Face Detection

And

Recognition System

AUTHOR: Mukund Agarwal

SUPERVISOR: Professor Nishan Canagarajah

Project Thesis submitted in support of the Degree of Bachelor of Engineering in Electronic and Communications Engineering

Mukund Agarwal

ABSTRACT

The project aims to develop and release a freeware “motion detection, face detection and face recognition” system with a reasonable performance rate. This system should work on a real time video input feed and should also take appropriate actions when a face is not recognised.

The implementation of these modules upon testing yields performance rate approximately equivalent to those established in their respective research papers.

Although the current implementation takes around 11 sec to process each frame, this can be improved by implementing the MATLAB® code in C, as many other commercial vendors have done.

In the event of an unrecognised face two actions are taken. First is to email that face to a mobile communication device. This email is sent from the PC via internet and is received by the mobile device via 2G or 3G or Wi-Fi connection. The time taken for the notification to be received is almost instantaneous but it does depend on the server load and the inbox refresh/push rate of the mobile device. The second is that the frame is added to a video file which can be later viewed by the user.

The software was released on the MATLAB® Central website.

DECLARATION AND DISCLAIMER

Unless otherwise acknowledged, the content of this thesis is the original work of the author. None of the work in this thesis has been submitted by the author in support of an application for another degree or qualification at this or any other university or institute of learning.

The views in this document are those of the author and do not in any way represent those of the University.

The author confirms that the printed copy and electronic version of this thesis are identical.

Signed:

Dated: 24th April 2009

Mukund Agarwal

ACKNOWLEDGEMENTS

The author would like to thank Professor Nishan Canagarajah for his guidance, support and encouragement throughout the year.

Thanks to Mr Dean and Ms Claire who are the regional managers of Sainsbury in Bristol for their time and advice on the initial commercial and practical aspect study.

Thanks also to ex-Bristol MSc students Jingwan Sun and current PhD student Seyid Mohsin who offered invaluable advice at various stages throughout the process.

Mukund Agarwal

CONTENTS

ABSTRACT ii
DECLARATION AND DISCLAIMER iii
ACKNOWLEDGEMENTS iv
CONTENTS v
1. INTRODUCTION 1
1.1 History, Uses And Aim 1
1.2 Previous Works 2
1.3 Structure Of The Thesis 3
2. SYSTEM DESIGN 4
3. MOTION DETECTION 5
3.1 Theory 5
3.2 Implementation 7
4. FACE DETECTION 9
4.1 Theory 9
4.1.1 Different types of algorithms 9
4.1.2 Comparison of algorithms 10
4.1.3 Voila and Jones method theory 11
4.2 Implementation 12
5. FACE RECOGNITION 13
5.1 Theory 13
5.1.1 Algorithm selection 13
5.1.2 Eigenfaces method 14
5.2 Implementation 15
6. ACTIONS 16
6.1 Email 16
6.2 Record 16
6.3 Radio Security Handset 16
7. GUI IMPLEMENTATIONS 17
7.1 Theory 17
7.2 Implementation 18
8. MODIFICATIONS & IMPROVEMENTS 20

8.1 Averaging RGB 20
8.2 Timer 21
8.3 Ratio Check 21
8.4 Resolution Check 22
8.5 Adding A New Subject To The Database 23
8.6 Average Closest Matches 24
8.7 'Out Of Memory' Tackle 26
8.8 Training Set Size 26
9. RESULTS & ANALYSIS 27
9.1 Hardware & Software Used 27
9.2 Database Used 27
9.3 Motion Detection 28
9.4 Face Detection 28
9.5 Face Recognition 29
9.6 Final System 30
9.6.1 Adding a new subject scenario 30
9.6.2 Recognising a face scenario 31
9.6.3 Unrecognised face scenario 31
9.7 Time 33
10. APPLICATIONS 34
10.1 Surveillance Systems 34
10.2 Shoplifter Alarm 34
10.3 Smart Rooms 34
11. SOFTWARE RELEASE 35
12. CONCLUSIONS 36
13. FURTHER WORK 37
13.1 Port To C 37
13.2 'Out Of Memory' Bug Fix 37
13.3 Improve Face Detection Rate 37
13.4 Improve Face Recognition Rate 38
13.5 Multiple Shots 38
13.6 Interactive Training Module 38
13.7 Sound Recorder Module 39
13.8 Advance GUI Features 39
14. REFERENCES 40

15. APPENDIX 42
15.1 Categories For Test Evaluation 42
15.2 Software Used 42

FIGURES

Figure 1 System Design 4
Figure 2 Establishing background and then calculating the motion 5
Figure 3 Background modelling technique comparisons 5
Figure 4 Difference image and its binary image 7
Figure 5 Difference image showing motion and non motion event 8
Figure 6 Motion Detection Flowchart 8
Figure 7 Plot showing time taken by each method. (Values are taken from their respective papers) 10
Figure 9 Haar wavelets extracting features 11
Figure 10 Cascaded classifier 11
Figure 8 Basic Haar wavelets 11
Figure 11 Face Detection Algorithm 12
Figure 12 FERET FAFB 13
Figure 13 FERET DUP2 13
Figure 14 Specimen's images taken from AT&T database and its resultant eigenface 14
Figure 15 Face Recognition flowchart 15
Figure 17 GUI implemented in Flash – During process 18
Figure 16 GUI implemented in Flash – During initialisation 18
Figure 18 A difference image containing motion and RGB value markers 20
Figure 19 Threshold and the resulting image 20
Figure 20 Expected detection area shape 21
Figure 21 Small false face detections removal 22
Figure 22 Small valid faces 22
Figure 23 Folder showing 10 faces shot files and the text file 23
Figure 24 A Euclidean distance graph of a correctly recognized face accompanied by the directory 24

Figure 25 A Euclidean distance graph of an incorrectly recognized face 25
Figure 26 Success rate Vs Processing Time 26
Figure 27 Sample from AT&T database of faces 27
Figure 28 Screenshots of part of GUI before and after motion 28
Figure 29 Five subjects face detection results 28
Figure 30 Eight face recognition results including two false recognitions 29
Figure 31 System showing the photo strip containing last 10 faces added in the database 30
Figure 32 System correctly recognising a face 31
Figure 33 System incorrectly recognising a face 31
Figure 34 UNKNOWN face email 32
Figure 35 More unrecognised faces 32
Figure 36 Unrecognised faces (False face detection) 32
Figure 37 Bar plot of time taken by each module 33
Figure 38 Screen shot of the software listing webpage 35

TABLES

Table 1 VeriLook 3.2 technical specifications 2
Table 2 Detection rate for various numbers of false positives on the MIT+CMU test set containing 130 images and 507 faces. (Published in [8]) 10
Table 3 Threshold and respective time taken 21
Table 4 Time taken in seconds for each module 33
Table 5 Time taken by different types of code implementation 37

1. INTRODUCTION

1.1 History, Uses And Aim

Motion detection, face detection and recognition are being applied to a variety of applications recently. The main reason for this change is to reduce unnecessary human intervention and increase overall system efficiency.

Face detection has been around for a while in our digital cameras, both digital SLR and compact cameras. When we point the camera to a scene they have the ability to find the faces and set the exposure and the focus automatically. Since we generally want the people in our picture to be the one who are sharp and well exposed, this is a very helpful technology.

Most of the biometrics security systems include facial recognition as one of the key security checks. Some of the surveillance systems in UK also implement this technology for crime prevention and also to stop passport and other various types of frauds.

In order to implement facial recognition sometimes, it's also necessary to include face detection if the developer wants to make the system more intelligent. This is compulsory where the developer doesn't have the option of telling the user where to look and stand. Motion detection is also included if performance and power consumption in a system are a major concern.

There have been developments of several algorithms by various researchers to detect and recognise faces. This process started from 1960 when the first semi automated algorithm was developed. The problem was that it required someone to locate facial features and then measure the distances before the calculations can be done. In 1988, a milestone was reached when principle component analysis was applied for face recognition. PCA allowed successful face recognition by using less than a hundred values. In 1991, reliable real time automated recognition was reached by using the residual error to detect faces when the eigenfaces techniques were being used.

Several vendors have implemented their own interpretations of those research methods. The price charged for each software development kit is normally nowhere in the budget of an average person. Only, big organisations have deep enough pockets to get hold of a license for that software from them.

The project involves the development and testing of open source software which will be based on these research methods. The software will implement the motion detection, face detection and face recognition modules using a live feed. The system will have a reasonable detection and recognition rate. It will also aim to provide a simple to use attractive graphical user interface (GUI). Focus will be also made on the actions taken in the event of unrecognised face.

1.2 Previous Works

Although there are several solutions available, two of them are seemed to be favoured by the majority of the customers. They are:

1. **Face Sensing Engine[1] :** This engine offers face detection, tracking, feature point extraction/tracking. And also subject identification from images and videos. Their customers include Casio and Nikon.

2. **VeriLook 3.2 algorithm[2] :** This algorithm is designed for biometric systems. It boosts the following features: multiple face detection, simultaneous multiple face processing, live face detection, face image quality determination, tolerance to face posture, multiple samples of the same face, identification capability, fast face matching, and compact face features template and a features generalization mode. Their customers include Lenova, and several passport and voter control systems.

VeriLook 3.2 specifications	
Recommended minimal image size	640 x 840 pixels
Multiple Face Detection time	0.07 sec
Single face processing time (after detecting all faces)	0.13 sec
Matching speed	100,000 faces/sec
Size of one record in the database	2.3 Kbytes
Maximum database size	unlimited
HARDWARE:	**PC with 2.8 GHz Intel Pentium 4 CPU**

Table 1 VeriLook 3.2 technical specifications

One of the applications which stand outs by making the face detection and recognition work together is **iPhoto 09**. It provides a very useful and powerful way to organise individual's photo library.

1.3 Structure Of The Thesis

The next section of the thesis (section 2) covers the overall design of the system.

Section 3, 4 and 5 concerns the motion detection module, face detection module and face recognition module respectively. In each of these sections the different approaches are discussed where appropriate. The performance of some of the approaches is also considered. At the end of each section the implementation is explained.

The various tasks performed in the actions module in the event of unrecognised face are discussed in section 6.

The choice of software used for the GUI development is justified in the beginning of section 7. In the second half of this section the implemented GUI and its features are explained.

Before testing all the modifications and improvements made to the implementations are discussed in detail in section 8.

The test setup and the results obtained are part of section 9. The results are also analysed in the same section.

Possible applications of this system are talked about in section 10. Also the software is released to the public in section 11.

Section 12 is the conclusions section where the individual modules and the whole system's relative performance are discussed.

Further work is suggested in section 13 which is then followed by references and the appendix.

Where ever a code reference is made or a function is seen, the use of that can be checked in the code provided in the optical media supplied with this thesis.

There is a glossary available in the appendix. All abbreviations used are expanded here. Software used summary sheet is also included here.

2. SYSTEM DESIGN

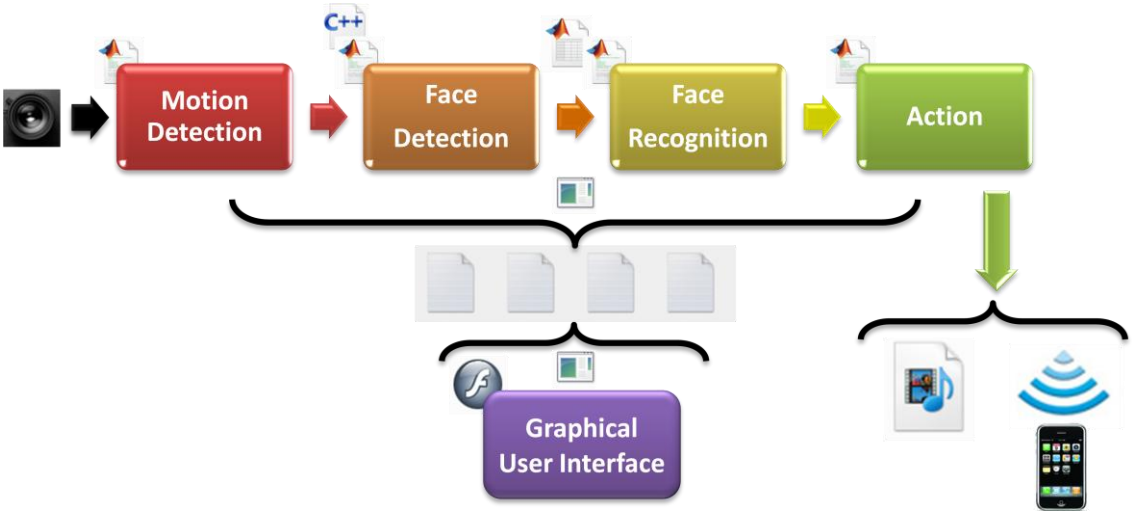

Figure 1 System Design

The video input will be taken by the motion detection module. This module is written in MATLAB®. Upon detection of motion this module then passes the frame to the face detection module.

