

EYE DETECTION

NURUL ASMIRA BT AHMAD AGIN

UNIVERSITI MALAYSIA PAHANG

“I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor of Electrical Engineering (Control and Instrumentation)”

Signature : .....

Supervisor : Nor Farizan Binti Zakaria

Date : .....23<sup>th</sup> November 2009.....

EYE DETECTION

NURUL ASMIRA BT AHMAD AGIN

“This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Control and Instrumentation)”

Faculty of Electrical & Electronics Engineering

Universiti Malaysia Pahang

NOVEMBER, 2009

“All the trademark and copyrights use herein are property of their respective owner. References of information from other sources are quoted accordingly; otherwise the information presented in this report is solely work of the author.”

Signature : .....

Author : Nurul Asmira Binti Ahmad Agin

Date : ..... 23<sup>th</sup> November 2009.....

*To My Beloved Mum, My Lovely Sisters, My Brothers & My Friends*

*Khazimah Binti Daud*

*Nor Maizatul Aida Binti Ahmad Agin*

*Nor Haidatul Akmar Binti Ahmad Agin*

*Nor Hafiza Binti Ahmad Agin*

*Ahmad Khairul Azmir Bin Ahmad Agin*

*Ahmad Khairul Azhar Bin Ahmad Agin*

*Ahmad Khairul Basri Bin Ahmad Agin*

*Ahmad Syaiful Amri Bin Ahmad Agin*

*Mohd Fahimi Bin Musa*

*Nor Shuhaida Binti Aidun*

*Thank's For Everything...*

## **ACKNOWLEDGEMENT**

Alhamdulillah, His Willingness has made it possible for the author to complete the final year project in time.

I would like to take this opportunity to express gratitude to my dedicated supervisor, Mrs. Nor Farizan binti Zakaria for guiding this project at every stage with clarity and that priceless gift of getting things done by sharing her valuable ideas as well as share his knowledge. I would also like to thank to all UMP lecturers who had helped directly or indirectly in what so ever manner thus making this project a reality.

Not forgotten are my best colleagues for their openhandedly and kindly guided, assisted, supported and encouraged me to make this project successful. My heartfelt thanks to my dearest family which always support and pray on me throughout this project. Their blessing gave me the high-spirit and strength to face any problem occurred and to overcome them rightly.

The great cooperation, kindheartedness and readiness to share worth experiences that have been shown by them will be always appreciated and treasured by me. Besides that, I also wish acknowledgement to the people who give support direct or indirectly to the project and during the thesis writing. Once again, thank you very much.

## **ABSTRACT**

Eye detection is a process where the position of the eyes is extracted from the face images. Eye detection is one of the applications in image processing. This project is important in human identification and it can improve today's identification technique that only involves the fingerprint to identify people. This eye detection is applicable in security system and drowsiness detection. This technology is still new and only few systems are applying this system as their security system. The most crucial part in eye detection system is to identify the eye location. This system is focus on four major parts of preprocessing phase which are image enhancement using median filtering, threshold process, feature extraction and the face and eye detection. This system used an offline face image database. The frontal face image is fed to the system which threshold the image to convert the image into binary image. This process produced an enhanced image after the filtering process. After these preprocessing, feature extraction process is performed to produces the features of the binary image. The face region is determined using rectangular function in MATLAB to limit the range of image to be process and to reduce the noise. Next, the eyes are detected using the morphology technique. A dilation technique is applied to build the square using the structuring element. The eye is view in a square region. Finally, the graphical user interfaces is designed to display the output image.

## **ABSTRAK**

Pengesanan mata adalah satu proses di mana kedudukan mata dikeluarkan dari imej muka tersebut. Sistem pengesanan mata merupakan satu sistem yang menggunakan pemprosesan imej. Projek ini penting dalam pengenalan manusia yang boleh meningkatkan lagi sistem pengenalan manusia pada hari ini yang hanya melibatkan cap jari sebagai pengenalan manusia. Sistem pengesanan mata ini boleh digunakan dalam sistem keselamatan dan pengesanan mengantuk. Teknolgi ini masih baru dan beberapa sistem menggunakannya sebagai sistem keselamatan. Bahagian paling genting dalam sistem pengesanan mata adalah untuk mengesan kedudukan mata itu. Sistem ini menfokuskan empat bahagian besar dalam proses permulaan iaitu pemulihan imej dengan menggunakan teknik median, teknik ambangan dan pengesanan wajah dan mata. Projek ini menggunakan pangkalan data imej muka tertutup. Muka hadapan wajah ditukar kepada imej binari. Teknik ini menghasilkan imej yang telah dipulihkan selepas proses penyaringan dijalankan. Selepas proses permulaan, Process pengeluaran ciri dijalankan ke atas imej binari. Bahagian wajah dikenal pasti dengan menggunakan teknik segi empat tepat yang terdapat di dalam Matlab untuk menghadkan lingkungan imej untuk diproses serta mengurangkan halangan. Selepas itu, mata dikesan menggunakan teknik morfologi. Teknik pengembangan digunakan untuk membina satu segi empat sama dengan menggunakan unsur pembinaan. Mata dikesan berada di dalam segi empat sama tersebut. Akhir sekali, GUI direka bagi mempamerkan keputusan imej.



## TABLE OF CONTENTS

<b>CHAPTER</b>	<b>TITLE</b>	<b>PAGE</b>
	<b>Acknowledgement</b>	i
	<b>Abstract</b>	ii
	<b>Abstrak</b>	iii
	<b>Table of Contents</b>	iv
	<b>List of Tables</b>	vii
	<b>List of Figures</b>	viii
	<b>List of Appendices</b>	ix
<b>1.0</b>	<b>INTRODUCTION</b>	<b>1</b>
	1.1 Background Study	1
	1.2 Problem Statement	3
	1.3 Objective	5
	1.4 Scope	5
	1.5 Thesis Organization	6
<b>2.0</b>	<b>LITERATURE REVIEW</b>	<b>8</b>
	2.1 An Overview	8
	2.2 Face and Eye Detection	10
	2.3 Image Analysis	12
	2.4 Image Acquisition	12
	2.4.1 Image Enhancement	13

	2.4.2 Mean Filtering	14
	2.4.3 Median Filtering	15
	2.5 Thresholding	16
	2.5.1 Mean Thresholding	17
	2.5.2 Median Thresholding	18
	2.6 Feature Extraction	18
3.0	METHODOLOGY	20
	3.1 An Overview of the Project	20
	3.2 Eye Detection Process	22
	3.3 Preprocessing	24
	3.4 Load and Show Image	25
	3.5 Threshold Process	26
	3.6 Filtering Process	30
	3.6.1 Median Filter	30
	3.6.2 Morphology (Dilation and Erosion)	33
	3.6.3 Dilating an Image	36
	3.6.4 Eroding an Image	37
	3.6.5 Combining Dilation and Erosion	39
	3.6.6 Morphological Opening	39
	3.7 Processing	41
	3.7.1 Feature Extraction	41
	3.7.2 Segmentation Process	42
	3.7.3 Canny Edge Detection	43
	3.7.4 Face and Eye Detection	46
	3.7.5 Eye Detection	49
	3.8 GUIDE MATLAB	50
	3.8.1 The Layout Editor	52
	3.8.2 Setting Properties for GUI Components	53
	3.8.3 GUI FIG-Files and M-Files	54

4.0	RESULT AND DISCUSSION	56
	4.1 Load Image from Database	56
	4.2 Preprocessing Result	57
	4.3 Face Region Detection	58
	4.4 Threshold and Filtering Process	59
	4.5 Eye Region Detecton	60
	4.6 Image of the Eye Detection Process	61
	4.7 GUI	62
5.0	CONCLUSION	63
	5.1 Conclusion	63
	5.2 Future Recommendation	64
	REFERENCES	66
	Appendices A-C	69-84

**LIST OF TABLES**

<b>TABLE NO</b>	<b>TITLE</b>	<b>PAGE</b>
3.16	Dilation and Erosion Rules	34
3.17	Dilation and Erosion Rules for Gray Scale Image	34

## LIST OF FIGURES

FIGURE NO	TITLE	PAGE
1.1	Example Of Frontal Face Image	6
3.1	Block Diagram for Eye Detection	21
3.2	Example Of Input Image	22
3.3	Eye Detection Process	23
3.4	Coding of load and show image.	25
3.5	Image Display in Figures Window	25
3.6	Binary Image	26
3.7	Coding to Convert RGB image into grayscales.	27
3.8	Grayscales Image.	28
3.9	Coding for converting grayscale to binary image.	28
3.10	Coding using <i>im2bw</i> function.	29
3.11	Binary image.	29
3.12	A set of binary number of the binary image.	29
3.13	Median filter coding.	31
3.14(a)	Original Gray scale Image	32
3.14 (b)	Image of 40% additive impulse	32
3.14(c)	Image of Processed Image using 3X3 Median Filter	32
3.14 (d)	Image of Processed Image using 5X5 Median Filter	33
3.15	Coding for erosion and dilation filter.	35
3.16	Image erosion and dilation.	35

3.17	Binary number of the image.	36
3.18	The square build by imdilate function.	37
3.19	Coding for eroding the image.	38
3.20	Image after eroding process	38
3.21	Erosion and Dilation Coding	40
3.22	Image after erosion and dilation process.	40
3.23	Coding for segmentation process.	42
3.24	Segmentation Output Image	43
3.25	Coding of Canny Edge Detection	45
3.26	Output image of Canny Edge Detection	45
3.27	Face Detection Coding	47
3.28	The rectangle of the Face Region	48
3.29	Face Detection Image.	48
3.30	Eye Detection Coding	49
3.31	Eye Detected image	50
3.32	Quick GUIDE Start Window	51
3.33	Layout Editor for GUI Design	52
3.34	GUI Design	53
3.35	GUI Layout after Debug	54
3.36	Overall Process of Eye Detection	55
4.1	Original Face Image	56
4.2	Face Image for Threshold Process (Black& White)	57
4.3	Face Region Detected	58
4.4	Face Image after Filtering Process	59
4.5	Eye Detected Image	60
4.6	Image of the Eye Detection Process	61
4.7	GUI	62

## LIST OF APPENDICES

Appendix A	Eye Detection Programming	69
Appendix B	GUI Programming	73
Appendix C	Eye Detection Test Result	82

## **CHAPTER 1**

### **INTRODUCTION**

#### **1.1 Background Study**

Human eyes detection has been a fundamental and challenging problem for computer vision area. It can be used to ease the problem of finding the locations of other facial features for recognition task and human-computer interaction purposes as well. [6] Eye detection is a process where the position of the eyes is extracted from the face images. Eye detection is one of the applications in image processing.



In electrical engineering and computer science, image processing is any form of signal processing for which the input is an image, such as photographs or frames of video. The output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional (2D) signal and applying standard signal-processing techniques to it. [12]

This project is involved image processing method that includes threshold, filtering, feature extraction, segmentation and detection method. In order to detect the position of the eye, the face region must be initializing first. This process is called face detection. Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces. [11]

Early face-detection algorithms focused on the detection of frontal human faces, whereas newer algorithms attempt to solve the more general and difficult problem of multi-view face detection. That is, the detection of faces that is either rotated along the axis from the face to the observer in-plane rotation, or rotated along the vertical or left-right axis or both. [11] This project is use MATLAB software and all the programming of the eye detection system is written in the M-file (editor). MATLAB is a high-performance language for technical computing. The name MATLAB stands for matrix laboratory. [18] All the process in build this eye detection system must followed the step of image processing process. Beside that, the programming will be test with many face images to make sure the system is suitable to be applied for all face images.

## 1.2 Problem Statement

Eye detection system is important in human identification that can improve today identification technique that only involves the fingerprint as the input to identify people. The aim of this project is to detect the eye from the face image. This involved image processing method that produced the enhanced image. Eye detection is the beginning of the iris recognition system that is very effectively to be applied in security system. Eye detection is applicable in security system and drowsiness detection. Security is the degree of protection against danger, loss and criminal.

Today security systems mostly use password security system as a protection system. A password is a series of characters or a short phrase used to protect access to a system or file. The password security has been applied for bank card and door lock accessing. The password usually contained six characters long. Every people will build a password for their bank card and usually a big company will create a password that will be used to enter the company. People must remember the entire password. A problem will occur when the people forget the password or the password has been stolen. The weaknesses of this system can big problem for example another person can access the bank card and the company without any permission.

So this project, eye detection is one of the solutions of this problem. This project can be proceeded to produce iris recognition that can be a new security system. This system is more safety and effective because everybody has a difference iris. The people do not have to remember anything, just put the eye near the iris detector and they can access the bank card, door or others application. This eye detection can improve today security system to be more safety.

Beside that this eye detection is also crucial in drowsiness detection. Automotive vehicles clearly bring great value to society, providing a cost-effective means of transporting goods and people all over the world. Unfortunately, there are a corresponding number of automotive accidents and fatalities which continue to increase worldwide, prompting the World Health Organization to project automotive related deaths as the number three cause of death by the year 2010, up from number nine in 2002. Together, the auto industry has been working with governments around the world, from both a regulatory and technology standpoint, to reduce the number of automotive related deaths. Through increased driver education and awareness of potential hazards, the automotive industry is striving to promote accident avoidance and prevention to increase overall automotive safety. [4]

The drowsiness detection application is highly computationally intensive and it warns when a driver is “nodding off” or falling asleep behind the wheel. [4] These mean that the eye detection is very important for drowsiness detection to avoid accident.

### **1.3 Objective**

The objective of this project is to:

- i. To detect the human eyes from the frontal face image.
- ii. To enhance the face image.
- iii. Design GUI system to display image.

### **1.4 Scope of Project**

The scope of this project is limited to the preprocessing phases which include four major steps. The main focus of the scope is to develop the system of eye detection focusing on preprocessing phases. This project is developed using MATLAB version 7.1. The additional scopes of project to be considered are:

- i. Only use offline database.
- ii. Use frontal face image for eye detection.
- iii. Eye detection using MATLAB.
- iv. Use single face image at one time.



Figure 1.1 Example of Frontal Face Image

## 1.5 Thesis Organization

Thesis is divided into five chapter essential chapters ranging from Chapter 1 until Chapter 5. The Chapter 1 gives an overview of the research conducted. It also discusses the problem statement, objective and the scope of the project. The problem statement is about the problems that occur in today life and the advantages of applying this project. The objective is about the aim that I want to achieve for this project and the scope of the project is about the limitation that I used in this project.

Meanwhile, Chapter 2 is the review of the previous research works conducted by many researchers outside. All relevant researches taken from the technical paper, journal, and books are discussed in details.

Chapter 3 reveals the techniques and the algorithms that are used to perform in this project. It also discusses the process flow of the project in detail.

All the result of the process is detailed out in Chapter 4. The output image for each process is displayed in this chapter. There are also discussions based on the result.

Finally, the Chapter 5 concludes the entire thesis and there are also some recommendations to improve this project. In this chapter also stated that all the objective of this project has been achieved.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 An Overview of the project**

Human recognition has been studied for more than twenty year. It also becomes ones of the most attractive and challenging areas in pattern recognition and computer vision because there are a lot of potential commercial applications [9]. There many research that has been done about face recognition that involves eyes detection. Several method and technique implemented for each research to get the actual eyes detection result.

This project is used MATLAB software. MATLAB is a high-performance language for technical computing. MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation Algorithm development Data acquisition Modeling, simulation, and prototyping Data analysis, exploration, and visualization Scientific and engineering graphics Application development, including graphical user interface building. [18]

MATLAB features a family of add-on application-specific solutions called toolboxes. The toolboxes allow us to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. There is also an image processing toolbox in MATLAB. [18]

The Image Processing Toolbox is a collection of functions that extend the capability of the MATLAB numeric computing environment. The toolbox supports a wide range of image processing operations, including Spatial image transformations Morphological operations Neighborhood and block operations Linear filtering and filter design Transforms Image analysis and enhancement image registration deblurring region of interest operations. [18]

Computer vision is the science and technology of machine that see. As a scientific discipline, computer vision is concern with the theory for building artificial systems that obtain information from images. The image data can take many forms, such as a video sequence, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technology discipline, computer vision seeks to apply the theories and models of computer vision to the construction of computer vision systems. [9]



The examples of the applications of computer vision systems include systems for controlling processes, detecting events, organizing information, modeling objects or environments and interaction that are a computer-human interaction. [9] Real Time Detection and Tracking of Human Eyes in Video Sequences from Zafer Savas used video sequence. “Eye tracking and eye movement based interaction using computer vision techniques have the potential to become an important component in future perceptual user interfaces. In general, the term “eye detection” is widely used when static face images are of concern and the main aim is to find the face region which contains both eyes, and eye tracking term is used referring to the process of continuously detecting eyes in video sequences which contains only face images”. [25]

“In this thesis the term eye tracking means real-time, continuously detection of human eyes individually. Eyes are the most important features of human face. So effective usage of eye movements as a communication technique in user-to-computer interfaces can find place in various application areas. Our cost effective, robust, fast and real-time tracking solution to this problem depends on simple computer vision techniques with easy to implement setup and has the property of tracking with scale and rotation invariance properties”. [24]

## **2.2 Face and Eye Detection**

Face recognition is a field of biometrics together with fingerprint recognition, iris recognition, and speech recognition and so on. Automatic extraction of human head, face boundaries, and facial features is critical in the areas of face recognition, criminal identification, security, surveillance systems, human computer interfacing, and model-based video coding. In general, the computerized face recognition includes four

steps.[19] First, the face image is enhanced and segmented. Second, the face boundary and facial features are detected. Third, the extracted features are matched against the features in the database. Fourth, the classification into one or more persons is achieved.

