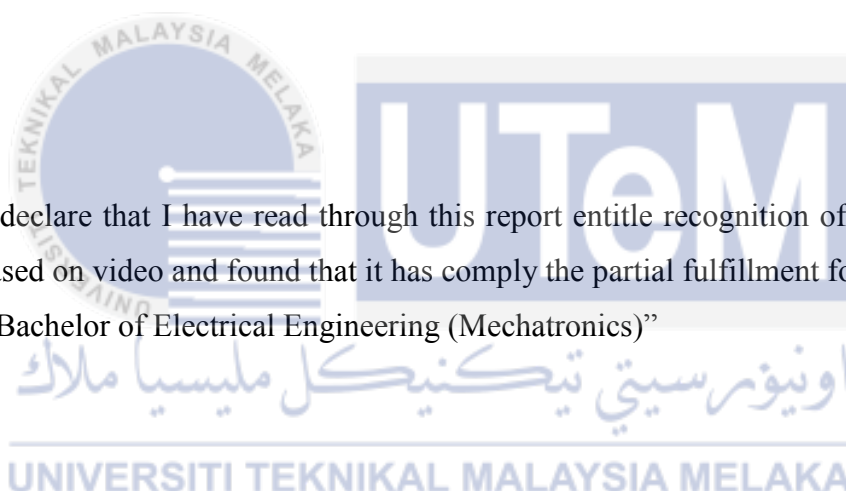


“I hereby declare that I have read through this report entitle recognition of face detection system based on video and found that it has comply the partial fulfillment for awarding the degree of Bachelor of Electrical Engineering (Mechatronics)”



Signature :

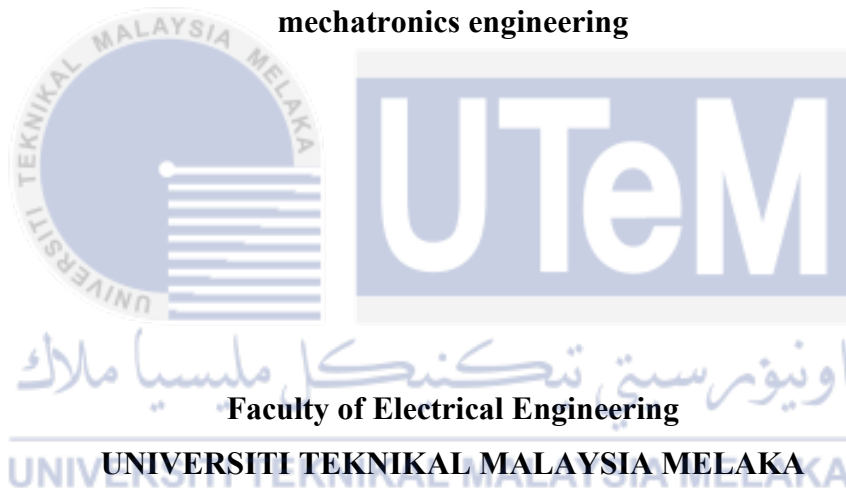
Supervisor's Name : NURSABILLILAH MOHD ALI

Date : 9/ 12 / 2015

RECOGNITION OF FACE DETECTION SYSTEM BASED ON VIDEO

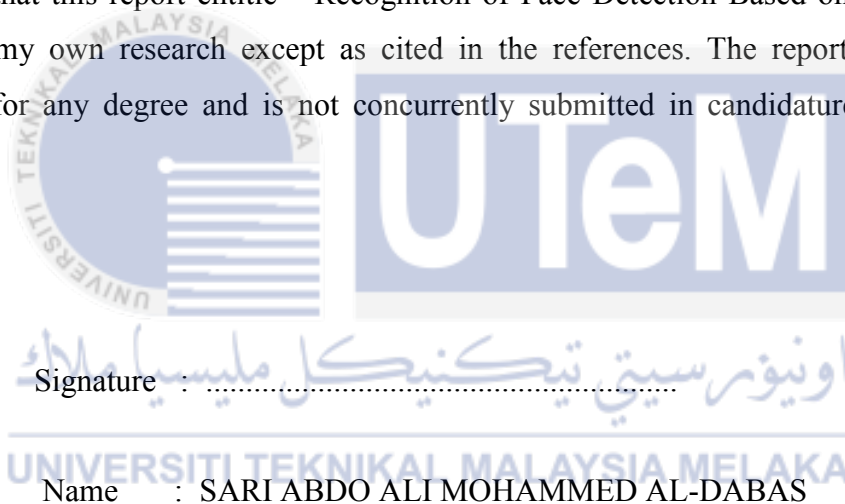
SARI ABDO ALI MOHAMMED

**A report submitted in partial fulfillment of the requirements for the degree of
mechatronics engineering**



2015

I declare that this report entitle “ Recognition of Face Detection Based on Video” is the result of my own research except as cited in the references. The report has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



Signature :

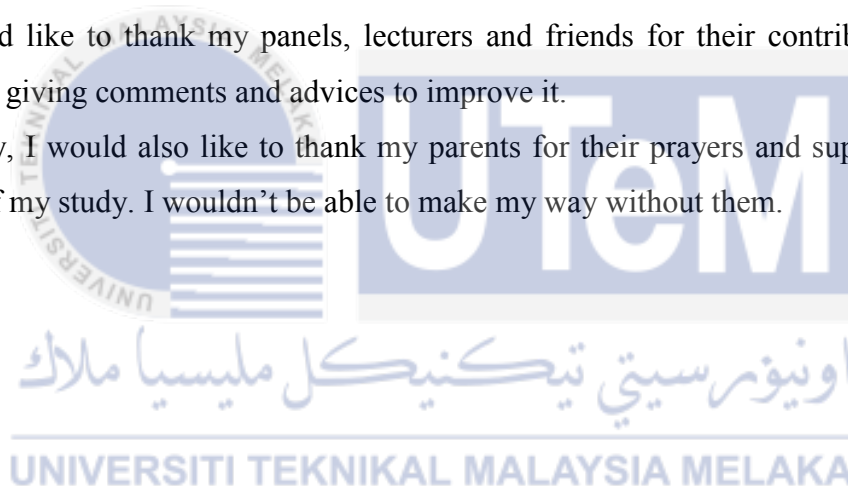
Name : SARI ABDO ALI MOHAMMED AL-DABAS

Date : 9 / 12 / 2015

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my supervisor NURSABILLILAH MOHD ALI for her sincere guidance along the project. Secondly, I also would like to thank my panels, lecturers and friends for their contribution into my project by giving comments and advices to improve it.

Lastly, I would also like to thank my parents for their prayers and support along the journey of my study. I wouldn't be able to make my way without them.



ABSTRACT

Nowadays, face recognition is an important aspect for security issues and identification. A lot of efforts were done by researches in the field of face recognition and detection. In this project of face recognition, the project divided into several stages, face detection, method of recognition and the database. Face recognition is implemented using visual studio 2010, OpenCV library and Emgu library. In order to link visual studio with OpenCV library there are some configurations has to be set before proceed with the coding process. There are three main aspects in this project; the first aspect is to develop an algorithm to detect frontal face. The second aspect is to develop an algorithm that recognizes a person identity on a video. The last aspect is to analyze the effects of illumination changes on the recognition. Face detection is the process of identify faces on video or on image and differentiate it from the objects on the background. Face detection is done using Haarcascade algorithm. Haarcascade classifier is trained using five hundred positive and negative images from internet. In order to get high rate of recognition, the image of the detected face has to be processed where it is converted from colored image (with green, blue and red colors) to grayscale image to reduce the data during the process. The image after that cropped and save the face only to reduce the noise of the background. Speeded up robust feature (SURF) is used to extract the facial features in order to match the images for faster recognition. Emgu library with c# is used to do the recognition stage. After the finishing the programing process and mount the hardware, the system should detect frontal faces and find the identity of people that have been detected as well as recognize the identity of the detected face.

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE
	ACKNOWLEDGEMENT	IV
	ABSTRACT	V
	LIST OF FIGURES	VIII
	LIST OF TABLES	X
1	INTRODUCTION	1
1.1	Project background	1
1.2	Motivation	2
1.3	Objective	5
1.4	Scope of the project	5
1.5	Problem statement	4
1.6	Thesis Outline	6
2	LITERATURE REVIEW	7
2.1	Introduction	7
2.2	Detection and recognition studies	7
2.3	SURF	9
2.4	Emgu CV	10
2.5	Face Recognition	10
2.6	Face detection methods	14
2.7	Database methods	15
2.8	Summary	17
3	METHODOLOGY	18
3.1	Experimental Objectives	18
3.2	Experimental Equipment	18
3.3	Project flowchart	18

3.4	Introduction	20
3.5	Software	20
3.5.1	OpenCV2.3.1	20
3.5.3	Visual Studio 2010	21
3.5.4	Configuration of openCV with Visual Studio 2010	22
3.6	Hardware	23
3.7	Face detection coding	24
3.8	Face recognition	27
3.9	High graphical user interface	29
4	RESULT AND DISSCUSION	30
4.1	Result	30
4.1.1	Haar training	30
4.1.2	Face detection	31
4.1.2	Design the graphical user interface	33
4.1.3	Face recognition	34
4.2	Discussion	36
4.2.1	Face detection	36
4.2.2	Face recognition	37
5	CONCLUSION AND RECOMMENDATION	42
5.1	CONCLUSION	42
5.2	5.2 RECOMMENDATION	43
	REFERENCES	44
	APPENDEX A	46
	APPENDEX B	48
	APPENDEX C	54

LIST OF FIGURES

NO	TITLE	PAGE
1.1	The application of face recognition	2
1.2	a) Samsung detection feature. b) Improved in the near future	2
1.3	Face recognition with local and global database	3
2.1	Face recognition processes [1]	7
2.2	Recognition steps for Cui Baoxia research [2]	8
2.3	Feature prototype of simple Haar-like and Center- Features [3]	9
2.4	SURF for matching similar points[5]	9
2.5	PCA algorithm	12
2.6	PCA and LDA projection	13
2.7	AT&T database	16
2.8	Yale database	16
3.1	Project flowchart	19
3.2	Emgu cv diagram	21
3.3	The configuration of C++ direction	22
3.4	Additional dependencies of release and debug mode	23
3.5	LOGITECH WEBCAM C210	23
3.6	Steps of SURF working process.	25
3.7	Creating an integral image	26
3.8	Generate features vector	27
3.9	The included libraries used in the program	28
3.10	Recognition methods in the project	28
3.11	Using highgui to enable the user to insert images into the program.	29
4.1	Creating positive haar file	30
4.2	Face detection	31
4.3	Detection of multiple faces	32
4.4	Undetected rotated faces	32
4.5	Undetected side face.	33
4.6	The graphical user interface design	34
4.7	Grayscale cropped image	34

4.8	Matching the similar points	35
4.9	A successful recognized frontal face	35
4.10	Recognition of frontal face with dark background	36
4.11	Main function of face detection code	37
4.12	Cropping the detected face	37
4.13	Recognition using PCA	38
4.14	Recognition with LDA	38
4.15	the result of PCA with Eigenface and LDA with fisherface	39
4.16	Improving PCA performance	40



LIST OF TABLES

Table	Title	Page
2.1	Emgu implementation in C#	9
2.2	Methods of face detection	13
3.1	Camera specification	23
4.1	experiment PCA and LDA	39



CHAPTER 1

INTRODUCTION

1.1 Project background

Face detection and face recognition become important issue in the new technology. These types of technologies have various applications in different fields. Security and identifications are mostly the biggest fields for face recognition and detection.

Face detection is the process of finding faces and being able to differentiate them from the background or other objects. Face detection has been carried out for up to around five decades and it considers the first step for any automatic face recognition system and computer Interact system. A computer does not know what the pictures look like or understand what face mean but it can recognize patterns, matrixes and numbers. Face detection is not straightforward. It has many methods with different algorithms to change the faces in a picture to data to be implemented in computers [1].

For example, the template-matching methods are used for face localization and detection by computing the correlation of an input image to a standard face pattern. The feature invariant approaches are used for feature detection of eyes, mouth, ears, nose, and so on. The appearance-based methods are used for face detection with Eigen face neural network and information theoretical approach [2].

Face recognition is a computer application which is used to recognize and confirm a person identity by capturing of image or video, and compare it with facial database of known faces to find a match [3].

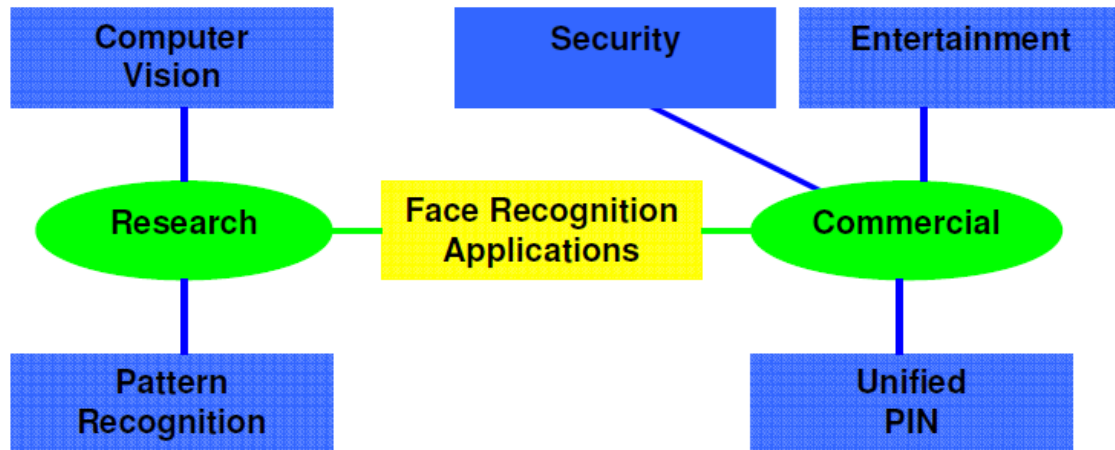


Figure 1.1 The application of face recognition

1.2 Motivation

Nowadays, face detection has many applications in most of the new devices in the market. It is used in digital cameras, smart phones and many other devices. A lot of researchers, engineers and technologists have enriched this field with hundreds of inventions from 1960s until now. Smart phones like Samsung brand as Figure 1.2.a show have face detection feature when the camera is open and display people detected faces. In the future if this Samsung application adds a database for people names it will be able to detect and recognize people identities as shown in Figure 1.2.b.



Figure 1.2 : A) Samsung application face detection

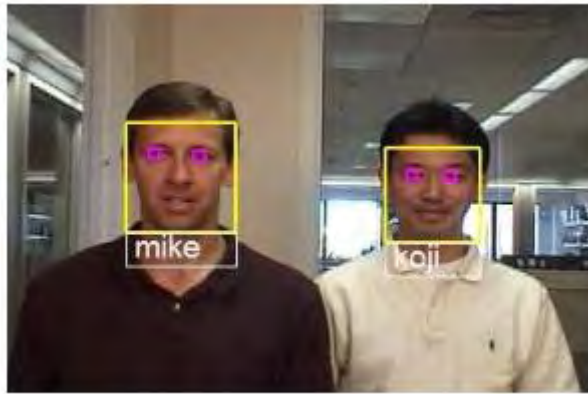


Figure 1.3 : b) Improved application in the near future

Recently, crimes have increased slightly in all over the world especially in Malaysia due to the multiracial and the verities of foreigners. Because of lack of security precautions there is a lot of robbery accidents reported every day. All the security cameras in street work for the purpose of recording. These records could be used for further investigation to recognize the people on the records. The database has to contain information such as names and pictures for all people in the country including the foreigners so that when security surveillance cameras detect a face of person, it can search for the related information in the database.



Figure 1.4: Face recognition with local and global database

This problem can be solved by using face recognition system for fast and accurate identification. Face recognition is applied in other applications such as smart doors computer enrollment and so no.

Face recognition can be used even in college or schools where a system can be designed to take attendance in classes. Calling the student names and checking their attendance take time that's why lecturers tend to pass the attendance list to student to sign as prove that they are in the class. But the problem is that some students sign for others even if they are not in the class. So as a solution for this, a face recognition system can be developed to eliminate such phenomena. With using the right algorithm and high resolution camera, the system have to achieve high rate of recognition.

Each year there is new formula, new algorithm and new programing language. In the future people might not need id identification or passports to travel. By linking a global database with face detection and recognition programs a great result will be gained.

