DETECTING LIVE PERSON FOR THE FACE RECOGNITION PROBLEM

By

Alrashed, H. H.

Sam@iSam.co.nz

07227310

Submitted in partial fulfilment of

the requirements for

the degree of

Master of Information Sciences

Massey University

2016

Table of Contents

List of Tables

Table of Figures

Abstract

Face recognition has been a challenging problem for computer vision scientists for the last few decades. Hence it was the center of attention for computer vision researchers. The purpose of this research is to improve the security of the face recognition system by identifying the liveness of a person in front of a camera to be recognised.

The objective was to detect if the images used to be recognised reflect a real person's face, i.e., a live person's face instead of just a static image of the face. This can be achieved by randomly asking the person to carry out certain tasks. Simple tasks such as blinking an eye or smiling can then be repeated randomly according to the instructions given by the new system, so even a video of the target face made previously would not be able to perform the authentication easily.

Each component of the system were tested separately. The accuracy of the face detection component was impressively at 98.93%. The eye blinking detection uses a new proposed method with a high accuracy of 91%. Face recognition component was also tested and had a high recognition rate of 96%.

Keywords: Face Recognition, Face Detection, Eigenfaces, OpenCV, Face Anti-Spoofing, Eye Detection, Smile Detection, Eye Blinking Detection

Acknowledgements

I would like to express my sincere gratitude and regards to my supervisors Doctor Andre Barczak and Doctor Napoleon Reyes for their guidance and support during the course of my thesis.

I must express my very profound gratitude to my beloved wife for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without her. Thank you.

Finally, I would also like to acknowledge Intel for providing an open source library that is heavily used in this project that is OpenCV library.

1. Introduction

Face recognition has been a major subject of studies in the past few decades. It is something that comes natural to us humans, our brains do complex analyses of faces in order to store useful information about them, which comes handy when trying to recognise a face by simply picking the match [1]. Our brains also distinguish between an image or video of a person and a real person. Scientists still do not fully understand how the brain functions [2]. However, that did not stop scientists from taking some tasks that the brain can do and try to break them down to simple steps to have a general understanding of that task. Computers are becoming an essential part of our lives such as desktop, phones, glasses and many more; their task is to make life easier. Since computers are growing dramatically in terms of performance, teaching a computer how to do brain-like tasks become more feasible than ever before.

The automatic identification of a person by their faces from an image or video stream has wide commercial and scientific applications. However, if the automatic identification is not secured and reliable, the use of it commercially and scientifically will be limited. As for most biometrics system, they are vulnerable to spoofing attacks.

Face recognition subject emerged in the early 1970s; however, its rapid development began in the 1990s, after the establishment of new technologies in the field of image processing and machine learning.

Face recognition system is a program that is used to identify faces automatically and verify the identity of a person from either image or a video. In general, the face recognition problem is composed of two stages. First, detecting a face, i.e. finding a face in an image frame regardless of who the person is. Second, identifying whom the person is in the frame. By comparing the features of the detected face to the image faces database, the system can identify the person in the image. To be able to label the person in the image, the machine should have been trained beforehand.

The training steps involve detecting a face then use image process techniques to ensure the clarity of the face for the machine. After that, applying one of many training algorithms to teach the system who that person is.

The main problem with the face recognition is that it can be breached very easily. Spoofing attack is a fatal threat for biometric authentication systems. There are numerous researches and publications on face recognition approaches, however, anti-spoofing and verifying the person is live and real is very limited [3]. The most obvious and common face recognition spoofing method is to use a static image of a person known to the system and expose it to the system's camera to authenticate.

The aim of this thesis is to develop methods to prevent spoofing face recognition system and detect if the person trying to authenticate using the face recognition system is a real person and therefore it is not an image or a video of that person.

The following steps will help to achieve this goal:

- Research and implement a face detection algorithm.
- Research and implement face recognition algorithm.
- Research and implement eye detection and eye blinking detection methods.

- Research and implement mouth and smile detection algorithm
- Integrate all above in graphical user interface GUI.

1.1. Motivation and Objective

The main motivation was to explore the security aspect of face recognition. Moreover, I wanted to create an easy and secure mechanism to access an application using only the basic hardware such as camera and PC or mobile device.

Therefore, my objectives were as follows:

- Create an application that can learn faces then identify them whenever the application sees those faces again.
- In order to create the application, I need to investigate the face recognition algorithms and understand the techniques to implement them.
- Finally, implement the liveness detection by blinking and smiling from the user.

To formulate the problem: given an image with a face in it, how can we detect the face then determine the identity of the person in the image from known people to the system. In the mean while, ensuring that the identified person is a live person rather than an image or a video sample of the same person.

The following flow-chart displays the complete integrated system of the face recognition with authentication of the persons' actions (Figure 1).

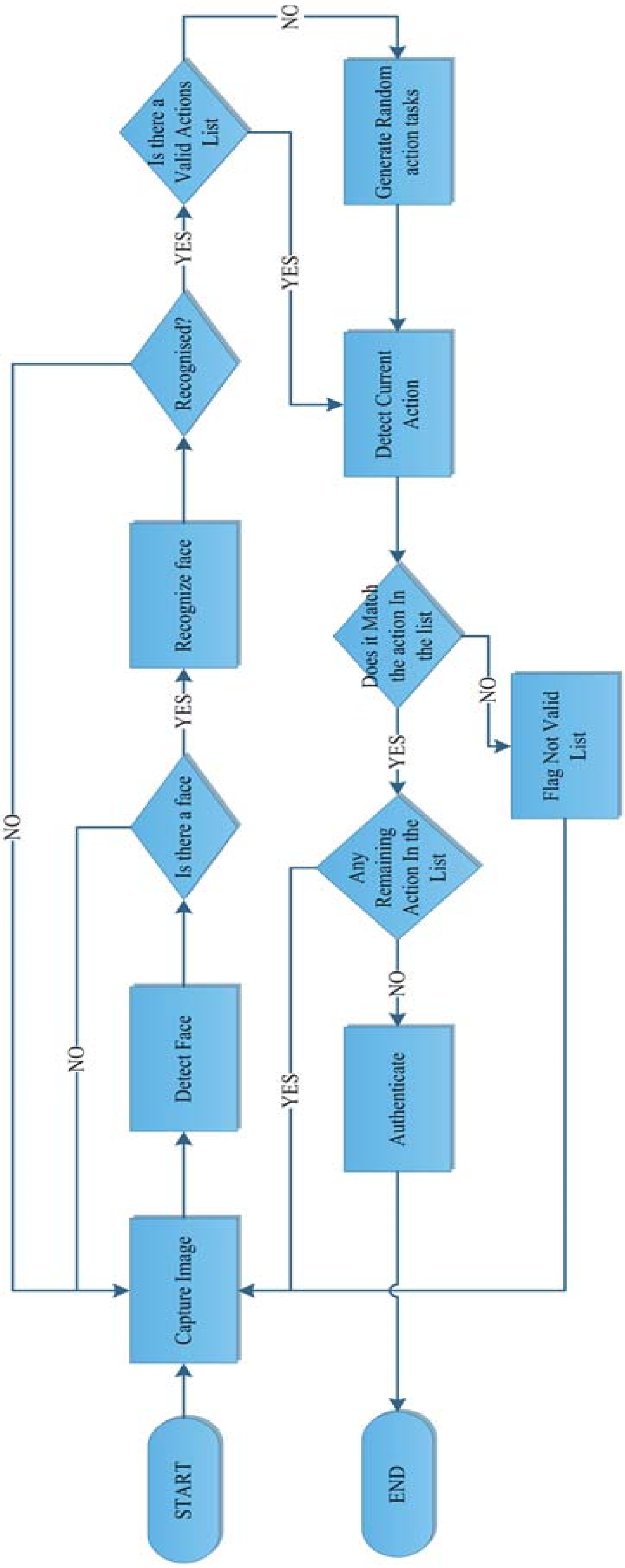

Figure 1 system Flow-chart

2. Literature review

We, as humans, use faces to recognize and identify our friends and family. Computers can now also identify people automatically using stored information such as figure, iris or face to identify a particular person. Earlier, many face recognition algorithms were used to achieve fully automated face identification process.

The first face recognition system was created in the 1960s [4]. It was not fully automated and it required manual inputs of the location of the eyes, ears, nose and mouth on the images then it calculates a distance to some common point then it compares it to the stored data. In 1971, Goldstein , Harmon and Lesk used some specific features of the human face such as hair color, nose size and lips thickness trying to automate the recognition process [5]. The main problem back in the 1960s and 1970s was that manual inputs were required. The late 1980s Sirovich and Kirby used Principal component analysis (PCA) a standard linear algebra technique to reduce the complexity of the face recognition problem [6]. In early 1990s Turk and Pentland found that using the Eigenfaces technique, the residual error could be used to detect faces in an image [7]. This was an important discovery of the history of the face recognition. It enables real-time and automated face recognition. Since then automated face recognition has been evolving and became a major interest for researchers in image possessing and computer scientists.

Face recognition, sometimes is called face identifying, is simply putting a label to known faces just like human as mentioned above. We learn the faces of our family and celebrities just by looking at their faces [8]. Since the 1970s there were many techniques and algorithms were developed for a machine to learn to recognize known faces. Most of the recent techniques involve at least four steps:

- Face detection. E.g. Viola and Jones method
- Face preprocessing. E.g. Filtering.
- Collecting and learning the faces. E.g. applying neural networks or AdaBoost.
- Face recognition. E.g. Eigenfaces, Fisherfaces.

Once the face is recognised and has been identified, the anti-spoofing techniques take place.

2.1. Object Detection

Object detection has been one of the most actively researched topics in image processing [9]. There are many algorithms developed for object recognition. The most commonly used one is Viola and Jones method for object detection.

2.1.1. Viola and Jones method

The basic principles that Viola and Jones method [10] is based on four techniques:

- Images use the integral* representation that allows a machine to compute the required object features.

- Using Haar-like features [10], the desired feature of the object can be found.

- Adaptive Boosting † used to select the most applicable features for the required object to this part of the image.

- All the selected features from the previous step are input to the classifier, which return the result of true or false.

Figure 2 and 3 illustrate the features type and selection.

The idea of the face detactor based on Haar-like features is that the image of the face captured by the camera is scanned for the most frontal faces. It detects the regions of the face, so that the eyes region should be darker than the forehead. Similarly, the mouth regions is darker than the cheeks, and so on. On average, 20 stages of comparisons of the regions determine if there is a face in the frame or not. However, this must be done for each possible size of the face as well as each possible position in the image. Therefore, it often does thousands of checks per image to conclude if a face is present in the image [8].

Viola and Jones method, as a method of supervised learning, is divided into two steps: training the classifier based on a large number of positive examples (that is to say, objects of interest, e.g. faces, mouth

---

* An image data structure used by Viola and Jones to compute the feature of the image faster

† A set of weak classifier companied together to create a single strong classifier

or eyes) and negative examples, and a detection phase by applying the classifier to unknown images.

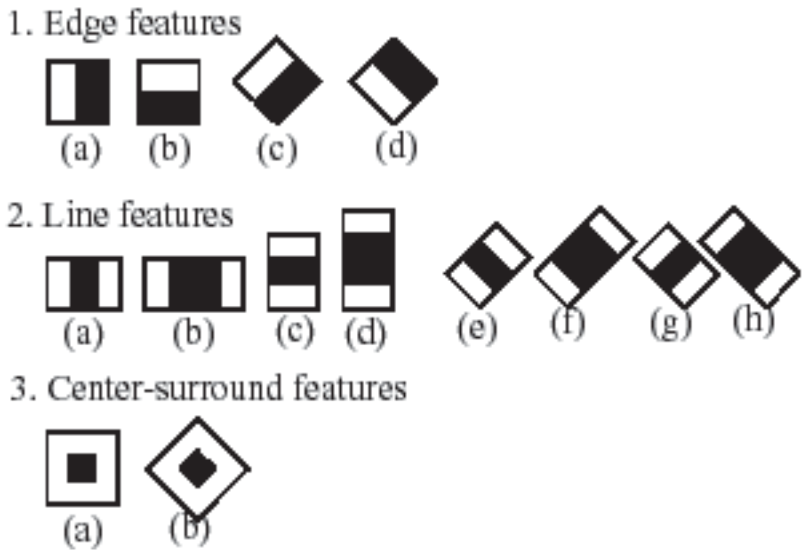

Figure 2 Type of features for Haar-like [11]

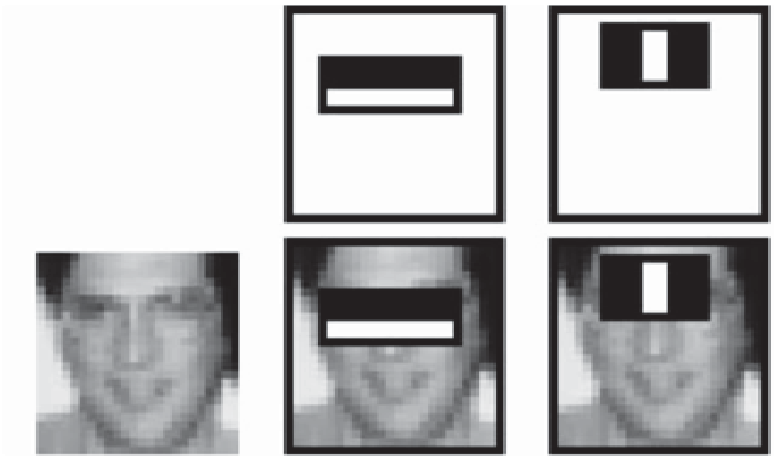

Figure 3 select a small region of the face to calculate its features [11]

The downside of Haar-based-cascade is the training of the classifier is very slow to train, it can take up to a week [8] using a modern PC ‡,

‡ Personal computer Intel dual core i5 (2.5GHz) or faster, RAM 4 GB or more, with low cost webcam

but once the classifier is trained it will detect object in real time. According to Baggio, et al. the classifier can be trained using at least 1000 unique objects (face, eye or mouth) (positive images) and 10,000 images that do not contain the desired object on them (negative images) [8]. The negative can be anything other than the object the classifier is training to detect. The training of the classifier produces an XML file to be used in the detection process. This file can then be ported into any devices or embedded system and used to detect faces. Therefore, the user devices need not to train the detection classifier but use the trained classifier.

2.1.2. Face Detection

Face detection is a process of locating a face inside an image frame, regardless of the identity of that face. Prior to the year 2000, there were many techniques for face detection, however they were mostly unreliable, slow and required manual inputs. In 2001 Viola and Jones developed the Haar-based-cascade Classifier that revolutionized the face detection method. It can detect objects in real time with an accuracy of 95% or more after Lienhart and Maydt improved it in 2002. It works not only for frontal face view but it can detect faces from side view as well [8]. In 2006 Ahonen, Hadid, and Pietikainen developed the LBP (Local Binary Patterns), which is based on the Haar-like cascade but instead, it uses the "histograms of pixel intensity comparisons, such as edges, corners, and flat regions" [8].

2.1.3. Eyes Detection

For the face detection, it is an essential part to detect the eyes within the face during the face detection to minimise the false positive (detecting an object similar to a face). Detecting the eye can help prepare the image for the training. This is because we can assume human eyes are always horizontal and on opposite location. The eyes on a person's face have a standard location and size, regardless of the face expression, lighting conditions or distance to the camera. "It is rare that the face detector and two eye detectors will all be fooled at the same time, so if you only process images with a detected face and two detected eyes then it will not have many false positives" [8]. Eye detection is used for many application such as eye tracking system for mouse control on a PC.

In the literature there are few methods for detecting an eye within human face. The most commonly used algorithm is Viola and Jones method for object detecting. The object of interest is the eye(s) which the classifier can be trained to detect within a face.

Other eye detection techniques and methods are Chrominance-based method, Skin Detection-Based method and template matching technique.

2.1.3.1. Chrominance-based method

This method was proposed by Hsu, Abdel and Jain to detect the eye based on illumination of the regions where the eyes are. According to this method, two separate eye maps are constructed from red (Cr) and blue (Cb) chrominance parts of the picture. The eye map from the distinctive parts depends on the perception that relative low Cb and low Cr values are present around the eyes. Also, the primary region of face is of high red part and generally high blue segment in a few conditions which implies these two segments can identify regions of eyes [12] (Figure 4)

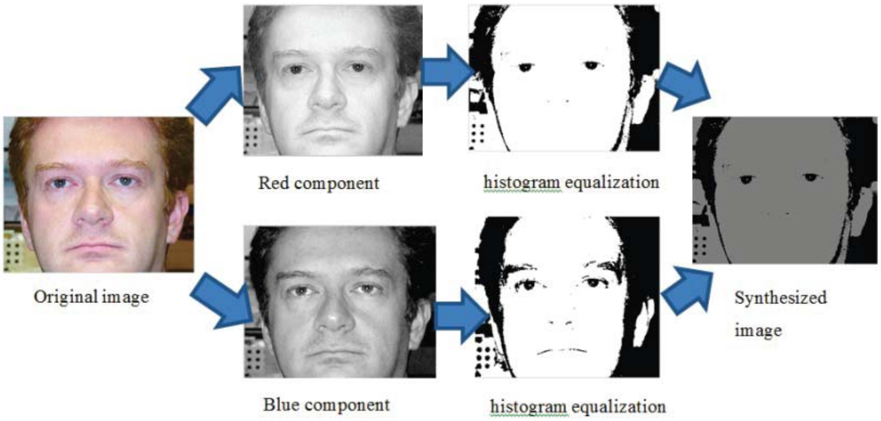

Figure 4 Chrominance based method flow [13]

2.1.3.2. Skin Detection-Based Method

This method requires pre-processing the face to determine the facial area, i.e eyes region, mouth region etc. The skin detection method uses the face skin colour. As the skin colour usually consistent across the face, a suitable threshold value function was found by Zhu, Xia, Zhou, & Zheng in 2014 in which the colour can be extracted and remove the skin region [13] Figure 5).

The remaining parts of the image are the eyes, hair and the background of the image. Because the area of the face was mapped in the preprocessing step, we can locate the eyes region and process it accordingly.

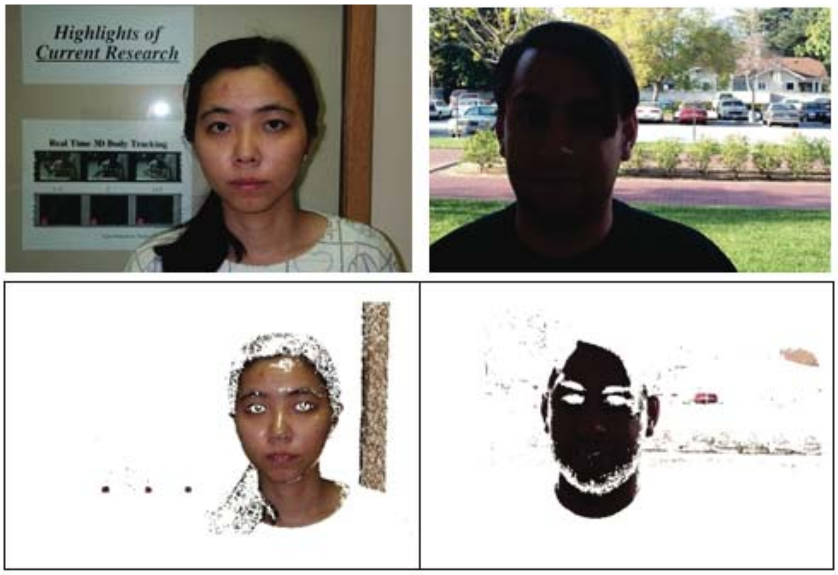

Figure 5 Skin Detection-Based Method flow [13]

2.1.4. Mouth Detection

The mouth detection is a set of instructions to detect the mouth within a person's face. The most popular methods is Viola Jones method for object detection. By using Local Binary Pattern to train images of mouths as positive images and using other parts of the face as negative images to create a classifier to detect a mouth within a face. In addition, the face region can be divided into two sections, the upper section and the lower one. The region of interest is the lower section where the mouth is located. This helps improve the accuracy of the mouth detection techniques.

