**FACE DITECTION & FACE RECOGNITION**

**USING OPEN COMPUTER VISION**

**CALASSIFIRES**

LAHIRU DINALANKARA

April 17, 2017

AINT513 Robotic Visual Perception and Autonomy

MSc Robotics

Plymouth University

1

# Introduction

The following document is a report on the mini project for the Robotic visual perception and autonomy. It involved building a face detection and recognition system using several classifiers available in open computer vision library(OpenCV). The advantage of face detection is it is an invasive identification system and faster than other systems since multiple faces can be analysed at the same time. The difference between face detection and identification is, face detection is to identify a human face from an image and locate the face. Face recognition is making the decision who’s face it is using an image database. In this project both are accomplished using different techniques and are described below. The report begins with a brief history of face recognition. This is followed by explanation of HAAR-cascades, Eigen-face, Fisher-face and Local binary pattern histogram (LBPH) algorithms. Next The methodology of the project is explained followed by analysing the results. Finally, a conclusion is provided on pros and cons on each algorithms and implementation.

# History of Face Recognition

Face recognition began as early as 1973 with the first automated system being introduced By Kanade using a feature vector of human faces. In 1983, principal component analysis(PCA) was introduced by Sirovich and Kirby. Using PCA, Turk and Pentland Eigen-face was developed in 1991 and is considered a major milestone in the technology[5]. Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms(LBPH)[2], [7]. In 1996 Fisher face was developed using Linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigen-face method [1]. Viola and Jones introduced a face detection technique using HAAR cascades and ADABoost[6]. In 2007 A face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes [3], [4]. In This project face detection is performed using HAAR cascades and face recognition by Eigen-face, Fisher-face and LBPH techniques.

# Haar Cascades

A Haar wavelet is a mathematical fiction that produces square shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations.

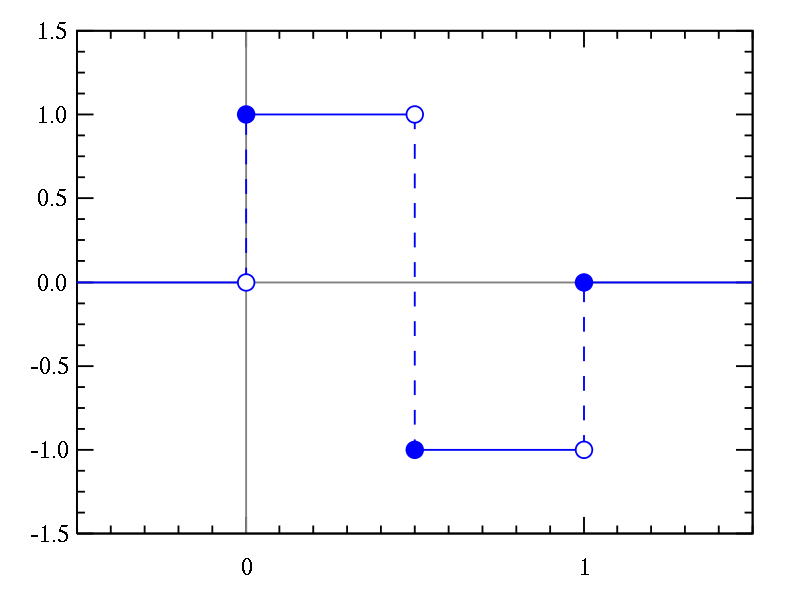


Figure 1: A Haar wavelet

By combining several wavelets, a cascade can be created that can be identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are used for object detection.

To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match.

Face direction is performed by matching different Haar-like-features with human faces. For example, forehead, eyebrows and eyes contrasts as well as the nose with the eyes as shown below.



Figure 3: Several Haar like features matched to the features of the authors face

A single classifier is not accurate enough. Several classifiers are combined as to provide an accurate face detection system as shown in the block diagram below.

A similar method was used in this project effectively to by identifying faces and eyes in combination resulting better face detection. Similarly, in viola Jones method, several classifies were combined to create stronger classifiers. ADA boost tests out several week classifiers on a selected location and choose the most suitable. It can also reverse the direction of the classifier and get better results if necessary. Weight update steps can be updated only on misses to get better performance. Resulting cascade is scaled by 1.25 on iterations to find faces with varied sizes. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones used a summed area table (integral image) to compute the matches fast. First developed in 1984, it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using integral image enables matching features with a single pass over the image.