1.3 Problem statement

Faces are a good biometric because of the facial differences between peoples. There are many parameters effect the rate of recognition such as the resolution, the position of the camera and the distance between a person and the camera. However, there are many challenges that trouble researchers in this field. A lot of effective factors can limit face detection and recognition. These factors affect the appearance of face such as enlightenment, face poses, occlusion (sunglasses, hairstyle, make up), and face expressions and camera quality.

To be able to recognize the identity, the programmer must consider the fact that people take photos or videos with different orientation, poses and facial expressions (smiles, sad and surprised faces as changeable facial expression). The brightness of the back ground changes the recognition and the detection of faces and might display wrong detection results.

In order to solve these problems, the method of haar cascade classifier is used in this project for detection. The detection accuracy increase with the increase of the database pictures. The problems caused by the facial expressions, the poses and the brightness effects can be eliminated by using pictures contain several pictures for each person with different facial expressions, poses and brightness.

1.4 Objective of Project

The objectives of this project are as followed:

- To develop an algorithm for frontal face detection.
- To develop algorithm face recognition
- To analyze the effect of light changes on face recognition.

1.5 Scope of Project

This project involves the implementation of openCV and its configuration with c++. It covers face detection method which is developing an algorithm that able to use the provided data set as in order to detect faces. The database used is a collection of images from internet (200 positive images containing faces and 300 negative images do not have faces). The project covers two sets of database, dataset for detection from the internet and dataset for face recognition.

The preprocessing involves image cropping for the detected faces in the video. The taken images have to go through preprocessing procedures where the pictures normalized and scaled then remove noise from the pictures. The project also covers face recognition which is the way taking the detected face and compare it with the pictures in the database to find match for it.

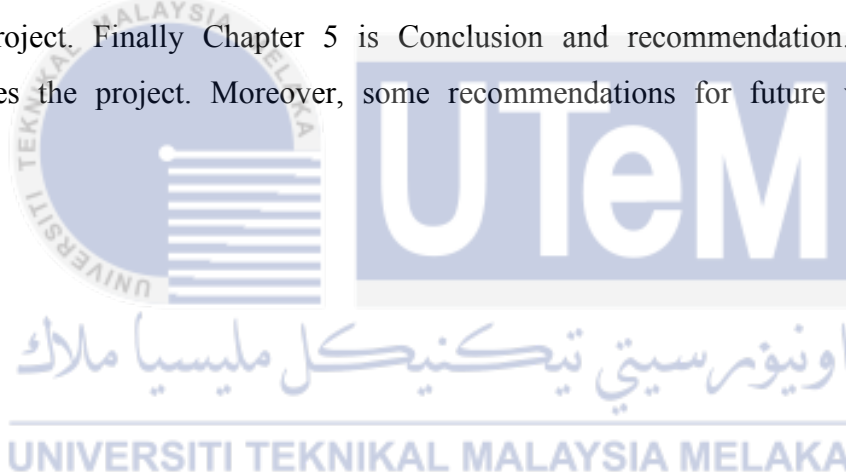
The recognition stage carried out using c# and Emgu library. The project will not cover the detection of side faces or a rotated faces as well as the use of other programing languages such as c#, Python, matlab.

1.6 Project Report Outline

Thesis outline have five chapters. Chapter 1 is an introductory for the project and explains the purpose of this project, the scope and the problem statement. In chapter 2 Literature review is discussed. This chapter provides a detailed background of what others have done in face detection and recognition field. It generally covers their work and compares their different methods and studies.

Chapter 3 shows the methodology that has been followed in the project and describes the procedures for the design in details as well as a flow chart, provide specifications and discuss the data and software implementation.

Chapter 4 is the result and discussion. This chapter illustrates and analysis the result of this project. Finally Chapter 5 is Conclusion and recommendation. This chapter summarizes the project. Moreover, some recommendations for future works will be provided.



CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter summarizes a review about face detection and recognition previous studies. The methods that many researchers have done in face detection and recognition, the information they collect, the advantages and disadvantages of each one of them.

2.2 Detection and recognition studies

The authors present a real time face recognition comprising of face detection and recognition [1]. The main purpose of their research is to make the system easy and fast focusing on exploring the use of image preprocessing as a preparatory step to decrease error rates. The researchers use AdaBoost algorithm to detect faces on video or picture. Figure 2.1 illustrates the steps followed to complete the recognition process.

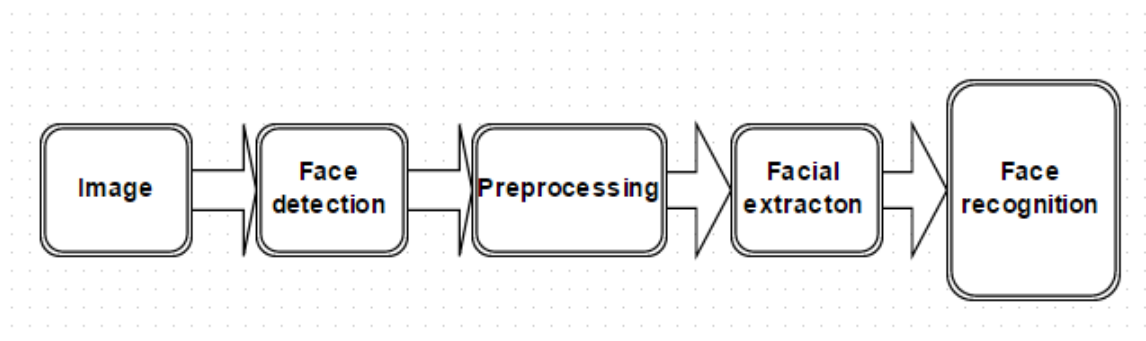


Figure 2.1: Face recognition processes [1]

Preprocess of detected face has five steps. These steps are VIZ, RGB Image to gray scale image, gamma correction, difference of Gaussian, masking and equalization of

normalization. The darkness is removed from the picture after processing the five steps. The result from the pre-process is in gray color.

This pre-process is important to remove noise and illumination effect in the image. The image is fed after that to histogram and fisher liner discriminate (FLD) classifier. The effect of this explains the difference on histogram for the image before and after preprocessing. The result of this research shows an improvement in recognition time process with less complexity.

Another research [2] develops face detection and recognition system based on Davinci technology. The system uses a moving target detection and tracking. The system detects a moving target. If there are faces detected, the faces will be located by AdaBoost algorithm classifier. The facial feature is processed using PCA. The method of Nearest Neighbor is used to match and compare the facial expressions



Figure 2.2: Recognition steps for Cui Baoxia research [2]

Figure 2.2 shows the steps of capturing a moving body, cropping the face and recognizing the face. The result of this system has a good performance in real time. Describing the real-time human face detection and recognition from video sequences in surveillance applications [3] has better result. The modest AdaBoost algorithm is used which can achieve accurate and fast detection.

Moreover, it is used to train the strong classifiers that form the cascaded multi-layer ear detector. The extended Haar-like feature is applied in order to construct the space of the weak classifiers. They used the feature based method system, which operates faster compared to pixel-based method.

Figure 2.3 demonstrates the extended set of Haar-like features prototype including 4 edge features, 2 center-surround features and 8 line features. The black areas indicate the negative weights while the white areas indicate the positive weights.

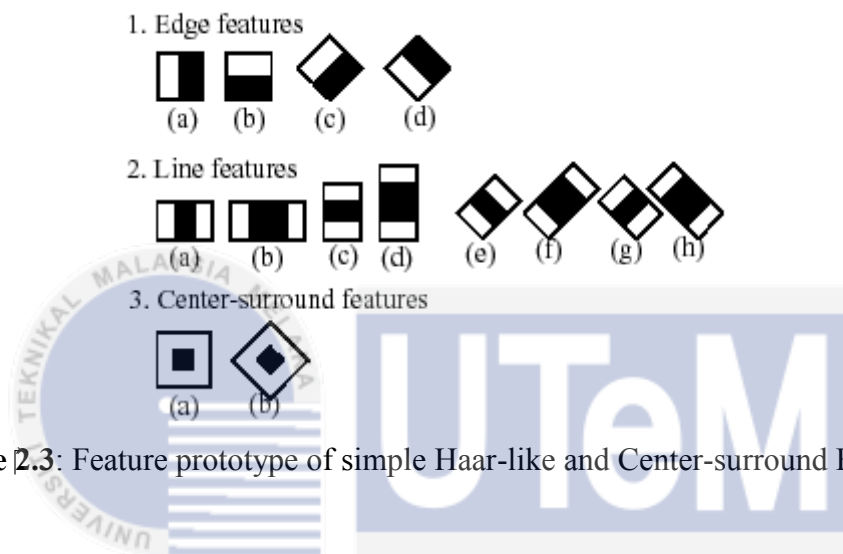


Figure 2.3: Feature prototype of simple Haar-like and Center-surround Features [3]

2.3 SURF

Speeded up robust feature is a fast algorithm for image detection and description presented by Dr. Herbert Bay. SURF can be considered as an updated version of SIFT. The functionality of SURF is parted to three main stages: the interest points, the neighborhood of the interest points, the end descriptor vectors. Surf function mainly is a matching tool to compare or match the similarities between two images [5][6]

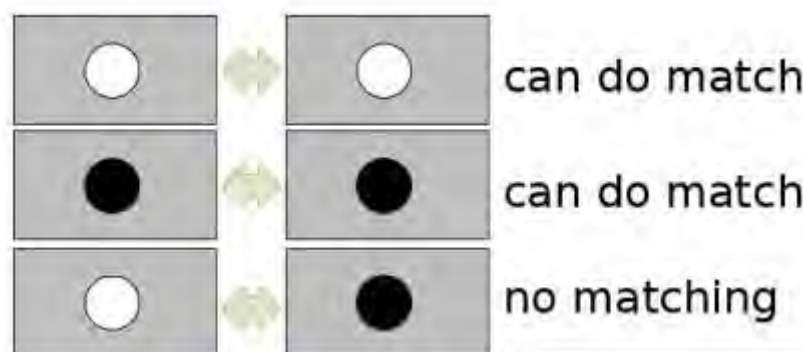


Figure 2.4: SURF for matching similar points[5]

SURF uses blob structure to find the interest points on a picture. The point detector search for the interest points repeatedly by using a method called Hessian-matrix. The fast variant method is applied to the neighborhood of the interest point to localize the interest points. The descriptors vector connects lines between the interest points of two images or between the reference image on the program and the displayed face in video.

2.4 Emgu CV

EMGU CV is an advance image processing cross platform .Net wrapper to open CV images processing libraries. Emgu CV is a new version of open cv which entirely written for C#. The advantage of EMGUcv is that it might be compiled for Mono and hence is capable of running on any platform Mono supports, including android, windows phone, mac and linux.

There are efforts have been expended to have a pure C# application meanwhile the headings have to stay ported, matched with managed C++ application anywhere heading libraries can merely be involved. But then again it is fit worth it always offers the comfort knowing that your code is cross platform.

Table 2.1 : Emgu implementation in C#

Name	Emgu CV (Open Source)	Emgu CV for Windows (Commercial Optimized)	
Development tools	Visual Studio 2010 and up	Visual Studio 2010 and up	Visual Studio 2013 and up
Platform	Windows	Windows	Windows 8.1 Store app *
Supported CPU Architecture	i386, x64	i386, x64	i386, x64
CUDA GPU Processing	✓	✓	X
OpenCL (GPU&CPU)	✓	✓	X

2.5 Face Recognition

Face recognition has two main types. The first type is image based recognition. The second is video based recognition. The image face recognition type depends on classification of the based single image that is been taken from a still camera. On the other

hand, the video based recognition depends on the sequence of the images or the frames extracted from the video which have more data about the face of the person.

There are a few advantages of using still-image face over video based systems. one of the advantages is that the images have higher resolution. As a result, face recognition algorithms can identify people faces more accurate. However, the still image based recognition is beneficial in organized situations where the faces pose and the light illumination are stable. An example of the controlled environment is while taking subjects picture at the airport check in .

The drawbacks of stillimage recognition happen when a organized environments are unattainable. Security camera is a good example of this situation which could be used to recognize people identities in a public place. In this case, video-based recognition produces better outcomes. The clear advantage of video-based face recognition happens in conditions where the photo resolution is slightly low and the video is nonstop. Videobased systems make the most of both spatial and temporal variations in a subjects face.

Face recognition research [7] implemented using PCA method with field-programmable gate array (FPGA). The skin colored method was used to detect faces and PCA algorithm for extracting the feature from a reduced frame picture. The detected face is saved and stored as image vector on the field-programmable gate array memory. The reference image in the database is subtracted and normalized then passed to matching circuit to do the recognition. Euclidean algorithm calculate the distance for the eginspee projection for the pictures, then based on that calculation ,the decision is made for recognizing the face.

Improving the recognition [8] combines the principal components analysis with the speed up rubust feature. In this research a significant results have achieved. Using the speed up rubust feature to extract the features from the face image. The data extracted is then introduced to the principal components analysis to change and project the data extracted to new space points.

2.6 Principle Component Analysis

An algorithm developed by Turk and Pentland which called Principle Component Analysis (PCA) that treats face recognition as a two dimensional recognition issues or problems [9]. The accuracy of this algorithm relies depends on the way that the faces are uniform in stance and enlightenment or illumination

PCA (Principle Component Analysis) can deal with minor varieties in these two variables or factors, however execution is maximized if such varieties are constrained. The algorithm essentially includes projecting a face onto a face space, which catches the most extreme variety among faces in a numerical structure or mathematical form.

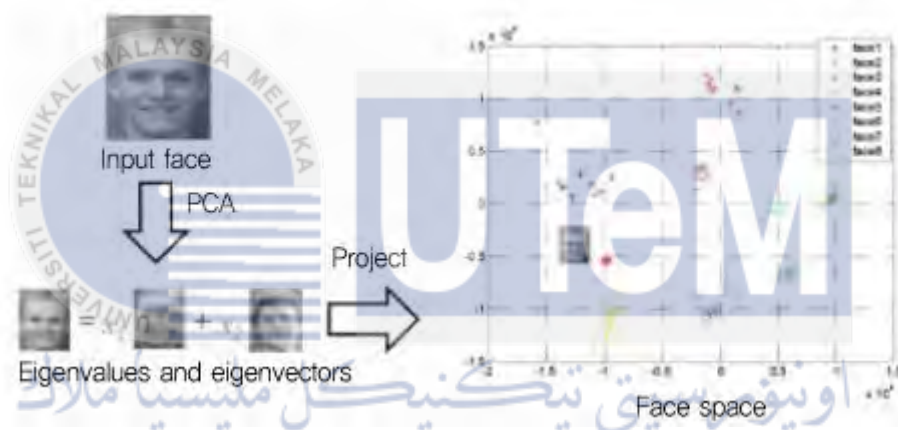


Figure 2.5: PCA algorithm

During the preparation and training stage, every face image is represented as a column vector, with every entry relating to a image pixel. These image vectors are then standardized with respect to normal average face. Next, the algorithm finds the eigenvectors of the covariance matrix of standardized appearances by utilizing a speedup method that lessens the quantity of multiplications to be performed.

This eigenvector matrix will be multiplied by each of the face vectors to acquire their comparing face space projections. In conclusion, the recognition threshold is computing by utilizing the most extreme distance between any two face projections [10]. During the recognition stage, a subject face is standardized regarding the normal or average face and after that projected onto face space utilizing the eigenvector ID or matrix. Next, the Euclidean separation is figured between this projection and every known

projection. The base estimation of these comparisons is chosen and compared with the threshold calculated during the preparation stage. Regarding to this , if the value is greater than the threshold, then the face is new. Else, it is a known face [10].

2.7 Linear Discriminant Analysis

Another common algorithm utilized as a part of face recognition is linear discriminant analysis (LDA). In spite of the fact that, this algorithm was at first developed for data classification, it has been adjusted to face recognition.

As figure 2.6 illustrate that, while PCA concentrates on discovering the most extreme variety inside of a pool of images, LDA recognizes the distinctions inside of an individual and those among people or individuals.