2.2. Face Pre-processing

After detecting a face in the frame shown in Figure 6 the face inside green rectangle can now be processed.

Face recognition is susceptible to changes in lighting conditions, face orientation, and face expression, so it is paramount to diminish these differences as much as possible. There are numerous techniques to eliminate those issues [8].

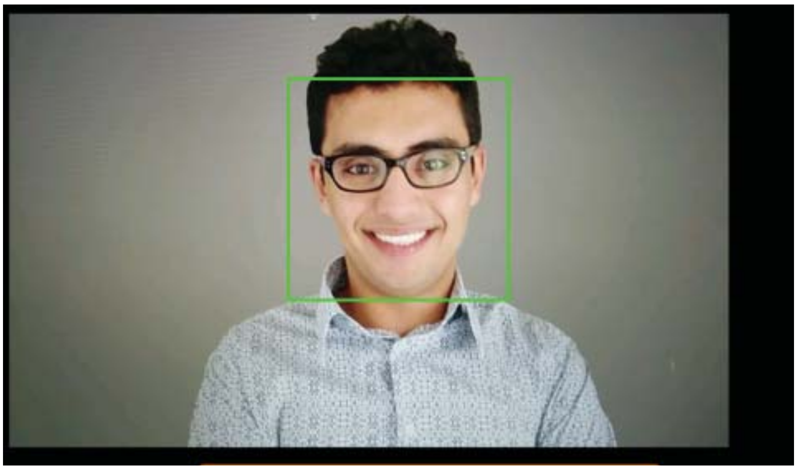

Figure 6 Face in the image was detected and surrounded by a green rectangle

Some of these techniques are:

Geometrical transformation and cropping of the image. This procedure includes resizing of the image and rotating the image as well as removing background. (Figure 7)

Figure 7 face image converted to gray scale and cropped

Histogram equalization and contrast stretching of the image. These processes standardize the brightness and contrast of the image (Figure 8). The definition of Histogram Equalization is to equalise the intensity distribution of an image or flattening the intensity distribution curve (refer to Figure 9 and Figure 10) [11].

Figure 8 the Histogram of the image equalized

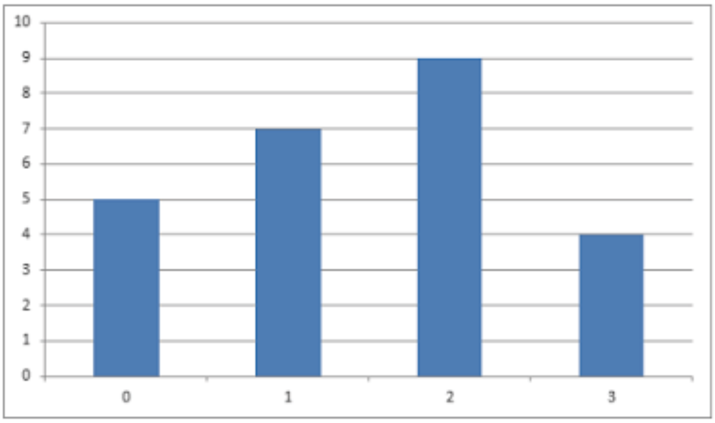

Figure 9 Image histogram [11]

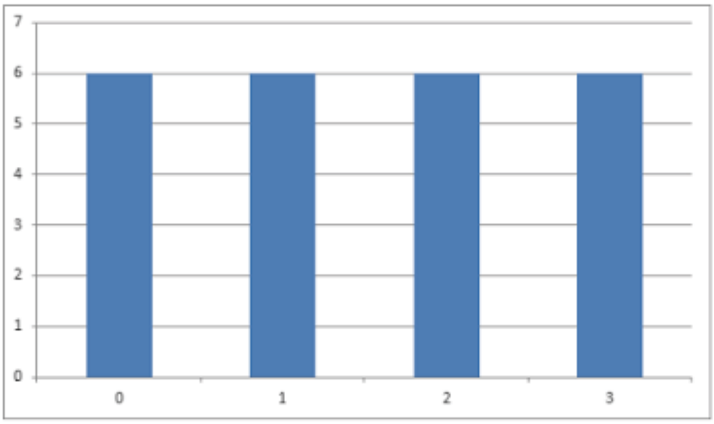

Figure 10 ideal Equalized histogram of the image of the above Figure 9 [11]

To illustrate the effect of histogram equalization further, Figure 11 shows an image with its histogram. The image histogram was not equalized.

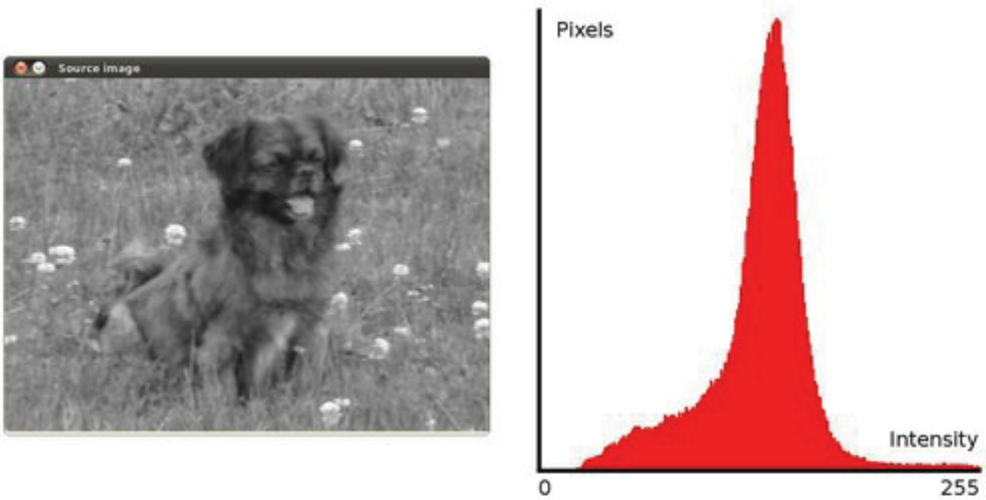

Figure 11 histogram was not equalized [11]

After applying the histogram equalization the results histogram and image in Figure 12.

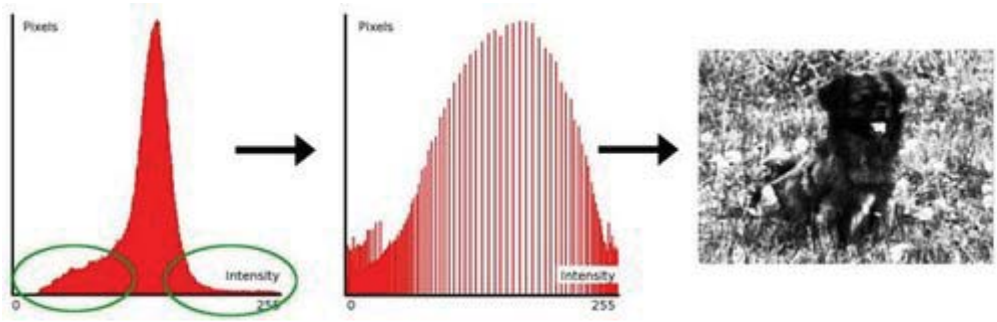

Figure 12 image after applying the histogram equalization [11]

Smoothing the face image before the process training or recognition is also important step in the face recognition case. Smoothing process eliminated the image noise by applying some filtering on the image such as Bilateral filter (Figure 13). There are various ways of smoothing an image. The one is used in Figure 13 is the bilateral filter to ensure that the edges of the face image are preserved. Bilateral filter is combination of two filters, Gaussian function of space that ensures only close pixels are considered for blurring. The other filter is Gaussian function of intensity difference that ensure only those pixels with similar intensity to central pixel is considered for blurring to preserve the edges of the image [11]. See Figure 14 after the applying bilateral filter.

Figure 13 Filter applied on the face

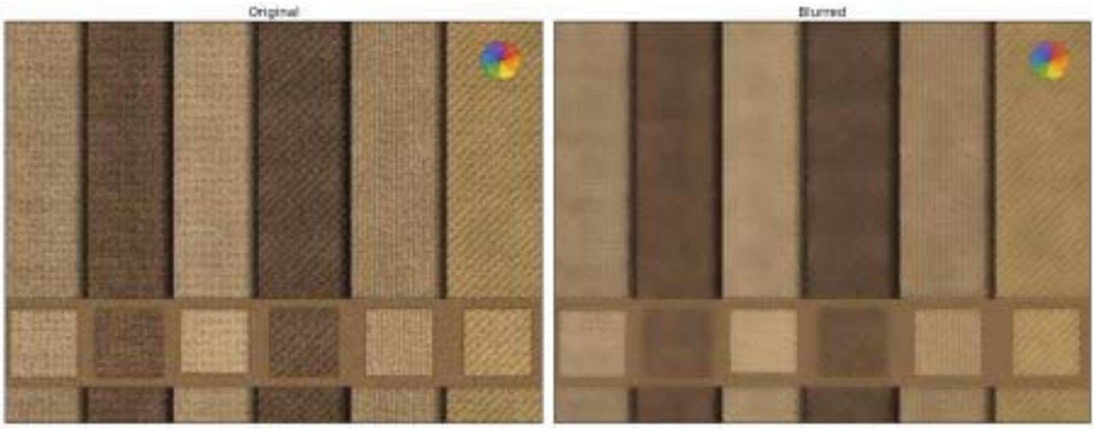

Figure 14 the right image is a result of applying the bilateral filter. Source [11]

The final step of preprocessing the image is applying an elliptical mask to the face. The elliptical mask removes some remaining background Figure 15. If the mask is not applied on the image the background of the image will be considered in the training of the image. As a result the recognition accuracy will drop significantly [8].

Figure 15 Elliptical mask applied on the image

2.3. Learning a Collection of Faces and Training the system

After preparing the face image for the training step, a collection of faces of the same person is required to train the system. There should be at least a one-second gap between each snap shot of the face. If the gap is less than that the face expression will not vary enough and the classifier will have fewer images to learn from [8]. This collection of images is called the Training Set.

After collecting enough images for the person (Training Set), we can now use one of many algorithms for training the system to learn the face. Like most algorithm in machine learning, training of the system must be completed first in order to identify and recognise a person. Numerous algorithms were developed over the last 20 years for the face recognition problem [14]. Some of the face recognition algorithms are: Fisherfaces (also referred to as Linear Discriminant Analysis), Hidden Markov Models (HMM) and Eigenfaces [14].

2.3.1. Fisherfaces (also referred to as Linear Discriminant Analysis)

Linear discriminant analysis (LDA) task is to find the projection in the space in which the difference between the different classes of objects is maximum. This requirement is stated as getting the most compact clusters corresponding to different classes, remote as far as possible. With the LDA it is possible to obtain a subspace of small dimension, in which images of clusters of individuals and "non-persons" overlap is minimal. Producing a classification in such space is easier to processes [15].

2.3.2. Hidden Markov Models

Hidden Markov Models (HMM) [16] is one of the ways to obtain the mathematical model (describing the properties of) some of the observed signal. The stationary Markov model (SMM) belongs to a class of stochastic models. Stochastic models try to describe the statistical properties of the signal only, without having information about its specific properties. The basis of stochastic models is necessary assumption that some parametric random process can describe the signal and that the process parameters can be estimated with sufficient accuracy to some certain way. The mood of the conditional Markov model (CMM) can be considered as the source of a random signal with well-defined characteristics. For tuned SMM it is possible to calculate the probability of the generation of this model of the test signal. The annex to the recognition of the problem by providing a feature vector object in the form of a signal (a set of consecutive observations), it is possible to simulate a class of objects with the help of SMM. Probability accessories test object class specified SMM is assessed as likely to generate a signal corresponding to its feature vector. Training CMM is to modify its parameters in order to maximize the probability of generating signals corresponding to the training [16].

To apply the CMM to the face recognition case, it is necessary to determine the way in which the face image is converted into a signal (a set of consecutive observations). The face can be naturally divided into several horizontal areas: forehead, eyes, mouth and chin. A person may be presented as a signal in which these areas in a particular order (usually from top to bottom, left to right) are transmitted. Thus, the face image is represented as a vector sequence of observations (each of the vectors is a horizontal band of pixels persons) which during training and recognition serially transmitted random process, simulating CMM [16].

2.3.3. Eigenfaces

Eigenfaces uses the Principal Component Analysis (PCA). Eigenfaces was developed by Turk and Pentland in 1991 [17]. The Eigenfaces algorithm is considered to be the first face recognition technique and most used one for many face recognition system baseline [8]. PCA method was "derived from Karhunen-Loeve's transformation. Given an s-dimensional vector representation of each face in a training set of images, Principal Component Analysis (PCA) tends to find a t-dimensional subspace whose basis vectors correspond to the maximum variance direction in the original image space. This new subspace is normally lower dimensional (t<<s). If the image elements are considered as random variables, the PCA basis vectors are defined as eigenvectors of the scatter matrix." [14].

The idea of the Eigenfaces was to represent each face image (from the previous step) into vectors using the image pixel value "a low-dimensional space" [17]. For example, if we have M images of a face and each image is N x N pixels. The matrices below Table 1 represent images of a face from the previous step.

Table 1 face two pixels P = pixel

P1	P2	P3	……	PN
PN+1	PN+2	…	…	…
…	…	…	…	…
…	…	…	PN^2-1	PN^2

The vectors of the images should look like this:

$$face_1 = vector_1(p1, \quad p2, \quad p3, \dots, pN^2)$$

$$face_2 = vector_2\ (p1, \quad p2, \quad p3, \dots, pN^2)$$

$$face_{\dots} = vector_{\dots}\ (p1, \quad p2, \quad p3, \dots, pN^2)$$

$$face_M = vector_M\ (p1, \quad p2, \quad p3, \dots, pN^2)$$

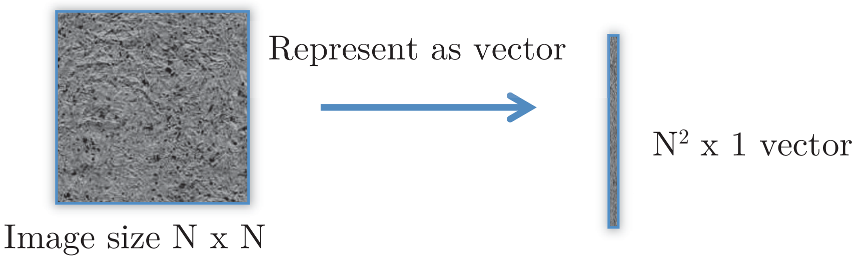

Figure 16 convert an image from 2D to 1D

After converting the images into vectors Figure 16, the average of each face vector is calculated then added to a new vector.

$$Average[i] = \frac{1}{C}\sum_{j=0}^{C} V[j][i]$$

Equation 1

Where C is the number of vector image and V is an array of vectors. For example (V [2][1]) represent pixel 1 in *face2* vector. See Equation 1 above.

The Average Vector or also known as the average face Equation 2:

$$AvgVector(Avrage[1], \ldots Avrage[N^2])$$

Equation 2

The next step is to calculate matrix correspondent to each image by subtracting the vector of each image from the Average Vector. This step is to normalize the image in order to remove the common features of the faces and keep only unique ones (Equation 3). Figure 17 shows the resulting images.

$$\Phi_1 = vector_1 - AvgVector$$

$$\Phi_2 = vector_2 - AvgVector$$

$$\Phi_{\ldots} = vector_{\ldots} - AvgVector$$

$$\Phi_M = vector_M - AvgVector$$

Equation 3

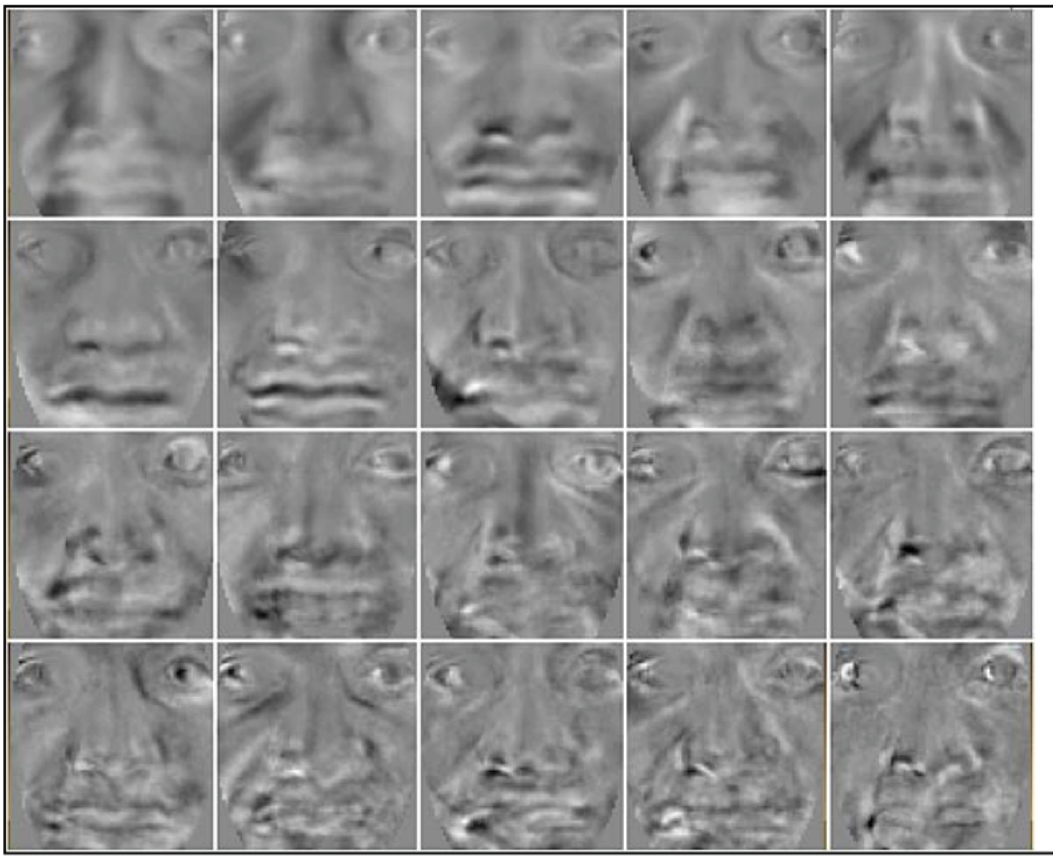

Figure 17 the resulting vector from subtracting the average from every image [8]

After normalizing the image, we can now compute the covariance matrix Equation 4,

$$C = \frac{1}{M}\sum_{n=1}^{M} \Phi_n \, \Phi_n^T = AA^T \quad (N^2 \; x \; N^2 \,) matrix$$

Equation 4

Where A = $[\Phi_1, \Phi_2, \Phi_3 \dots \Phi_M]$ (N^2 x M matrix).