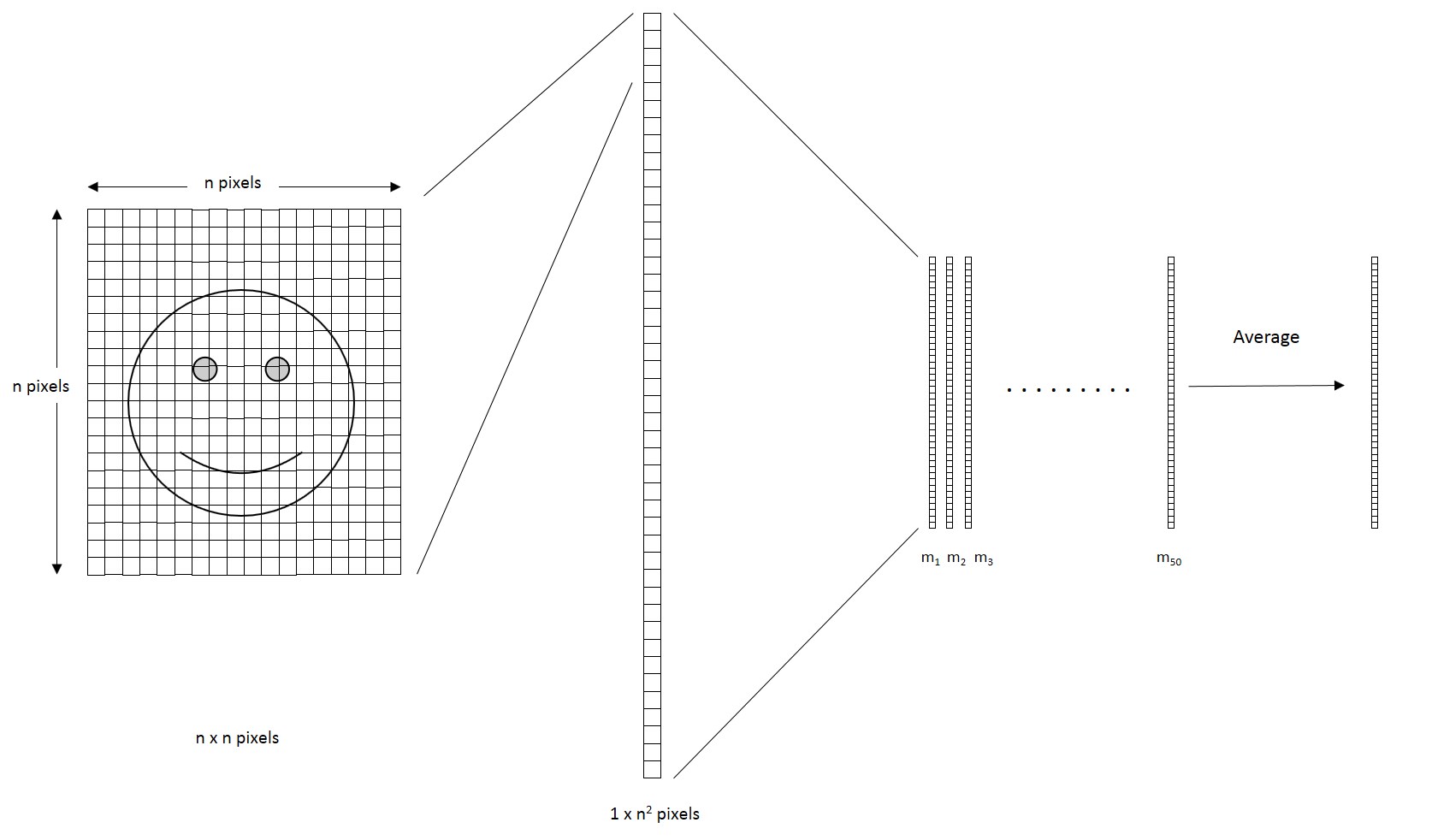
# Face Recognition

The following sections describe the classifier algorithms Eigen-face, Fisher face, Local binary pattern histogram and how they are implemented in OpenCV.

## Eigen Face

Eigen-face is based on PCA that classify images to extract features using a set of images. It is important that the images are in same lighting condition and the eyes match in each image. Also, images used in this method must contain the same number of pixels and in grayscale.

For this example, consider an n x n pixeled image. All images are treated as a vector by concatenating each row to create a vector with a length *n*2 matrices. All the images in the dataset are stored in a single matrix resulting a matrix with columns corresponding the number of images.



The matrix is averaged (normalised) to get a average human face.

Then each image is subtracted by the average face, resulting an adjusted face in each column that contains features unique to that face.

In the resulting matrix, each column is a representation of the amount of difference to the average of the dataset faces.

The next step is computing covariance matrix from the result (This is a large n x n matrix).

To obtain the Eigen vectors from the data, Eigen analysis is performed using principal component analysis.

From the result, where covariance matrix is diagonal, where it has the highest variance is considered the 1st Eigen vector.

2nd Eigen vector is the direction of the next highest variance and it is in 90 degrees to the 1st vector.

3rd will be the next highest variation and so on.

Each column is considered an image and visualised, resembles a face and called Eigen-faces.

When a face is required to be recognised, the image is imported, resized to match the same dimensions of the test data as mentioned above.

The weights then are calculated by projecting the image on to each of the Eigen-faces.

These weights correspond to the similarity of the features extracted from the different image sets in the dataset to the features extracted from the input image.

The input image can be identified as a face by comparing with the whole dataset.

By comparing with each subset, the image can be identified as to which person it belongs to.

By applying a threshold detection and identification can be controlled to eliminate false detection and recognition.

PCA is sensitive to large numbers and assumes that the subspace is linear. If the same face is analysed under different lighting conditions, it will mix the values when distribution is calculated and cannot be effectively classified. This makes to different lighting conditions poses a problem in matching the features as they can change dramatically.

## Fisher-face

Fisher-face technique builds upon the Eigen-face and is based on Linear discriminant analysis (LDA) derived from Ronald Fishers’ linear discriminant technique used for pattern recognition. However, it use labels for classes as well as the data point information.

When reducing dimensions, PCA looks at the grates variance, while LDA, using labels, looks at an interesting dimension such that, when you project to that dimension you maximise the difference between the mean of the classes normalised by their variance.

LDA maximises the ratio of the between-class scatter and the within-class scatter matrices.

Due to this, different lighting conditions in images has a limited effect on the classification process using LDA technique.

Eigen-face maximises the variations while Fisher-face maximises the mean distance between and different classes and minimises variation within classes.

This enables Fisher-face to differentiate between feature classes better than Eigen-face and can be observed in figure.

Furthermore, it takes less amount of space and is the fastest algorithm in this project.

Because of these PCA is more suitable for representation of a set of data while LDA is suitable for classification.

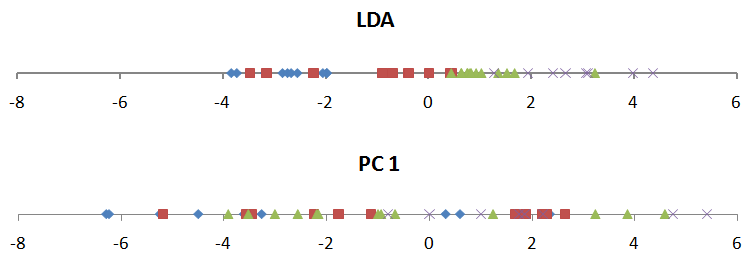


Figure 6: Note that in LDA the different data is more separated than in PCA

## Local Binary Pattern Histogram

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang [2].

The combination of LBP with histogram of oriented gradients descriptor was introduced in 2009 that increased its performance in certain datasets [7].

For feature encoding, the image is divided into cells (4 x 4 pixels).

Using a clockwise or counter-clockwise direction surrounding pixel values are compared with the central as shown in figure. The value of intensity or luminosity of each neighbour is compared with the centre pixel.

Depending if the difference is higher or lower than 0, a 1 or a 0 is assigned to the location.

The result provides an 8-bit value to the cell.

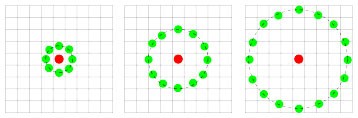
The advantage of this technique is even if the luminosity of the image is changed, the result is the same as before.

For larger cells histograms are used to find the frequency of occurrences of the values making the process faster.

By analysing the result in the cell, edges can be detected as the values change.

By computing the values for all cells and concatenating the histograms, feature vectors can be obtained.

images can be classified by processing with an ID attached.



Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face.

Eigen-face and Fisher-face compute dominant features of the whole training set while LBPH analyse them individually.

Methodology

Below are the methodology and descriptions of the applications used for data gathering, face detection, training and face recognition.

Face Detection

First task of the project was creating a face detection system using Haar-cascades. Although, training is required for Haar-cascades, OpenCV has a robust tried and tested Haar-cascades for face and eye detection. So, these were used instead of making new files. using Haar-cascades for face alone caused faces to be identified in random objects and eye cascades were used to make the system more robust. By using face and eye cascades in conjunction provided better results. The flowchart below shows the data flow in the face detection and capture system.