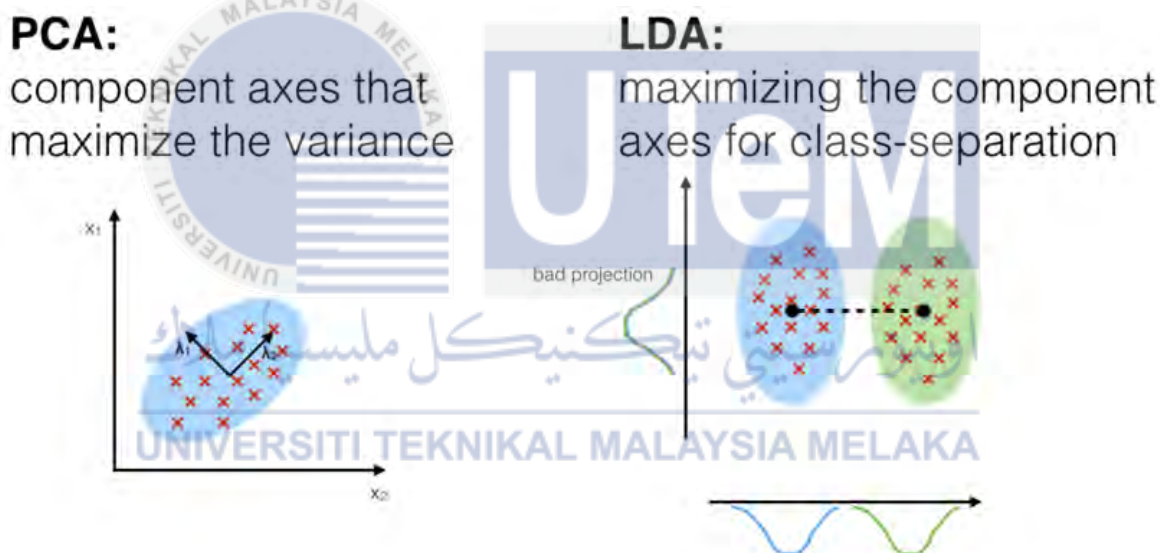


Figure 2.6: PCA and LDA projection

That is, the face space made in LDA gives higher weight to the varieties between people or individuals than those of the same person. Subsequently, LDA is less sensitive to lighting, posture, and expression varieties [11]

The drawback is that this calculation of algorithm is fundamentally more complex than PCA. As an input, LDA takes in an arrangement of appearances for face with multiple images for every person. Normally, these pictures or images are labeled and divided into inside of classes and between-classes. The previous catches varieties or captures

variations inside of the image of the same individual while the recent catches variety among classes of people or individuals.

LDA therefore computes the inside-class scatter matrix and the between-class scatter matrix, characterized by two individual mathematical formulas or scientific equations. Next, the ideal projection is picked such that it "maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the inside of class scatter matrix of the projected samples" [11].

This guarantees the between-class varieties are allocated higher weight than the inside of class varieties. To keep the inside of class scatter matrix from being singular, PCA is normally connected to starting or initial image set.

At last, a surely understood mathematical formula is utilized to decide the class to which the target or object face belong or has a place. Since we have reduced the weight of between class variety, the outcomes or results will be generally relatively insensitive to varieties.

2.8 Face detection methods

Face detection can be implemented in any application by using different methods. Currently, there are four methods are used for face detection

Table 2.2: Methods of face detection

Type	Description	Advantages	Disadvantages
Knowledge based method[12]	Kbm is knowledge based methods. This method extract patterns of the face from the image. It scan images line by line top to down and lift to right	easy to use with simple rules	-Hard to detect faces under uncontrolled background. -Cannot be applied with different poses
Learning base	Learning based methods	- Detect faces even	The results in this

method[13]	are a pre-trained method. Lbm uses statistics and probability distribution function to train the system before doing the detection stage.	with different orientations . - show a good experimental result.	method depend on the training sets. If it does not have enough training then the system shows low rate of detection.
3)Invariant feature [13]	This method extract facial feature such as eyes, nose, mouth.	Invariant to faces pose or expression	Not suitable for real time or changed background. it is also not accurate due to its need to feature combination.
Template matching	This method has pre-defined rules for extracting facial features	A simple method to apply	The scale sizes of the image affect the algorithm. Difficult to find faces in different poses.

2.9 Database methods

- AT&T database: AT&T is an ORL (Otorhinolaryngology) faces database consist of 40 sets of pictures for 40 people. Everyone has 10 photos with different poses and illumination. This method is useful and easy for initial testing. The disadvantage of AT&T is that it need a dark background it be well functioned.



Figure 2.7: AT&T database

- Yale database: These sets of database consist of total of 15 people. The photos are taken with 320×243 pixels of one female and 14 male. Each one of these 15 people should have 11 grayscale photos [8].

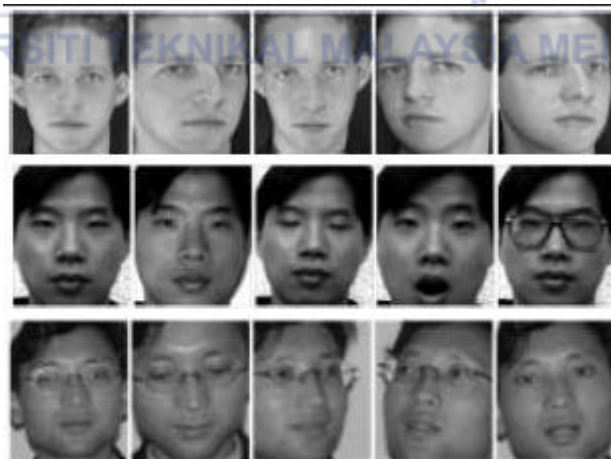


Figure 2.8: Yale database

2.10 Summary

This chapter has discussed some of the related work that has been done in the field of face recognition and detection. [1] uses AdaBoost method to detect faces and histogram to reduce the noise in order to increase recognition rate. [2] used AdaBoost also for face detection and PCA for extracting the facial feature.

The recognition rate is 95%. If the rate is less than 95% then the recognition will fail. [3] combines AdaBoost and haar-like algorithm to detect faces. It uses PC-ICA method to extract the face feature for recognition preprocessing. After long training they reach 95.2% of recognition rate. The disadvantage of this method is that it needs more training to get a good result. Dr. Herbert Bay develops SURF method which achieves faster recognition compared with the other algorithms.

From the information discussed in this chapter, the integration of detection method (which is the haar cascade classifier) with SURF feature is one of the best methods to get fast recognition.

CHAPTER 3

METHODOLOGY

3.1 Experimental Objectives

The experiments objectives of the project are to detect frontal faces on videos and capture the frame where the frontal face is detected. The face detection will be performed by Haarcascade method. After the detection stage is complete, the facial feature will be carried out by SURF to find similar face from the database.

3.2 Experimental Equipment

- 1) Visual studio 2010 full version
- 2) OpenCV2.3.1
- 3) PC\ Laptop
- 4) LOGITECH WEBCAM C210
- 5) Cmake software

3.3 Project flowchart

The flow chart shows the approaches for implementing the project. Starting the process, downloading all the software that will be used in the project. The next step is face detection. Face recognition stage comes after face detection.

If there are no errors in the process, the system check in the database to find match then display either the information of the people identities or display no matches.

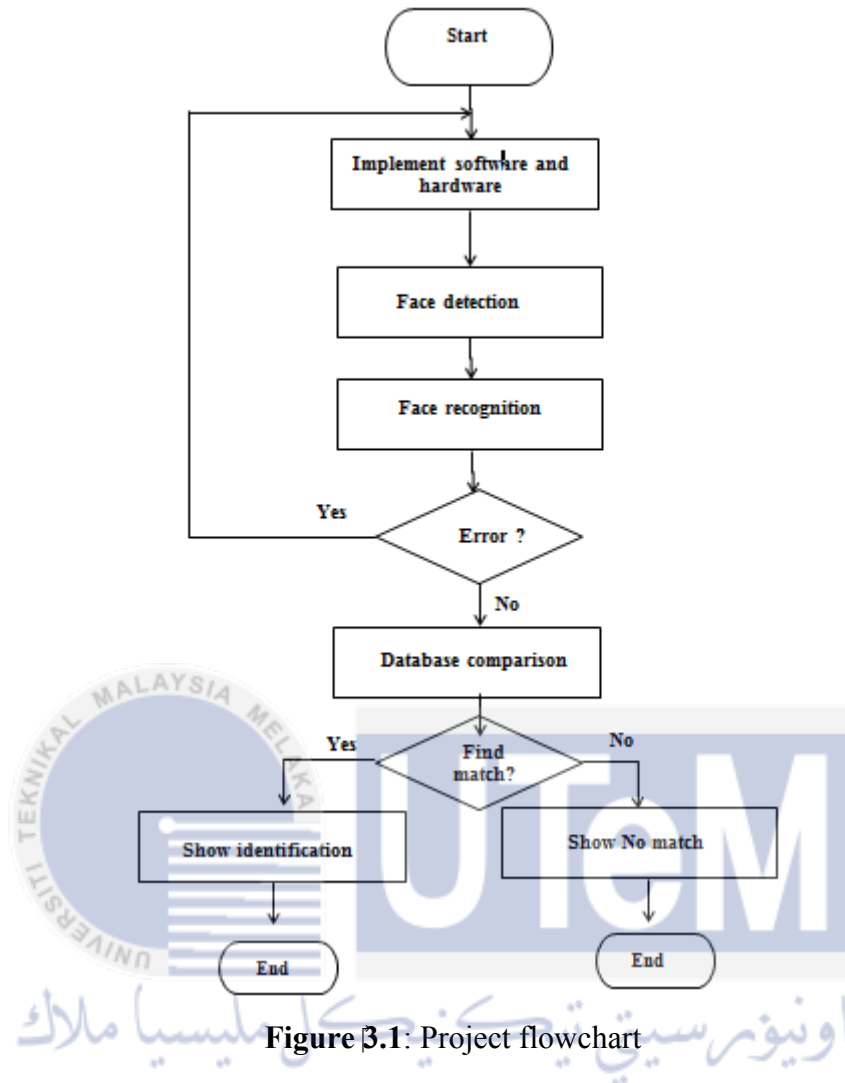


Figure 3.1: Project flowchart

3.4 Description of project methodology

In this chapter, the specifications of the project are discussed in detail. The methodology explains the aspects of the project from the first step of software implementation until the project is completed. The purpose of the project is to detect faces on a picture or a video then recognize the captured face identity.

Mainly, the project can be divided into three parts. The first part is the software and hardware implementation. The second part is coding process. The last part is developing database for recognition phase.

3.5 Software

The software used in this project are programs created by for different purposes. Some of them are free for research reasons such as opencv library and some are not such as visual studio full version

3.5.1 OpenCV2.3.1

OpenCV (Open Source Computer Vision) is a free library of programing functions launched in 1999 and developed in Russia Intel center. Its main purpose is for real time image processing. There are several versions of openCV and for this project openCV 2.3.1 is used. After downloading openCV 2.3.1, the extraction of its files has to be on C drive. The purpose of OpenCV is to provide robust, fast routines for standard picture processing techniques. There are lots of often used techniques in picture processing for computer vision applications. OpenCV provides operation of lots of these functions allowing for fast algorithm fulfillment.

OpenCV proposes to use a chessboard pattern to generate the set of 3D scene points required for calibration. This pattern creates points at the corners of each square, and since this pattern is flat, However can freely assume that the board is located at $Z=0$ with the X and Y axes well aligned with the grid. Calibration process simply consists of showing the chessboard pattern to the camera from different view sides [7].

3.5.2 EMGU CV

EMGU CV is an advance image processing cross platform .Net wrapper to open CV images processing libraries. Allowing opencv V functions to be called from .NET compatible languages such as C#, VB, VC++, and Iron Python and so on. The wrapper can be compiled by Visual Studio, xamarin studio and unity; it can run on Windows, linux, mac, android and windows phone.

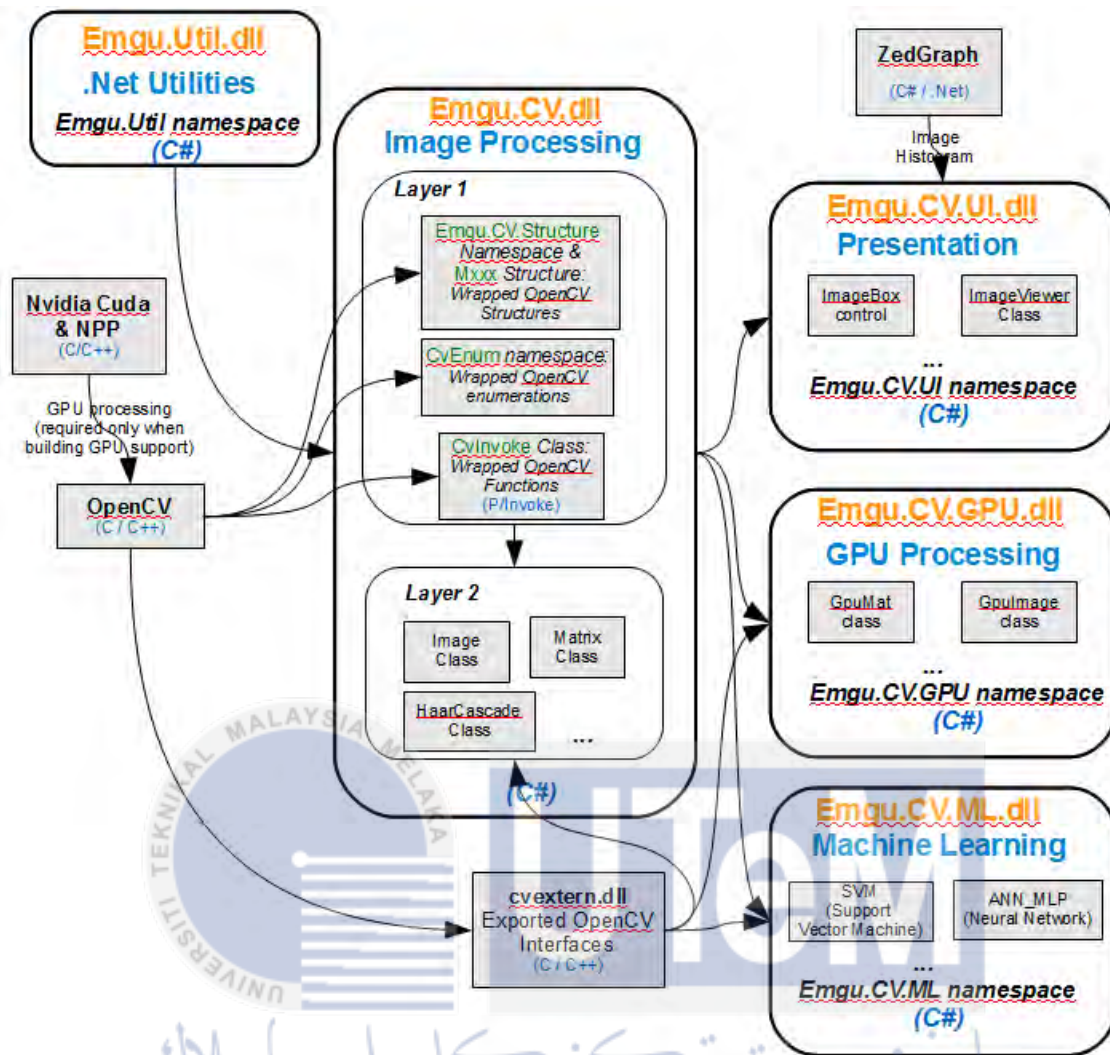


Figure 3.2: Emgu cv diagram

3.5.3 Visual Studio 2010

Visual studio is an integrated tool created by Microsoft for developing computer programs in windows environment. Visual Studio is a complete situated of advancement apparatuses for building Asp.net Web applications, desktop applications, and versatile applications. Visual Basic, Visual C#, and Visual C++ all utilize the same coordinated nature's turf (IDE), which empowers apparatus imparting and facilitates the production of blended dialect arrangements. Visual studio 2010 is better than the other versions because it is easier configure with openCV library. It is also has a new features in C++, better code intelligence, improved multi-focusing on backing, multi-screen help and enhanced code routes.

3.5.4 Configuration of openCV with Visual Studio 2010

The configuration of openCV takes time as well as has to be arranged carefully. Otherwise the program might not run correctly. For this purpose, several steps are considered as followed:

- Downloading openCV 2.3.1 and extract it on a folder inside drive C.
- Cmake software is installed. Cmake is a tool to build files on openCV with easy configuration.
- Setting a new path for the project on C drive.
- Create a new project on C++, Select empty Project, then name it as “hello-world” or any other name, after that we select a location to place project
- After the project is created, the solution proprieties still on the default setting. In order to link C++ with openCV library, the configuration properties is adjusted using the instruction in Appendix A.