This module is written in MATLAB® and C++ and searches for faces like features in the frame passed. If faces are found then their locations and the image frame are passed to the face recognition module.

This module is purely coded in MATLAB®. It uses the coordinates to extract the faces from the frame and convert it into similar format as the ones in its database. A 1 : N (face found in the frame compared with all faces available in database) comparison is made. If the face is not found in the database then the face is passed to the action module.

This module is also written in MATLAB®. It stores the frame to a video file. It also emails the face to a user defined address via the internet.

The email is received on a mobile communication device over Wi-Fi or 3G or 2G data network.

Throughout the system, a graphical user interface developed in Flash and written in Action script 3.0 runs. The synchronisation between the system and the GUI is maintained by commands written in text files using pre compiled executables developed in C++.

3. MOTION DETECTION

3.1 Theory

Figure 2 Establishing background and then calculating the motion

Any motion detection algorithm is based on the following principle: At the start of the algorithm frame(s) is taken from the camera feed. This frame or a series of frames are used to establish the background model. There are two types of background modelling:

1. **Adaptive** background model is where a series of frames are used and an average is calculated from all of them over a period of time. This is useful if the objects move continuously in a scene.
2. **Non-adaptive** background model is where a frame is taken and saved as the background. Stauffer[3] describes in his paper that this approach is only useful in very static, indoor environments.

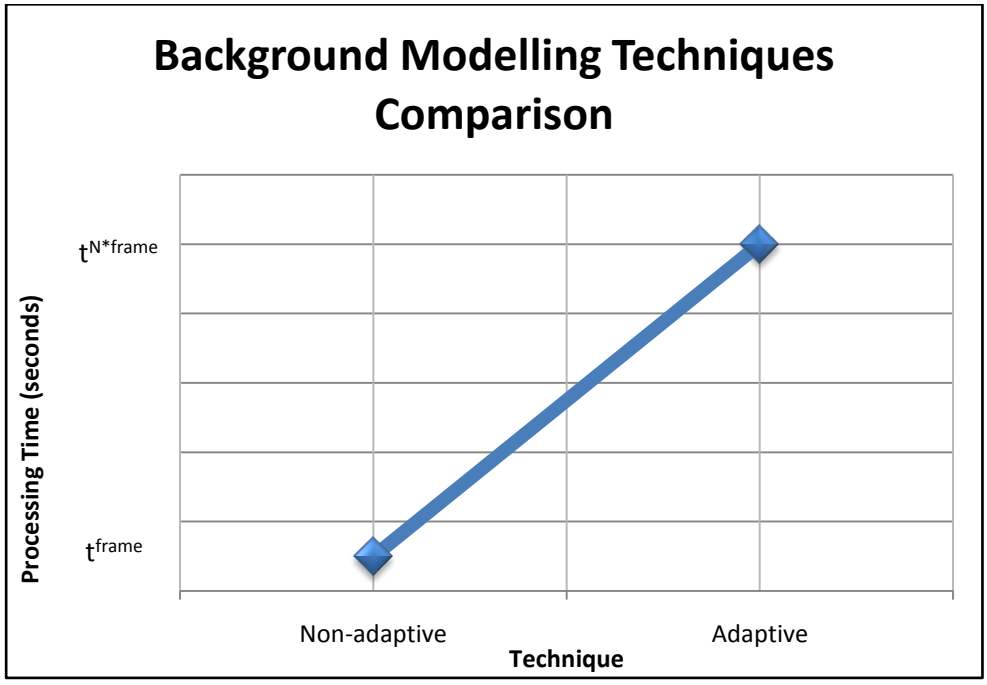

Figure 3 Background modelling technique comparisons

From figure 3 it can be seen that there is an inverse correlation between the technique's effectiveness and complexity of implementation. As we are less concerned with tracking people in this system non-adaptive background model better suits the systems needs. Also non-adaptive technique consumes less processing power which is also necessary for the system as it needs to work real time.

After the background is established, the next frame from the feed is then subtracted with the previous frame. The difference between the two images is then used to assess whether there is motion or not.

There are two ways this difference can be used:

1. **Direct difference** is where the algorithm just checks if the difference is greater than zero. This is highly accurate as even a slight change in the next image of the image stream will set this condition true.

2. **Threshold Vs difference** is where the algorithm checks if the difference is greater than a threshold. Let's say if the system is deployed in a room and there are curtains present in the scene which flutter from time to time. If a suitable threshold is applied then that fluttering can be ignored by the algorithm. And this sometimes can be useful when the system is only supposed to alert the user when an unknown person enters in the scene or in other words when a 'big' change in the scene is detected.

If there was motion then the latest image is taken as the new background. In the case of no motion, no change is needed for the background as there is no point in storing the same or closer image again. Then the algorithm loops back to the beginning.

3.2 Implementation

The implementation is done using the software package MATLAB®. The motion detection module is closely based on the motion detection graphical user interface developed by David Tarkowski[4]. The functions used are taken from the image acquisition and processing toolboxes provided in MATLAB®.

A video input object is created by using the 'videoinput' function this provides a communication bridge between MATLAB® and the video capture device. The adapter which enables this communication is 'winvideo' for this system.

The implemented code uses 'imabsdiff' function (1) to calculate the absolute difference between two images. This function utilises Intel Performance Primitives Library (IPPL) which accelerates its execution time. IPPL contains basic functions used in image and signal processing.

$$\text{imabsdiff}(\text{frame new}, \text{frame background}) = \text{frame new} - \text{frame background} \quad \ldots (1)$$

$$\text{graythresh}\{\text{imabsdiff}(\text{frame new}, \text{frame background})\} = \text{result} \quad \text{where} \quad 0 \leq \text{result} \geq 1 \quad \ldots (2)$$

Figure 4 Difference image and its binary image

The resultant difference image is then passed to the 'graythresh' function (2). This function uses Otsu's method[5] to reduce the graylevel image to a binary image (figure 4) and then calculate the optimum threshold which ensures that intra class variance is minimal.

So if the passed image's resolution is 1280x1024 pixels and is in RGB mode, then it will go through each of the (1280x1024x3) = 3932160 cells and will process all of them to give one value. This value will be normalized. If there is motion present then the difference image will contain some other values then black {RGB (0, 0, 0)} in those cells. If nothing changed between those two consecutive frames then this means that all of the cells will be zero. In summary this function processes each RGB value and summarises the difference in one value. (Figure 5)

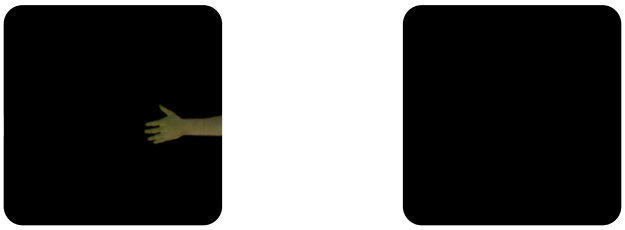

Figure 5 Difference image showing motion and non motion event

If motion is detected then the new image is stored as the new background image. After this the algorithm moves on to the face detection module.

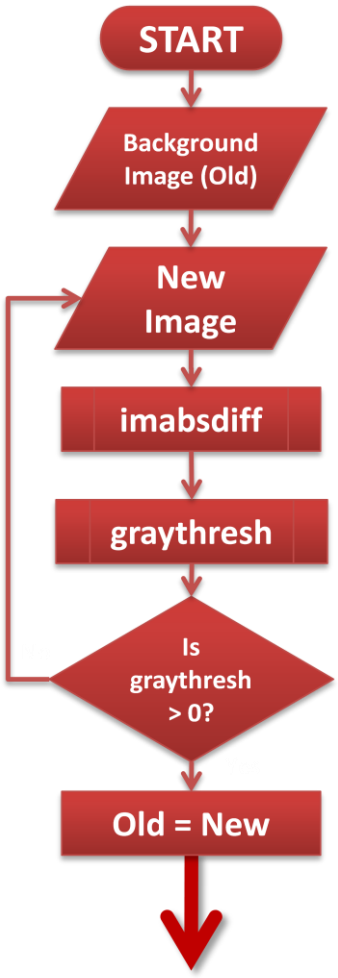

Figure 6 Motion Detection Flowchart

4. FACE DETECTION

4.1 Theory

4.1.1 Different types of algorithms

There are three main approaches considered for implementing face detection. They are 'Neural network based face detection', 'Image pyramid statistical method' and 'Voila & Jones'. Each implementation is discussed below:

1. **Neural network based face detection**[6] (paper by Henry A. Rowley, Shumeet Baluja, and Takeo Kanade) – Faces are detected at multiple scales by calculating an image pyramid. Each image in that pyramid is then scanned by using a fixed size sub-window and its content is corrected. The correction is to make sure that the lighting is non uniform. The histogram is also equalized. This corrected sub window is then passed through many parallel neural networks. The networks decide if there are any faces in the window. All of these multiple outputs are passed through an AND gate to give the final result. As it's an AND gate the number of false detection is reduced.

2. **Image pyramid statistical method**[7] (paper by Henry Schneiderman and Takeo Kanade) – The basic mechanics of this algorithm is also to calculate an image pyramid and scan a fixed size sub-window through each layer of this pyramid. The content of the sub window is subjected to a wavelet analysis and histograms are made for the different wavelet coefficients. The orientation of the object is determined by differently trained parallel detectors. These detectors are trained so that they are sensitive to the different orientations and one which received the highest hit is considered as the best estimate of the actual orientation.

3. **Voila and Jones method**[8] (paper by Paul Viola and Michael Jones) – The main focus of this method was to achieve a reasonably high face detection rate in the smallest amount of time possible. The detector uses features which are calculated using 'Integral Image'. This is a new type of image representation which is introduced in this method's paper. Also a new type of learning algorithm was introduced in this paper which inherits Adaptive Boosting (AdaBoost) algorithm features. This learning algorithm is then applied which selects the critical visual features and generates classifiers. Background or in other words the less interesting regions are discarded by a cascade classifier.

4.1.2 Comparison of algorithms

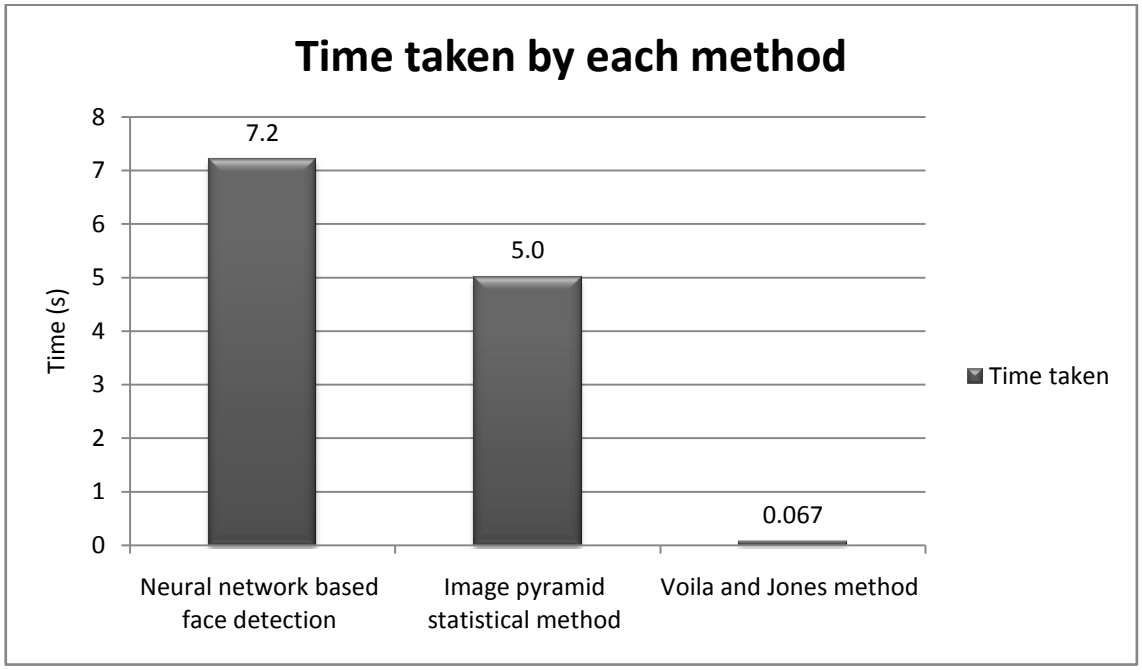

Figure 7 Plot showing time taken by each method. (Values are taken from their respective papers)

The plot above shows the time taken for each method to process an image. The images used to establish those benchmarks were approximately 320x240 pixels and were processed on relatively similar processor.

