



**Report**  
**on**  
**Face & Eye Detection**

**Supervised by:**

Supta Richard Philip  
Senior Lecturer, Department of CSE  
City University, Dhaka, Bangladesh

**Submitted by:**

Nilottom Ghosh (ID: 153402352)  
Nazmul Islam (ID: 161412331)  
Sadia Islam (ID: 161412321)  
Mehedi Hasan (ID: 161412328)

41th Batch, Department of CSE  
City University  
Dhaka, Bangladesh

15<sup>th</sup> July, 2019

# Table of Contents

<b>ABSTRACT .....</b>	<b>3</b>
<b>CHAPTER 1.....</b>	<b>4</b>
<b>1. INTRODUCTION.....</b>	<b>4</b>
1.1 THE HISTORY OF FACE RECOGNITION .....	4
1.2 PROBLEM STATEMENT: .....	5
1.3 PROPOSED SOLUTION:.....	5
1.3.1 FACE & EYES DETECTION USING HAAR-CASCADES .....	5
1.3.2 FACE RECOGNITION .....	7
• EIGENFACE .....	7
• FISHERFACE.....	8
• LOCAL BINARY PATTERN HISTOGRAM.....	9
<b>CHAPTER 2.....</b>	<b>10</b>
<b>2. BACKGROUND STUDY .....</b>	<b>10</b>
2.1 RELATED WORK .....	10
<b>CHAPTER 3.....</b>	<b>11</b>
<b>3. METHODOLOGY .....</b>	<b>11</b>
3.1 PRE-REQUISITES .....	11
3.2 IMPORTING IMAGES IN OPENCV .....	11
3.3 FACE DETECTION .....	12
3.4 FACE RECOGNITION PROCESS .....	12
3.5 COLLECTING THE IMAGE DATA.....	13
3.6 TRAINING THE CLASSIFIERS .....	13
3.7 THE FACE RECOGNITION .....	15
3.8 RESULTS.....	15
<b>DISCUSSION .....</b>	<b>16</b>
<b>CONCLUSION .....</b>	<b>17</b>
<b>REFERENCES.....</b>	<b>18</b>

## **Abstract**

Identifying a person with an image has been popularized through the mass media. However, it is less robust to fingerprint or retina scanning. This report describes the face detection and recognition mini-project undertaken for the visual perception and autonomy module at City University. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement those using Python. For face detection, Haar-Cascades were used and for face recognition Eigenfaces, Fisherfaces and Local binary pattern histograms were used. The methodology is described including flow charts for each stage of the system. Next, the results are shown including plots and screen-shots followed by a discussion of encountered challenges. The report is concluded with the authors' opinion on the project and possible applications.

# Chapter 1

## 1. Introduction

The following document is a report on the mini project for Robotic visual perception and autonomy. It involved building a system for face and eyes detection using several classifiers available in the open computer vision library (OpenCV). Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analyzed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision “whose face is it?” 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 the explanation of HAAR-cascades, Eigenface, Fisherface and Local binary pattern histogram (LBPH) algorithms. Next, the methodology and the results of the project are described. A discussion regarding the challenges and the resolutions are described. Finally, a conclusion is provided on the pros and cons of each algorithm and possible implementations.

### 1.1 The History of Face Recognition

Face recognition began as early as 1977 with the first automated system being introduced by Kanade using a feature vector of human faces. In 1983, Sirovich and Kirby introduced the principal component analysis (PCA) for feature extraction. Using PCA, Turk and Pentland Eigenface was developed in 1991 and is considered a major milestone in technology. Local binary pattern analysis for texture recognition was introduced in 1994 and is improved upon for facial recognition later by incorporating Histograms (LBPH). In 1996 Fisherface was developed using linear discriminant analysis (LDA) for dimensional reduction and can identify faces in different illumination conditions, which was an issue in Eigenface method. Viola and Jones introduced a face detection technique using HAAR cascades and ADABOOST. In 2007, a face recognition technique was developed by Naruniec and Skarbek using Gabor Jets that are similar to mammalian eyes. In This project, HAAR cascades are used for face detection and Eigenface, Fisherface and LBPH are used for face recognition.

## 1.2 Problem Statement:

The purpose of this report is to follow up on a 10-week project on face & eyes detection give insight on how feasible it is to use a face recognition attendance system in a University environment. The system should be built to be used for a prolonged period of time anywhere in the University campus where attendance would be tracked.