Now we can calculate the eigenvectors of AA^T however the matrix is very large and it will require a great deal of processing power to compute. Therefore, the matrix $A^T A$ is used instead which is (M x M) size matrix.

Let eigenvectors of $AA^T = u_i$ and eigenvectors of $A^T A = v_i$

So we can compute the eigenvectors Equation 5 v_i of $A^T A$

$$A^T Av_i = \mu_i . v_i$$

Equation 5

The relationship between u_i and v_i can be found as follows Equation 6:

$$A^T Av_i = \mu_i . v_i$$

$$AA^T Av_i = \mu_i . A v_i$$

$$CAv_i = \mu_i . A v_i \quad OR \quad Cu_i = \mu_i . u_i$$

Equation 6

Where $u_i = A . v_i$

Hence AA^T and $A^T A$ have the same eigenvalues, the eigenvectors are related as the following Equation 7 [17].

$$u_i = A \times v_i$$

Equation 7

So that we can calculate the highest eigenvector of AA^T from u_i A x v_i.

Finally we keep only the largest K eigenvectors. K is the result of the highest eigenvalues [17].

The mean is subtracted from each face Φ(Equation 8) and (Equation 9) in the training and the results can be implemented as a linear combination of the largest eigenvectors [17].

$$\Phi_i - \text{mean} = \sum_{j=1}^{K} w_j \ u_j$$

Equation 8

$$w_j = u_j^T \Phi_i$$

Equation 9

Where u_j is the Eigenfaces.

2.4. Face Recognition

Now the algorithm is ready to identify a person from their face images. The recognition of the faces has less processing than the training. Firstly after detecting a face in the frame we converted it into a grey scale image then transform the face into a vector representation. After that we use the Euclidean Distance to determine the minimum distance between the new vector face and the eigenvectors stored. A threshold for the maximum allowable distance between the input face and the training set must be defined. If the distance is larger than the pre-defined threshold we can classify the input face as an unknown face. Otherwise, the face is associated with one of the stored faces [17]. Once the face is known the identity of the person in the image needed to be determined. This can be done by reconstructing the closest vectors from the training set and comparing the input face to it. With some confidence value the function is returned, we then can determine the person's identity. [8]

2.5. Eye Blinking Detection

Detecting the eyes in a person's face is an essential step to implement a blinking detection or blinking count. Detecting and extracting the eyes region would limit the number of pixels which require processing in each frame. Therefore, extracting the eye regions from a person's face for detecting eye blinking is vital step in the blinking detection problem. There are few methods in the literature which will be discussed in this section.

2.5.1. Optical and normal flow

The optical flow is used to determine the eyelid movements within a region extracted by a simple motion detector. The motion detector is based on the difference between two frames after applying the threshold. Once the optical flow fields are calculated we extract the vertical and horizontal vectors (Figure 18). If the downward motion is dominating, this indicates the eye blinked [18] (Figure 19). However according to Heishman and Duric in their paper , using image flow to detect eye blinks in color videos, most of the Optical flow methods are usually slow and cannot be used in real-time application [19]. Heishman and Duric added "Human motions are articulated and non-rigid, thus global assumptions about motion cannot usually be made" [19]. Instead (Heishman & Duric, 2007) have used the normal flow in their paper.

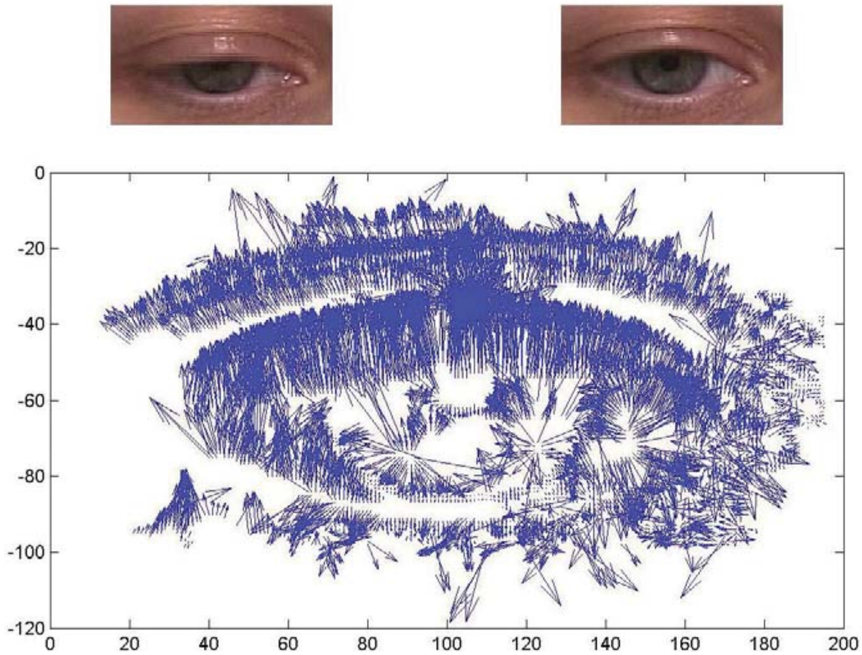

Figure 18 example of flow fields showing eye open [19].

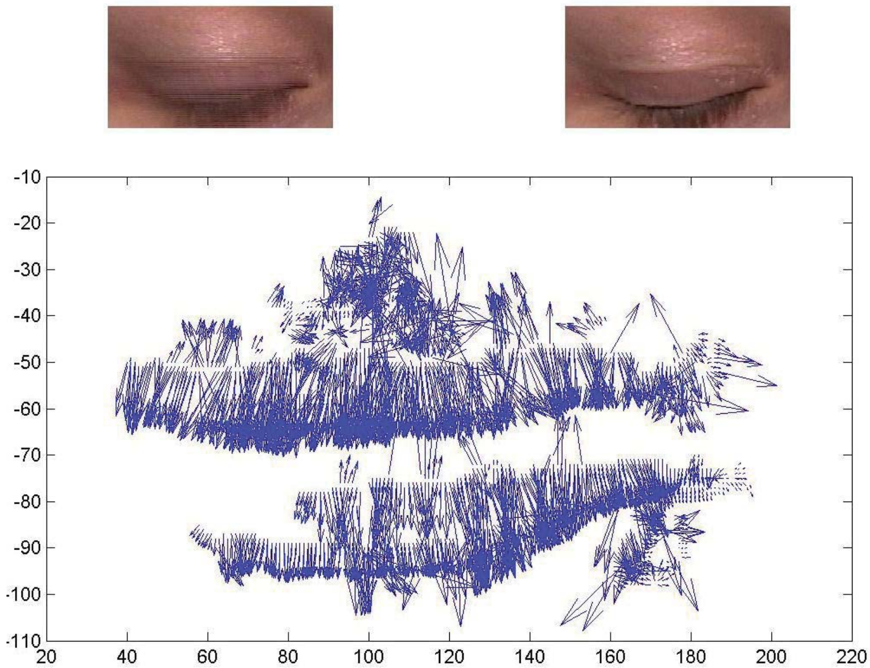

Figure 19 dominating field motion is downward i.e. the eye is blinked [19].

2.5.2. Neural network

The human biology is a highly complex system. The information processing in a human at lightning fast speeds is a mystery upon which countless people try to understand and explain it. The nervous system of humans is made up of neurons that are interconnected.

The amazing thing is that they make use of electrochemical signals to relay information from one part of the body to another. The discovery of this genius in human neural cells gave room for the same type of system to be used in computational models.

Hence the Artificial Neural Network as a model receives its inspiration from the human neurological system and the biological neural network encompassed within every being [20].

This model uses a simulation to enforce the functionalities and structure of a human system. The model works in a threefold system; which consists of multiplication, followed by summation and activation respectively.

The model works in perfect unison, strictly following a preset patterned sequence of affairs. Upon entrance, the neuron inputs are multiplied after being weighted. These weighted inputs are then summed up in the mid-section and transferred to the activation function.

This is where the response is activated and the output is yielded. Jason Brownlee explains in his book that there are two major categories of neural networks. They include feed –forward networks and recurrent networks.

In the feed-forward network, a layer of cells and neurons is involved, which propagates the signal forward, from where they are received through a network structure. These signals are pushed on to the other side to the point from where they are read as output.

The recurrent networks on the other hand are networks where the structures are interlinked with each other and the entire system is interconnected. These neural networks are complex structures, which are very difficult to integrate. But once they are installed, they are really quick in their operations.

In the modern age, neural networks have countless applications. They have the capability of changing the world as we know it, with the rapidly increased demand for faster systems.

According to the paper published by Prof. Leslie Smith [21], Artificial Neural Network enabled human and computers to compute and solve numerous issues, which would have been otherwise impossible to solve by using traditional computers [21]. The applications of this method include, on the forefront, the analysis of investment. They can help in predicting changes in values and trends of stock prices, currencies through complex calculations that would otherwise take a lot of time and energy [21].

In the same way, this technique is used to detect the human eye blinking. As the eye has two states, either open or close, in the computer representation that is zero or one. By extracting the features of the eye then train the network to produce the correct results, in this case open eyes or closed eyes. Multilayer feed-forward network architecture, which allow signal to travel in one direction only, is a combination of layers. These layers are inputs, hidden and outputs. Between each node there is an initial weight that has a random value, often initialized to 0.1 then it is adjusted by computing element in each node. Bias nodes are also attached to each hidden node and the output node. Figure 21 shows two hidden layers, plus input and out layers. Bias units or threshold nodes are required for universal approximation. Without bias units, the feed-forward network would always assign 0 output to 0 input. Therefore, it would be impossible to approximate functions, which assign nonzero output to zero input.

One of the novel idea in this work is the design of the customized “eye features” that actually work in practice for detecting eye blinks. The “eye features” used to train the network are calculated based on predefined sub-regions in the eye that were empirically found to be effective for the task. Four Haar-like features are extracted from these pre-defined sub-regions of the eye. These values are then fed to the network for training, to detect eye blinks.

The following figures (Figure 20) depict the different sub-regions used, and the equivalent haar-like features extracted out of them.

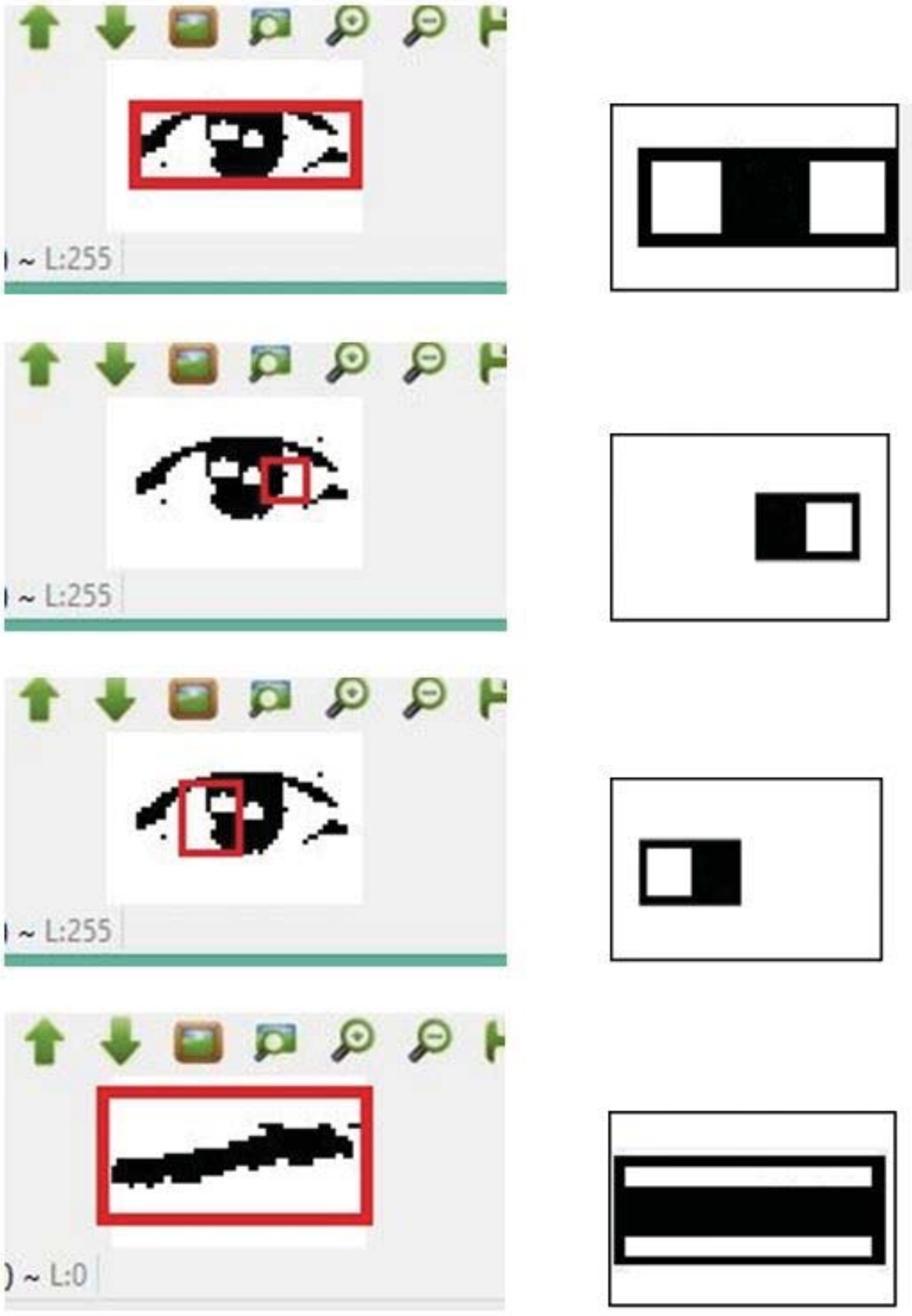

Figure 20 eye features extracted from pre-defined sub-regions of the eye

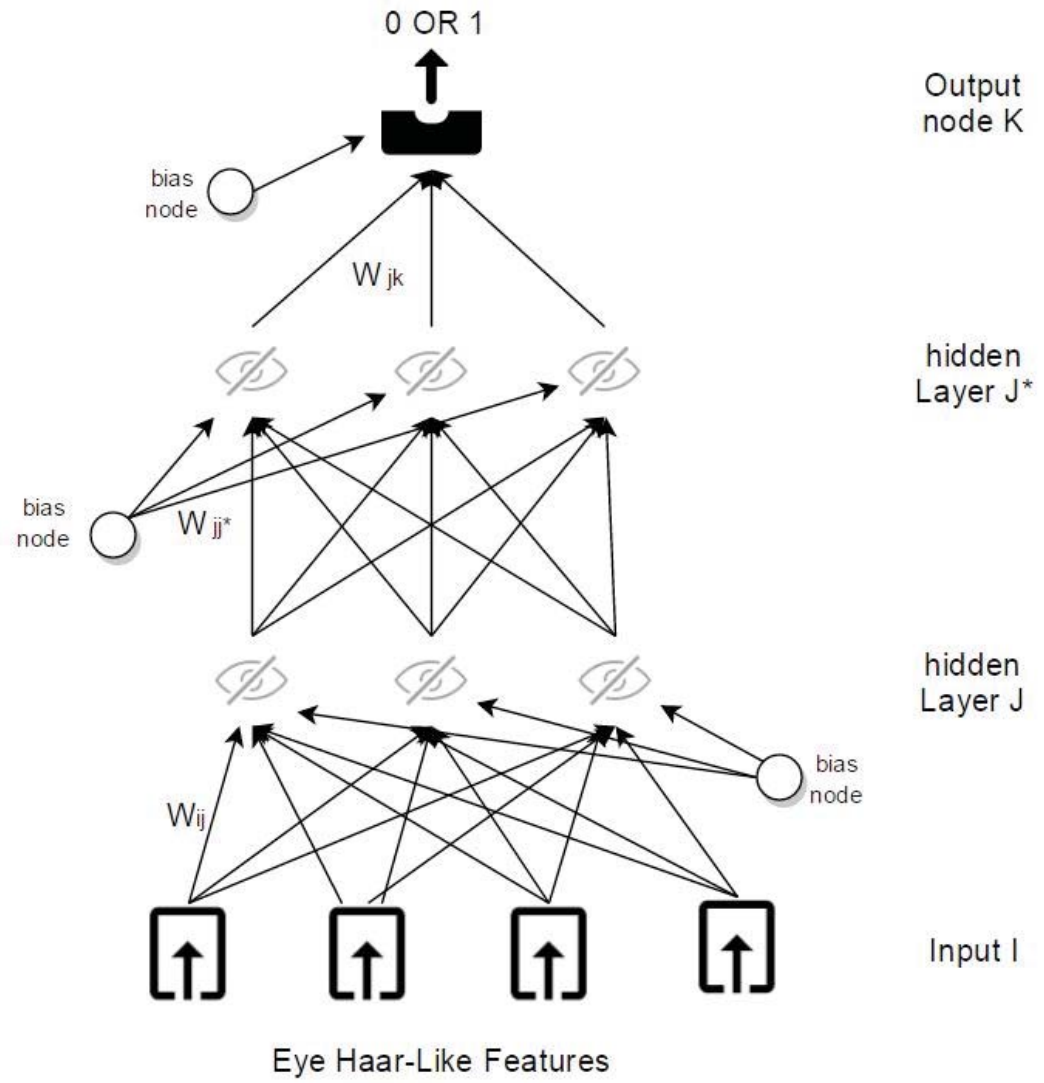

Figure 21 A Multilayer Feed-forward Network

2.5.2.1. The back-propagation Training

To train the network, the following steps [22], [23], [24] were applied iteratively to minimise the error over training set:

1. Put one of the training patterns to be learned on the input units.

Find the values for the hidden unit and output unit using the initial weights assigned when initialising the network.

2. Find out how large the error is on the output unit (error signal)

$$\delta_k = (t_k - o_k).\, f'(net_k)$$

Where t_k is the target output, O_k is the actual output and $\boldsymbol{f'}$ is the derivative of the activation function:

$$f(net_k) = \frac{1}{1 + exp^{-net_k}}$$

3. Use one of the back-propagation formulas to adjust the weights leading into the output unit.

$$W_{hk(New)} = W_{hk(Old)} + \eta\delta_k o_h$$

Where η is a small positive constant called Learning Rate and O_h is the signal going out from the hidden layer to the output.

4. The following formula is used to find out errors for the hidden layer unit.

$$\delta_h = f'(net_h)\sum_k \delta_k\, W_{kh}$$

5. Adjust the weights leading into the hidden layer unit using the following formula.

$$W_{ih(New)} = W_{ih(Old)} + \eta\delta_h o_i$$

Where η is small positve constant called Learning Rate

6. Repeat steps 1 through 6 for each iteration.

2.6. Smile Detection

There are a large number of research publications on facial expression but only a few are focused on smile detection [25]. “A Smile may involve up to seven different pairs of facial muscles: Zygomatic Major, Zygomatic Minor,Risorious, Buccinator, Levator Labii Superioris, Levator Anguli Oris, and Orbicularis Oculi. While modulation of these muscles can generate a very wide variety of smiles, in practice, the prototypical smile only needs to involve the Zygomatic Major” [26]. Shan in this article [25] proposed the intensity differences between pixels methods that apply on a grayscale face images. The method used AdaBoost to choose and combine the intensity difference of the pixel to create a strong classifier to detect a smile on a face [25]. Smile Detection by boosting pixel differences was compared to the Gabor filters and local binary pattern (LBP) by [25]. The accuracy of Gabor filters is higher however the advantage of this method was the speed of which the smile was detected. The detection of the smile is real time which is required for many nowadays commercial applications (Table 2).