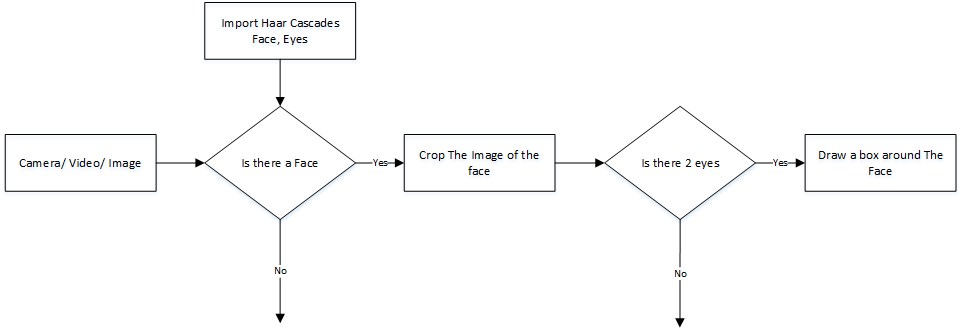


Figure 8: Flow chart of the face detection application

First, objects are created for face and eye detection using classifier class in OpenCV by using cv2.CascadeClassifier() and loading XML files for the face and for the eye detection. Then A camera object is created using cv2.VideoCapture() to capture images. By using detectMultiScale() function that detect object of various sizes and returns the Cartesian co-ordinates of the surrounding rectangle faces of different sizes are detected from the image. The first cascade verifies that there is a human face in the image and crops it. If so the eye cascade verifies there are two eyes in the cropped face. When all the criteria are satisfied a box is drawn around the detected face to illustrate that a face is detected in the location.

## Face Recognition Algorithms

For this project three algorithms are implemented independently. These are Eigen-face, Fisher face and Linear binary pattern histograms respectively. All three can be implemented using OpenCV libraries and Python or C++. There are three stages for the face recognition process and all three must go through all three stages.

1. Collecting data and storing in memory with IDs
2. Finding features unique to the image groups, classifying them and storing the information as XMLfiles
3. Determining similarities of features of an input image to the features in the saved XML files.

### Collecting the image data

First stage involves collecting different faces for feature extraction and classification. In some instances this is done manually using a photo editing software and cropping the photos. This is a time consuming and inefficient process since it is required to collect many images. Furthermore, PCA and LDA requires the same number of pixels in all the images. As a solution automatic face capture application was designed. The application uses Haar-cascades to detect a viable face for saving. The Flow chart illustrated below depicts the data flow in the application.

When the application starts, it requires the name of the person to be entered. This is stored in a text file with a generated ID for the user. Next the image from the camera is analysed using the face detection system described earlier. However, there is a new addition to the application. Haar cascade for detecting eyes with spectacles is far superior in detecting eyes while the eye cascade alone detects random objects. Using this cascade two eyes are detected in the image and then the tilting of the head is calculated. Next by using a rotational matrix the image is rotated and face detection cascade is implemented. This provides a stable image that is easier to recognise using the classifiers. The Image is then cropped and saved in a folder with the ID and an incrementing number to be identified and used later. A loop runs this program until 50 images are collected from the person. This application made data collection manageable and efficient.

### Training the Classifiers

OpenCV enables the creation of XML files that contain classified features from a set of images using 3 face recognition algorithms. For this, priviously saved images are imported, converted to grayscale with their IDs and saved into two lists with same indexes. After all the images has been stored in the lists the recogniser objects are created using face recogniser class. Each recogniser can take in parameters and are shown below with the syntax:

### cv2.face.createEigenFaceRecognizer()

1. Takes in the number of components for the PCA for crating Eigen-faces and it is noted that 80 canprovide with satisfactory reconstruction capabilities.
2. Similar to Eigen-face, if the distance to the most similar face is higher than this threshold, theobject will return a -1, that can be used to inform that the face is unrecognisable.

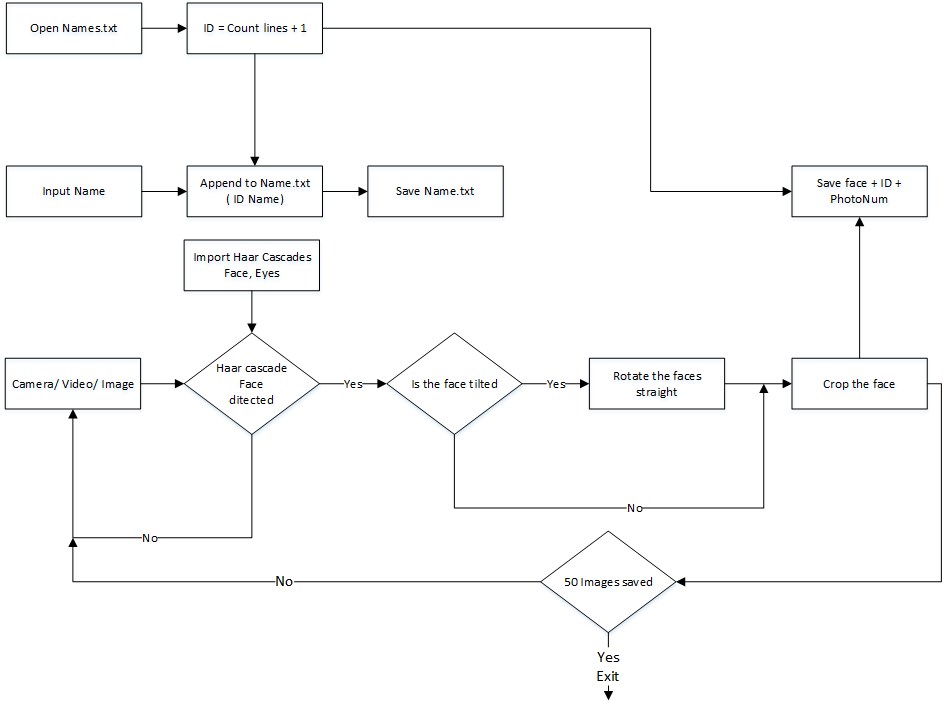


Figure 9: The Flowchart for the image collection

### cv2.face.createFisher-faceRecognizer()

1. Similar to Eigen-face, the first argument is the number of components for the LDA for the creationof Fisher-faces. and it is noted that 80 can provide with satisfactory reconstruction capabilities.
2. Takes in the threshold when recognising the faces. If the distance to the most similar face is abovethis threshold, the object will return a -1, that can be used to inform that the face is unrecognisable. **cv2.face.createLBPHFaceRecognizer()**
3. The radius from the centre pixel to build the local binary pattern.
4. The Number of sample points to build the pattern. Having a large number will slow down thecomputer.
5. The Number of Cells to be created in X axis.
6. The number of cells to be created in Y axis.
7. A threshold value similar to Eigen-face and Fisher face. if the threshold is passed the object willreturn -1.