In other algorithms the speed of detection wasn't taken into account as such. Hence, we can clearly see that 'Voila and Jones' method work significantly faster when compared with others. It is also considered as one of the breakthroughs in the face detection industry.

False detections Detector	10	31	50	65	78	95	167
Rowley-Baluja-Kanade	83.2%	86.0%	-	-	-	89.2%	90.1%
Schneiderman-Kanade	-	-	-	94.4%	-	-	-
Viola-Jones	76.1%	88.4%	91.4%	92.0%	92.1%	92.9%	93.9%

Table 2 Detection rate for various numbers of false positives on the MIT+CMU test set containing 130 images and 507 faces. (Published in [8])

Table 2 shows the detection rate results of the different algorithms. From this we can clearly say that 'Voila and Jones' method also provides reasonably high detection rate.

After analyzing each algorithm's complexity, Voila & Jones seems a better choice. It offers a high detection rate combined with a very low processing time which is what the system needs.

4.1.3 Voila and Jones method theory

This method uses 'Haar' wavelets for feature extraction from the images. These wavelets also allow feature evaluation.

A & B – Edge features

C – Line features

D – Four rectangle Features

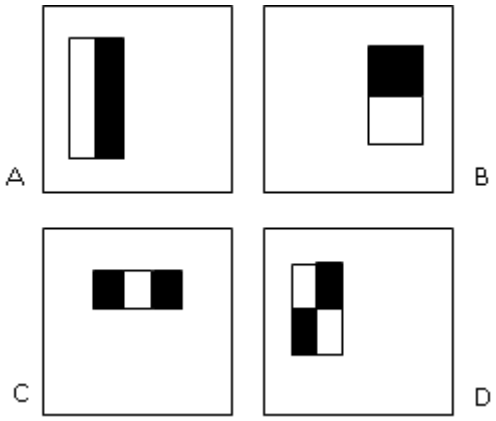

Figure 8 Basic Haar wavelets

They are formed of one low interval and high interval or in other words are single wavelength square waves. A square wave is a pair of one light and one dark adjacent rectangles. The calculation of these wavelets is relatively easy as the white areas are just subtracted from the black ones. Figure 8 shows the four basic types of Haar wavelets in 2D.

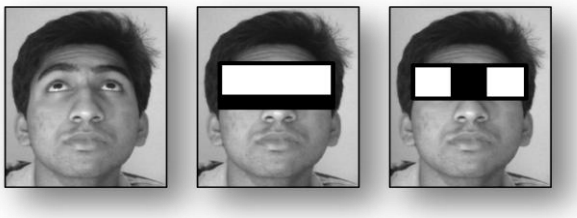

Figure 9 Haar wavelets extracting features

The feature extraction is made faster by integral image which is a special representation of the image. A machine learning method, called 'AdaBoost' enables classifier training and feature selection. All of the detected features are then combined efficiently by using a cascaded classifier. This is shown in figure 10.

Figure 10 Cascaded classifier

4.2 Implementation

After researching, a MATLAB® executable (MEX) implementation of 'Voila and Jones' method was found on MATLAB® Central website. Developed by Sreekar Krishna[9], the code uses Haar classifier provided by Intel for Open Source Computer Vision Library (Open CV).

We only run the face detection and the other module once motion is detected. If there is no motion then there is no point in running them as it will just cause unnecessary processing load. The argument here is that if someone came in the scene then there was an initial motion and those modules have already processed that image once. The gray scale image used in motion detection is passed to the face detection function which is written in C++. The function then returns the number of faces detected and their respective coordinates.

A bug was found in the downloaded code. When multiple faces were detected the passed array from the C++ code to MATLAB® environment wasn't in the right format as expected. This was corrected by changing the format of the array.

Once detection coordinates were acquired, the original coloured image is displayed and the locations of the faces are marked on the picture.

If there are any faces detected then those coordinates and the image are passed to the face recognition module. Else we loop back to the motion detection.

Figure 11 shows the flowchart which is implemented in the code.

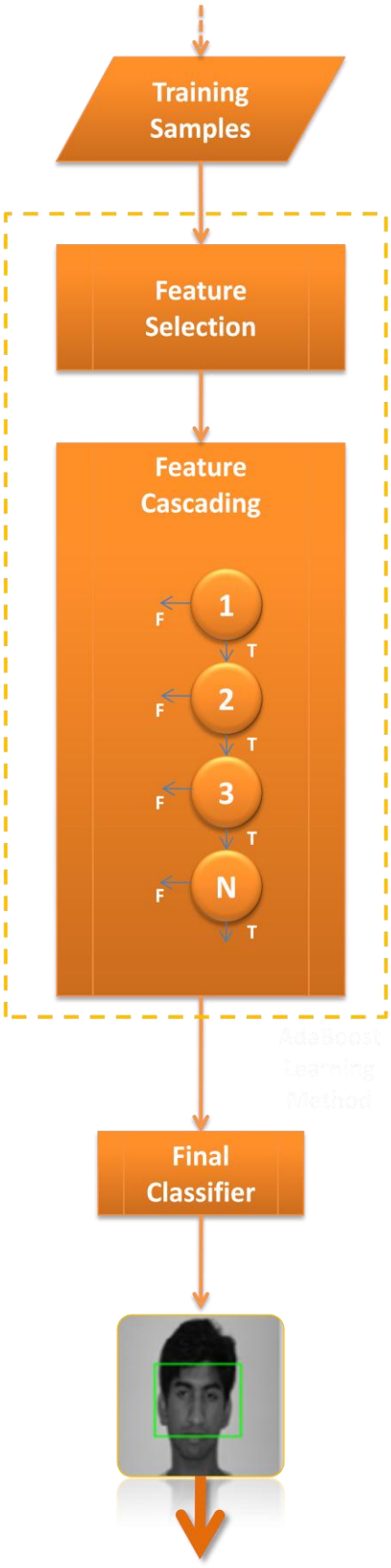

Figure 11 Face Detection Algorithm

5. FACE RECOGNITION

5.1 Theory

5.1.1 Algorithm selection

There are various available implementations of face recognition software. The Face Recognition Technology (FERET) program has already analysed most of those algorithms and have published detailed performance results for each of these approaches.

The main goal of this program was to test each algorithm on the same data sets so a genuine comparative results study can be formed. It is managed by the Defence Advanced Research Projects Agency (DARPA) and the National Institute of Standards and Technology (NIST) so that the best solution can be known and deployed.

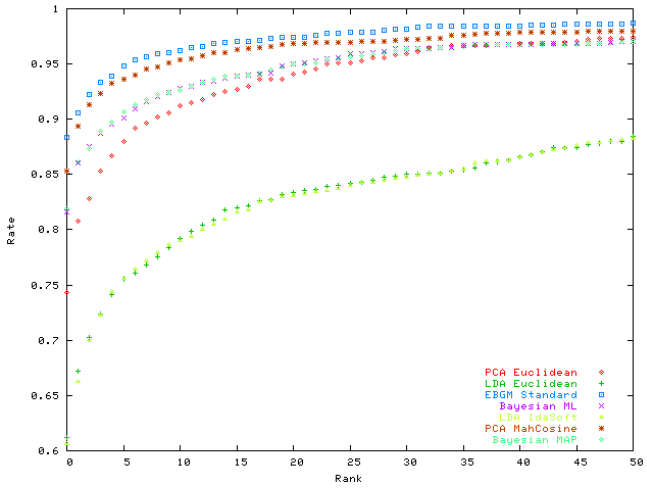

Figure 12 FERET FAFB

Figure 12 shows the FERET[10] FAFB results. In this test each algorithm was run with faces with different facial expressions. All other conditions were kept the same.

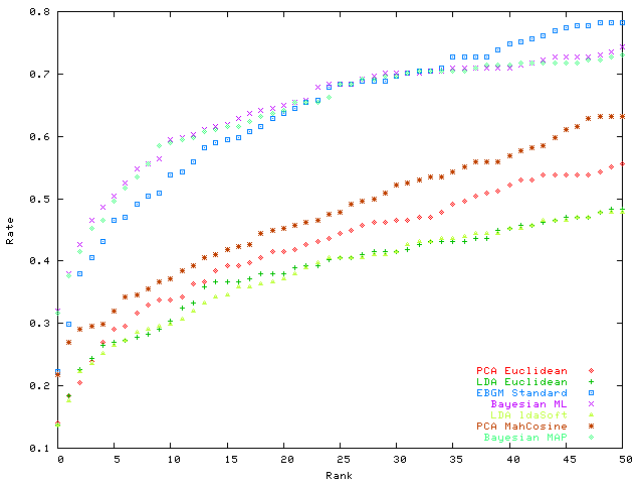

Figure 13 FERET DUP2

Figure 13 shows the FERET DUP2 results. In this test case all of the algorithms were given picture of faces taken with an 18 months time difference.

Out of these algorithms the most promising one seems to be (PCA Euclidean) Eigenfaces method for our system needs. From the results, this method provides a reasonable detection rate when compared with other algorithms. It is based on the principal component analysis (PCA). The main factor for selecting this method is that it's not complex, and is really fast which saves the processing time. It was developed by Sirovich and Kirby[11]. Matthew Turk and Alex Pentland at MIT[12] released a paper regarding the use of this method for face classification.

5.1.2 Eigenfaces method

This method is based upon Principal component analysis (PCA). An initial set of images of faces are used to create a training set. The number of face shots of each person stored in the database depends on how much processing time they will take. These faces are then broken down into individual vectors. The magnitude of each vector represents the brightness of individual sectors of the gray scale image. A covariance matrix is formed by normalizing these vectors. After this eigenvectors are derived from this covariance matrix and a set of eigenvectors of an image forms an eigenface as shown in Figure 14. Eigenface helps in just focusing at the main face features rather than the whole face data. In other words it enables us to find the weight of each face.

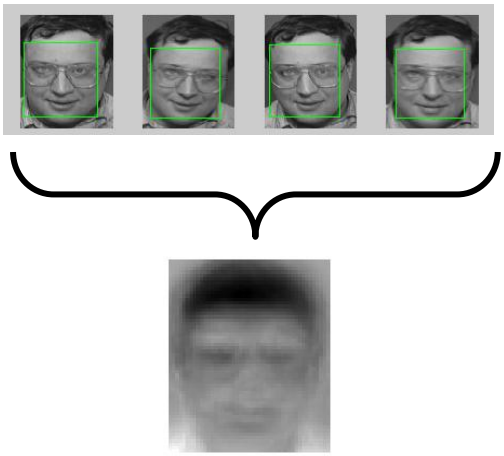

Figure 14 Specimen's images taken from AT&T database and its resultant eigenface

When a new face image is acquired the weight of that face is calculated and then subtracted from the each of the weights of other images in the database. Those difference numbers represents how much different each image is from the original image. The lower the number the closer is the match. This difference is also known as the max Euclidean distance.

5.2 Implementation

The implementation uses the face recognition MATLAB® code developed by Ali Behboodian[13]. The code is meant to work on pictures only. The reason for using external code is that more work is required to make the face recognition work on a live feed and hence having a baseline will save time.

The face detection module passes the frame captured by the camera and the coordinates of the detected faces to the face recognition function.

The recognition function decides if there is any close valid match from the database. In the scenario where the face is 'unknown' the 'ACTION' module is called. This is repeated for all of the faces detected.

After this the function loops back to the beginning for a new frame.

Figure 15 shows the flowchart of the implemented code.

