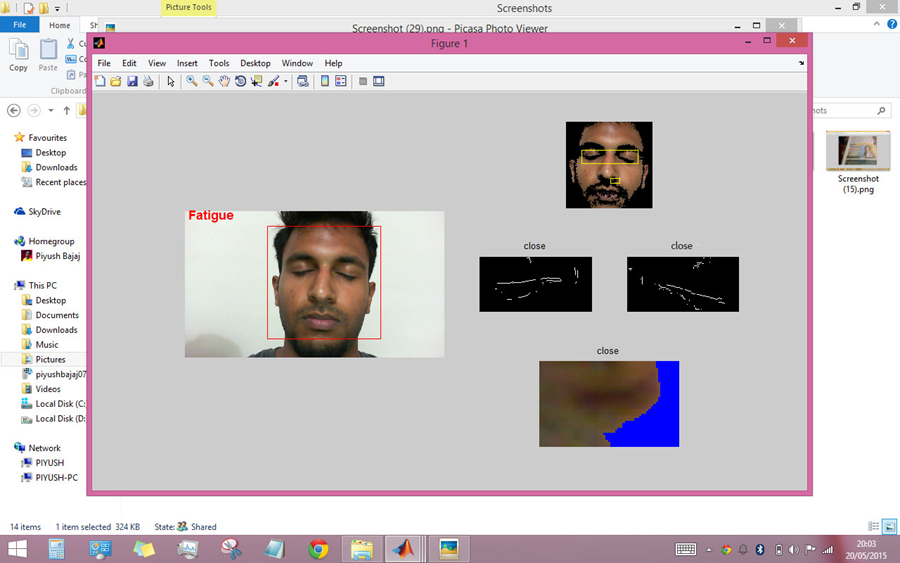
When the person’s face is captured by the camera first it will extract the background and foreground classes then for the extracted face part segmentation is done. By observing the eye and mouth state it will check for driver fatigue.

Here we can observe that eye is opened and mouth is closed so there is no sign of fatigue detected. Hence Alarm is not generated.