After creating the objects the images that was saved in the earlier stage are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is also gathered at the same time by splitting the file name, and stored in another vector.By using **FaceRecognizer::train(NumpyImage, ID)** all three of the objects are trained. It must be noted that resizing the images were required only for Eigen-face and Fisher face, not for LBPH face. Afterwords, the configuration model is saved as an XML file using **FaceRecognizer::save(FileName)**. In the project all the three are trained and saved in one application for convenience. The flow chart below shows the data flow in this application.

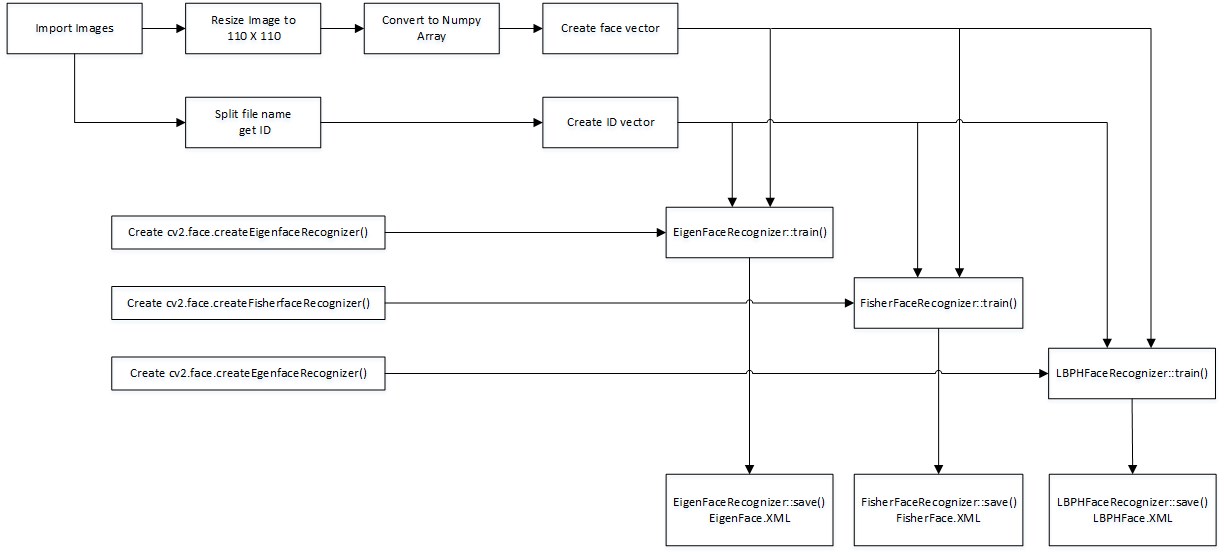


Figure 10: Flowchart of the training application

### The Face Recognition

For the face recognition application first the required face detection object is created using the desired threshold. faces must be detected in the image given. This is done using the same technique used for the image capture application. For each face detected, they are cropped and processed through the face recogniser object using **FaceRecognizer::predict()** which return the ID of the most similar class and the distance to the face that is called the confidence. This process is same for all the three algorithms and returns the ID from a class, or if the threshold level is passed returns -1. The confidence changes massively depending on the algorithm used and can be seen in the results section below. Finally, the text file with the names and the IDs is read and arranged in two lists. The way the list is constructed, the ID matches the index of the name and is used to display the name for each of the faces. If the ID is -1, it will return the unknown face string informing the user that the face cannot be recognised. The flow chart for the application is shown below.

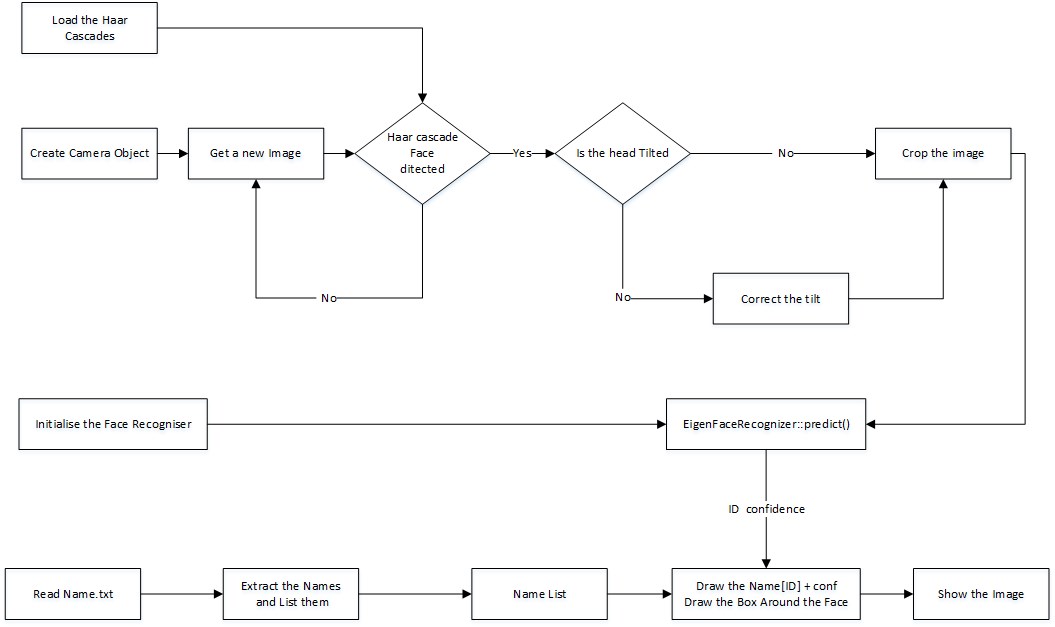


Figure 11: Flowchart of the face recognition application

# Results

Several tests were performed to find the optimal values for the algorithms and the performance. The images in the dataset at the time of these testes are shown below.



ID: 2 ID: 3 ID: 13 ID: 14 ID: 15 ID: 17 ID: 20 ID: 21

An application was written to record the IDs and confidence level for each algorithms on different photos of people. For this first task photos with single person is used for ease of analysing data. An Face recogniser algorithm was created, and trained 200 times and each time the photo was tested and the respond recorded. The application used a loop that incremented the number of components on each cycle. The results are analysed below.



Figure 12: Image Used for this test

The resulting ID change is plotted below in Figure 13. It can be seen at the start, when the components were 1, it identified the face as ID-17. The next 199 times it identified between ID-20 and ID-21, which is the same person.

