## UNIVERSITI MALAYSIA PAHANG

# BORANG PENGESAHAN STATUS TESIS♦

**JUDUL**: **REAL TIME FACE DETECTION SYSTEM**

**SESI PENGAJIAN**: **2007/2008**

Saya **AMY SAFRINA BINTI MOHD ALI ( 860314-35-5606 )**

mengaku membenarkan tesis (Sarjana Muda/~~Sarjana~~ /~~Doktor Falsafah~~)* ini disimpan di Perpustakaan dengan syarat-syarat kegunaan seperti berikut:

1. Tesis adalah hakmilik Universiti Malaysia Pahang (UMP).
2. Perpustakaan dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institusi pengajian tinggi.
4. **Sila tandakan ( √ )

| | | |
|---|---|---|
| ☐ | **SULIT** | (Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972) |
| ☐ | **TERHAD** | (Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan) |
| √ | **TIDAK TERHAD** | |

Disahkan oleh:

_____ (TANDATANGAN PENULIS)

_____ (TANDATANGAN PENYELIA)

Alamat Tetap:

**NO 22 SOLOK KG JAWA 8,**
**11900 BAYAN LEPAS,**
**PULAU PINANG.**

**MOHD ZAMRI BIN IBRAHIM**

Tarikh: **12 MAY 2009**

Tarikh: : **12 MAY 2009**

CATATAN:
- * Potong yang tidak berkenaan.
- ** Jika tesis ini SULIT atau TERHAD, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali tempoh tesis ini perlu dikelaskan sebagai atau TERHAD.
- ♦ Tesis dimaksudkan sebagai tesis bagi Ijazah doktor Falsafah dan Sarjana secara Penyelidikan, atau disertasi bagi pengajian secara kerja kursus dan penyelidikan, atau Laporan Projek Sarjana Muda (PSM).

# REAL TIME FACE DETECTION SYSTEM

AMY SAFRINA BINTI MOHD ALI

This thesis is submitted as partial fulfillment of the requirements for the award of the Bachelor of Electrical Engineering (Hons.) (Electronics)

Faculty of Electrical & Electronics Engineering
Universiti Malaysia Pahang

MAY, 2009

"I hereby acknowledge that the scope and quality of this thesis is qualified for the award of the Bachelor Degree of Electrical Engineering (Electronics)"

Signature : _____

Name : MOHD ZAMRI BIN IBRAHIM

Date : 12 MAY 2009

Signature : _____

Author : AMY SAFRINA BINTI MOHD ALI

Date : 12 MAY 2009

*Dedicated to mom, dad and the whole family*

*Thank you for everything…*

## ACKNOWLEDGEMENT

This is a fulfilling moment when the report has been completed successfully after a few months of hard work.

First of all I would like to express my gratitude and appreciation to my supervisor, Mr. Mohd Zamri Bin Ibrahim for all his cooperation, guidance, ideas, sharing, facilitation and advice throughout the semesters. Without his guidance, I won't be able to finish this report today.

Also, I would like to extend my gratitude to all staffs in the Faculty of Electrical Engineering UMP, who have been helping out to make my work successful.

I would like to thank my parents for showing great patience, support and understanding throughout the project time frame especially when I'm far from home. Also, to my siblings especially my sister, Nuratikah for giving me support and motivation. Many thanks to my friends and juniors who always gave me ideas and advices to keep on the right track.

Last but not least, I would like to extend my gratitude to everyone who has been helping me directly or indirectly from the beginning until the final stage of this project. All the helps and cooperation from various parties are truly appreciated.

# ABSTRACT

A face detection system is a computer application for automatically detecting a human face from digital image or video frame from a video source. This project is used web camera to capture the image in real time. This face detection system used Haar Classifier method to detect face and extract human face. Haar Classifier technique can detect human face very face and can achieve high detection accuracy. This system is build using Visual Studio C++ 8 edition and Opencv to setup the interface between web camera and computer. This system also used Graphical User Interface (GUI) to design client window. Besides that this system used Graphic Device Interface (GDI) library to select the interest region. This system can detect the face image and can automatically save the image. This system can be applied in the banking system to reduce the number of forgery.

# ABSTRAK

Sistem pengecam muka ialah applikasi komputer untuk mengesan muka dari imej digital atau paparan gambar bergerak dari sumber paparan gambar bergerak. Projek ini menggunakan kamera web untuk mengambil gambar masa nyata. Sistem pengesan muka ini menggunakan pengelasan Haar untuk mengesan muka dan mengekstrak muka. Teknik pengelasan Haar dapat mengesan muka dengan laju dan mampu mencapai tahap pengesanan yang tinggi. Sistem ini menggunakan *Visual Studio C++* edisi 8 dan *OpenCV* untuk membentuk antara muka diantara kamera web dan komputer. Sistem ini juga mengunakan grafik antara muka pengguna mereka tingkap pengguna. Sistem ini menggunakan alat grafik antara muka untuk memilih permukaan yang dikehendaki. Sistem ini dapat mengesan imej muka dan dapat menyimpan imej tersebut secara automatik. Sistem ini dapat di applikasikan pada sistem bank untuk mengurangkan kes penipuan.

# TABLE OF CONTENT

# LIST OF FIGURE

# LIST OF TABLE

# LIST OF ABBREVIATION

| | | |
|---|---|---|
| MFC | - | Microsoft Foundation Classes |
| GUI | - | Graphic User Interface |
| AdaBoost | - | Adaptive Boost |
| PC | - | Personal Computer |
| IDE | - | Integrated Development Environment |
| COM | - | Computer |
| GDI | - | Graphic Device Interface |
| ROI | - | Region of Interest |
| OpenCV | - | Open Source Computer Vision |
| USB | - | Universal Serial Bus |

# LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

A face detection system is a computer application for automatically detecting human face from a digital image or a video frame from a video source. Face detection is a pre-processing of face recognition. It is also used for the security system.

## 1.1 Overview

Face detection is an important first step for applications in several areas, including biometrics, human-computer interfaces, and surveillance. Nowadays, the importance of the automatic face detection and tracking system has increased as it is needed for video surveillance and new user interfaces. The goal of this research effort is to construct a face detection system using a webcam in real-time. This system used Visual Studio C++ to develop algorithms that are accurate and computationally efficient. This project is used Haar Classifier Technique to detect face location.

## 1.2 Objective

There are few objectives to design face detection system. The objective of face detection are :

- To design real time face detection system.
- To utilize the face detection system based on Haar Classifier.
- To develop face detection system using Visual C++ 8 edition.

## 1.3 Work scope

The scopes and guidelines are listed to ensure the research is conducted within its intended boundary. This is to ensure the research is heading to the right direction to achieve its intended objectives.

The first scope of this project is to develop a face detection based on Haar Classifier. Haar Classifier is used because it achieved high detection accuracy. Haar Classifier can efficiently reduce or increase the class variability and making the classification easier.

The second scope is to extract the human face. It will extract the desire image, for this system it will extract human face using Haar Classifier.

Another scope of this project is used software Visual C++ 8 edition to verify the algorithm and show the result. It is used to create face detection system that can detect face in real time.

## 1.4 Problem Statement

This project is to improve the face detection system by using Haar Classifier to get higher accuracy result. Haar Classifier is used for face detection because it can detect the desire image very fast. The algorithm has been used for the detection which achieved high detection accuracy.

## 1.5 Thesis Outline

Chapter 1will describes the introduction of this system, the objective of this project, the problem statement, the work scope and overview of this project.

Chapter 2 will review about the information find on all the material or data used in the development of the system.

Chapter 3 will explain all the method use to develop this system. This chapter will explain briefly about Visual C++ and OpenCV in designing the face detection system.

Chapter 4 will include all the results and the explanation about the results after all the development process has done.

Chapter 5 will show the summary after all and come up with some recommendations for some improvements.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

Face detection is a computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies. This system is using Haar Classifier Technique because it is most popular face detection algorithms. This chapter will discuss briefly about Opencv, Visual C++, Haar Like Features and the Canny Edge Detection.

## 2.2 OpenCV

OpenCV is a computer vision library and it focuses mainly on real-time image processing. OpenCV is a collection of algorithms and sample code for various computer vision problems. This library allows high level functions for computer vision and image processing. Image processing and computer vision algorithms are proposed to programmers in order to create powerful applications in the domain of digital vision. OpenCV offers many high-level data types [8]. OpenCV is opensource to run on many computer platforms. OpenCV uses the IplImage structure to create and handle images. OpenCV are complementary tools to build efficient and effective image processing and computer vision applications. OpenCV are written in C and C++ language and it contain over 500 functions. OpenCV also available on Windows, Linux and MacOSX. Example

applications of the OpenCV library are Human-Computer Interaction (HCI), Object Identification, Segmentation and Recognition, Face Recognition, Gesture Recognition, Motion Tracking, Ego Motion, Motion Understanding, Structure From Motion (SFM), Stereo and Multi-Camera Calibration and Depth Computation, Mobile Robotics.

## 2.3 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It can be used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight [9].

Other built-in tools include a forms designer for building Graphic User Interface (GUI) applications, web designer, class designer, and database schema designer. It allows plug-ins to be added that enhance the functionality at almost every level including adding support for source control systems (like Subversion and Visual SourceSafe) to adding new toolsets like editors and visual designers for domain-specific languages.

Visual Studio supports languages by means of language services, which allow any programming language to be supported by the code editor and debugger, provided a language specific service has been authored. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), and C# (via Visual C#). Language-specific versions of Visual Studio also exist which provide more limited language services to the user.

Visual C++ supports COM as well as the Microsoft Foundation Classes (MFC) library. For MFC development, it provides a set of wizards for creating and customizing MFC boilerplate code, and creating GUI applications using MFC.

This project has utilized Visual Studio C++ edition 2005 language and its component in creating the Graphic user Interface (GUI) to interfacing the hardware and the computer. Microsoft Visual C++ is the fastest and easiest way to create applications for the Microsoft Windows. Visual C++ provides a complete set of tool to simplify rapid application development. A few component of Microsoft Visual C++ is Microsoft DirectX SDK that implemented in this project has greatly improved the function of software.

The Visual C++ language was chosen as a programming language to setup the interface between hardware and PC. There are the ActiveX technology allows author to use the functionality provided by other applications that make easy to implement.

## 2.4 Webcam

**Figure 2.1** : Logitech Quickcam Pro 5000

Webcams are video capturing devices connected to computers or computer networks often using USB or if they connect to networks, ethernet or Wi-Fi. Webcam connected to PC can act as web-accessible cameras with certain software which the software uploads picture to a server, which can produce an input to system. Usually, this kind of software is programmed to work with almost every webcam. This software can be configured in many ways, and will often include options for image size and quality and time stamping images.

The QuickCam Pro 5000 have capability of acquiring 30 frames per second and capture still image for about 1.3 mega pixels. With the bundled, it is known it have software enhanced to help lighting condition. Logitech helps to achieve optimal performance under different light conditions.

In this project, the webcam function as the eye of this system to capture a real time image. The webcam will capture the image in the system.

## 2.5 Haar Like Features

Haar-like features are digital image features used in object detection. A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image.

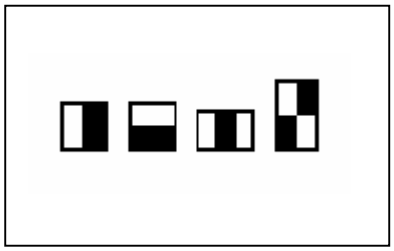

**Figure 2.2** : A set of basic of Haar Like Feature.(center)

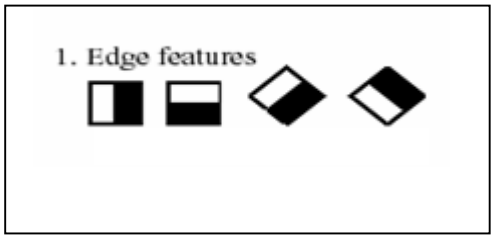


**Figure 2.3** : A set of extended Haar Like Feature ( edge features)

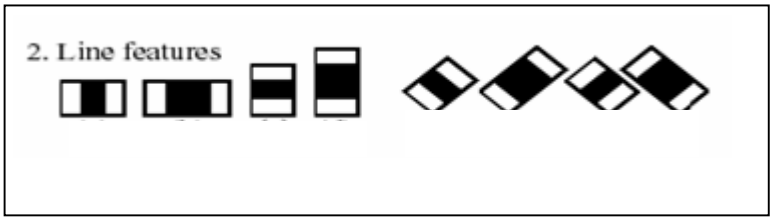


**Figure 2.4**: A set of extended Haar Like Feature (Line features)

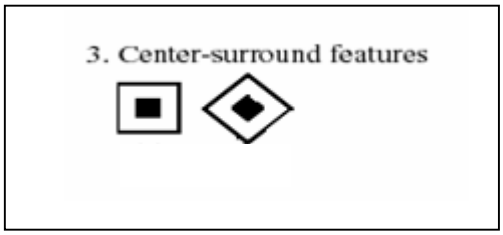


**Figure 2.5** : A set of extended Haar Like Feature (center-surrounded features).

Each Haar-like feature consists of two or three jointed “black” and “white” rectangles.The feature used in a particular classifier is specified by its shape (shown in Figure 2.2, 2.3, 2.4, 2.5), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in case of Figure 2.4 line feature the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature including the two white stripes and the black stripe in the middle and the sum of the image pixels under the black stripe multiplied by three in order to compensate for the differences in the size of areas. The sums of pixel values over a rectangular regions are calculated rapidly using integral images [7].