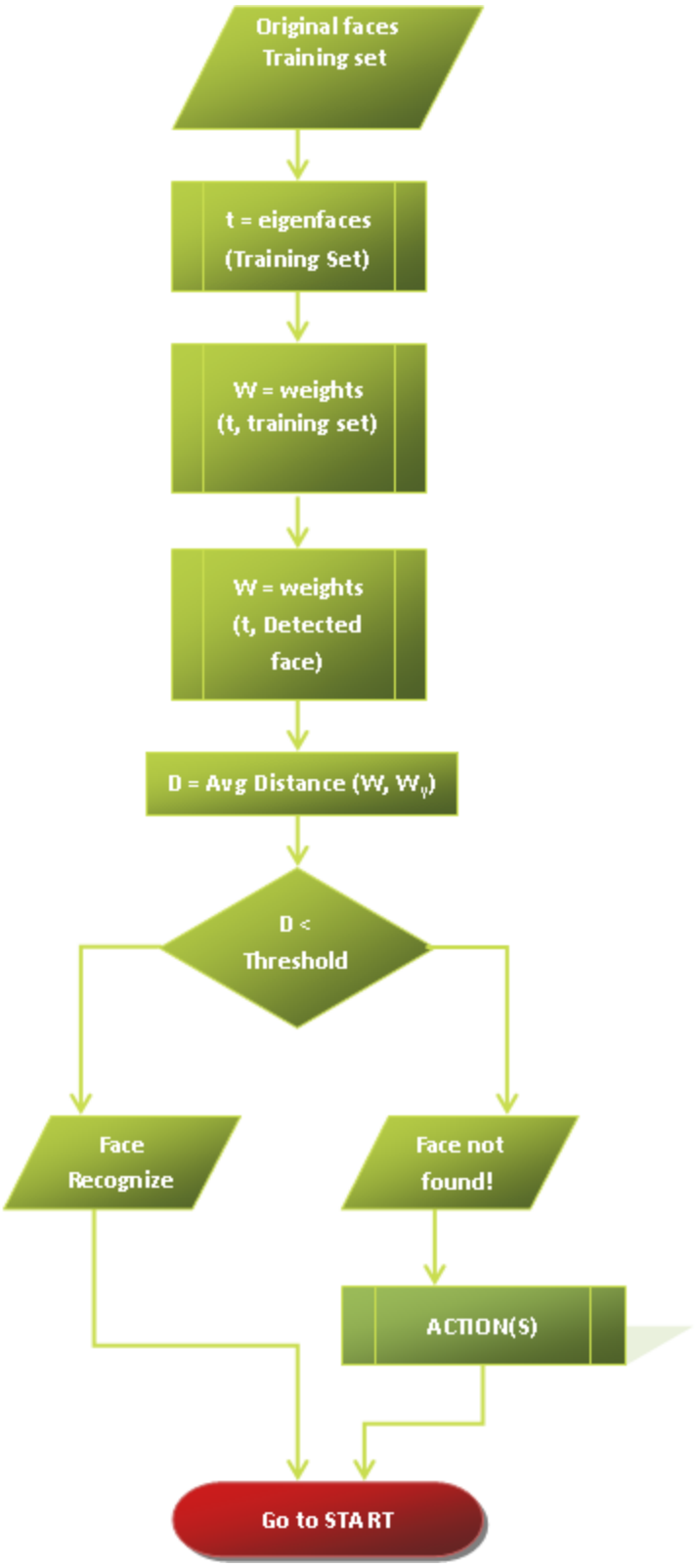

Figure 15 Face Recognition flowchart

6. ACTIONS

Once the face recognition module is unable to recognise someone the action module is called. The following actions are taken.

6.1 Email

The unrecognized face image is emailed to a predefined email address. The image is sent as an attachment and also a warning “INTRUDER DETECTED” is written in the body of the email. The subject of the email is Face Detection/Recognition System. We use ‘sendmail’ function after defining the Simple Mail Transfer Protocol (SMTP) in MATLAB®. This means that in a scene if there are three unrecognized faces then the user should get three separate emails containing those images. This can be really useful if the system is used as a room security system.

The email account is created on Google Mail. The log in information is given below:

Username: Mafacedetrec@gmail.com

Password: **intruder_MA**

6.2 Record

The other action is that a picture of the whole scene is added to a video file when an intruder is detected. When the user accesses the video, the clip will basically have a record of all the activity. The video is stored in AVI file format by using the ‘avifile’ and ‘addframe’ functions in MATLAB®.

6.3 Radio Security Handset

This action is not implemented, but it can be if the system is deployed in an area which is secured by security personal. When there is unrecognized face the software can radio a pre recorded message to them using a transmitter on their designated radio frequency.

After this the function loops back to the beginning.

7. GUI IMPLEMENTATIONS

7.1 Theory

Two approaches to implement the GUI were considered. First one was to implement it in MATLAB®. Although developing the GUI in MATLAB® should be faster, it's not user attractive enough.

Another major issue are the GUI handles in MATLAB®. Normally the main GUI structure is written first and then where necessary the handles are passed to the function who wants to plot. The problem arose when the handles were required by 'localframecallback' function. This function is part of the video object and gets a new frame from the video acquisition device. It only expects two arguments and it can't be redefined to contain three arguments to include handles as it's the object property which is fixed by MATLAB®. A possible solution was to save the handles in a MAT file and then load that file inside the function. This didn't succeed as the link between the handles and the GUI objects was not preserved when the data type was loaded. Another possible solution was to use 'persistent' data type for handles which is a global variable data type for this package, but handles are predefined for the GUI in the 'GUI_OpeningFcn' and hence there data type cannot be modified.

Instead of passing handles in those functions we can save the image as a jpg on the disk and then later on use the main function to load that picture. This is not as convenient as passing handles as more data will be required to be saved to the disk i.e. coordinates, names, counters.

Due to all these factors, a Flash application seems to be the best choice, because the data is still required to be saved on the disk by MATLAB® so it can be accessed later on. Flash also provides an opportunity to make the application look more user friendly and attractive.

The next section shows the implementation in Flash.

7.2 Implementation

The figure below shows the implemented GUI in Flash using ActionScript3.0.

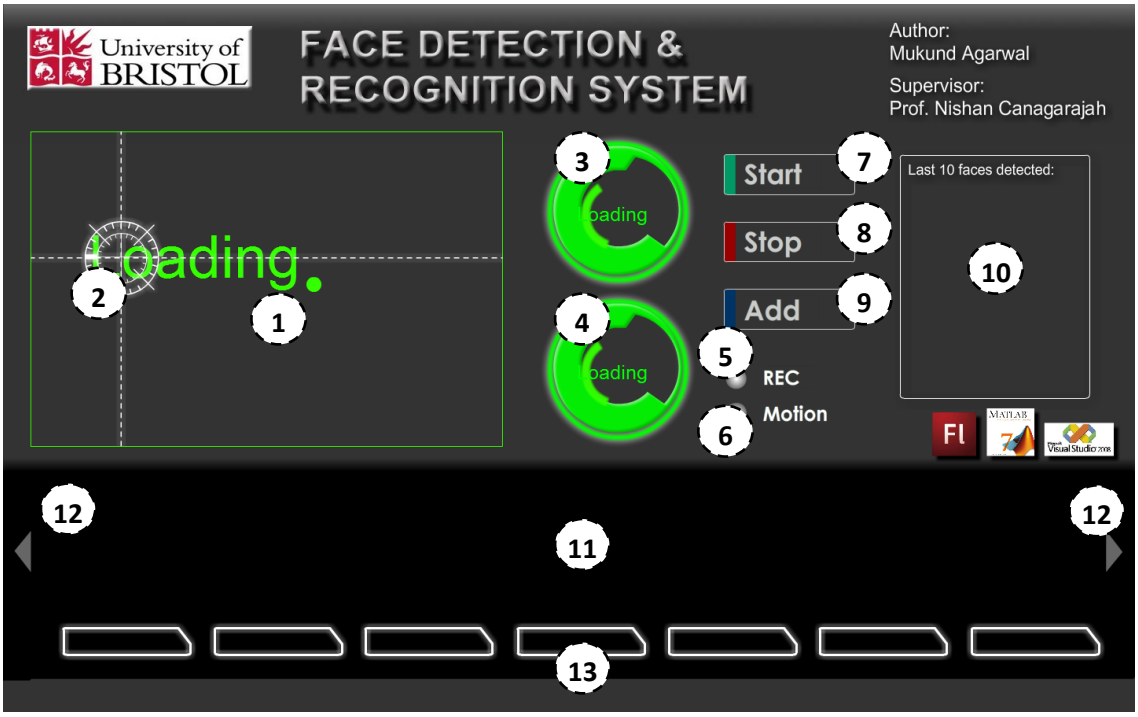

Figure 16 GUI implemented in Flash – During initialisation

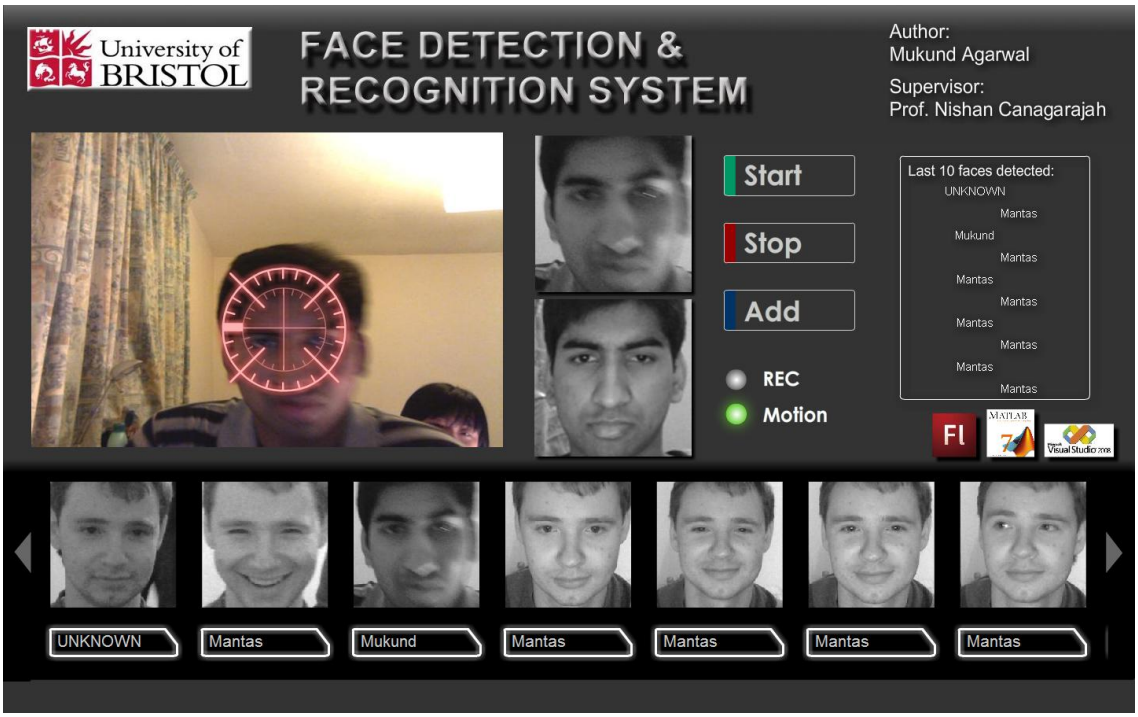

Figure 17 GUI implemented in Flash – During process

The numbered circles on Figure 16 show all of the important features implemented in the GUI. They are explained below in sequence:

1. **Live feed:** This shows the live feed frame by frame.
2. **Coordinate lock on:** This is a small animation which marks the detected faces on the live feed. It does that by turning red and increasing in size. So if 3 faces are detected then there will be 3 'lock on' red crosshairs.
3. **Detected face:** The detected faces are shown in this box in their sequence of detection.
4. **Closest face:** The best estimation by the face recognition module for the detected face is shown here.
5. **Record indicator:** When red it indicates that the current frame from the video feed is added to the video file.
6. **Motion indicator:** When green it indicates that there is motion in the current frame when it was compared with the previous frame.
7. **Start:** This button is to start the MATLAB® code.
8. **Stop:** This button to stop the MATLAB® code.
9. **Add:** This is to call the function which adds a new person's face to the database.
10. **Name of recognized faces:** This box contains the name of the last 10 faces detected by the system.
11. **Recognized faces:** The scrollable photo strip shows the last 10 faces detected. The pictures of only seven faces are shown in the GUI the rest are accessible by the navigation arrows.
12. **Navigation arrows:** They enable the photo strip to be scrolled.
13. **Recognized face labels:** These labels contain the name of the recognized faces. If the face is not recognized then the label displays 'UNKNOWN'.

8. MODIFICATIONS & IMPROVEMENTS

8.1 Averaging RGB

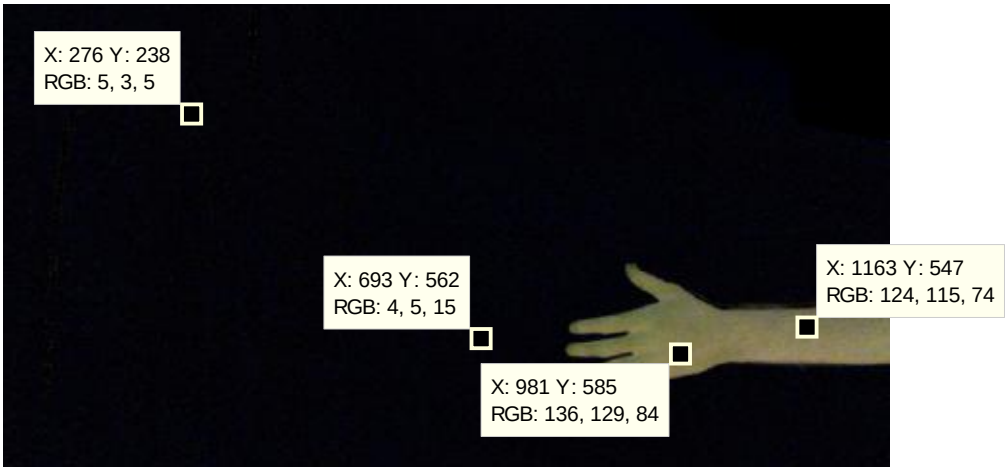

Figure 18 A difference image containing motion and RGB value markers

This is the same picture as seen in Figure 2 & 4. Normally the whole scene is transferred to the face detection module when motion is detected. The face detection module takes an average of 1.1 seconds on the hardware described in section 9.1 to process one picture of 1280x1024 pixels. The time is calculated using 'tic, toc' function, which is a stopwatch timer.