In order to detect faces and find the facial features correctly, researchers have proposed a variety of methods which can be divided into two categories. One is based on gray-level template matching, and the other is based on computation of geometric relationship among facial features. Face detection is the most important part of face identification and it is difficult due to varying of illumination, pose of head, and face expression.[19]

Eye detection is also the most important and critical task of face detection [6, 7]. In the paper, they propose a new approach of eye detection using gray histograms in eye candidate regions. Their eye detection algorithm works on face region determined by the method presented in [17]. In this project the face region is determine first. The eye region is detected after that. This method has been used by other researcher in their project for example Antonio Haro Myron Flickne and Irfan Essa. They stated in their project paper that the faces are usually chosen to be located first because they occupy more of the image than their features. [10]

The structure image is defined as the binary image which contains the structure feature of human face. Then, a binary eye pair template is used to find eye pair candidates in the image. All the eye pair candidates are then rescaled to a fixed size and sent to an SVM classifier that verifies the candidates and obtains real eye pairs.[21] This articles give some information for eye detection methodology and also about the filtering process. Qiong Wang and Jingyu Yang used binary template matching, SVM and variance filter in their project. Beside that they also use a face database that is from BioID face database as their input image.[14]

### **2.3. Image Analysis**

Image analysis is the extraction of meaningful information from images, mainly from digital images by means of digital image processing techniques. Image analysis tasks can be as simple as identifying a person from their face. [12] This project used the offline database that contains the frontal face image. With the development of computer vision, computer graphic and pattern recognition especial with the deep research of face detection and face animation, many research institutes have built kinds of face databases to make experiment about correlative research and test algorithms.

The existing face databases almost are 2D images or videos applied in research of face detection, face recognition, face tracing, face feature selection, face expression and illumination. Faces in those databases vary in race, age, sex, pose, expression and illumination. [14]

### **2.4 Image Acquisition**

According to Gonzalez C.R and Richard E.W (2002), “There are many ways where image can be acquired. The decision to use any technique is solely based on an individual.” Image acquisition is the first step to get the face image using acquisition device. There two type of image acquisitions that are offline and online process. Offline image acquisition is acquired image using digital cameras, scanner and others, while online image acquisition is acquired image using the webcam devices or the other devices that can capture image for the real time system.

This project is an offline system where the analysis is only processes the image from the face image database. In this project, the image is acquired from the offline database that contains the collection of the face image. The entire image is in RGB format. The image acquisition process is starting by inserting the face image into the system by creating the link between the system and the database. In this project, I only used the 2 dimension image for the all process. The image is load from the database and displayed in the figure windows.

#### **2.4.1 Image Enhancement**

Image enhancement is performed in a digital image to improve its overall quality. This is done using noise reduction in which it attempts to recover an underlying perfect image for a degraded image [8]. Many techniques applied in noise reduction are by extracting the summation of neighborhood pixels with the density of a pixel in the window at a given location. [23] This technique is applied using a window with 3 x3 pixels. The gray level values are started as G1, G2, G3, G4, G5, G6, G7, G8, and G9. Meanwhile the deviational values for each pixel are given as D1, D2, D3, D4, D5, D6, D7, D8, and D9. Therefore the linear spatial for window S is

$$S = G1D1 + G2D2 + \dots..G9D9 \quad (2.1)$$

### 2.4.2 Mean Filtering

The concept of mean filtering is produce the average pixel value of a given window with size of  $N \times N$ .

$$S = \frac{1}{N^2} \sum_{(r,c) \in W} d(r, c) \quad (2.2)$$

$N^2$  is the amount of pixels in window  $N \times N$ .

The mean filter is a simple sliding-window spatial filter that replaces the center value in the window with the average (mean) of all the pixel values in the window. Mean shift filtering is a data clustering algorithm commonly used in computer vision and image processing. For each pixel of an image (having a spatial location and a particular color), the set of neighboring pixels (within a spatial radius and a defined color distance) is determined. For this set of neighbor pixels, the new spatial center (spatial mean) and the new color mean value are calculated. These calculated mean values will serve as the new center for the next iteration. The described procedure will be iterated until the spatial and the colors (or grayscale) mean stops changing. At the end of the iteration, the final mean color will be assigned to the starting position of that iteration.

### 2.4.3 Median Filtering

The median filtering is under the category of morphological filter. Median filtering is known to be a good filter in removing shot noise or known as salt-and-pepper noise while preserving some edges. When it comes to dense noise, it is not advisable to use this technique because median filtering degrades thin lines and small features. [8] These techniques can be applied in eye detection system:

$$F(x,y) = \text{median} \{ g(s,t) \} \quad (2.3)$$

Median filtering is better than mean filtering due to lesser blurring effect to the image compared to mean filtering. Median filtering is performed by acquiring the median value of neighborhood pixel in a given dimension like 3 x3 pixels. The main purpose of the median filtering is to ensure that all the neighborhood pixels contain minimal differences in pixel value.

Median filtering is a simple and effectively. Median filter is function to smooth the image and it also can eliminate certain of the image's imperfections. However, it unlike a linear low-pass filter, which inevitably adds a blur around the contours, it better preserves the sharp variations of the image. The equation of the median filter is  $m(k,l) = \left( \frac{MN+1}{2} \right)$ . Let  $\{a(k,l)\}$  be an image. The median filter associate the mean value  $m(k,l)$  with the point with coordinates  $(k,l)$ , in the  $(M \times N)$  rectangular window, centered on  $(k,l)$ . If we assume  $N$  and  $M$  to be odd, and if  $u(n)$  denotes the sorted sequence ( $u(n) \geq u(n-1)$ ) obtained from the array  $[a(I,j)]$  ( $M \times N$ ) where the equation of

i

i is  $i \in \{(k - \frac{(M-1)}{2}), \dots, k + \frac{(M-1)}{2}\}$  and  $j \in \{(\ell - \frac{(N-1)}{2}), \dots, \ell - \frac{(N+1)}{2}\}$ .

i) The median filter algorithm:  $m(k, l) = (\frac{(MN+1)}{2})$ .

ii)  $i \in \{(k - \frac{(M-1)}{2}), \dots, k + \frac{(M-1)}{2}\}$  and  $j \in \{(\ell - \frac{(N-1)}{2}), \dots, \ell - \frac{(N+1)}{2}\}$

iii)  $j \in \{(\ell - \frac{(N-1)}{2}), \dots, \ell - \frac{(N+1)}{2}\}$

## 2.5 Thresholding

Thresholding is an essential technique in image processing. The main objective of the thresholding process is to separate the pixels of the given image into two classes, namely objective and background. The technique is conducted whereby each grayscale pixels which contain values above the given threshold are put into a one class and vice versa. In short, the whole process of thresholding is converting the image into a binary image. This is because mathematical manipulation can be done using values since each pixel values is either 1 or 0 at a given location.[23]

The gray scale level values below or equal to this threshold are classified as background, whereas the values above this threshold values are classified as object. Generally, thresholding can be divided into 2 main groups which is global and local thresholding value for the whole image. Therefore at any point  $(x, y)$  for which  $f(x,y) > T$  is called an object, otherwise it is called background. Meanwhile, in local thresholding,

the threshold values changes dynamically across an image. The local thresholding method is used when is poor illumination condition or when the background is uneven. In general, the global threshold technique can be applied using the following algorithm:

$$g = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases} \quad (2.4)$$

where T is threshold value.

### 2.5.1 Mean Thresholding

The mean thresholding is performed by calculating the total number of values at each point of the histogram. Each value at histogram represents the values of each pixel at a given location. The values are added and divided by the total number of pixels to produces the mean value. This mean value is then used as the global threshold value.

$$T = \frac{\sum_{j=1}^n I(j)}{n} \quad (2.5)$$

*T* is the threshold value.



### **2.5.2 Median Thresholding**

The median thresholding is conducted in a similar way of mean thresholding but instead of acquisition the mean values for T, median values are acquired. The histogram values is arranged in increasing order,  $X(1) < X(2) < X(3) < \dots < X(n)$  and the median is identified Ron de Beer (1997). After acquisition the median values or now known as threshold value, T, the binary conversion process is conducted.

### **2.6 Feature Extraction Approach**

Feature extraction is another method that can be used during locating the eye from the face image. A good process of feature extraction can provide clear view of the pattern available in an image. A feature extraction technique was researched by Fletcher L.A, Kasturi R (2001) whereby morphological approach, dilation and erosion has been applied in their project. [10] The binary approach was used in order to differentiate the background and fore ground. After performing this, the values in neighborhood pixels are identified and if the values exceed a given region, new pixel value is applied. By performing this process, the newer image is more prominent then the previous image and it is easier to locate the pattern. In the feature extraction, segmentation process is applied.

Segmentation refers to the process of partitioning a digital image into multiple segments that is a set of pixels. The goal of segmentation is to simplify and change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries in images. [13] More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. [13]

## **CHAPTER 3**

### **METHODOLOGY**

#### **3.1 An Overview of the project**

The aim of this project is to develop an eye detection system that can be applied as a drowsiness detection system. The focus will be placed on designing a system that will accurately find the eye position. Detection process involves a sequence of a face image, and the observation of eye location. The analysis of face images is a popular research area with applications such as face recognition, virtual tools, and human

identification security systems. This project is focused on the localization of the eyes, which involves looking at the entire image of the face, and determining the position of the eyes, by developed the image processing algorithm. After insert a facial image, pre-processing is first performed by binarizing the image. The top and sides of the face are detected to narrow down the area of where the eyes exist. Using the sides of the face, the centre of the face is found, which will be used as a reference when comparing the left and right eyes.

This project is generated in MATLAB using the image processing toolbox. MATLAB is a short form for "matrix laboratory" that refers to both the numerical computing environment and to its core programming language. MATLAB created by The Math Works. MATLAB allows one to easily manipulate matrices, plot functions and data, implement algorithms, create user interfaces, and interface with programs in other languages. Although it specializes in numerical computing, an optional toolbox interfaces with the Maple symbolic engine, making it a full computer algebra system. It is used by more than one million people in industry and academia and runs on most modern operating systems, including Windows, Mac OS, Linux and UNIX.[15]

Image processing process will be applied in this project. Digital image processing is referring to processing of two dimensional pictures by a digital computer. The aim of the image processing is to extract important futures from image data from which a description, or understanding of the scene can be provided by the machine. A digital image is an array of real or complex numbers represented by a finite number of bits. An image given in the form of a photograph or chart is first digitized and stored as a matrix of binary digits in computer memory. Figure 3.1 is the block diagram for eye detection process.

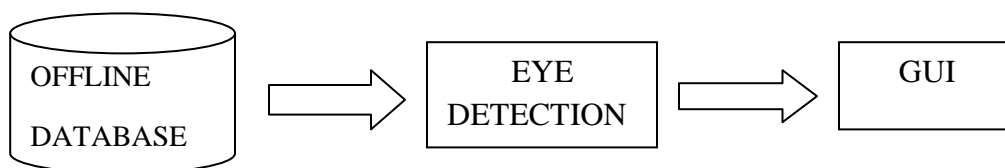


Figure 3.1: Block Diagram for Eye Detection process.

### 3.2 Eye Detection Process

The first process, an input image that is a face image will be extracted from an offline database. Database is a collection of data ease of storage, retrieval, searching, and sorting by computerized mean. This project is only use an offline database. Offline database is unavailable databases that become offline by explicit user action and remain offline until additional user action is taken. The entire face image is store in the database. The input of this project is a face image. The image will be displayed in acquisition process in image processing. The face image will be used for the eye detection process.



Figure 3.2: Example of Input Image

Eye detection process is involved an image processing process. This process included pre-processing process that is threshold and filtering process. After these pre-processing, the process is continued with feature extraction process. Then, the face region is defined from the overall image. Next, the eyes are determined using the segmentation process from the face image. This involved the detection of eyes point. The next process is to design GUI. The position of the eye detected displayed in GUI. The eye detection process is followed the detection process in figure 3.3.

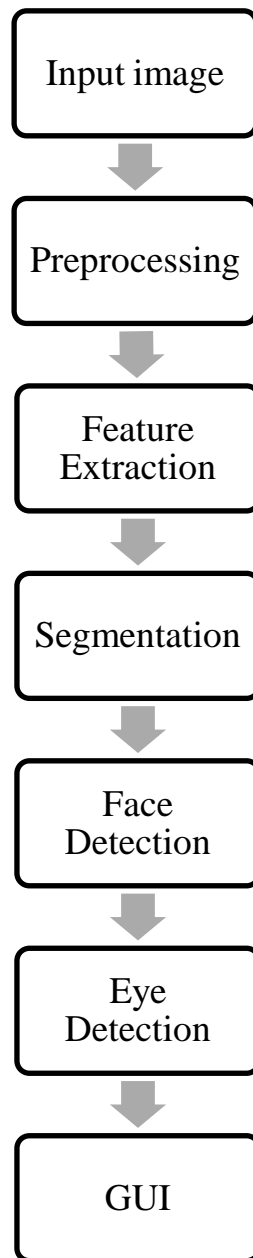


Figure 3.3: Eye Detection Process

### 3.3 Preprocessing

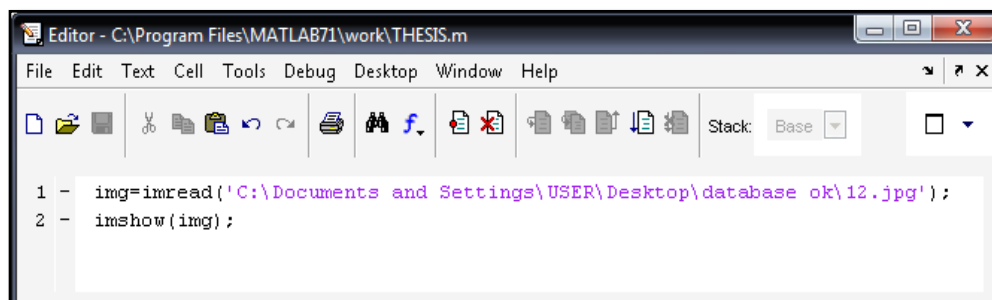
Preprocessing is a preliminary processing of data in order to prepare it for the primary processing or for further analysis. The term can be applied to any first or preparatory processing stage when there are several steps required to prepare data for the user.

The first step of the eye detection system is to insert the frontal face image. In this process I used an offline face database that contains several number of human face images. For this project, I made a limitation of the face image pose that is only in frontal face image is used. This made the process more easily to detect the face and eye location. After inputting a face image from the offline database, the process is proceed with the next pre-processing process that is threshold process or also called binarization process.

Basically, the original face image is followed the skin color. All people have a difference of the skin color. This can be assumed that face image is a RGB image. In image processing process that is important to convert the image into a grayscale and black and white color to make all the image is the same. There is needed the process to change the image into black and white color. This is to make the process of the face and eye detection can be proceed and suitable for each image. The process of this step is called threshold process.

### 3.4 Load and Show Image

This project used the offline database that contains the frontal face image. The entire face image is in RGB format. The face image is inserted into the eye detection system using the MATLAB function for read and display image. *Imread* function is used to read and load a face image from the database while *imshow* function is used to display the output image in the figure. Figure 3.4 below is the coding for read and display the original image from the database. Figure 3.5 is displaying the output image of the coding. All the coding is written in MATLAB M-file.



```
Editor - C:\Program Files\MATLAB71\work\THESIS.m
File Edit Text Cell Tools Debug Desktop Window Help
[Icons] Stack: Base
1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - imshow(img);
```

Figure 3.4: Coding of load and show image.

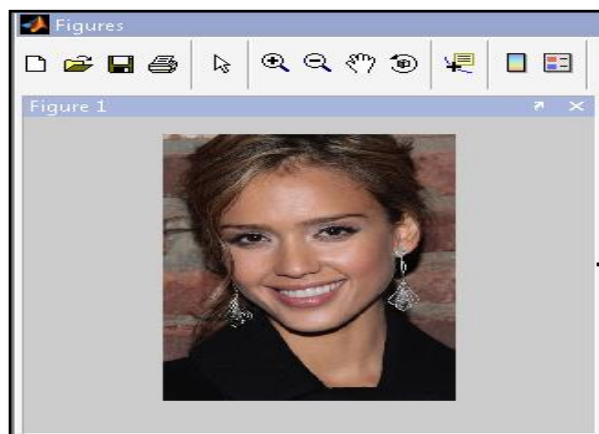


Figure 3.5: Image Display in Figures Window



### 3.5 Threshold Process

Threshold process is a process to convert the face image into a form of binary number that is 1 and 0. This entire number is considering as black or white. A binary image is an image in which each pixel assumes the value of only two discrete values. In this case the values are 0 and 1, 0 representing black and 1 representing white. With the binary image it is easy to distinguish objects from the background. The grayscale image will be converted into a binary image by threshold process. The output binary image has values of 0 or black for all pixels in the original image with luminance less than level and 1 white for all other pixels.

Thresholds are often determined based on surrounding lighting condition. The threshold value will be determined by observing the process for a difference face image. From the observing a certain threshold value will be finding to be effective. The criteria used in choosing the correct threshold was based on the idea that the binary image of the face image should be majority white, allowing a few black blobs from the eyes, nose and lips.