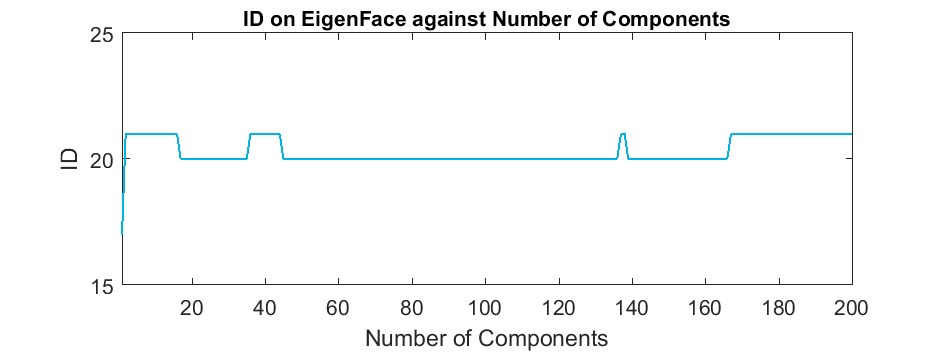


Figure 13: The ID from the face recogniser changes between two classes of the same person

The change of Confidence is plotted next, showing the increasing distance when the number of components are increased. From this plot it appears the best results can be obtained by keeping the number of components below 20.

Next, The same application is used to test the Fisher face algorithm. The photo is the same and the results are shown below.

The Results from Fisher face is more stable than of Eigen-face and is steady on ID-21, which is for the correct person.

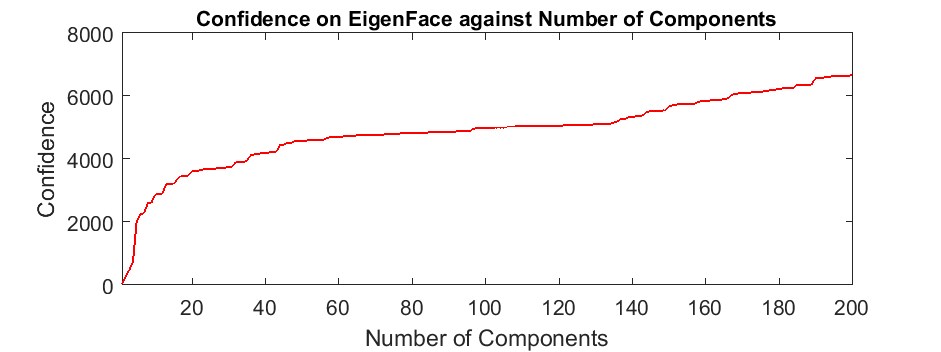


Figure 14: The Confidence from the face recogniser changes between two classes of the same person

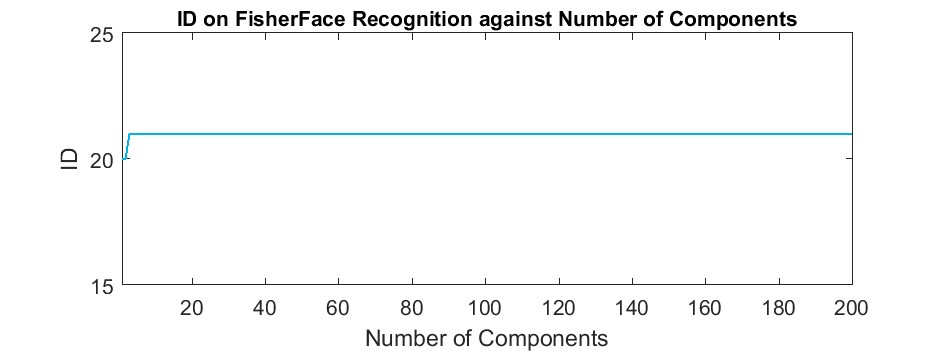


Figure 15: The Confidence from the face recogniser changes between two classes of the same person

The confidence level on Fisher increase until the number of components are around 10 and stay steady. It appears for Fisher face The best Number of components are less than 10.

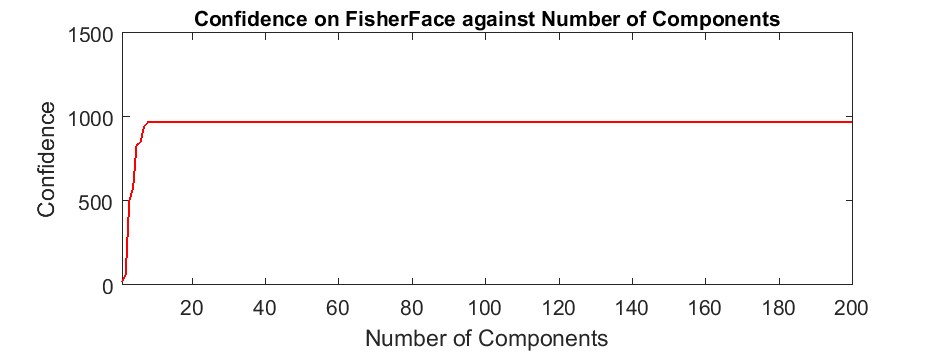


Figure 16: The Confidence from the face recogniser changes between two classes of the same person

Local binary pattern histogram has more than one parameter to change. All are incremented to the maximum limit and the results are below.

The first parameter to be incremented is the radius from the centre pixel. Since the image size is 110 X 110, maximum radius is 54. The ID is steady all the way to 54.

The returned confidence level is graphed below, against the radius. The confidence is fluctuating rapidly. The lowest confidence level (shortest distance to the dataset) is when the radius is at 1 and 48.

Since the best confidence is at 1 and 48, it was set to 1. Next parameter changed is the number of neighbours from 1 - 13. Further increase caused the computer to stall and force close the application. The returned ID is plotted below. The return is steady until 9 neighbours and it changed to ID-20. Although, it must be noted that this ID belongs to the same person.