Haar features are based on Haar wavelets, which are functions that consist of a brief positive impulse followed of a brief negative impulse. In image processing, a Haar feature is the difference between the sum of all pixels in two or more regions. The value of a Haar-like feature is the difference between the sum of the pixel gray level values within the black and white rectangular regions. Compared with raw pixel values, Haar-like features can reduce or increase the in-class or out-of-class variability, and thus making classification easier. AdaBoost (Adaptive Boost) is an iterative learning algorithm to construct a “strong” classifier using only a training set and a “weak” learning algorithm. A “weak” classifier with the minimum classification error is selected by the learning algorithm at each iteration. The Adaboost boosting algorithm is used for feature selection by constructing a weak classifier out of each Haar feature [11]. Specifically, a threshold-based binary classifier is created from each Haar feature so that

the weighted training error is minimized. During each round of boosting, the single best weak classifier for that round is chosen corresponding to a particular Haar feature. The final result of boosting is a strong classifier whose output is computed as a thresholded linear combination of the weak classifiers. AdaBoost provides an effective learning algorithm and strong bounds on generalization performance. This classification method is fast and effective for face detection.

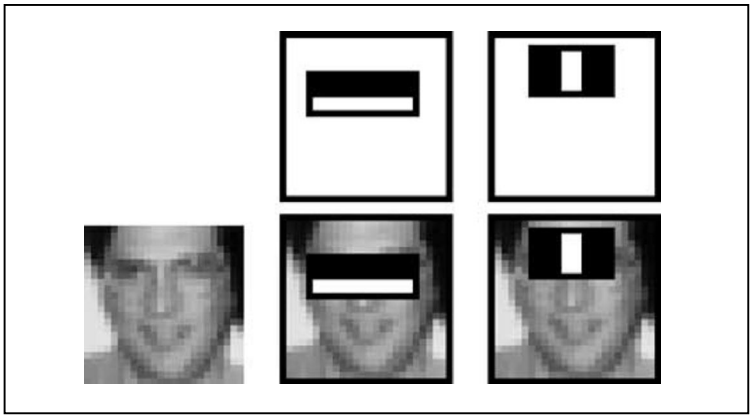

**Figure 2.6** : Haar Like Feature

The first and second features selected by AdaBoost. The two features are shown in the top row and then overlayed on a typical training face in the bottom row. The first feature measures the difference in intensity between the region of the eyes and a region across the upper cheeks. The feature capitalizes on the observation that the eye region is often darker than the cheeks. The second feature compares the intensities in the eye regions to the intensity across the bridge of the nose.

Figure 2.7 and figure 2.8 below shows the application of the real time face detection using Haar Classifier technique. This face detection system can be applied at the Automated Teller Machine (ATM) to detect the face of the ATM user. This system can reduce the forgery.

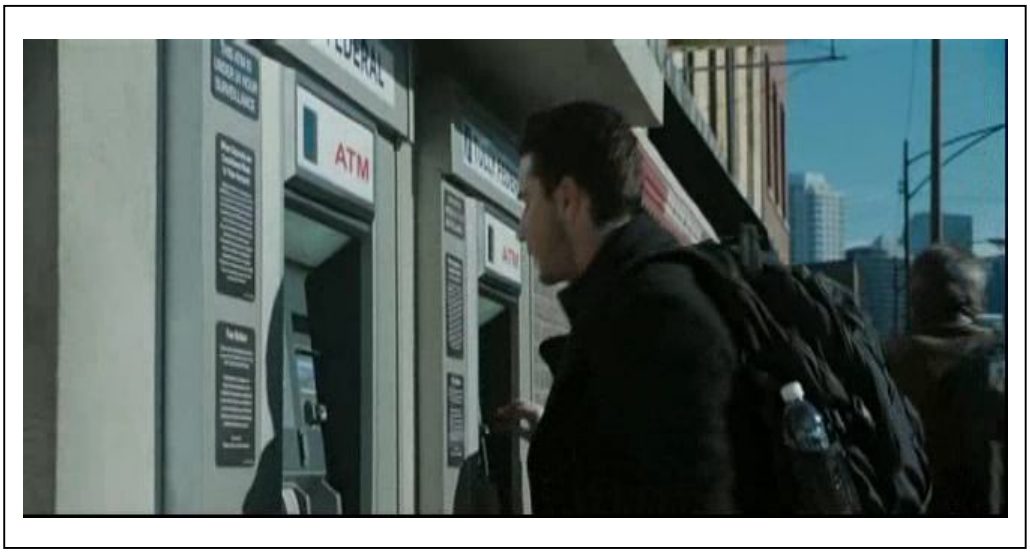


**Figure 2.7** : Automated Teller Machine (ATM).


**Figure 2.8** : The real time face detection.

## 2.5 Edge Detection System

Edges are the boundaries separating regions with different brightness or color. J.Canny suggested an efficient method for detecting edges. It takes grayscale image on input and returns bi-level image where non-zero pixels mark detected edges.

**Figure 2.9** : The implement of Canny Edge Detection.

Edge detecting an image significantly reduces the amount of data and filters out useless information, while preserving the important structural properties in an image. The Canny edge detection algorithm is known to many as the optimal edge detector. Canny's intentions were to enhance the many edge detectors already out at the time he started his work[12].

The edge detection system is to mark the points in a digital image at which the luminous intensity changes sharply. Sharp changes in image properties usually reflect important events and changes in properties of the world. Edges are significant local changes of intensity in an image. Edges typically occur on the boundary between two different regions in an image [10].

The effect of the Canny operator is determined by three parameters, the width of the Gaussian kernel used in the smoothing phase, and the upper and lower thresholds used by the tracker. Increasing the width of the Gaussian kernel reduces the detector's sensitivity to noise, at the expense of losing some of the finer detail in the image. The

localization error in the detected edges also increases slightly as the Gaussian width is increased.

In this system, the Canny edge detection is used to reducing the angle of gradient to one of the four sectors as shown in figure 2.10 below. The algorithm passes the 3x3 neighborhood across the magnitude array. At each point the center element of the neighborhood is compared with its two neighbors along line of the gradient given by the sector value. If the central value is non-maximum, that is, not greater than the neighbors, it is suppressed [4].

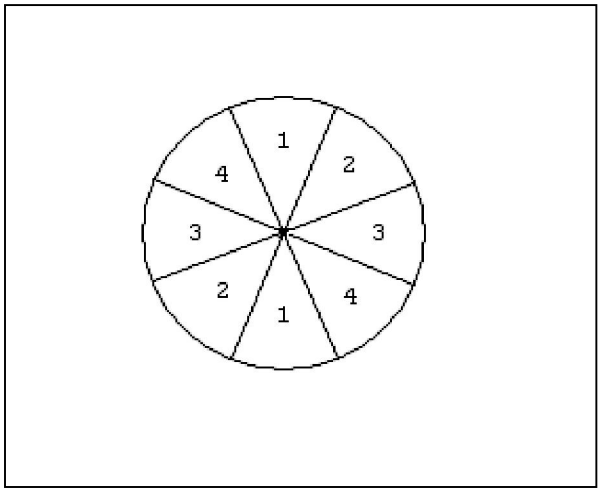


**Figure 2.10** : Angle of gradient.

# CHAPTER 3

# METHODOLOGY

## 3.0 Introduction

This chapter will explain about Haar Classifier method to detect the face using OpenCV library and Visual C++. This chapter also will explain about how to create the openCV library and the step to create the dialog based application to design the face detection system. This chapter will discuss about software implementation. It will discuss briefly how to capture the image, detect image, extract and store the image.

## 3.1 System Framework

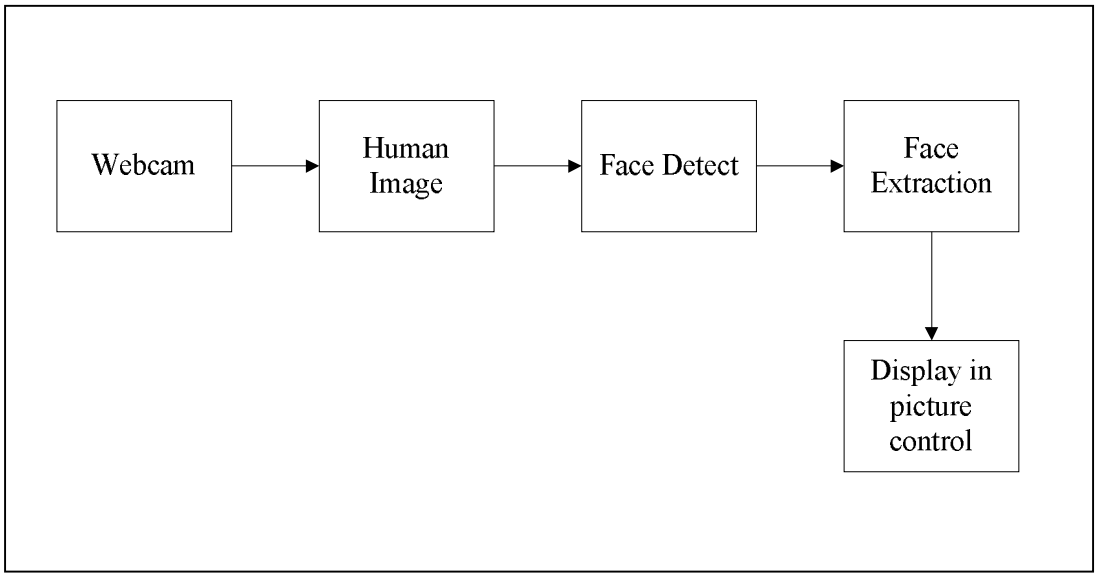


**Figure 3.1**: Block Diagram of Face Detection System.

In this system, firstly the webcam will capture the image using OpenCV. Then the system will detect face location in real time. Haar Classifier is used to detect the face location. After the system detects the face location, the face image will be extract. Lastly the image will display in picture control and the image will be saved automatically.

## 3.2 Software Design

### 3.2.1 Microsoft Visual C++

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It used to develop Graphical User Interface (GUI) application. The Visual C++ language was chosen as a programming language to setup the interface between hardware and PC.

