VII TH SEMESTER MINOR PROJECT

REPORT ON

"DOCTOR'S FACE RECOGNATION SYSTEM"

BY

BHARGAV RAM - 301403317025

ANCHAL GUPTA - 301403317012

TANMAY CHANDRAKAR -301403317088

SWAPNIL GAJBHIYE -301403317084

Under the guidance of ***Ms.MADHURI GUPTA MAM***

DEPARTMENT OF INFORMATION TECHNOLOGY

SHRI SHANKRACHARYA GROUP OF INSTITUTION,BHILAI

CHHATISHGHARH, NOVEMBER-DECEMBER 2020

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 project describes the face detection and recognition mini-project undertaken for the visual perception and autonomy module at Plymouth university. It reports the technologies available in the Open-Computer-Vision (OpenCV) library and methodology to implement them 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.

1.1. Project Overview

The goal of this project is to detect and locate criminal faces in a color image. A set of seven training images were provided for this purpose. The objective was to design and implement a face detector that will detect criminal faces in an image similar to the training images.

The problem of face detection has been studied extensively. A wide spectrum of techniques have been used including color analysis, template matching, neural networks, support vector machines (SVM), maximal rejection classification and model based detection. However, it is difficult to design algorithms that work for all illuminations, face colors, sizes and geometries, and image backgrounds. As a result, face detection remains as much an art as science.

Our method uses rejection based classification. The face detector consists of a set of weak classifiers that sequentially reject non-face regions. First, the non-skin color regions are rejected using color segmentation.
A set of morphological operations are then applied to filter the clutter resulting from the previous step. The remaining connected regions are then classified based on their geometry and the number of holes. Finally, template matching is used to detect zero or more faces in each connected region. A block diagram of the detector is shown in Figure 1.1.

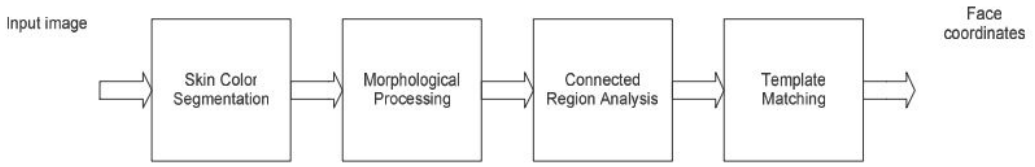


Figure 1.1. Block diagram of face detector

1.2 SOFTWARE AND HARDWARE

Hardware specification

Processor:	ALL PROCESSOR
RAM:	1 GB or more
Hard Disk:	5 GB free space

Software Requirement

Operating System:	Window 7/8/10
IDE Software:	Visual Studio
Database technology:	MySql 2005

2.1 The History of Face Recognition

During 1964 and 1965, Bledsoe, along with Helen Chan and Charles Bisson, worked on using the computer to recognize human faces (Bledsoe 1966a, 1966b; Bledsoe and Chan 1965). He was proud of this work, but because the funding was provided by an unnamed intelligence agency that did not allow much publicity, little of the work was published. Based on the available references, it was revealed that the Bledsoe's initial approach involved the manual marking of various landmarks on the face such as the eye centers, mouth, etc., and these were mathematically rotated by computer to compensate for pose variation. The distances between landmarks were also automatically computed and compared between images to determine identity.

Given a large database of images (in effect, a book of mug shots) and a photograph, the problem was to select from the database a small set of records such that one of the image records matched the photograph. The success of the method could be measured in terms of the ratio of the answer list to the number of records in the database. Bledsoe (1966a) described the following difficulties:

This project was labeled man-machine because the human extracted the coordinates of a set of features from the photographs, which were then used by the computer for recognition. Using a graphics tablet (GRAFACON or RAND TABLET), the operator would extract the coordinates of features such as the center of pupils, the inside corner of eyes, the outside corner of eyes, point of widows peak, and so on. From these coordinates, a list of 20 distances, such as the width of mouth and width of eyes, pupil to pupil, were computed. These operators could process about 40 pictures an hour. When building the database, the name of the person in the photograph was associated with the list of computed distances and stored in the computer. In the recognition phase, the set of distances was compared with the corresponding distance for each photograph, yielding a distance between the photograph and the database record. The closest records are returned.

