CHAPTER: 1
INTRODUCTION

1. INTRODUCTION

1.1 About Fatigue Driving

It is an important factor for causing accidents in traffic that the driver's fatigue [3]. Many countries are engaged in research in this area actively now, there has been the fatigue detector that can be divided into contact and non-contact types in the market, the principles are as follows:

1.1.1 **Fatigue can cause EEG changes:** The EEG is not on the performance of the same when the cerebral cortex is in excitement or inhibition. According to the EEG's frequency distribution and waveforms, assumes the function status of the brain activity, so as to speculate whether the driver is fatigued. However the EEG is vulnerable to interference from external factors and there are so many differences in individual physiological response.

1.1.2 **Head posture:** When the driver is fatigued, the head will always downward-sloping. According to statistics, the correlation coefficient of head position and fatigue degree is about 0.8. However, some driver's head posture will not change basically, the correlation coefficient will be negative and the system's judge and early warning failed.

1.1.3 **Steering wheel's rotation amplitude and handgrip strength:** System detects the driver's mental state by monitoring steering wheel's movements and patterns. With the deepening of the driver fatigue, the number of greatly rotation will be increase; the handgrip strength will become larger.

1.1.4 **Road tracker:** This method monitors the time and the deviation degree of vehicles leaving from the white lines by installing camera in the same perspective with the driver on the vehicle. This measurement requires the white line must be exist and clear enough on the road, so the interference of outside conditions is very great.

1.2 Objective of the Fatigue Detection

How to effectively monitor and prevent driver fatigue driving has much real significance to reduce traffic accidents and personnel mortality [4]. After Comparison of the above fatigue alarm, the system through many studies of the driver's eyes, the research work in this paper include the four parts, i.e.,

Driver’s face detection,

Driver’s eye locating and tracking,

Driver’s eye state recognition and

Driver’s fatigue state identification.

CHAPTER: 2
LITERATURE SURVEY

2. LITERATURE SURVEY

2.1 Existing System

An important factor for causing accidents in traffic that the driver's fatigue. Many countries are engaged in research in this area actively now. How to effectively monitor and prevent driver fatigue driving has much real significance to reduce traffic accidents and personnel mortality.

When the driver is fatigued, the head will always downward-sloping. According to statistics, the correlation coefficient of head position and fatigue degree is about 0.8. However, some driver's head posture will not change basically, the correlation coefficient will be negative and the system's judge and early warning failed.

System detects the driver's mental state by monitoring steering wheel's movements and patterns. With the deepening of the driver fatigue, the number of greatly rotation will be increase; the handgrip strength will become larger.

This method monitors the time and the deviation degree of vehicles leaving from the white lines by installing camera in the same perspective with the driver on the vehicle. This measurement requires the white line must be exist and clear enough on the road, so the interference of outside conditions is very great.

The EEG is not on the performance of the same when the cerebral cortex is in excitement or inhibition. According to the EEG's frequency distribution and waveforms, assumes the function status of the brain activity, so as to speculate whether the driver is fatigued. However the EEG is vulnerable to interference from external factors and there are so many differences in individual physiological response.

2.2 Proposed System

2.2.1 Eyes Location

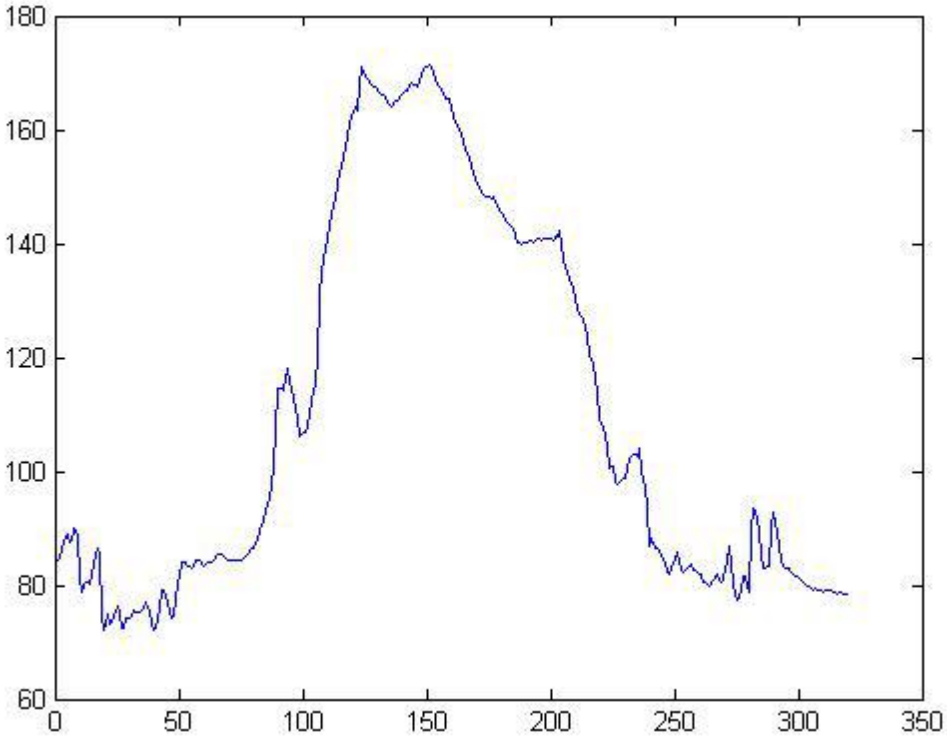
In order to make the image smoothing, doing some treatments before eyes location, including image denoising and enhancement, which is a prerequisite to ensure precise eyes location achieve the better result [5].

The first step: Locating the eye region roughly

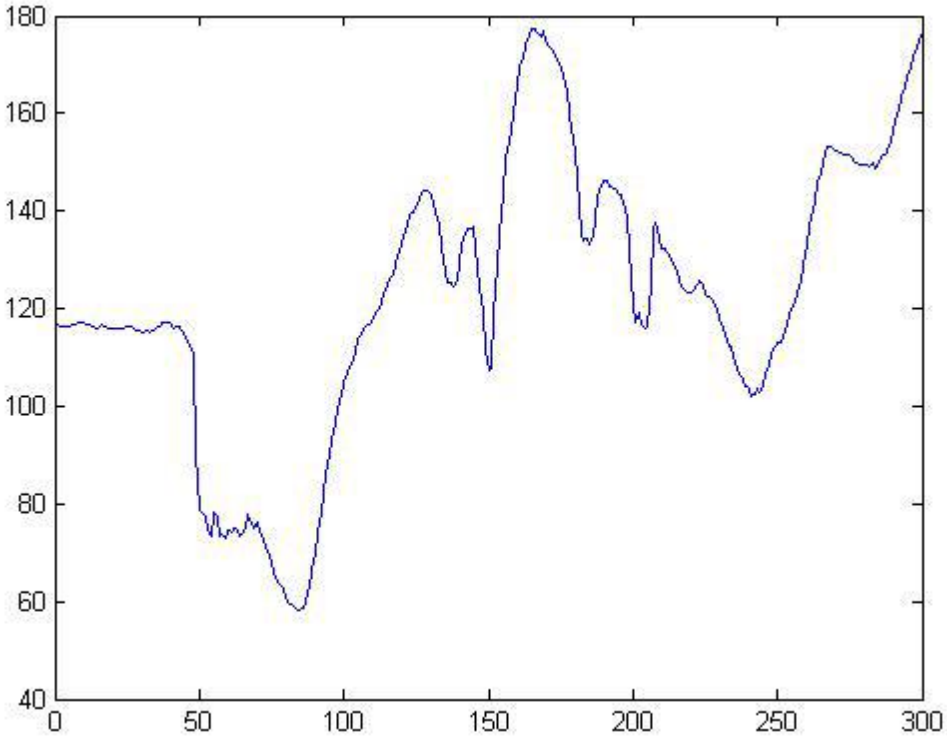
The edge feature analysis method means, making use of the vertical gray-scale projection curve of the image determined the left and right borders of the face according to the convex peak width, then making use of the horizontal gray-scale projection curve of the gotten region determined roughly the up and down border of the eyes location region. The region that corresponds to a face is a convex peak with a certain width by observing the vertical gray-scale projection curve of a number of different single-face images.

The left and right borders of the convex peak generally represented the left and right borders of the face [7]. When the left and right borders of the face are established, take the region of the face between the left and right borders as the study object, and then make the horizontal gray-scale projection curve of the image, something will be found by observing. The first minimum point of the horizontal gray-scale projection curve corresponds to the crown of the head, the maximum point corresponds to one of the forehead, the secondary maximum point corresponds to the central of the nose, and take the region between the central of the nose and the crown of the head as the rough located region (See Figure 2.1).

(a) The Original Image



(b) The result of projection from vertical direction



(c) The result of projection from horizontal direction

Figure 2.1: The results of projection from vertical and horizontal direction

After the edge feature analysis processing, the image is shown in the figure 2.2

Figure 2.2: The image after the edge feature analysis processing

The second step: sifting the similar eye points collection

The primary problem is selecting the appropriate template prior to the template matching [8]. In the follow-up algorithm, it is necessary to use the relative position between two eyes to locate the two eyes from a number of similar points, so long as to ensure that there are two real eye-points among a number of similar eye-points.

In order to reduce the two eyes' sensitivity to the eye template and improve the robustness, the system adopts the synthetic eye template of the two eyes (See Figure 2.3).