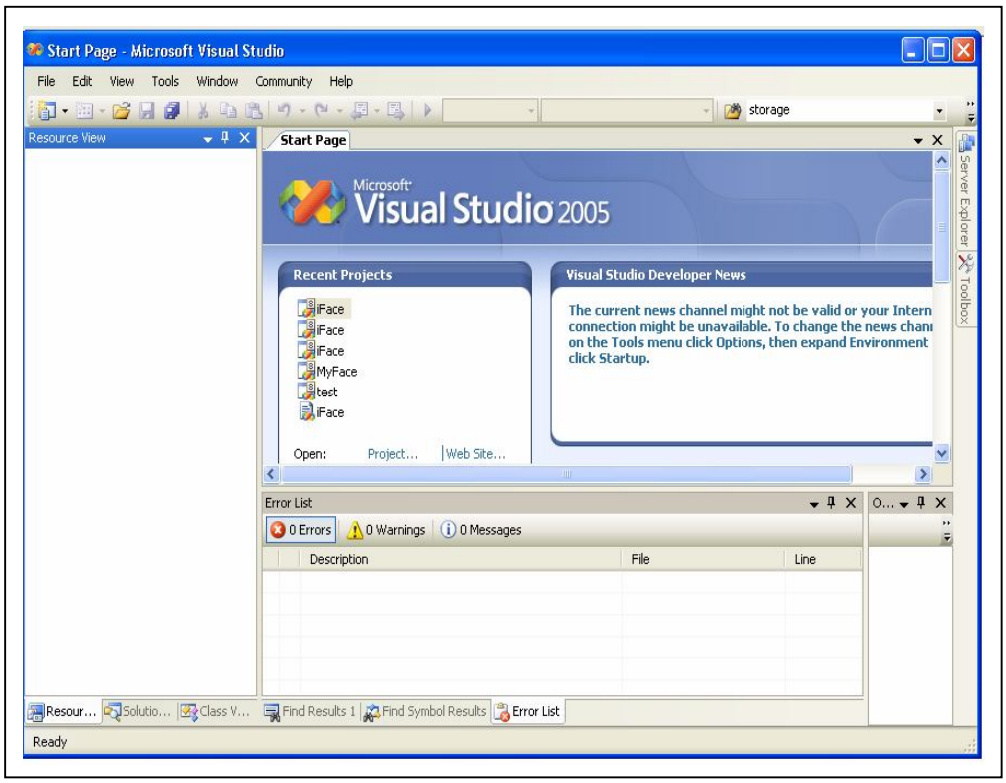


**Figure 3.2** : Microsoft Visual Studio main page

### 3.2.2 Set-up the Application

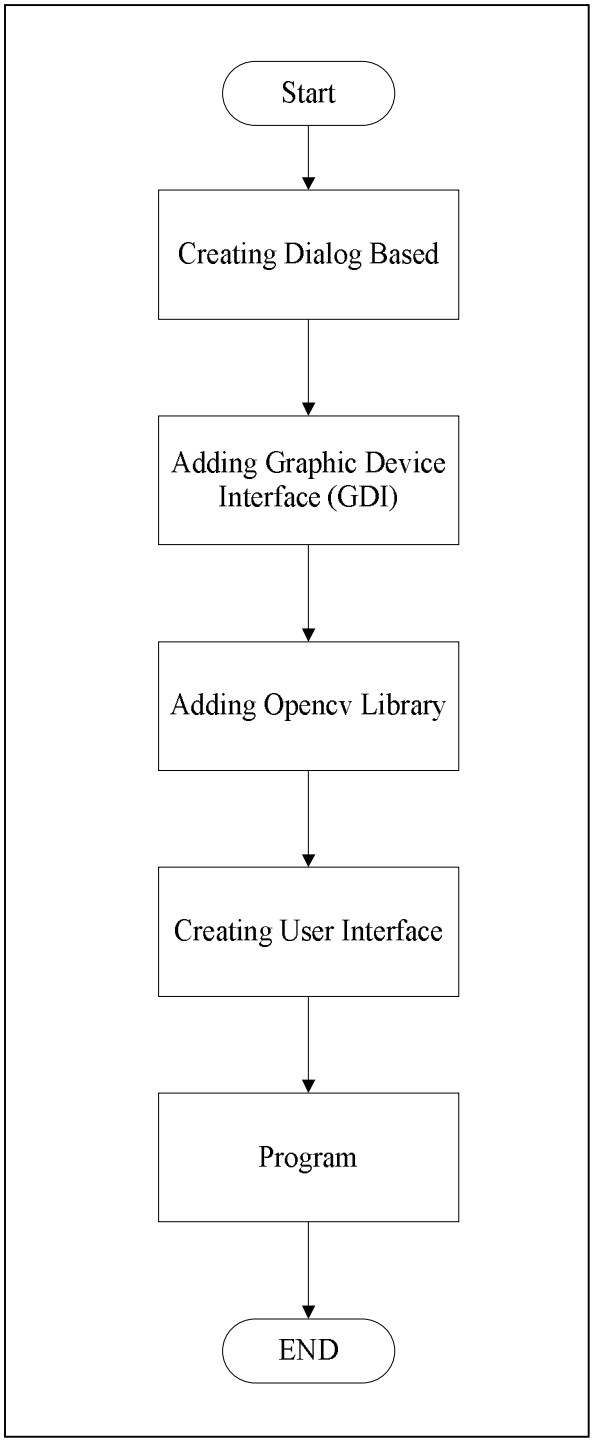


**Figure 3.3 :** Flowchart Set-up the application

### 3.2.2.1 Creating Dialog-based Application

This project used Microsoft Foundation Classes (MFC) to design the user interface. Firstly, MFC application was chosen to design face detection system. To create the dialog based application select “new project” at file button. Then the figure 3.4 will appear. Next click “MFC” and choose MFC Application and name the project.

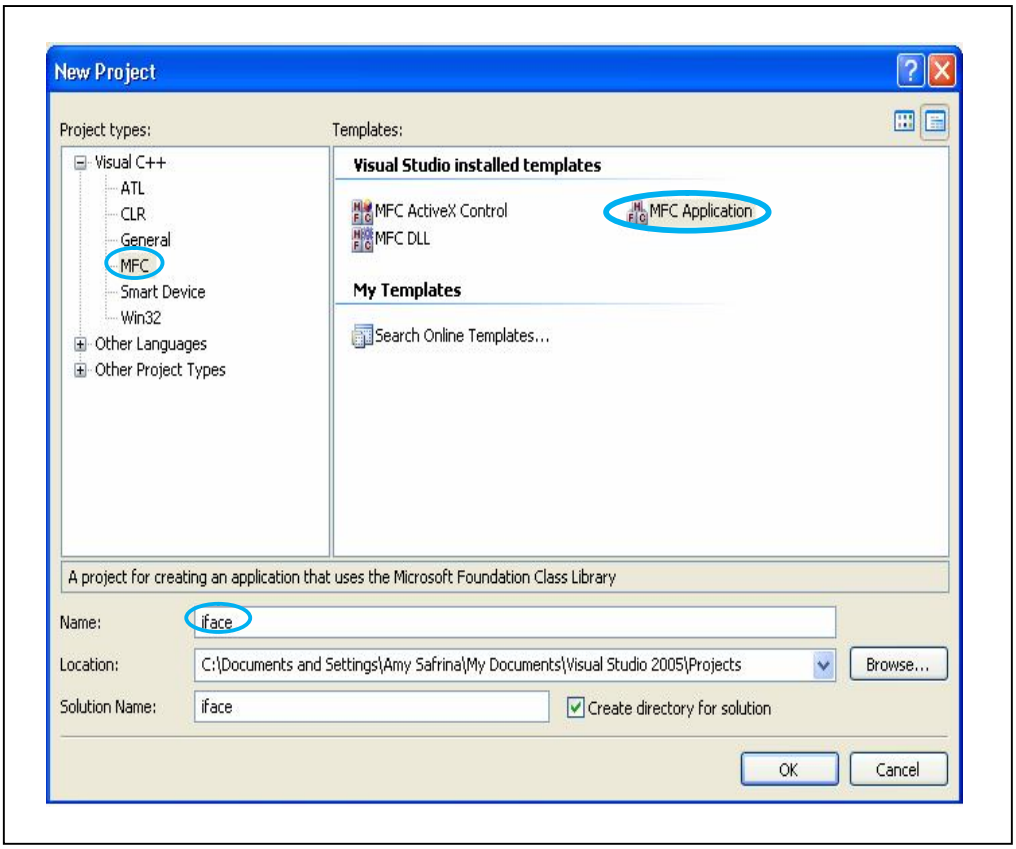


**Figure 3.4** : Create MFC application.

### 3.2.2.2 Adding Graphic Device Interface Library

The header files, resource files and source files will generate. The Graphic Device Interface (GDI) which can be downloading from Codersource.net has to add in this system. To add gdi.lib right click at the name of the project (iFace for this project) then choose "Add" and "Existing Item" shown in figure 3.5. Then add GdiPlus.lib to this system.

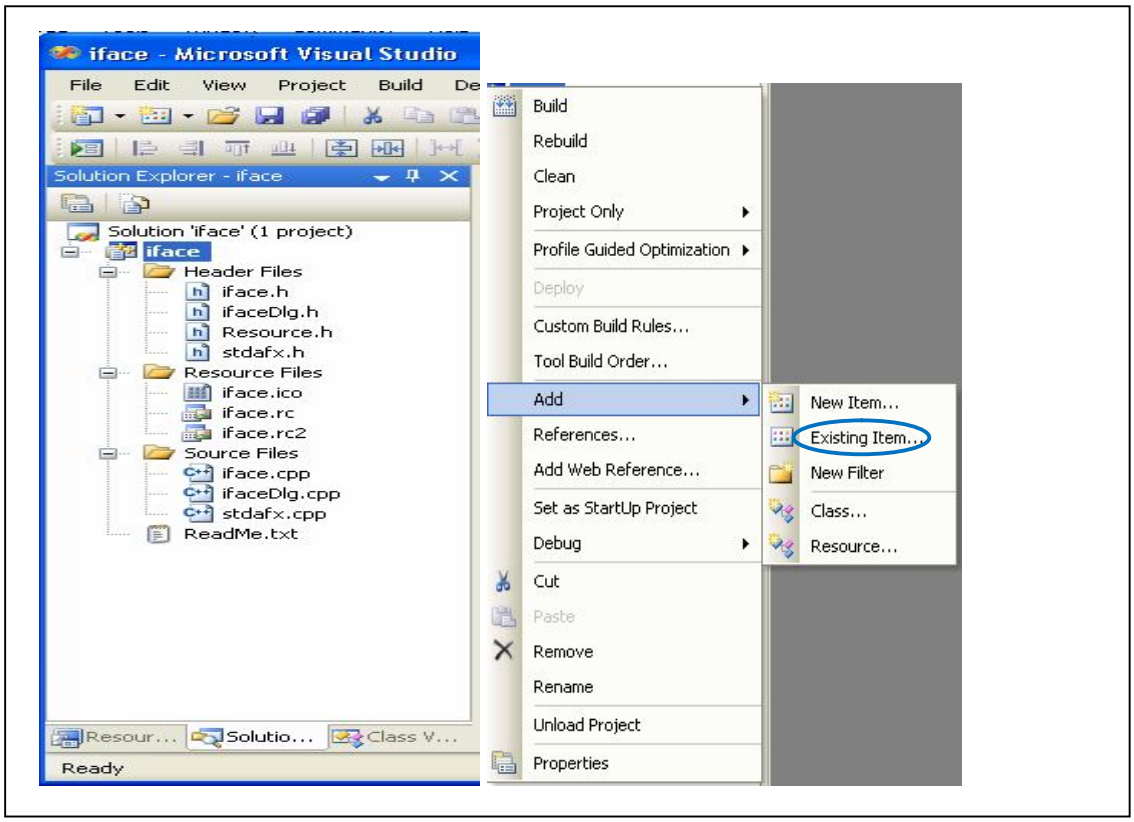


**Figure 3.5** : Solution window.

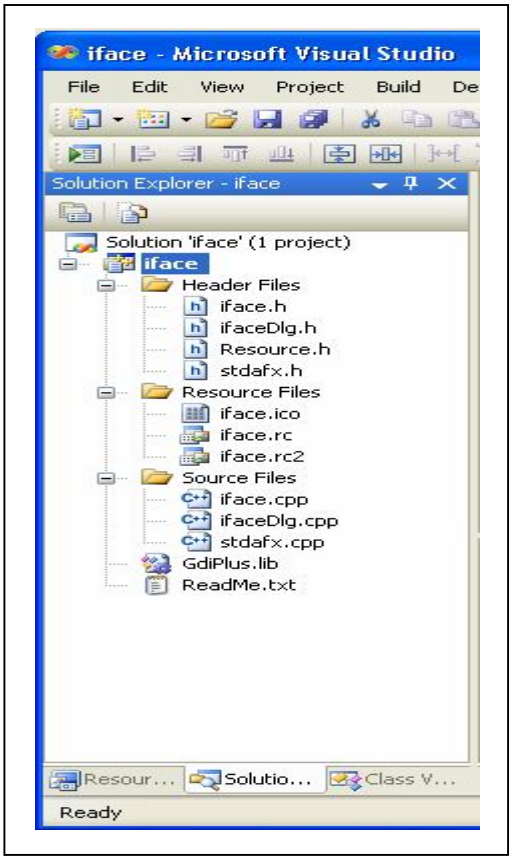


**Figure 3.6** : Solution Window after GdiPlus.lib was added.

### 3.2.2.3 Adding OpenCV Library

This project used OpenCV to run the face detection system. In order to build project using OpenCV the required library and directives must be include in project properties. To include additional directives, right click at the project name (iFace) then "C/C++" will be chosen. Then click at "General" under "C/C++" to include the Opencv library.