## 1.3 Proposed solution:

We proposed a system for face and eyes detection using several classifiers available in the open computer vision library (OpenCV). Face recognition is a non-invasive identification system and faster than other systems since multiple faces can be analyzed at the same time. The difference between face detection and identification is, face detection is to identify a face from an image and locate the face. Face recognition is making the decision “whose face is it?” using an image database. In this project both are accomplished using different techniques and are described below.

### 1.3.1 Face & Eyes Detection using Haar-Cascades

A Haar wavelet is a mathematical function that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognize signals with sudden transformations. An example is shown in figure 1. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different color intensities. These sets are used in Viola Jones face detection technique in 2001 and since then more patterns are introduced [10] for object detection as shown in figure 1.

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 detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in figure 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 in figure 3.

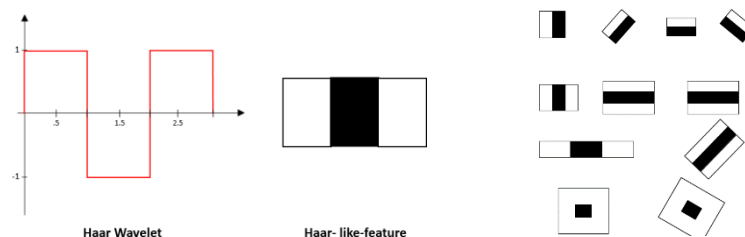


Figure 1: A Haar wavelet and resulting Haar-like features.

In this project, a similar method is used 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 is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable [7]. It can also reverse the direction of the classifier and get better results if necessary. Furthermore, Weight-update-steps can be updated

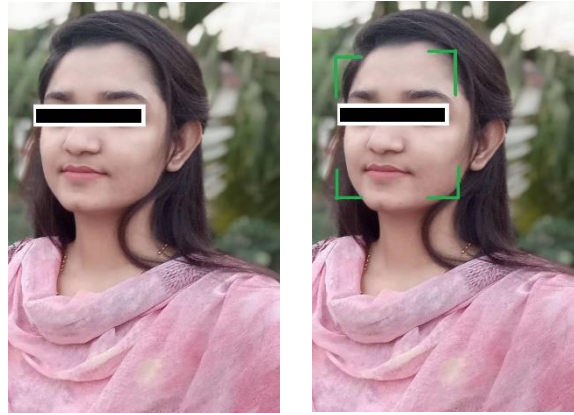


Figure 2: Several Haar-like-features matched to the features of authors face.

Only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to find different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time. Viola Jones [7] used a summed area table (an integral image) to compute the matches fast. First developed in 1984 [11], it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection. Using an integral image enables matching features with a single pass over the image.

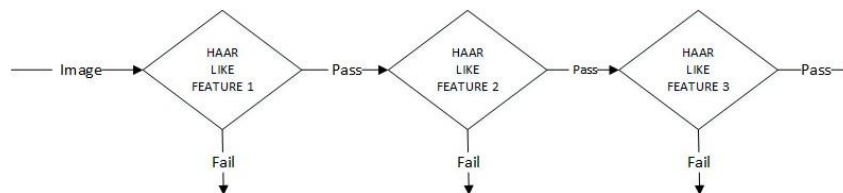


Figure 3: Haar-cascade flow chart

### 1.3.2 Face Recognition

The following sections describe the face recognition algorithms Eigenface, Fisherface, Local binary pattern histogram and how they are implemented in OpenCV.

#### • Eigenface

Eigenface is based on PCA that classify images to extract features using a set of images. It is important that the images are in the 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 image with  $n \times n$  pixels as shown in figure 4. Each row is concatenated to create a vector, resulting a

$1 \times n^2$  matrix. 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 an average human face. By subtracting the average face from each image vector unique features to each face are computed. In the resulting matrix, each column is a representation of the difference each face has to the average human face. A simplified illustration can be seen in figure 4.