Table 2 accuracy of the 3 different approaches for the smile detection [25]

Approach / Feature	Accuracy (%)
Gabor	89.55±0.63
LBP	87.10±0.76
Raw Pixel Values	80.38±1.04

2.7. Test Data

A preview of the dataset samples used for testing static images is showing in Figure 22 The ORL face database Figure 22. The dataset used are FRETE dataset [28], CMU/VASC MIT image database [29] and Closed Eyes in the Wild (CEW) datasets [30]. The eye-blinking and smile detection tests were carried out on volunteers and myself by recording a short video of the examinees and myself and analyzed the videos by applying the eye blinking and smile detection algorithm accordingly.

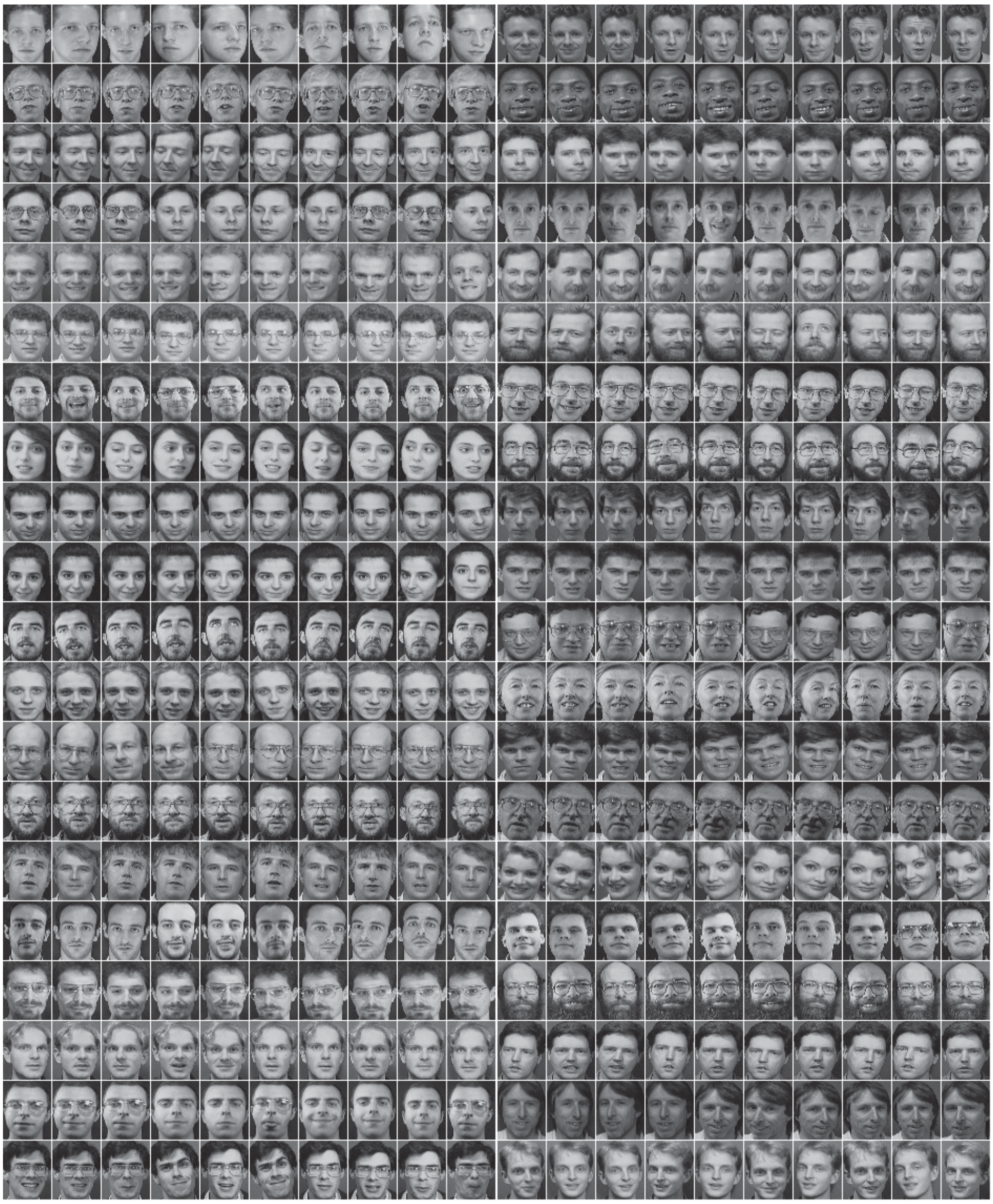

Figure 22 The ORL face database [27]

2.8. OpenCV Library

OpenCV (Open Source Computer Vision) is a library that was designed by Intel to process images. Because it was meant for image processing it is loaded with many of algorithms and functions to help computer scientists solve vision related problems. OpenCV offers applications that will help to train cascade classifier. In addition, OpenCV libraries provide three face recognition algorithms built-in [11].

2.9. QT Creator

Qt Creator is an integrated development environment which is a cross-platform IDE i.e. Linux, Mac Os and Windows. Qt Creator is part of the software development kit for the Qt GUI application development framework. Qt Creator application uses C++ compiler from the GNU Compiler Collection on Linux [31].

2.10. Summary of the Literature review

After examining and researching the literature, we found that the most reliable and accurate method for the face detection is Viola and Jones method for the face detection. The accuracy of Viola and Jones method is over 95% of the time will detect the face correctly. Furthermore, the speed of the detection is in real-time. Similarly with the eye and mouth detection, we used the object detection method by Viola and Jones but with different classifier for the eye and the mouth respectively.

The number of research in the literature for the eye blinking detection was very limited. Therefore, we have proposed a solution for the eye blinking based on the object detection method by Viola and Jones. In addition, we have examined the neural network capabilities for detecting the eye. A novel idea for extracting the eye features using pre-define region of the eye image. Those features can be then fed to the neural network to determine whether the eye is opened or closed.

The smile detection method proposed by Shan [25] was not the most accurate method, however, the speed that the method detects a smile on the face was in real-time. The detection of the smile in real-time is essential for the system to function as expected.

3. Methodology

3.1. Face detection

The method used for the face detection is the Local Binary Pattern (LBP). “The basic idea of the LBP-based face detector is similar to the Haar-based one, but it uses histograms of pixel intensity comparisons, such as edges, corners, and flat regions” [8]. Haar-like based classifier was slower to detect a face in the frame, Table 3 and Table 4 shows the average for Haar-like and LBP classifiers with and without a face in the frame to be processed. (Figure 23)

Table 3 Comparing Haar-cascade vs. LBP with face in the image

Haar Fps	LBP Fps	Status
2.94	13.59	Face in frame
2.97	13.56	Face in frame
2.94	13.71	Face in frame
2.94	13.71	Face in frame
2.97	13.59	Face in frame
2.94	13.71	Face in frame
2.95	**13.71**	**Average**

Table 4 Comparing Haar-cascade vs. LBP with No face in the image

Haar Fps	LBP Fps	Status
4.29	17.22	No Face in frame
3.60	17.25	No Face in frame
3.60	15.39	No Face in frame
4.2	15.39	No Face in frame
3.96	17.22	No Face in frame
3.6	15.39	No Face in frame
3.875	**16.31**	**Average**

Table 2 LBP vs Haar Frame per Second

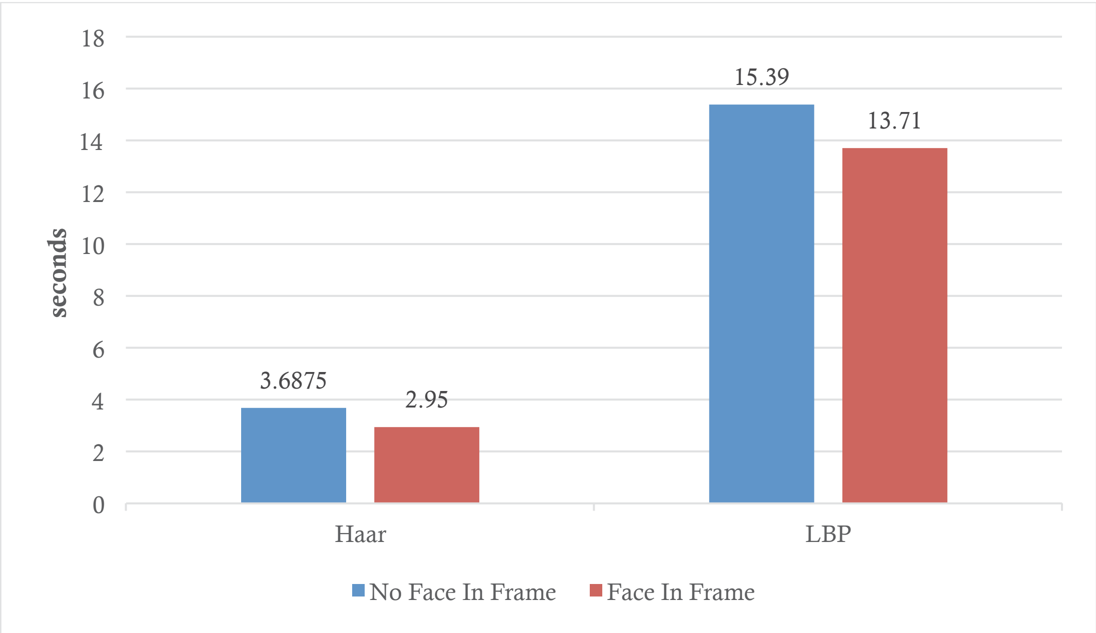

Figure 23 Comparing frame rate per second Haar-cascade vs. LBP with and without face in the image

Training the classifier on around 500 faces takes about a day to complete using the Local Binary Pattern (LBP). The resulted classifier did not detect faces accurately due to the lack of samples. Therefore, I obtained a trained classifier from the OpenCV repository [11].

When the camera starts capturing frames, the search for face happens on every frame captured. When there is a face detected, a green rectangle is drawn around the face (Figure 24)

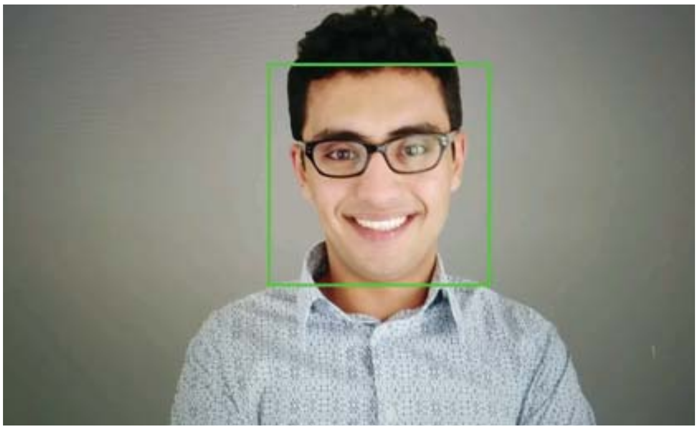

Figure 24 A face detected on the frame

3.2. Face processing

First of all the image was converted to gray scale. After that the face image is resized to 128 x 128 pixels to minimize the computational processes on the image. The next step was equalizing the histogram of the image. Then we have applied bilateral smoothing filter on the image to reduce the noise of the image because the histogram equalizer introduces significant amount of noise to the image. The final step for pre-processing the image was to apply the elliptic mask to eliminate the background of the image. (Figure 25)

Figure 25 the processing steps from the detected to final image

3.3. Face Images Acquisition

In the literature, the recommended gap between every snapshot is at least one second [8]. Therefore, a sleep function is used between every image was collected. In addition, to the 500 millisecond pause, processing the image require approximately another half a second. That adds up to one second between every image taken of the face during the training. The label of the face being trained shown in Figure 26 as text field is added as a name of the image along with a number at the end. For example, Sam-1.jpg, Sam-2.jpg etc.

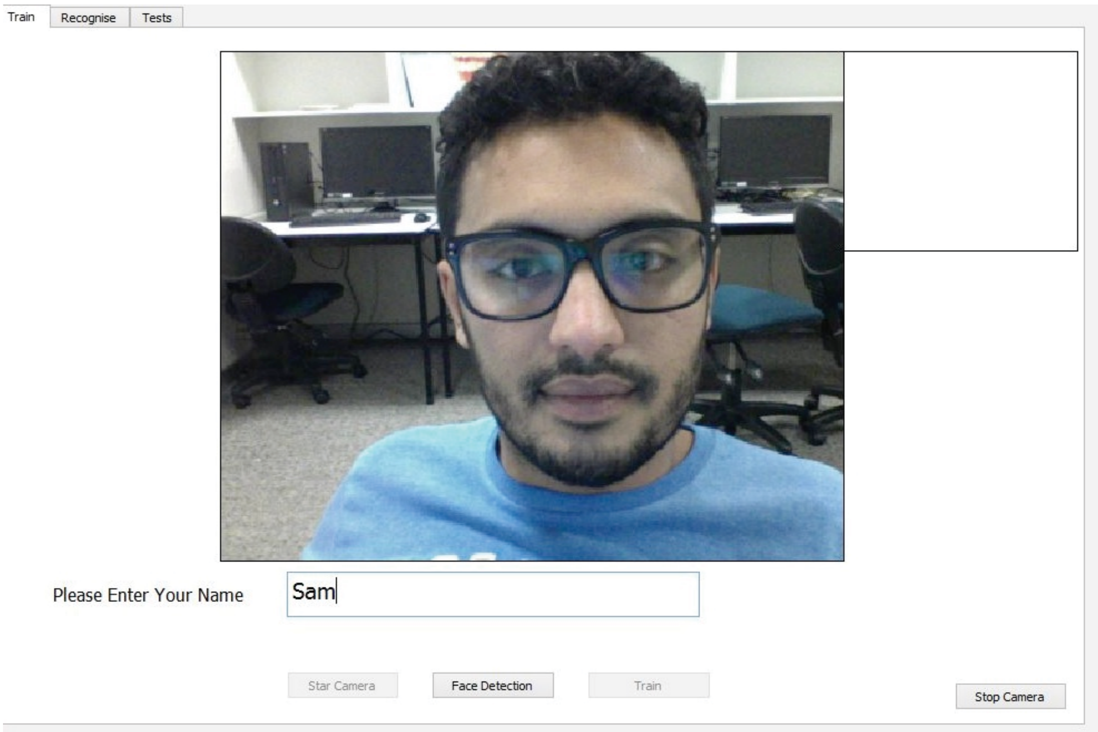

Figure 26 the Label of the face is added in order to train the classifier

The images of the face are added into an array of images, after pre-processing each image. Because the gap between every image collected is approximately one second, ten images of the faces were sufficient to obtain an accurate face recognizer. The face expression, the lighting, or the angle of the snapshot will vary between the face images, therefore the classifier is trained with only 10 images. This can be extended, however the improvement of the accuracy of recognising the face is not significant (Figure 27). Moreover, the performance of training new faces process will degrade. For example, if we have 30 images to train, the training time will take 30 seconds to complete. On the other hand, when we have 10 images to train, the process will complete in approximately 10 seconds.

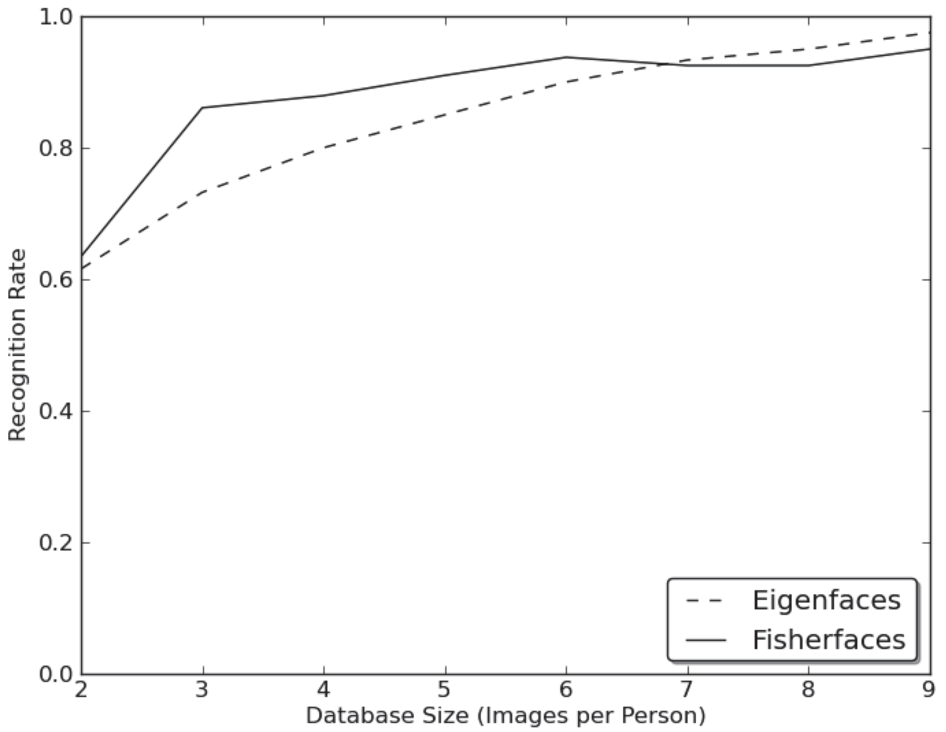

Figure 27 accuracy of recognising between Eigenfaces and Fisherfaces with 9 images [11]

3.4. Learning faces

When the images of the face are collected and preprocessed, then we can apply the Eigenfaces algorithm to train the system.

Training the function parameters are a vector of faces that was collected and preprocessed and ready for training and the name of the person to label the face. We use OpenCV FaceRecognizer[§] class to create the Eigenfaces model and train the face. The FaceRecognizer Train method accepts a vector face and corresponding vector labels. The labels must be integers, so a mapping between the name and an

§ An abstract base class which provides a unified access to all face recognition algorithms in OpenCV integer for every person takes place here. The integers are stored in comma-separated values (csv) file as the following:

```
Name, label
Sam, 1
Mishel, 2
Oxana, 3
```

Then we pass the vector faces and the integer label to the Train method in the FaceRecognizer class. After the training is completed, we then use the method setLabelInfo** to set to associate the trained label to a given name. An XML file is generated containing information about the face. This is the trained classifier that we save locally to use for the recognition.

3.5. Recognizing Face

Now the system is trained, so that we can use the recognition module. In the recognize module, a face detection is also implemented to detect a face first. So for every frame captured by the camera we try to detect a face. After the face is detected, we pass the image containing a face to the proceeding function. We pre-process the image in preparation for it to go through the face recognition methods. The image is converted to gray scale initially, then it is resized to 128x128 pixels. After that we apply histogram equalization to improve the contrast of the image and eliminate the lighting variant in the face. Applying a smoothing filter before we apply the elliptic mask on the face. Finally we pass the face image to the recognition function to identify the person identity/name if it exist in the previously trained classifier. (Figure 28)

** A function in OpenCV FaceRecognizer class to set the label in the training model

Figure 28 face image passed to the recognition function

In the recognition function we firstly load the trained classifier from a file using Load method in the FaceRecognizer. Alter the model has been loaded successfully we use the FaceRecognizer predict method which takes the image of the face e.g. Figure 28 and a float type value as parameters. The float is a reference of the prediction confidence. The lower the value of prediction confidence, the more accurate prediction we can get i.e. if we use the exact same image to train and predict, the prediction confidence will be 0. The predict function returns a label type integer. Every label has been associated with a name in the classifier so we use the getLabelInfo[††] in the FaceRecognizer passing the returned label value from Predict function to determine the name of the person.