This time may be minimized if instead of passing the whole image only the sections which contain the motions are passed. If we look at the RGB values marked on figure 18, we can see that for the non motion regions i.e. the black region the RGB values are almost zero. If every pixel's RGB value is averaged and then compared against a threshold we will have a simple numerical idea which regions contains motion and which do not. The motion region can then be extracted by a histogram.

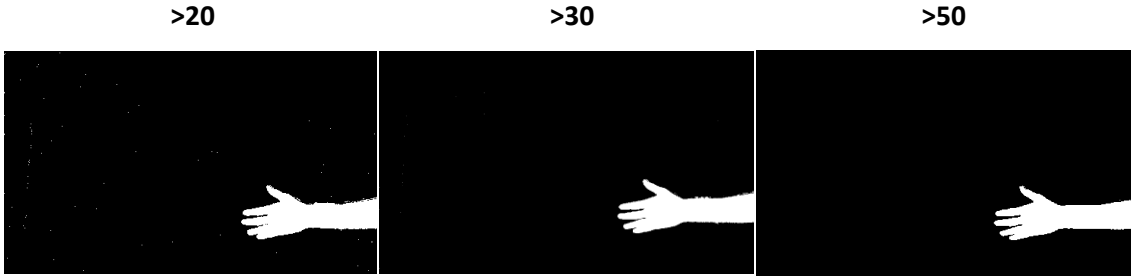

Figure 19 Threshold and the resulting image

The best threshold is 50 as we can see from figure 19.

Threshold	Time (seconds)
20	33.406
30	32.992
50	32.948

Table 3 Threshold and respective time taken

Although this approach provides us a simplified approach the average time taken is 33 seconds by the results in table 3, which is definitely not acceptable. And this time doesn't even contain the motion area extraction via histogram.

Hence, it is more efficient to just send the whole scene to the face detection module rather than just motion areas.

8.2 Timer

The software can be used as a room surveillance system. If the user starts the program, the first frame which the camera will pass may contain the image of the user using that terminal. In the next acquired frame the user will be fully or partially out of the scene which will mean there will be a difference between the background and the new image, resulting into 'motion = true'. So this will record a video of the user leaving which is not useful to the user and is a waste of processing power and storage space.

In order to prevent this, a delay is introduced before the main algorithm is started. This will enable the user to have sufficient time to leave the scene. A timer for 10 second is used by implementing the 'timer' function in the code provided in the MATLAB® library.

8.3 Ratio Check

Initially a ratio check was done to make sure that the detected face coordinates approximately form a square. Before, the results contained some rectangles which weren't faces most of the time. Even if they were faces, the other data in the rectangle area will disturb the face recognition results. The ratio was calculated by dividing the larger side with the smaller one and checking if the result or in other words the ratio was smaller than 3:2. Figure 20 shows the approximate shape of a detected face area expected.

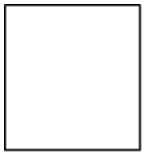

Figure 20 Expected detection area shape

After the bug fix described in section 4.2, the rectangle detections were not seen again. The misreading of the format was causing the rectangles to be displayed.

8.4 Resolution Check

Figure 21 Small false face detections removal

Normally in any picture which doesn't have a background of the same colour, there are multiple false face detections (as seen in Figure 21 as small green boxes). They are recognised as possible faces because when the face detection algorithm searches for Haar like features in the image some patterns match those criteria. When this 'false' information is passed to the face recognition module it will start processing them and will search for the closest face in the database. This has a direct severe impact on the processing time taken by each image.

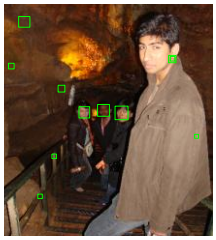

Figure 22 Small valid faces

Even if there is a face detected far away in the scene the resolution of that face won't be high enough for the face recognition module to successfully work (as detected in Figure 22). Using this argument we can safely say that all the small face detections whether false or not can be ignored.

In order to reduce these false detections a filter is applied to the results. This filter checks if the detected face's width and height are larger than a certain magnitude. Figure 21 also shows this filter applied on the left side images. It removes all the small false face detections but keeps the correct face detection result.

This filter doesn't work if the false face detection is of a greater magnitude then the filter thresholds, but now face recognition module receives less false detections.

8.5 Adding A New Subject To The Database

When the user clicks on the 'ADD' button, instead of calling the 'localUpdateFig' function 'localUpdateFig_updateDB' is called. The main difference is in this function is that it doesn't pass the detected face to the face recognition module instead this function stores the face in the database.

It takes ten shots of the faces from the live feed. Each shot is saved in the directory with a unique ID as shown in Figure 23. At the end of the algorithm the user is asked to enter the name of the person, which is then stored in a text file. After this the 'load_database' functions is called which loads all the images from the database including the new ones and then creates a MAT file. This MAT file is used every time when the face recognition module is called.

The user has to make sure that the person whose record he wants to enter in the database is the only one in the scene. The entry can also be made manually by making a new folder and copying the specified content but the user needs to make sure that all the counters are also updated else the database entry will be overwritten.

After the entry has been made, the GUI displays those 10 faces shots in the photo strip along with the shot number. This is just to provide an acknowledgement and also a check for the user.

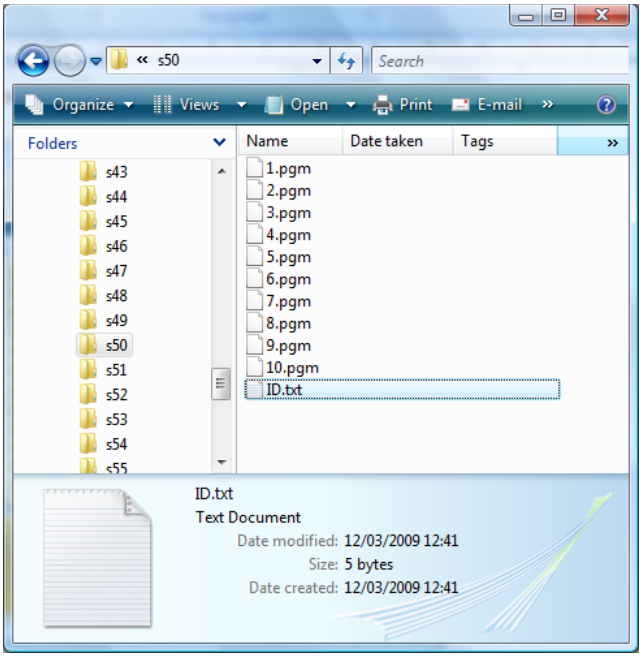

Figure 23 Folder showing 10 faces shot files and the text file

8.6 Average Closest Matches

This is an important modification to the face recognition module. When a face is sent to the module all of the faces in the database are compared and a variable containing the Euclidean distances (5.1.2) for each of the faces is calculated. When this variable was plotted an interesting observation is made.

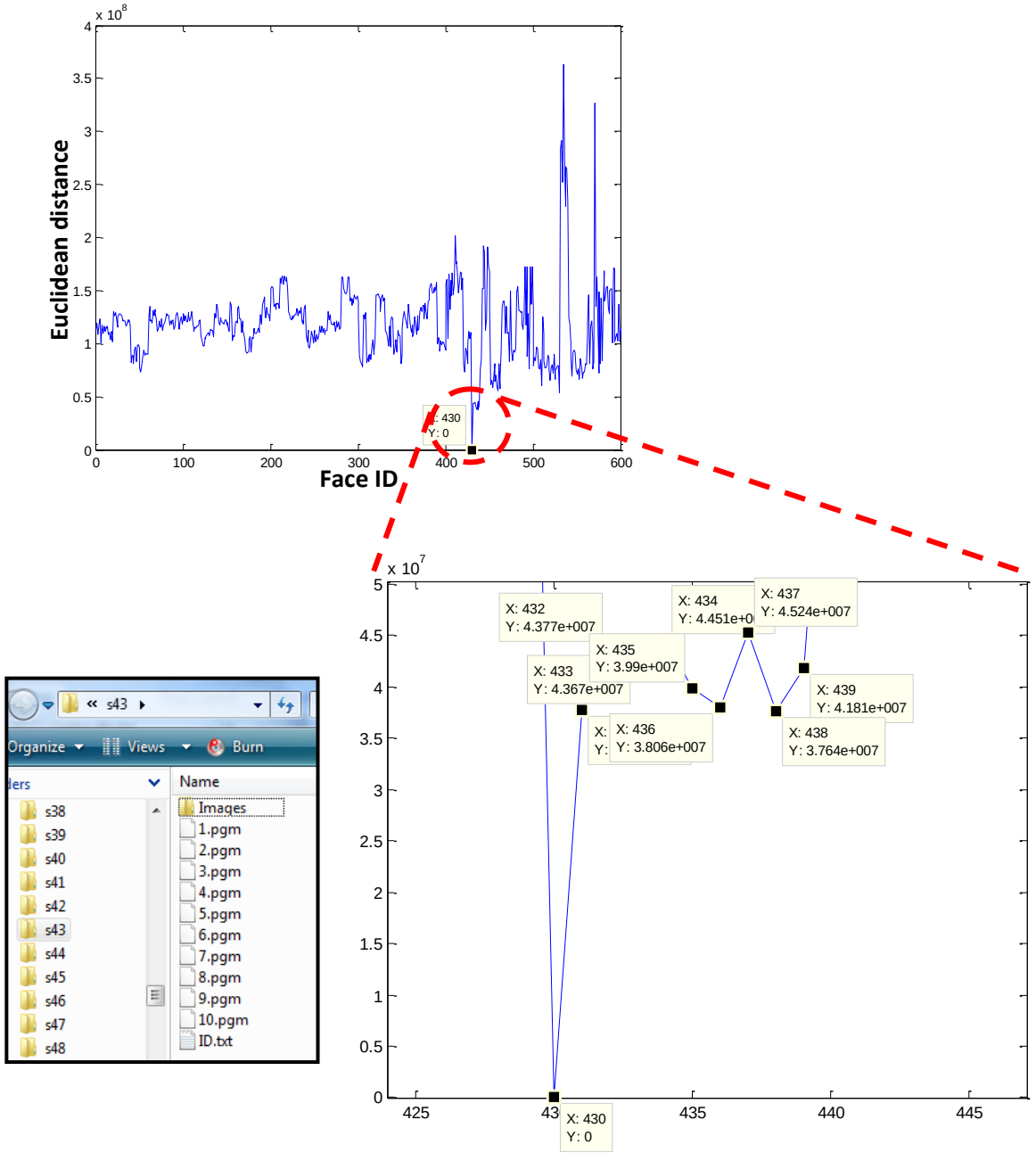

Figure 24 A Euclidean distance graph of a correctly recognized face accompanied by the directory

Figure 24 shows the Euclidean distance graph of a face which was correctly recognized. It can be seen that the ten shortest distances recorded are from the same set of faces. The face used was from this database itself and hence one of them shows zero as distance.