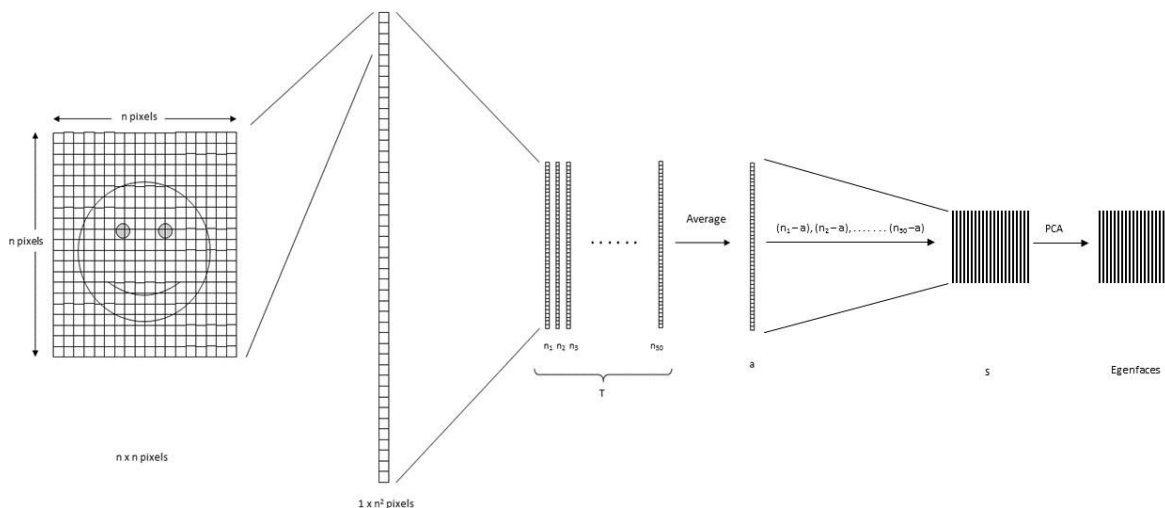


Figure 4: Pixels of the image are reordered to perform calculations for Eigenface

The next step is computing the covariance matrix from the result. 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 1<sup>st</sup> Eigen vector. 2<sup>nd</sup> Eigen vector is the

direction of the next highest variance, and it is in 90 degrees to the 1<sup>st</sup> vector. 3<sup>rd</sup> will be the next highest variation, and so on. Each column is considered an image and visualized, resembles a face and called Eigenfaces. When a face is required to be recognized, the image is imported, resized to match the same dimensions of the test data as mentioned above. By projecting extracted features on to each of the Eigenfaces, weights can be calculated. 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 analyzed 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.

### • Fisherface

Fisherface technique builds upon the Eigenface and is based on LDA derived from Ronald Fishers' linear discriminant technique used for pattern recognition. However, it uses labels for classes as well as data point information [6]. When reducing dimensions, PCA looks at the greatest 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 [6]. LDA maximises the ratio of the between-class scatter and within-class scatter matrices. Due to this, different lighting conditions in images has a limited effect on the classification process using LDA technique. Eigenface maximises the variations while Fisherface maximises the mean distance between and different classes and minimises variation within classes. This enables LDA to differentiate between feature classes better than PCA and can be observed in figure 5 [12]. 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.



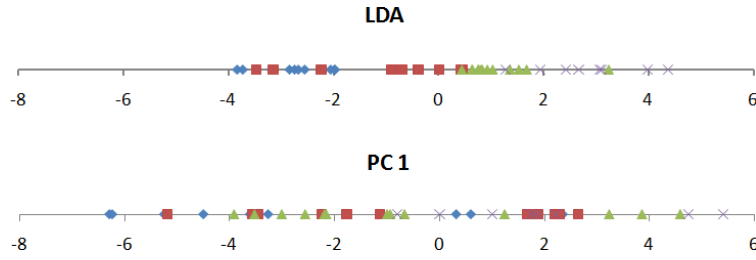


Figure 5: The first component of PCA and LDA. Classes in PCA looks more mixed than of LDA

## • Local Binary Pattern Histogram

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang [4]. The combination of LBP with histogram oriented gradients was introduced in 2009 that increased its performance in certain datasets [5]. 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 6. 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