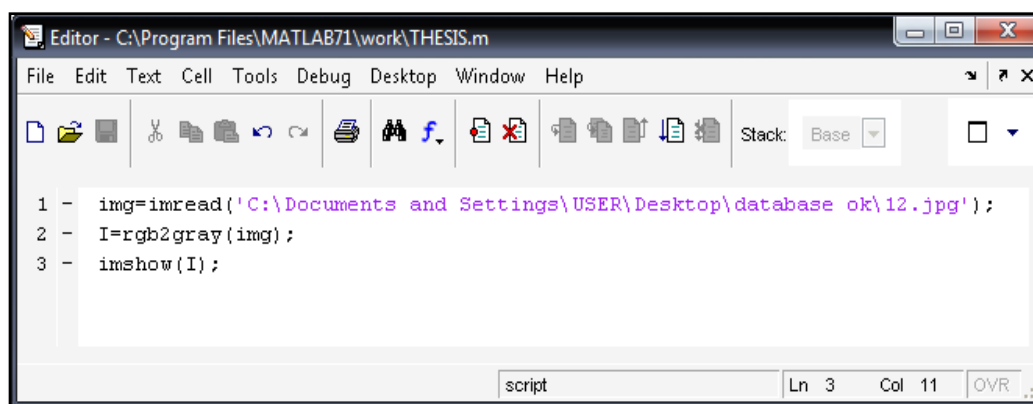


Figure 3.6: Binary Image

This picture is the example of the threshold image. Figure 3.6 is an image of an optimum binary image for the eye detection algorithm, in that image the background is uniformly black, and the face is primary white. This will allow finding the edges of the face. Using the sides of the face, the centre of the face will be found, which will be used as a reference when comparing the left and right eyes.

In this project I used Otsu threshold that is listed in the MATLAB library. If the input image is not a grayscale image, *im2bw* function converted the input image to grayscale, and then converted this grayscale image to binary by thresholding. For this project, the technique is converted the RGB image to black and white color. The output image replaces all pixels in the input image with luminance greater than level with the value 1 (white) and replaces all other pixels with the value 0 (black). The level is specified in the range [0, 1], regardless of the class of the input image. The function *graythresh* can be used to compute the level argument automatically.

In a process to convert the RGB face image into grayscale image, I applied the *rgb2gray* function. This function is directly converting the RGB image into grayscale image. Figure 3.7 show the coding to convert RGB image to grayscale image and figure 3.8 is the output image of the process.

The image shows a screenshot of a MATLAB script editor window. The title bar reads "Editor - C:\Program Files\MATLAB71\work\THESIS.m". The menu bar includes "File", "Edit", "Text", "Cell", "Tools", "Debug", "Desktop", "Window", and "Help". Below the menu bar is a toolbar with various icons for file operations and editing. The main text area contains three lines of MATLAB code:

```
1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');  
2 - I=rgb2gray(img);  
3 - imshow(I);
```

The status bar at the bottom indicates the current file is "script", and the cursor is at line 3, column 11. There is also an "OVR" indicator.

Figure 3.7: Coding to Convert RGB image into grayscale.

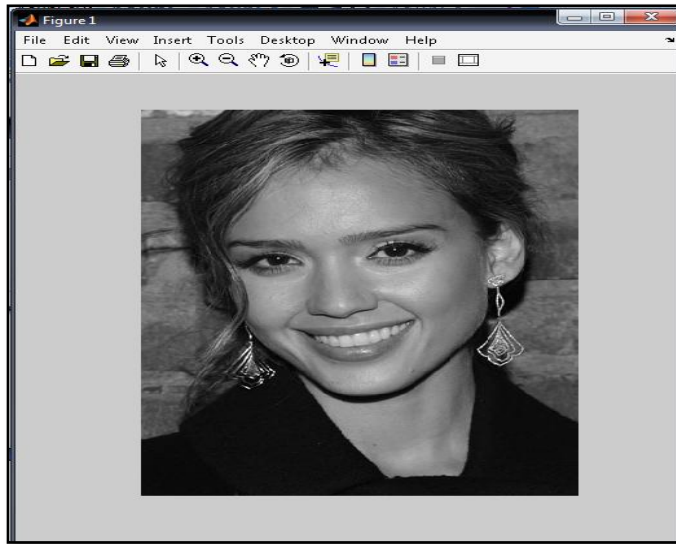


Figure 3.8: Grayscale Image.

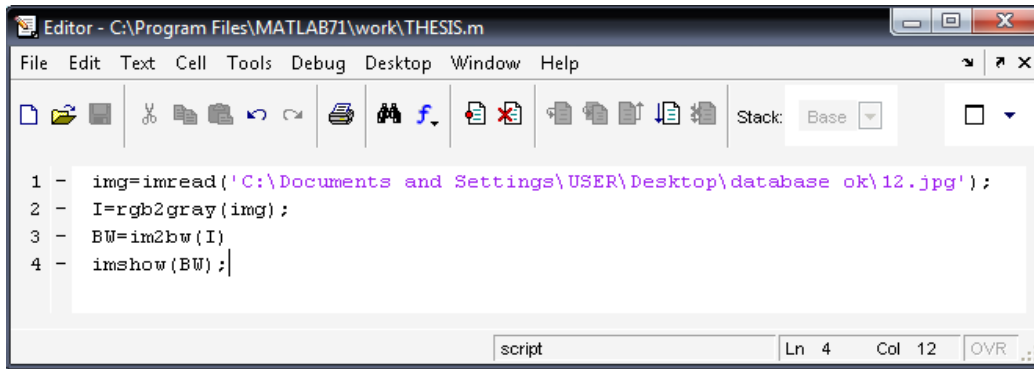
The same coding is proceeding for the next process that is to convert the grayscale image into the binary image. The function of Otsu threshold is applied to convert the grayscale image into the binary image. The Otsu function is *graythresh*. Figure 3.9 is shown the coding for this process. Beside that, there is a way to directly convert the grayscale image into binary image that is by applying the *im2bw* function. This will make the programming simpler. The coding using *im2bw* function is shown in figure 3.10. Both functions will develop the same output that is a binary image as in the figure 3.11.

```

1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - level = graythresh(img);
3 - BW = im2bw(img, level);
4 - imshow(BW)

```

Figure 3.9: Coding for converting grayscale to binary image.



```

1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - I=rgb2gray(img);
3 - BW=im2bw(I)
4 - imshow(BW);

```

Figure 3.10: Coding using *im2bw* function.

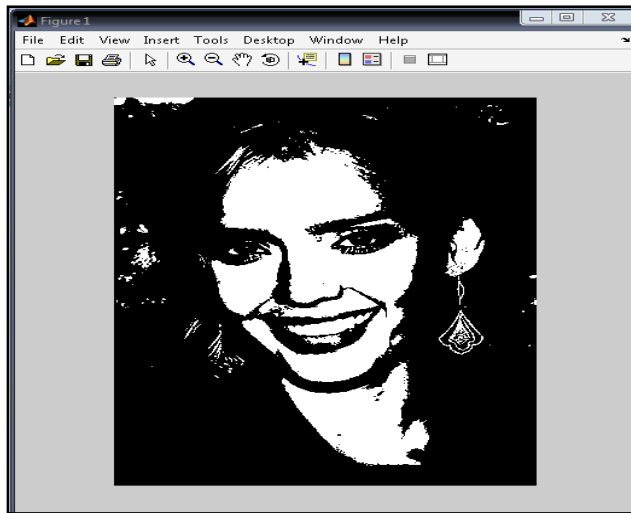
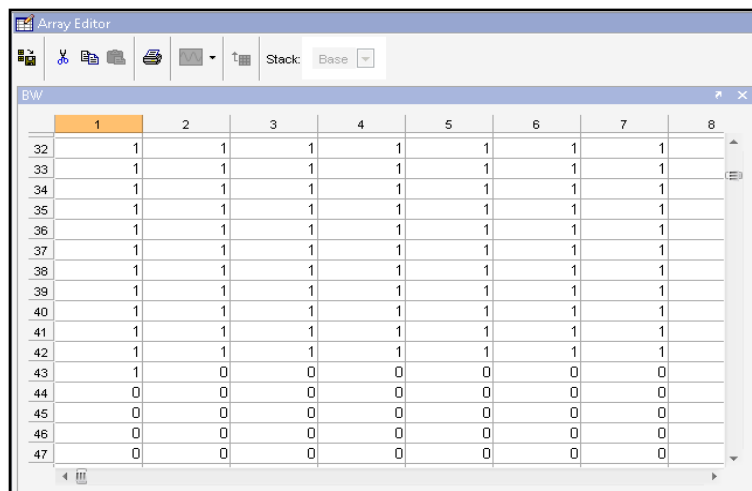


Figure 3.11: Binary image.



	1	2	3	4	5	6	7	8
32	1	1	1	1	1	1	1	1
33	1	1	1	1	1	1	1	1
34	1	1	1	1	1	1	1	1
35	1	1	1	1	1	1	1	1
36	1	1	1	1	1	1	1	1
37	1	1	1	1	1	1	1	1
38	1	1	1	1	1	1	1	1
39	1	1	1	1	1	1	1	1
40	1	1	1	1	1	1	1	1
41	1	1	1	1	1	1	1	1
42	1	1	1	1	1	1	1	1
43	1	0	0	0	0	0	0	0
44	0	0	0	0	0	0	0	0
45	0	0	0	0	0	0	0	0
46	0	0	0	0	0	0	0	0
47	0	0	0	0	0	0	0	0

Figure 3.12: A set of binary number of the binary image.

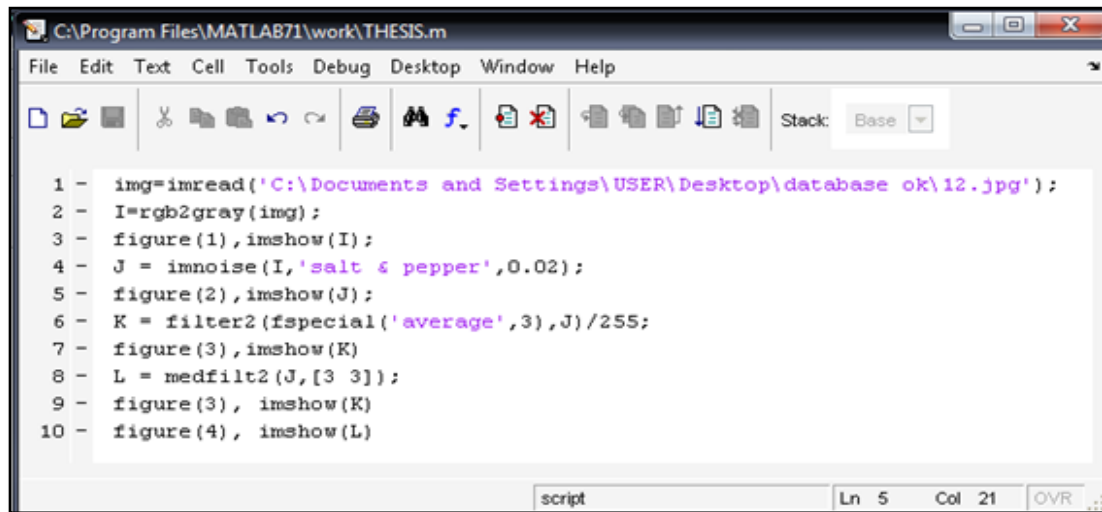
After a set of binary number for the face image performed after the threshold process, the process will proceed with filtering process. Filtering process is a process to remove the noise of the image. The filtering process will be repeated until a clear image appears. The removal of noise in the binary image will produce an enhanced and more quality image.

### **3.6 Filtering Process**

#### **3.6.1 Median Filter**

This project applied Median filter method to remove the noise from the image. Median filter is ideal for removing the noise. This type of manifests itself as outlier pixel intensities, completely independent of signal content. Median filtering is similar to using an averaging filter, in that each output pixel is set to an average of the pixel values in the neighborhood of the corresponding input pixel. However, with median filtering, the value of an output pixel is determined by the median of the neighborhood pixels, rather than the mean.

The median is much less sensitive than the mean to extreme values (called outliers). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. The *medfilt2* function implements median filtering. The output image after applying the median filter is shown in figure 3.14.



```

C:\Program Files\MATLAB71\work\THESIS.m
File Edit Text Cell Tools Debug Desktop Window Help
1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - I=rgb2gray(img);
3 - figure(1), imshow(I);
4 - J = imnoise(I, 'salt & pepper', 0.02);
5 - figure(2), imshow(J);
6 - K = filter2(fspecial('average', 3), J)/255;
7 - figure(3), imshow(K);
8 - L = medfilt2(J, [3 3]);
9 - figure(3), imshow(K);
10 - figure(4), imshow(L)
script Ln 5 Col 21 OVR

```

Figure 3.13: Median filter coding.

Figure 3.14b show that the image corrupted with 40% impulse noise and Figure 3.14c and 3.13d show the effect of passing the corrupted image through a 3x3 and 5x5 median filter, respectively. This is an example where the quantitative fidelity metrics. This can be seeing in Figure 3.14d, the images look better than Figure 3.14c. The example of the median filter action is shown below.

Median filter is actually a specific form of a rank filter, where the  $i^{\text{th}}$  pixel intensity in the sorted list of neighborhood pixel is chosen as the output. Thus for the case of the median filter, the index  $i$  is simply the middle point. If the neighborhood is sorted in ascending order, and the index  $i$  is chosen to be the first element, then the minimum filter. Likewise, if the last element is chosen, the filter is a maximum filter.

Passing an image through a minimum filter is known as erosion that is dark areas in the image become more pronounced, and oftentimes images are repeatedly passed through erosion filter to aggrandize the effect. [23]

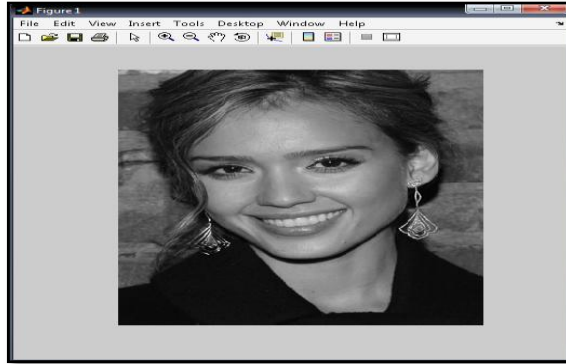
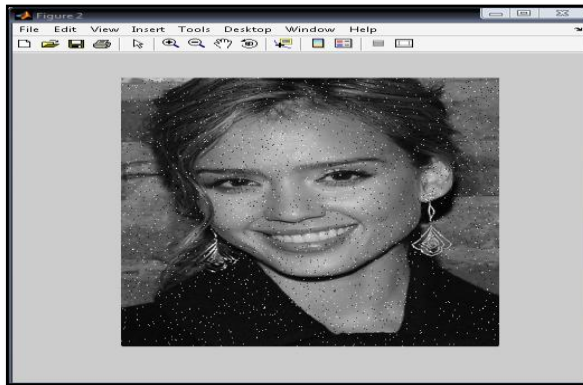
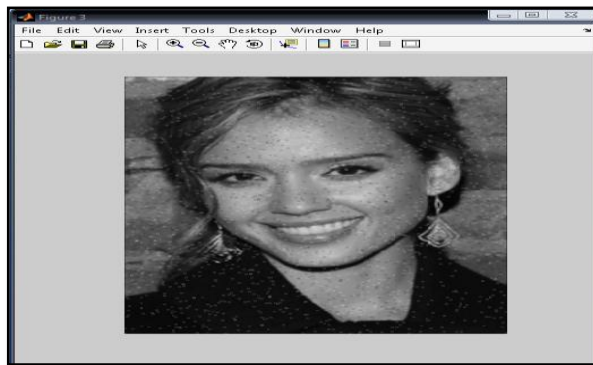


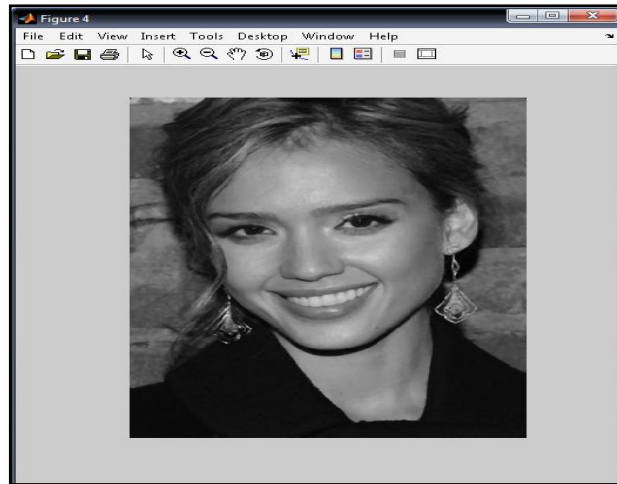
Figure 3.14: a) Original Gray scale Image



b) Image of 40% additive impulse



c) Processed image using 3x3 median filters.



d) Processed Image using 5x5 median filters.

### 3.6.2 Morphology (Dilation and Erosion)

This project also implemented the morphology technique. Morphology technique is the study of the shape and form of objects. Morphological image analysis can be used to perform object extraction, image filtering operations, such as removal of small objects or noise from an image, image segmentation operations, such as separating connected objects, and measurement operations, such as texture analysis and shape description. [18]

Dilation and erosion are two fundamental morphological operations. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. Dilation and erosion operations, the state of any given pixel in the



output image are determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as dilation or erosion. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. [18]

Operation	Rules
Dilation	The value of the output pixel is the maximum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to the value 1, the output pixel is set to 1.
Erosion	The value of the output pixel is the minimum value of all the pixels in the input pixel's neighborhood. In a binary image, if any of the pixels is set to 0, the output pixel is set to 0.