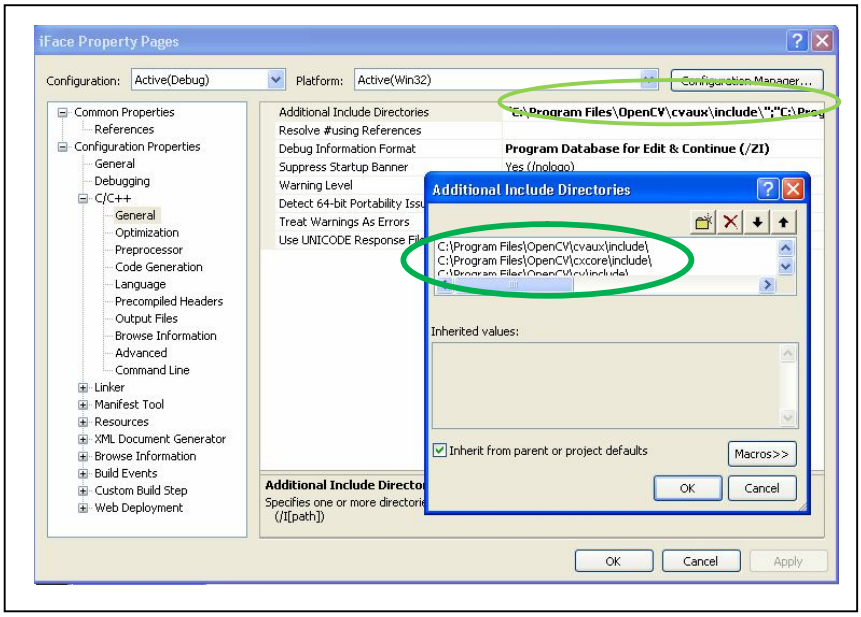


**Figure 3.7** : Project Properties window.

The utilized directive that have to include in Visual C++ are :

- C:\Program Files\OpenCV\cvaux\include\
- C:\Program Files\OpenCV\cxcore\include\
- C:\Program Files\OpenCV\cv\include\
- C:\Program Files\OpenCV\otherlibs\highgui\
- C:\Program Files\OpenCV\otherlibs\cvcam\include\

### 3.2.3 Creating User Interface Window

To create user interface, firstly double click at “IDD_IFACE_DIALOG”. To create button, drag the button from dialog editor at the toolbox to the form. Same step to create picture control. Then double click at the button or picture control to write the program shown in figure 3.8.

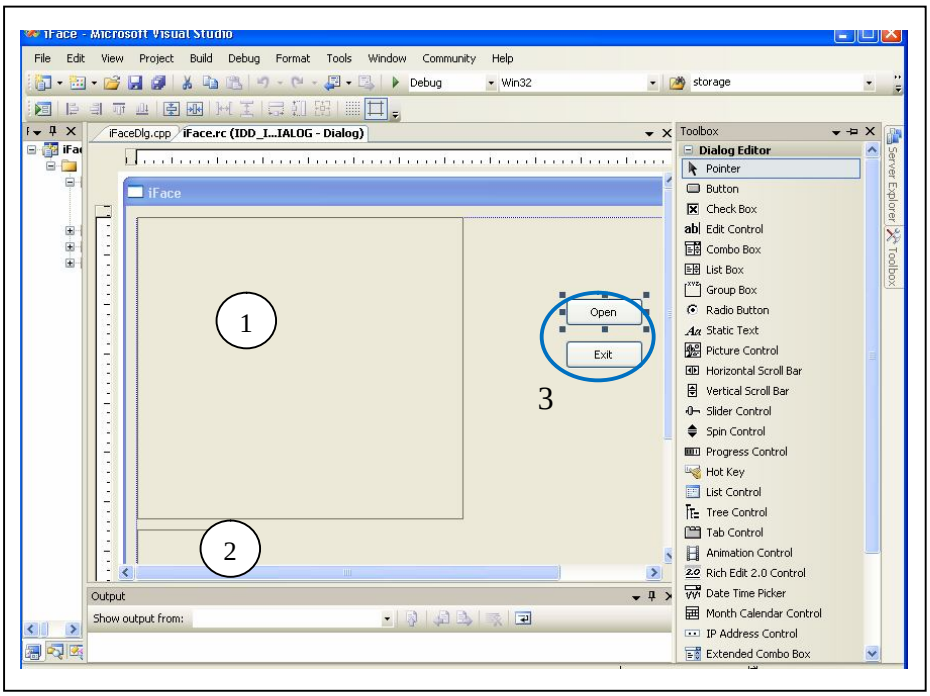


**Figure 3.8** : User Interface Dialog Box.

**Table 3.1** : Description of the dialog editor.

| No | Dialog Editor | Description |
|---|---|---|
| 1 | Picture Control | Display image |
| 2 | Picture Control | Display image |
| 3 | Button | To initiate an action |

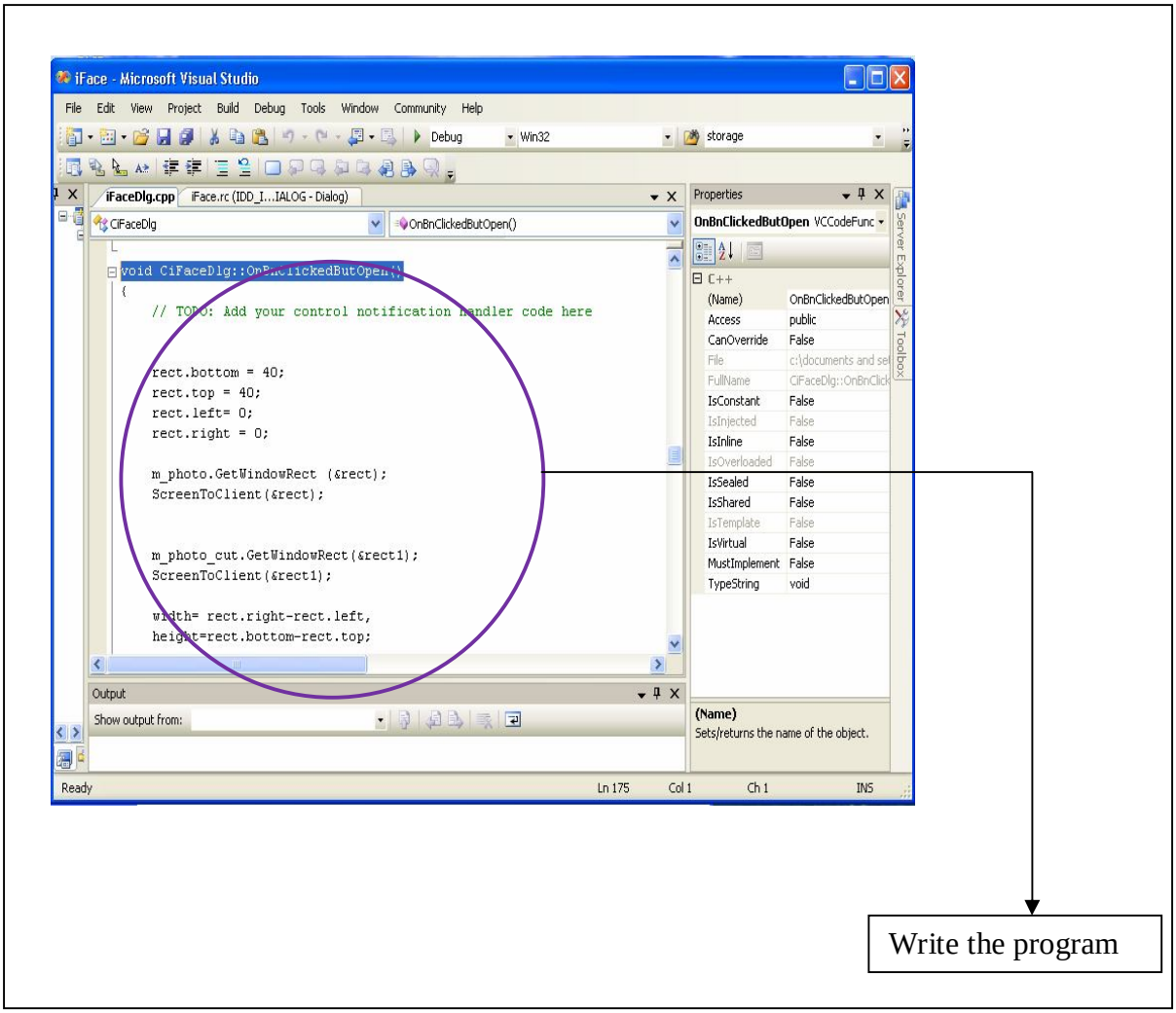


**Figure 3.9** : The dialog window.

## 3.3 Software Implementation

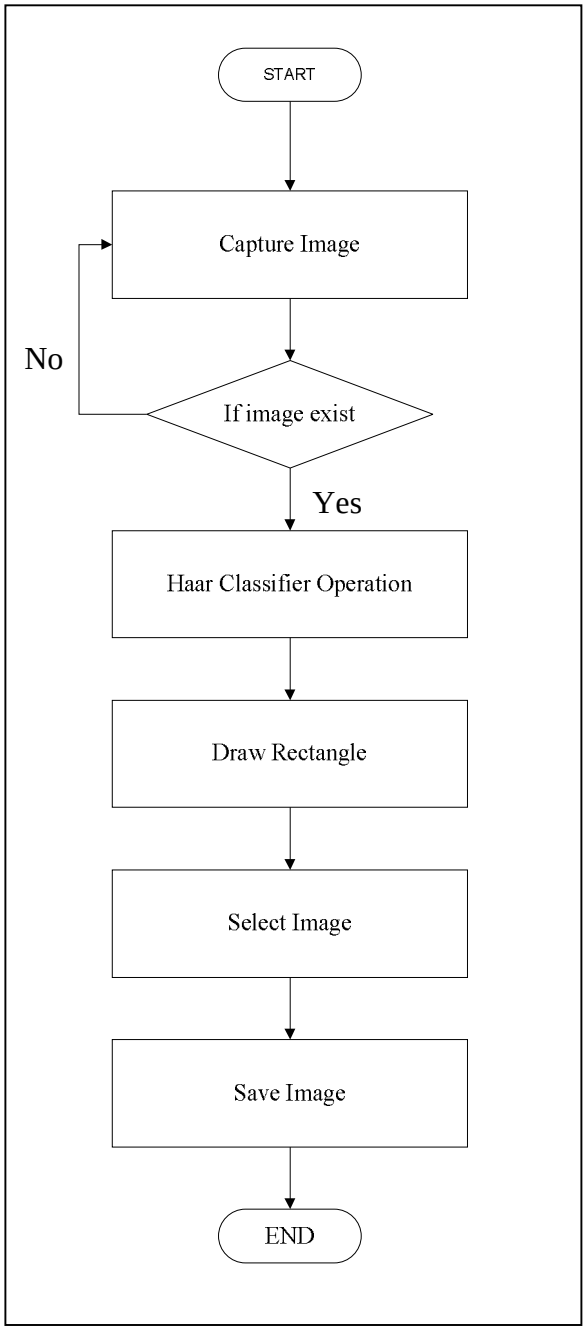


**Figure 3.10** : The flow diagram for the face detection

### 3.3.1 Establish Webcam

In this system, webcam connection is used USB connector. In Visual C++, must declare the “CvCapture*capture” function which give a signal to webcam to function with the system as shown in figure. The “CvCapture*capture” function is important to make sure the webcam can function smoothly with the system. Besides “IplImage*frame, *frame_copy” also need to be declare.