(a) Left Template (b) Right Template (c) Synthesis Template

Figure 2.3: The schematic of the eye template

In order to select the similar eye-points, it is desirable first to establish the similarity metric. The general way is doing the relevant operation to the local image and the image template, the cross-correlation coefficient obtained in this way is regarded as the similarity metric (See Formula 1). Two parameters are used to describe the synthetic template: template height M , width template N .

$$\rho_{xy}=\frac{\sum_{i,j=1}^{M,N}\left(T(i,j)-\overline{T}\right)\left(S_r(i+x,j+y)-\overline{S_r}\right)}{M\times N\times\sqrt{\frac{1}{M\times N}\sum_{i,j=1}^{M,N}\left(T(i,j)-\overline{T}\right)^2}\times\sqrt{\frac{1}{M\times N}\sum_{i,j=1}^{M,N}\left(S_r(i+x,j+y)-\overline{S_r}\right)^2}} \quad (1)$$

Therein, N is the synthetic eye template, the size is $M \times N$;T is the average of the eye template image; r S T is the average of the local image that matches with the template in the expected face recognition image; (x, y) is the coordinates of search points in the face image.

According to the above formula, operating P_{xy}, always have $\left|P_{xy}\right| \leq 1$, and the greater the P_{xy} , the higher the matching. However, due to the synthetic eye template exists a certain error and image acquisition will be affected by external conditions, when the interference, these may lead to the greatest similarity is not the real eye point, so locating the eye point can not only be determined by the size of the similarity.

In order not to miss the real eye point, the way is selecting roughly a similar eye point collection including the two real eye points (See Figure 2.4) $A=\{(X_i, Y_i) \big/ i= 1,2,\ldots n\}$, and then obtains the two real eye points through prior knowledge calibration. n is a optional coefficient.

Figure 2.4: The image of the screening similar eye-point collection

The third step: obtaining the real eye-point through calibration

The relative positions of two eyes to meet the relationship: the absolute value of the difference of the abscissa of the eyes' focal point is a within certain range; its longitudinal coordinates should be close to the same in the permitted range of the gesture change; the angle between the two eyes' line and the horizontal direction is from 45 °to 135°. After the edge feature analysis and of the eye template matching, found the main interferential point in the similar eye-point collection is the eyebrows. In order to locate the real eye-point accurately, it is first to add up a large number of face images whose actual size is equal to the human face, then get the statistics of the position relationship between two eyes; combining the any two similar eye-points, there are C_n^2 groups; then calculating the distance function values of each group by using the distance function (See Formula 2).

$$D(i,j)=\begin{cases} f\left(\left|X_i-X_j\right|,\left|Y_i-Y_j\right|\right) & c\times N<\left|X_i-X_j\right|<b\times N \text{且} \left|Y_i-Y_j\right|\le a\times M \\ A & \text{其它} \end{cases}$$
$$(i,j=1,2,\cdots,n;i\neq j) \qquad (2)$$

Therein, $D(i, j)$ is defined as a distance between the two similar eye-points i, j , which can reflect the relative position between the two similar eye-points and the closeness of the relative position of the real two eyes [6]. The higher the closeness, the smaller $D(i, j)$, on the contrary the greater. The values of parameters a, b, c are gotten by counting according to the relative position of the partial face images in the library.

You can get

$$a = \frac{\overline{Y} + (2 \sim 3) \times \sigma_y}{M}, \quad b = \frac{\overline{X} + (2 \sim 3) \times \sigma_x}{N}, \quad c = \frac{\overline{X} - (2 \sim 3) \times \sigma_x}{N};$$

$\overline{X}$ and σ_x are respectively the mean and the standard deviation of the horizontal distance, $\overline{Y}$ and σ_r are respectively the mean and standard deviation of the vertical distance; A take a larger constant.

The group of the minimum distance (i , j) in the calculation results is the located result of the two eyes (See Figure 2.5), whose coordinates in the image are (X_i , Y_i) , (X_j , Y_j) respectively.

Figure 2.5: The real eye-point image after correction

2.2.2 Eye Tracking

This system adopts the improved target tracking algorithm when it traces the eyes. The essence of target tracking is that it carries on the pinpoint while recognizing target in the image sequence.

The target tracking algorithm realized in this system divides into two parts: the primary algorithm and the modified algorithm. The primary algorithm is based on the template matching technology, namely, after pinpointing the eye point to the first frame image, it selects this eye point in the image as the tracking object and extracts appearance information of this eye point as the new eye template, in the following sequence image, it will match the candidate image region and this new eye template, then take the most similar image region as the position that this eye point in the current image.

The modified algorithm adopts the method of selecting candidate image region. It reduces the match times greatly, and then reduces the computation complexity of the system. The system uses the image gathering card for gathering image, and the rate is 25 frame per second, while the pilot driving, the head's amount of exercise is very small, therefore the position difference between the two neighboring frame images is very small, namely, it can obtain the roughly position of the eye point in the next image after pinpointing real eye point.

After adopts the target tracking algorithm, the system does not need to carry on an eye pinpointing for every frame image in the image sequence, but only repositions the eye point to the image which loses the tracking object, thus it improves operating efficiency of the system greatly and satisfies real-time request of the system too.

CHAPTER: 3
DESIGN ANALYSIS

3. DESIGN ANALYSIS

3.1 System Design

In order to enhance the accuracy rate of detection to the fatigue state of the pilot, this system extracts four state variables from the eye condition: It contains the frequency of blink, the average degree of opening eyes, the eye stagnation time and the longest time of closing eyes [2]. According to the parameter value of the pilot's sober condition by statistics, it can make the corresponding judgment by the fatigue state of the pilot.

Considered the changing driving environment, the detection system must work normally in the night or the situation of inadequate lighting, therefore, this system adopts the camera with light source(automatic opening when the light is inadequate) to gather sequence image, in order to reduce the disturbance from the external environment. When the system reads in the frame image, it carries on the denoising and the image intensification process to the image first, and then obtains two real eye points. Afterward, it adopts the target tracking method to track the already targeted eye point [1]. At last, it can calculate the area of the eyes and make the judgment and the early warning to the fatigue state of the pilot. The functional block diagram of the system is shown in the following Figure 3.1.

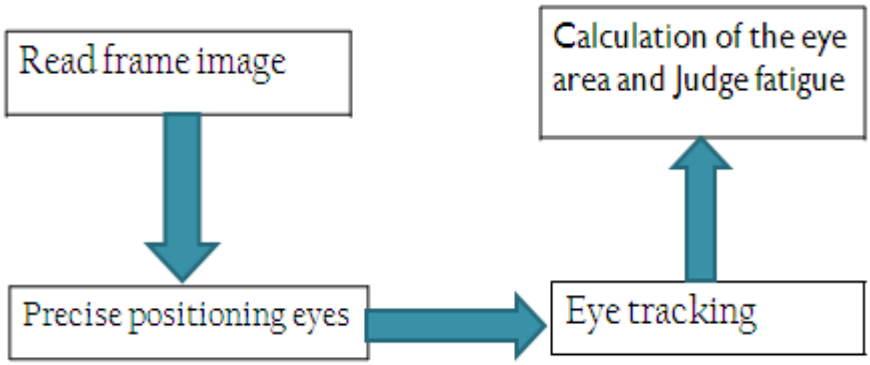


Figure 3.1: System Principle Diagram

Interface Requirements:

User Interface:

Every user may not be skilled at handling the interfaces. Hence the product that we developed used a simple and easy to use GUI Input from user is via keyboard.

Hardware Interface:

The minimum requirements that are required to interact with a simple GUI are well enough to support this product.

Software Interface:

This product is developed in Windows XP environment using MATLAB. The toolboxes used to develop this product are Image Processing Toolbox and Computer Vision System Toolbox. This project is completely implemented using these two tool boxes.

3.2 Hardware and Software Requirements

Hardware Requirements:

Processor	:	Intel® Core™ i3 2.53 GHz / Above
RAM	:	RAM 2 GB / Above
HDD	:	120 GB / Above

Software Requirements:

Operating System	:	Windows XP and above
Developing Environment	:	MATLAB R2011a

3.3 UML Diagrams:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. UML is a very important part of developing objects oriented software and the software development process.

UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software.

Definition:

UML is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of the software system.

UML is a language:

It will provide vocabulary and rules for communications and function on conceptual and physical representation. So it is modeling language.

UML Specifying:

Specifying means building models that are precise, unambiguous and complete. In particular, the UML address the specification of all the important analysis, design and implementation decisions that must be made in developing and displaying a software intensive system.

UML Visualization:

The UML includes both graphical and textual representation. It makes easy to visualize the system and for better understanding.

UML Constructing:

UML models can be directly connected to a variety of programming languages and it is sufficiently expressive and free from any ambiguity to permit the direct execution of models.

Building blocks of UML:

The vocabulary of the UML encompasses 3 kinds of building blocks

Things

Relationships

Diagrams

Things:

Things are the data abstractions that are first class citizens in a model. Things are of 4 types Structural Things, Behavioral Things, Grouping Things, Annotational Things.

Relationships:

Relationships tie the things together. Relationships in the UML are Dependency, Association, Generalization, Specialization.

UML Diagrams:

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships).

There are two types of diagrams, they are:

Structural Diagrams

Behavioral Diagrams

Structural Diagrams:

The UML‘s four structural diagrams exist to visualize, specify, construct and document the static aspects of a system. I can View the static parts of a system using one of the following diagrams. Structural diagrams consist of Class Diagram, Object Diagram, Component Diagram, and Deployment Diagram.

Behavioral Diagrams:

The UML’s five behavioral diagrams are used to visualize, specify, construct, and document the dynamic aspects of a system. The UML’s behavioral diagrams are roughly organized around the major ways which can model the dynamics of a system. Behavioral diagrams consists of Use case Diagram, Sequence Diagram, Collaboration Diagram, State chart Diagram, Activity Diagram.