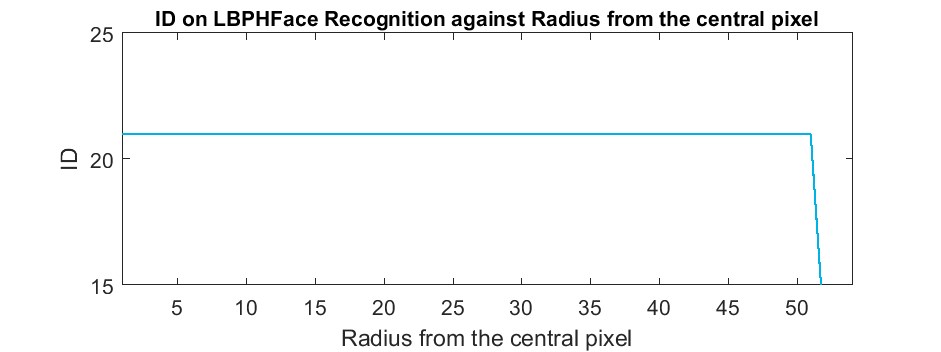


Figure 17: The ID returned from LBPH

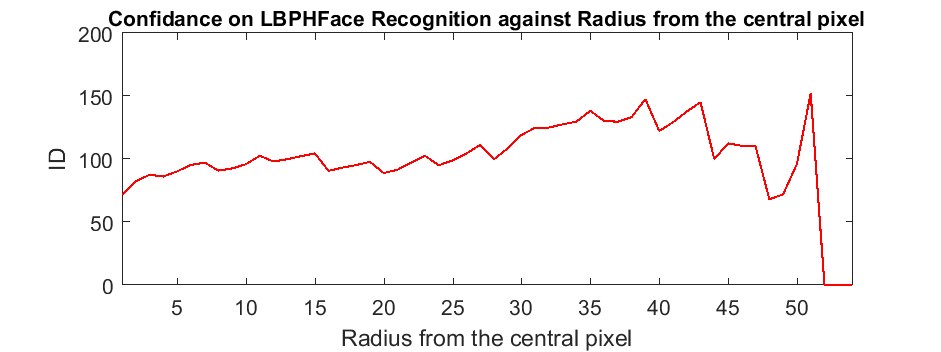
15 

Figure 18: The confidence returned from LBPH

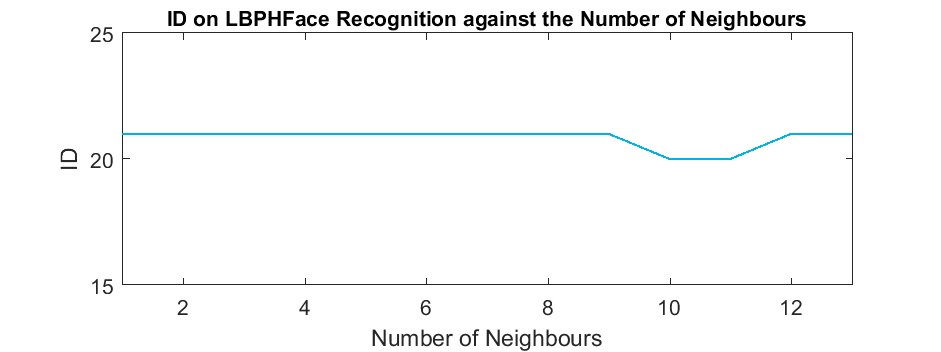


Figure 19: The ID returned from LBPH changing neighbours

The confidence level continuously increased while increasing the number of neighbours. Because of this, 1 neighbour will be included to the next test.

The last parameter to be changed on the LBPH is the number of cells. Although, X and Y axis cells can be changed independently, for this task both are given the same value.

The ID return is plotted below. The ID changed from ID-20 to ID -21 and stay steady.

The returned confidence is plotted below. The confidence level was extremely good when cells were less than 8 per side. The confidence start to increase rapidly after 10 ending up more than 1700 at 50.

## Recogniser on another subject

The above tests were performed on the author of the project and had two images in the dataset. For a secondary test authors’ colleague provided a photo which was 7 years old. The photo contains two people and the following plots only contain data on the colleague. The photo is shown below.

The following images show the ID and confidence from the tests. The Eigen-face, Fisher face and

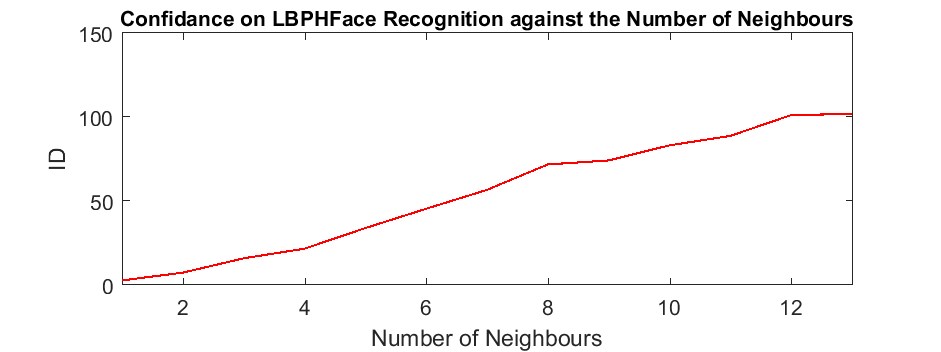


Figure 20: The Confidence returned from LBPH changing neighbours

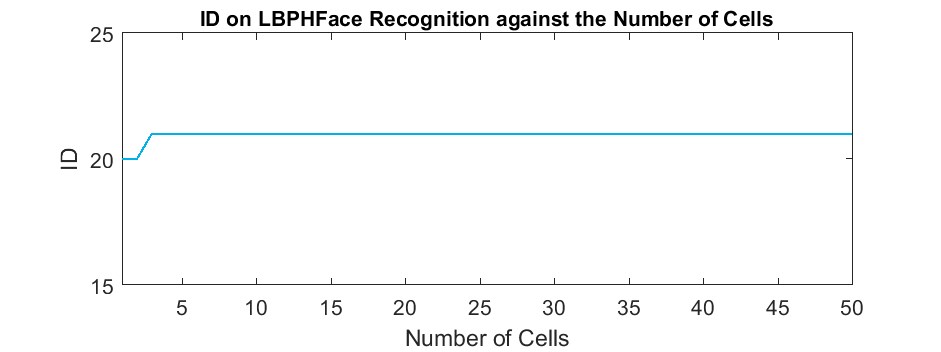


Figure 21: The ID returned from LBPH changing the number of cells

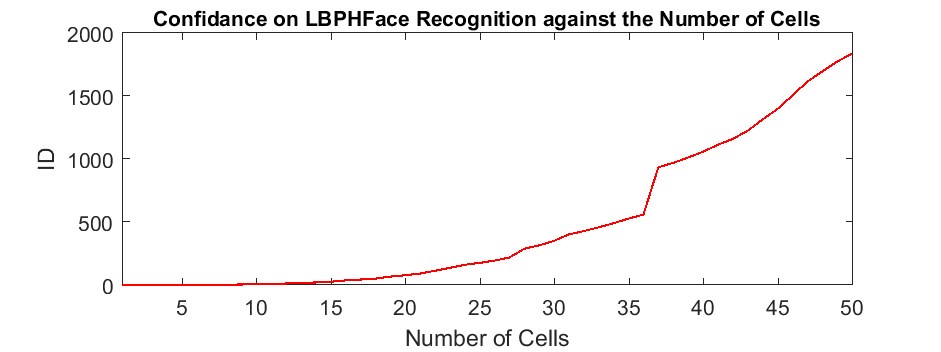


Figure 22: The confidence returned from LBPH changing the number of cells

three separate pairs for the LBPH are below. All the IDs are steady except at the moment of changing the radius from 37 pixels. The ID change to 21 and stay steady until 50 which is incorrect. The results were satisfactory, as this photo is 7 years younger than the dataset photos.