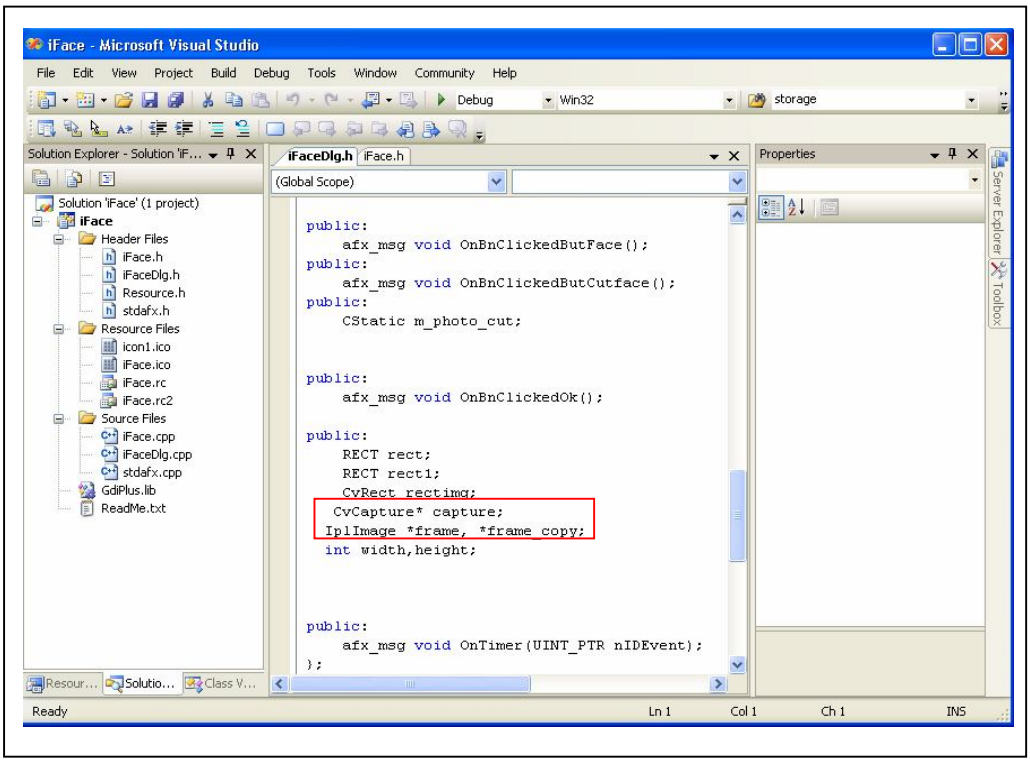


**Figure 3.11** : Header Files

### 3.3.2 Capturing Image

This coding is used to capture image in real time and save the image in the frame_copy. The system always looping for capturing frame (image).

```
for(;;)
  {

    if (!cvGrabFrame(capture))
      break;

    frame = cvRetrieveFrame(capture);

    if (!frame)
    break;

    frame_copy = cvCreateImage( cvSize(frame->width,frame-
         >height),IPL_DEPTH_8U, frame-
          >nChannels);

   if( frame->origin == IPL_ORIGIN_TL )
   cvCopy( frame, frame_copy, 0 );

   else

   cvFlip( frame, frame_copy, 0 );
           }


 cvReleaseImage(&frame_copy);
 cvReleaseCapture(&capture);
```

**Figure 3.12** : Source code for capturing image

**Table 3.2:**Description of capturing image.

| No | Code | Description |
|---|---|---|
| 1 | cvGrabFrame(capture) | Capture the frame and load it in IplImage. |
| 2 | cvRetrieveFrame(capture) | Get the image |
| 3 | cvCreateImage | Create a new image in frame_copy. |
| 4 | cvCopy( frame, frame_copy, 0 ) | Copy the image from frame to frame_copy. |
| 5 | cvFlip( frame, frame_copy, 0 ) | Reflects an array around horizontal or vertical axis, or both. |
| 6 | cvReleaseImage(&frame_copy) | Release the image. |
| 7 | cvReleaseCapture(&capture) | Release the capture memory. |

### 3.3.3 Display Image

The image from the camera will display in picture control Visual C++.

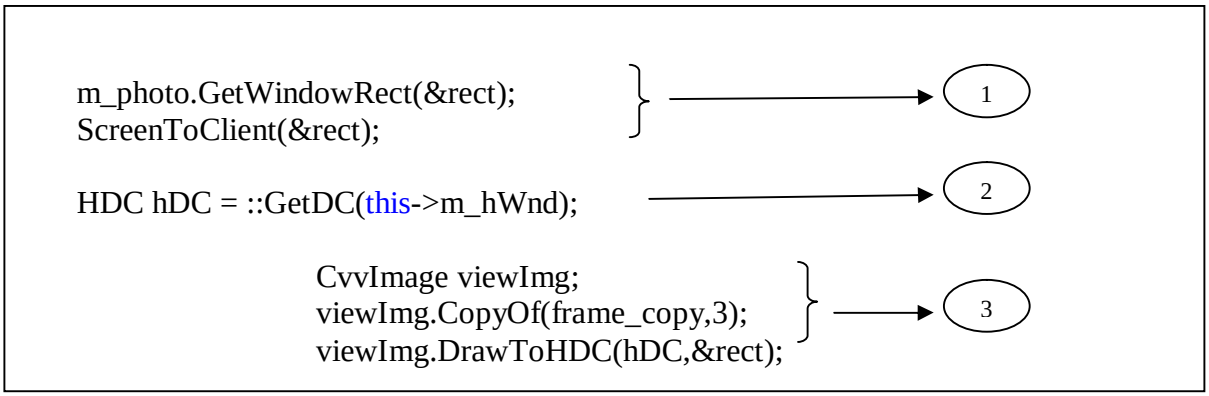


**Figure 3.13** : Source code for displaying the image.

**Table 3.3** : Description of the image display coding

| No | Code | Description |
|---|---|---|
| 1 | m_photo.GetWindowRect(&rect); | The picture control display |
| 2 | HDC hDC = ::GetDC(this->m_hWnd); | Control the image display in picture control |
| 3 | viewImg | To view the image in picture control |

### 3.3.4 Detecting Face

The face will detect using Haar and the face image will be store in sequence.

```
                                                    (1)      (2)   (3)
CvSeq* faces = cvHaarDetectObjects( frame_copy, cascade, storage,

1.1,2,0,CV_HAAR_DO_CANNY_PRUNING,cvSize(30, 30) );
                                                    (4)
```

**Figure 3.14** : Source code for detecting image.

**Table 3.4** : Description of cvHaarDetectObjects

| No. | Code | Description |
|---|---|---|
| 1 | Frame_copy | Contain image from the camera |
| 2 | cascade | Contain more than one image |
| 3 | storage | Store the image |
| 4 | cvSize(30,30) | The size of the image |

Graphic device interface was used to create rectangle for drawing the face. Before draw rectangle, the face dimension and scale have to find.

```
Pen blackpen(Color(255,255,0),3);
Graphics graphics(this->m_hWnd);
graphics.DrawRectangle(&blackpen,m_facepoint.x+11,m_facepoint.y+11,100,100);
```

(1) (6) (2) (3) (4) (5)

**Figure 3.15** : Source code for draw the rectangle

**Table 3.5** : Description of selected region coding.

| No | Code | Description |
|---|---|---|
| 1 | Graphics graphics(this->m_hWnd) | To control the rectangle image display at the picture control |
| 2 | &blackpen | Used pen to draw the rectangle |
| 3 | m_facepoint.x+11 | The face point at the x-axis |
| 4 | m_facepoint.x+11 | The face point at the y-axis |
| 5 | 100 | The width and the height of the rectangle |
| 6 | graphics.DrawRectangle | GDI function to draw rectangle |

### 3.3.5 Region of Interest

It is possible to select some rectangular part of the image. The selected rectangle is called Region of Interest (ROI). The structure IplImage contains the field roi for this purpose. If the pointer not null, it points to the structure ROI that contains parameters of selected ROI, otherwise a whole image is considered selected.

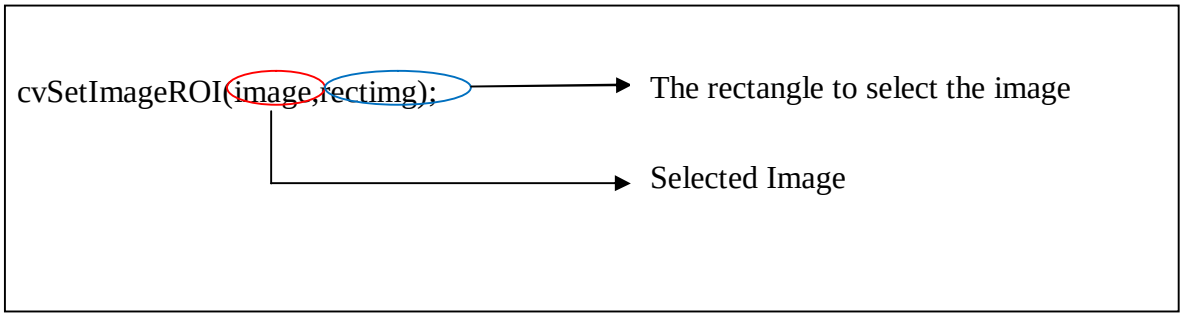


**Figure 3.16** : Source code for image selection.

### 3.3.6 Save Image

The face image that has been extract will be saved.

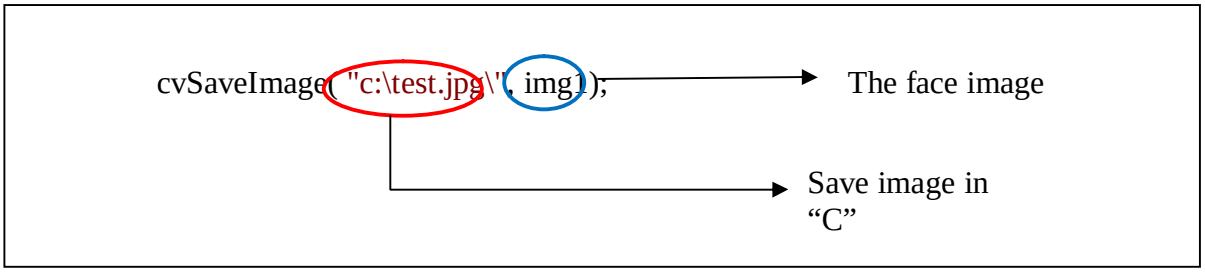


**Figure 3.17** : Source code for save the image.

# CHAPTER 4

# RESULT AND DISCUSSION

## 4.0 Introduction

This chapter will discuss about the result that obtain from this project. This chapter also will explain about the user interface window and the result that obtain when capture no image, capture still face, capture moving face, capture face from magazine, capture multiple face and capture the other object. This chapter also will explain about the error occur during the system running.

## 4.1 User Interface

This face detection system was used Visual C++ to setup the interface between the camera and PC. In this system, Visual C++ language and openCV was chosen to make sure the system running smoothly. The Graphical User Interface (GUI) was used to create the user interface window. The main page of the user interface is shown in figure 4.1 below. Button “Open” is for capturing image and detecting human face image. While button “Exit” is used to closed or end this system. Lastly button “Info” will describe the author.