Because it is unlikely that any two pictures would match in head rotation, lean, tilt, and scale (distance from the camera), each set of distances is normalized to represent the face in a frontal orientation. To accomplish this normalization, the program first tries to determine the tilt, the lean, and the rotation. Then, using these angles, the computer undoes the effect of these transformations on the computed distances. To compute these angles, the computer must know the three-dimensional geometry of the head. Because the actual heads were unavailable, Bledsoe (1964) used a standard head derived from measurements on seven heads.

After Bledsoe left PRI in 1966, this work was continued at the Stanford Research Institute, primarily by Peter Hart. In experiments performed on a database of over 2000 photographs, the computer consistently outperformed humans when presented with the same recognition tasks (Bledsoe 1968). Peter Hart (1996) enthusiastically recalled the project with the exclamation, "It really worked!"

By about 1997, the system developed by Christoph von der Malsburg and graduate students of the University of Bochum in Germany and the University of Southern California in the United States outperformed most systems with those of Massachusetts Institute of Technology and the University of Maryland rated next. The Bochum system was developed through funding by the United States Army Research Laboratory. The software was sold as ZN-Face and used by customers such as Deutsche Bank and operators of airports and other busy locations. The software was "robust enough to make identifications from less-than-perfect face views. It can also often see through such impediments to identification as mustaches, beards, changed hairstyles and glasses—even sunglasses".

In 2006, the performance of the latest face recognition algorithms was evaluated in the Face Recognition Grand Challenge (FRGC). High-resolution face images, 3-D face scans, and iris images were used in the tests. The results indicated that the new algorithms are 10 times more accurate than the face recognition algorithms of 2002 and 100 times more accurate than those of 1995. Some of the algorithms were able to outperform human participants in recognizing faces and could uniquely identify identical twins.

U.S. Government-sponsored evaluations and challenge problems have helped spur over two orders-of-magnitude in face-recognition system performance. Since 1993, the error rate of automatic face-recognition systems has decreased by a factor of 272. The reduction applies to systems that match people with face images captured in studio or mugshot environments. In Moore's law terms, the error rate decreased by one-half every two years.

Low-resolution images of faces can be enhanced using face hallucination

3.1 Problem And Solution

The problem that has been identified that the face recognition process has been very slow due to system factors. Most facial recognizers do not save the database of multiple faces as there are storage linkage issues in most of the recognition system.

Most of the facial recognition devices are quite expensive to bear in such a large scale so that everyone could be benefited.

Facial recognition devices needs to be integrated properly in an institution and according to studies most of the face recognition devices has been needing a yearly or half- yearly maintenance which again costs quite a lot for a big organization.

So, we decided to create a system of face recognition in such a way which would need minimal maintenance and easy to install so that a specific skilled man is not required to perform the overall operation of the facial recognition system. A face recognition system which eliminates the manual mistakes and control the overall system. Since the face reorganization controls every single event electronically therefore it reduces the possibility of error.

A system so easy to integrate and operate such that it could be used in each and every street of the country to identify criminals or wanted people and it could be installed or operated from each and every specified police station.

Overall, such a system is developed by our team which each and every person can operate very easily in a huge scale such that the organization does not have to bear extra charges of the skilled person and can be used to detect criminals all over the area.

4.1 Face Detection Using Haar-Cascades Method

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 recognize signals with sudden transformations. By combining several wavelets, a cascade can be created that can 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 introduced for object detection as shown in figure 4.1.

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 4.2.