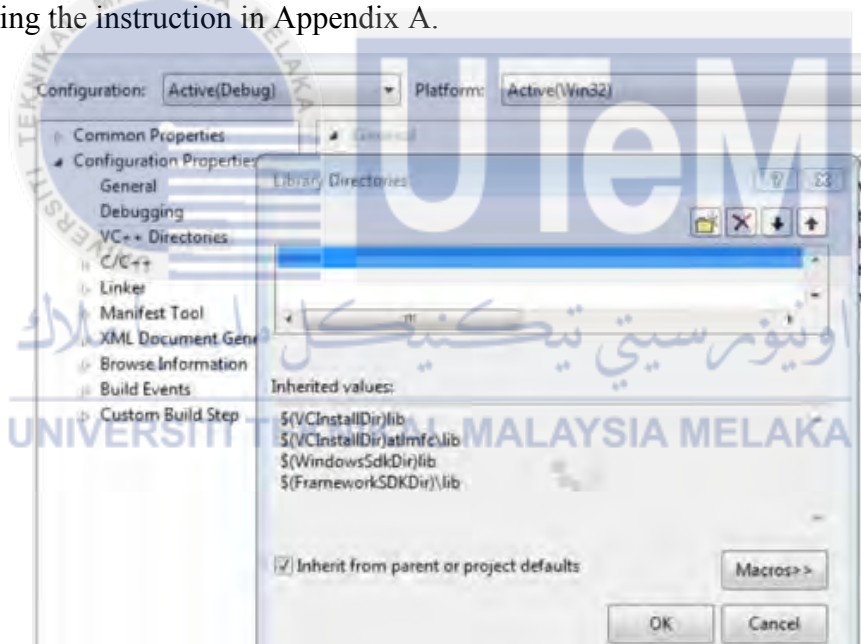


Figure 3.3: The configuration of C++ direction

Under the configuration ‘linker’, there are debug and release new configuration. These new sets of configuration must replace the old configurations in order to successfully be able to run the program coding. By choosing Input “additional dependencies” “edit it and add the release and debug library files on Appendix A.

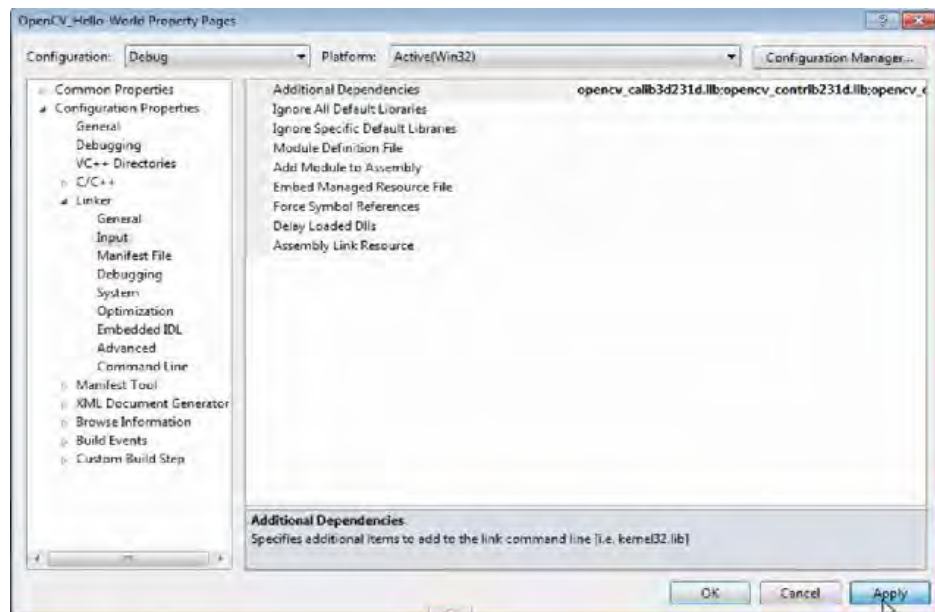


Figure 3.4: Additional dependencies of release and debug mode

After adding the additional dependencies the linkage between visual studio

3.6 Hardware

The hardware for this project simply is a LOGITECH WEBCAM C210 and acer laptop. The Logitech c210 has a good video quality and transmit 30 frames per second.



Figure 3.5: LOGITECH WEBCAM C210

Table 3.1 Camera specification

Product	LOGITECH WEBCAM C210
Usage	Best video choice for Skype, Yahoo Video Messenger and IC
Sensor	High-speed CMOS sensor
Photo capture	Up to 6400*4800 (30.0M, need 2GB RAM)
Video capture	Up to 2048 1536
Video support	640 480 (VGA), 1280 1024 (HD-SVGA), 1600 1200 (HDUXGA), 2048 1536 (HD-WUXGA)
Microphone	Built-in microphone
Interface	USB 2.0
Transmission rate	640 480, 30 F/S
Focus range	3 cm to infinity

3.7 Face detection coding

The program requires instructions to be able to differentiate the faces on a video or picture. For this purpose, a haar cascade algorithm is used to detect the faces in the video. A haar cascade provides a set of training images that have to include inside the program. In this project, only the frontal face needs to be detected. To accomplish that, haarcascade_frontalface.xml must be created.

When creating this classifier we need positive and negative images. The positive images have the frontal face data and the negative images contains the background that should be exist on pictures behind the detected face. After creating frontalface.xml, it has to be added to header file in visual studio. The last step is to create a new project and begin the coding. The face detection coding is attach in appendixB.

To reduce the processed data, the image or the sequence of frames should be converted to grayscale. SURF method is scale invariant feature will be used for feature extraction. After finish this process, the program will identify and display the information of the person that is detected as well as the rate of recognition. Speeded up robust feature finds the interest points in an image by using Hessian matrix. Hessian matrix is 2*2 second order derivative matrix.

$$H(x, \sigma) = \begin{pmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{pmatrix} \quad (3.1)$$

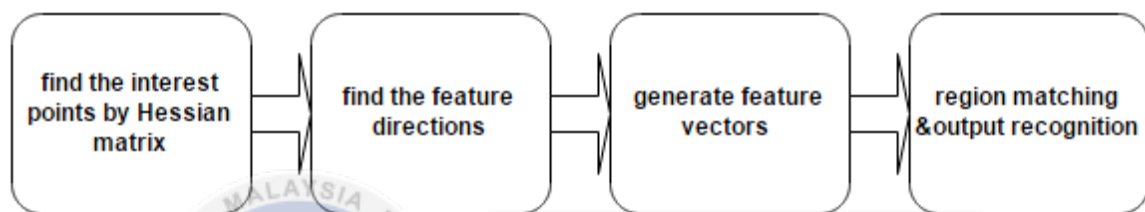


Figure 3.6 a: Steps of SURF working process.

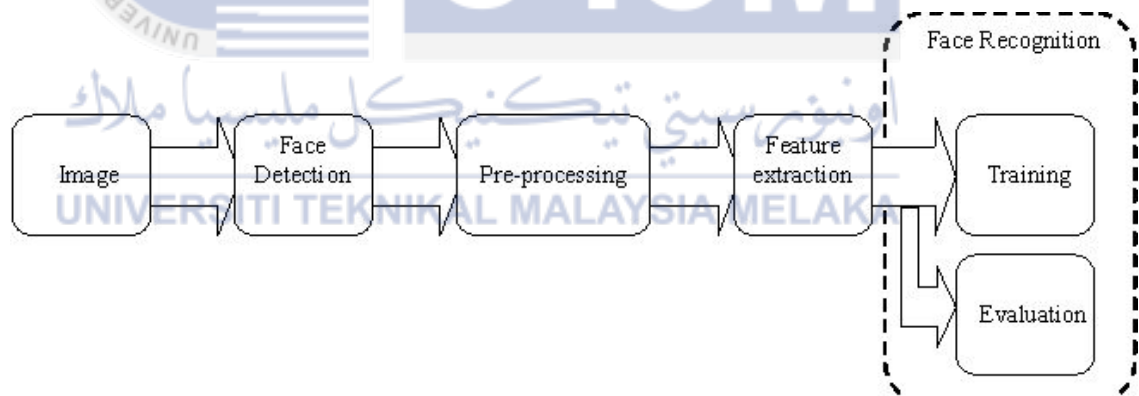


Figure 3.7 b): Steps of the overall process

SURF uses Hessian-matrix to find the interest points in an image. Hessian-matrix is an approximation method to convert the original image to integral image which is a type of box filters. The recognition of faces applied on gradient images. In order to analyse the gradient, we have to know the eigenvalues of Hessian matrix. After forming the gradient from the eigenvalues, a threshold value is set to a specific value in order to control the number of the interest points. The original image and the created integral image have the same size.



Figure 3.8: Creating an integral image

Summing the intensity values in each box of the original image create an integral image. The operation of creating integral image begin from the top left of the original image to the bottom right of it. The feature direction is found by using Haar transform which evaluate the primary direction of the features. The next step is to generate vector feature as shown in Figure 3.7

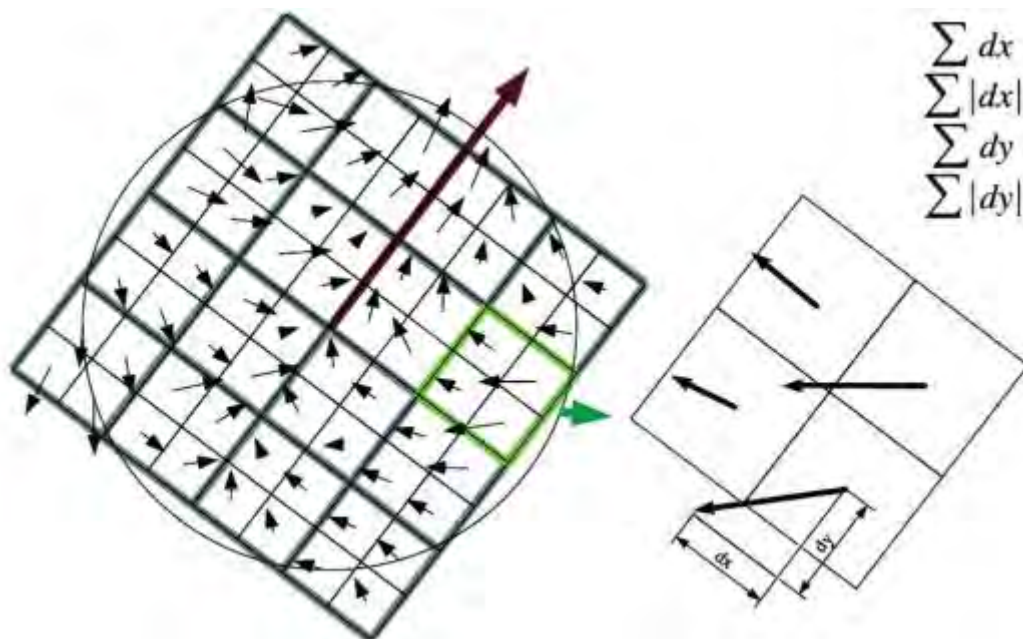


Figure 3.9: Generate features vector

There 16 boxes and each one contain four values of vectors. These 4 values are $\sum dx$, $\sum dy$, $\sum abs(dx)$ and $\sum abs(dy)$. After this step, SURF is finished its process and its ready for matching the images.

3.8 Face recognition

The objective of face recognition is done by using Emgu library with visual studio 2010 c#. which is easier than open cv and c++. The following steps have been done to achieve the face recognition.

Before the code is written, there are some libraries must be included in the beginning in order for the program to define the terms used in the coding. Figure 3.8 show these libraries.

```

using Emgu.CV.UI;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using System.IO;
using System.Xml;

```

Figure 3.10: The included libraries used in the program

After the libraries are included, the Haar cascade classifier is loaded so the program should be able to identify the face of the person from the other objects. Then the training stage begins to complete the recognition. The more data we have in the training stage, the better result of the recognition is gained.

The recognition in the project uses two method. The first method is done using PCA algorithm associated with Eigen face method. The second method is LDA algorithm with fisher face. Figure 3.9 shows both methods in the recognition bar.

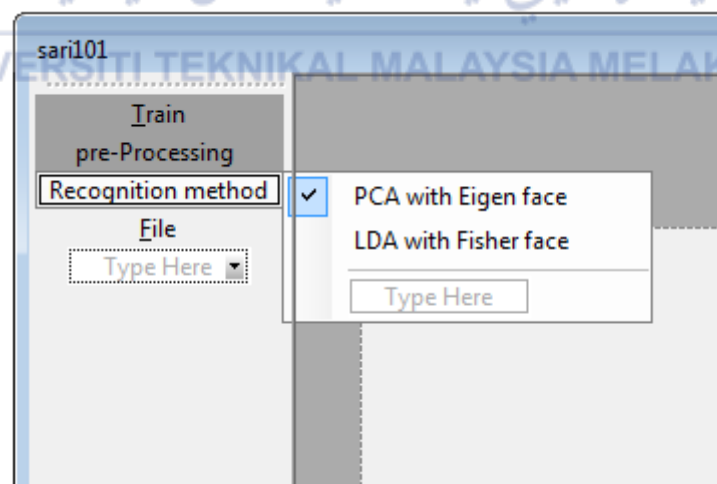


Figure 3.11: Recognition methods in the project

3.9 High graphical user interface

High graphical user interface enable the user to open a window, creates the trackbar , grab a frame and so on. Figure3.10 shows that the user can take number of pictures with different facial expression as training images to be used as database, The program should recognize the detected face as soon as the recognition button is pressed

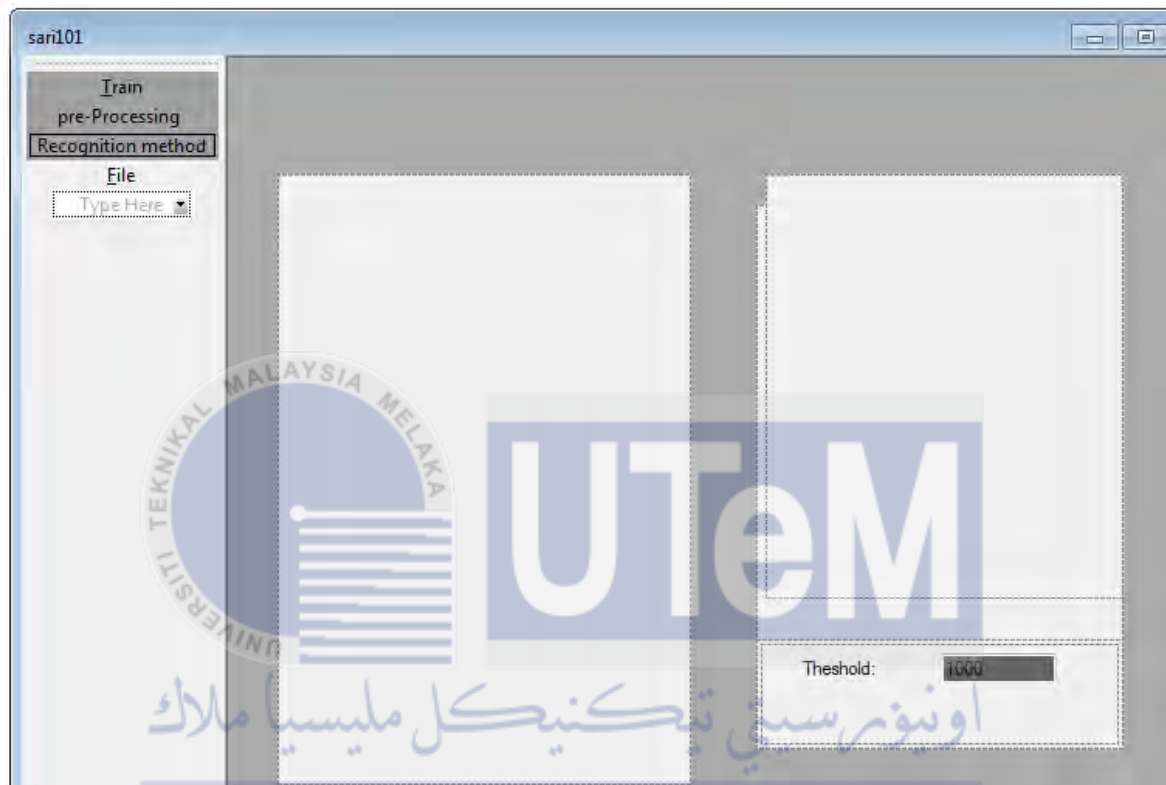


Figure 3.12: Using highgui to enable the user to insert images into the program.