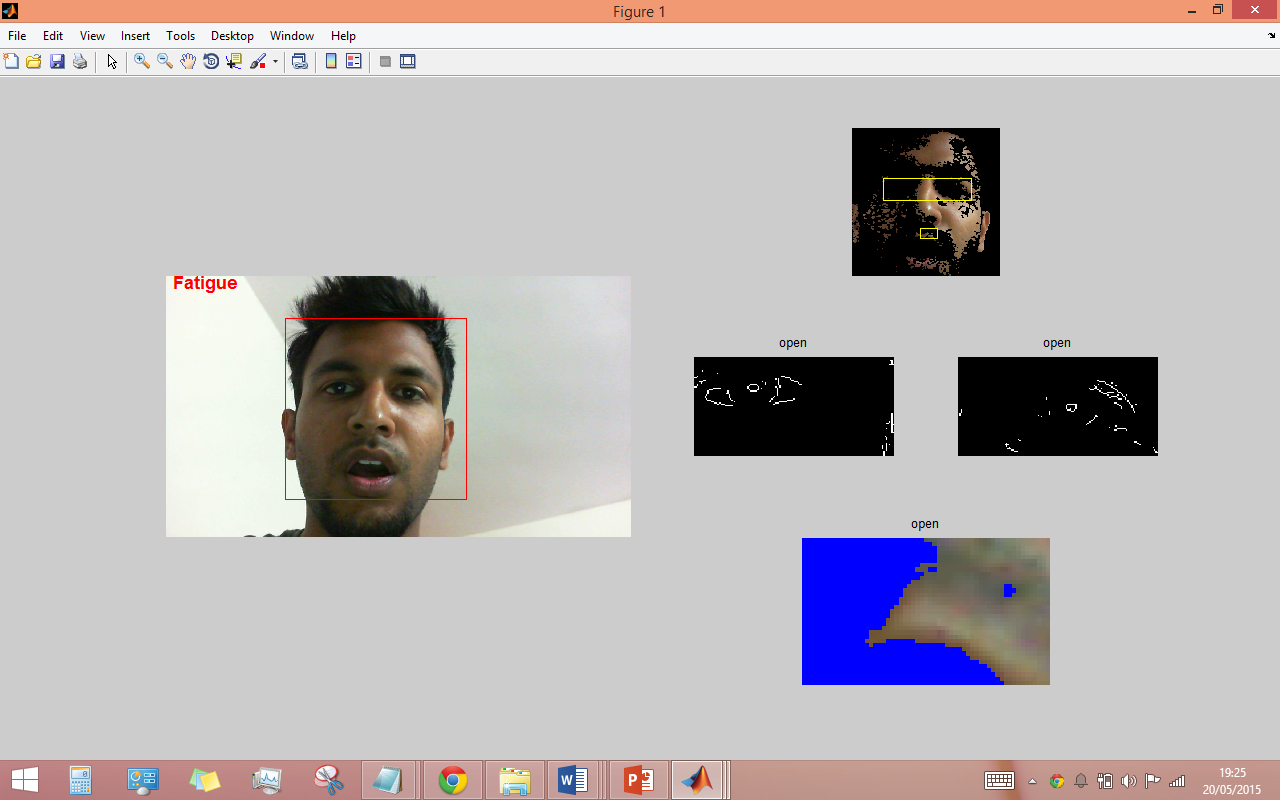
****

**Snapshot for driver Non-Fatigue**

 **Snapshot for driver Fatigue**

****

**Snapshot for driver Fatigue**



**Snapshot for driver Fatigue**

# CONCLUSION AND FUTURE WORK

## 7.1 Conclusion

The system proposed in this paper is acceptable level of performance and an average accuracy of 93.18%. The high fatalities of road accidents, which is primarily due to human errors committed out of fatigue, justifies the use of this system to alarm drivers at the time of driving. High-speed data processing and great accuracy distinguish this system from the similar ones. The development and improvement of this system can save the lives of millions of people annually.

The processing rate or framing of this camera is 15 fps and the first video sequence is related to the state where the head is in a lowered position which includes 85 sample frames. The second video sequence deals with the recording of the open or closed state of the eyes in which 48 image frames indicate that eyes are closed in a 6 seconds period while 65 frames show that the eyes are normally open. The third video sequence shows the yawning or the frequent opening of the driver's mouth. And finally the fourth video sequence is a combination of all three modes and its recon ling takes a longer time. The average accuracy (AAC), the detection rate (DR) and false alarm rate (FAR) has been calculated. These three factors, which have been proposed for assessing the detection accuracy of the video sequence, indicate the acceptable performance of the proposed system in detecting the signs of fatigue in driver's face at the time of driving.

## 7.2 Future Enhancement

In future works, a driver’s distraction identification system will be developed. With its complex and ever-changing nature, including the effect of the light and the condition of shooting environment, it makes the skin segmentation of human faces in color images severely affect face detection, and also makes it an important research topic. A method of face-region segmentation based on skin detection has been proposed in this paper, which partly comes from other studies. Compared with the conventional method of segmentation, we put these methods into this article, such as adjudging the images with the light interference, enhancing the images and improved threshold segmentation. Determination of the light interference not only improves the accuracy of image segmentation in the follow-up processing, but also expands the scope of application with skin segmentation in color images. Image enhancement mainly deals with the skin-likelihood image which is transformed through Gaussian model, aiming at getting the gray images with better and higher contrast. In this article, we use the method that combines the histogram with Otsu, which is the initial use of histogram threshold method to determine the threshold, and then we set it as the threshold of Otsu, and respectively, find the best.

On detecting the drowsiness the system generates the alarm to wake up the driver. Ultrasonic sensors are connected on the left and behind the car to detect the distance between the car and the road side and to detect any other car behind. Based on the situation the system tries to reduce the speed and to stop the car.When the driver is feeling drowsy the future technology would be enhanced so that the sensors are applied in cars and the car gives the indication to the neighboring vehicle and just moves towards the lane and park the vehicle.

# BIBLIOGRAPHY

[1] C. C. Liu, S. G. Hosking, "Predicting driver drowsiness using vehicle measures, recent insights and future challenges," J. safety Res, 40 (4) , pp 239 -245.2009 .

[2] G. Hosseini, H. Hossein-Zadeh, A "Display driver drowsiness warrning system", International Conference of the road and traffic accidents, Tehran University, 2006 .

[3] L. M. Bergasa, J. Nuevo, "Real-Time system for monitoring driver vigilance," IEEE Transactions on Intelligent Transportation Systems, 7(1) , pp 63-77. 2006.

[4] M. J. Flores, 1 M. Anningll and A Escalera "Real-Time Warning System for Driver Drowsiness Detection Using Visual Information", J Intell Robot System, DOT 10.100 7 /sI0846 -009 -9391 -1 , pp 10 -33. 18 N. 2009.

[5] M. Rizon and T. Kawaguchi, "Automatic eye detection using intensity and edge information”, Proceedings IEEE TENCON, vol. 2, pp. 415- 420, 2000.

[6] P. Gejgus, M. Sparka, "Face Tracking in Color Video Sequences", The Association for Computing Machinery Inc., New York, 2003.

[7] R G Gonzales, and RE. Woods, " Digital Image Processing," Second Edition, Prentice Hall, 2002.

[8] T. Azim, M.A. Jaffar, AM. Mirza. "Automatic Fatigue Detection of Drivers through Pupil Detection and Yawning Analysis." In: Proc. Fourth International Conference on Innovative Computing, Information and Control, 2009, pp. 441 -445.