The Trainer (Trainer-All.py) application is used to train all three algorithms and recogniser (RecogniserAll.py ) is used for testing the results. The two images shown below are with and without proper calibration of the trainer. The values are as followed: Eigen-face = number of components 15, Threshold 4000

Fisher face = number of components 5, Threshold 400

LBPH = Radius 2, Neighbours 2, Cells 7, Threshold 15

The following images are before and after calibration from the web camera face recognition system. The web camera used for this is a Microsoft Lifecam.



Figure 23: A 7 years old photo of Samuel Westlake and his friend

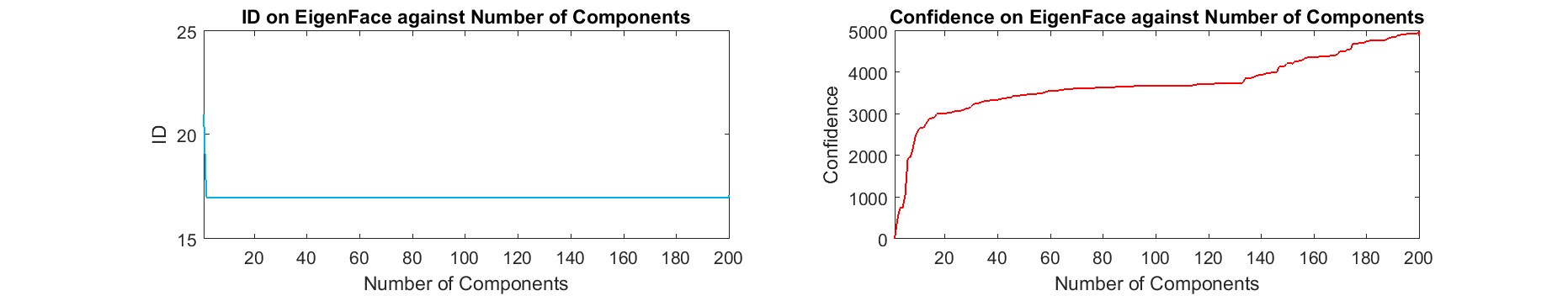


Figure 24: ID and Confidence for Eigen-face. Note the ID is at 17 which is correct

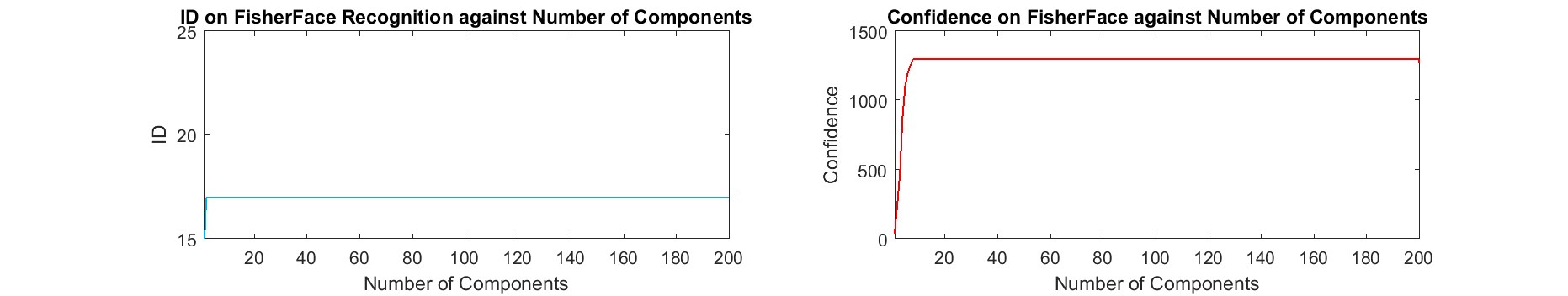


Figure 25: ID and Confidence for Fisher face. Note the ID is at 17 which is correct

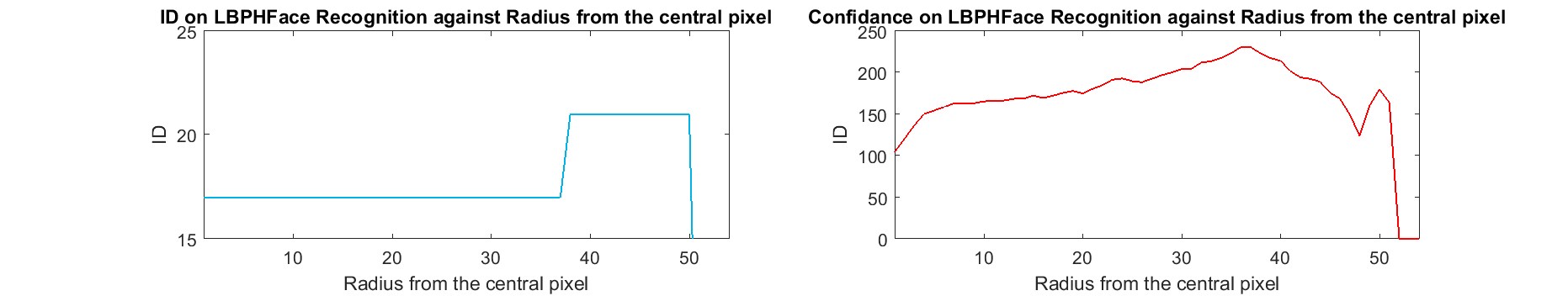


Figure 26: ID and Confidence for LBPA neighbours. Note the ID fluctuating at 37 pixels

# Conclusion

Using Haar-cascades for face detection worked extremely well even with subjects that had spectacles and long hair. It also performed in real time, without any noticeable frame lag. The face recognition was satisfactory as well, as can be noticed from the results. The algorithms can be used to detect faces as well. However, This makes the whole process considerably slower due to all the pixels that the algorithm