CHAPTER 4

4 RESULT AND DISSCUSION

4.1 Result

The results have been achieved is discussed in real time image capturing from video face detection and recognition. Some of the objectives are achieved and will be shown in this chapter and some will be shown as expected result.

4.1.1 Haar training

The haar cascade file is an important factor for face detection. Creating this file take long time which depend on the amounts of images that will be used to create it. In this project 500 positive and negative images are downloaded from the internet for the purpose of creating the Haar files.

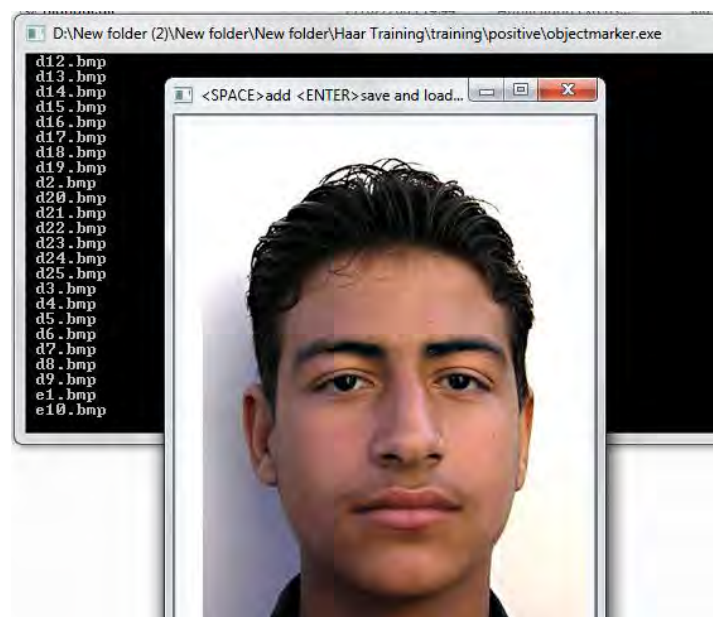


Figure 4.1: Creating positive haar file

In order to create haar file, all the images containing the faces must be cropped one by one. The same process is repeated for the negative images that don't contain faces as well. After xml haar file is created, it can be used in the program for detect faces.

4.1.2 Face detection

The program is used to detect frontal face using Haarcascade classifier which is a type of feature base methods. Figure 4.2 shows that the frontal face has detected and display a rectangle on the face.



Figure 4.2: Face detection

Figure 4.3 shows that multiple faces can be detected successfully. Even a rough sketching of a face is detected. The faces are detected with different facial expressions and brightness.

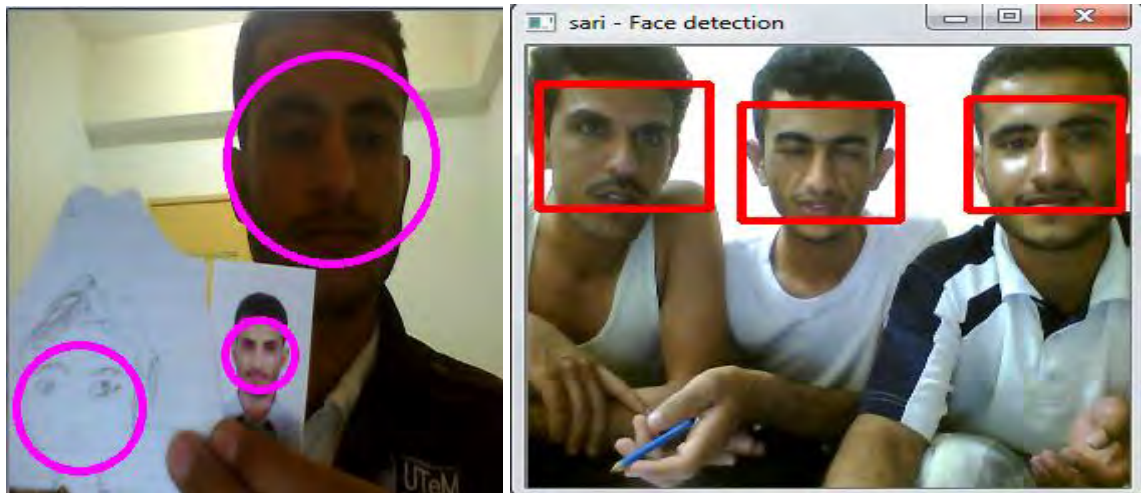


Figure 4.3: Detection of multiple faces

Although the Haar cascade is a very efficient method, it still is not able to detect rotated faces.



Figure 4.4: Undetected rotated faces

As it is objected to detect frontal face only, Figure 4.5 shows that side face is not detected.

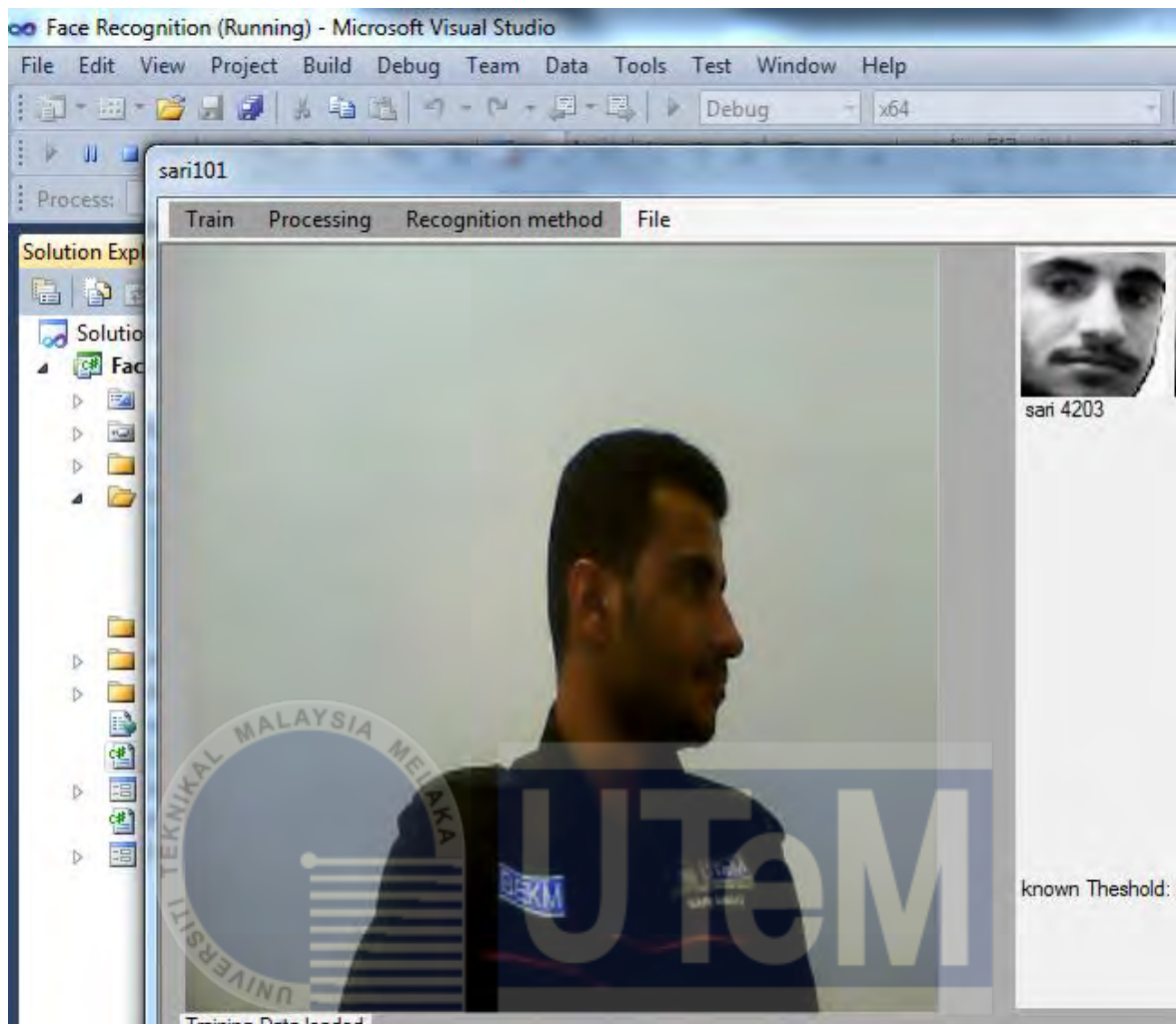


Figure 4.5: Undetected side face.

4.1.2 Design the graphical user interface

The GUI enables the user to interact with the program. Using the windows application in the visual studio the windows can be designed as the user want and attached it with the coding.

Figure 3.6 shows the form that been used in the project. Using the proprieties in the visual studio the form can be designed, adjusted, changed, add a button and a labels according to the use.

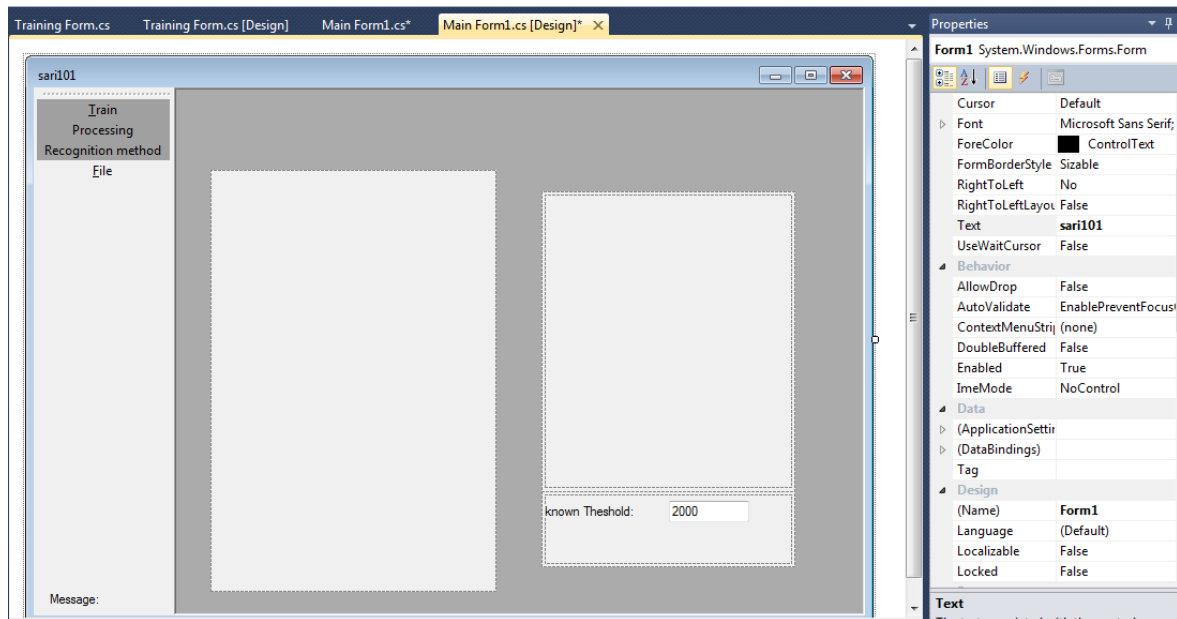


Figure 4.6: The graphical user interface design

4.1.3 Face recognition

In order to perform the recognition, the program crops an image at instantly when it detect the face and then covert it to grayscale to be used later in the recognition.



Figure 4.7: Grayscale cropped image

SURF is used to compare between the cropped images with the faces on video and match the face that is similar to it. The algorithm of SURF is not perfected yet and still under process. It uses only a few interest points for now.