3.3.1 Use Case Diagram:

A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, associations between actors and users. In general, it shows a set of use cases and actors and their relationships. The creation of a use case model is an excellent vehicle for elicitation of functional requirements.

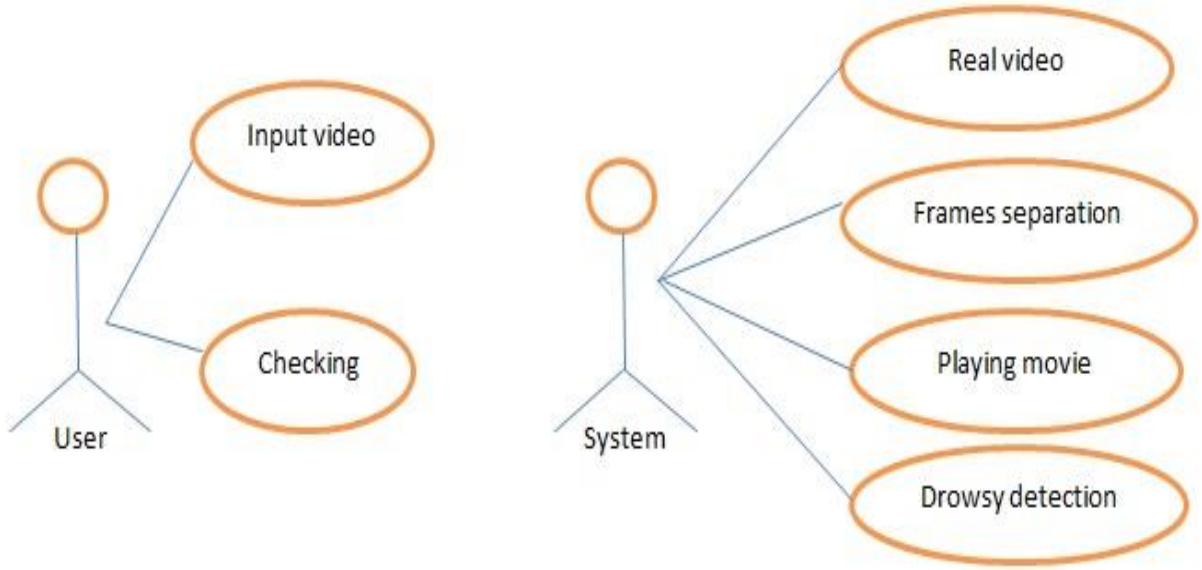


Figure 3.2: Use Case Diagram for Drowsy Detector

3.3.2 Sequence Diagram

Sequence diagram are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and its environment. A sequence diagram has two components are vertical dimension represents time, the horizontal dimension represents different object. The vertical line is called object's lifeline.

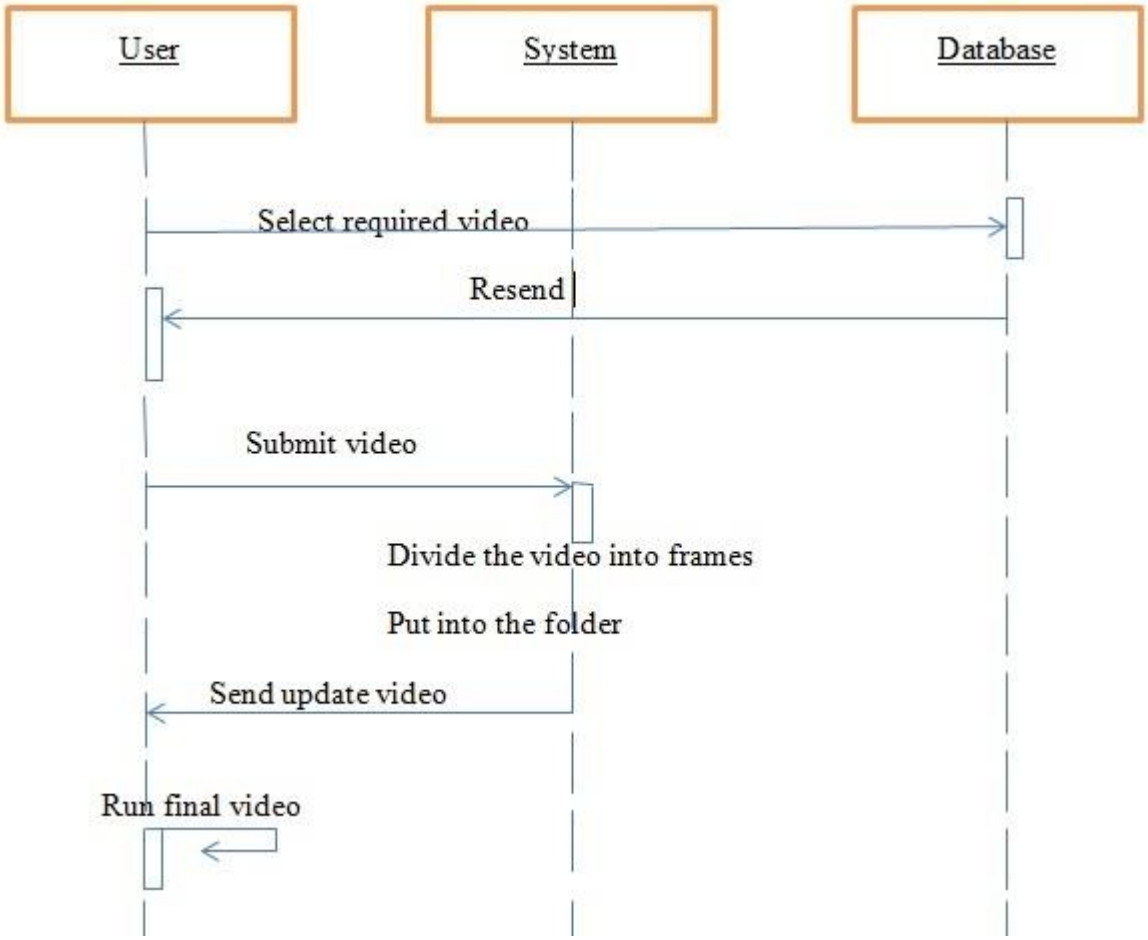


Figure 3.3: Sequence Diagram for Drowsy Detector

3.3.3 Collaboration Diagram

Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. The collaboration diagram represents interactions among objects in terms of sequenced messages.

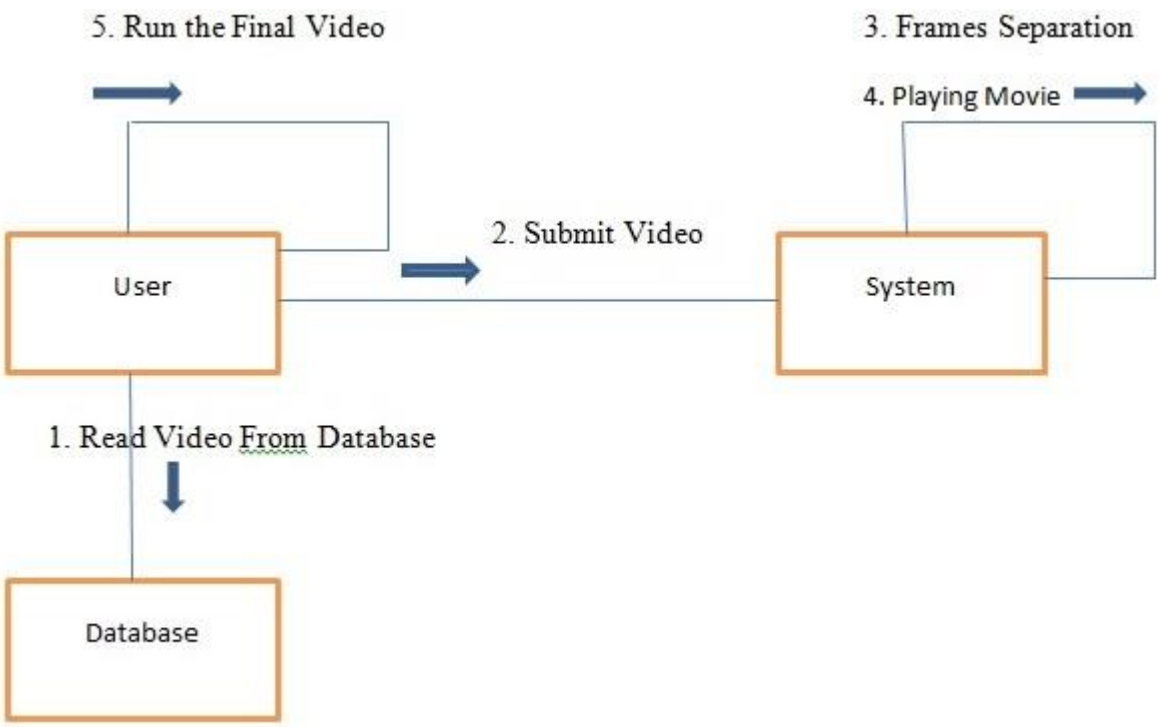


Figure 3.4: Collaboration Diagram for Drowsy Detector

3.3.4 Activity Diagram

The purpose of activity diagram is to provide a view of flows and what is going on inside a use case or among several classes. An activity is shown as around box containing the name of the operation.