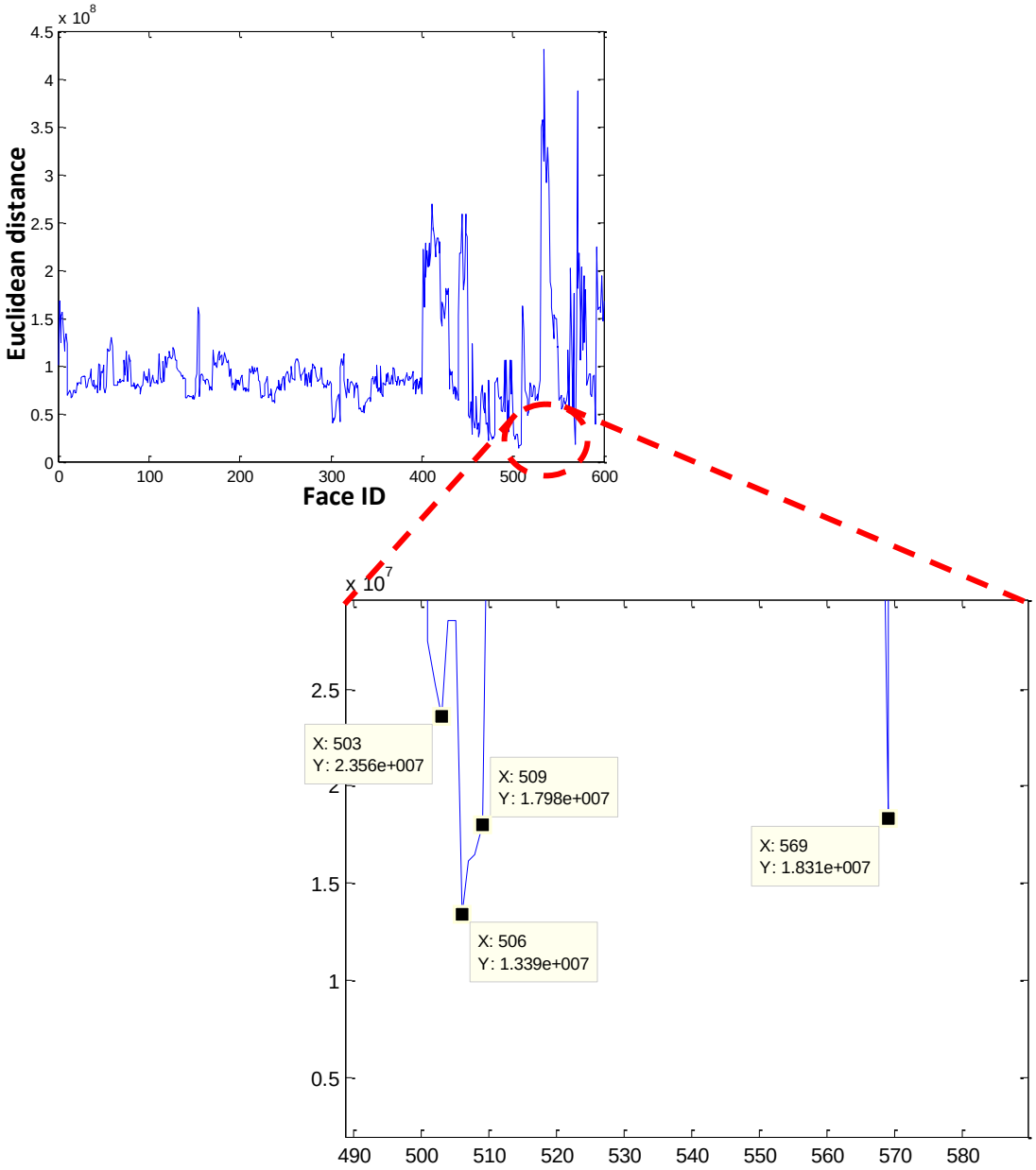

Figure 25 A Euclidean distance graph of an incorrectly recognized face

In figure 25, we see a scenario where the best face estimate is incorrect. There is another face which is pretty close to the minimum one. This can be used as an advantage to automatically detect false positives as normally the system will just display the closest face as the best estimate when it's not even the correct person. To implement this we look at the three closest faces to the input face. Then a check is made if they are from the same set/directory, if they are not then that means that it's an unknown face and hopefully it will be a case of 'correct rejection'. (see appendix 15.1)

In summary, an average of closest matches is taken to see if they are indicating to the same data set.

8.7 'Out Of Memory' Tackle

Sometimes, the software runs continuously for 4-5 minutes and after that the program crashes giving 'Out of Memory' warning. This is because when the database is loaded for the face recognition module, sometimes the memory can't allocate the space required for the variables. The memory is constantly being used by the video acquisition device and other programs.

A solution to this problem was to reduce unnecessary variables and use 'clear' to free memory after each function. This did improve the run time but still can't prevent the crash. Currently, the program sometimes crashes after 10-12 minutes.

8.8 Training Set Size

The face recognition rate can be improved if the training set size is bigger. As more information is being given the results are more accurate but at an expense of the processing time. The best set size for the system seems to be between 9-12 shots of each individual which gives around 70% success rate and takes around 11 seconds. The exact number implemented is 10 shots which mean that the AT&T database of faces can remain.

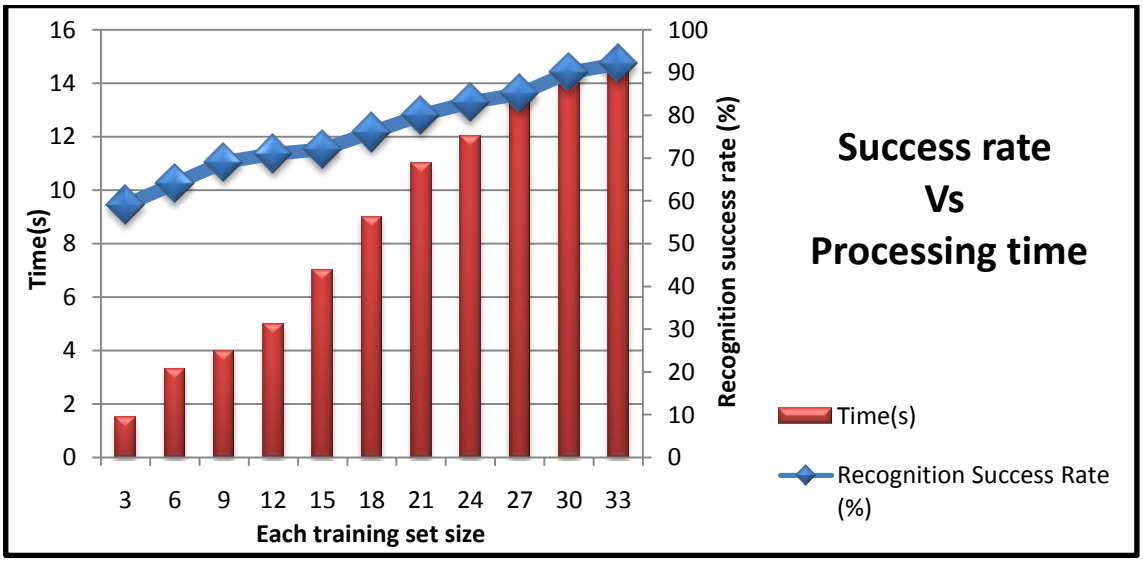

Figure 26 Success rate Vs Processing Time

9. RESULTS & ANALYSIS

9.1 Hardware & Software Used

The software used is mentioned in the 'Software used' in appendix 15.2. It runs on a conventional 2GHz Intel Core 2 Duo CPU T7250 with 2GB of DDR2 SDRAM. This information is necessary so that the performance results taken can be known what they are relative to.

The camera used is a 'Sony Visual Communication Camera VGP-VCC6' providing a video feed of 1280x1024 pixels. It is integrated in the laptop.

The email is received via an Apple iPhone 3G over a 3G cellular data network. The reason for specifically using a mobile device is to test how long will it take for the information to reach. The time taken information is crucial if the system is used as a security system.

9.2 Database Used

AT&T database of faces[14] is used for testing. This database contains 40 subjects and each of the subjects has 10 frontal position photos. The photos contain different lighting and facial conditions and they were also taken at different times against a dark background. Some of them are with different facial details like glasses, smiling and eyes closed. This gives the system a good variation to be tested upon. Figure 27 shows a sample of the database.

Figure 27 Sample from AT&T database of faces

9.3 Motion Detection

A separate script was written to test the motion detection working with the GUI. Live feed was supplied and whenever there was motion in the scene the indicator in the GUI turned to green. This is as expected as the code for this is not complex and nothing should go wrong.

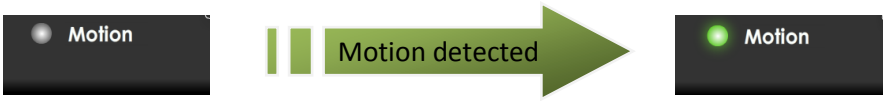

Figure 28 Screenshots of part of GUI before and after motion

9.4 Face Detection

A test script was written to test the face detection module separately on the AT&T database and also some other sets added by the aid of another script (Appendix 15.2). The results were published to HTML files by MATLAB®. Out of 420 faces, 382 were detected successfully by the module. This means that the module yields a 91% face detection rate on frontal position faces.

The faces which weren't detected were those who either were looking sideways at 30-40° or didn't have enough contrast between the features for the Haar wavelet to identify them. Figure 29 below shows some of the results of this test.

Figure 29 Five subjects face detection results

This method has already been tested on the MIT+CMU test set which contains 507 faces and 130 images given in table 2. The only difference is the implementation, but to save time it can be safely assumed that this implementation's detection rate if not same is near to the implementation in that table.

9.5 Face Recognition

Another test script was written to test the face recognition module on the AT&T database and also some other sets added. These results were also published to HTML files by MATLAB®. Out of 420 faces, 396 were recognised successfully by the module. This means that the module yields a 94% face recognition rate on profile photos.

The faces which weren't recognised correctly were mostly those with severe variation in the lighting condition. Also in some of the false negative cases the difference in the face expression was large enough to confuse the algorithm.

Figure 30 below shows some of the results of this test.

Figure 30 Eight face recognition results including two false recognitions

This method performance has already been evaluated by FERET as shown in figure 12 and 13. The profile shot performance is the main concern to this system as the face detection module can only pass faces with less than 30-40° of variation as seen by the results in section 9.4.

9.6 Final System

After all individual modules were tested the whole system was tested. There were following three tests conducted:

1. Adding a new subject
2. Recognising a face
3. Unrecognised face

9.6.1 Adding a new subject scenario

This is to test the modification introduced in section 8.5. As described, after the user clicks on the 'ADD' button the shots of the faces are taken. The user is prompted in the MATLAB® command window to enter the new face name. As we can see from figure 31 the shots taken are loaded in the photo strip.

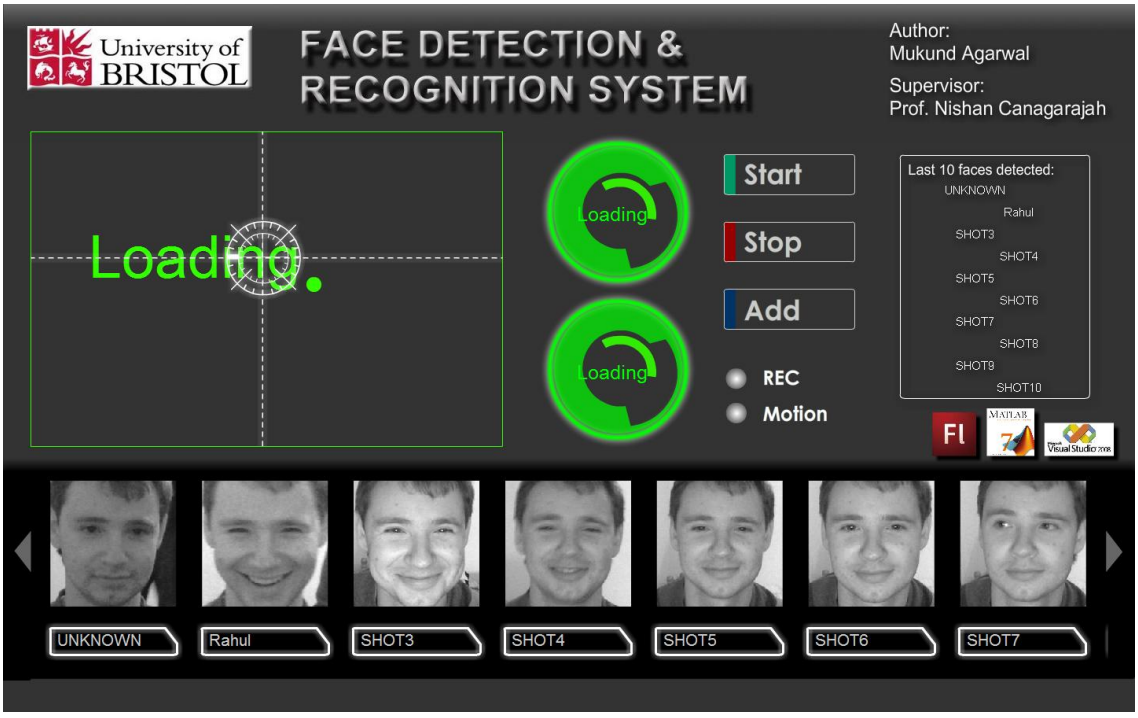

Figure 31 System showing the photo strip containing last 10 faces added in the database

9.6.2 Recognising a face scenario

In figure 32 the system correctly recognises a face. Only motion notification is activate which is expected.