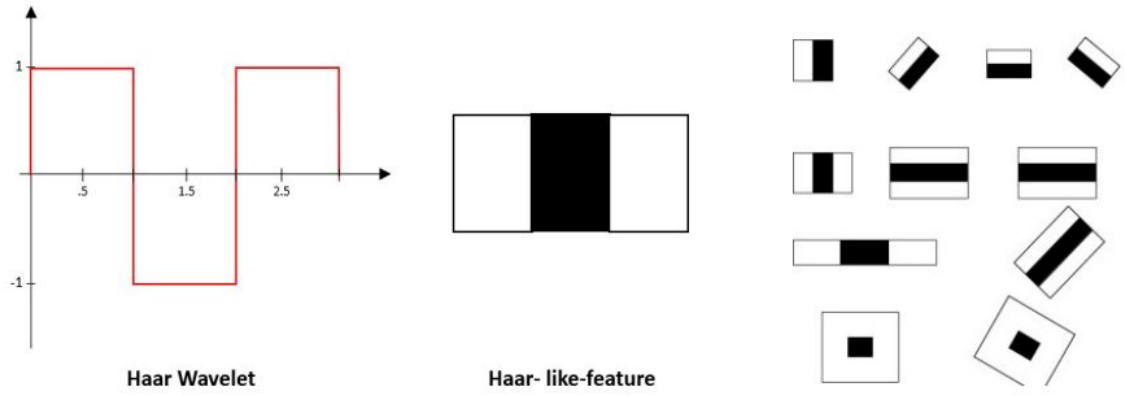


Figure 4.1: A Haar wavelet and resulting Haar-like features.

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. It can also reverse the direction of the classifier and get better results if necessary. 2. Furthermore, Weight-update-steps can be updated

Figure 4.2: Several Haar-like-features matched to the features of 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 used a summed area table (an 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 an integral image enables matching features with a single pass over the image.

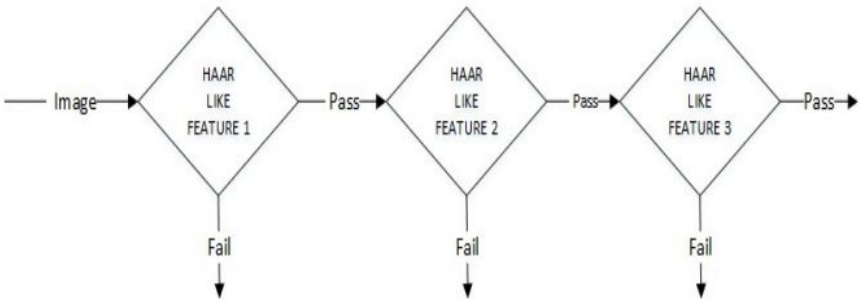


Figure4.3:Haar-cascadeflowchart

* Face Recognition

Histogram the following sections describe the face recognition algorithms Eigenface, Fisherface, Local binary pattern and how they are implemented in Open CV.

4.2 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 x n pixels as shown in figure 4. Each raw is concatenated to create a vector, resulting 1x 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 4.4.

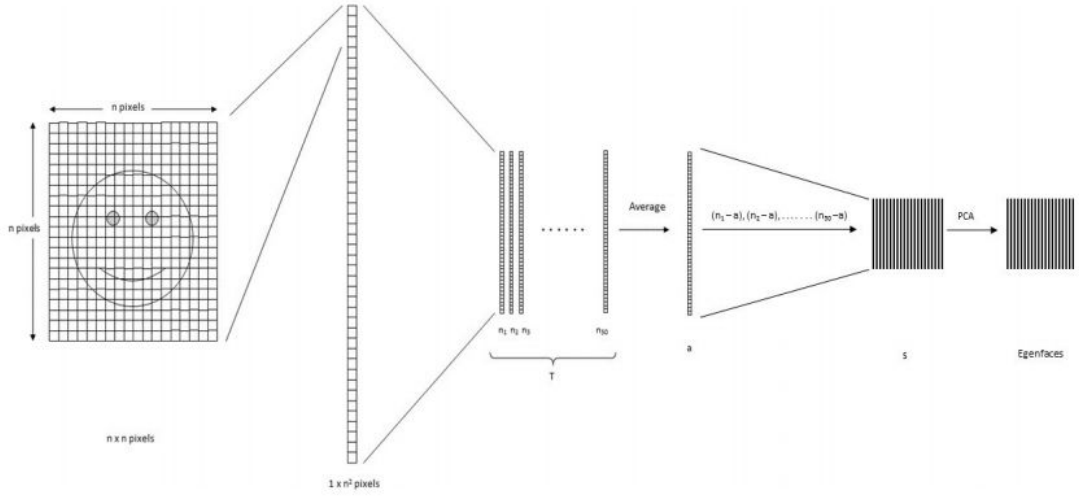


Figure 4.4: Pixels of the image are reordered to perform calculations for Eigenface

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 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 visualized, resembles a face and called Eigenface. 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 Eigenface, 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 calculated and cannot be effectively classified. This makes to different lighting 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 conditions poses a problem in matching the features as they can change dramatically.