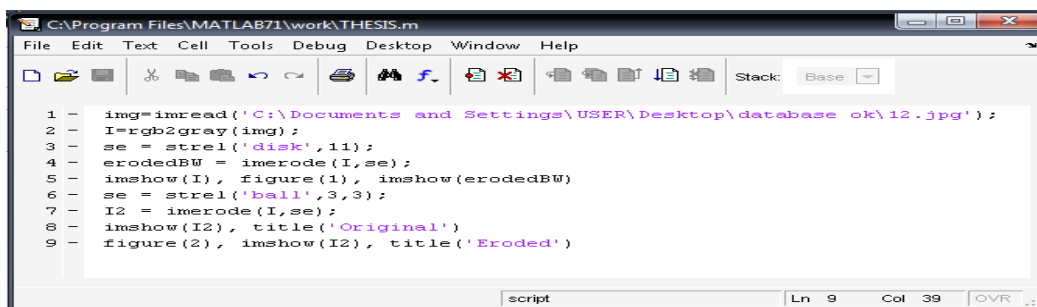
Table 3.1: Dilation and Erosion Rules

Operation	Rules
Dilation	<p>Pixels beyond the image border are assigned the minimum value afforded by the data type.</p> <p>For binary images, these pixels are assumed to be set to 0. For grayscale images, the minimum value for uint8 images is 0.</p>
Erosion	<p>Pixels beyond the image border are assigned the maximum value afforded by the data type.</p> <p>For binary images, these pixels are assumed to be set to 1. For grayscale images, the maximum value for uint8 images is 255.</p>

Table 3.2: Dilation and Erosion Rules for Grayscale Image

Dilation is an application of maximum filter, where the bright areas in the images are emphasized. The global natural of image is remain the same, while the dark (erosion) or bright (dilation) regions dominate, the degree of which is dependent on how many time the filter is applied. All the programming of the eye detection system is written in MATLAB and also use the functions that are include in the MATLAB function references or library.

Figure 3.16a and 3.16b show the image for erosion and dilation. The two operations are coupled, as in image segmentation, where the goal is to deconstruct an image into sections that ideally map to objects of interest. Dilation followed by erosion can be used in segmentation process to enhance the contrast of portions of an image.



```

1 - img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - I=rgb2gray(img);
3 - se = strel('disk',11);
4 - erodedBW = imerode(I,se);
5 - imshow(I), figure(1), imshow(erodedBW)
6 - se = strel('ball',3,3);
7 - I2 = imerode(I,se);
8 - imshow(I2), title('Original')
9 - figure(2), imshow(I2), title('Eroded')

```

Figure 3.15: Coding for erosion and dilation filter.

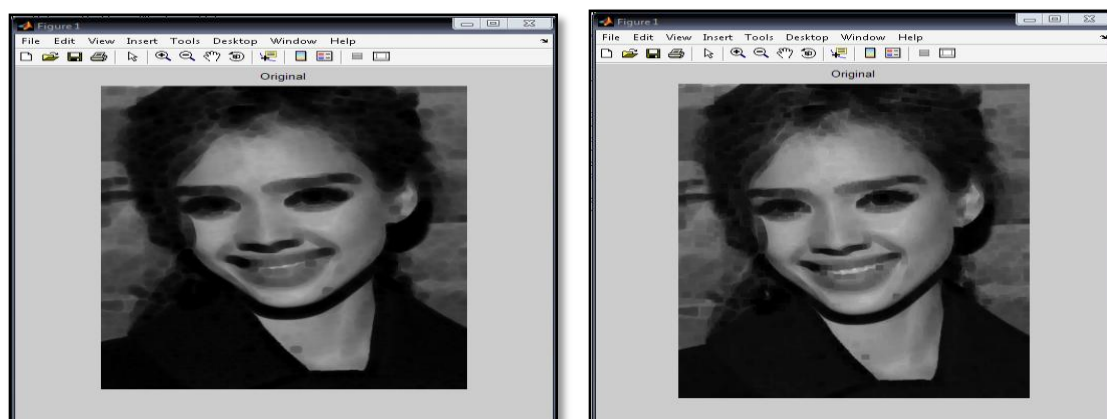


Figure 3.16: Image erosion and dilation. a) Image after two passes of a 3x3 minimum filter. b) Image after two passes of 3x3 maximum filter.

### 3.6.3 Dilating an Image

In order to dilate an image, *imdilate* function is used. The *imdilate* function accepts two primary arguments. The input image to be processed is grayscale, binary, or packed binary image. A structuring element object, returned by the *strel* function, or a binary matrix defining the neighborhood of a structuring element *imdilate* also accepts two optional arguments, PADOPT and PACKOPT. The PADOPT argument affects the size of the output image. The PACKOPT argument identifies the input image as packed binary. Packing is a method of compressing binary images that can speed up the processing of the image. This example dilates a simple binary image containing one rectangular object.[18]

```
BW = zeros(9,10);
```

```
BW(4:6,4:7) = 1
```

BW =									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	1	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 3.17: Binary number of the image.

To expand all sides of the foreground component, the example uses a 3-by-3 square structuring element object.  $SE = strel('square',W)$  creates a square structuring element whose width is  $W$  pixels.  $W$  must be a nonnegative integer scalar. In order to dilate the image, pass the image  $BW$  and the structuring element  $SE$  to the *imdilate* function. Dilation adds a rank of 1's to all sides of the foreground object.

$$BW2 = \text{imdilate}(BW, SE)$$

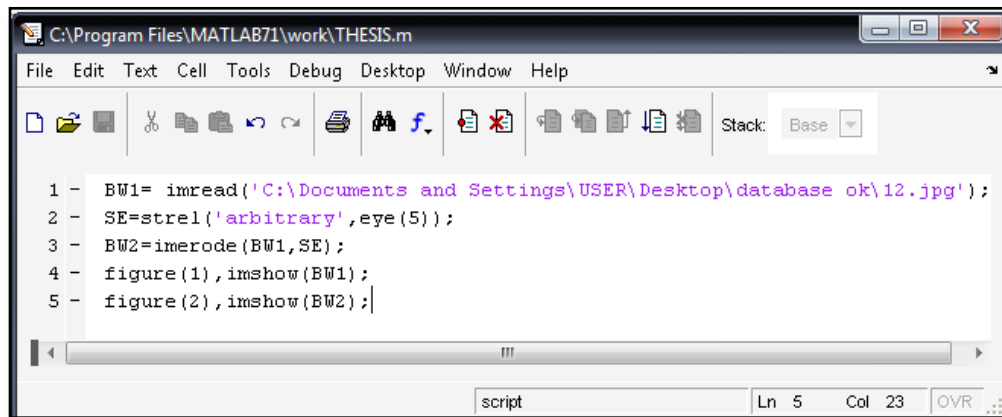
BW2 =									
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figure 3.18: The Square Build by *imdilate* function.

### 3.6.4 Eroding an Image

The *imerode* function is used to erode an image. The *imerode* function also accepts two primary arguments. The input image to be processed is grayscale, binary, or packed binary image. A structuring element object, returned by the *strel* function, or a binary matrix defining the neighborhood of a structuring element *imerode* also accepts three optional arguments PADOPT, PACKOPT, and M. The PADOPT argument affects the size of the output image. The PACKOPT argument identifies the input image as packed binary.

If the image is packed binary, *M* identifies the number of rows in the original image. The following example erodes the binary image *circbw.tif*. Read the image into the MATLAB workspace. The following codes creates a diagonal structuring element and then call the *imerode* function, passing the image *BW* and the structuring element *SE* as arguments.



```
C:\Program Files\MATLAB71\work\THESIS.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
1 - BW1= imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - SE=strel('arbitrary',eye(5));
3 - BW2=imerode(BW1,SE);
4 - figure(1),imshow(BW1);
5 - figure(2),imshow(BW2);
script Ln 5 Col 23 OVR
```

Figure 3.19: Coding for eroding the image.

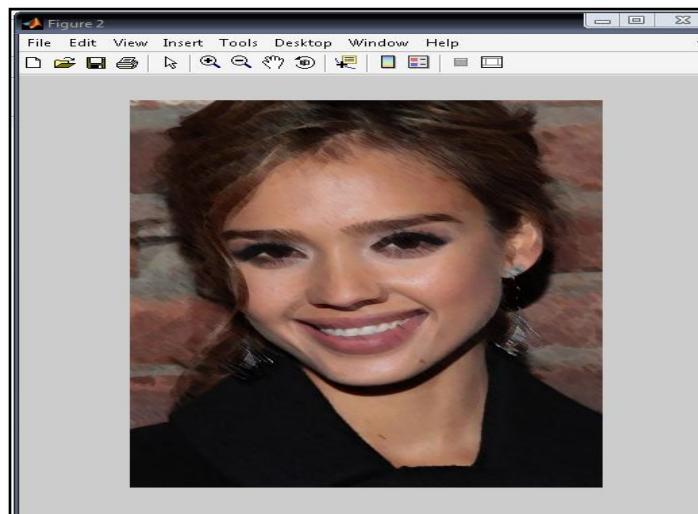


Figure 3.20: Image after eroding process

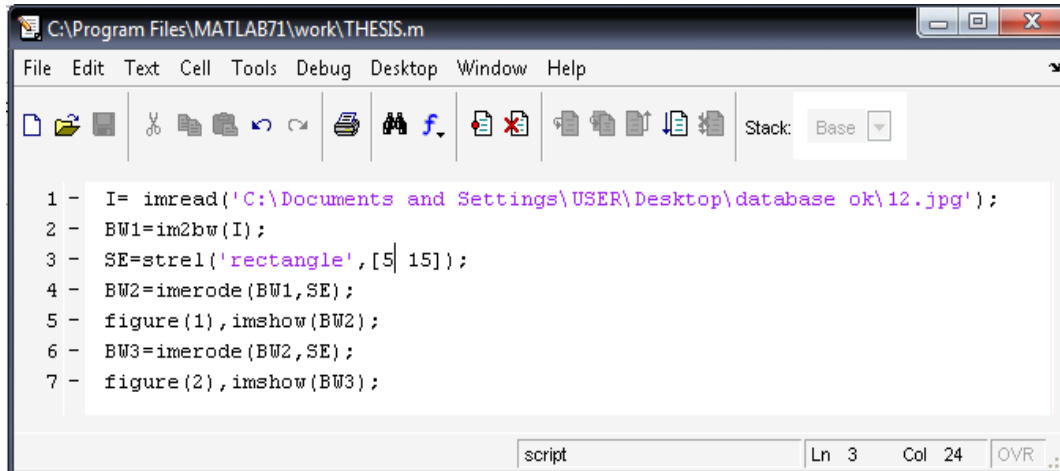
### 3.6.5 Combining Dilation and Erosion

Dilation and erosion are often used in combination to implement image processing operations. For example, the definition of a morphological opening of an image is erosion followed by dilation, using the same structuring element for both operations. The related operation, morphological closing of an image, is the reverse, which consists of dilation followed by erosion with the same structuring element. The following section uses *imdilate* and *imerode* to illustrate how to implement a morphological opening. The toolbox already includes the *imopen* function, which performs this processing. The toolbox includes functions that perform many common morphological operations.

### 3.6.6 Morphological Opening

Morphological opening can be used to remove small objects from an image while preserving the shape and size of larger objects in the image. Create a structuring element using *strel* function. The structuring element should be large enough to remove the lines when erode the image, but not large enough to remove the rectangles. It should consist of all 1's, so it removes everything but large contiguous patches of foreground pixels. Erode the image with the structuring element. This removes all the lines, but also shrinks the rectangles and to restore the rectangles to their original sizes. Figure 3.21 is the

coding of the erosion and dilation function on the face image. The output image is shown in figure 3.22.



```
C:\Program Files\MATLAB71\work\THESIS.m
File Edit Text Cell Tools Debug Desktop Window Help
[Icons] Stack: Base
1 - I= imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - BW1=im2bw(I);
3 - SE=strel('rectangle',[5 15]);
4 - BW2=imerode(BW1,SE);
5 - figure(1), imshow(BW2);
6 - BW3=imerode(BW2,SE);
7 - figure(2), imshow(BW3);
script Ln 3 Col 24 OVR
```

Figure 3.21: Erosion and Dilation Coding

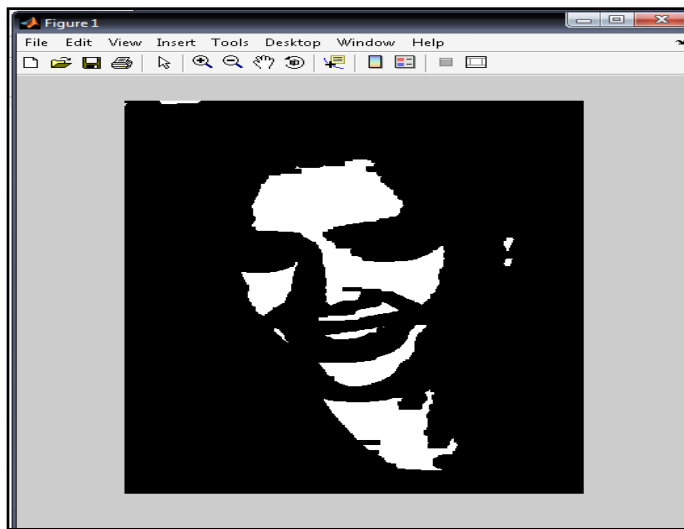


Figure 3.22: Image after erosion and dilation process.

## **3.7 Processing**

### **3.7.1 Feature Extraction**

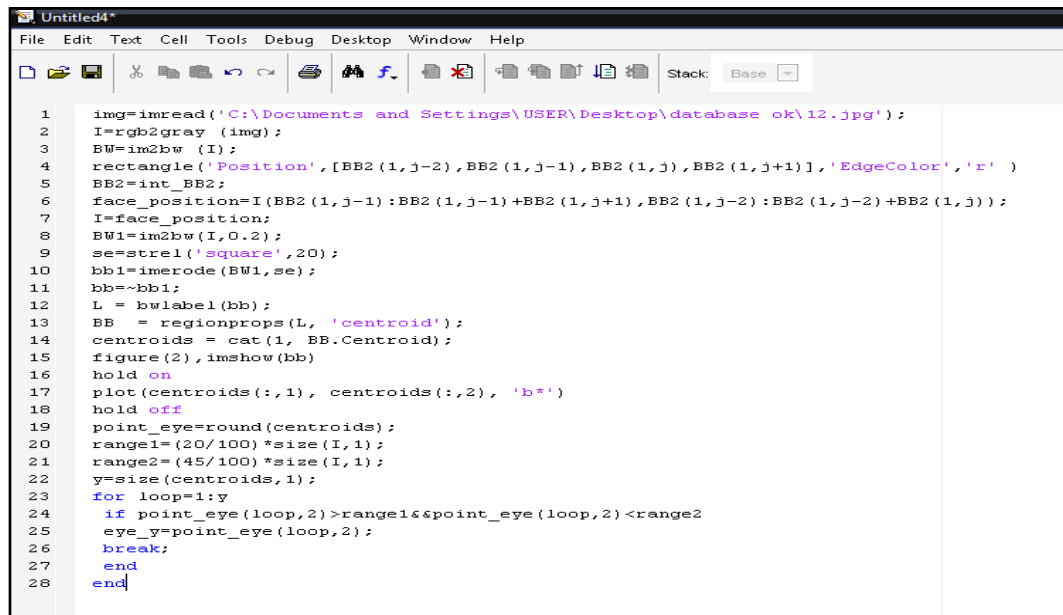
After the preprocessing process, the output of the preprocessing is implemented to the next process that is post processing. Processing part is included feature extraction, segmentation and a process to detect the face and eye. The next process is a feature extraction. Feature extraction is a process to extract the data that is the binary image to determine the point of the eye. This is important to determine which image points are relevant for further processing. The different between object classes is discriminated based on a set of features.

In this project, the features that are chosen are the region of the face and the both eye region. A large set of different features is investigated. Feature extraction process is important to extract the face region from the overall image that is displayed. The region is used for the further process. After the face region is detected, the process is proceed by detected the region of the eye using in the face region. The best subset out of a large feature set is selected and the region is classified as the face and the eye. In this feature extraction, a segmentation process is occurred.



### 3.7.2 Segmentation Process

Segmenting an image is the process by which a computer attempts to separate objects within an image from the background as well as from other objects. The segmentation rules are the rules that will determine the formation of regions. The segmentation rules are based on analyzing the color and edge properties of a region.



```

1  img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2  I=rgb2gray (img);
3  BW=im2bw (I);
4  rectangle ('Position', [BB2 (1, j-2), BB2 (1, j-1), BB2 (1, j), BB2 (1, j+1)], 'EdgeColor', 'r' )
5  BB2=int_BB2;
6  face_position=I (BB2 (1, j-1) : BB2 (1, j-1) + BB2 (1, j+1), BB2 (1, j-2) : BB2 (1, j-2) + BB2 (1, j));
7  I=face_position;
8  BW1=im2bw (I, 0.2);
9  se=strel ('square', 20);
10 bb1=imerode (BW1, se);
11 bb=~bb1;
12 L = bwlabel (bb);
13 BB = regionprops (L, 'centroid');
14 centroids = cat (1, BB.Centroid);
15 figure (2), imshow (bb)
16 hold on
17 plot (centroids (:, 1), centroids (:, 2), 'b*')
18 hold off
19 point_eye=round (centroids);
20 range1=(20/100)*size (I, 1);
21 range2=(45/100)*size (I, 1);
22 y=size (centroids, 1);
23 for loop=1:y
24     if point_eye (loop, 2) > range1 && point_eye (loop, 2) < range2
25         eye_y=point_eye (loop, 2);
26         break;
27     end
28 end

```

Figure 3.23: Coding for segmentation process.