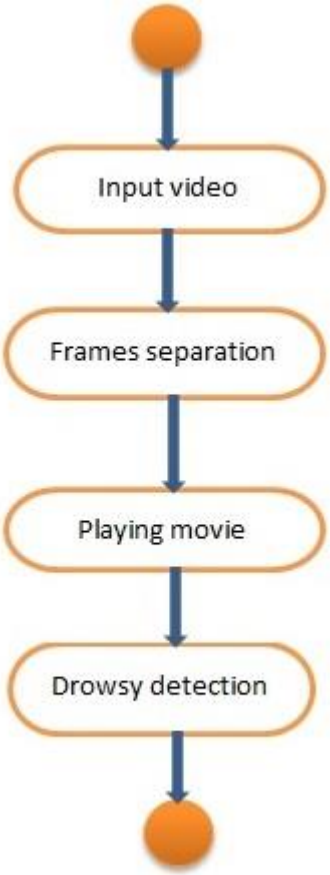


Figure 3.5: Activity Diagram for Drowsy Detector

CHAPTER – 4
TECHNOLOGY DESCRIPTION

4. TECHNOLOGY DESCRIPTION

4.1 Introduction to MATLAB

MATLAB (Matrix Laboratory) is an interactive software system for numerical computations and graphics. As the name suggests, MATLAB is especially designed for matrix computations: solving systems of linear equations, computing Eigen values and eigenvectors, factoring matrices, and so forth. In addition, it has a variety of graphical capabilities, and can be extended through programs written in its own programming language. MATLAB is designed to solve problems numerically, that is, in finite-precision arithmetic. Therefore it produces approximate rather than exact solutions. It is mathematical software that offers an Integrated Development Environment (IDE) with its own programming language, called M and is available for UNIX, Windows Apple Mac OS X platforms. It was created in the seventies by Cleve Moler, chairman of the computer science at the University of New Mexico. He tried to design a new language able to use LINPACK and EISPACK without any knowledge about Fortran.

This software was quickly expanded to other universities and was really welcome by the applied mathematics community. In the eighties Jack Litte, an engineer, joined Moler, because he realized it had a tremendous commercial potential. They wrote MATLAB in C, and founded Math works in 1984.MATLAB then was turned into a language for technical computing of high performance. It integrates visualization, computation and an easy-to-use programming environment where problems and solutions are expressed in familiar mathematical notation.

The typical uses of MATLAB are:

Math and computation

Algorithm development

Modeling, simulation and prototyping

Data analysis, exploration and visualization

Scientific and engineering graphics

General purpose commands in MATLAB

global : Declares variables to be global.

help : Searches for a help topic.

lookfor : Searches help entries for a keyword.

quit : Stops MATLAB.

who : Lists current variables.

4.1.1 Introduction to M-files

Scripts and Functions:

MATLAB can also be used as a programming language. To program in MATLAB you simply create a text file containing MATLAB commands exactly as you would type them interactively in the MATLAB window. The file may have any legal UNIX name, and should end with a **.**m extension. These files may be placed in the root directory, or a directory named MATLAB. (Any other directories would have to be explicitly added to the MATLAB path.) There are two types of m-files in MATLAB. One is called a script. This is simply a list of MATLAB commands with no header. The other type is a function. Functions have a header line that may look something like: Function y=fun1(x) Functions may be passed arguments, and may return results. To invoke a script or a function simply type the filename (without the .m extension) into the MATLAB window. Also the m-files that are created are included in the help listing. If we perform a help on a specific m-file, help will return any comments which appear before the first line of actual code in the m-file.

Debugging M-files

This section introduces general techniques for ending errors in M-files. Debugging is the process by which you isolate errors in the program or code. Debugging helps to correct two kinds of errors:

1. Syntax errors

   For example omitting a parenthesis or misspelling a function name.

2. Run-time errors

   Run-time errors are usually apparent and difficult to track down.

Correcting an M-file

To correct errors in an M-file, we can adopt any of the following ways,

Quit debugging

Do not make changes to an M-file while MATLAB is in debug mode

Make changes to the M-file

Save the M-file

Clear breakpoints

Run the M-file again to be sure it produces the expected results.

Thus the MATLAB has many utilities that can be used for efficient tasking. It provides the user with many toolboxes that acts as a guide in many areas.

4.2 Functions used:

1. Vision.VideoFileReader ():-

VideoFileReader reads video frames and audio samples from video file.

Syntax:

HVFR = vision.VideoFileReader

Description:

It returns a video file reader System object, HVFR, to read video and/or audio from a video file.

2. Vision.ColorSpaceConverter ():-

ColorSpaceConverter does the image color space conversion.

Syntax:

HCSC = vision.ColorSpaceConverter ('Property Name', Property Value ...)

Description:

It returns a color space conversion object, HCSC, with each specified property set to the specified value.

3. Vision.ForegroundDetector ():-

ForegroundDetector detects foreground using Gaussian Mixture Models.

Syntax:

H = vision.ForegroundDetector ('Property Name', Property Value ...)

Description:

It returns a foreground detector System object, H, with each specified property set to the specified value.

ForegroundDetector properties:

NumTrainingFrames - Number of initial video frames used for training the background model

LearningRate - Learning rate used for parameter updates

MinimumBackgroundRatio-Threshold to determine the gaussian model in the mixture model that constitutes the background process

NumGaussians - Number of distributions that make up the foregroundbackground mixture model

InitialVariance - Initial variance to initialize all distributions that compose the foreground-background mixture model.

4. Vision.BlobAnalysis ():-

BlobAnalysis gives the properties of connected regions.

Syntax:

HBLOB = vision.BlobAnalysis ('Property Name', Property Value ...)

Description:

It returns a blob analysis object, HBLOB, with each specified property set to the specified value.

BlobAnalysis properties:

AreaOutputPort - Enables blob area output

CentroidOutputPort - Enables coordinates of blob centroids output

BoundingBoxOutputPort- Enables coordinates of bounding boxes output

OrientationOutputPort - Enables output vector whose values represent angles between ellipses' major axes and x-axes

PerimeterOutputPort - Enables output vector whose values represent estimates of blob perimeter lengths

MaximumCount -Maximum number of labeled regions in each input image

NumBlobsOutputPort -Enables output of a scalar value that represents actual number of labeled regions in each image

MinimumBlobAreaSource -Source of minimum blob area

MinimumBlobArea -Minimum blob area in pixels

MaximumBlobAreaSource -Source of maximum blob area

MaximumBlobArea -Maximum blob area in pixels.

5. Vision.ShapeInserter ():-

ShapeInserter draws rectangles, lines, polygons, or circles on images.

Syntax:

HSHAPEINS = vision.ShapeInserter ('PropertyName', PropertyValue ...)

Description:

It returns a shape inserter System object, HSHAPEINS, with each specified property set to the specified value.

ShapeInserter properties:

Shape -Type of shape(s) to draw

BorderColorSource -Source of border color

BorderColor -Border color of shape

CustomBorderColor -Intensity or color value for shape's border

FillColor -Fill color of shape

CustomFillColor -Intensity or color value for shape's interior

ROIInputPort -Enables defining area for drawing shapes via input

Antialiasing -Enables performing smoothing algorithm on shapes

6. Vision.TextInserter ():-

TextInserter draws text on image or video stream.

Syntax:

HTXTINS = vision.TextInserter (TEXT, 'PropertyName', PropertyValue ...)

Description:

It returns a text inserter object, HTXTINS, with the Text property set to TEXT and other specified properties set to the specified values.

TextInserter properties:

Text - Text string to draw on image or video stream

ColorSource - Source of intensity or color of text

Color - Intensity or color of text

Location - Top-left corner of text bounding box

Font - Font face of text

FontSize - Font size in points

Antialiasing - Perform smoothing algorithm on text edges.

7. Vision.VideoPlayer ():-

VideoPlayer play video or display image

Syntax:

HVP = vision.VideoPlayer ('PropertyName', PropertyValue ...)

Description:

It returns a video player System object, HVP, with each specified property set to the specified value.

VideoPlayer properties:

Name - Caption to display on video player window

Position - Scope window position in pixels.

8. step ():-

Gives the step response of dynamic systems.

Syntax:

STEP (SYS1, SYS2,..., T)

Description:

It plots the step response of several systems SYS1, SYS2... on a single plot. The time vector T is optional. You can also specify a color, line style, and marker for each system.

9. int32 ():-

Convert to signed 32-bit integer.

Syntax:

I = int32(X)

Description:

It converts the elements of the array X into signed 32-bitintegers. X can be any numeric object, such as a DOUBLE. The values of an INT32 range from -2,147,483,648 to 2,147,483,647.

10. aviread ():-

Read AVI file.

Syntax:

MOV = aviread (FILENAME)

Description:

It reads the AVI movie FILENAME into the MATLAB movie structure MOV.

11. frame2im ():-

Return image data associated with movie frame.

Syntax:

[X, MAP] = frame2im (F)

Description:

It returns the indexed image X and associated color map MAP from the single movie frame F.

12. imabsdiff ():-

Absolute difference of two images.

Syntax:

Z = imabsdiff(X, Y)

Description:

It subtracts each element in array Y from the corresponding element in array X and returns the absolute difference in the corresponding element of the output array Z.

13. im2double():-

Convert image to double precision.

Syntax:

im2double (I1)

Description:

im2double takes an image as input, and returns an image of class double. If the input image is of class double, the output image is identical to it. If the input image is not double, im2double returns the equivalent image of class double, rescaling or offsetting the data as necessary.

14. rgb2gray ():-

Convert RGB image or color map to gray scale

Syntax:

I=rgb2gray (RGB)

newmap=rgb2gray (map)

Description:

I=rgb2gray(RGB) converts the true color image to RGB to the gray scale intensity image I. rgb2gray converts RGB images to gray scale by eliminating the hue and saturation information while retaining the luminance.

15. release ()

Release a COM interface.

Syntax:

release(object)

Description:

release (OBJ) releases the interface and all resources used by the interface. Once an interface has been released, it is no longer valid.Subsequent operations on the MATLAB object that represents that interface will result in errors.

CHAPTER -5

TESTING

5. TESTING

5.1 Introduction to Testing:

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements. Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. It represents interesting anomaly for the software.