* Computing the Eigenfaces

Before generating eigenfaces, face images are normalized to line up the eyes and mouths, then all resampled at the same pixel resolution. Eigenfaces are then extracted out of the image data by means of principal component analysis (PCA) in the following manner:

1. Given M face images with the size of $h \times w$, each image is transformed into a vector of size $D(=hw)$ and placed into the set
   $\{\Gamma 1, \Gamma 2, \cdots, \Gamma M\}$.
   The face images should be appropriately scaled and aligned, and the backgrounds (and possibly non-face areas such as hair and neck) should be constant or removed.
2. Each face differs from the average by the vector $\Phi i = \Gamma i - \Psi$, where the average face is defined by $\Psi = 1M \sum Mi = 1 \Gamma i$.
3. The covariance matrix $\mathbf{C} \in \mathrm{R}D \times D$ is defined as
   $\mathbf{C} = 1M \sum i = 1M \Phi i \Phi \mathrm{T} i = \mathbf{A}\mathbf{A}\mathrm{T}$,
   where $\mathbf{A} = \{\Phi 1, \Phi 2, \cdots, \Phi M\} \in \mathrm{R}D \times M$.
4. Determining the eigenvectors of $\mathbf{C}$ is an intractable task for typical image sizes when $D \gg M$. However, to efficiently compute the eigenvectors of $\mathbf{C}$, one may first compute

the eigenvectors of the much-smaller $M \times M$ matrix $\mathbf{A}^T\mathbf{A}$. The eigenvector and eigenvalue matrices of $\mathbf{A}^T\mathbf{A}$ are defined as

$\mathbf{V}=\{\mathbf{v}1,\mathbf{v}2,\cdots\mathbf{v}r\}$

and $\Lambda=diag\{\lambda 1,\lambda 2,\cdots\lambda r\}$, $\lambda 1\geq\lambda 2\geq\cdots\lambda r>0$, where r is the rank of $\mathbf{A}$. Note that eigenvectors corresponding to eigenvalues of zero have been discarded.

5. The eigenvalue and eigenvector matrices of $\mathbf{C}$ are Λ and $\mathbf{U}=\mathbf{A}\mathbf{V}\Lambda-1/2$, where $\mathbf{U}=\{\mathbf{u}i\}$ is the collection of eigenfaces.

* PCA and SVD

This section discusses the close relationship between PCA and SVD. PCA seeks to find k "principal axes," which define an orthonormal coordinate system that can capture most of the variance in data. For example, given the data matrix $\mathbf{A}$, the covariance matrix (outer product matrix) can be written as $\mathbf{C}=\mathbf{A}\mathbf{A}^T$, and the principal components $\mathbf{u}i$

$\mathbf{A}\mathbf{A}^T\mathbf{u}i=\lambda i\mathbf{u}i$

where $\mathbf{u}i$ are the eigenvectors of $\mathbf{A}\mathbf{A}^T$ associated with eigenvalues λi . When $\mathbf{A}\mathbf{A}^T$ is too large to be efficiently decomposed, one can circumvent this by computing the inner product matrix $\mathbf{A}^T\mathbf{A}$ as in Step 4,

$\mathbf{A}^T\mathbf{A}\mathbf{v}i=\lambda i\mathbf{v}i$

where $\mathbf{v}i$ are the eigenvectors of $\mathbf{A}^T\mathbf{A}$ associated with eigenvalues λi . Note that λi is nonnegative, and $\mathbf{u}i=\lambda-0.5i\mathbf{A}\mathbf{v}i$ for those nonzero eigenvalues. This explains Step 5, which is written in the matrix form.

The SVD of $\mathbf{A}$ is the following factorization:

$\mathbf{A}=\mathbf{U}\Delta\mathbf{V}^T=\sum\delta i\mathbf{u}i\mathbf{v}^Ti$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices, the diagonal matrix Δ contains the singular value δi , which could be positive, zero, or even negative. Connecting to PCA, $\mathbf{U}$ and $\mathbf{V}$ are the eigenvector matrices of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$, and $\delta 2i=\lambda i$ for any i .

* Using Eigenfaces in Face Processing

The eigenfaces span an *m*-dimensional subspace of the original image space by choosing the subset of eigenvectors $\mathbf{U}\hat{}=\{\mathbf{u}1,\cdots,\mathbf{u}m\}$ associated with the *m* largest eigenvalues. This results in the so-called *face space*, whose origin is the average face, and whose axes are the eigenfaces (see Figure 4.5). To perform face detection or recognition, one may compute the distance within or from the face space.