Once the person has been identified, we then start the detecting eye, mouth within that face.

[††] A function in OpenCV FaceRecognizer class to get the label in the trained model

3.6. Eye Detection

In order to process the movement of the eye (Blink) we need to detect the eye first. For the blinking detection, the decision was made to use only one eye to detect the blinking. Therefore, we are only detection one eye on the face. The method used to detect the eye is Viola and Jones object detection method. After testing some of the eye detection on the literature, Haar-like Cascade classifier for eyes was the most accurate. Therefore, Viola and Jones eye detection method was used and combined with the extraction of the eye region (Figure 29). The idea was to apply two masks over the face to cover it except the left eye.

Figure 29 mask apply to the face image to extract the eye region.

Mask A Point (0, 0) and Point (row/2, col)

Mask B Point (0, col/2), Point (row, col)

In addition, the face has the elliptic mask so the results showing in Figure 30.

Figure 30 eye region after applying the mask as well as the elliptic mask on the face

Just like detecting the face previously, however, the classifier is trained to detect human eye. The classifier detects open and close eye, this is very important to ensure the accuracy of the blinking detection in the next section. Once an eye has been detected, we extract the eye and pass it to the eye blinking detection function.

3.7. Eye Blinking Method

By using only the eye region in the image, we can perform a real time operation in a very small amount of time. The reason is that because we have a very small portion of the image to process, which is the eye region. In addition, we have used only one eye to detect the blink, this is because human naturally blink their two eyes at the same time. Therefore, using one eye is sufficient enough to detect the blinking Figure 30 and Figure 31 show the eye region and the eye to be processed. This is an effective method to reduce the workload on the process and it will not increase the chance of fault detection of the blinking.

Figure 31 extracting and processing one eye only

The method used and proposed as new method for detecting eye blinking is Viola and Jones object detection using Haar feature-based cascade classifiers. Firstly we acquire the image of closed eye from Closed Eyes In The Wild (CEW) dataset [30]. The dataset contains 1192 closed eye images for the right eye and 1192 for the left eye. Because we are using the left eye, the dataset chosen to train the classifier is the left eye. However, the features of the closed eye of both sides are very similar (Table 5) and (Table 6). After that we train the 700 closed eye images as a positive image and we use the face images from FERET dataset [28] as negative for the training. The negative image does not contain any face with closed eye. Training the classifier required few hours to complete with 50 stages. The training produced a closed left-eye classifier that can detect only closed eye. We then used the remaining 492 out of 1192 closed eye images as the test set to verify our classifier. The results were impressive with over 91% correctly classified as closed.

Table 5 results on testing the closed eye detection on the left eye

True Detection	**False detection**	**Missed**
91.87% (452/492)	0	40

After that we used the closed right eye images to test the classifier which resulted in 86.71 % of correctly classified eye as closed.

Table 6 results on testing the closed eye detection on the right eye

True Detection	**False detection**	**Missed**
86.71% (979/1192)	0	213

This approach enabled us to detect the eye blinking with very high accuracy. After the face has been identified and the eye region was extracted, we then used the proposed method to detect the blinking. The blinking detection function takes an eye image as parameter, which then detects whether the eye opened or closed. If the eye is open the function terminates here. Otherwise, if the eye is closed, the function returns "eye closed". Moreover, to avoid multiple blinking detection in one blink we keep track of the last frame that has been processed, if last frame had a closed eye on it, we do not detect a blink but return still blinking. The method used to track the previous frame was a Boolean variable (true/false) is passed to the blinking detection function along with the eye image.

Once we know whether the eye is blinking or not, we then pass the same frame to the smile detection function to determine if a smile is also present in a given frame.

3.8. Smile Detection

For detecting the smile, we used the method proposed by [25] which is detecting the smile by Boosting pixel difference. The image passed to the smile detection function is the face image, which was detected in the face detection function in Section 3.5 Figure 28.

Firstly, we apply the mouth detection algorithm on the image to detect the mouth region. Once the mouth is detected, we use [25] approach to determine the intensity of the smile based on the relationship between two pixels' intensities as image features. The value of the intensity of the smile is between 0 and 1. The higher the value, the stronger the smile is. The threshold was set based on the experiment result, which is 75%. So if the intensity is greater than 75%, it will detect a smile.

To avoid multiple smile detection with one smile, we keep a record of the last frame that was processed and its intensity value. If the value is greater than 75% of the last frame, we do not detect as another smile but the person is still smiling.

3.9. Random Instructions

In this research, one of the main ideas is that it uses random instructions generated by the system. The number of instructions is set by the user to indicate the level of security. For example, a user that needs a moderate level of security should set up the system to generate from 3 – 5 random instructions. This setting can be adjusted to up to 9 instructions for higher level of security. Every instruction can either be a smile or an eye blink that the user must perform in order to authenticate via the system.

The random generator was implemented as a class in C++, which was named Random. The class Random generates a random number using the system time as the seed for the starting point of the pseudorandom algorithm [32]. Then, dividing the number that was generated by 2. If the remainder of the calculation is 0, then the action will be a smile. On the other hand, having a remainder from the division operation means the action will be an eye blink.

The class Random will also generate a random list of actions; the number of actions in the list is configurable by the user to determine level of security required. The class will store the list actions to compare it with the user real input later on. We compare the actions that were generated randomly in the Random class against the actions list that was generated using the user input by verifying that the order of actions from both list are identical. If any of the actions order is incorrect the list will be regenerated with a new set of instructions.

3.10. System Flowchart

The following system flowchart shows the integration of all the components of the system. This includes the face detection, face recognition, eye detection, and eye blinking detection and smile detection. In addition, it illustrates the flow of the random instructions used to verify the users' interactions. The actions in the flowchart are referring to an eye blink and a smile (Figure 32).

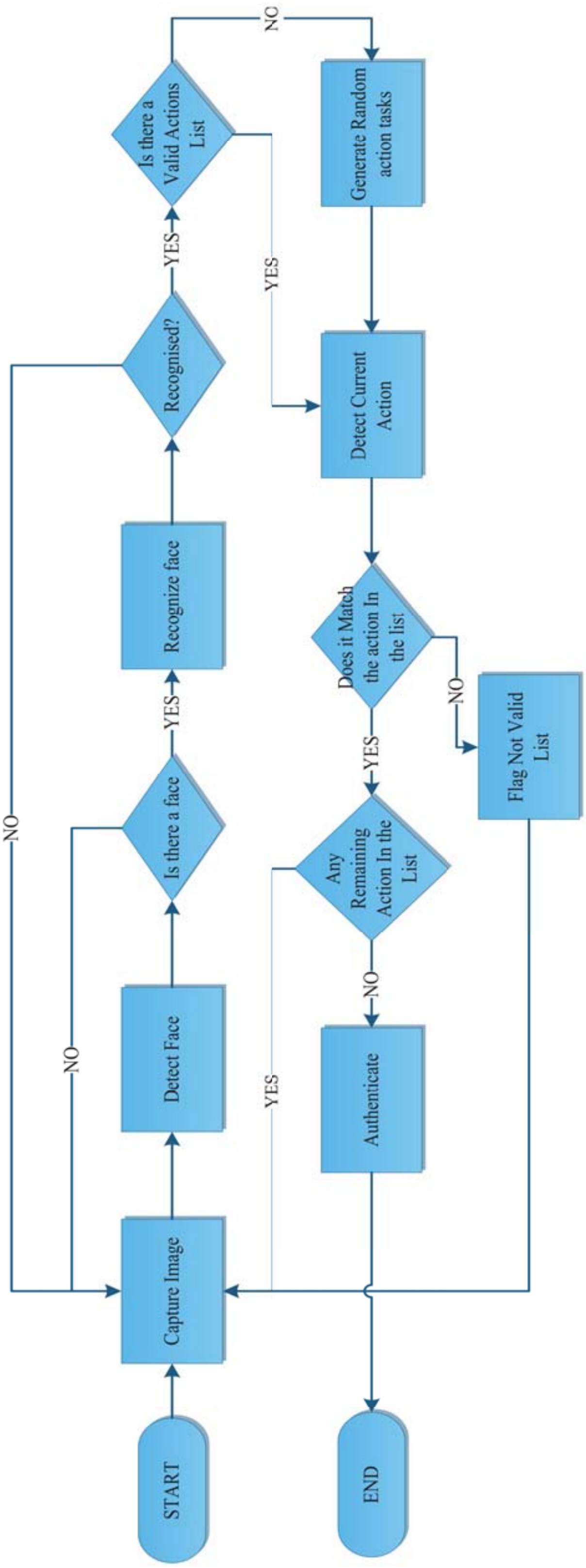

Figure 32 the full system flowchart

The flowchart illustrates the face recognition and training modules. As Figure 33 shows there are two paths, the first one is the training path, which will train the system using the face and create a classifier for that person's face. The second path is the recognition part that will detect the face and tries to match it to a known face in the system.

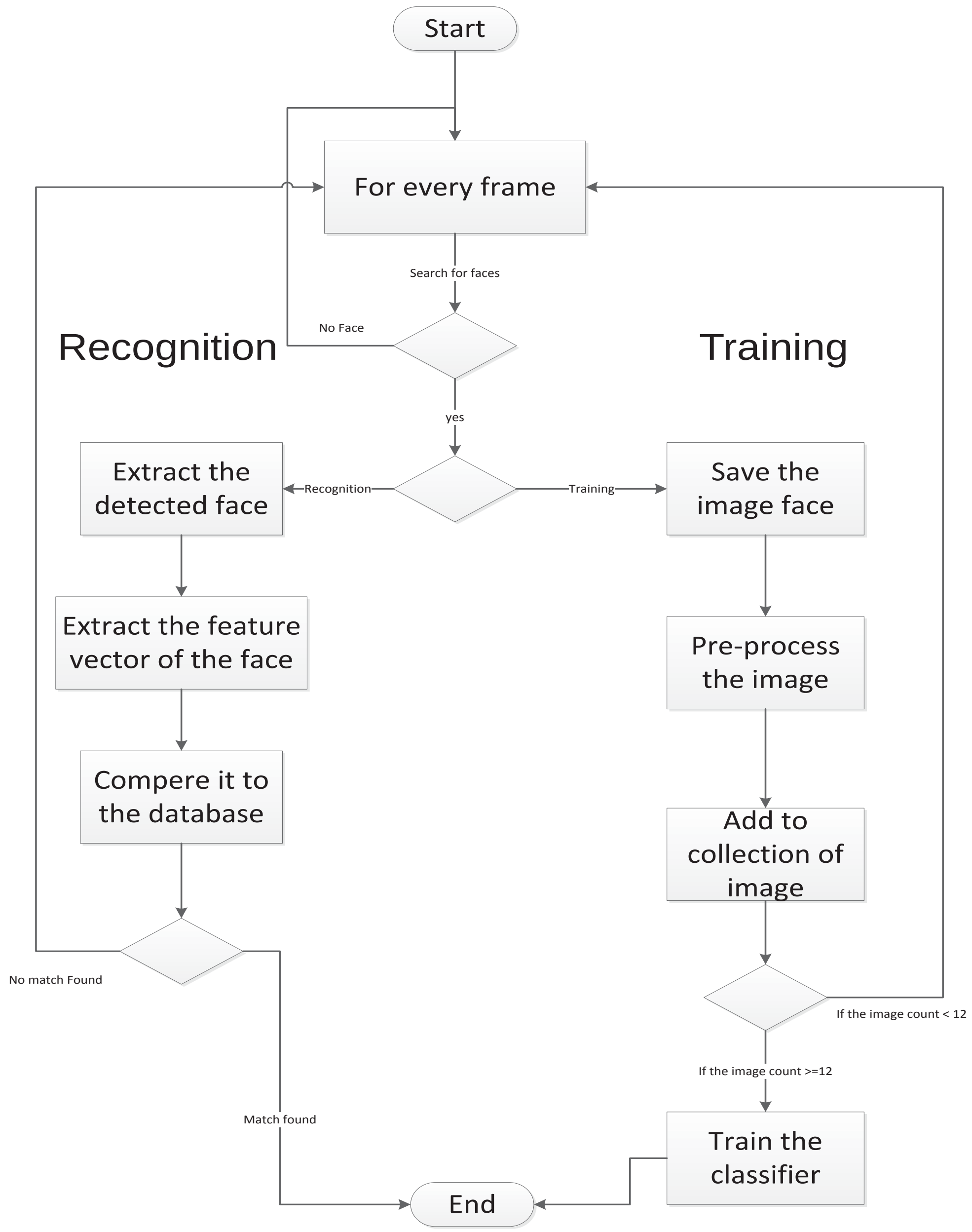

Figure 33 flow chart of the face recognition module

4. Work done and Outcome

Using QT Creator with the OpenCV SDK, I have created a simple application with two modules enabling the user to select either train a new face or recognize a known face (Figure 34).

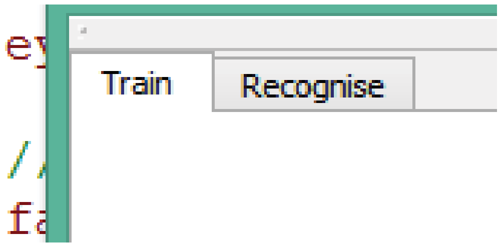

Figure 34 The application main modules

4.1. Training

'Start Camera' button will open the default camera then the 'Face detection' button will start the detection algorithm. This will only detect faces without any training (Figure 35).

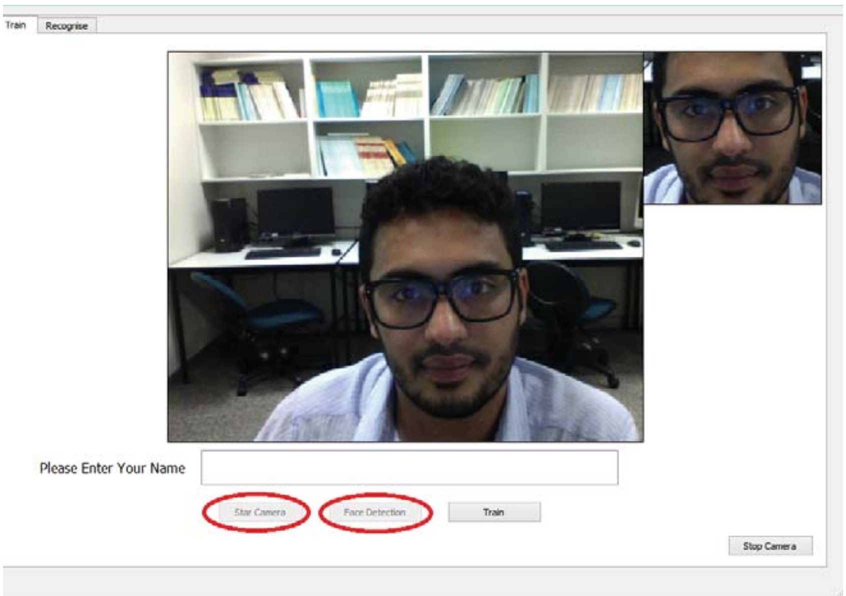

Figure 35 detecting faces without training in the training window

To ensure every face is associated with a label i.e. name of the person the Train Button will not be activated until the user input a name Figure 36. The training button will start the training algorithm. It will capture 10 images and start the training with a gap of 1000 milliseconds between each snapshot of a face (Figure 37). Also a progress bar of the acquisition of the image was added to the training module (Figure 38).