Testing is the process of detecting errors. It performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also. Generally, the testing phase involves the testing of the developed system using various test data. Preparation of the test data plays a vital role in the system testing. After preparing the test data the system under study was tested using those test data. While testing the system, errors were found and corrected by using the testing steps and corrections are also noted for future use. Thus, a series of testing is performed for the proposed system, before the system was ready for the implementation.

Thus the aim of testing is to demonstrate that a program works by showing that it has no errors. The fundamental purpose of testing phase is to detect the errors that may be present in the program. Thus testing allows developers to deliver software that meets expectations, prevents unexpected results, and improves the long term maintenance of the application. Depending upon the purpose of testing and the software requirements, the appropriate methodologies are applied. Wherever possible, testing can also be automated.

5.1.1 Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, altogether, Testing is a process of executing a program with the intent of finding an error. A successful test is one that uncovers an as yet undiscovered error. A good test case is one that has a high probability of finding error, if it exists. The software more or less confirms to the quality and reliable standards.

5.1.2 Levels of testing:

In order to uncover the errors present in different phases we have the concept of levels of testing. The basic levels of testing are

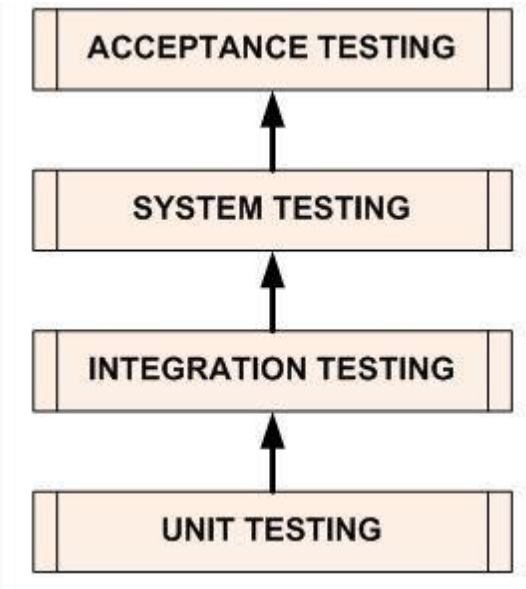


Figure 5.1: Levels of Testing

5.2 Testing Strategies:

5.2.1 Unit testing

Unit Testing is a level of the software testing process where individual units or components of a software or system are tested. It is also known as component testing, refers to tests that verify the functionality of a specific section of code. The purpose is to validate that each unit of the software performs as designed. All modules must be successful in the unit test before the start of the integration testing begins. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

Unit testing focuses verification effort on the similar unit of software design the form. This is known as form testing. In this testing step, each module is found to be working satisfactorily, as regard to the expected output from the module. Each module has been tested by giving different sets of inputs, when developing the module as well as finishing the development so that each module works without any error. The inputs are validated when accepting from the user. Each module can be tested using the following two strategies:

White Box Testing:

White box testing is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. We tested step wise every piece of code, taking care that every statement in the code is executed at least once. White-box testing can be applied at the unit, integration and system levels of the software testing process. It is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test.

The white box testing is also called Glass Box Testing. We have generated a list of test cases sample data, which is used to check all possible combinations of execution paths through the code at every module level. This testing has been uses to find in the following categories:

Execute internal data structures to ensure their validity.

Guarantee that all independent paths have been executed.

Execute all logical decisions on their true and false sides.

Black Box Testing:

The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It treats the software as a "black box", examining functionality without any knowledge of internal implementation. This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level i.e., here the module will be treated as a black box that will take some input and generate output. Output for a given set of input combinations are forwarded to other module. This testing has been uses to find in the following categories:

Initialization and termination errors.

Incorrect or missing functions

Performance errors

5.2.2 Integration Testing

Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. After the unit testing we have to perform integration testing. The goal here is to see if modules can be integrated properly, the emphasis being on testing interfaces between modules. It works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

The testing of combined parts of an application to determine if they function correctly together is the main motto of integration testing. There are two methods of doing integration testing. They are Bottom-up integration testing and Top Down integration testing. All modules are combined in the testing step. Then the entire program is tested as a whole.

5.2.3 Validation Testing

At the culmination of the integration testing, the software is completely assembled as a package, interfacing errors have been uncovered and corrected and final series of software validation testing begins. Validation checks that the product design satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements.

5.2.4 System Testing

System testing is a level of the software testing process where a complete, integrated system or software is tested. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets quality standards. Here the entire software system is tested. The reference document for this process is the requirements document, and the goal is to see if software meets its requirements. Here entire project has been tested against requirements of project and it is checked whether all requirements of project have been satisfied or not.

5.2.5 Output Testing

After performing validation testing, the next steps are output testing of the proposed system, since no system could be useful if it does not produce the desired output in the specified format. The output generated are displayed by the system under consideration or tested by asking the user about the format required by them. Here the output format is considered in two ways. One is on the screen and the other is on the printed form.

5.2.6 Acceptance Testing

Acceptance Testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery. Acceptance test is performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here is focused on external behavior of the system, the internal logic of program is not emphasized.

In this project we have collected some data and tested whether project is working correctly or not. Test cases should be selected so that the largest number of attributes of an equivalence class is exercised at once. The testing phase is an important part of software development. It is the process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. User acceptance of a system is the key factor for the success of any system. The system under consideration was tested for user acceptance by constantly keeping in touch with the perspective system users at the time of developing and making changes whenever required. This is done with regard to the following points.

Input screen design

Output screen design

Menu driven system

5.3 Test Approach:

Testing can be done in two ways:

Bottom up approach

Top down approach

Bottom up Approach

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will

when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

Top down Approach

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

5.4 Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

5.5 Test cases Results

Testing is the set of activities that can be planned in advance and conducted systematically.

The underlying motivation of program testing is to affirm software quality.

5.6 Sample Screenshots

First of all the screenshots of the input video are shown followed by the screens of the output video that has been obtained in MATLAB.

5.6.1 Screenshots of Input video

The screenshots of the video that was used in this project are as follows.