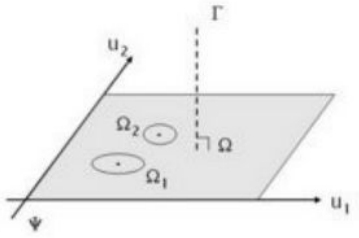


Figure 4.5: Visualization of a 2D face space, with the axes representing two Eigenfaces.

Figure 4.6: Original images (Row 1) and their projections into the face space (Row 2).

* Face detection

Because the face space (the subspace spanned by the eigenfaces) defines the space of face images, face detection can be considered as detecting image patches that lie close to the face space. In other words, the projection distance δ should be within some threshold $\theta\delta$. The point-to-space distance δ is the distance between the face image and its projection onto the face space, and it can be computed as

$\delta=\|(\mathbf{I}-\mathbf{U}\hat{}\mathbf{U}\hat{}\mathsf{T})(\Gamma-\Psi)\|$

where **I** is the identity matrix. As shown in Figure 4.6, the distance between an image (above)

and its face space projection (below) is much smaller for a face than for a nonface (tree) image.

* Face recognition

A new face Γ is projected into the face space by $\Omega=\mathbf{U}\hat{}\top(\Gamma-\Psi)$, where $\mathbf{U}\hat{}$ is the set of significant eigenvectors. Note that the weight vector Ω is the representation of the new face in face space. One simple way to determine which face class Γ belongs to is minimizing the Euclidean distance

$\epsilon k=\|\Omega-\Omega k\|$

where Ωk is the weight vector representing the kth face class. The face Γ is considered as belonging to class k if the the minimum ϵk is smaller than some predefined threshold $\theta\epsilon$; otherwise, it is classified as unknown. Figure 3.6 illustrates the projection and recognition by visualizing face space as a plane.

Eigenface Extensions

The Eigenface approach was an important step towards appearance-based recognition in computer vision. However, the method is sensitive to variation in lighting, scale, pose, facial expression, and occlusion. To work well, the face must be presented in frontal view, at the appropriate scale, in a similar lighting condition, in a defined (typically neutral) expression, and unoccluded. To handle the variety of challenges faced by real face recognition systems, many modifications and extensions have been proposed since the original work.

The idea of *View-Based Eigenfaces* (Pentland et al. 1994) is to build a set of separate eigenspaces, each capturing the variation of face images in a common view. When classifying a new face Γ , the first step is to determine its view. This can be accomplished by computing the minimum distance δl to individual eigenspace l .

$\delta l=\|(\mathbf{I}-\mathbf{U}\hat{}l\mathbf{U}\hat{}\top l)(\Gamma-\Psi l)\|$

where Ψl and $\mathbf{U}\hat{}l$ are the average face and eigenfaces of view l . Once the proper view is determined, the image is projected onto the corresponding eigenfaces, and then recognized as in SectionFigure 4.6.

Another variation of the eigenface approach, called *Eigenfeatures*, or *Modular Eigenfaces*, is to compute eigenspaces for facial features, yielding eigeneyes, eigennoses, and eigenmouths. One may wish to improve face recognition performance by incorporating the eigenfeatures with the eigenfaces. This can be viewed as representing face images in a layered approach, where a coarse representation (i.e., eigenface) of the whole face is augmented by a fine representation (i.e., eigenfeature) of facial features. The eigenface approach has also been used to represent local face patches, rather than the whole face, which are then combined in a multi-classifier approach.

The idea of Eigenfaces has been extended very widely, for example, Fisherfaces (Belhumeur et al. 1997), Kernel Eigenfaces (Yang 2002), and others. Being aware that PCA is optimal for pattern representation, rather than classification, Fisherfaces seek the linear discriminants of the face distribution based on the Fisher criterion. Since PCA only captures the second order statistics (i.e., covariance matrix) of face images, Kernel Eigenfaces were developed to capture the higher order information. This can be accomplished by using the Kernel Principal Component Analysis (KPCA). Finally, the Eigenfaces technique has been even applied to other areas, such as eigenvoice (Kuhn et al. 2000) and eigengait (Huang et al. 1999).

4.3 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. 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 maximize the difference between the mean of the classes normalized by their variance.