Figure 3.23 is showing the coding for segmentation process. The coding is using the morphology function to remove the background of the image and only show the face region image. The structuring element function is applying to detect the position of the face from the overall image. The face region that has been detected is cropped using the *crop* function and only the cropped face region is display in the figure window. The output of the segmentation process is shown in figure 3.24 below.

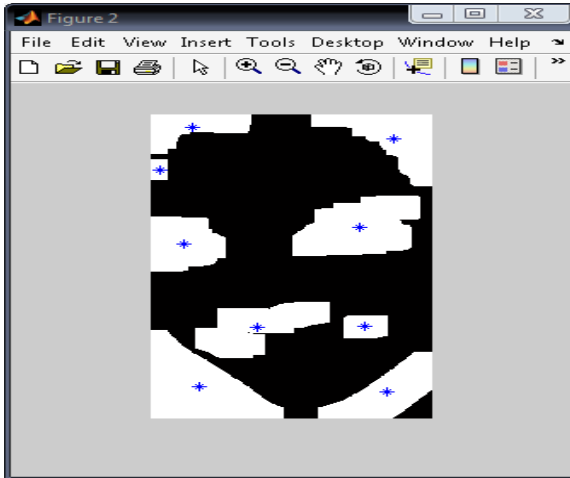


Figure 3.24: Segmentation Output Image

### 3.7.3 Canny Edge Detection Methods

A step up from pixel methods based solely on the intensity at a given point is edge detection algorithms. These use standard image-processing techniques such as the Canny, Sobel and Laplacian operators to extract lines from within the image. This project will applied Canny Edge Detection in segmentation process. The Canny edge detects or proceeds in a series of stages. The input is low-pass filtered with Gaussian kernel. This smoothing stage serves to suppress noise, and in fact most edge detection system utilizes some form of smoothing as a pre-processing step. The gradient image  $G_x$  and  $G_y$  will be obtained using the Sobel operator.

The magnitude of the gradient then will be calculated, perhaps using the absolute value approximation ( $|G| = |G_x| + |G_y|$ ). The direction of the individual gradient vectors, with respect to the x-axis, will be found using the formula  $\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$ . In the event that  $G_x = 0$ , a special case must be taken into account and  $\theta$  will then either set to 0 or 90, as determine by inspecting the value of  $G_y$  at that particular pixel location. These direction vectors are then typically will be quantized into a set of eight possible directions.

The edge directions will be used, the resultants edges are “thinned” via a process known as non-maximal suppression. Those gradient values that are not local maxima will be flagged as non-edges pixel and set to zero. Spurious edges will be removed through a process known as hysteresis thresholding. Two thresholds will be used to track edges in the non-maxima suppress gradient. In essence, edge contour will be created by traversing along an edge until the gradient dips below a threshold. Canny edge detection is a good detection method because there are low probability of not marking real edge points, and falsely marking non-edge points and also a good localization method.

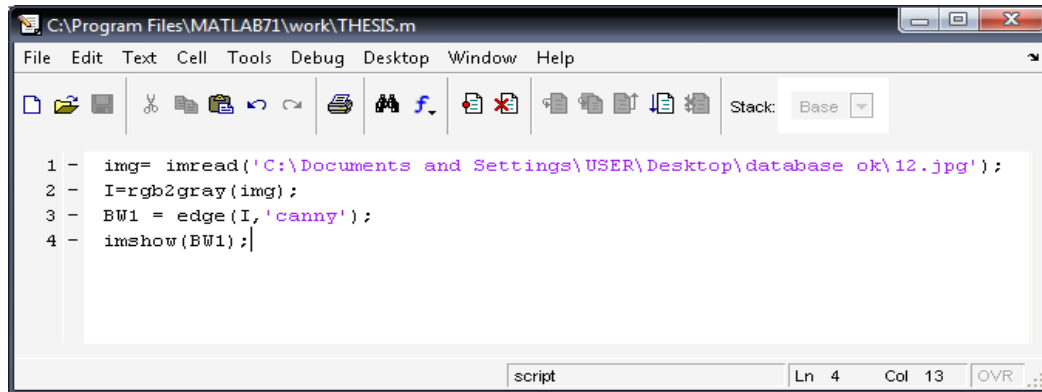
Canny Edge Detection:

$$SNR = \frac{|\int_{-w}^w G(-x)f(x)dx|}{n_0 \sqrt{\int_{-w}^w f^2(x) dx}} \quad (2.6)$$

•  $f$  is the filter,  $G$  is the edge signal, denominator is the root-mean squared response to noise  $n(x)$  only.

$$\text{Localization} = \frac{1}{\sqrt{E[x_0^2]}} = \frac{|\int_{-w}^w G'(-X)f'(x)dx|}{n_0 \sqrt{\int_{-w}^w f'^2(x)dx}} \quad (2.7)$$

- a measure that increases as localization improves.
- Use reciprocal of the rms distance of the marked edge from the center of the true edge.



```
C:\Program Files\MATLAB71\work\THESIS.m
File Edit Text Cell Tools Debug Desktop Window Help
Stack: Base
1 - img= imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - I=rgb2gray(img);
3 - BW1 = edge(I,'canny');
4 - imshow(BW1);
script Ln 4 Col 13 OVR
```

Figure 3.25: Coding of Canny Edge Detection

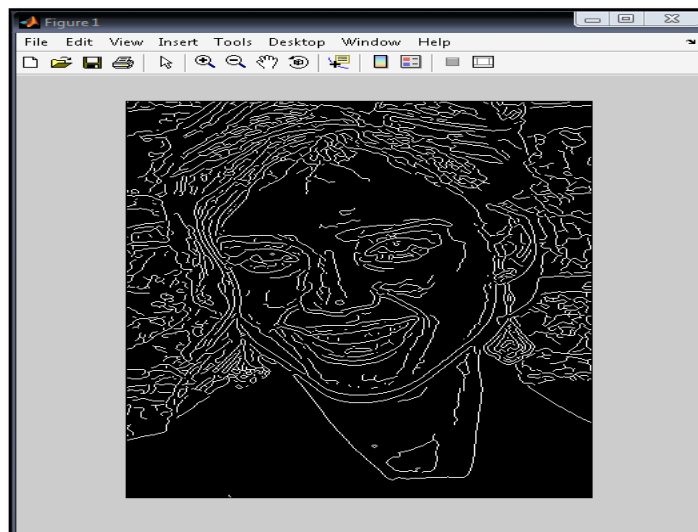


Figure 3.26: Output image of Canny Edge Detection.

### 3.7.4 Face Detection

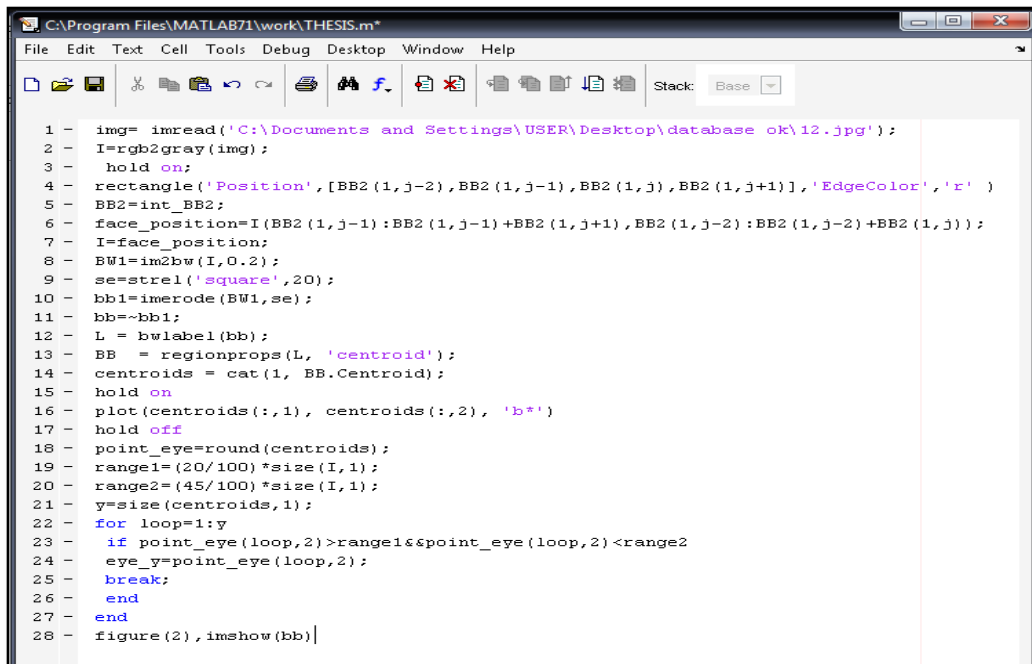
After the segmentation process, the process proceeds to the face and eye detection process that is to determine the accurate position of the face and eye from the image. In this project I applied the function of rectangle in order to display a region of the face region and the eye region.

The function of rectangle drew a rectangle with position [0, 0, 1, 1]. The command of rectangle function is *rectangle ('Position', [x, y, w, h])*. This command will draw the rectangle from the point x, y and having a width of w and a height of h. Specify values in axes data units. The values of x and y can range from 0 to 1. A value of [0, 0] creates a rectangle with square sides. Location and size of rectangle property is specified in the data units of the axes. The point defined by x, y specifies one corner of the rectangle, and width and height define the size in units along the x-and y-axes respectively. This face region detection region is important to limit the space for the eye detection. Beside that this region made the process for estimating the region of eye more easily and also avoid any false in determine the eye.

This project is applied the morphology technique that is a morphological structuring element to build the square for the face region. For all shapes except 'arbitrary', structuring elements are constructed using a family of techniques known collectively as structuring element decomposition. The principle is that dilation by some large structuring elements can be computed faster by dilation with a sequence of smaller structuring elements. For example, dilation by an 11-by-11 square structuring element can be accomplished by dilating first with a 1-by-11 structuring element and then with an 11-by-1 structuring element. [18] The command of the structuring element for square

shape is  $E = \text{strel}(\text{'square'}, W)$  where width is  $W$  pixels.  $W$  must be a nonnegative integer scalar.

All the coding for the face detection is shown in figure 3.26. For face detection process, it used the face region that has been displayed during the segmentation process before. In figure 3.23, there are many blue points displayed. The blue point is the centroids that is determined as the eye, mouth and nose point. The centroids is used for build a rectangle as shown in figure 3.25. The entire centroids are used as a guide to build a rectangle on the face image. In building the rectangle, the structuring element is applied and the rectangle is shown in figure 3.28. The rectangle also built by following the edge color of the face region.



```

1 - img= imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2 - I=rgb2gray(img);
3 - hold on;
4 - rectangle('Position', [BB2(1,j-2), BB2(1,j-1), BB2(1,j), BB2(1,j+1)], 'EdgeColor', 'r' )
5 - BB2=int_BB2;
6 - face_position=I(BB2(1,j-1):BB2(1,j-1)+BB2(1,j+1), BB2(1,j-2):BB2(1,j-2)+BB2(1,j));
7 - I=face_position;
8 - BW1=im2bw(I,0.2);
9 - se=strel('square',20);
10 - bb1=imerode(BW1,se);
11 - bb=~bb1;
12 - L = bwlabel(bb);
13 - BB = regionprops(L, 'centroid');
14 - centroids = cat(1, BB.Centroid);
15 - hold on
16 - plot(centroids(:,1), centroids(:,2), 'b*')
17 - hold off
18 - point_eye=round(centroids);
19 - range1=(20/100)*size(I,1);
20 - range2=(45/100)*size(I,1);
21 - y=size(centroids,1);
22 - for loop=1:y
23 -     if point_eye(loop,2)>range1&&point_eye(loop,2)<range2
24 -         eye_y=point_eye(loop,2);
25 -         break;
26 -     end
27 - end
28 - figure(2), imshow(bb)

```

Figure 3.27: Face Detection Coding

Figure 3.28 is showing the rectangle of the face region. This rectangle is applied to the grayscale image and the image is shown in figure 3.29. The red rectangle is determined as the face region.

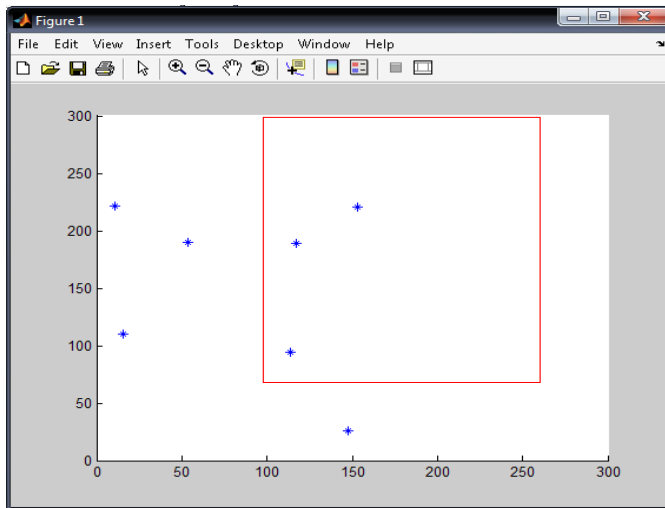


Figure 3.28: The rectangle of the Face Region

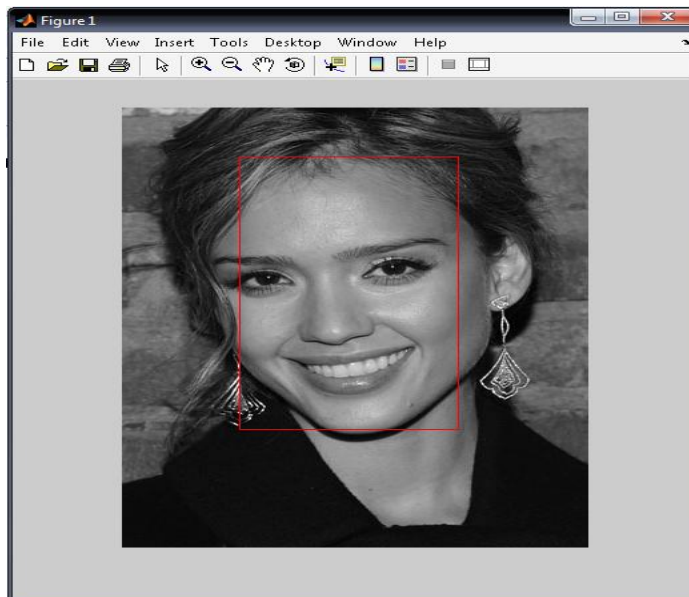
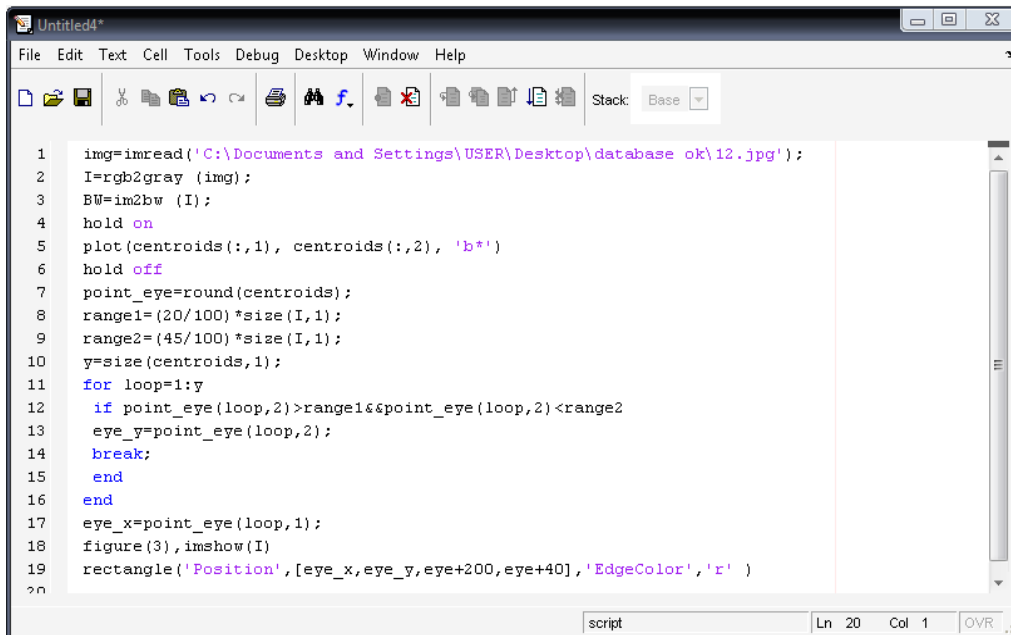


Figure 3.29: Face Detection Image.