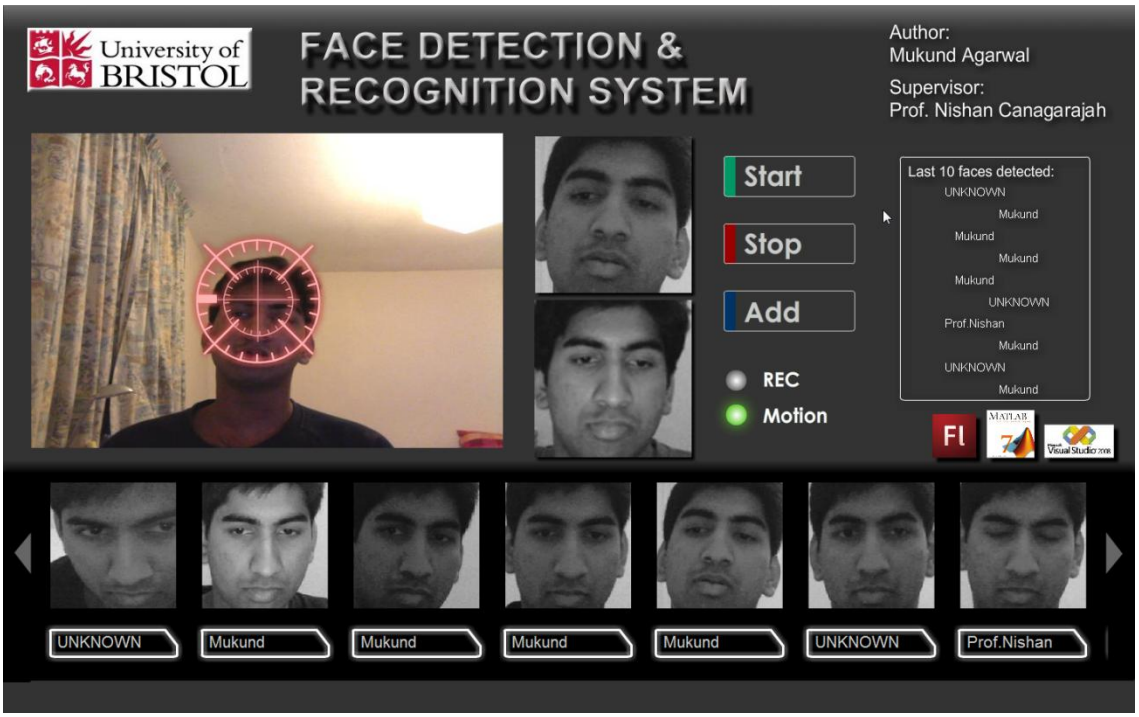

Figure 32 System correctly recognising a face

9.6.3 Unrecognised face scenario

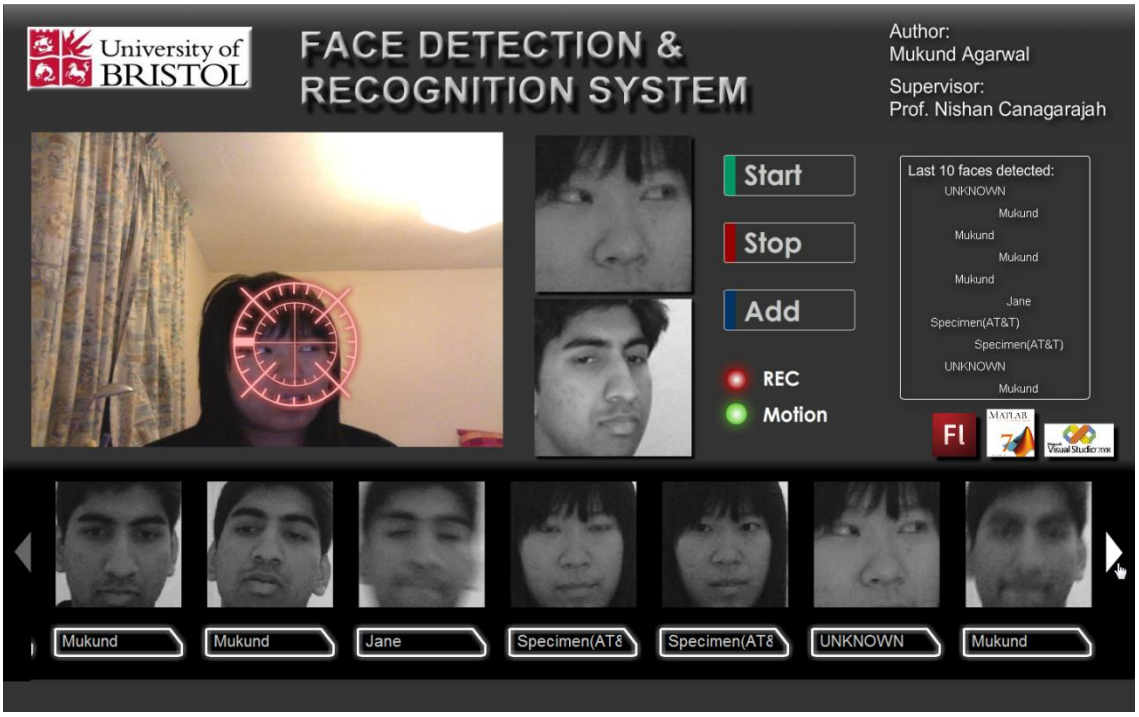

Figure 33 System incorrectly recognising a face

In figure 33 the system detects a face correctly but is unable to recognise. As expected the motion notification is green but this time the record notification is also red. This means that frame has been added to the video file.

We know that the system does not recognise the face as even though it calculates the closest estimate to be someone else, in the photo strip the label reads 'UNKNOWN'. This photo is then emailed to the mobile device as shown in figure below.

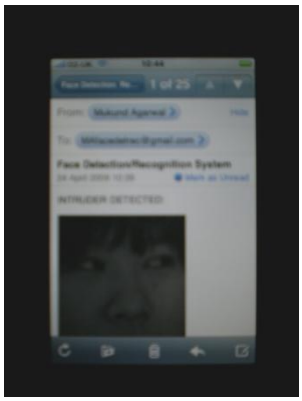

Figure 34 UNKNOWN face email

Below are some more examples of unrecognised faces being received on the mobile device. These photos were taken on the poster presentation day as it provided the best opportunity to test the face recognition module on live data.

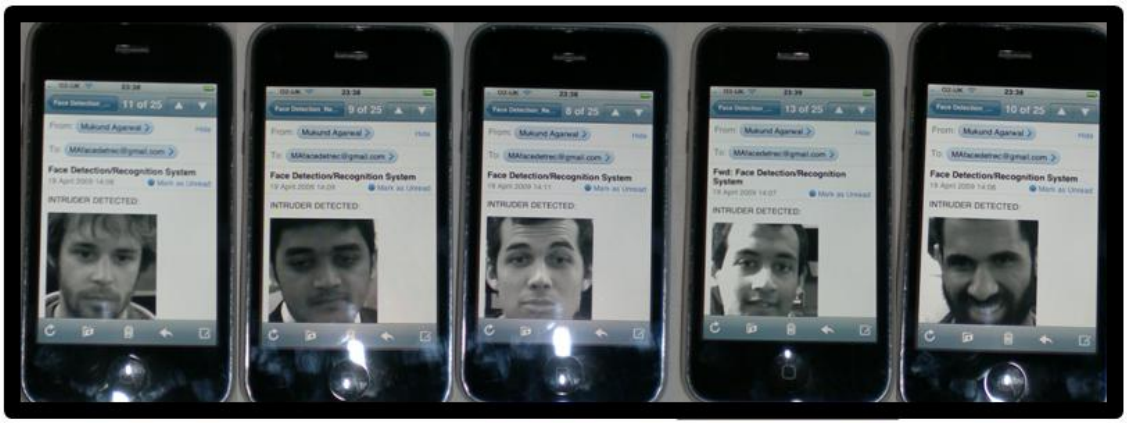

Figure 35 More unrecognised faces

Figure 36 shows an example of false face detections but the face recognition module is working correctly by not recognising them. In the right picture in figure 36 there are two faces and hence the Eigen vectors obtained will be corrupted.

Figure 36 Unrecognised faces (False face detection)

9.7 Time

After all individual modules were tested for their performance; another test was conducted to check the time taken by each module to process one frame. This will allow us to analyse where most of the time is spent and if there is a possibility to reduce it. The stopwatch functions 'tic, toc', mentioned in section 8.1 were used here again for this analysis. Table 4 shows the time recorded for each module.

Module \ Run	1	2	3	Average
Motion Detection	1.1	1.2	1.1	1.1
Face Detection	0.9	0.9	0.7	0.8
Loading Database	2.2	2.5	2.3	2.3
Face Recognition	7.2	7.6	7.3	7.3
Action Module	0.5	0.3	0.3	0.3
TOTAL	11.9	12.5	11.7	12.0

Table 4 Time taken in seconds for each module

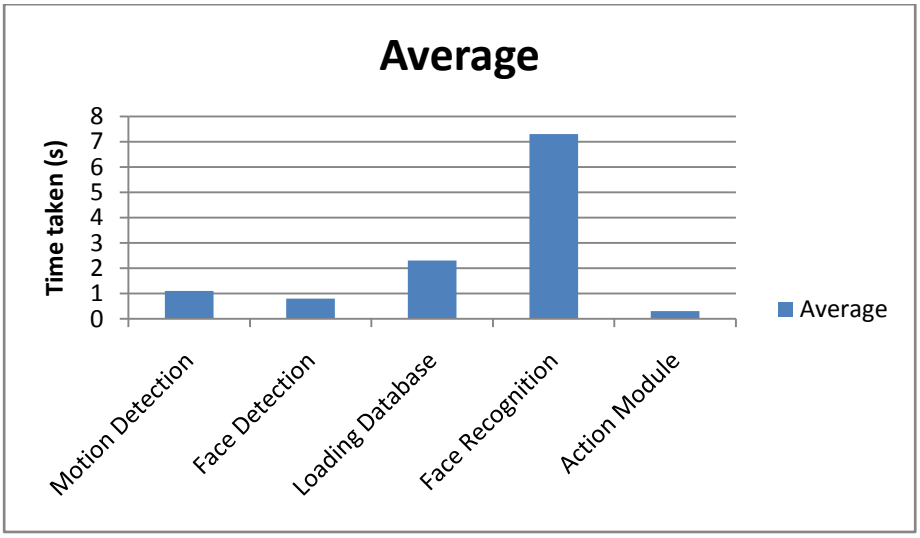

Figure 37 Bar plot of time taken by each module

Figure 37 shows a plot of the time taken by each module listed in Table 4. This graph provides us a better understanding of where the majority of the time is spent. The face recognition module which also includes the database load function takes the majority of the processing time.

Possible improvements are discussed in section 13 as part of possible future work.

10. APPLICATIONS

10.1 Surveillance Systems

The system can be used as a personal security system. This can be popular with students who can't afford insurance. Students are constantly the victims of burglary as found by the crime statistics report[15]. The system is available for free of cost (see section 11). Students can leave the system running when they leave the room. If the burglar enters the room the system will start doing its job.

Even if the burglar steals the PC running the system an email will be sent. If the student sees the e-mail he can directly call the police and the burglar may be caught. Also, the system as we see at least provides some sort of information which then can be passed to the police as evidence.

10.2 Shoplifter Alarm

The system can be used at the entrance of supermarkets. Most of them have a database of known shoplifters. It's the duty of the security personal to keep checking that none of those persons have entered the premises and if they have they are being monitored. This is not practical as most of the times the shop lifters change their appearances by wearing glasses, caps etc. (This problem was found by conducting a meeting with the regional managers of Sainsbury's). The system can compare all the customers entering the premises with that database automatically. If a shop lifter is detected the security can be informed by the method described in section 6.3. The only issue is that the camera has to be facing the customers as they walk in so that it can approximate capture profile view of their faces.

10.3 Smart Rooms

Currently most of the smart homes[16] rely on user input. When the user enters the room and if the system is running the room can be personalized automatically according to user's preference rather than waiting for the user to input. Personalization can be defined as the room's temperature, lighting colour, dimmer settings, curtain settings, music being played, photos being displayed on digital photo frames etc. If two users are detected then the algorithm may be complex enough to set the room's ambience which takes equal elements from both of the users likings. The information containing who is in the room can be extracted from our system by another function.