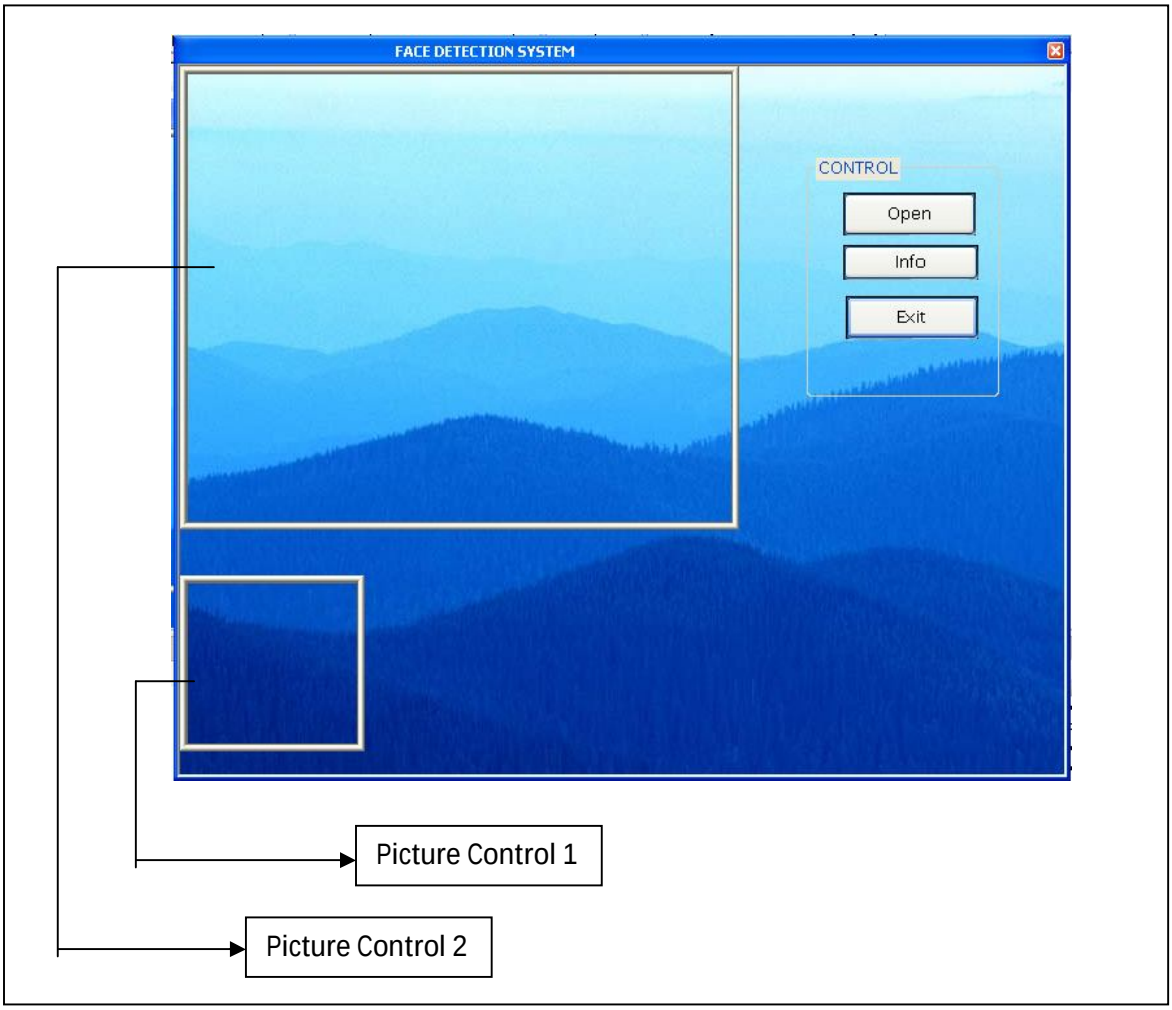


**Figure 4.1** : User Interface Window

## 4.2 Experimental

### 4.2.1 No Image

When the “Open” button was clicked, the camera automatically will function. OpenCV library was used to make sure the camera functioning. There will no detection if there no image. The figure 4.2 below show that the system was not detects anything where there no rectangle was draw in the picture control. If there no detection, there no image will be displayed in the picture control 2.

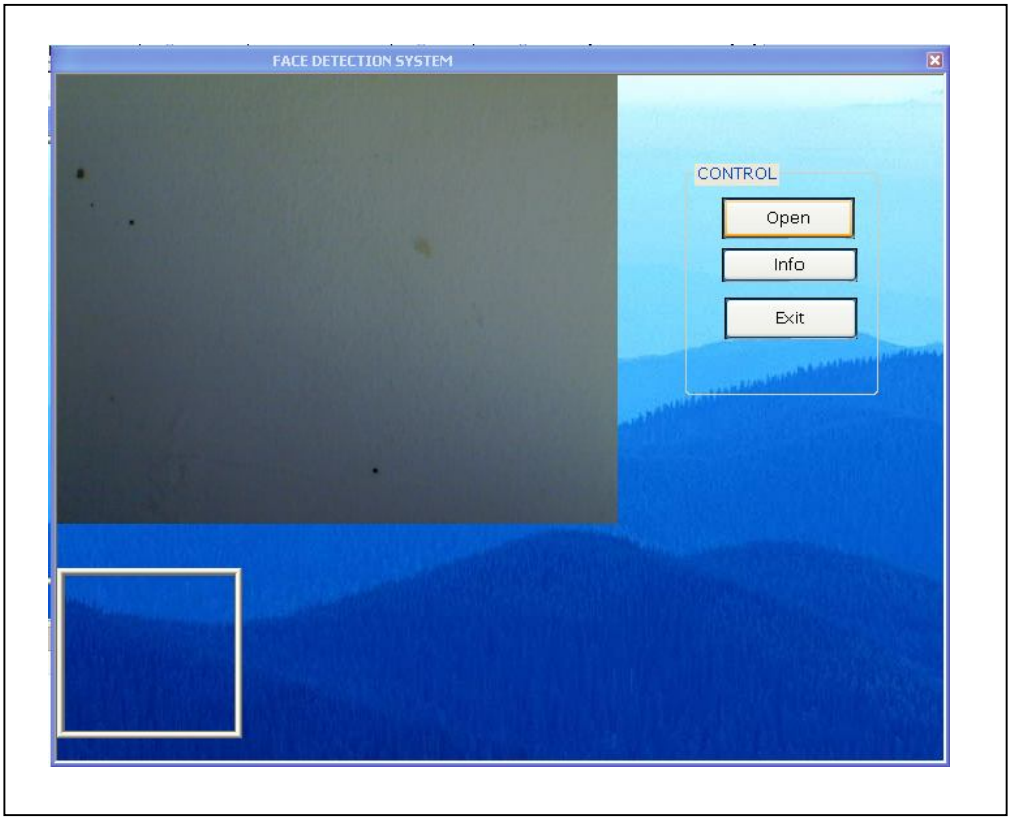


**Figure 4.2** : Camera ON but without image.

### 4.2.2 Human Image (still image)

When there was human image, the system will detect the human face image. Then the rectangle will be draw at the face as shown in figure 4.3. The rectangle was draw using graphical device interface (GDI). After the face was detected, the face image will display in the picture control 2. To display the face image in the picture control 2, selected rectangle is called region of interest (ROI) was used to selected the interest region.

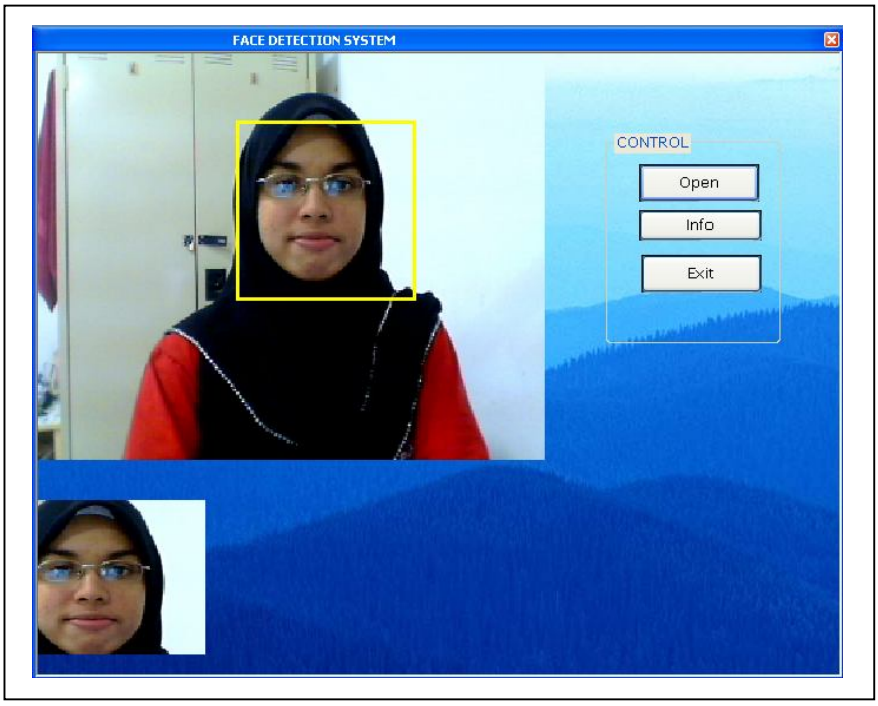


**Figure 4.3** : Camera ON with still human face image.

### 4.2.3 Moving image

This is a real time face detection system therefore the rectangle will followed the face image as shown in figure 4.4 below . The face region will displayed in picture control 2. The region which displayed in the picture control 2 also follow the face moving.

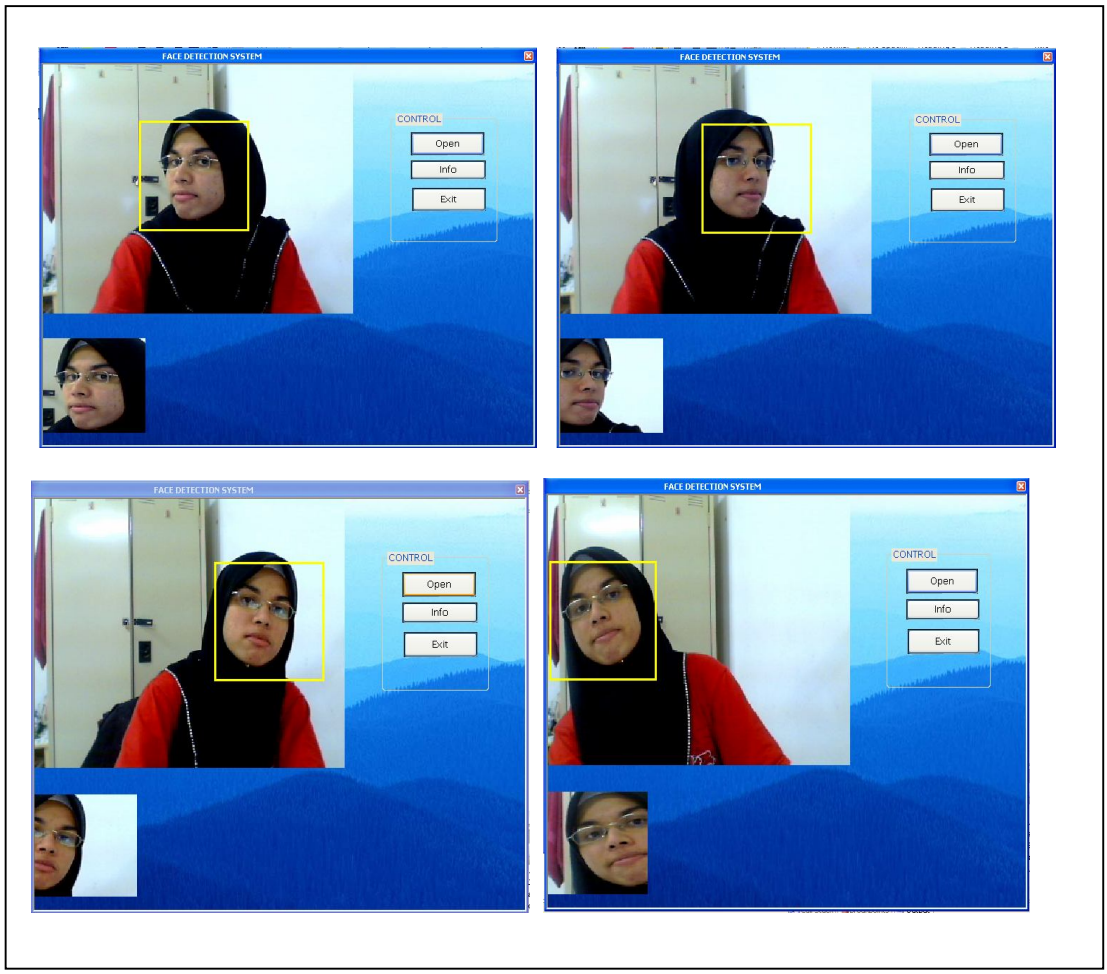


**Figure 4.4** : Camera ON with moving image

### 4.2.4 Multiple human image

This system can detect more than one face. This system had a problem to display more than one human face image in the picture control. This system cannot display multiple human face images in the same time.