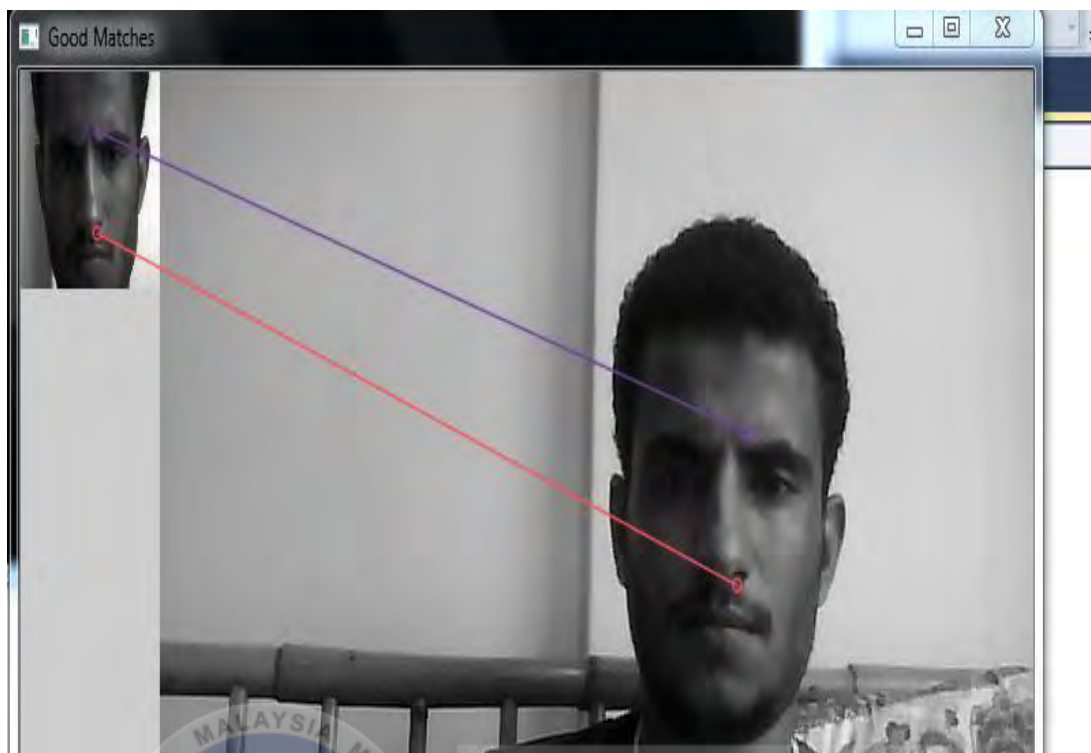


Figure 4.8: Matching the similar points

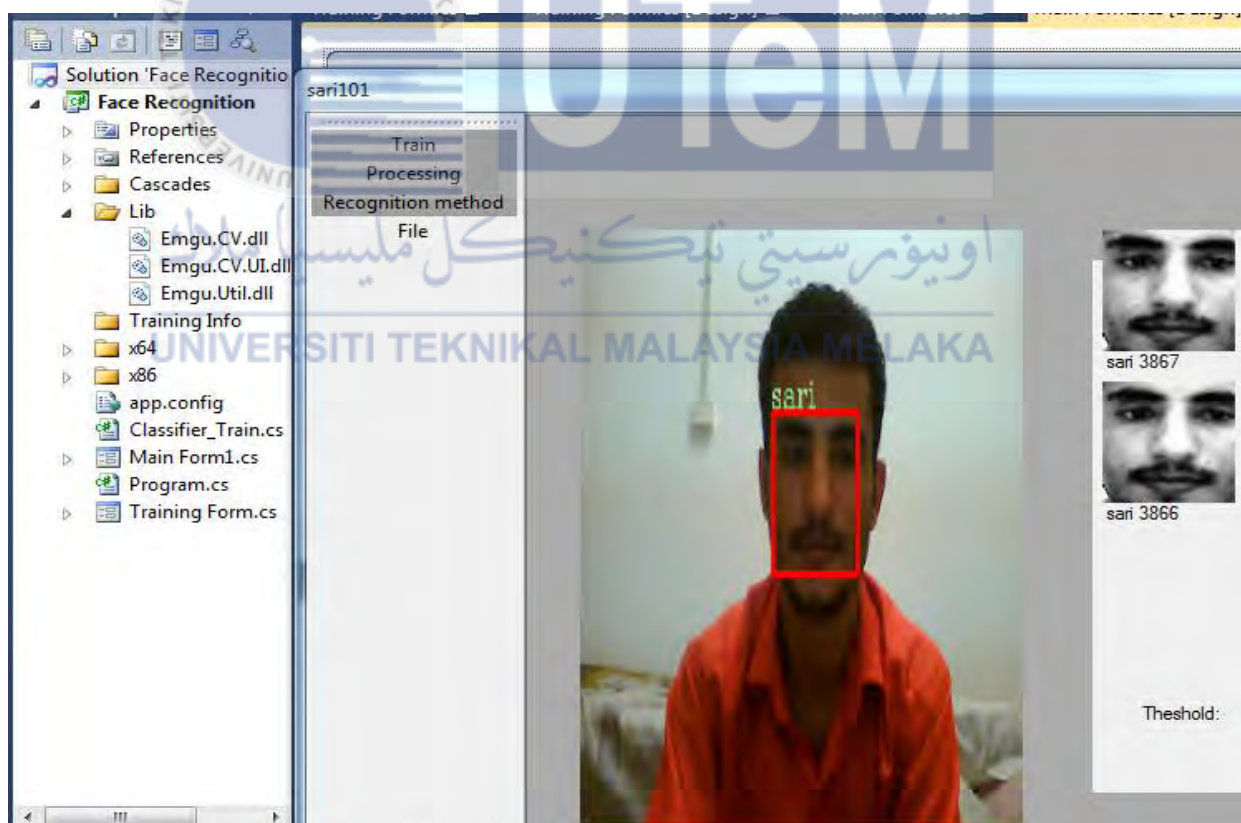


Figure 4.9: A successful recognized frontal face

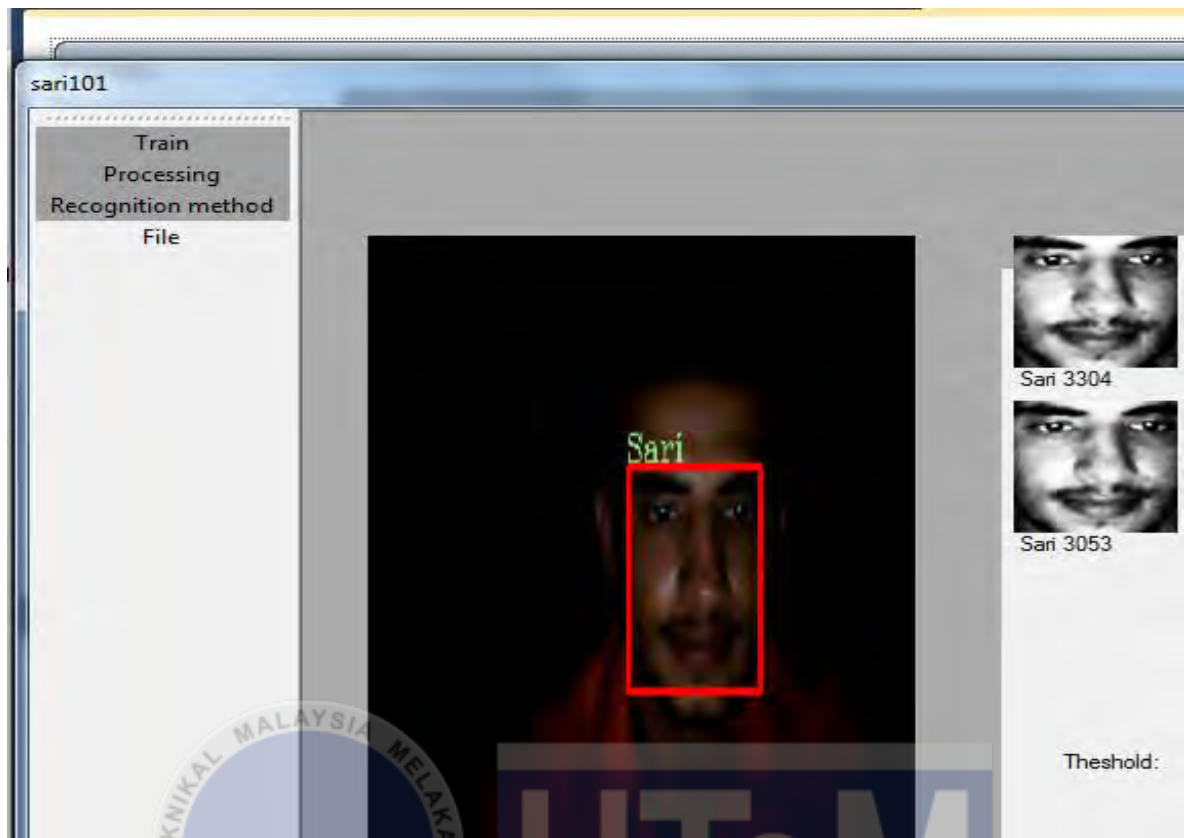


Figure 4.10: Recognition of frontal face with dark background

4.2 Discussion

This part explains the result gained in the project step by step according to the objectives. The first objective is the detection stage.

4.2.1 Face detection

To get a good detection percentage, Haar cascade file should contain as much as possible of images with different poses and skin colors people. Figure 4.8 shows the main function of the coding of face detection.

VideoCapture command is used to open video stream window named sari. The data is processed inside the program as matrixes (Mat frame, targetFrame). Face_cascade_load command loads the Haar cascade file into program.

The command `sari.open (-1)` is to link the external camera with created video window. `face_cascade.detectMultiScale (...)` is to detect faces with different scales. To draw a circle on the face we use `ellipse ( )` and `rectangle ( )` to draw a rectangular shape. Looping is needed in order to draw more than one circle if there multi-faces on the video.

```
int main( void ){
    VideoCapture sari;
    Mat frame,targetFrame;
    face_cascade.load( face_cascade_name );
    sari.open( -1 );
    while ( sari.read(frame) ) {
        std::vector<Rect> faces;
        Mat frame_gray;
        //-- Detect faces
        face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
            0|CASCADE_SCALE_IMAGE, Size(30, 30) );
        for ( size_t i = 0; i < faces.size(); i++ ) {
            Point center( faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2 );
            ellipse( frame, center, Size( faces[i].width/2, faces[i].height/2 ), 0, 0, 360,
                Scalar( 255, 0, 255 ), 4, 8, 0 );
        }
        imshow( window_name, frame );
        int c = waitKey(10);
        if( (char)c == 27 ) { break; } // escape
    }
    return 0;
}
```

Figure 4.11: Main function of face detection code

4.2.2 Face recognition

For recognition, as shown in figure 4.12, the detected face data is converted from BGR image to grayscale image then it has to be equalized to remove the noise. The image is cropped to minimize the effects of the background in the process to avoid false recognition. SURF is applied to the cropped image to extract the facial features. From the results obtained, the integration of haar classifier is significantly a good method for face recognition.

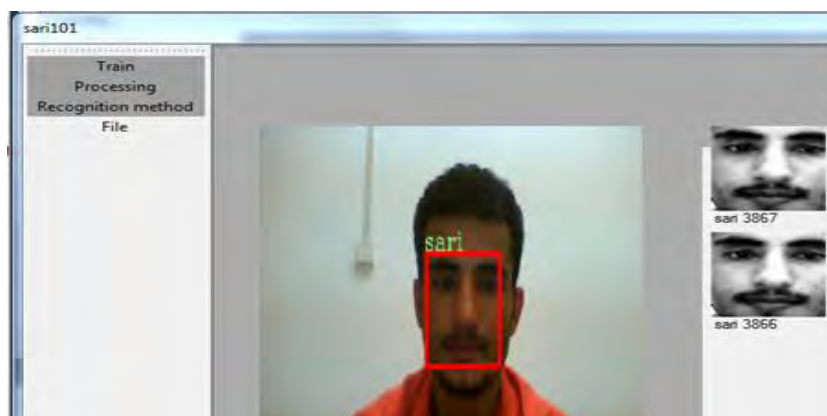


Figure 4.12: Cropping the detected face

The method of PCA shows better result than LDA. When the distance of the detected face varies from 1 meter to three meter PCA can detect the face in the video and display the identity of the person as it is shown in figure 4.13

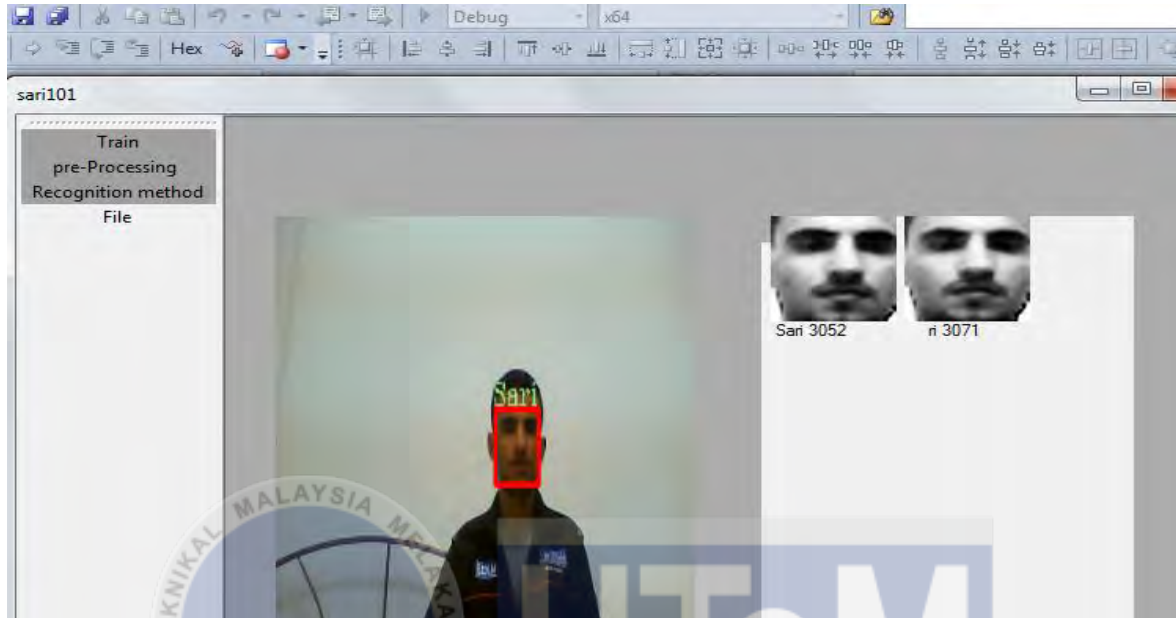


Figure 4.13: Recognition using PCA

Whereas, the LDA method as it is illustrated in figure 4.14 succeed in detecting the face but fail to recognize the identity of the detected face. LDA shows a good result when the distance is less than two meters and succeed to identify the identity of the detected face.

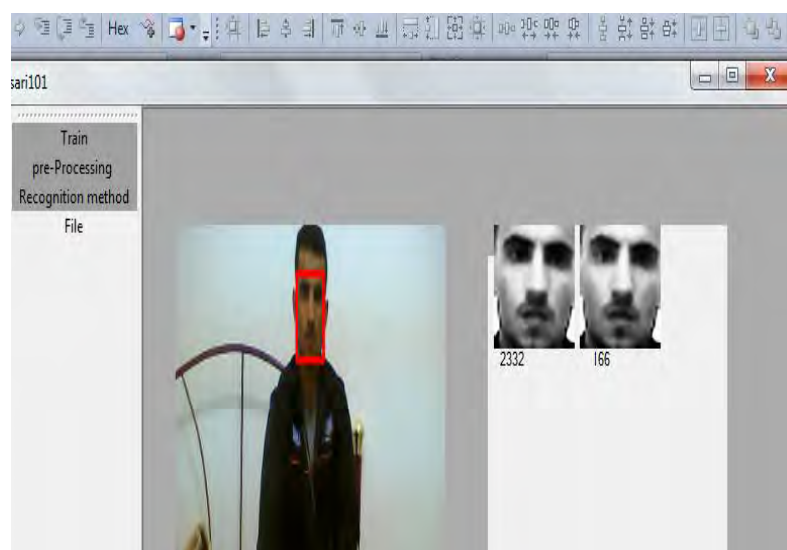


Figure 4.14: Recognition with LDA

The result gained can be illustrated in table 4.1. the table shows that PCA begin with average result recognition when the Eigen images are 2 images only. The LDA gain better result than PCA. But with the increase in the number of the training images the PCA performance improved and show 90% of the recognition. This percentage can be improved more if the camera used has a high resolution. The result is shown using the graph in figure 4.15

Table4.1: experiment PCA and LDA

No	Method	Training pictures	Recognition rate	Error (%)
1	PCA with Eigenface	2	65%	35
2		4	72%	28
3		6	82%	18
4		8	90%	10
5	LDA with fisherface	2	70%	30
6		4	80%	20
7		6	82%	18
8		8	80%	20

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

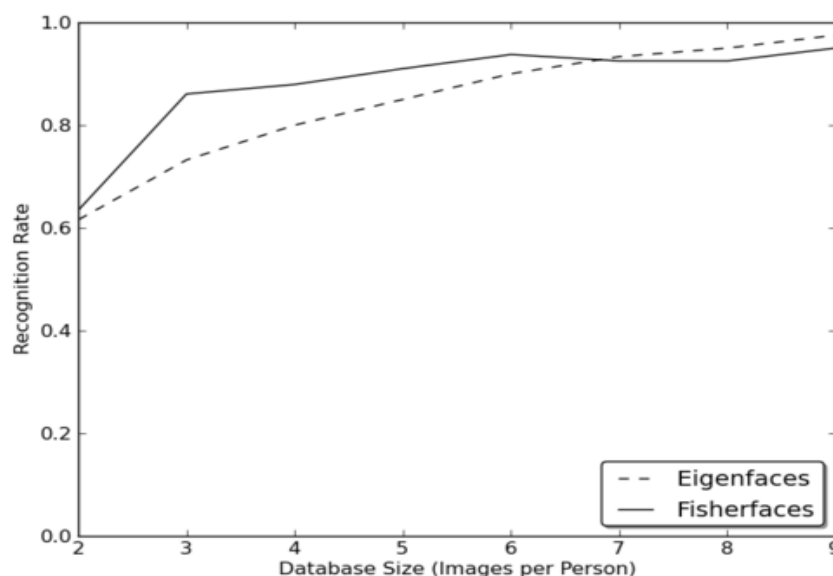


Figure 4.15: The result of PCA with Eigenface and LDA with fisherface

Improving the recognition can be adjusted in the program using many ways. One of these ways is increasing the number of the images in the training data. Figure 4.16 shows the result of improving PCA algorithm performance. The number of the recognized faces in the program can be improved by increasing the number of images of the Eigen faces in the training stage.

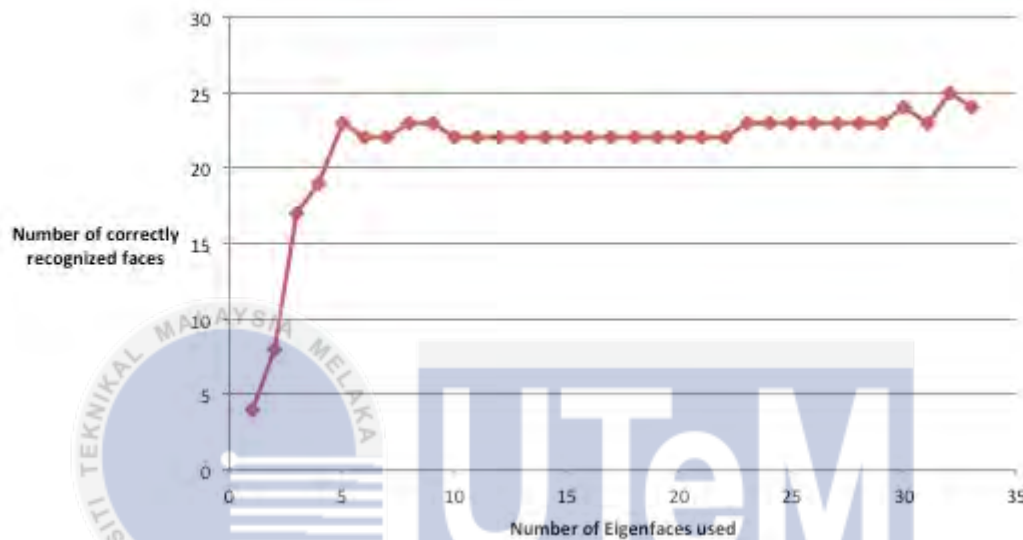


Figure 4.16: Improving PCA performance

The effect of light on the recognition can be significant. It is better to use average lights in front of the lenses of the camera. Using strong bright light on the background make the program makes wrong data which might lead to failed recognition results. Figure 4.16 shows that the program can detect and recognize the face of a person even with dark background as long as there is light on the person face. However, using light on the background will disturb the quality of the pictures that is taken by the camera. Figure 4.18 show that the program fail to identify the identity of the person because of the light.