11. SOFTWARE RELEASE

The software has been released via the MATLAB® Central website. This site provides an open exchange for the MATLAB® user community and hence is the best place for listing the software. Also, the page contains the links to the other MALTAB codes which were referred and utilised in the development.

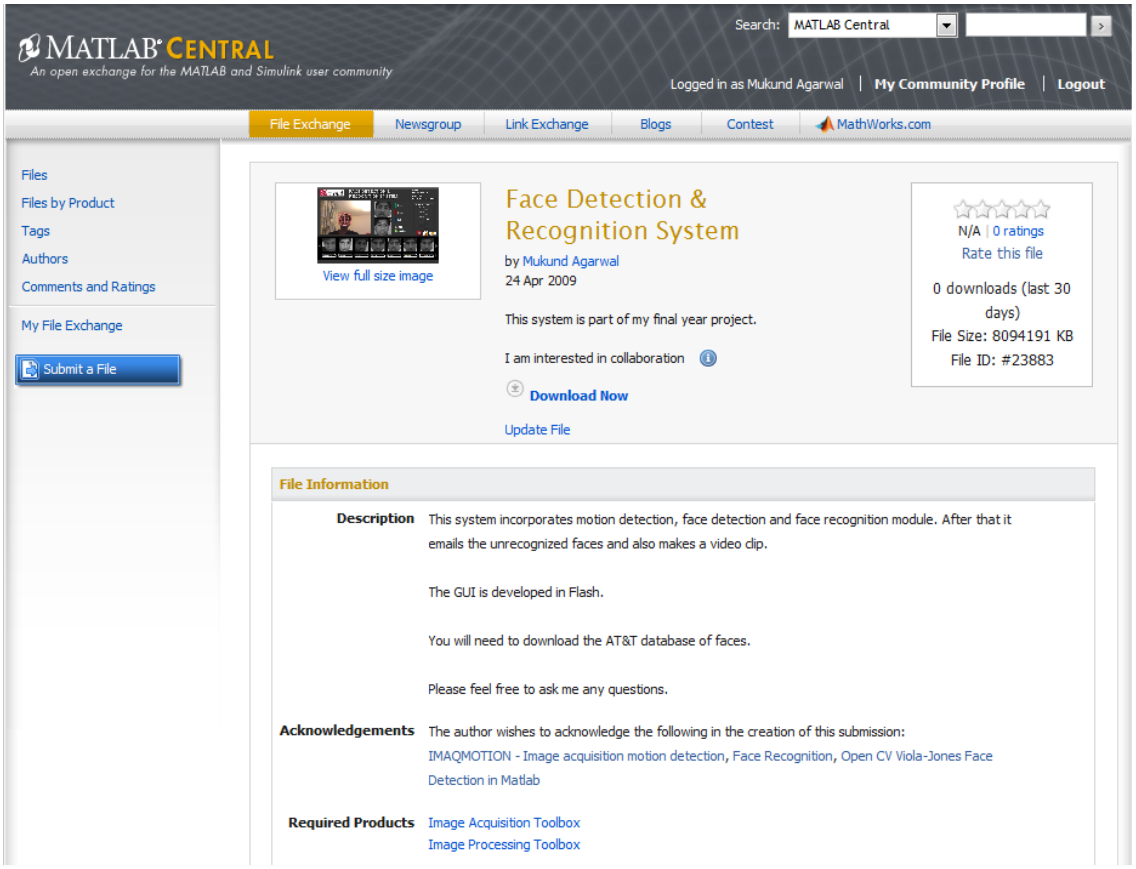

Figure 38 Screen shot of the software listing webpage

The file ID is #23883.

12. CONCLUSIONS

The motion detection module works with no issues observed.

The face detection module does seem to achieve a high correct detection rate but is only limited to profile photos of subjects. The Haar wavelet sometimes misinterprets face like patterns as faces. This false face detection has also been observed in commercial software such as iPhoto 09.

The face recognition module also provides reasonable correct recognition rate but fails to do this in a sufficient time which downgrades the whole system performance. This explains the reason why all of the commercial vendors have released their products in C language. MATLAB® takes more time to convert the code to machine level code when compared with C.

Most of the false positives and false negative cases are mainly due to change in lighting, expressions or non uniform background surrounding the face. Again this has been observed in iPhoto 09 as well.

Both of the aspects of the action module work perfectly. The emails containing the unrecognised faces are sent to the correct email address almost instantaneously. The video clip is recorded containing the right frames.

This system's performance (time) is definitely not low enough for it to be accepted as a substitute for the commercial vendor's products, but if further work is done to transfer the code to C it may be a close challenger. Again the reliability of this system is good enough for a freeware software but not for a commercial product.

Suggestions are made in section 12.7 for a more user configurable system. This will be a better solution as the user can select according to his requirements and also may appeal in particular to students.

13. FURTHER WORK

13.1 Port To C

The time taken to process one frame by the final system is definitely too long. One of the ways to reduce this is to port the MATLAB® code to C code. An effort was made to use the MATLAB® Compiler to produce an executable version of the system. Although, the compilation was successful the executable didn't launch successfully.

There are two ways of porting the code the C. The code can be either directly written in C or it can be compiled to produce an executable by using the MATLAB® compiler. An analysis was carried out to find which will be the best approach. A simple program was written which writes a text file both in MATLAB® and C. The table below shows the results obtained.

Code type	Average Time (seconds)
MATLAB®	0.106
MATLAB® exe	2.033
C	0.039

Table 5 Time taken by different types of code implementation

It can be clearly seen that implementing the whole code in C is the best method to get the most efficient system. As the time was limited no further work was done in this area.

13.2 'Out Of Memory' Bug Fix

In section 8.7, it was observed that sometimes the systems used to crash after 10-15 minutes. If this can be resolved due to further debugging or hardware upgrade then the system will be more reliable.

13.3 Improve Face Detection Rate

This system assumes that the user will be approximately looking in the camera direction. This may not be the case most of the time if the system is being used for surveillance purpose.

This can be fixed by instead of just using the four original set of Haar features, an extensive set of Haar features can be used to improve the face detection. Work in this area has already been carried out by Fabrizio Dini[17].

13.4 Improve Face Recognition Rate

The face recognition success rate can definitely be further improved. The face recognition module was not tested to its full extent in section 9.5. The number of Eigen signatures (Eigenvectors) can be increased to see if there is any improvement in the performance. Alternatively, it can be also checked that if by reducing them is there any significant drop in the performance. If there is no change in the performance as such, than by reducing the number of signatures some valuable processing time can be saved.

There are other various available methods which are more accurate. They can be implemented but the time taken has to be taken into consideration. The most interesting and promising methods are getting a better average face by using twenty shots with different lighting and ages[18] & 3D face recognition[19] .

13.5 Multiple Shots

Current implementation only uses one face to compare with the database. In this modification multiple shots can be taken in sequence of the same face and checked the average closest match.

This will mean that object tracking will also be required to make sure that the multiple shots are of the same face. Also a higher frame processing rate is required for it to be effective.

13.6 Interactive Training Module

An interactive training module can be introduced in the system. This will enable the user to tell the system that if the result of the face recognition module is correct or not. If it's not, then the user can be provided the option of simply just clicking on a button which will start a learning algorithm in the face recognition module. This will also reduce the false positive and false negative results.

This approach is currently implemented in iPhoto 09.

This module may be extended to face detection as well which may reduce the false face detection rate.

13.7 Sound Recorder Module

In the event of an unrecognised face the action module is called. A sound recorder module can be included here. This may be useful in some scenarios especially where the lighting is poor enough for face detection to fail but decent enough for motion detection to work.

13.8 Advance GUI Features

Currently the user can only run the system in one mode, which is leave the software running, faces will be detected in the scene and unrecognised faces will start the action module. The user can be given the following options by introducing small changes to the software and the GUI:

1. **Motion detection only mode:** If motion is detected then the video recording will start and a shot of the scene will be emailed. This can be really useful as there may be non human face objects moving in the scene which will normally go undetected under the face detection module.

2. **Motion and face detection only mode:** Sometimes face recognition gives results which are false positive. If the current system is deployed as a room security system and the intruder is estimated to be the owner by the face recognition module then the owner will not be emailed and this is a major issue. Hence, in this mode by not calling the recognition module a better security system is established. Also, this is perfect for a scenario where the user locks the premises as no face/person is expected in the scene.

3. **Minimize mode:** Currently the GUI runs in full screen mode which might not be the right thing to do if it's being used as a security system. It will be better if the intruder is unaware of the system.

There are other modes which can be researched and included in the system. The main aim is to provide more options to the user so that the software's functionality can be used to its full extent.

14. REFERENCES

[1] Face Sensing Engine website – [http://www.oki.com/en/fse/]

[2] VeriLook 3.2 algorithm information - [http://www.neurotechnology.com/verilook-technology.html#algorithm]

[3] Adaptive background mixture models for real-time tracking by Chris Stauffer

[4] David Tarkowski, IMAQMOTION - Image acquisition motion detection, [http://www.mathworks.com/MATLAB®central/fileexchange/5470]

[5] Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 9, No. 1, 1979, pp. 62-66.

[6] Rowley, Baluja, and Kanade, “Neural Network-Based Face Detection,” PAMI, January 1998

[7] Henry Schneiderman and Takeo Kanade, “A Statistical Method for 3D Object Detection Applied to Faces and Cars Henry Schneiderman and Takeo Kanade”

[8] Viola, P.; Jones, M.: Rapid, "Object Detection Using a Boosted Cascade of Simple Features”, TR2004-043 May 2004

[9] Sreekar Krishna, “Open CV Viola-Jones Face Detection in MATLAB®” [http://www.mathworks.com/MATLAB®central/fileexchange/19912]

[10] *CSU Face Identification Evaluation System 5.0 – FERET results [*http://www.cs.colostate.edu/evalfacerec/algorithms/version5/CSUBaselineResultsV5/index.html]

[11] L. Sirovich and M. Kirby, "Low-dimensional Procedure for the Characterization of Human Faces," Journal of the Optical Society of America A, Vol. 4, pp. 519-524, 1987.

[12] M. Turk, A. Pentland, “Eigenfaces for Recognition”, Journal of Cognitive Neuroscience, Vol. 3, No. 1, 1991, pp. 71-86

[13] Ali Behboodian, “Face Recognition” - 09 Oct 2007 - http://www.mathworks.com/MATLAB®central/fileexchange/16760

[14]The Database of Faces at a glance - [http://www.cl.cam.ac.uk/research/dtg/attarchive/facesataglance.html]

[15] Crime in England and Wales 2007/08 – Police recorded Crime & British Crime Survey

[16] Smart homes information – [www.smarthomes.ie[

[17] Fabrizio Dini, “An application of Voila-Jones algorithm: face detection and tracking”, 2008 – Pg 7 – University of Florance, Italy.

[18] 100% face recognition claim (false but method is high scoring)- [http://www.theregister.co.uk/2008/01/28/hundred_percent_face_recognition_claim/]

[19] “Overview of the Face Recognition Grand Challenge” - IEEE Conference on Computer Vision and Pattern Recognition 2005

15. APPENDIX

15.1 Categories For Test Evaluation

There are four different categories for test evaluation:

1. **Correctly recognized:** a face that is "known" and recognized by the system
2. **Correctly rejected:** an "unknown" face that is rejected
3. **False positive:** a face that is recognized as someone else in the training set; the face could be both "known" or "unknown"
4. **False negative:** a "known" face that is rejected

Source: Lin Yi, "Face Recognition Using Eigenfaces and Neural Network", University of Kansas

15.2 Software Used

All of these codes are included in the optical media supplied with this thesis.

Filename/Algorithm/ Package	**Supplier/Source/Author/ website**	**Use/Modifications made/ Student written**
imaqmotion.m MATLAB®	MATLAB® Central – Witten by David Tarkowski	All of these codes have been used to establish baselines and hence save time. Several modifications have been made by the end of the system.
FaceDetect.cpp MATLAB® & MSVS	MATLAB® Central – Witten by Sreekar Krishna	
Face_recognition.m MATLAB®	MATLAB® Central –Witten by Ali Behboodian	
Project_vold_v4.m Various performance check MATLAB®	Written by myself	This is the main code body of the system. There were several performance analysis scripts written.
Project8.exe - FLASH		This is the GUI code.
SOFTWARE PACKAGE USED		
MATLAB® 7.7.0 (R2008b)	Mathworks	Used to develop the main code.
MATLAB® Compiler 4.10	Mathworks	Used for the trial of exe compilation.
Adobe Flash CS 3 ActionScript 3.0	Adobe	Used to develop the GUI.
Visual Studio 2008 Professional Edition	Microsoft	Used to compile the face detection code.