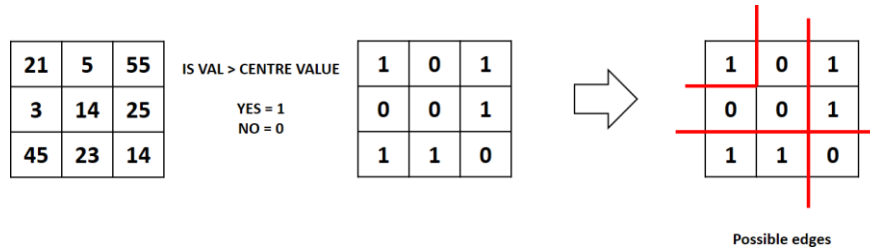


Figure 6: Local binary pattern histogram generating 8-bit number

is changed as in figure 7, the result is the same as before. Histograms are used in larger cells to find the frequency of occurrences of values making process faster. By analysing the results in the cell, edges can be detected as the values change. By computing the values of 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. Eigenface and Fisherface compute the dominant features of the whole training set while LBPH analyse them individually.

Increase Brightness yet, same results

42	10	110	IS VAL > CENTRE VALUE  YES = 1 NO = 0	1	0	1
6	28	50		0	0	1
90	46	28		1	1	0

Figure 7: The results are same even if brightness is changed

## Chapter 2

### 2. Background Study

Face Detection has been one of the hottest topics of computer vision for the past few years. This technology has been available for some years now and is being used all over the place. From cameras that make sure faces are focused before you take a picture, to Facebook when it tags people automatically once you upload a picture (before you did that manually remember?). Or some shows like CSI used them to identify “bad guys” from security footage (ENHANCE! – then insert crime pun) or even unlocking your phone by looking at it! We look at our phone, and it extracts your face from an image (the nerdy name for this process is face detection). Then, it compares the current face with the one it saved before during training and checks if they both match (its nerdy name is face recognition) and, if they do, it unlocks itself. As we see, this technology not only allows me to be any type of dog I want. People are getting pretty interested in it because of its ample applications.

#### 2.1 Related work

Face detection could also be done using Matlab , C, C++, C# or Java.

Matlab: Programming language constructed in its own frame work and IDE included in one improvement workspace. Matlab offers several tools (libraries) toward helping us to do technical programming promptly. For face processing plus recognition, it offers many already constructed

tools and archives suit for matrix calculation (image essentially a collection of pixel values) and programming, several tools for image processing by now constructed in it such as gray scaling, cropping, masking, rotating, convolution, plus many other kind of normal image processing requisite. The Matlab code implements the face recognition scheme. It usages the AT&T database.

Python: A choice for the beginner in image processing, recognition, motion detection. Python offers an easy to recognize over its simplicity plus suppleness of its language, plus there are several libraries constructed for Python to do several scientific calculation counting Image processing, recognition, as well as Motion detection.

**Since Python is bit easier than other languages for beginners so we are using python**

## **Chapter 3**

### **3. Methodology**

#### **3.1 Pre-requisites**

Hands-on knowledge of Numpy and Matplotlib is essential before working on the concepts of OpenCV. Make sure that you have the following packages installed and running before installing OpenCV.

Python

Numpy

Matplotlib

#### **3.2 Importing Images in OpenCV**

```
import numpy as np
```

```
import cv2
```

```
import matplotlib.pyplot as plt
```

```
%matplotlib inline
```

### 3.3 Face Detection

First stage was creating a face detection system using Haar-cascades. Although, training is required for creating new Haar-cascades, OpenCV has a robust set of Haar-cascades that was used for the project. Using face-cascades alone caused random objects to be identified and eye cascades were incorporated to obtain stable face detection. The flowchart of the detection system can be seen in figure 8. Face and eye.