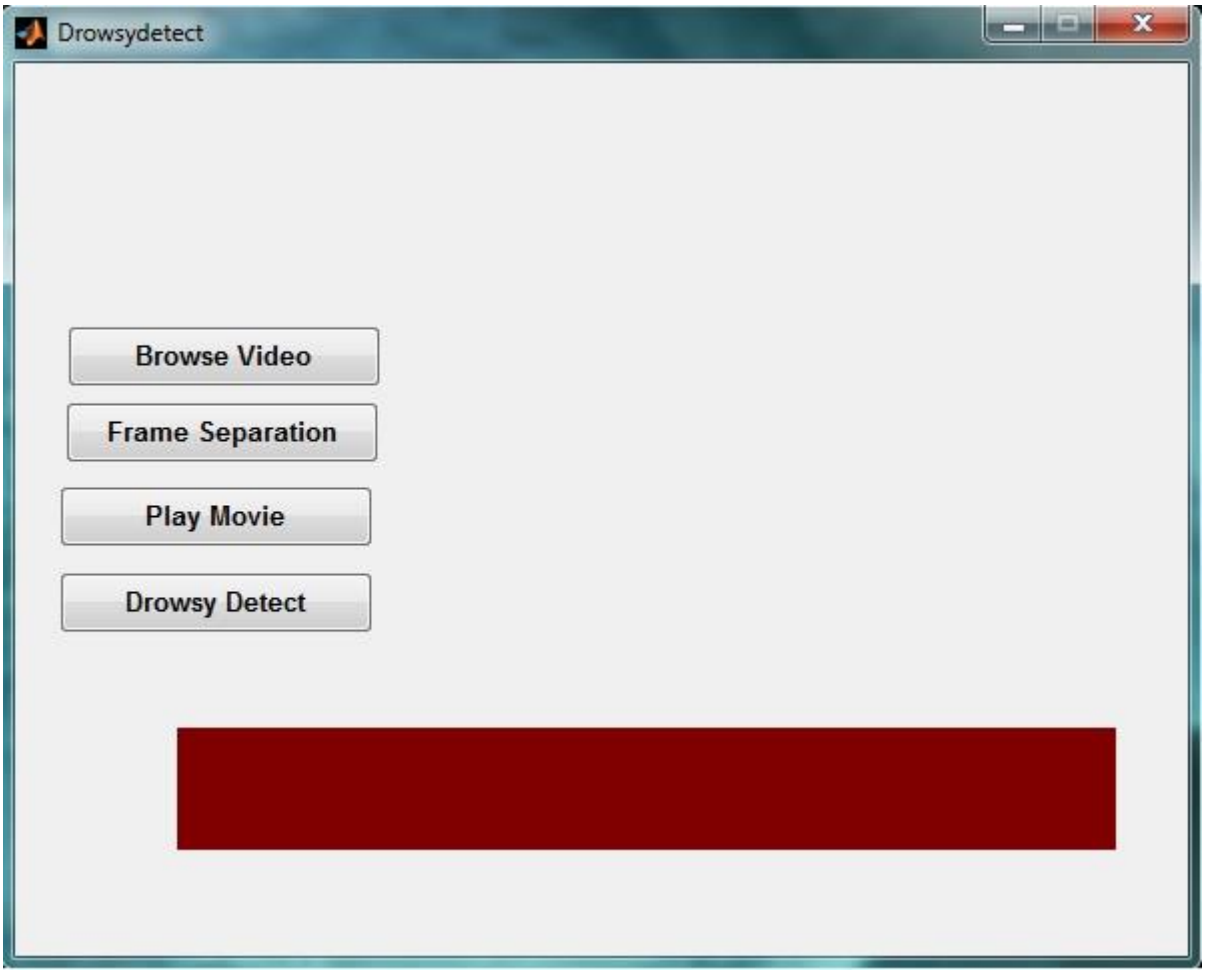


Figure 5.2: GUI of Project Screenshot 1

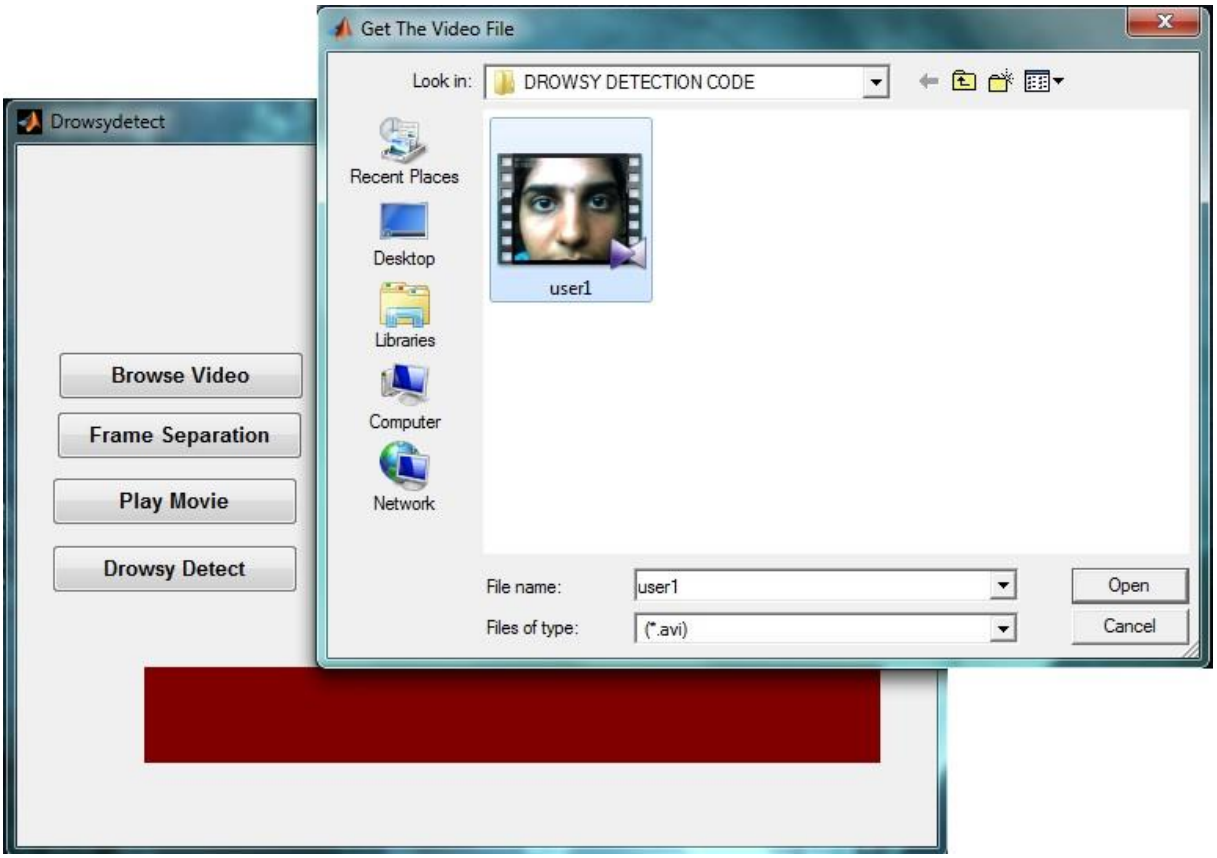


Figure 5.3: Browsing Required Video Screenshot 2

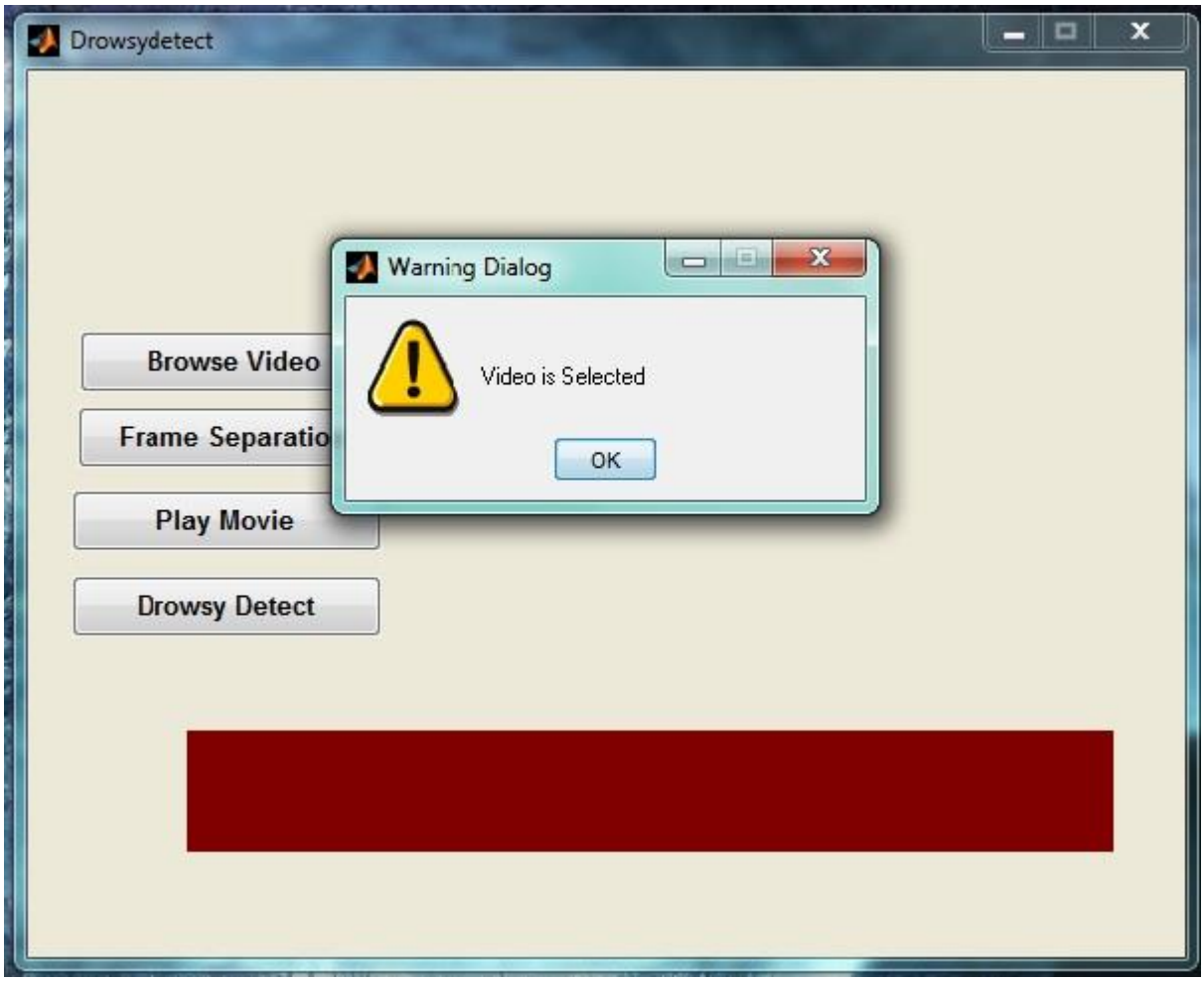


Figure 5.4: Video Selected Screenshot 3

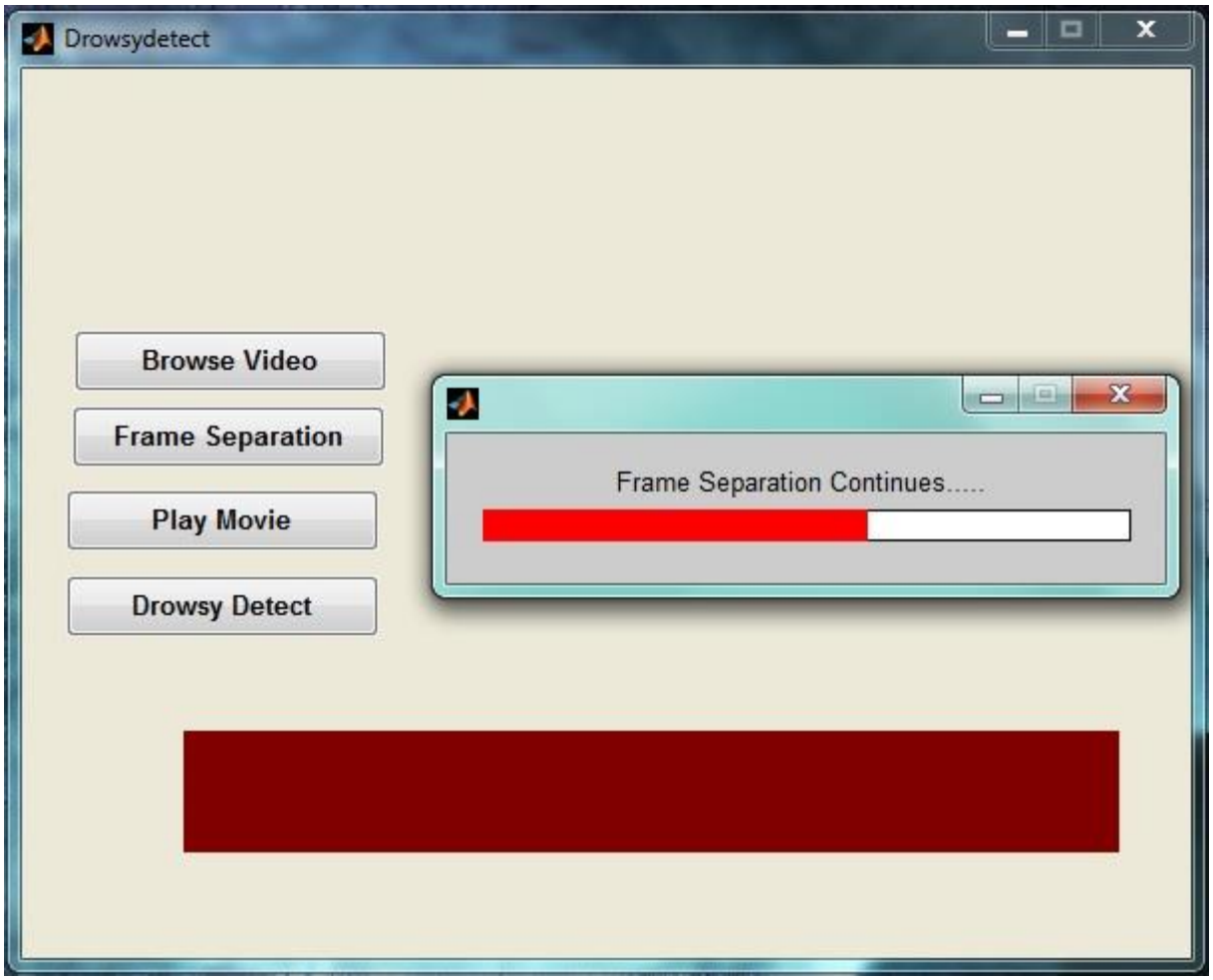


Figure 5.5: Frame Separation of Input Video Screenshot 4