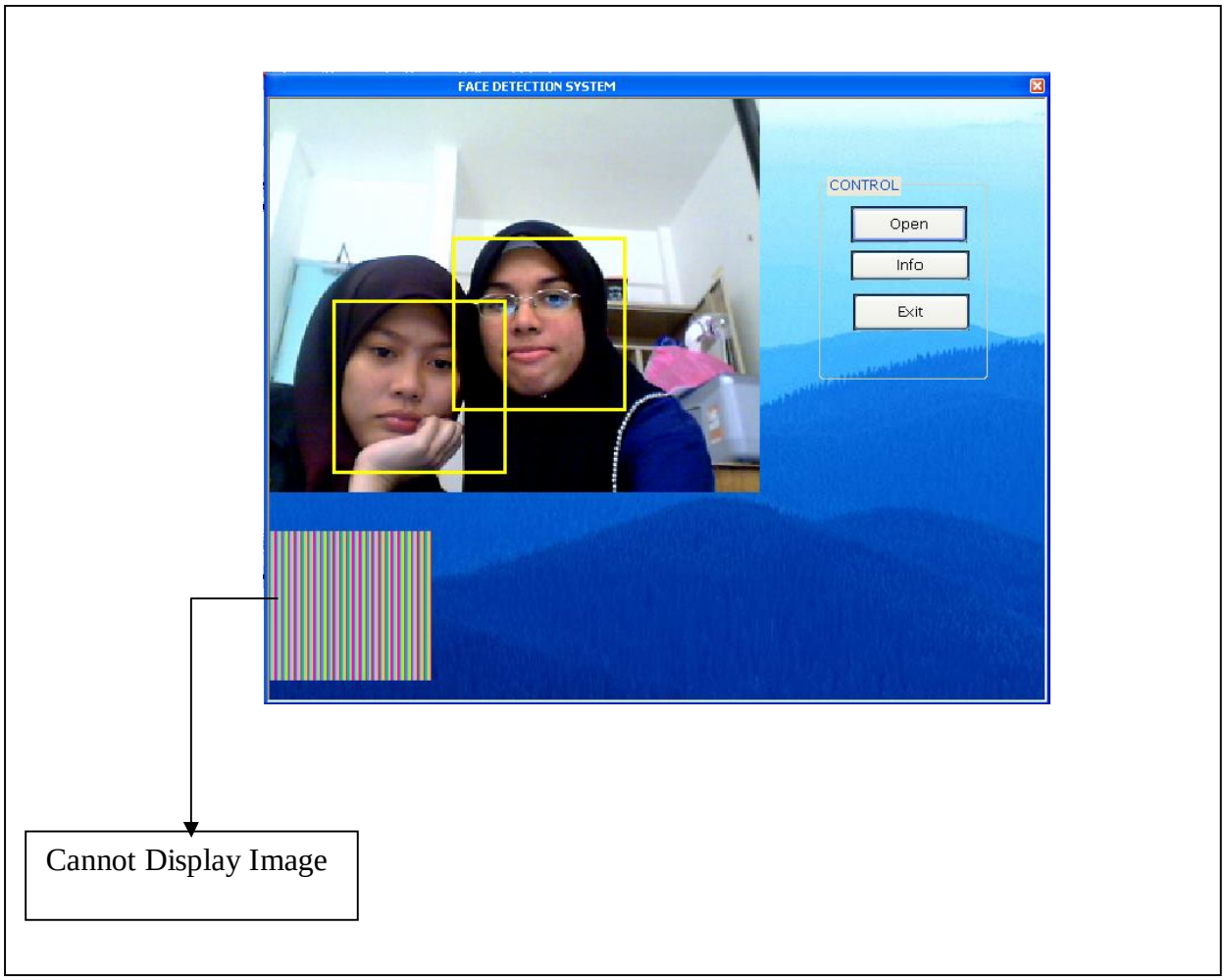


**Figure 4.5** : Multiple Image

### 4.2.5 Covered human face

This face detection system cannot detect the face, if the face is not complete shown as shown in figure 4.6. This is one of the weaknesses of this system. The system cannot find face dimension point. Therefore the system cannot detect the image. Face dimension point is the important point for detecting face region.

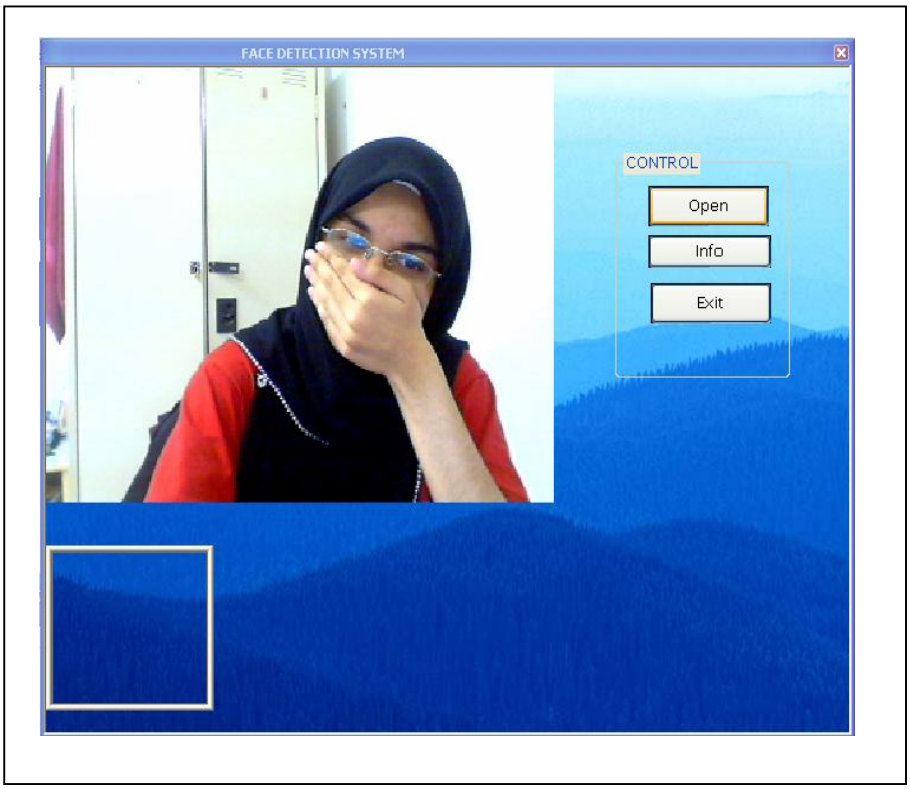


**Figure 4.6** : Camera ON with incomplete face image

### 4.2.6 Human face image from the magazine

This system also can detect human face from the magazine as shown in figure 4.7 below. This system can detect human face whether it is real human face or picture of human face. It is because the system still can find the face dimension point at the picture image.

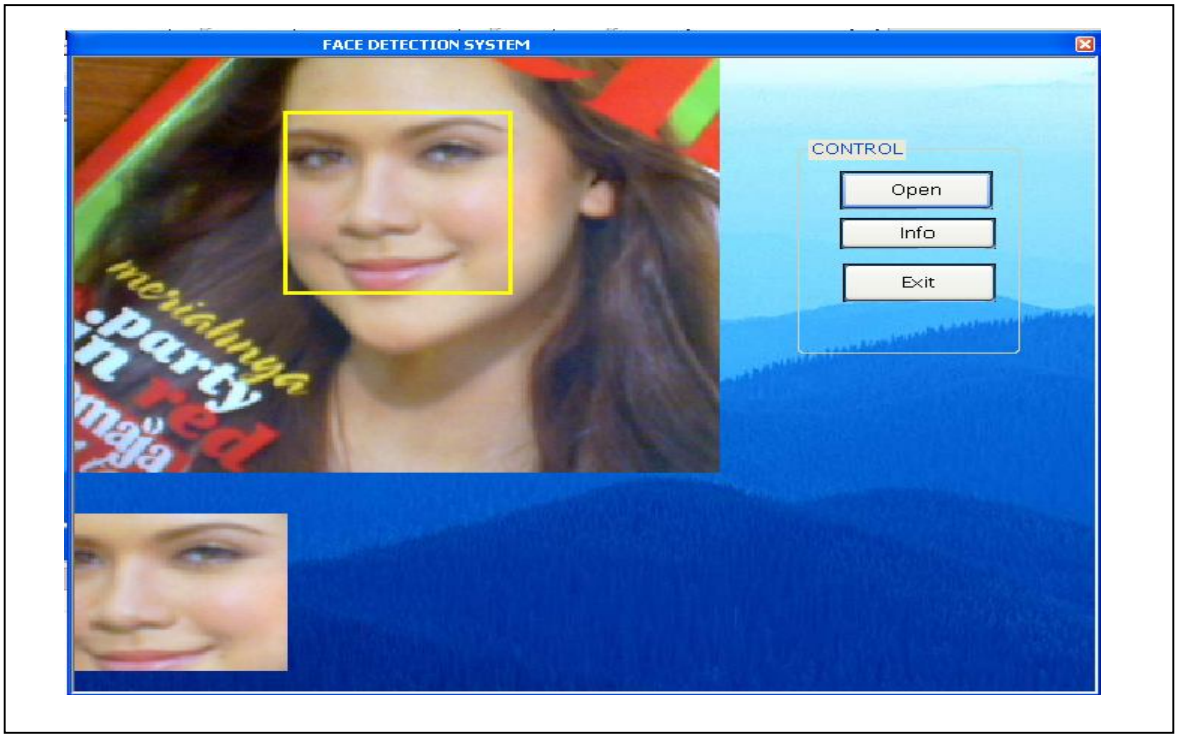


**Figure 4.7** : Camera ON with the human image in magazine.

### 4.2.7 Non face image

This system only detects human face. This system cannot detect other object as shown in figure 4.8 below. It is because this system only train face image using Haar training process. This process will calculate the threshold and convert in to xml file. Then the xml file will put into the coding. Therefore to detect other object just exchange the xml file.

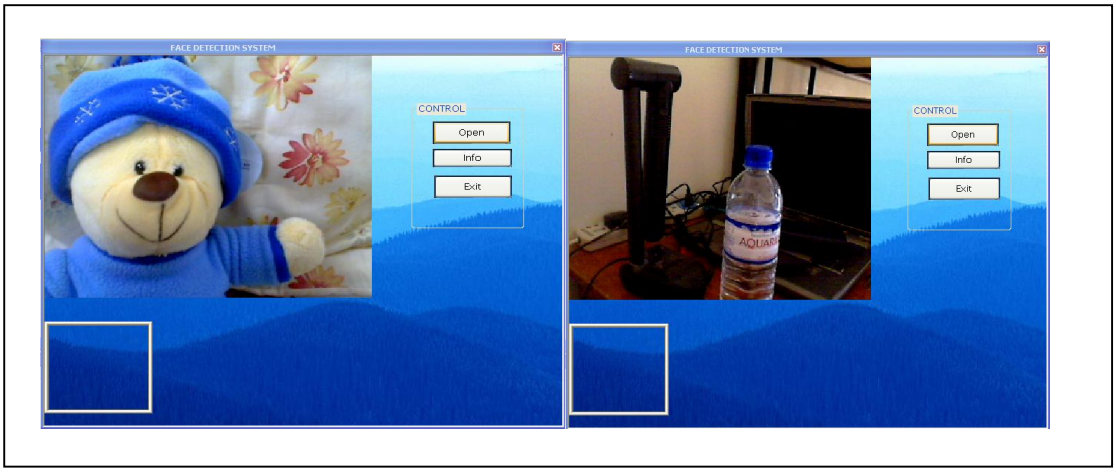


**Figure 4.8:** Camera ON with non face image.

## 4.3 Performance of Face Detection System

This system had a weakness where suddenly it can detect object as shown in figure 4.9 below. These occur because the system was not stable or there had an error in the face detection programming. This is also because this system does not have skin filter which the system can only detect the human skin color.

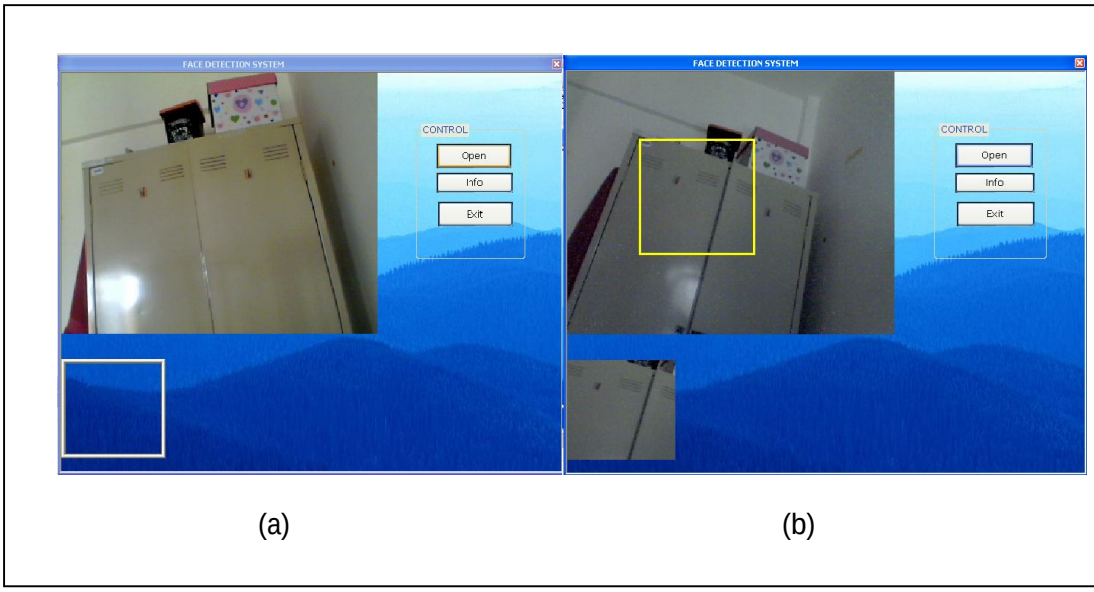


**Figure 4.9** : Camera ON with (a)no detection (b) detection.

## 4.4 System Design

When “Info” button was clicked, the information of the author window will appear as shown in figure 4.10 below. Button “OK” at the author window for closed the author window.