### 3.7.5 Eye Detection

Eye detection is the aim of this project. In this project the position of the both eye is detected. The eye region is detected and displayed in a rectangle range. For eye detection process I also applied the same method for face detection. The structuring element that is a morphology technique is applied into this process so that the region of the eye can be detected. The process is preceded using the output image of the segmentation process. The centroids points in that image are used as a guide to detect the position of the eye. A bounding box is build around the two points that is identifying as the eye. We can just determine the position of the eye by looking at the two identical points. The eye is detected and displayed in the figure 3.31.



```

1  img=imread('C:\Documents and Settings\USER\Desktop\database ok\12.jpg');
2  I=rgb2gray (img);
3  BW=im2bw (I);
4  hold on
5  plot(centroids(:,1), centroids(:,2), 'b*')
6  hold off
7  point_eye=round(centroids);
8  range1=(20/100)*size(I,1);
9  range2=(45/100)*size(I,1);
10 y=size(centroids,1);
11 for loop=1:y
12     if point_eye(loop,2)>range1&&point_eye(loop,2)<range2
13         eye_y=point_eye(loop,2);
14         break;
15     end
16 end
17 eye_x=point_eye(loop,1);
18 figure(3),imshow(I)
19 rectangle('Position',[eye_x,eye_y,eye+200,eye+40],'EdgeColor','r' )
20

```

Figure 3.30: Eye Detection Coding



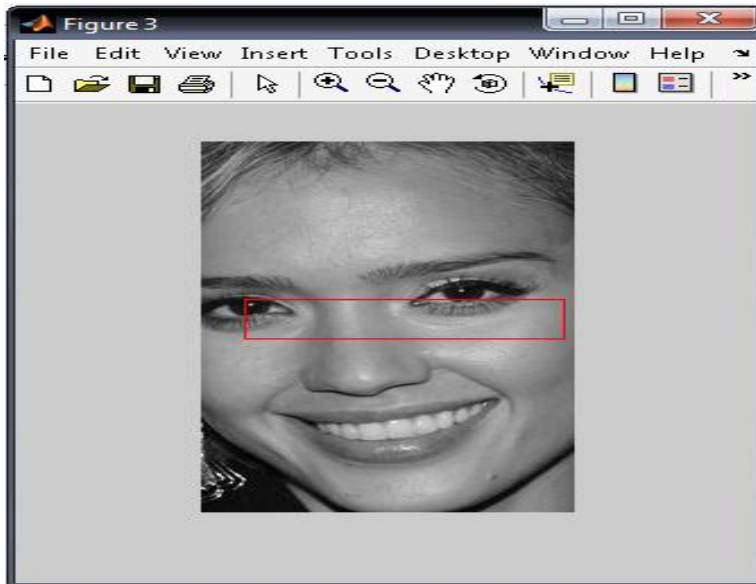


Figure 3.31: Eye Detected image.

### 3.8 GUIDE MATLAB

All the process above is written in MATLAB and referring to Image Processing Toolbox User's Guide. Finally, after the face and the eye position are detected, the Graphical User Interface (GUI) is designed. Graphical User Interface is a type of user interface which allows people to interact with electronic device. GUI also will be developed in MATLAB. GUI is developed to display the output image of the overall process. There are two important tasks in build the GUI. Those are layout design and GUI programming. In MATLAB, the GUI is build using GUIDE.

GUIDE is the MATLAB Graphical User Interface development environment, which provides a set of tools for creating graphical user interfaces (GUIs). These tools greatly simplify the process of designing and building GUIs. The GUIDE tools are used to lay out the GUI. A GUI is layout using the GUIDE Layout Editor. The GUI design is easily by clicking and dragging GUI components such as panels, buttons, text fields and sliders into the layout area. After the lay out of the GUI is designed then the GUI is programmed.

GUIDE automatically generates an M-file that controls how the GUI operates. The M-file initializes the GUI and contains a framework for all the GUI callbacks the commands that are executed when a user clicks a GUI component. By using the M-file editor, the code is added to the callbacks to perform the functions of the GUI.

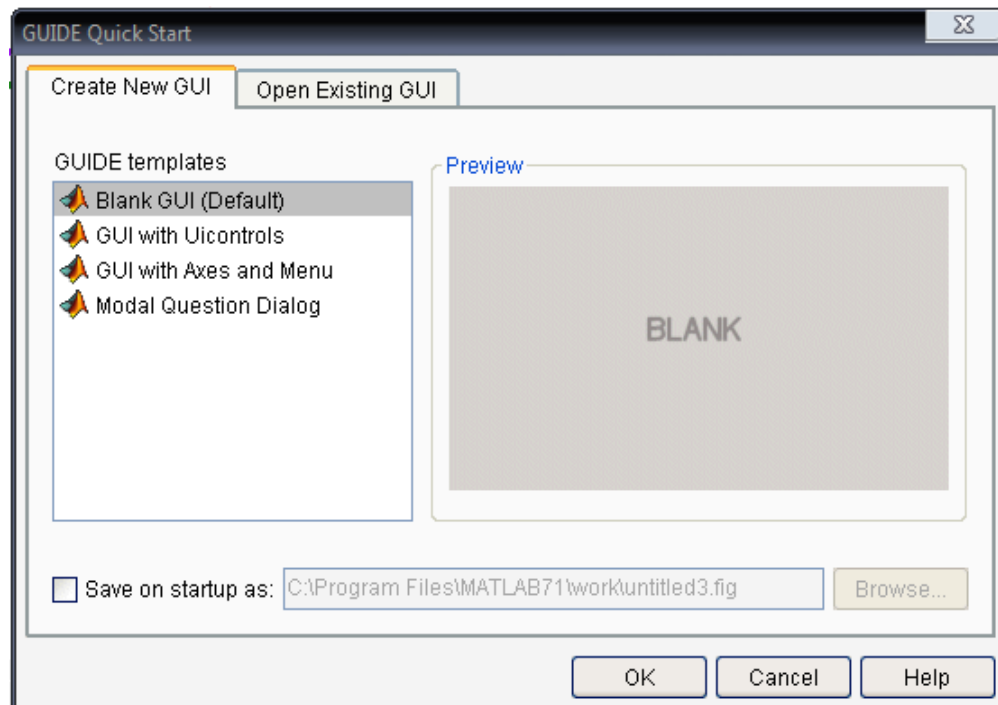


Figure 3.32: Quick GUIDE Start Window

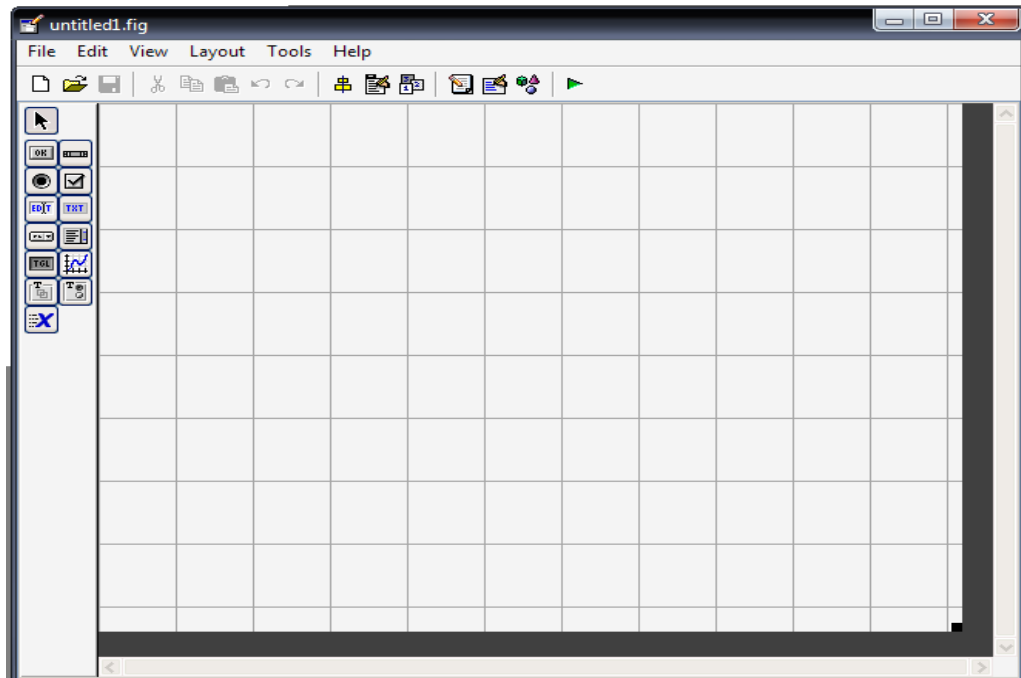


Figure 3.33: Layout Editor for GUI Design

### 3.8.1 The Layout Editor

In GUIDE, the Layout Editor is displayed. Layout Editor is the control panel for all of the GUIDE tools. The following figure shows the Layout Editor with a blank GUI template. The GUI is lay out by dragging components, such as push buttons, pop-up menus, or axes, from the component palette, at the left side of the Layout Editor, into the layout area. All the dragged component is appeared as in the following figure 3.34.

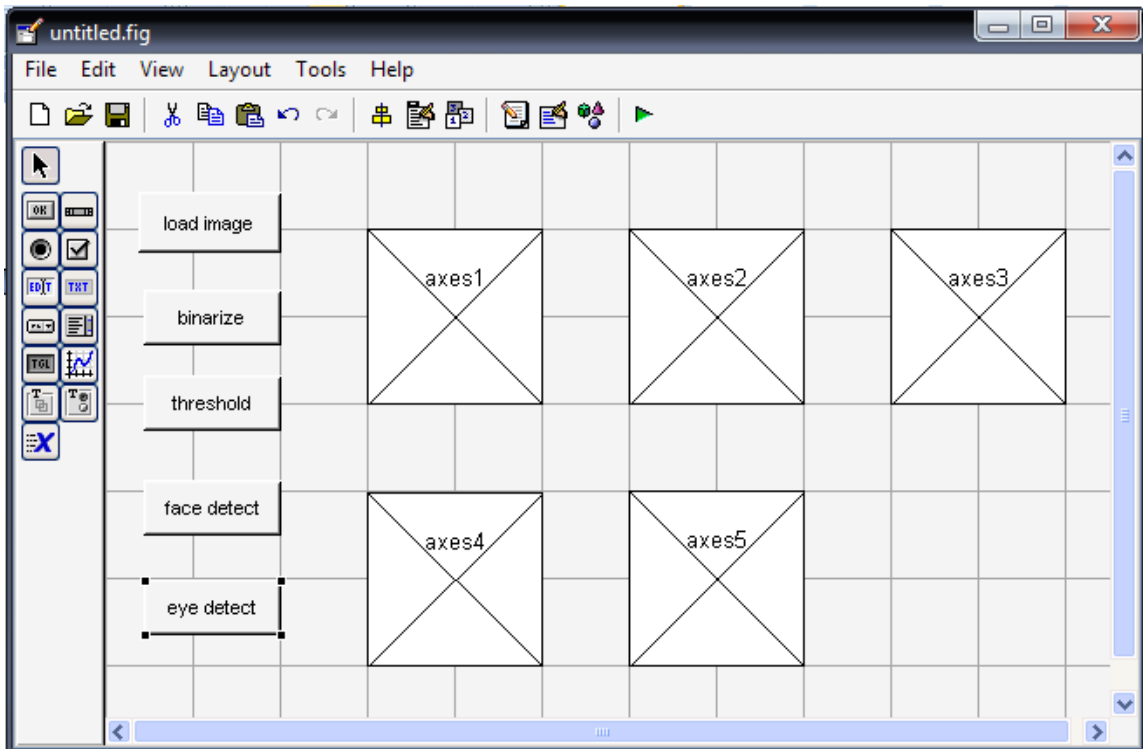


Figure 3.34: GUI Design

### 3.8.2 Setting Properties for GUI Components

After all the component appear in the figure, the next procee is to set the properties of each GUI component by selecting the Property Inspector from the View menu to display the Property Inspector dialog box. Select a component in the Layout Editor and the Property Inspector displayed that component's properties. After all the

component properties is set, run the layout editor. The layout is displayed like the figure 3.6.3 .

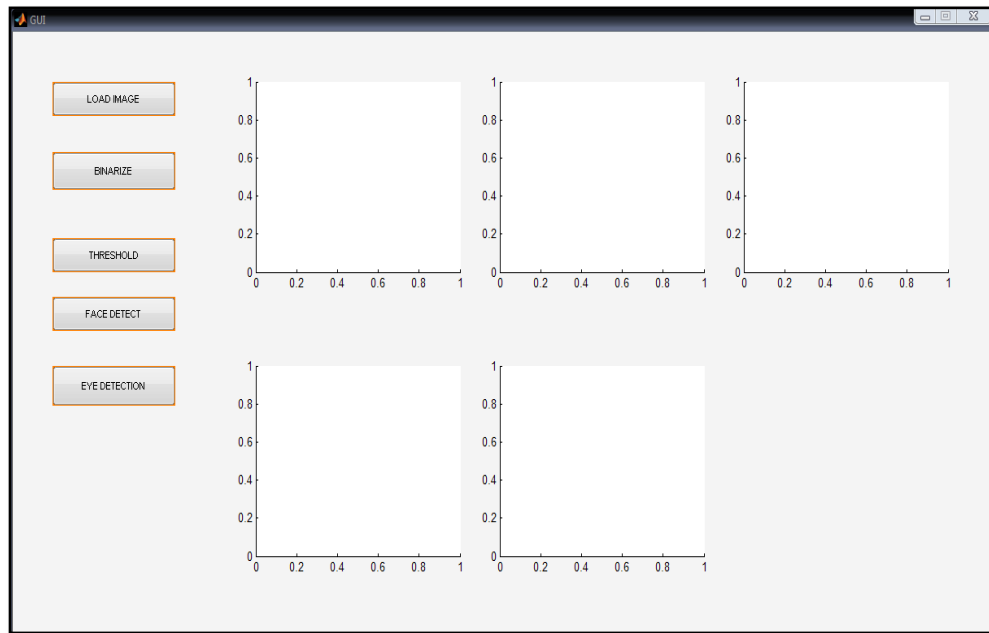


Figure 3.35: GUI Layout after Debug

### 3.7.3 GUI FIG-Files and M-Files

GUIDE stores a GUI in two files, which are generated the first time I save or run the GUI. A FIG-file with extension `.fig` contains a complete description of the GUI layout and the components of the GUI. An M-file, with extension `.m` contains the code that controls the GUI, including the callbacks for its components. These two files

correspond to the tasks of laying out and programming the GUI. All the work for the layout design is stored in the FIG-file and the GUI programming is stored in the M-file. The overall process for this project is followed the process in figure 3.36.

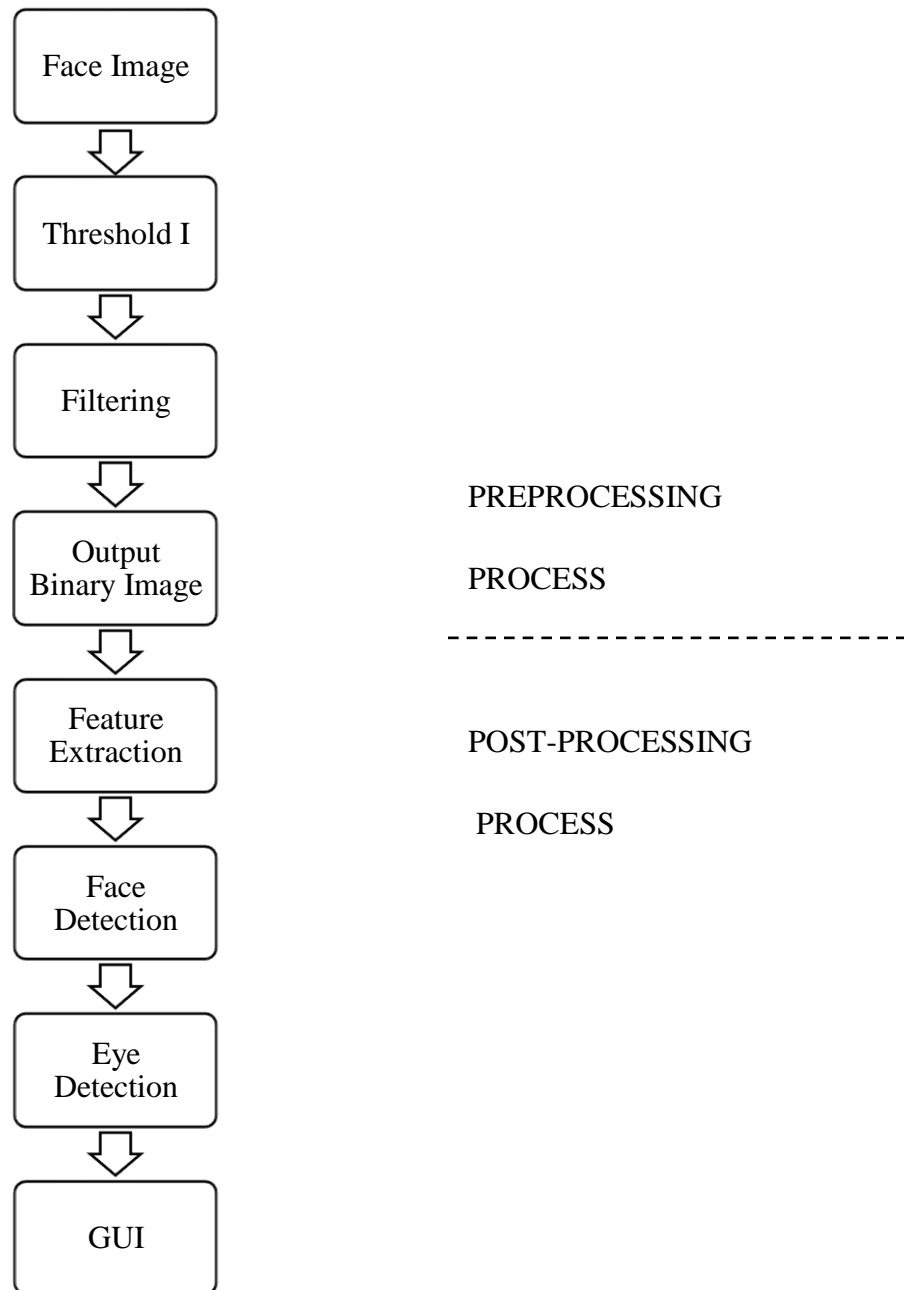


Figure 3.36: Overall Process of Eye Detection

## CHAPTER 4

### RESULT & DISCUSSIONS

#### 4.1 Load Image from Database



Figure 4.1: Original Face Image

Figure 4.1 is the example of the face image that has been use in this eye detection process. This face image is loaded using the MATLAB function that is *imread* function. This function reads a grayscale or color image from the file specified by the string filename, where the string specifies the format of the file.