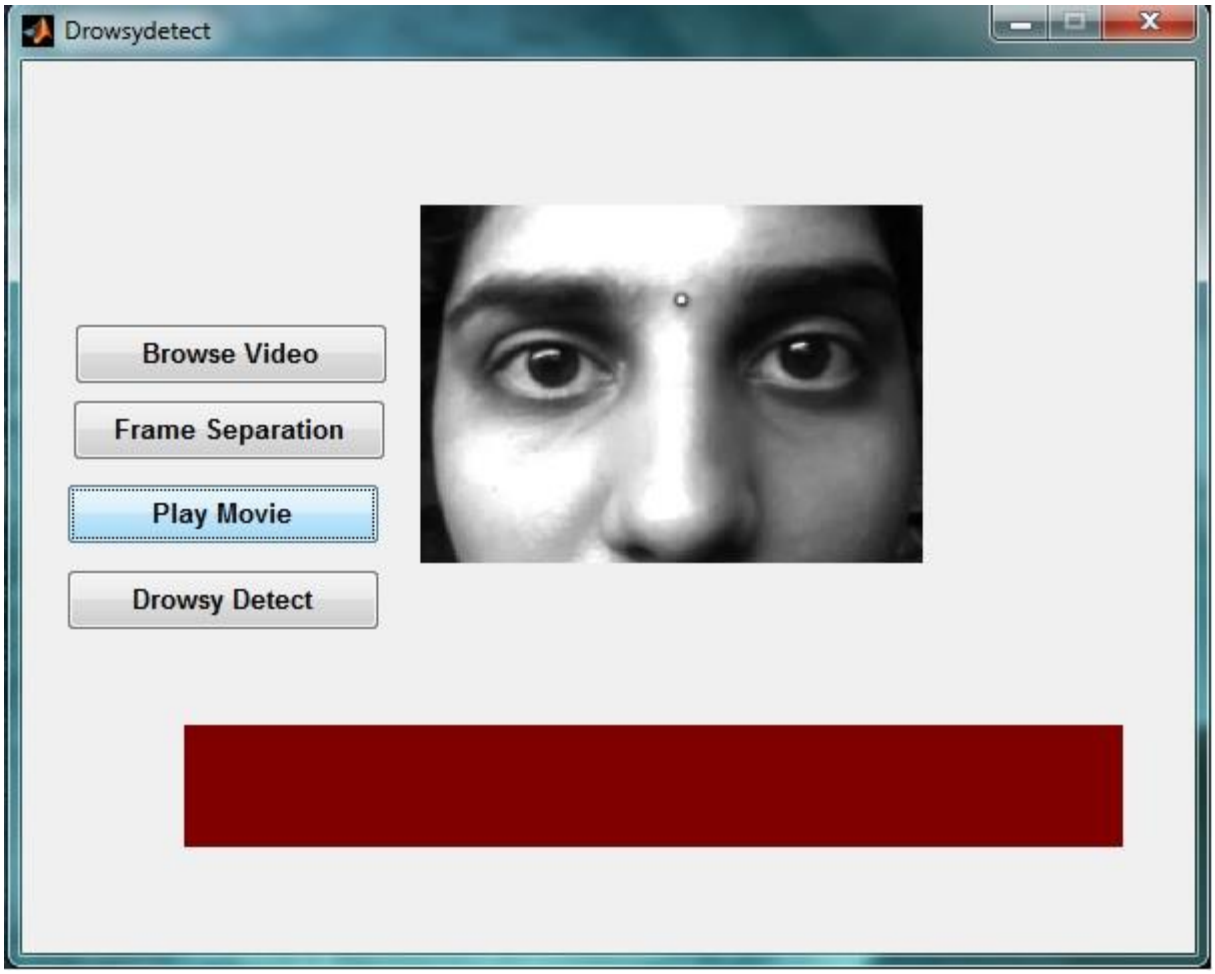


Figure 5.6: Playing Movie of Frame Separated Input Video Screenshot 5

5.6.2 Screen Shots of Output Video

When the code is executed in MATLAB, the output video with the drowse detection is displayed. The screen shots of the output video that was obtained in this project are as follows.