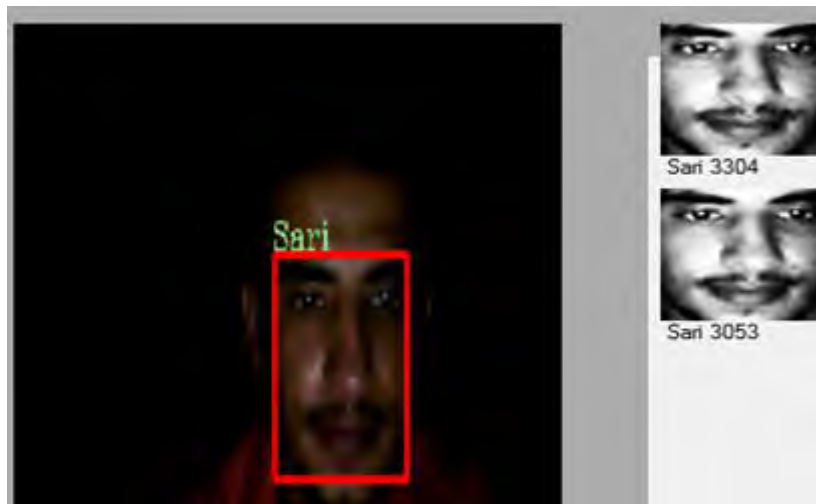


Figure 4.17: The effect of dark background on the recognition



Figure 4.18: The effect of strong bright background on the recognition

CHATER 5

CONCLUSION AND RECOMMENDATION

5.1 CONCLUSION

In short, FYP title of recognition of face detection is explained in detail in this report. Face detection and recognition is challengeable field. Face detection is to identify faces on a video from the other objects. The face recognition is to find the detected face identity. The literature review in chapter 2 discussed some of the related work that already has been done by researchers.

From the previous study, the haar cascade detection method is used in this project with Surf to recognize the identities of the detected faces. Moreover, the recognition methods used in this project are PCA and LDA. PCA method uses Eigenface method for training the program to recognize the identity of the detected face. LDA uses fisherface to train the program in the recognition stage.

Chapter 3 shows the methodology of implementing the project in the real time as well as the software and the hardware that will be used during the process. It also shows how to configure c++ with openCV library. Chapter three also shows the steps to accomplish the recognition and the design of the graphical user interface.

The result and the discussion are explained in chapter4. Each stage of the project is functioned separately and still need to be combined together in single stage. Currently, a collected sample of images from the internet is used in the detection. These samples of images are used to create haar cascade classifier to achieve the detection objective. Specifically, five hundred pictures are used to create the haar classifier for the face

detection. Two methods are used for the recognition database. A small sample of detection database is shown in appendix C. All of the objectives have been successfully achieved.

5.2 5.2 RECOMMENDATION

For future work, it is recommended to use more accurate camera with higher resolution as well as to include calibration to measure the distance for the detected faces from the camera in order to increase the accuracy of the recognition. It is also recommended to expand the recognition database so the program can detect both males and females with wearing Hijab. The higher resolution camera is recommended because it is used to enhance the extraction of features as it is going to provide a great deal of information. Furthermore, the execution time of the program could be noticeably reduced when using a higher speed processor.

Furthermore, different algorithm should be implemented and compared to the existing one to ensure better results to be implemented in a real application. In addition, there are many other problems which need to be addressed in the future. The algorithms developed here can be extended to enable the building of measure performance of the distance detection outdoor using stereo vision system

REFERENCES

- [1] K. Veeramanikandan, R.Ezhilarasi, R. Brindha: International Journal of Emerging Science and Engineering, An FPGA-Based Real-Time Face Detection & Recognition System across Illumination (IJESE)ISSN: 2319–6378, Volume-1, Issue-5, March 2013
- [2] Cui Baoxia, Cui Junjie,Duan Yong; Intelligent security video surveillance system based on DaVinci technology Shenyang University of Technology, Shenyang, Liaoning, 110870, China
- [3] Zhen Lei1, Chao Wang1, Qinghai Wang1, Yanyan Huang2 Real-time Face Detection and Recognition for Video Surveillance Applications*2009
- [4].web.cecs.pdx.edu/~mperkows/CLASS_479/Biometrics/FaceRecognition/IntroFaceDetectRecognition.ppt
- [5]docs.opencv.org/trunk/doc/py_tutorials/py_feature2d/py_surf_intro/py_surf_intro.html#
- [6] Herbert Bay a , Andreas Ess a , Tinne Tuytelaars b , and Luc Van Gool a,b, Speeded-Up Robust Features (SURF)2007
- [7] S. M. R. P. De Silva, H. M. N. Dilum Bandara, P. W. H. Dasun Weerasinghe, Cost Effective Face Recognition Using a Web Cam
- [8] Rabia Jafri, and Hamid R. Arabnia, A Survey of Face Recognition Techniques, June 2009
- [9] W. Zhao, R. Chellapra, P.J. Phillips, A. Rosenfeld, “Face Recognition: A Literature Survey,” ACM Computing Surveys, Vol. 35, No. 4, December 2003, pp. 399-458

- [10] M.A. Turk, A.P. Pentland. "Face Recognition Using Eigenfaces," IEEE Conference on Computer Vision and Pattern Recognition, pp.586--591, 1991.
- [11] P. N. Belhumeur, J. P. Hespanha, D. J. Kriegman, "Eigenfaces vs. Fisherfaces: Recognition using class specific linear projection," IEEE Trans. Pattern Anal. Machine Intell., vol. 19, pp. 711–720, May 1997.
- [12] R. Chellapa, C.L. Wilson and S. Sirohey, "Human and machine recognition of faces: A survey," Proceedings of the IEEE, vol. 83, no. 5, pp. 705-740, May 1995.
- [13] Video Text Detection book , vol. 83, no. , pp. 78-79, May 2014.
- [14] David G. Lowe "Object Recognition from Local Scale-Invariant Features Computer Science Department University of British Columbia"



APPENDIX A

C++ library directions

C:\OpenCV231\include

C:\OpenCV231\include\opencv

C:\OpenCV231\build\x86\vc10\bin

Debug:

Opencv_calib3d231d.lib

Opencv_contrib231d.lib,

Opencv_core231d.lib,

Pencv_features2d231d.lib,

Opencv_flann231d.lib,

Opencv_gpu231d.lib,

Opencv_haartraining_engined.lib,

Opencv_highgui231d.lib,

Opencv_imgproc231d.lib,

Opencv_legacy231d.lib,

Opencv_ml231d.lib,

Opencv_objdetect231d.lib,

Opencv_ts231d.lib,

Opencv_video231d.lib

Release:

Opencv_calib3d231.lib

Opencv_contrib231.lib

Opencv_core231.lib

Opencv_features2d231.lib

Opencv_flann231.lib

Opencv_gpu231.lib

Opencv_haartraining.lib

Opencv_highgui231.lib
Opencv_imgproc231.lib
Opencv_legacy231.lib
Opencv_ml231.lib
Opencv_objdetect231.lib



APPENDIX B

Face detection coding

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <opencv2\objdetect\objdetect.hpp>
#include <opencv2\video\video.hpp>
#include <highgui.h>
#include <opencv2\imgproc\imgproc.hpp>
#include <string.h>
using namespace std;
using namespace cv;

int main( void ){
    VideoCapture sari;
    Mat frame,targetFrame;
    face_cascade.load( face_cascade_name );
    sari.open( -1 );
    while ( sari.read(frame) ) {
        std::vector<Rect> faces;
        Mat frame_gray;
        //-- Detect faces
        face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
        0|CASCADE_SCALE_IMAGE, Size(30, 30) );
        for ( size_t i = 0; i < faces.size(); i++ ) {
            Point center( faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2 );
            ellipse( frame, center, Size( faces[i].width/2, faces[i].height/2 ), 0, 0, 360, Scalar( 255,
            0, 255 ), 4, 8, 0 );
        }
        imshow( window_name, frame );

        int c = waitKey(10);
        if( (char)c == 27 ) { break; } // escape
    }
    return 0;
}
```

Face recognition coding

```
#include "stdafx.h"
#include <iostream>
#include <stdio.h>
#include <opencv2\objdetect\objdetect.hpp>
#include <opencv2\video\video.hpp>
#include <highgui.h>
#include <opencv2\imgproc\imgproc.hpp>
#include <string.h>
using namespace std;
using namespace cv;

void detectfaces( Mat frame );

String face_cascade_name = "haarcascade_frontalface_alt.xml";

CascadeClassifier face_cascade;

String window_name = "sari - Face detection";

void detectfaces( Mat frame )
{
    std::vector<Rect> faces;
    Mat frame_gray;

    cvtColor( frame, frame_gray, COLOR_BGR2GRAY );
    equalizeHist( frame_gray, frame_gray );

    face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
0|CASCADE_SCALE_IMAGE, Size(30, 30) );

    long double ii=0;

    for ( size_t i = 0; i < faces.size(); i++ )
    {
        Point center( faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2 );

        rectangle(frame,Point (faces[i].x,faces[i].y),Point (faces[i].x+faces[i].width,
faces[i].y+faces[i].height),Scalar(0,0,255),4,8,0);

        Mat faceROI = frame_gray( faces[i] );
        std::vector<Rect> eyes;

        ii++;
        imwrite(std::to_string(ii)+".jpg",faceROI);
    }
}
```

```

    imshow( window_name, frame );
}

imshow( window_name, frame );
}

int main( void )
{
    VideoCapture sari;

    Mat frame,targetFrame;

    face_cascade.load( face_cascade_name );

    sari.open( -1 );

    while ( sari.read(frame) ) {
        // to resize the original frame
        cv::resize(frame,targetFrame, cv::Size(300,300));
        detectfaces( targetFrame);
        int c = waitKey(10);
        if( (char)c == 27 ) { break; }
    }
    return 0;
}

```

Code 2

```

#include "opencv2/objdetect.hpp"
#include "opencv2/videoio.hpp"
#include "opencv2/highgui.hpp"
#include "opencv2/imgproc.hpp"

```

```

#include <iostream>

```

```

#include <stdio.h>

```

```

using namespace std;

```

```

using namespace cv;

```

```

/** Function Headers */

```

```

void detectAndDisplay( Mat frame );

```

```

/** Global variables */
String face_cascade_name = "haarcascade_frontalface_alt.xml";
String eyes_cascade_name = "haarcascade_eye_tree_eyeglasses.xml";
CascadeClassifier face_cascade;
CascadeClassifier eyes_cascade;
String window_name = "Capture - Face detection";

/** @function main */
int main( void )
{
    VideoCapture capture;
    Mat frame;

    //-- 1. Load the cascades
    if( !face_cascade.load( face_cascade_name ) ){ printf("--(!)Error loading face cascade\n"); return -1; };
    if( !eyes_cascade.load( eyes_cascade_name ) ){ printf("--(!)Error loading eyes cascade\n"); return -1; };

    //-- 2. Read the video stream
    capture.open( -1 );
    if ( ! capture.isOpened() ) { printf("--(!)Error opening video capture\n"); return -1; }

    while ( capture.read(frame) )
    {
        if( frame.empty() )
        {
            printf(" --(!) No captured frame -- Break!");
            break;
        }

        //-- 3. Apply the classifier to the frame
        detectAndDisplay( frame );
    }
}

```

```

    int c = waitKey(10);
    if( (char)c == 27 ) { break; } // escape
}
return 0;
}

/** @function detectAndDisplay */
void detectAndDisplay( Mat frame )
{
    std::vector<Rect> faces;
    Mat frame_gray;

    cvtColor( frame, frame_gray, COLOR_BGR2GRAY );
    equalizeHist( frame_gray, frame_gray );

    //-- Detect faces
    face_cascade.detectMultiScale( frame_gray, faces, 1.1, 2,
0|CASCADE_SCALE_IMAGE, Size(30, 30) );

    for ( size_t i = 0; i < faces.size(); i++ )
    {
        Point center( faces[i].x + faces[i].width/2, faces[i].y + faces[i].height/2 );
        ellipse( frame, center, Size( faces[i].width/2, faces[i].height/2 ), 0, 0, 360, Scalar( 255,
0, 255 ), 4, 8, 0 );

        Mat faceROI = frame_gray( faces[i] );
        std::vector<Rect> eyes;

        //-- In each face, detect eyes
        eyes_cascade.detectMultiScale( faceROI, eyes, 1.1, 2, 0
|CASCADE_SCALE_IMAGE, Size(30, 30) );

        for ( size_t j = 0; j < eyes.size(); j++ )
        {

```



```

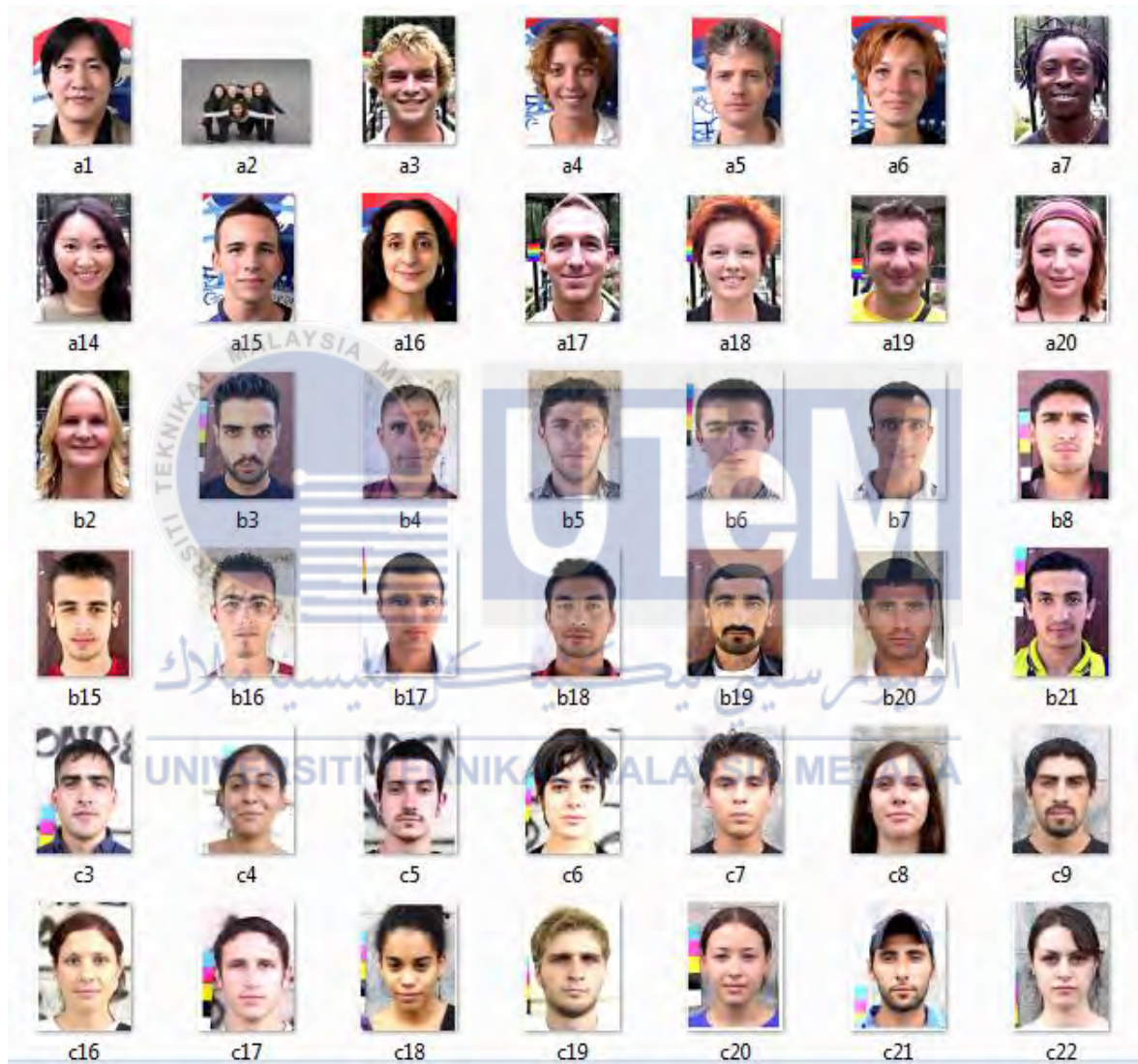
    Point eye_center( faces[i].x + eyes[j].x + eyes[j].width/2, faces[i].y + eyes[j].y +
eyes[j].height/2 );
    int radius = cvRound( (eyes[j].width + eyes[j].height)*0.25 );
    circle( frame, eye_center, radius, Scalar( 255, 0, 0 ), 4, 8, 0 );
}
}
//-- Show what you got
imshow( window_name, frame );
}