This function or command returned the image data in the array (A). If the file contains a grayscale image, A is a two-dimensional (M-by-N) array. If the file contains a color image, A is a three-dimensional (M-by-N-by-3) array. The class of the returned array depends on the data type used by the file format. For most file formats, the color image data returned uses the RGB color space. [18]

## 4.2 Preprocessing Result

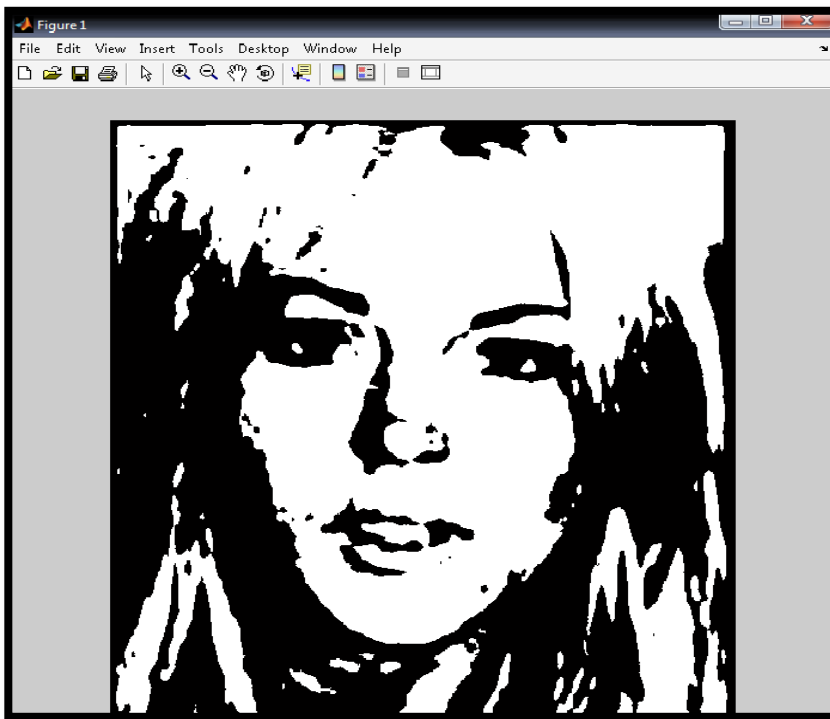


Figure 4.2: Face image for Threshold Process (Black & White).



Figure 4.2 above is the figure when the original image is threshold by the threshold process. Threshold process is the process to convert the original image that is in RGB format into black and white format. This also called binarization process; where 0 representing black and 1 representing white. In this project I applied the function of *rgb2gray* first to convert the RGB image to the gray scale color image and then I applied the function *im2bw* to convert the gray scale image to black and white image. All the function is included in the MATLAB Functions.

### 4.3 Face region detection

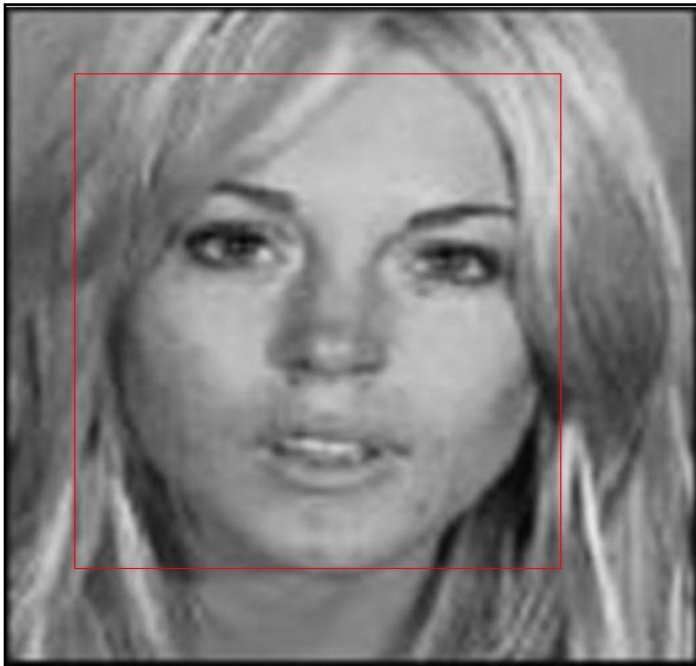


Figure 4.3: Face Region Detected

Figure 4.3 is the figure of the region of face that is detected using dilation and erosion technique. Before proceed to the next step, I has detect the face region. This step

is importance to make the limitation and reduce noise of image for the next process. In MATLAB I applied the rectangle function to build the bounding box on the face image.

#### 4.4 Threshold and Filtering Result

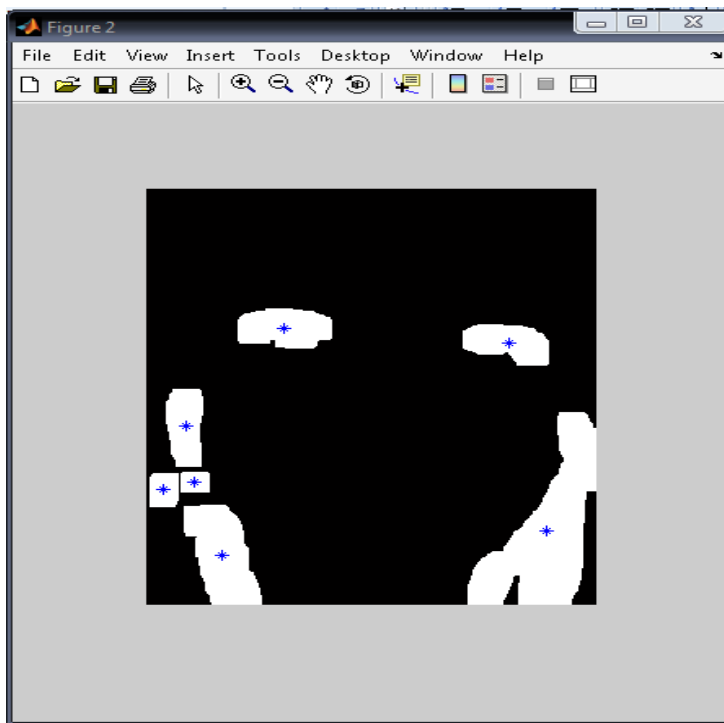


Figure 4.4: Face Image after Filtering process.

Figure 4.4 is the image after the filtering process. In this process, the several noises have been removed. In the image we can saw the region of eye that is in white color. There are also other white points that can be known as noise. From the image we can clearly identify the position of the eye. As a human our right and left eye is always parallel to each other so the two points that are parallel to each other is identify as the aye region.

## 4.5 Eye region detection

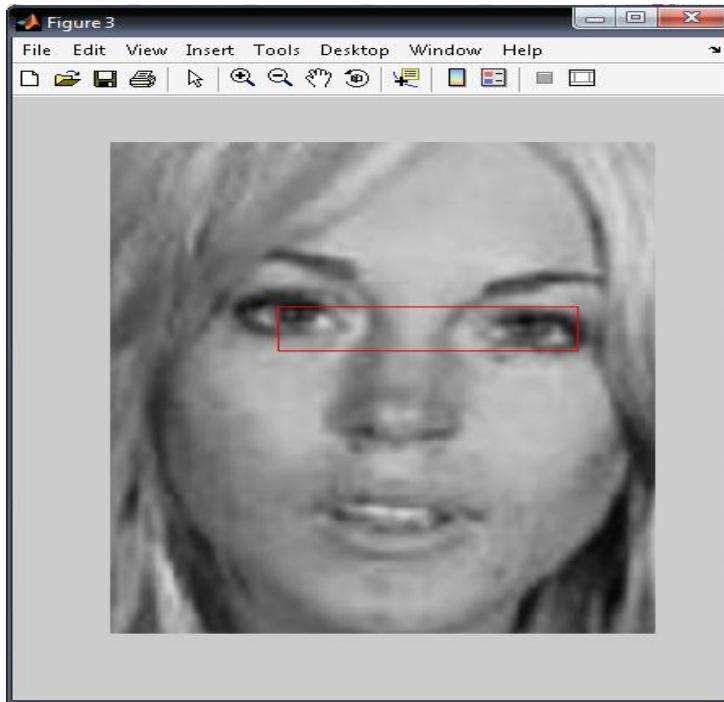


Figure 4.5: Eye Detected Image

Figure 4.5 is the image of eye region that has been detected from the face image. In eye detection process I applied the morphology technique that are dilation and erosion technique. Dilation technique is used to build the square of the eye region. The structuring element is implemented in this step. In order to build the square, I applied the square structuring element.

#### 4.6 Image of the Eye Detection Process

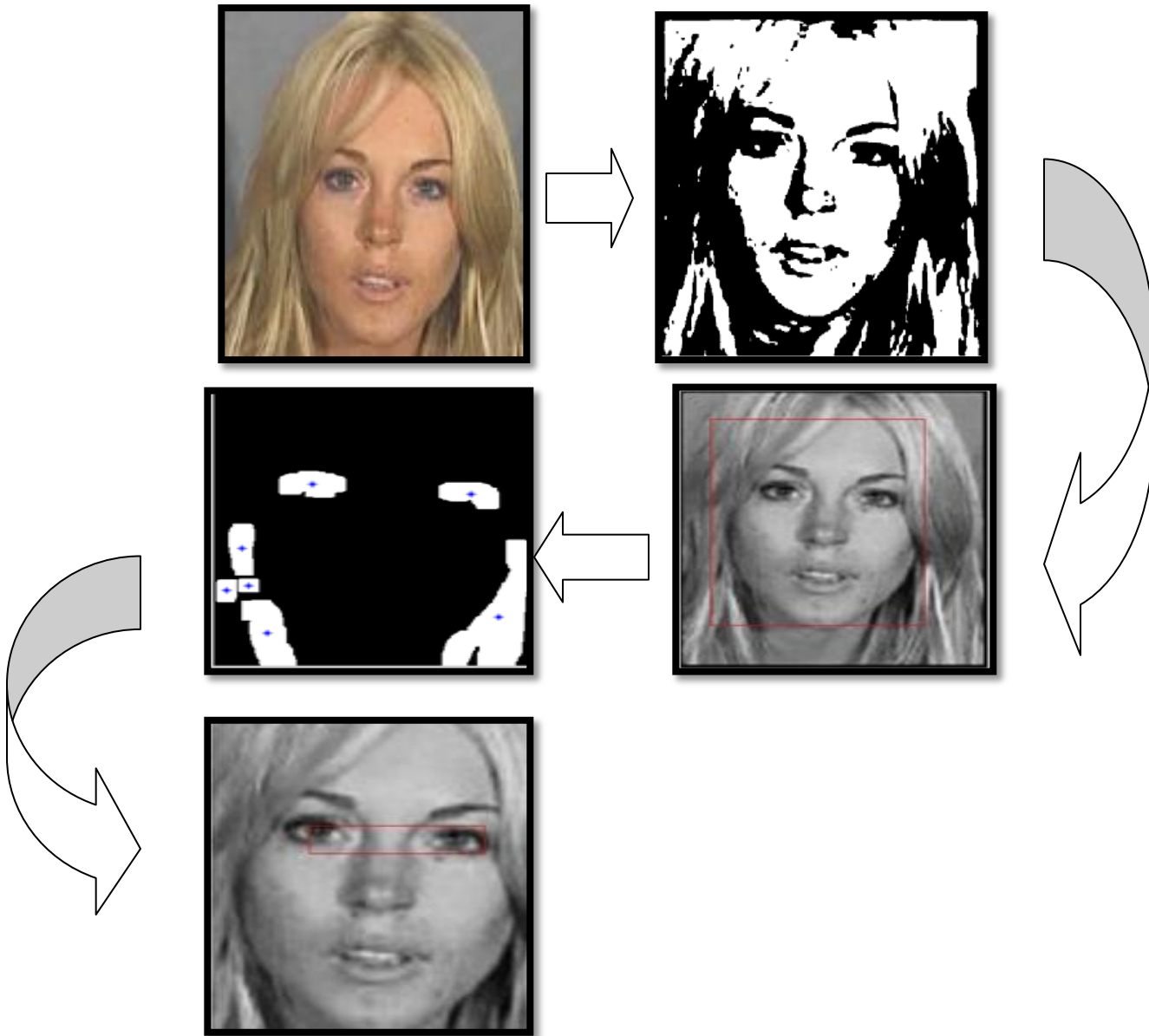


Figure 4.6: Processed Image

## 4.7 GUI

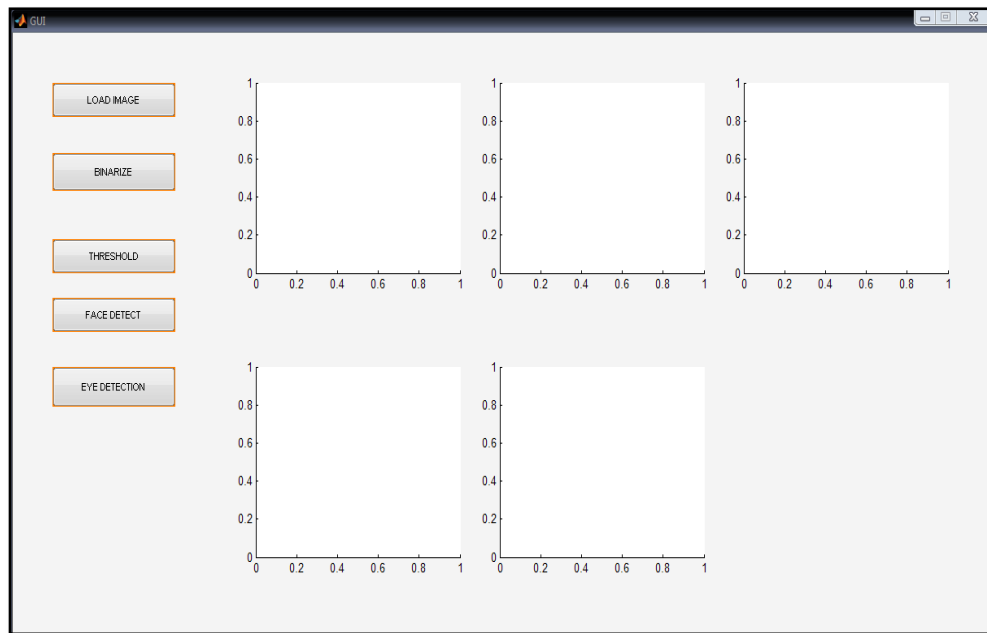


Figure 4.7: GUI

Figure 4.7 is a displayed of GUI for eye detection. Graphical User Interface (GUI) builds using GUIDE MATLAB. The entire buttons in the GUI have their own function. Load Image button is to load a face image from the file that the location of the file has been stated in the M-file. The face image will be display in the GUI. There is only use one face image to be display at one time.

Detect Face button is to display the image of the face detected image. All the programming of the face detection is written in the GUI programming part. While the Detect Eye button is to display the eye detected image. All the programming for eye detection is written in the GUI programming part. The two buttons, OK and Cancel button is function to exit from the GUI. The entire button is functioning by following the programming that is stored in the M-file.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATIONS**

#### **5.1 Conclusion**

The aim of this project is to detect the eye from the face image and to produce a quality human face image using the MATLAB image processing library. The enhanced image is produced from the image processing process. This project introduced the eye detection using MATLAB. Image processing process is the main process of the eye detection. The preprocessing process is applied to input an image that is human face image. Preprocessing is including the threshold and filtering process.

Threshold process is to convert the input image into binary image in the system. While filtering process is applied to enhance the contrast of dark regions and to improve the quality and also remove the unwanted noise that occurs on the image. Therefore, facial images with poor contrast are enhanced, which makes the clustering process more effective. Eye pair candidates are extracted by using a binary template matching technique in feature extraction process and also used the morphology technique to build the rectangle and square of the face and eye region. The eye will be detection by referring the MATLAB image processing library and is written in m- file (editor). The output of this project that is eye image is displayed in Graphical User Interface (GUI).

## **5.2 Future Recommendation**

As a recommendation, this project should be proceeding to an iris recognition system. Basically, this project is the start point for the advance face recognition system. This project is applicable to the security system. This project exactly is the beginning of the advance project that is an iris recognition system. Iris recognition system is a system that uses to identify the face image and the identity of the image by detecting the iris and matching the image with the database. This project is an offline system that only uses the image from the offline database. This can be more functioning if we applied it for online system and at real time system.

This project can be a new security system that is more effective than using the password system. The new security system that can use this project is a system to access

the computer or notebook, this because the old system that use a password is not fully safe. Beside that, this project can be applied in drowsiness detection system. This can avoid the accident that cause of the drowsiness. This can reduced the numbers of accident in this country.