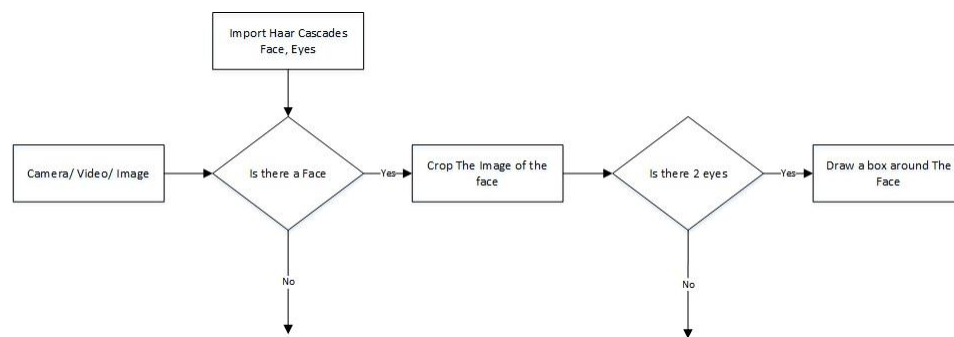


Figure 8: The Flow chart of the face detection application

Classifier objects are created using classifier class in OpenCV through the `cv2.CascadeClassifier()` and loading the respective XML files. A camera object is created using the `cv2.VideoCapture()` to capture images. By using the `CascadeClassifier.detectMultiScale()` object of various sizes are matched and location is returned. Using the location data, the face is cropped for further verification. Eye cascade is used to verify there are two eyes in the cropped face. If satisfied a marker is placed around the face to illustrate a face is detected in the location.

### 3.4 Face Recognition Process

For this project three algorithms are implemented independently. These are Eigenface, Fisherface and Linear binary pattern histograms respectively. All three can be implemented using OpenCV libraries. There are three stages for the face recognition as follows:

Collecting images IDs

Extracting unique features, classifying them and storing in XML files

Matching features of an input image to the features in the saved XML files and predict identity

### **3.5 Collecting the image data**

Collecting classification images is usually done manually using a photo editing software to crop and resize photos. Furthermore, PCA and LDA requires the same number of pixels in all the images for the correct operation. This time consuming and a laborious task is automated through an application to collect 50 images with different expressions. The application detects suitable expressions between 300ms, straightens any existing tilt and save them.

Application starts with a request for a name to be entered to be stored with the ID in a text file. The face detection system starts the first half. However, before the capturing begins, the application check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the position of the eyes are analyzed. If the head is tilted, the application automatically corrects the orientation. These two additions were made considering the requirements for Eigenface algorithm. The Image is then cropped and saved using the ID as a filename to be identified later. A loop runs this program until 50 viable images are collected from the person. This application made data collection efficient.

### **3.6 Training the Classifiers**

OpenCV enables the creation of XML files to store features extracted from datasets using the FaceRe- cognizer class. The stored images are imported, converted to grayscale and saved with IDs in two lists with same indexes. FaceRecognizer objects are created using face recogniser class. Each recogniser can take in parameters that are described below:

#### **cv2.face.createEigenFaceRecognizer()**

1. Takes in the number of components for the PCA for crating Eigenfaces. OpenCV documentation mentions 80 can provide satisfactory reconstruction capabilities.
2. Takes in the threshold in recognizing faces. If the distance to the likeliest Eigenface is above this threshold, the function will return a -1, that can be used state the face is unrecognizable

## **cv2.face.createFisherfaceRecognizer()**

1. The first argument is the number of components for the LDA for the creation of Fisherfaces. OpenCV mentions it to be kept 0 if uncertain.
2. Similar to Eigenface threshold. -1 if the threshold is passed.

## **cv2.face.createLBPH FaceRecognizer()**

1. The radius from the centre pixel to build the local binary pattern.
2. The Number of sample points to build the pattern. Having a considerable number will slow down the computer.
3. The Number of Cells to be created in X axis.
4. The number of cells to be created in Y axis.
5. A threshold value similar to Eigenface and Fisherface. if the threshold is passed the object will return -1

Recognizer objects are created and images are imported, resized, converted into numpy arrays and stored in a vector. The ID of the image is gathered from 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 Eigenface and Fisherface, not for LBPH. Next, the configuration model is saved as a XML file using FaceRecognizer.save(FileName). In this project, all three are trained and saved through one application for convenience.