LDA maximizes 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 maximizes the variations while Fisherface maximizes the mean distance between and different classes and minimizes variation within classes.

This enables LDA to differentiate between feature classes better than PCA and can be observed in figure 4.7. 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.

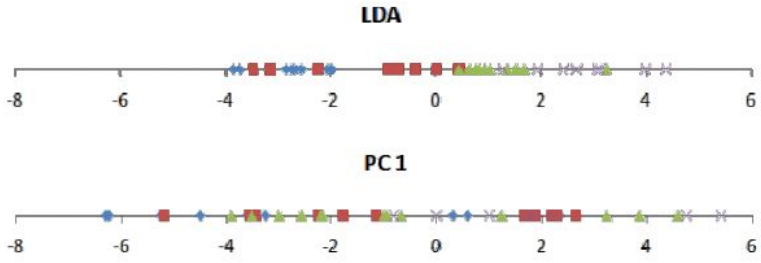


Figure 4.7: The first component of PCA and LDA. Classes in PCA looks more mixed than of LDA

* Computing the Fisherfaces

The theoretical argument given in the preceding section shows how to obtain the Bayes optimal solution for the 2-class homoscedastic case. In general, we will have more than 2-classes. In such a case, we reformulate the above stated problem as that of minimizing within-class differences and maximizing between-class distances.

Within class differences can be estimated using the within-class scatter matrix, given by

$\mathbf{S}w=\sum Cj=1\sum nji=1(\mathbf{x}ij-\mu j)(\mathbf{x}ij-\mu j)T$,

where $\mathbf{x}ij$ is the ith sample of class j, μj is the mean of class j, and nj the number of samples in class j.

Likewise, the between class differences are computed using the between-class scatter matrix,

$\mathbf{S}b=\sum Cj=1(\mu j-\mu)(\mu j-\mu)T$,

where μ represents the mean of all classes.

We now want to find those basis vectors $\mathbf{V}$ where $\mathbf{S}w$ is minimized and $\mathbf{S}b$ is maximized, where $\mathbf{V}$ is a matrix whose columns $\mathbf{v}i$ are the basis vectors defining the subspace. These are given by,

$|\mathbf{V}T\mathbf{S}b\mathbf{V}||\mathbf{V}T\mathbf{S}w\mathbf{V}|$.

The solution to this problem is given by the generalized eigenvalue decomposition

$\mathbf{S}b\mathbf{V}=\mathbf{S}w\mathbf{V}\mathbf{\Lambda}$,

where $\mathbf{V}$ is (as above) the matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of corresponding eigenvalues.

The eigenvectors of $\mathbf{V}$ associated to non-zero eigenvalues are the Fisherfaces. There is a maximum of $C-1$ Fisherfaces. This can be readily seen from the definition of $\mathbf{S}b$. Note that in our definition, $\mathbf{S}b$ is a combination of C feature vectors. Any C vectors define a subspace of $C-1$ or less dimensions. The equality holds when these vectors are linearly independent from one another.

* Technicalities

To obtain the Fisherfaces, we need to compute the inverse of $\mathbf{S}w$, i.e., $\mathbf{S}-1w$. If the sample feature vectors are defined in a p-dimensional space and p is larger than the total number of samples n, then $\mathbf{S}w$ is singular. There are two typically used solutions to this problem. In the first solution, we project the sample vectors (or, equivalently, the between- and within-class scatter matrices) onto the PCA space of r dimensions, with $r\leq rank(\mathbf{S}w)$ and compute the Fisherfaces in this

PCA space. The second solution is to add a regularising term to $\mathbf{S}w$. That is, $\mathbf{S}w+\epsilon\mathbf{I}$, where $\mathbf{I}$ is the identity matrix and ϵ is a small constant.

One can also substitute the between- and within-class scatter matrices for other measurements that do a similar job. First, note that these two matrices are symmetric and positive semi-definite. Hence, each defines a metric. This means we can substitute these matrices with others as far as they define metrics whose goals are to minimize within-class variance and maximize between-class distances. For example, we can substitute the within-class scatter matrix for the sample covariance matrix.