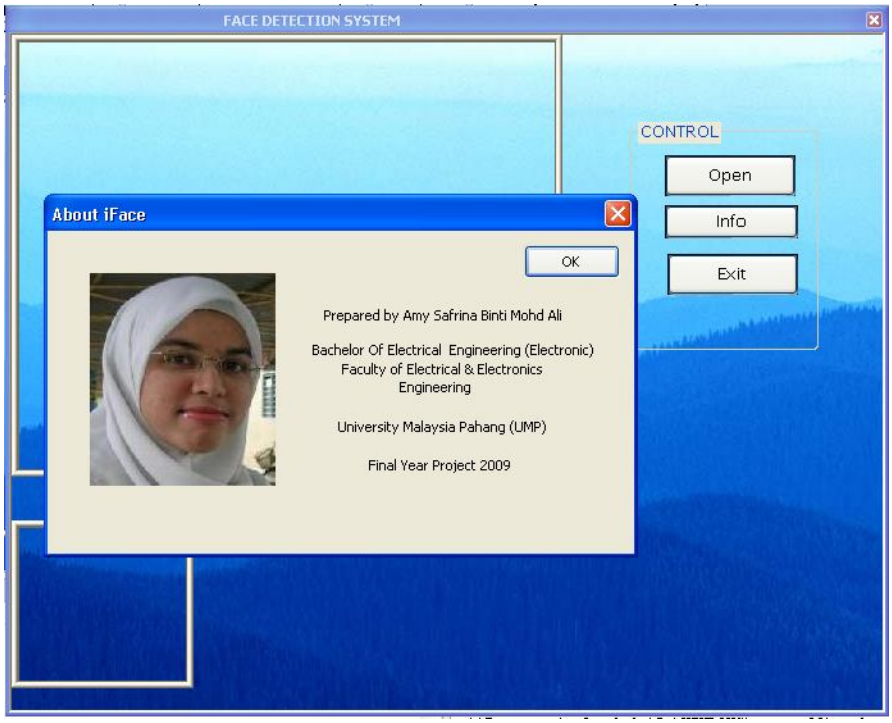


**Figure 4.10** : The information of the author window.

## 4.5 Costing

The overall cost of this project is RM 5 250. The software for face detection system is RM 5000 and the webcam that used to capture the image is RM 250.

# CHAPTER 5

# CONCLUSION AND RECOMMENDATION

OpenCV is a computer vision library and it focuses mainly on real-time image processing. This system is used Opencv to capture real time image. Opencv allows high level function for image processing. It also offers many high level data type for real time image. Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It used to develop Graphical User Interface (GUI) application. The Visual C++ language is used as a programming language to setup the interface between the camera and PC. Haar Like Feature is used for detecting faces. It can calculate very fast. The algorithm has been used for the detection which achieved high detection accuracy.

## 5.0 Conclusion

This face detection system is used to detect human face. This system is real time face detection. It can detect still image of human face, moving human face and human face from magazine, news paper or books.

When there is a face, rectangle will draw at the face region. The point of face dimension is important for detecting human face. Without this point, the rectangle cannot draw at the face. The rectangle is draw using graphical device interface (GDI). This system cannot detect if the face is not full shown. This happen because the system cannot find the face dimension.

This system will not detect other object. This system is set to detect human face by using Haar Classifier. This is because the system had train face image using haar training process.

## 5.1 Recommendation

For a future recommendation, proceed face detection to face recognition using Principle Component Analysis to build more security system. Principle Component Analysis is one of the most successful techniques that have been used in image recognition.

The second recommendation is to use human senses such as eye or nose for the detection system. Face detection used the whole face to detect, if the face is not fully shown the system cannot detect.

The system is unstable as at some times, it can detect other object than face. It is recommended to reprogram the system to improve and make it more stable. It also recommended adding the skin filter into this program to detect the human skin colour.

The last recommendation is to make the system can display more than one face image in the picture control. To make the system more efficient, design the system that can display the face image sequentially.

## 5.2 Commercialization

The face detection system can be commercialized in the market. This system can be used in the banking system to reduce the number of forgery. This system also can be used as a security system at home and supermarket to make sure the safety to the people.

## REFERENCES

[1] Biometric Authentication, A Machine Learning Approach, by Kung Mak Lin,2005.

[2] Fuzzy Models And Algorithms For Pattern Recognition And Image Processing, by James C. Bezdek, James Keller, Raghu & Nikhil, Series Editor.

[3] Invariant Object Recognition Based On Elastic Graph Matching, Theory And Application, by Raymond Lee & James Liu.

[4] Open Source Computer Vision Library, Reference Manual, by Intel Corporation, 2001.

[5] Detecting Faces In Images : A Survey, by Ming-Hsuan Yang, Member, IEEE, David J. Kriegman, Senior Member, IEEE, and Narendra Ahuja, Fellow, IEEE, January 2002.

[6] Face Detection And Tracking In Video Using Dynamic Programming, by Ziu Liu and Yao Wang, Department of Electrical Engineering Polytechnic University Brooklyn, 2000.

[7] Hand Detection with a Cascade of Boosted Classifier Using Haar-Like Features, by Qing Chen, University of Ottawa, May 2006.

[8] Robert Laganiere, 15 Jan 2009, URL http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow/cvision.htm.

[9] Microsoft Visual Studio, Wikipedia, 20 Feb 2009, URL http://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[10] Edge Detection, 4 March 2009, URL http://users.utcluj.ro/~tmarita/IPL/EdgeDetection.pdf

[11] Adaboost by Jiri Matas and Jan S˘ ochman, 13 March 2009, URL http://www.robots.ox.ac.uk/~az/lectures/cv/adaboost_matas.pdf

[12] Canny Edge Detection, A Computational Approach to Edge Detection, Nov 2000.

# APPENDIX

**PROGRAMMING FOR REAL TIME FACE DETECTION SYSTEM**

```
// iFaceDlg.cpp : implementation file

#include "stdafx.h"
#include "iFace.h"
#include "iFaceDlg.h"
#include <cv.h>
#include <highgui.h>
#include <cvaux.h>
#include <stdio.h>
#include <cmath>
#include <string.h>
#include <ctime>
#ifdef _DEBUG
#define new DEBUG_NEW
#endif

// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
        public:
         CAboutDlg();

// Dialog Data
         enum { IDD = IDD_ABOUTBOX };

         protected:
```

```
virtual void DoDataExchange(CDataExchange* pDX);

 // DDX/DDV support
// Implementation
protected:
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
}
void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
END_MESSAGE_MAP()

// CiFaceDlg dialog
CiFaceDlg::CiFaceDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CiFaceDlg::IDD, pParent)
{
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CiFaceDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	DDX_Control(pDX, IDC_PHOTO, m_photo);
```

```
	DDX_Control(pDX, IDC_PHOTO_CUT, m_photo_cut);
}

// CiFaceDlg message handlers

BOOL CiFaceDlg::OnInitDialog()
{
	CDialog::OnInitDialog();
	// IDM_ABOUTBOX must be in the system command
	  range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) ==		IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX,
strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon
```

```
	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CiFaceDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CiFaceDlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting
		SendMessage(WM_ICONERASEBKGND,
		reinterpret_cast<WPARAM>(dc.GetSafeHdc()),
		 0);
		// Center icon in client rectangle
```

```
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;
                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
      }
        // The system calls this function to obtain the cursor to
          display while the user drags
        //  the minimized window.
            HCURSOR CiFaceDlg::OnQueryDragIcon()
{
return static_cast<HCURSOR>(m_hIcon);
}

void CiFaceDlg::OnBnClickedButOpen()
{
        // TODO: Add your control notification handler code here
        rect.bottom = 40;
        rect.top = 40;
        rect.left= 0;
        rect.right = 0;

        m_photo.GetWindowRect (&rect);
```

```
        ScreenToClient(&rect);

        m_photo_cut.GetWindowRect(&rect1);
        ScreenToClient(&rect1);

        width= rect.right-rect.left,
         height=rect.bottom-rect.top;

        storage = 0;
        cascade = 0;
        capture = 0;
        cascade =
        (CvHaarClassifierCascade*)cvLoad("C://Program
Files/OpenCV/data/haarcascades/haarcascade_frontalface_alt2.x
ml", 0, 0, 0);
        storage = cvCreateMemstorage(0);
        capture = cvCaptureFromcam(0);


        if(capture)
        {
                SetTimer(1,50,NULL);
           }
       }

void CiFaceDlg::OnBnClickedOk()
{
        // TODO: Add your control notification handler code here
        OnOK();
}
```

```
void CiFaceDlg::OnTimer(UINT_PTR nIDEvent)
{
	// TODO: Add your message handler code here and/or call default
	if(nIDEvent == 1)
		{

		if (!cvGrabFrame(capture))
		Break;

		frame = cvRetrieveFrame(capture);

		if (!frame)
		break;
	frame_copy = cvCreateImage( cvSize(frame-
	>width,frame->height),IPL_DEPTH_8U, frame-
	 >nChannels);

	 if( frame->origin == IPL_ORIGIN_TL )
	 cvCopy( frame, frame_copy, 0 );

	 else

	    cvFlip( frame, frame_copy, 0 );
	    m_scalex=float(rect.right-rect.left)/(float)frame_copy->width;
	    m_scaley=float(rect.bottom-rect.top)/(float)frame_copy->height;		// x y
scale;
		HDC hDC = ::GetDC(this->m_hWnd);
		CvvImage viewImg;
		viewImg.CopyOf(frame_copy,3);
		viewImg.DrawToHDC(hDC,&rect);
```

```
 double scale = 1.3;
IplImage*gray = cvCreateImage(cvSize(frame_copy->width, frame_copy->height),8,1);
IplImage*small_img = cvCreateImage(cvSize(cvRound(frame_copy->width),cvRound(
frame_copy->height)),8,1);




  int i;

 cvCvtColor( frame_copy, gray, CV_BGR2GRAY);
cvResize(gray, small_img, CV_INTER_LINEAR);
cvEqualizeHist(small_img, small_img);
cvClearMemStorage(storage);

 if( cascade )
{
CvSeq*faces = cvHaarDetectObjects(frame_copy,          cascade, storage, 1.1, 2, 0,
cvSize(30, 30));


for( i = 0; i < ( faces->total : 0); i++)

{
      CvRect* r = (CvRect*)cvGetSeqElem( faces, i);
      CvPoint center;
      RECT rect;

center.x = cvRound((r->x + r->width*0.5));
```

```
center.y = cvRound((r->y + r->height*0.5));
radius = cvRound((r->width + r->height)*0.25*scale);

m_facepoint.x = center.x*m_scalex;
m_facepoint.y = center.y*m_scaley;

m_facerect.X = m_facepoint.x-radius*m_scalex*0.7+11;
m_facerect.Y = m_facepoint.y-radius*m_scaley*0.7+11;
m_facerect.Width=radius*2*m_scalex*0.9;
m_facerect.Height=radius*2*m_scaley*0.7;

float yy=(m_facerect.X + m_facerect.Width*0.5);
Pen blackpen(Color(255,255,0),3);
Graphics graphics(this->m_hWnd);
DrawRectangle(&blackpen,m_facepoint.x+11-75,m_facepoint.y+11-75,150,150);

rectimg.height = 100;
rectimg.width = 100;
rectimg.x = m_facepoint.x+10-100;
rectimg.y = m_facepoint.y+10-100;

IplImage *img1=cvCreateImage(cvSize(100,100),8,3);

cvSetImageROI(frame_copy, rectimg);

viewImg.CopyOf(frame_copy,3);
viewImg.DrawToHDC(hDC,&rect1);

 cvResize( frame_copy,img1 );
 cvSaveImage( "c:\\test.jpg", img1);
```

```
frame_copy = cvCreateImage( cvSize(frame_copy->width,frame-
>height),IPL_DEPTH_8U, frame_copy->nChannels);
        }
            }

  cvReleaseImage(&gray);
  cvReleaseImage(&small_img)
}

CDialog::OnTimer(nIDEvent);
}


void CiFaceDlg::OnBnClickedButton1()
{
        // TODO: Add your control notification handler code here
        CAboutDlg *Pdlg = new CAboutDlg;
        Pdlg->DoModal ();

        delete Pdlg;
}
```