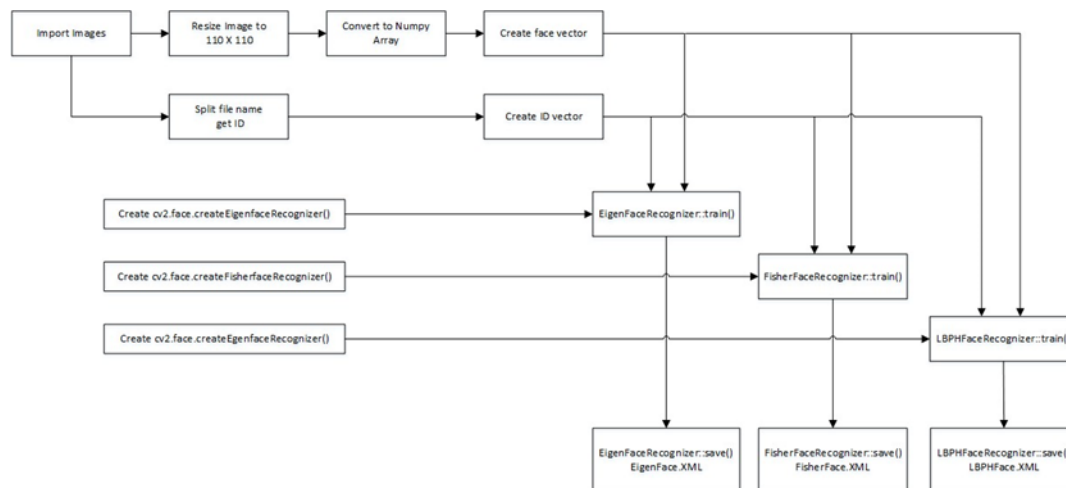


Figure 9: Flowchart of the training application

### 3.7 The Face Recognition

Face recogniser object is created using the desired parameters. Face detector is used to detect faces in the image, cropped and transferred to be recognised. This is done using the same technique used for the image capture application. For each face detected, a prediction is made using FaceRecognizer.predict() which return the ID of the class and confidence. The process is same for all algorithms and if the confidence his higher than the set threshold, ID is -1. Finally, names from the text file with IDs are used to display the name and confidence on the screen. If the ID is -1, the application will print unknown face without the confidence level.

### 3.8 Results

Three applications were written to iter- ate through the parameters of each algorithm. On each iteration, the algorithm is trained using different parameters and tested against a photo. The resulting data is plotted at the after finishing the tests. Images after Face & Eye Detection:

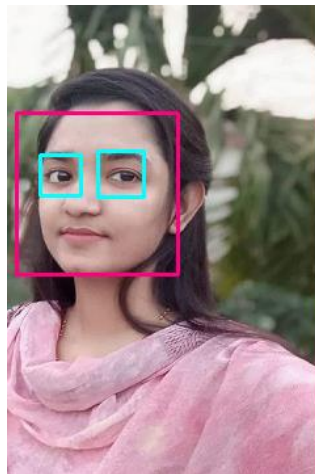


Figure 10: After Processing Image

## **Discussion**

Early attempts on Eigenface and Fisherface was disappointing since the LBPH alone recognised a face. By designing an application to test the algorithms, and after calibration with new data, both algorithms performed well. The tester applications also allowed accurate threshold settings. Another problem was people tilting their head when images taken for the data. This was fixed with an application that identifies the locations of the eyes and rotate the image to correct the off-set. It was noticed that some early-stage images in the data-set different brightness's. To resolve this, before taking an image, the brightness was averaged to prevent dark images. These changes to the system improved the performance noticeably.



## **Conclusion**

This paper describes the mini-project for visual perception and autonomy module. Next, it explains the technologies used in the project and the methodology used. Finally, it shows the results, discuss the challenges and how they were resolved followed by a discussion. Using Haar-cascades for face detection worked extremely well even when subjects wore spectacles. Real time video speed was satisfactory as well devoid of noticeable frame lag. Considering all factors, LBPH combined with Haar-cascades can be implemented as a cost effective face recognition platform. An example is a system to identify known troublemakers in a mall or a supermarket to provide the owner a warning to keep him alert or for automatic attendance taking in a class.

## References

- [1] Takeo Kanade. Computer recognition of human faces, volume 47. Birkh user Basel, 1977.
- [2] Lawrence Sirovich and Michael Kirby. Low-dimensional procedure for the characterization of human faces. *Josa a*, 4(3):519–524, 1987.
- [3] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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.
- [8] 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.
- [9] S Mar elja. Mathematical description of the responses of simple cortical cells. *JOSA*, 70(11):1297– 1300, 1980.
- [10] Rainer Lienhart and Jochen Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–I. IEEE, 2002.
- [11] John P Lewis. Fast template matching. In *Vision interface*, volume 95, pages 15–19, 1995.
- [12] Meng Xiao He and Helen Yang. Microarray dimension reduction, 2009.