The Fisherfaces obtained with the approach described thus far are based on the linear assumption mentioned above. This assumption holds when the classes are homoscedastic Normals. In general, this assumption is violated. To resolve this problem, one can add another metric into the equation. The goal of this new metric is to map the original heteroscedastic (meaning with different covariances) problem into a homoscedastic one. This mapping function can be converted into a kernel mapping thanks to the Representer's Theorem. And, the metric is given by the Gram (or Kernel) matrix. For this reason, this alternative method is called Kernel LDA (or, KLDA for short). Several authors have also defined heteroscedastic measures of within- and between-class differences. Yet, another alternative to lessen the Normal assumption is to represent the samples in each class as a mixture of Normal distributions. In this approach, the trick is how to determine the number of mixtures per class. A popular solution is given by the algorithm Subclass Discriminant Analysis (SDA) and its kernel extension KSDA. Kernel methods are generally preferred when the number of training samples is sufficiently large to facilitate the learning of the nonlinear mapping.

* Fisherface extensions

Recently, Two-dimensional LDA (2DLDA), a tensor extension of LDA, is proposed. Different from the LDA which requires the input patterns to be converted to one-dimensional vectors, the 2DLDA directly extracts the proper features from image matrices based on Fisher’s Linear Discriminant Analysis.

4.4 Local Binary Pattern Histogram

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang. The combination of LBP with histogram oriented gradients was introduced in 2009 that increased its performance in certain datasets. 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 4.8.

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 as in figure 4.9, the result is the same as before.

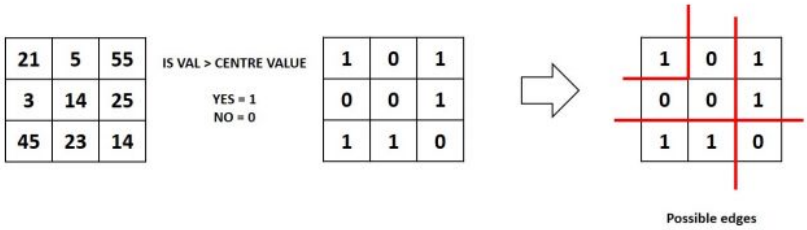


Figure 4.8: Local binary pattern histogram generating 8-bit number

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.

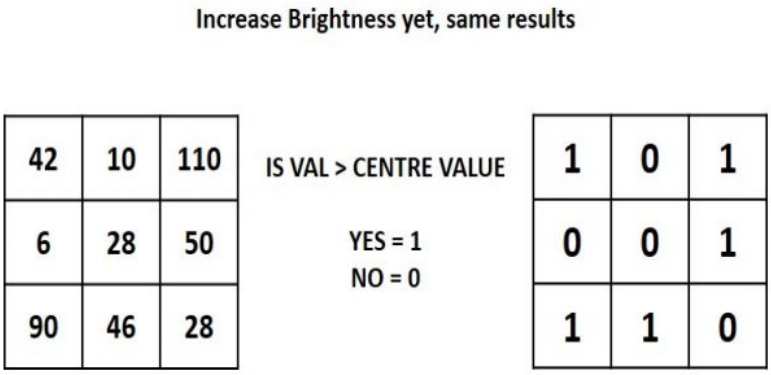


Figure 4.9: The results are same even if bright

4.5 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 4.10. Face and eye classifier objects are created using classifier class in OpenCV through the **cv2.**

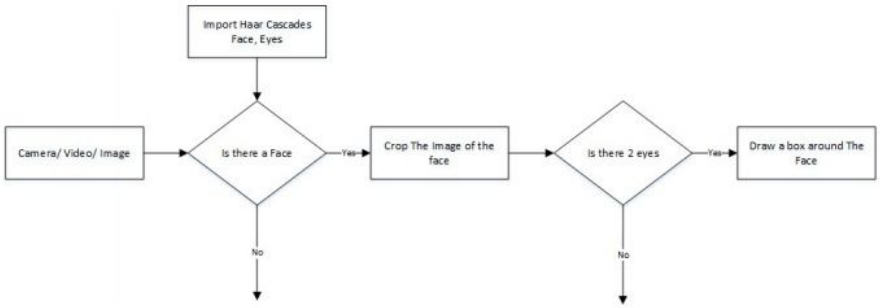


Figure 4.10: The Flow chart of the face detection application

CascadeClassifter() and loading the respective XML files. A camera object is created using the **cv2. Video Capture()** to capture images. By using the **Cascade Classifier. detect Multi Scale()** 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.