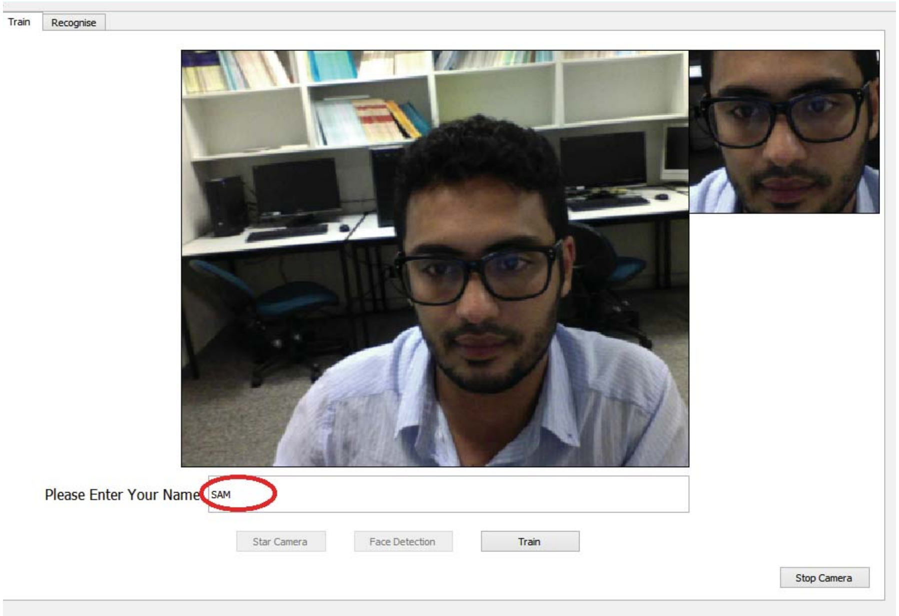

Figure 36 training will start after adding a name

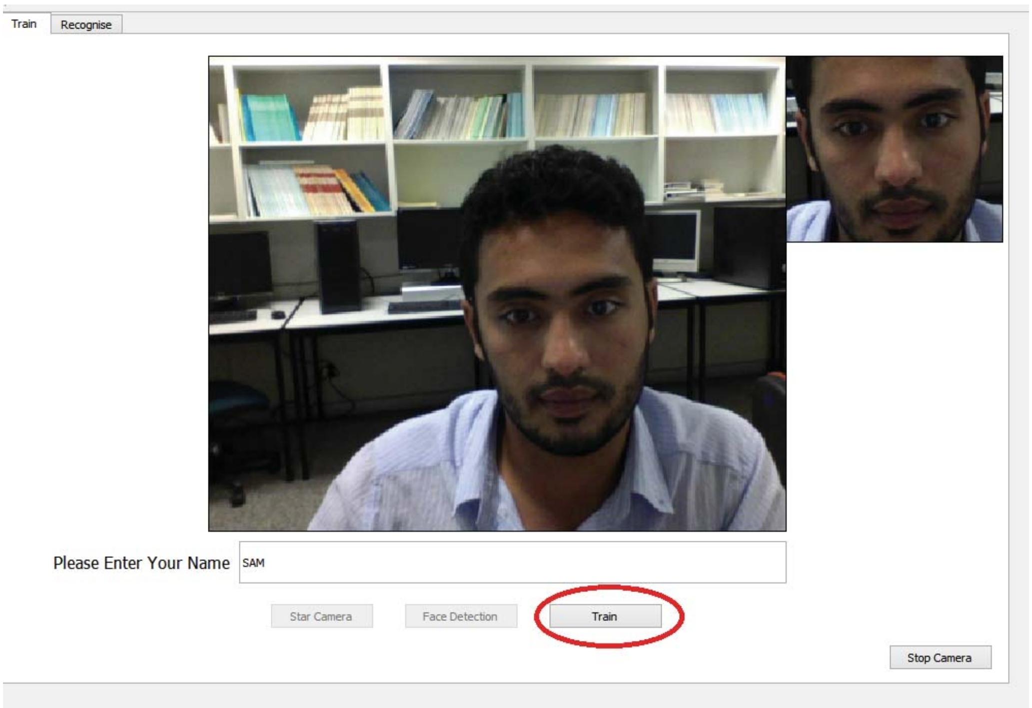

Figure 37 multiple snapshots will be capture per training

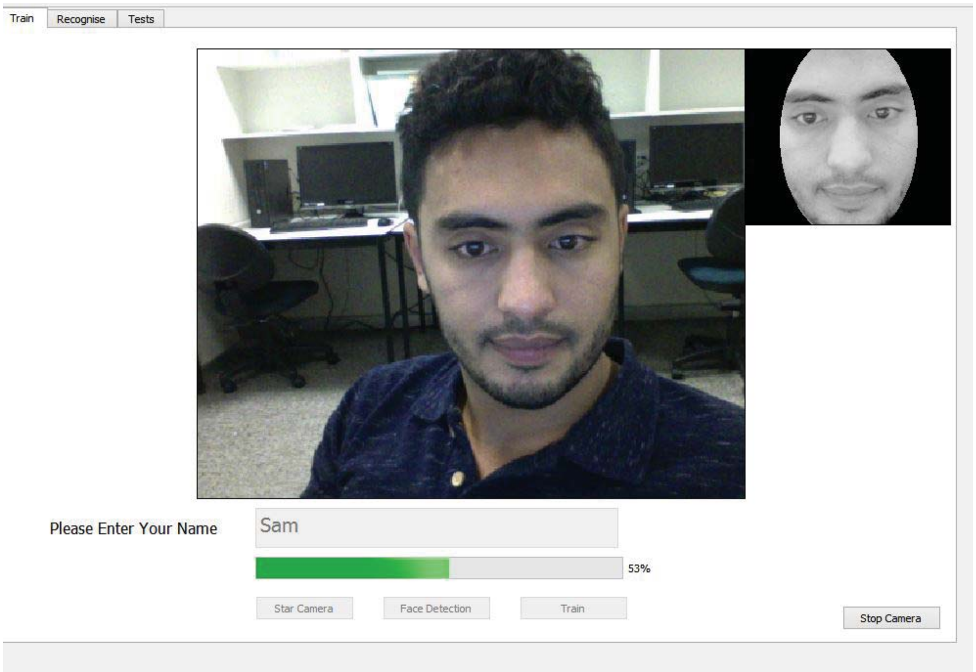

Figure 38 showing the progress bar and the actual image to be trained.

4.2. Recognise Module

In the recognise module, we can start the camera but the system will not be processing any image. However, the 'Face Detection' button will start detecting faces in the frames coming from the camera. Once a face is detected, the image face is passed to the predict function to check if there is any close match in the training set. If there is, it calculates the probability of the closest face found. If the probability is higher than a pre-defined value, it prints the welcoming message along with the instructions (Figure 39).

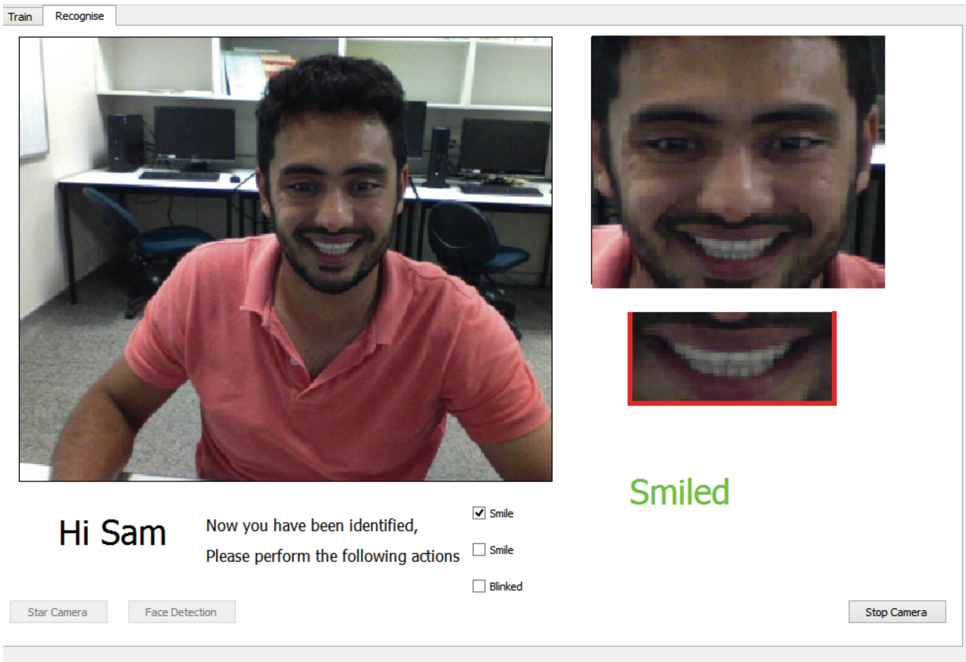

Figure 39 person has been recognised and the liveness detection is activated.

The instructions have been generated randomly by the class Random section 3.9 in this research. The user then must follow the instructions in the same order e.g. Blink, Blink and Smile (Figure 40). If the instructions were followed correctly and in the right order the system will authenticate, otherwise the face recognition will start over from the beginning (Detect, recognise, instructions) with a new set of randomly generated instructions (Figure 41).

The reason for repeating the face recognition again is to ensure the same person that has been identified by the system is one who is performing the instructions

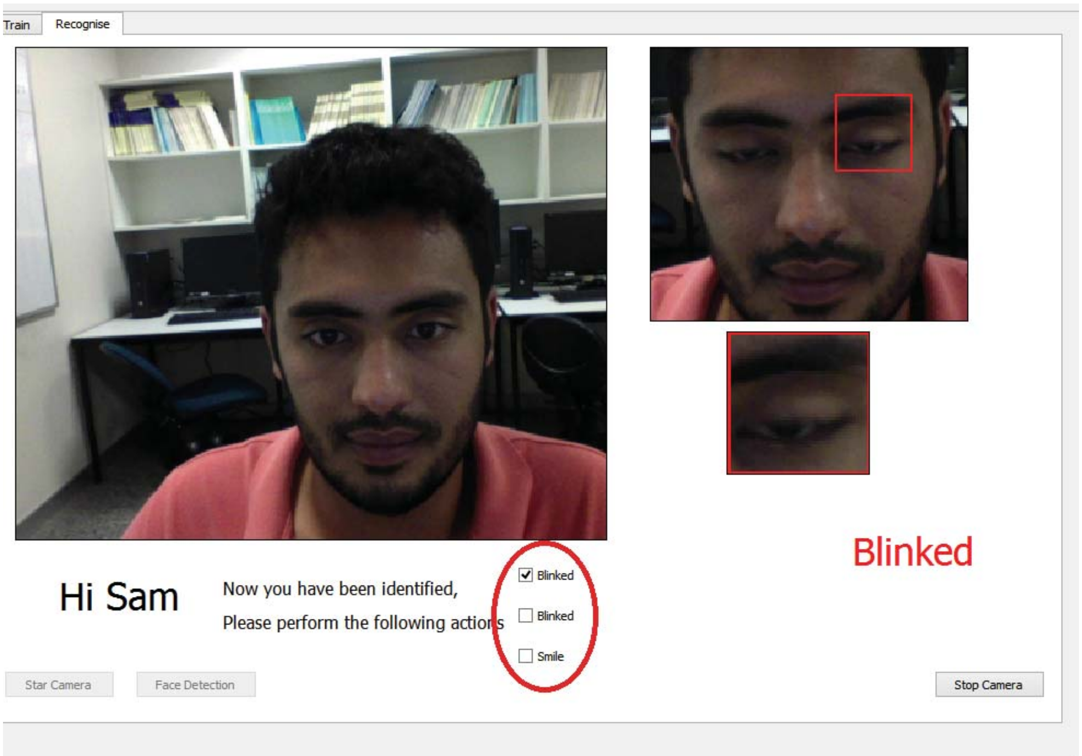

Figure 40 the set of instructions are Blink, Blink Smile and detected the first blink

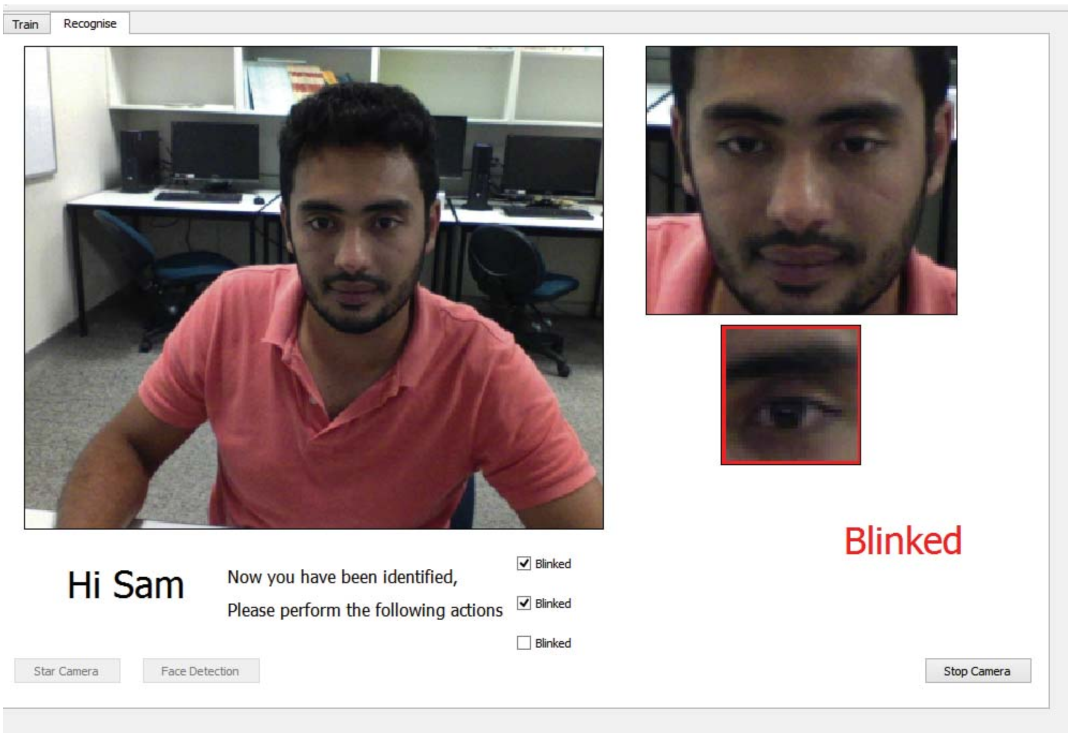

Figure 41 new set of instructions created after a failed attempt to follow the instructions

If there was no match found in the predict function, it will not proceed processing the image but it will rather use another frame to process.

A further improvement to the system was made by adding the action log, the mouth that has been detected and the preview of the image to be detected after the pre-processing (Figure 42).

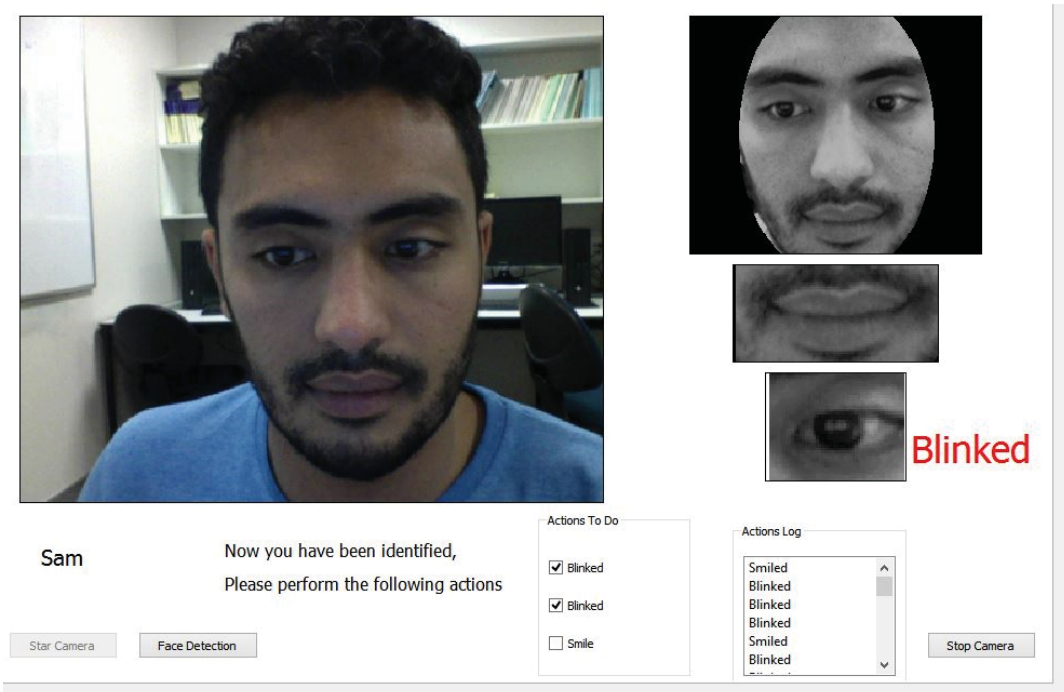

Figure 42 adding the action log and new mouth window

4.3.Issues during the Development

The main issue faced was the limitation of a standard web camera, which was used throughout to detect the eye blinking. The human eye can blink within 200 milliseconds and thus, the standard camera with some processing time will capture around 9 to 14 frames per second. Sometimes the eye blink is missed due to the speed of the human eye blinking and the time spent on processing other frames. Also, trying to detect the blink through eyeglasses was challenging because glasses were reflecting the lights and unexpected results from the blinking methods were returned.

5. Experimental Results

Because the system is companion of multiple methods and algorithms, the tests were split as following:

- Test the face detection on static images
- Test the eye detection on static images
- Test the mouth detection on static images
- Test the face recognition on static image
- Test the eye blinking detection from recorded video
- Test the smile detection from recorded video.
- Integrated system test

The images were obtained from FERET [28] and CMU_MIT_images [29]. FERET images contain only one face per image whereas CMU_MIT images contain one or more faces per image.

The variables for detection experiment results for FERET dataset are parameters of the cascade classifiers, which are scaling factor, the minimum number of neighbours and the maximum object size. Scaling Factor is the parameter to specify how much the image size is reduced at each image scale. Minimum number of neighbours is a value that specifies the minimum number of neighbours each candidate should have to retain the neighbor. Maximum Size is the maximum possible object size to detect. Objects greater than the maximum size are ignored.

5.1.1. Test face detection

For verifying the detection, a test application was created to read an image and detect the face in the image. The face detection tests were carried out on images contain only one face per image FERET [28]. These images also used to detect eyes and mouth.

The face detection experiment results for FERET dataset obtained by varying the three parameters of the cascade classifier, scaling factor, minimum number of neighbours and the maximum object size.

Table 7 face detection Experimental Results

True Detection	False Detection	Parameters		
		scaleFactor	minNeighbours	maxSize
99.78%	3.01%	1.1	3	Size(30| 30)
98.93%	0.22%	1.2	4	Size(35| 35)
96.62%	0.07%	1.3	5	Size(40| 40)
88.54%	24.25%	1.4	5	Size(25| 25)
85.71%	4.08%	1.5	5	Size(20| 20)
89.90%	0.59%	2	3	Size(30| 30)
8.52%	0.00%	3	3	Size(25| 25)
99.85%	9.99%	1.05	3	Size(15| 15)

The best parameters that would give the lowest false detection with high detection rate are scaling factor: 1.2, minimum number of neighbours: 4 and the maximum object size: (35, 35) pixels (Table 8).

Table 8 FERET image test results table for the face detection

True Detection	False detection	Missed
98.93% (2693/2722)	6	29/2722

Further testing on the face detection was carried out on the images containing one or more faces in them. Each face with different angle and/or expression. The image source is CMU_MIT_images, Figure 43 and Figure 44 show sample of the dataset from CMU_MIT_images [29]. The cascade classifier parameters used in this test are the optimal one found in the previous experiments (scaling factor: 1.2, minimum number of neighbours: 4 and the maximum object size: (35, 35) pixels)

The results are showing below in the (Table 9).

Table 9 CMU_MIT_images image test results table for the face detection

True Detection	**False detection**	**Missed**
92.94% (158/170)	7	12/170

Note that CMU_MIT images used for test and experiment on detecting only faces. This is because the dataset contains images with multiple faces. Multiple faces are not part of the system because it only detects one face at a time.

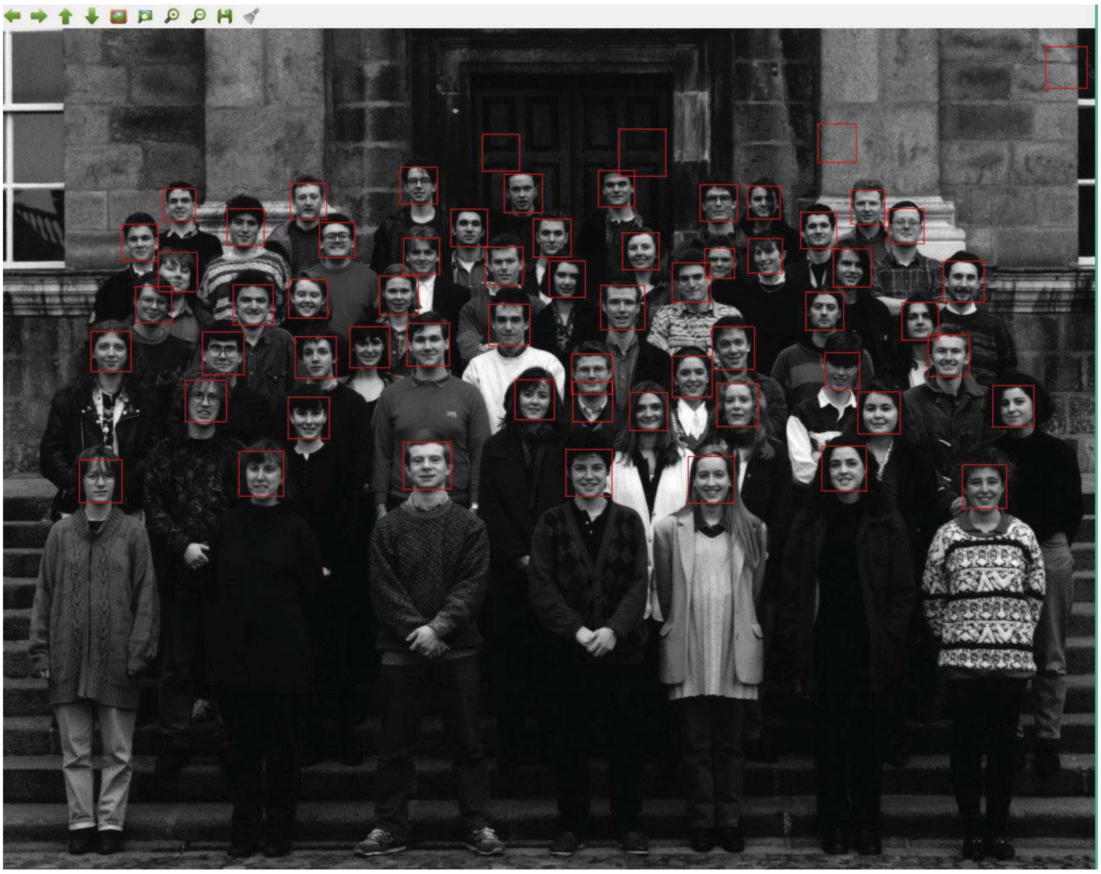

Figure 43 CMU_MIT_images sample image

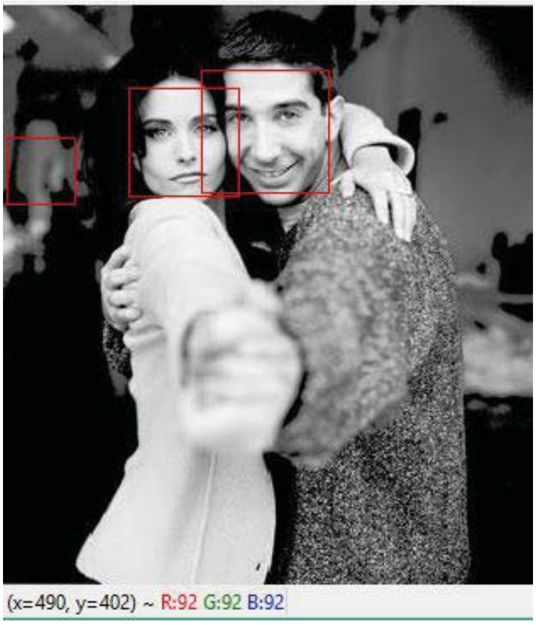

Figure 44 CMU_MIT_images sample image

5.1.2. Test the eye detection

FERET images [28] were used to test the eye detection on a face. The results are also obtained by varying the eye cascade classifier parameter scaling factor, minimum number of neighbours and the maximum object size (Table 10).