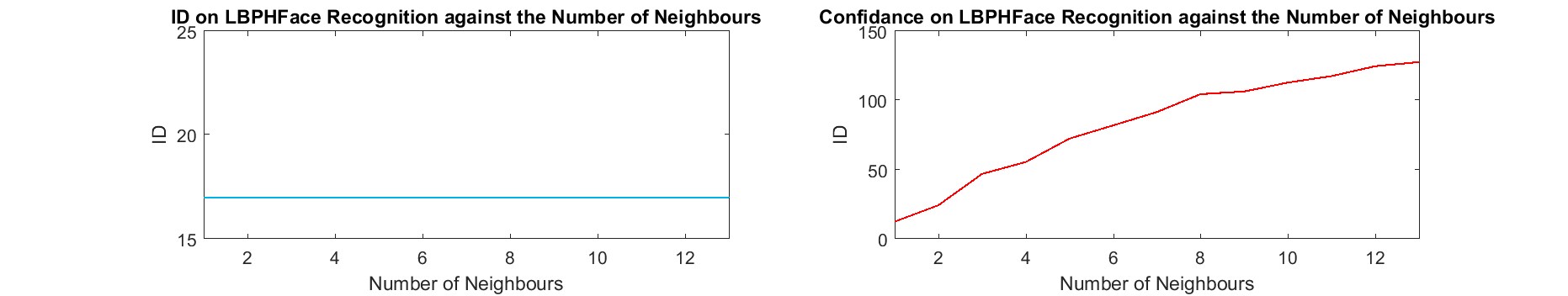
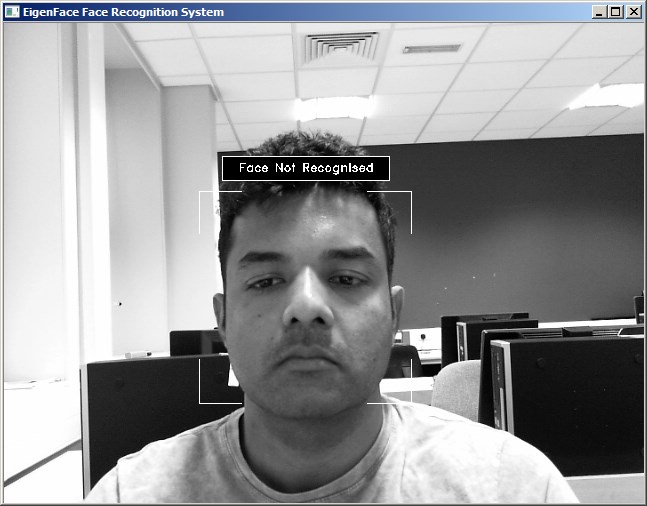
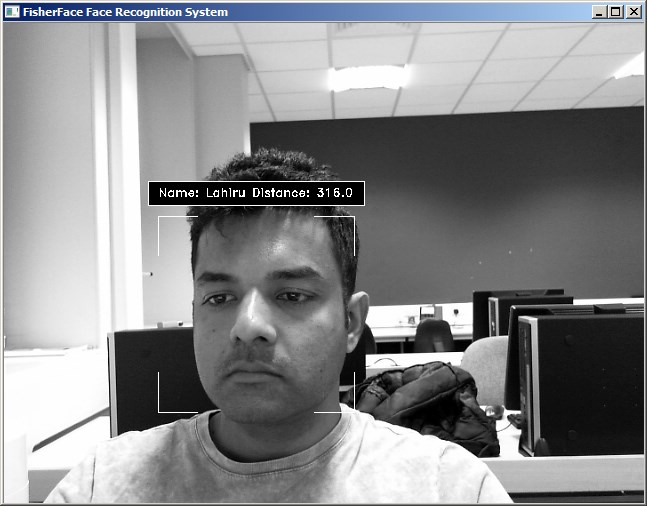
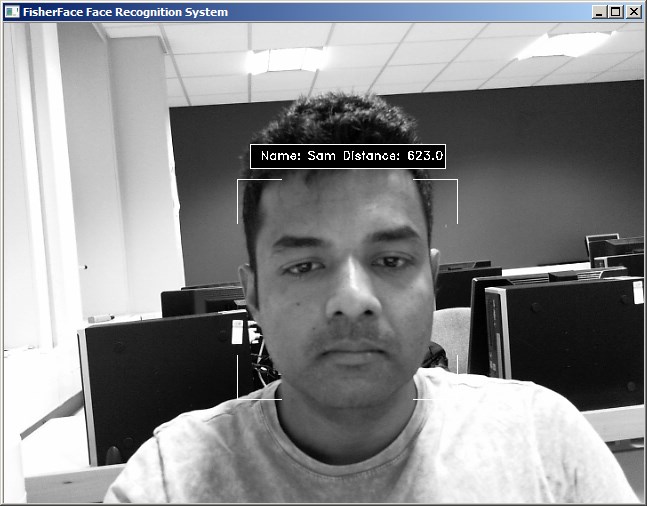


Figure 27: ID and Confidence for LBPA cells. Note the ID-17 is steady.



Before Calibration After Calibration

Figure 28: Before and after Eigen-face Calibration threshold: 4000



Before Calibration After Calibration

Figure 29: Before and after Fisher face Calibration threshold: 400

is required to sift through. By using the Haar-cascades to get the face area and passing it to the face recognition algorithm, excellent speeds were archived.

The earlier attempts on Eigen-face and Fisher face was disappointing since the LBPH alone recognised a face. This was soon rectified by creating the test application for all three algorithms. After analysing the data, the system was trained with parameters to minimise the confidence while retaining the accuracy of the ID. When tested with the new set by using video input as well as images, the system performed with satisfying accuracy. It must be noted that at times the Eigen-face and Fisher face algorithms varied between two IDs. However, LBPH algorithm performed without fail after training.

Regarding the time it took to train Eigen-face with 421 images over 200 times with increasing components took over 45 minutes and created a file 140MB large. The Fisher face took approximately 30 minutes and created a file 3MB large. The LBPH algorithm took slightly over 3 minutes to train through all



Before Calibration After Calibration

Figure 30: Before and after LBPH face Calibration threshold: 15



Figure 31: After calibration of Eigen-face the author and Mr. Westlake. Notice both faces are detected successfully

the parameters and created a file 41.5MB large. However it must be noted that these parameters are difference from that of Eigen-face and Fisher face parameters.

The speed of the three algorithms varies as well although LBPH provides the most accurate predictions, it is the slowest from the three. Fisher provides the fastest results while Eigen-face is in the middle. Considering all factors, Fisher face recogniser combined with Haar-cascades can be used for applications that require fast processing and relative accuracy. An example is a system to identify known troublemakers in a mall or a super market. For high accuracy but slower processing, LBPH with Haar-cascades can be implemented. For example, for attendance in a class where students walk past the camera relativity slowly and provide the system to identify the subject within time.

# References

1. P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman. Eigenfaces vs. fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, Jul 1997.
2. Dong chen He and Li Wang. Texture unit, texture spectrum, and texture analysis. *IEEE Transactions on Geoscience and Remote Sensing*, 28(4):509–512, Jul 1990.
3. John G Daugman. Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2(7):1160–1169, 1985.
4. S Marˆcelja. Mathematical description of the responses of simple cortical cells. *JOSA*, 70(11):1297– 1300, 1980.
5. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.
6. P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–511–I–518 vol.1, 2001.
7. X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th International Conference on Computer Vision*, pages 32–39, Sept 2009.