## REFERENCE

- [1] Anil K. Jain . “Fundamental of Digital Image Processing”.
- [2] Abdallah S. Abdallah “Investigation of New Techniques for Face Detection”.
- [3] Bart Kroon, Alan Hanjalic, Sander M.P. Maas, Eye Localization for Face Matching
- [4] Brooke Williams,Zoran Nikolic,Gaganjot Maur, “Transforming Performance to Safety in Automotive Applications”
- [5] C. C. Chiang et al, “A novel method for detecting lips, eyes and faces in real time”, *Real-Time Imaging*, 9, 2003.
- [6] G. C. Feng et al, “Multi-cues eye detection on gray intensity image”, *Pattern Recognition* 34(5), 1033-1046..
- [7] G.C. Feng, P.C. Yuen (1998). “Variance Projection and Its Application to Eyes detection for Human Face Recognition”. *Pattern Recognition Letters* 809-906

- [8] Gerard Blanchet and Maurice Charbit(2006) “Digital Signal and Image Processing Series”.
- [9] [http://en.wikipedia.org/wiki/Computer\\_vision](http://en.wikipedia.org/wiki/Computer_vision),Computer Vision
- [10] Haro, Flickner, Essa (1999),Detecting and Tracking Eyes By Using Their Physiological
- [11] [http://en.wikipedia.org/wiki/Face\\_detection](http://en.wikipedia.org/wiki/Face_detection)
- [12] [http://en.wikipedia.org/wiki/Image\\_processing](http://en.wikipedia.org/wiki/Image_processing)
- [13] [http://en.wikipedia.org/wiki/Segmentation\\_\(image\\_processing\)](http://en.wikipedia.org/wiki/Segmentation_(image_processing))
- [14] <http://www.bjpu.edu.cn/sci/multimedia/mul-lab/3dface/overview.htm> database
- [15 ] <http://www.wordiq.com/definition/MATLAB> method
- [16] Kapil Ahuja and Yifei Ma (Spring 2007). “Final Project Report-Computer Vision (Art Gate)”
- [17] Li Song Jin. “A method of face detection using geometrical structure of human face”, Kuahakwontongbo(Bulletin of natural science of Academy of Science of D. P. R of Korea) 51(1) 21-26.
- [18] MATLAB (Version 7.1) image processing references.
- [19] M. H. Yang et al, “Detecting Faces in Images : A Survey”, IEEE trans. on PAMI. 24(1), 34-59.
- [20] Maurice Yew Fong Leong, “Implementation of a real- time automated Base Face Recognition System”.
- [21] Qiong Wang Jingyu Yang, “Eye Detection in Facial Images with Unconstrained Background”.
- [22] Richard C. Dorf. (2005) “Circuit, Signal, and Speech and Image Processing”.

- [23] Shehrzad Qureshi(2005) “Embedded Image Processing on DSP”.
- [24] [www.elsevier.com/locate/cviu](http://www.elsevier.com/locate/cviu), Special issue: eye detection and tracking.
- [25] Zafer Savas (2005). “Real-Time Detection and Tracking of Human Eyes in Video Sequences”

**Appendix A**

**EYE DETECTION PROGRAMMING**

## Eye detection Programming

```

img=imread('C:\Documents and Settings\USER\Desktop\database
ok\07.jpg');
I=rgb2gray (img);
BW=im2bw (I);
[n1 n2]=size (BW);
r=floor (n1/10);
c=floor(n2/10);
x1=1;x2=r;
s=r*c;

for i=1:10
y1=1;y2=c;
for j=1:10
if (y2<=c | y2>=9*c) | (x1==1 | x2==r*10)
loc=find(BW(x1:x2, y1:y2)==0);
[o p]=size (loc);
pr=o*100/s;
if pr<=100
BW(x1:x2, y1:y2)=0;
r1=x1;r2=x2;s1=y1;s2=y2;
pr1=0;
end
imshow(BW);
end
y1=y1+c;
y2=y2+c;
end

x1=x1+r;
x2=x2+r;
end
L = bwlabel(BW,8);
BB = regionprops(L, 'BoundingBox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);
int_BB2=round(BB2);

```

```

[s1 s2]=size(BB2);
mx=0;
for k=3:4:s2-1
p=BB2(1,k)*BB2(1,k+1);
if p>mx & (BB2(1,k)/BB2(1,k+1))<1.8
mx=p;
j=k;
end
end
figure(1),imshow(I);
hold on;
rectangle('Position',[BB2(1,j-2),BB2(1,j-
1),BB2(1,j),BB2(1,j+1)],'EdgeColor','r' )
BB2=int_BB2;
face_position=I(BB2(1,j-1):BB2(1,j-1)+BB2(1,j+1),BB2(1,j-2):BB2(1,j-
2)+BB2(1,j));
I=face_position;
BW1=im2bw(I,0.2);
se=strel('square',20);
bb1=imerode(BW1,se);
bb=~bb1;
L = bwlabel(bb);
BB = regionprops(L, 'centroid');
centroids = cat(1, BB.Centroid);
figure(2),imshow(bb)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
point_eye=round(centroids);
rangel=(20/100)*size(I,1);
range2=(45/100)*size(I,1);
y=size(centroids,1);
for loop=1:y
if point_eye(loop,2)>rangel&&point_eye(loop,2)<range2
eye_y=point_eye(loop,2);
break;
end
end

```

```
eye_x=point_eye(loop,1);  
figure(3),imshow(I)  
rectangle('Position',[eye_x,eye_y,eye+200,eye+40],'EdgeColor','r' )
```

**Appendix B**  
**GUI PROGRAMMING**



## Eye detection GUI Layout programming

```

function varargout = SEMINAR2(varargin)
% SEMINAR2 M-file for SEMINAR2.fig
%   SEMINAR2, by itself, creates a new SEMINAR2 or raises the
existing
%   singleton*.
%
%   H = SEMINAR2 returns the handle to a new SEMINAR2 or the handle
to
%   the existing singleton*.
%
%   SEMINAR2('CALLBACK',hObject,eventData,handles,...) calls the
local
%   function named CALLBACK in SEMINAR2.M with the given input
arguments.
%
%   SEMINAR2('Property','Value',...) creates a new SEMINAR2 or
raises the
%   existing singleton*. Starting from the left, property value
pairs are
%   applied to the GUI before SEMINAR2_OpeningFunction gets called.
An
%   unrecognized property name or invalid value makes property
application
%   stop. All inputs are passed to SEMINAR2_OpeningFcn via
varargin.
%
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only
one
%   instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SEMINAR2

% Last Modified by GUIDE v2.5 18-Nov-2009 19:20:45

% Begin initializationcode - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @SEMINAR2_OpeningFcn, ...
'gui_OutputFcn',  @SEMINAR2_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
if nargin && ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

```

```

if nargin
[varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before SEMINAR2 is made visible.
function SEMINAR2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to SEMINAR2 (see VARARGIN)

% Choose default command line output for SEMINAR2
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes SEMINAR2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = SEMINAR2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in LOADIMAGE.
function LOADIMAGE_Callback(hObject, eventdata, handles)
% hObject    handle to LOADIMAGE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB

```

```

% handles      structure with handles and user data (see GUIDATA)
[filename, pathname] = uigetfile({'*.jpg'; '*.bmp'; '*.gif'; '*.*'}, 'Pick
an Image File');
img = imread([pathname, filename]);
axes(handles.axes1);
imshow(img);

handles.S = img;
guidata(hObject, handles);

% --- Executes on button press in GRAYSCALES.
function GRAYSCALES_Callback(hObject, eventdata, handles)
% hObject      handle to GRAYSCALES (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
img = handles.S;
axes(handles.axes2);
I=rgb2gray(img);
imshow(I);

% --- Executes on button press in BINARIZE.
function BINARIZE_Callback(hObject, eventdata, handles)
% hObject      handle to BINARIZE (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)
imgbw=handles.S;
axes(handles.axes3);
BW=im2bw(imgbw);
imshow(BW);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject      handle to pushbutton4 (see GCBO)
% eventdata    reserved - to be defined in a future version of MATLAB
% handles      structure with handles and user data (see GUIDATA)

% --- Executes on button press in THRESHOLD.

```

```

function THRESHOLD_Callback(hObject, eventdata, handles)
% hObject    handle to THRESHOLD (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

I=handles.S;
axes(handles.axes4);
BW1=im2bw(I,0.2);
se=strel('square',10);
bb1=imerode(BW1,se);
bb=~bb1;
L = bwlabel(bb);
BB = regionprops(L, 'centroid');
centroids = cat(1, BB.Centroid);
imshow(bb)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
point_eye=round(centroids);
range1=(20/100)*size(I,1);
range2=(45/100)*size(I,1);
y=size(centroids,1);
for loop=1:y
if point_eye(loop,2)>range1&&point_eye(loop,2)<range2
eye_y=point_eye(loop,2);
break;
end
end

% --- Executes on button press in FACEDETECT.
function FACEDETECT_Callback(hObject, eventdata, handles)
% hObject    handle to FACEDETECT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

img = handles.S;
axes(handles.axes5);
I=rgb2gray (img);
BW=im2bw (I);

```

```

[n1 n2]=size (BW);
r=floor (n1/10);
c=floor(n2/10);
x1=1;x2=r;
s=r*c;

for i=1:10
y1=1;y2=c;
for j=1:10
if (y2<=c | y2>=9*c) | (x1==1 | x2==r*10)
loc=find(BW(x1:x2, y1:y2)==0);
[o p]=size (loc);
pr=o*100/s;
if pr<=100
BW(x1:x2, y1:y2)=0;
r1=x1;r2=x2;s1=y1;s2=y2;
pr1=0;
end
imshow(I);
end
y1=y1+c;
y2=y2+c;
end

x1=x1+r;
x2=x2+r;
end
L = bwlabel(BW,8);
BB = regionprops(L, 'BoundingBox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);
int_BB2=round(BB2);
[s1 s2]=size(BB2);
mx=0;
for k=3:4:s2-1
p=BB2(1,k)*BB2(1,k+1);
if p>mx & (BB2(1,k)/BB2(1,k+1))<1.8
mx=p;

```

```

j=k;
end
end
% figure(1),imshow(bb);
hold on;
rectangle('Position',[BB2(1,j-2),BB2(1,j-
1),BB2(1,j),BB2(1,j+1)],'EdgeColor','r' )
BB2=int_BB2;
face_position=I(BB2(1,j-1):BB2(1,j-1)+BB2(1,j+1),BB2(1,j-2):BB2(1,j-
2)+BB2(1,j));
I=face_position;
BW1=im2bw(I,0.2);
se=strel('square',20);
bb1=imerode(BW1,se);
bb=~bb1;
L = bwlabel(bb);
BB = regionprops(L, 'centroid');
centroids = cat(1, BB.Centroid);
% figure(2),imshow(bb)

% --- Executes on button press in EYEDETECT.
function EYEDETECT_Callback(hObject, eventdata, handles)
% hObject    handle to EYEDETECT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
img = handles.S;
axes(handles.axes6);
I=rgb2gray (img);
BW=im2bw (I);
[n1 n2]=size (BW);
r=floor (n1/10);
c=floor(n2/10);
x1=1;x2=r;
s=r*c;

for i=1:10
y1=1;y2=c;
for j=1:10

```

```

if (y2<=c | y2>=9*c) | (x1==1 | x2==r*10)
loc=find(BW(x1:x2, y1:y2)==0);
[o p]=size (loc);
pr=o*100/s;
if pr<=100
BW(x1:x2, y1:y2)=0;
r1=x1;r2=x2;s1=y1;s2=y2;
pr1=0;
end
imshow(I);
end
y1=y1+c;
y2=y2+c;
end

x1=x1+r;
x2=x2+r;
end
L = bwlabel(BW,8);
BB = regionprops(L, 'BoundingBox');
BB1=struct2cell(BB);
BB2=cell2mat(BB1);
int_BB2=round(BB2);
[s1 s2]=size(BB2);
mx=0;
for k=3:4:s2-1
p=BB2(1,k)*BB2(1,k+1);
if p>mx & (BB2(1,k)/BB2(1,k+1))<1.8
mx=p;
j=k;
end
end
% figure(1),imshow(bb);
hold on;
rectangle('Position',[BB2(1,j-2),BB2(1,j-1),BB2(1,j),BB2(1,j+1)], 'EdgeColor','r' )
BB2=int_BB2;

```

```

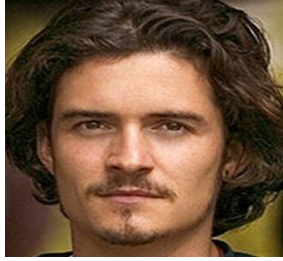

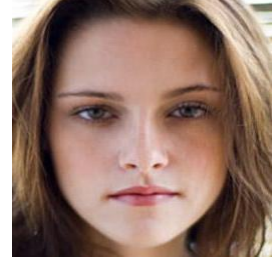



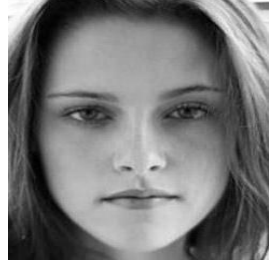
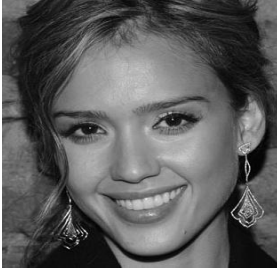




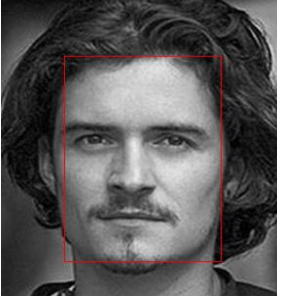
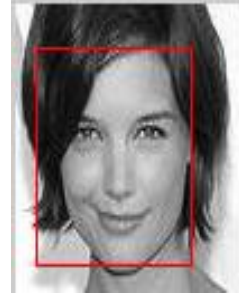
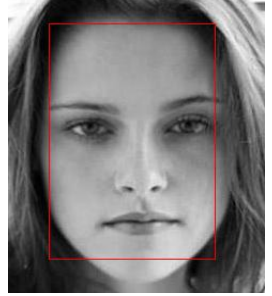
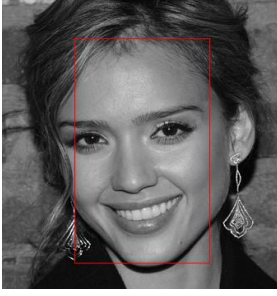
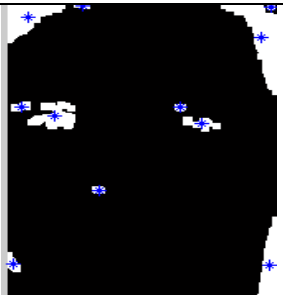
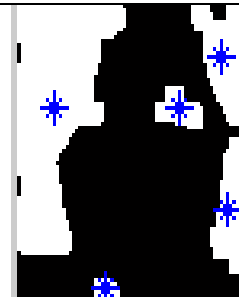





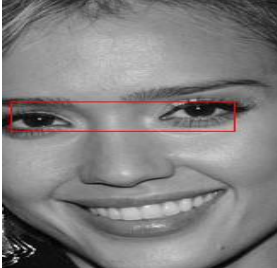
face_position=I(BB2(1,j-1):BB2(1,j-1)+BB2(1,j+1),BB2(1,j-2):BB2(1,j-
2)+BB2(1,j));
I=face_position;
BW1=im2bw(I,0.1);
se=strel('square',20);
bb1=imerode(BW1,se);
bb=~bb1;
L = bwlabel(bb);
BB = regionprops(L, 'centroid');
centroids = cat(1, BB.Centroid);
% imshow(bb)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
point_eye=round(centroids);
range1=(20/100)*size(I,1);
range2=(45/100)*size(I,1);
y=size(centroids,1);
for loop=1:y
if point_eye(loop,2)>range1&&point_eye(loop,2)<range2
eye_y=point_eye(loop,2);
break;
end
end
eye_x=point_eye(loop,1);
imshow(I)
rectangle('Position',[eye_x,eye_y,eye_x+130,eye_y+30],'EdgeColor','r' )













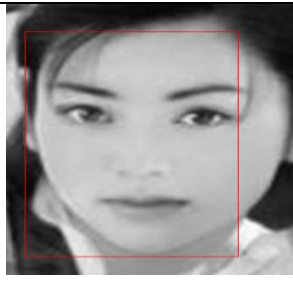
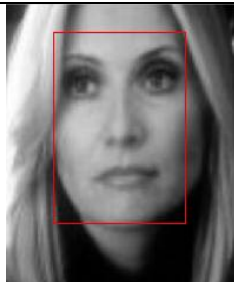



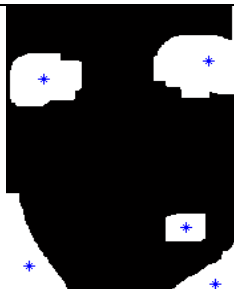
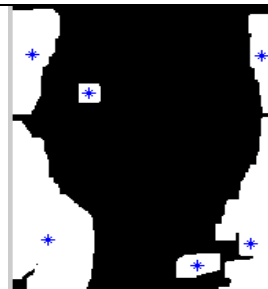
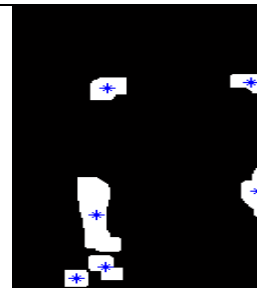
```



**Appendix C**

**EYE DETECTION TEST RESULT**

Original Image				
Gray Scale Image				
Binary Image				
Face Detection				
Region Image				
EYE Detection				

Original Image				
Gray Scale Image				
Binary Image				
Face Detect				
Region Image				
EYE Detect	