Table 10 eye detection test results on FERER [28] dataset

True Detection	False Detection	Parameters		
		scaleFactor	minNeighbours	maxSize
78.95%	4.34%	1.1	3	Size(30| 30)
88.39%	9.11%	1.05	3	Size(15| 15)
40.63%	1.58%	2	2	Size(30| 30)
45.52%	15.98%	1.4	5	Size(25| 25)
73.07%	0.84%	1.2	4	Size(35| 35)
1.03%	0.00%	3	3	Size(25| 25)
44.12%	9.77%	1.5	5	Size(20| 20)
56.65%	1.10%	1.3	5	Size(40| 40)

The best parameters for the eye that would give the lowest false detection with high detection rate are scaling factor: 1.05, minimum number of neighbours: 3 and the maximum object size: (15, 15) pixels. It is a little slow comparing it to higher scaling factor; however, the region of the eye is small so it detects the eye in real time.

After finding the optimal parameter to detect the eye, the initial tests were conducted without any mask applied to the face i.e. in every detected face we try to find an eye in the whole face. Therefore, the expected result is two eyes per image (Table 11).

Table 11 results of the image eye detection without mask.

True Detection	False detection	Missed
91.25% (4968/5444)	861	476/5444

91.25% correctly detected is a good result, however the false detection is very high with 861 objects classified as eyes but they are not.

The results improved significantly after applying a mask to the face shown in Figure 45. The true detection percentage is lower but the false detection improved from 861 in Table 11 to 205 in Table 12. Because the region of interest is only the top part of the face where human eyes are, the masks cover the lower part of the face (Figure 46). The expected results were in every face we detect also two eyes (Table 12).

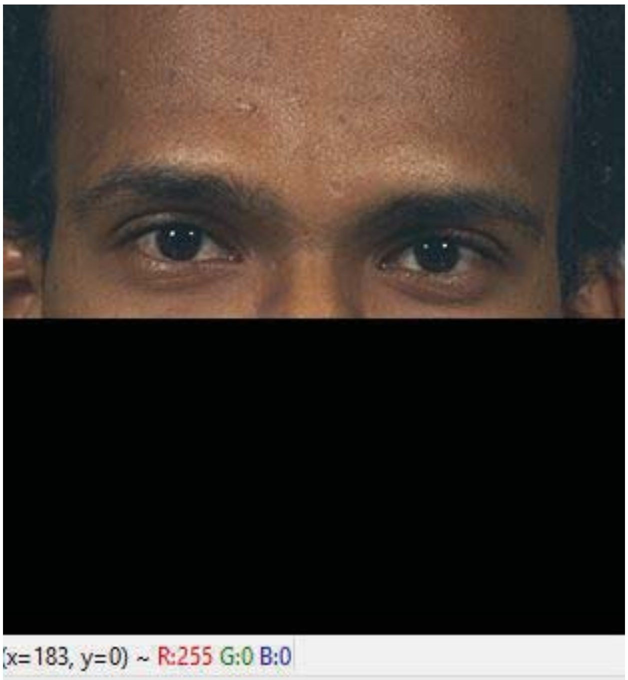

Figure 45 masks cover the lower part of the face

Figure 46 masks cover the lower part of the face

Table 12 results from the eye detection with mask covering the lower part of the face.

True Detection	False detection	Missed
81.194%(4460/5444)	205	984/5444

The final test was to cover one eye and only try to detect the other (Figure 47). The reason of this test was because we were interested in an eye that blinks, so we could determine the blink by one eye. This will save computational power and improve the speed and results. The expected outcome was one eye per image. The original face image and the resulting image are shown in (Figure 48). The result of the true detection improved from the previous test to 88.39% correctly classified as an eye Table 13.

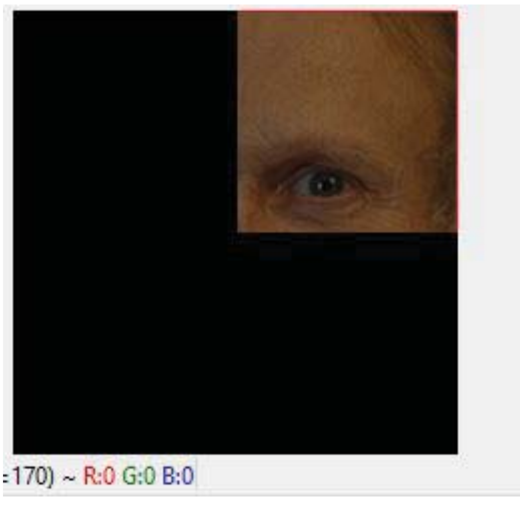

Figure 47 mask covering all face except the eye region

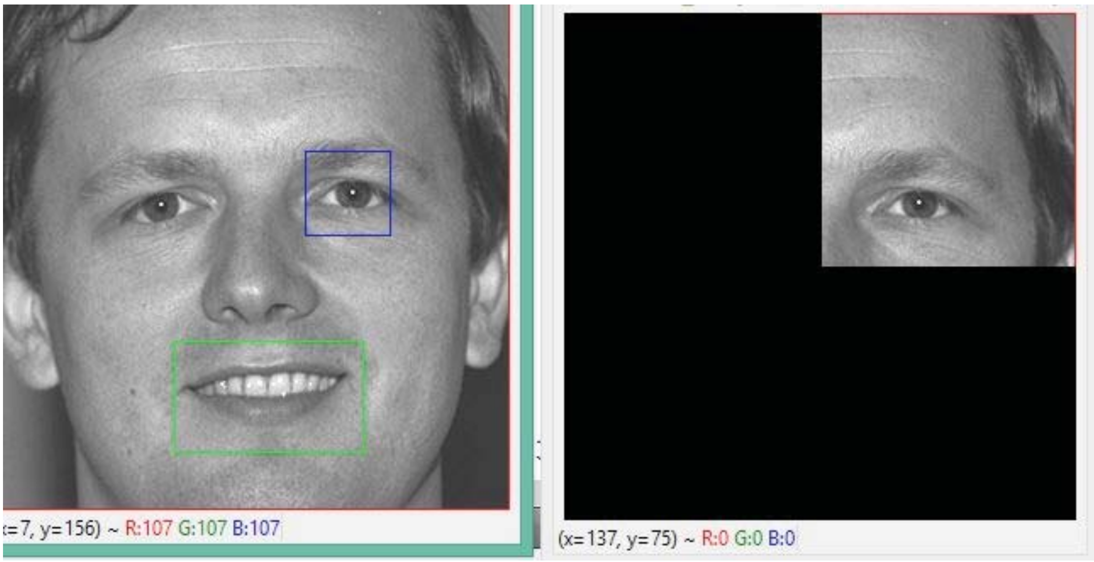

Figure 48 left image is the original face and right one is the image with mask covering all face except the eye region

Table 13 results from the eye detection with mask covering the lower and top left part of the face.

True Detection	False detection	Missed
88.39% (2406/2722)	248	316/2722

5.1.3. Test the mouth detection

FERET images [28] were also tested on the mouth detection. The same variables used in the eye and face detection cascade classifiers are also used on the mouth cascade classifier to determine the lowest false detection rate while keeping the true detection rate high.

Table 14 mouth detection test results on FERET images

True Detection	False Detection	Parameters		
		scaleFactor	minNeighbours	maxSize
98.75%	18.59%	1.1	3	Size(30| 30)
99.38%	37.07%	1.05	3	Size(15| 15)
77.19%	7.35%	2	2	Size(30| 30)
84.72%	102.20%	1.4	5	Size(25| 25)
97.13%	5.55%	1.2	4	Size(35| 35)
4.04%	0.00%	3	3	Size(25| 25)
80.12%	32.59%	1.5	5	Size(20| 20)
93.98%	4.59%	1.3	5	Size(40| 40)

The best parameters for the mouth that would give the lowest false detection with high detection rate are scaling factor: 1.2, minimum number of neighbours: 4 and the maximum object size: (35, 35) pixels (Table 14).

The initial test was applying the mouth detection methods on the whole face with each face containing one person and therefore one mouth was expected per image. The results were the following (Table 15):

Table 15 mouth detection applied on the whole face without mask

True Detection	False detection	Missed
98.86% (2691/2722)	2253	31/2722

This is a very poor detection rate shown in Table 15 with more than 80% of false positive detection. Therefore, we apply a mask on the face to split the face into upper and lower parts. The upper part is 70% of the height of the face and it is covered with mask. The lower part is the remaining 30%, which shows the mouth region only (Figure 49).

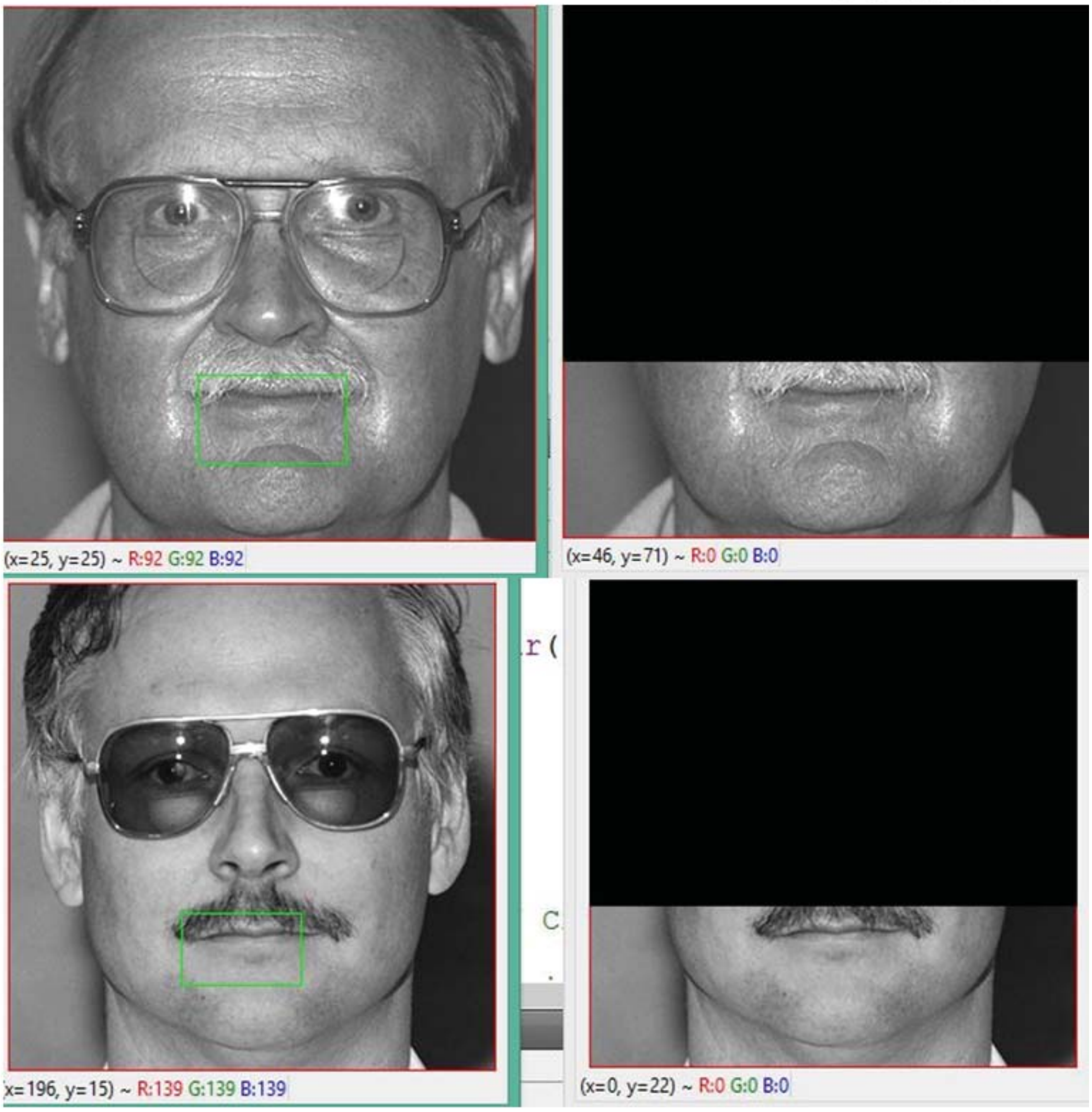

Figure 49 mask applied to the face 70% the upper part covered and 30% the lower part

Table 16 mouth detection applied on the masked face

True Detection	False detection	Missed
98.86% (2667/2722)	251	55/2722

Clearly there is huge improvement in the results in Table 16. Focusing the search on the moth region in the face was the best option.

The following tables Table 17, Table 18 and Table 19 are showing the detection results summary.

Faces: Total number of images = 2722

Table 17 face detection experiment results

Parameters			True Detection	False Detection	Detection Missed
scaling factor	minimum neighbours	Max Size			
1.1	3	Size(30| 30)	**2716**	82	6
1.05	3	Size(15| 15)	**2718**	272	4
2	2	Size(30| 30)	**2447**	16	275
1.4	5	Size(25| 25)	**2410**	660	312
1.2	4	Size(35| 35)	**2693**	6	29
3	3	Size(25| 25)	**232**	0	2490
1.5	5	Size(20| 20)	**2333**	111	389
1.3	5	Size(40| 40)	**2630**	2	92

Eye: Total number of images = 2722

Table 18 Eye detection experiment results

Parameters			True Detection	False Detection	Detection Missed
scaling factor	minimum neighbours	Max Size			
1.1	3	Size(30| 30)	**2149**	118	573
1.05	3	Size(15| 15)	**2406**	248	316
2	2	Size(30| 30)	**1106**	43	1616
1.4	5	Size(25| 25)	**1239**	435	1483
1.2	4	Size(35| 35)	**1989**	23	733
3	3	Size(25| 25)	**28**	0	2694
1.5	5	Size(20| 20)	**1201**	266	1521
1.3	5	Size(40| 40)	**1542**	30	1180

Mouth: Total number of images = 2722

Table 19 Mouth detection experiment results

Parameters			**True Detection**	**False Detection**	**Detection Missed**
scaling factor	minimum neighbours	Max Size			
1.1	3	Size(30| 30)	**2688**	506	34
1.05	3	Size(15| 15)	**2705**	1009	17
2	2	Size(30| 30)	**2101**	200	621
1.4	5	Size(25| 25)	**2306**	2782	416
1.2	4	Size(35| 35)	**2644**	151	78
3	3	Size(25| 25)	**110**	0	2612
1.5	5	Size(20| 20)	**2181**	887	541
1.3	5	Size(40| 40)	**2558**	125	164

The parameters for the face detection that gave the most accurate results and lowest false positive rate are scaling factor is 1.2, minimum number of neighbours is 4 and the maximum object size is (35, 35) pixels (Table 17).

For the eye detection best parameters are scaling factor is 1.05, minimum number of neighbours is 3 and the maximum object size is (15, 15) pixels (Table 18). Despite the fact that the scaling factor is small which in return makes the detection slower, the region we are processing is very small so the effect of the slowness is minimal.

Finally, the mouth detection parameters were used because the results of the detection were acceptable: scaling factor is 1.2, minimum number of neighbours is 4 and the maximum object size is (35, 35) pixels (Table 19).

5.2. Test the face recognition on static image

This test was carried out to determine the accuracy of the face recognition. The images used in this test were from FERET [28]. FERET images library contain multiple faces of the same persons. So I have split the library into test set and training set. In order to train the classifier we needed at least 10 images for each person, there were only few images per person in the FERET datasets [28]. So instead in the training set, we read the image of the person then rotate it to left or right by few degrees (-3,+3 randomly) then train the classifier. Then added random images to train in the same classifier so that the trained classifier contains more than one person. After that, we read the same person image from the test set and try to recognise it. The results are in the following Table 20. (Figure 50)

Table 20 Face recognition on static image results

Confidence level	count	Results
5-10	451	Predicted correctly
10-15	705	Predicted correctly
15-20	155	Predicted correctly
>20	42	Predicted correctly
infinity	8	Predicted incorrectly

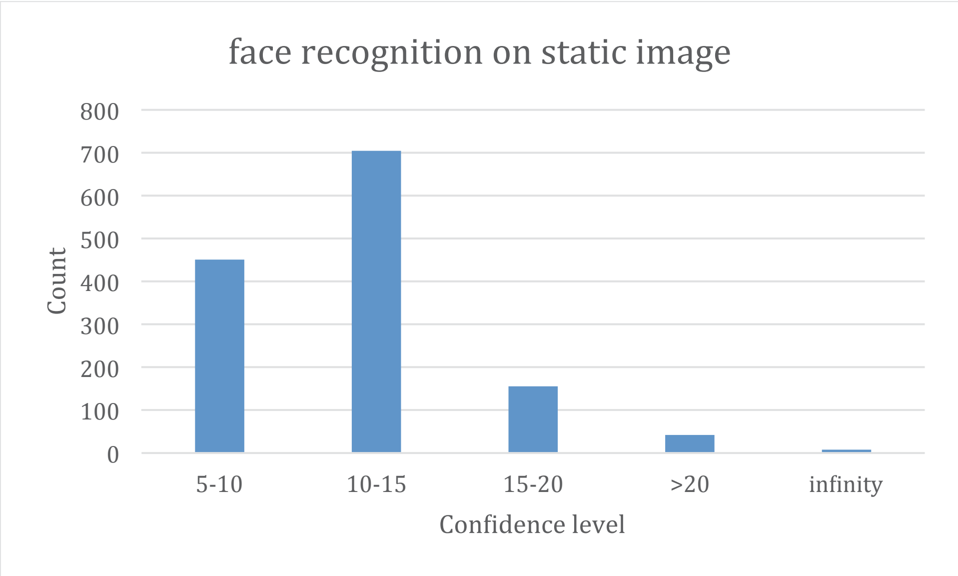

Figure 50 Face recognition on static image results on column chart

5.3. Test the eye blinking and smile detection from recorded video

In this test, a video recorder was created to record a short video. Each video contains blinks or smiles. Each examinee records 100 short video blinking and 100 smiling. Then we load each videos that contains blinks and applied the blink detection. Similarly, we loaded the videos that contain the smiles and applied the smile detection. For each set, the expected results were 100 blinks and 100 smiles. The parameters for the eye and mouth detection were set to optimal to minimise false detection or miss-detecting of the eye or the mouth. In the first experiment, the blinking was natural, i.e. around 200ms per blink. The results are in the following Table 21.