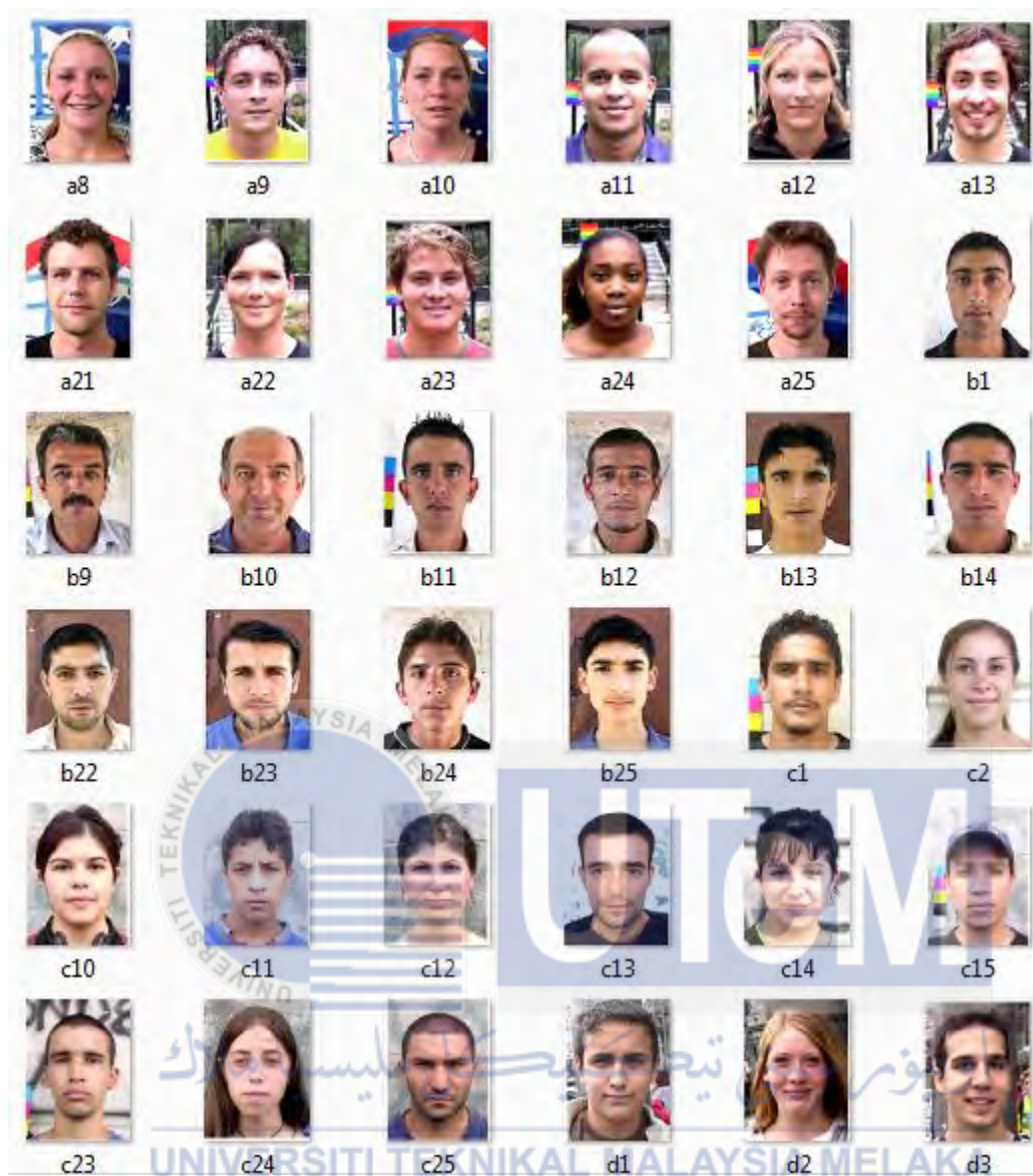
```



APPENDIX C

Positive sample of haar cascade dataset

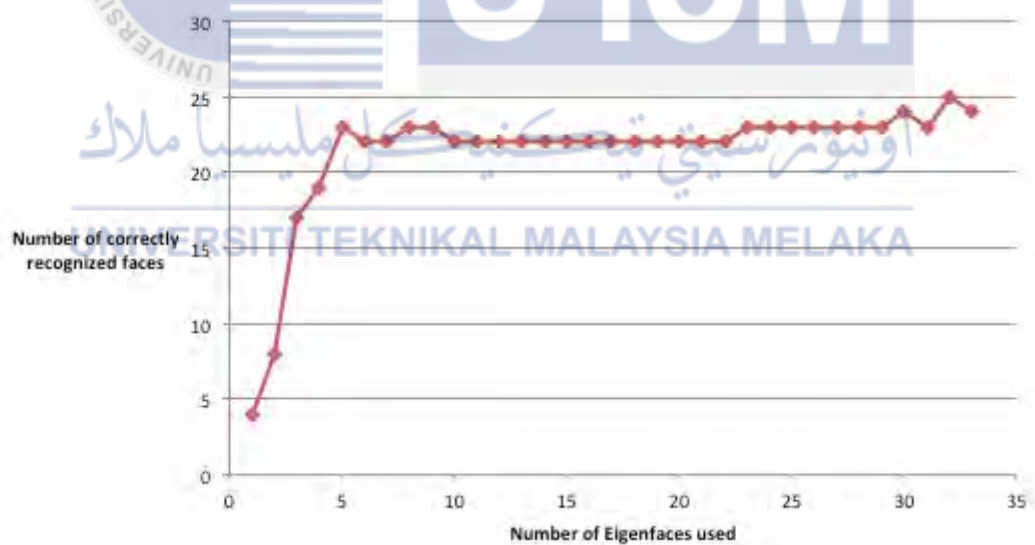
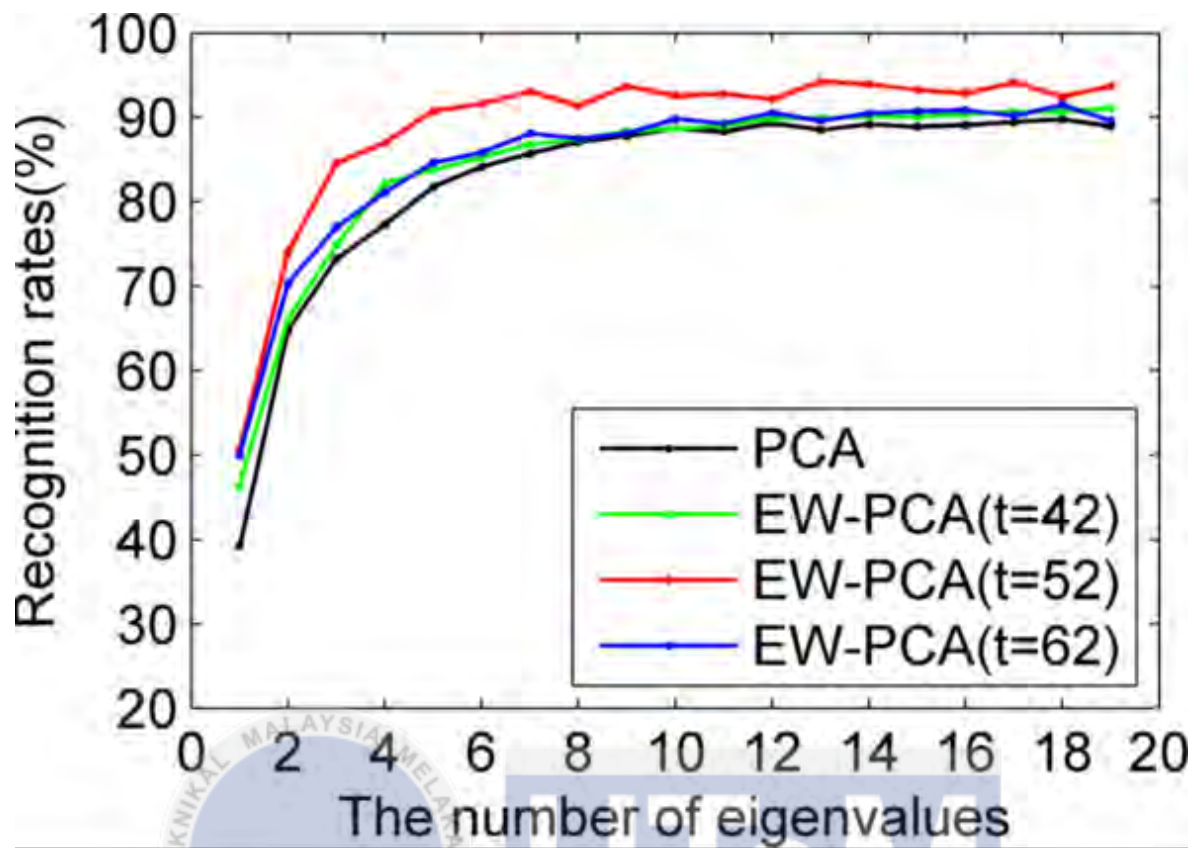


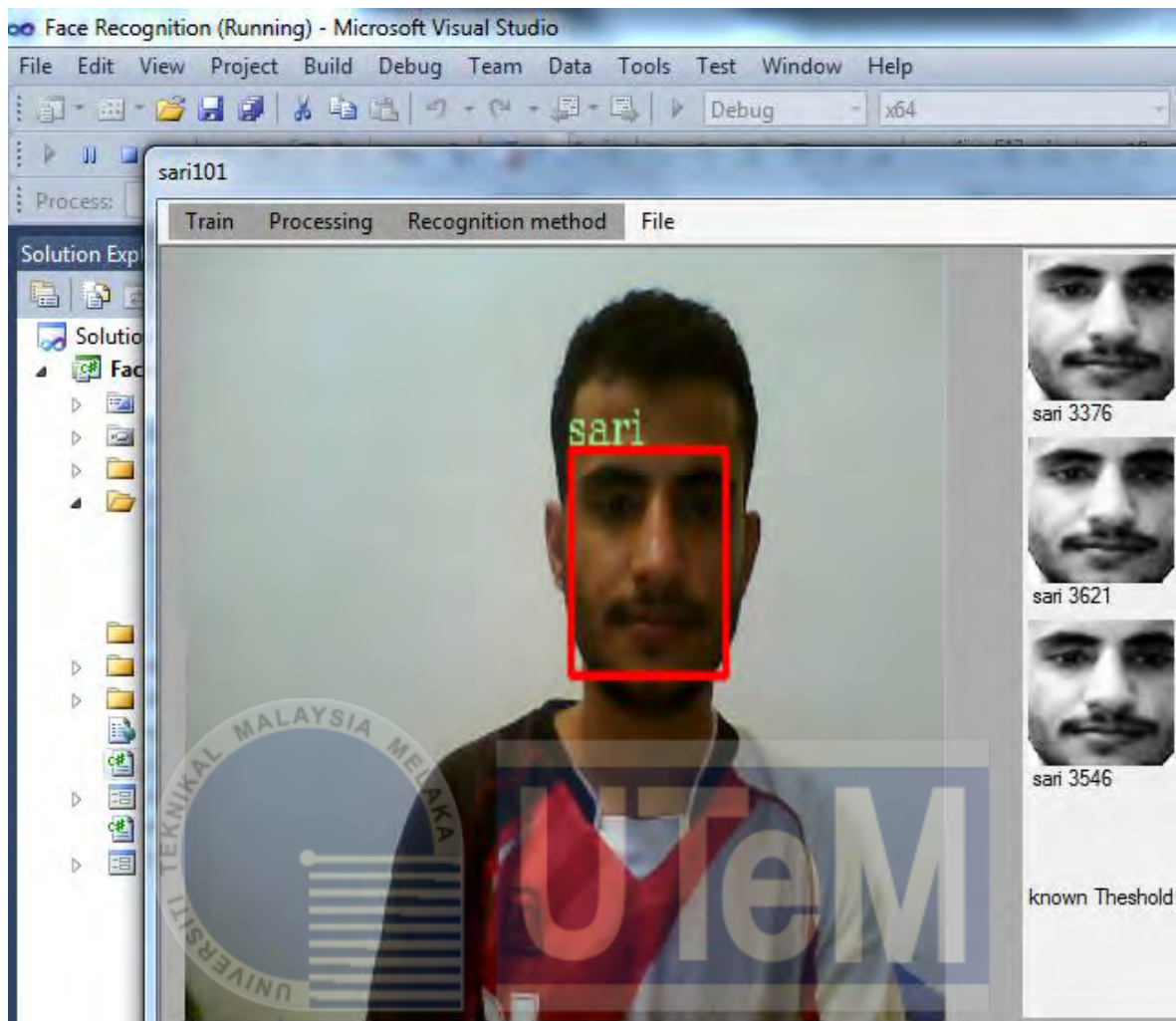


Negative sample of haar cascade dataset

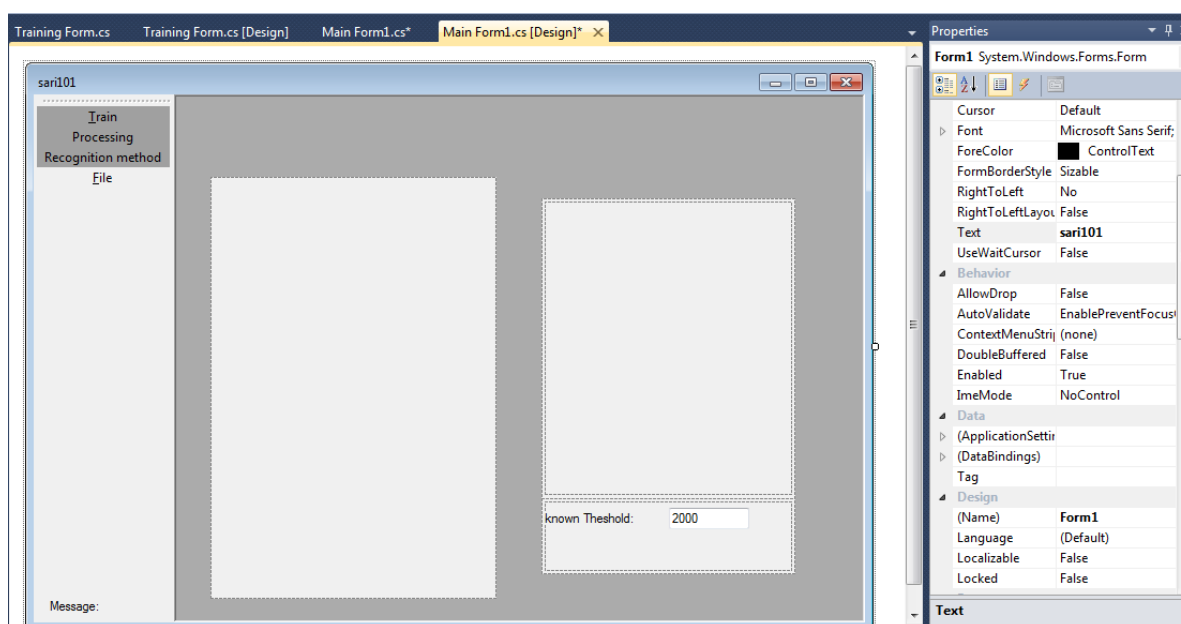


UNIVERSITI TEKNIKAL MALAYSIA MELAKA





اونيورسيتي تيكنيكل مليسيا ملاك
UNIVERSITI TEKNIKAL MALAYSIA MELAKA



Project activities									
		2014				2015			
Task	Sep.	Oct.	Nov.	Dec.	Sep.	Oct.	Nov.	Dec.	Jan
Literature Review									
Algorithm Design									
PSM1 report									
Experimental Setup									
Algorithm Design									
Testing & Troubleshooting									
Thesis Writing									