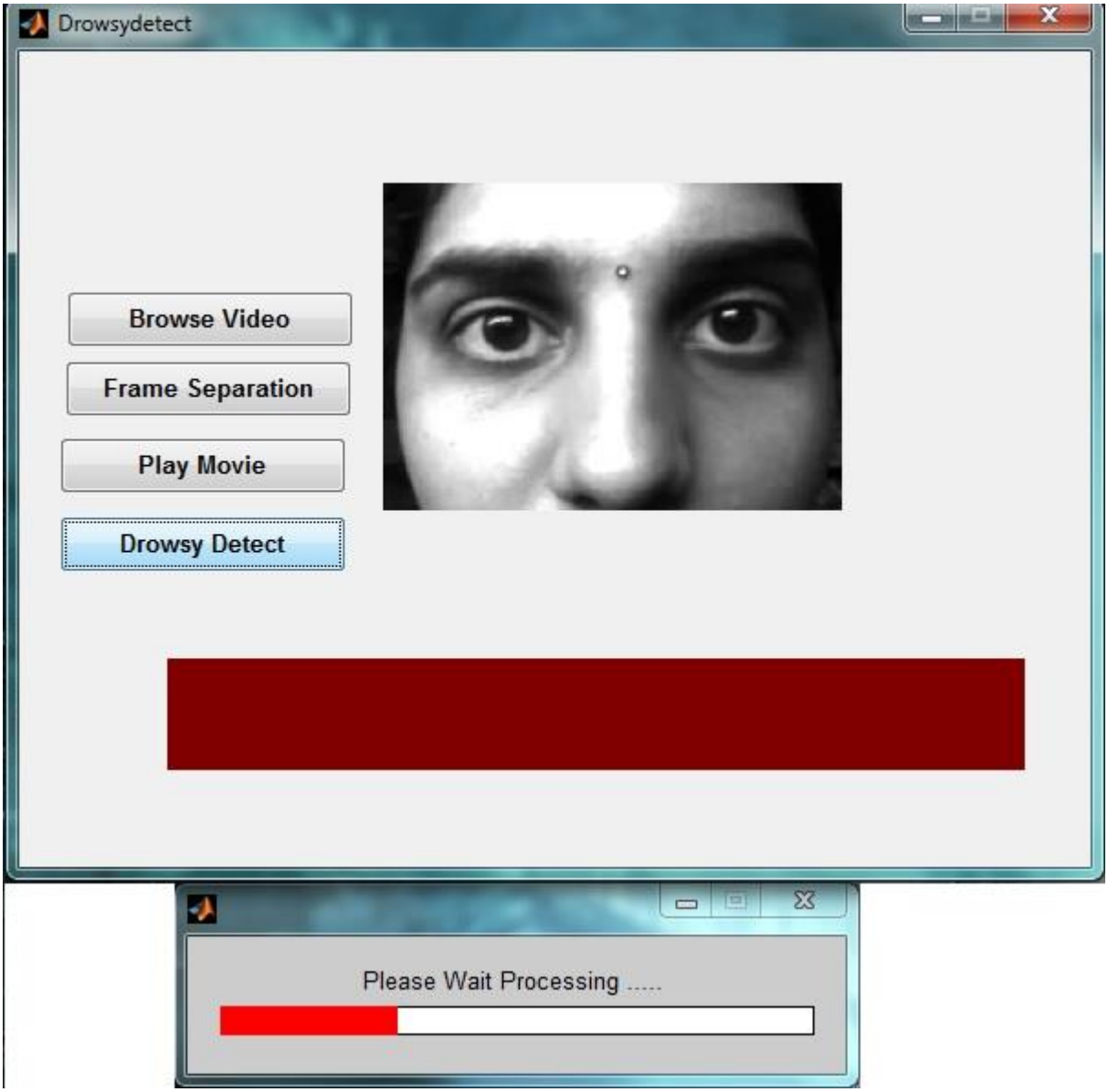


Figure 5.7: Processing Drowsy Detection of Input Video Screenshot 6

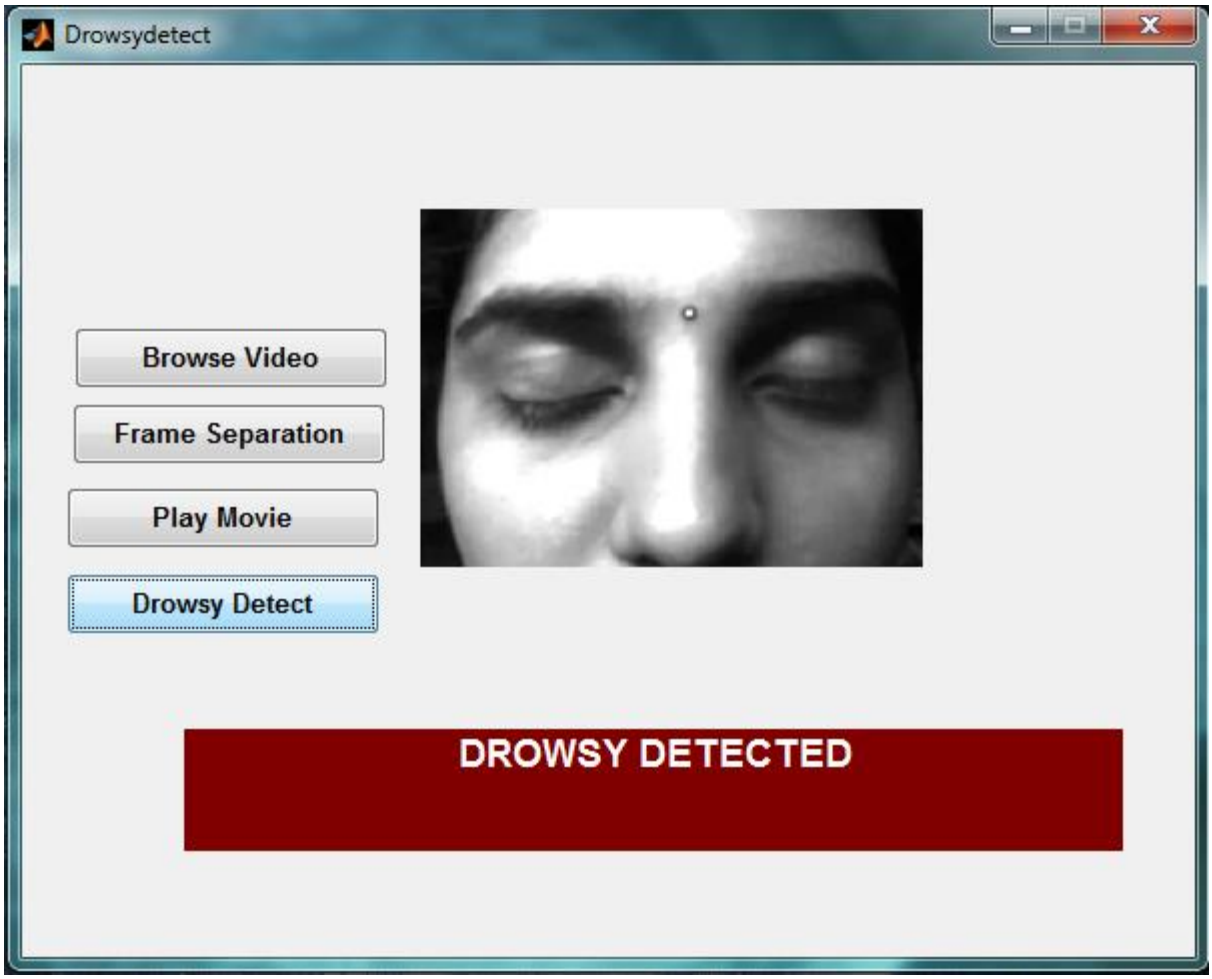


Figure 5.8: Drowsy Detection Output Video Screenshot 7

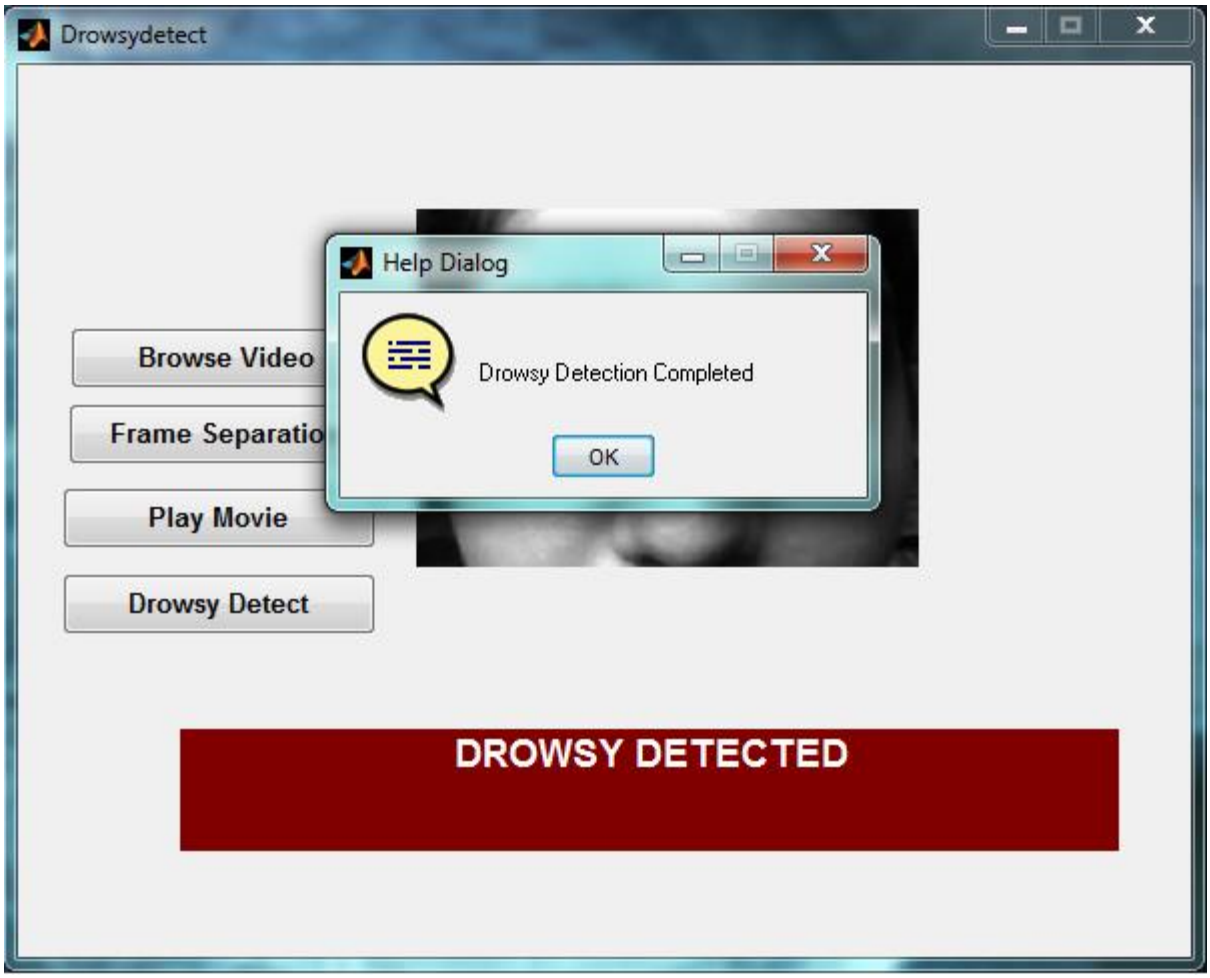


Figure 5.9: Drowsy Detection Completed Output Video Screenshot 8

CHAPTER: 6
CONCLUSION

6. CONCLUSION

The system is capable of accurate positioning eye point. Using four parameters of eye states can effectively detect the driver's fatigue status. In order to improve the accuracy grade, our system should be using some other methods as a supplementary means, such as

- Road tracking,
- Head position,
- The rotation rate and the grip force of the steering wheel, which are the main directions to improve our system accuracy.

CHAPTER: 7
BIBLIOGRAPHY

7. BIBLIOGRAPHY

[1] Chen Terrence, Zhou Xiang Sean, Comaniciu Dorin,Huang Thomas S. Total Variation Models for Variable Lighting Face Recognition [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(9):1519□1524.

[2] Haycocks G.Driver sleepiness as a factor in car HGV. Accidents, transport research laboratory[J]. Report, 1995, HS, 169.

[3] Home JA,Reyner LA.Sleep related vehicle accidents[J]. BMJ, 1995,30:565-567.

[4] NCSDR/NHTSA. Expert panel on driver fatigue & sleepiness. Drowsy driver and automobile crashes[J]. Report, 1998, HS. 808. 707.

[5] NHTSA. Expert Panel on Driver Fatigue & Sleepiness. Drowsy driver and automobile crashes Report. 1998.

[6] Wang Lei, Mo Yu-Long, Qi Fei-Hu. Automatic Eye Feature Extraction Based on Hough Transform And Eyrlid Deformable Template. J.Infrared Millim[J]. Waves 1999, 18(1): 53-60.

[7] Zhao W. Robust Image-Based 3D Face Recognition [D]. Doctoral Thesis, 1999, Center for Auto Research, Univ. of Maryland.

[8] Zhong Wei, Liu Zhi-Ming, Zhou Ji-Liu.Study on Precise Eyes Location in Face Detection [J]. Computer Engineering and Applications, 2004, 36, 73-76.