Table 21 Test the eye blinking and smile detection from video natural video speed

	blink	smile
examinees 1	**108**	**114**
examinees 2	**112**	**110**
examinees 3	**92**	**109**
examinees 4	**106**	**118**
examinees 5	**115**	**103**

Further testing on the same set of videos of the eye with a slow motion of the blink to reproduce user blinking for one second Table 22.

Table 22 eye blinking and smile detection test from slow motion video

	blink	smile
examinees 1	**134**	**127**
examinees 2	**141**	**119**
examinees 3	**113**	**125**
examinees 4	**129**	**131**
examinees 5	**136**	**121**

The results above from Table 21 and Table 22 show that using the human natural blinking speed was better in terms of false positive detection. Also the smile intensity of slower motion video was not as accurate as opposed to natural speed video (not slow motion applied).

5.4. Integrated System Test

The system was trained to recognise the 7 examinees. After the training, the recognition module is used to test the system, with a static list of actions.

Table 23 Experimental test results on the system. 1 = true, 0 = false E: examinee

Actions Order			E1	E2	E3	E4	E5	E6	E7
Smile	**Smile**	**Smile**	1	1	1	0	1	0	1
Smile	**Smile**	**Blink**	1	1	0	1	1	1	0
Smile	**Blink**	**Smile**	0	0	1	1	1	1	0
Smile	**Blink**	**Blink**	1	1	1	0	1	0	1
Blink	**Smile**	**Smile**	1	0	1	0	1	1	0
Blink	**Smile**	**Blink**	1	1	1	1	1	1	1
Blink	**Blink**	**Smile**	0	0	1	1	1	1	1
Blink	**Blink**	**Blink**	1	1	0	1	1	0	0
Total			**6**	**5**	**6**	**5**	**8**	**5**	**4**
Total Sum			***39/56***						

On average the system authenticated users around 70% of the time on small dataset. The test method used to measure the accuracy of the system as a whole was to hard-coded the actions lists and repeat the same actions on 7 live faces (Table 23).

6. Conclusion and Future Work

In this research, we have explored the face recognition case study and its potential flaws. A system was developed to address the security issues in the face recognition problem and prevent spoofing face recognition system. The system offers a unique reliable biometric face recognition system that is hard to circumvent the authentication. Furthermore, it does not require any additional hardware other than a standard camera. Implementing and combining multiple algorithms have assisted in achieving these results. The major implemented components were the face recognition algorithm, the eye blinking detection algorithm, and the smile detection algorithm. The system was able to correctly authenticate and verify 70% of the test users were actually live people in front of the camera, which was illustrated in section 5.4. Testing the separate components of the system gave a much higher results but that is was due to the fact the test samples for the separate parts were much larger than the integrated system tests.

An eye blinking method was introduced as a new way of detecting the eye blinking in section 3.7. By utilizing the Viola and Jones method for object detection, we were able to detect the blink of the eye in a face.

The instructions provided by the system are randomly generated as mentioned above in section 3.9. The user must perform the actions in the exact same order in order to authenticate after the face recognition is completed. The recognition and liveness authentication will start from the beginning if an incorrect action was performed as shown in section 4.2. The system needs to be trained to learn new faces in order to recognise them by using the training module as explained in section 4.1.

One of the future works of the system is to port it to mobile devices such as Android operating system or Windows phones and determine the limitations of such demanding system on a mobile device with limited processing power.

One of the limitations of the system is that it requires a high collaboration by the user, as it needs the users' response to every instruction. This can be refined in future work to obtain the actions from the user in an easier matter while preserving the main objective of the authentication system.

This authentication system can be further utilised in conjunction with other high-end hardware detector by integrating it with the unique random actions generator part of this system. For example, Intel RealSense 3D Camera [33] can be used with the random actions generator to create a robust and accurate authentication system.

7. Bibliography

[1] S. C. Dakin and R. J. Watt, "Biological “bar codes” in human faces," *Journal Of Vision,* vol. 9, no. 4, 2009.

[2] G. Marcus, "Ray Kurzweil’s Dubious New Theory of Mind," 15 11 2012. [Online]. Available: http://www.newyorker.com/books/page-turner/ray-kurzweils-dubious-new-theory-of-mind. [Accessed 2014].

[3] G. Pan, L. Sun, Z. Wu and S. Lao, "Eyeblink-based anti-spoofing in face recognition from a generic webcamera," *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference,* pp. 1-8, October 2007.

[4] S. Tikoo and N. Malik, "Detection, Segmentation and Recognition of Face and its Features Using Neural Network," *J Biosens Bioelectron Journal of Biosensors & Bioelectronics,* vol. 7, no. 2, 2016.

[5] A. J. Goldstein, L. D. Harmon and A. B. Lesk, " Identification of Human Faces," *proceedings IEEE,,* vol. 59, no. 5, 1971.

[6] L. Sirovich and M. Kirby, "A low dimensional Procedure for the characterization of human faces," *Journal of the Optical Society of America A,* vol. 4, no. 3, pp. 519-524, 1987.

[7] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," Maui, 1991.

[8] D. L. Baggio, S. Emami, D. M. Escriva, K. Ievgen, N. Mahmood, J. Saragih and R. Shikrot, Mastering openCV with practical computer vision projects: Step-by-step tutorials to solve common real-world computer vision problems for desktop or mobile, from augmented reality and number plate recognition to face recognition and 3D head tracking., Birmingham: Packt Publishing Limited, 2012.

[9] P. N. Druzhkov, V. L. Erukhimov, N. Y. Zolotykh, E. A. Kozinov, V. D. Kustikova, I. B. Meerov and A. N. Polovinkin, "New object detection features in the OpenCV library.," *Pattern Recognition and Image Analysis Pattern Recognit. Image Anal,* vol. 21, no. 3, pp. 384-386, 2011.

[10] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," *IEEE Transactions On Computer Vision and Pattern Recognition* , vol. 1, pp. 511 - 518, 2001.

[11] OpenCV, 2014. [Online]. Available: http://opencv.org/. [Accessed 2014].

[12] R. L. Hsu, M. M. Abdel and A. Jain,, "Face detection in color images," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 358-367, 2002.

[13] D. Zhu, S. Xia, X. Zhou and J. Zheng, "Hybrid method for human eye detection," in *In Control and decision conference*

(2014 CCDC), The 26th Chinese, 2014.

[14] M. Grgic and K. Delac, "Algorithms," 07 11 2008. [Online]. Available: www.face-rec.org/algorithms/. [Accessed 2014].

[15] M.-H. Yang,, D. Kriegman, and N. Ahuja, "Face Detection Using Multimodal Density Models," *Computer Vision and Image Understanding,* vol. 84, p. 264–284 , 2001.

[16] L. R. Rabiner and B. H. Juang, "An introduction to hidden Markov models," *IEEE3(1),* pp. 4-16, 1986.

[17] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal Of Cognitive Neuroscience,* vol. 3, pp. 71-86, 1991.

[18] M. Lalonde, D. Byrns, L. Gagnon, N. Teasdale and D. Laurendeau, "Real-time eye blink detection with GPU-based SIFT tracking," in *In Computer and Robot Vision, 2007. CRV'07. Fourth Canadian Conference* , 2007.

[19] R. Heishman and Z. Duric, "Using image flow to detect eye blinks in color videos," *In Applications of Computer Vision, 2007. WACV'07. IEEE Workshop,* pp. 52-52, 2007.

[20] A. Krenker, B. Janez and A. Kos, "Introduction to the Artificial Neural Networks, Artificial Neural Networks - Methodological Advances and Biomedical Applications," Prof. Kenji Suzuki (Ed.). InTech, DOI: 10.5772/15751, 2011. [Online]. Available: http://www.intechopen.com/books/artificial-neural-networks-methodological-advances-and-biomedical-applications/introduction-to-the-artificial-neural-networks.

[21] L. Smith, "Using a Framework to Specify a Network of Temporal Neurons," *Paper presented at 1st Slovak Symposium on Neural Networks and their Applications,* 1996.

[22] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition," *Foundations (eds Rumelhart, D. E. & McClelland, J. L.),* vol. 1, pp. 318-362, 1986.

[23] M. OpenCourseWare, "Artificial Intelligence Course," n.d..

[24] N. H. Reyes, *Neural Networks Lecture, 159302 Paper, Computer Science, Massey University,* 2013.

[25] C. Shan, "Smile Detection by Boosting Pixel Differences," in *Image Processing, IEEE Transactions on, 21(1)*, 2012.

[26] J. Whitehill, G. Littlewort, I. Fasel, M. Bartlett and J. Movellan, "Developing a practical smile detector," in *In Proc. IEEE Int. Conf. Automatic Face and Gesture Recognition* , 2008.

[27] S. Lawrence, C. L. Giles, A. C. Tsoi and A. D. Back, "Face Recognition: A Convolutional Neural Network Approach," *IEEE Transactions on Neural Networks, Special Issue on Neural Networks and Pattern Recognition,* vol. 8, pp. 98-113, 1997.

[28] P. J. Phillips, H. Moon, S. A. Rizvi and P. J. Rauss, "The FERET Evaluation Methodology for Face Recognition Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, pp. 1090-1104, 2000.

[29] K. Sung, T. Poggio, H. Rowley, S. Baluja and T. Kanade, "CMU/VASC MIT images database," 2000. [Online]. Available: http://vasc.ri.cmu.edu/idb/html/face/frontal_images/.

[30] F. Song, X. Tan, X. Liu and S. Chen, "Eyes Closeness Detection from Still Images with Multi-scale Histograms of Principal Oriented Gradients," *Pattern Recognition,* 2014.

[31] "Qt Documentation," [Online]. Available: http://doc.qt.io/qtcreator/.

[32] A. Rand, million random digits with 100,000 normal deviates, Rand Corporation, 1955.

[33] M. Draelos, Q. Qiu, A. Bronstein and G. Sapiro, "Intel realsense= real low cost gaze," in *In Image Processing (ICIP), 2015 IEEE International Conference*, 2015.

[34] T. Ahonen, A. Hadid and M. Pietikainen, "Face description with local binary patterns: Application to face recognition. Pattern Analysis And Machine Intelligence,," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 12, pp. 2037-2041, 2006.

[35] G. Bradski and A. Kaehler, Learning OpenCV : computer vision with the OpenCV library, Farnham: O'Reilly, 2008.

[36] T. Ahonen, A. Hadid and M. Pietikainen, "Face recognition with local binary patterns," in *European Conference on Computer Vision; Computer vision - ECCV 2004*, London, 2004.

[37] s. Z. Li and A. K. Jain, Handbook of face recognition, New York: Springer, 2005.

[38] W. Zhao, R. Chellappa and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Surveys,* vol. 35, no. 4, pp. 399-458, 2003.

[39] J. Wright, A. Ganesh and Y. Ma, *IEEE Transactions On Pattern Analysis And Machine Intelligence,* vol. 31, no. 2, 2009.

[40] R. Lienhart and J. Maydt, "An Extended Set of Haar-Like Features for Rapid Object Detection," in *Proceedings International Conference on Image Processing*, 2002.

[41] P. Belhumeur, J. Hespanha and D. Kriegman, " Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 19, no. 7, pp. 711-720, 197.

[42] G. Chetty and M. Wagner, "Multi-level liveness verification for face-voice biometric authentication," in *In Biometric Consortium Conference, 2006 Biometrics Symposium: Special Session on Research*, 2006.

[43] G. Edwards, C. Taylor and T. Cootes, "Interpreting Face Images using Active Appearance Models," in *International Conference on Face & Gesture Recognition*, 1998.

[44] K. Arai and R. Mardiyanto, "Real Time Blinking Detection Based on Gabor Filter," *International Journal of Human Computer Interaction (IJHCI), Volume (1): Issue (3),* pp. 33-45,

2010.

[45] L. Wang, X. Ding, C. Fang, C. Liu and K. Wang, "Eye blink detection based on eye contour extraction," in *In IS&T/SPIE Electronic Imaging. International Society for Optics and Photonics.*, 2009.

[46] "The Database of Faces," [Online]. Available: http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

8. Appendix

8.1. Source Code

Random Class Source Code

```
#ifndef RANDOM
#define RANDOM
#include <time.h>
#include <string>
#include <vector>
#include <opencv2/opencv.hpp>
#include <QStringList>

struct mylist{
  int actions[3];
};

class Random
{
  public:
  mylist actions;
  Random()  //constructor
  {
    actions.actions[0]=0;
    actions.actions[1]=0;
    actions.actions[2]=0;
    Names[0]="Smile";
    Names[1]="Blinked";
    matchable_ = false;
    srand (time(NULL));
  }  //ENDconstructor

  bool getRandomAction()
  {
    return rand() % 2 == 1;
  }
```

```
void SetNewRandomList()
{
  actions.actions[0]=0;
  actions.actions[1]=0;
  actions.actions[2]=0;
  Action.clear();
  ActionString.clear();
  UserAction.clear();
  bool simlpe = true;
  matchable_ = false;

  for(int i=0; i<3; i++)
  {
    simlpe = getRandomAction();
    ActionString.append(Names[simlpe]);
    Action.push_back(simlpe);
  }
}//END SetNewRandomList

QStringList GetActionList()
{
  return ActionString;
}

void SetUserAction(bool action)
{
  if(!matchable_)
  {
    UserAction.push_back(action);
    size_t current = UserAction.size()-1;
    if(UserAction[current]==Action[current])
    {
      actions.actions[current]=1;
    }
    if(UserAction.size()==Action.size())
    {
      matchable_ = true;
    }
  }
}  //END SetUserAction
```

```
  bool isMatchable(){
    return matchable_;
  }



  bool DoesMatch()
  {
    if(Action.size() > UserAction.size())
      return false;
    bool isMatched = true;
    for(size_t i=0; i< Action.size(); i++)
    {
      if(Action[i] != UserAction[i])
      {
        isMatched = false;
        break;
      }
    }
    return isMatched;
  }  //END DoesMatch


  private:
  bool matchable_;
  QString Names[2];
  QStringList ActionString;
  std::vector<bool> Action, UserAction;
};
#endif // RANDOM
```

Camera Thread Header Source Code

```
#ifndef CAMERATHREAD_H
#define CAMERATHREAD_H
#include "opencv/highgui.h"
#include <qthread.h>
#include <QImage>
#include <qwaitcondition.h>
#include <opencv2/opencv.hpp>

class CameraThread : public QThread
{
   Q_OBJECT
   public: CameraThread(int device);
   CameraThread(int device, int height, int width);
   void Stop();
   cv::Mat getLastFrame();
   void ConnectCamera();
   int getHieght();
   int getWidth();
   bool IsCameraOn();
   QWaitCondition frame_syn_;

Signals:
   void updateLabelOrignal(QImage extra);

private:
   void run();
   cv::VideoCapture cap;
   int x_, y_, device_num_;
   cv::Mat frame ;
   bool run_, is_connected_;
};
#endif // CAMERATHREAD_H
```

Camera Thread Source Code

```
#include "camerathread.h"
#include <MatToQImage.h>
#include <stdio.h>
#include <ctype.h>
#include <fstream>


CameraThread::CameraThread(int device)
{
   device_num_ = device; ConnectCamera();
}

CameraThread::CameraThread(int device, int height, int
width)
{
    device_num_ = device;
    x_ = width; y_ = height;
    ConnectCamera();
}

void CameraThread::Stop()
{
   run_ = false;
}

cv::Mat CameraThread::getLastFrame()
{
   return frame;
}
```

```
void CameraThread::run()
{
   run_ = true;
   std::cout<<"Started CameraThread"<<std::endl;
   cap.set(CV_CAP_PROP_FRAME_WIDTH,x_);
   cap.set(CV_CAP_PROP_FRAME_HEIGHT,y_);
   while(run_)
   {
     cap >> frame;
     frame_syn_.wakeAll();
     emit updateLabelOrignal(MatToQImage(frame));
   }
   std::cout<<"Finshed CameraThread"<<std::endl;
}

void CameraThread::ConnectCamera(){
   is_connected_ = true;
   cap.open(device_num_);
   if(!cap.isOpened())
   {
     is_connected_ = false;
     std::cout<<"Camera failed to connected."<<std::endl;
     if(device_num_!=0)
     {
        std::cout<<
          "Opening the default camera..."<<std::endl;
        cap.open(0);
        if(cap.isOpened())
        {
           is_connected_ = true;
           std::cout<<
              "Default camera is connected."<<std::endl;
        }
        else
           std::cout<<
               "Camera failed to connected."<<std::endl;
     }
   }
   else
      std::cout<<"Camera is connected."<<std::endl;
}
```

```
bool CameraThread::IsCameraOn()
{
   return is_connected_;
}

int CameraThread::getHieght()
{
   return y_;
}

int CameraThread::getWidth()
{
   return x_;
}
```

Eye Blinking Detection

```
bool HasBlinked(cv::Mat eye)
{
   int blink=0;
   if(!eye.empty())
   {
     cv::resize(eye, eye, cv::Size(50,50));
     equalizeHist(eye, eye );
     // eye = ContrastStretching(eye, true);
     blink = detectClosedEye(eye, islastFrameBlinkd);
     if (blink ==1)
     {
        //Blinked
        blinkCount++;
        islastFrameBlinkd=true;
        return true;
     }else if (blink == 2)
     {
        islastFrameBlinkd=true;
     }else
     {
        islastFrameBlinkd=false;
     }
   }
}
```

```
int detectClosedEye(Mat src,bool islastFrameBlinkd)
{
  std::vector<Rect> eyes;
  Mat eyeROI;
  int count=0;
  closed_eyes_cascade_.detectMultiScale( src, eyes, 1.7,
    3, 1, Size(30, 30) );
  namedWindow( "Eye", 1 );
  for( size_t i = 0; i < eyes.size(); i++ )
  {
    cv::rectangle(src, eyes[i], Scalar( 0, 0, 255 ) );
    count++;
    eyeROI = src( eyes[i] );
    if (!eyeROI.empty())
    {
      imshow( "Closed Eye", eyeROI );
    }
  }
  if (count == 1 and islastFrameBlinkd == false)
  {
    return 1;
  }
  else if (count == 1 && islastFrameBlinkd == true)
  {
    return 2;
  }
  return 0;
}
```