4.6 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:

1. Collecting images IDs
2. Extracting unique features, classifying them and storing in XML files
3. Matching features of an input image to the features in the saved XML files.

4.7 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 require 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. The Flow chart for the application is shown in figure 9.

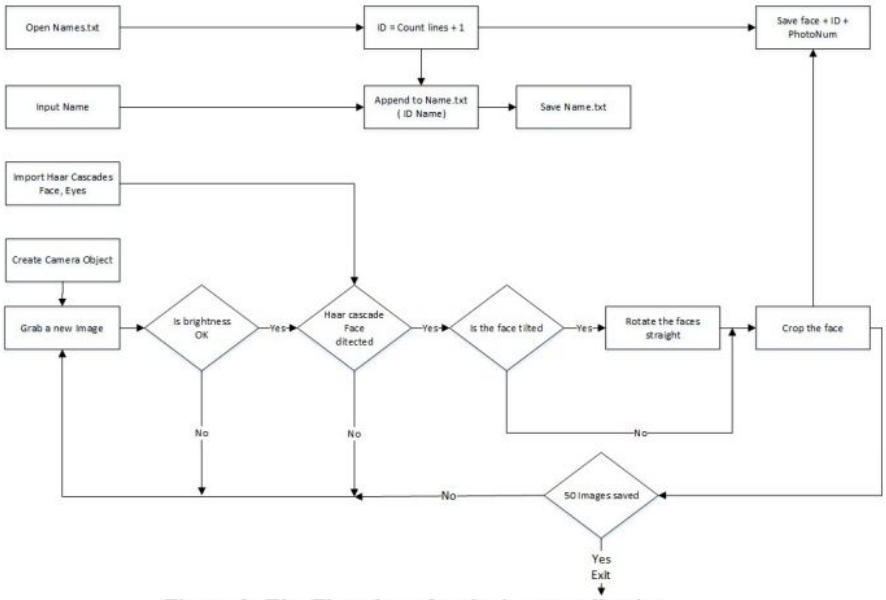


Figure 9: The Flowchart for the image collection

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 applications check for the brightness levels and will capture only if the face is well illuminated. Furthermore, after the face is detected, the positions of the eyes are analyses. 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.

4.8 Training the Classifiers

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

*** CV2.Face.CreateEigen Face Recognizer()**

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.CreateFisher face Recognizer()**

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.create LBPH Face Recognizer()**

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 **Face Recognizer. Trains (Numpy Image, 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 **Face**

Recognizer. save(File Name). In this project, all three are trained and saved through one application for convenience. The flowchart for the trainer is shown in figure 4.11.

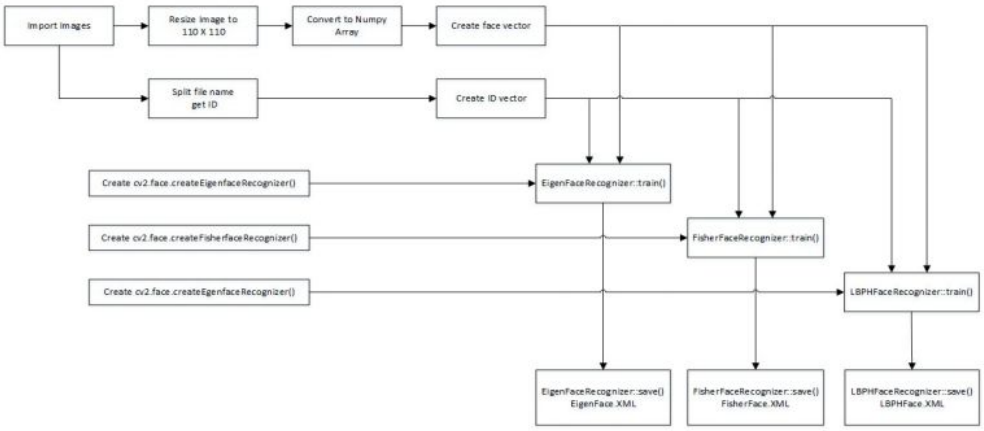


Figure 4.11: Flowchart of the training application

* The Face Recognition

Face recognizer object is created using the desired parameters. Face detector is used to detect faces in the image, cropped and transferred to be recognized. This is done using the same technique used for the image capture application. For each face detected, a prediction is made using **Face Recognizer. 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. The flow chart for the application is shown in figure 4.12.

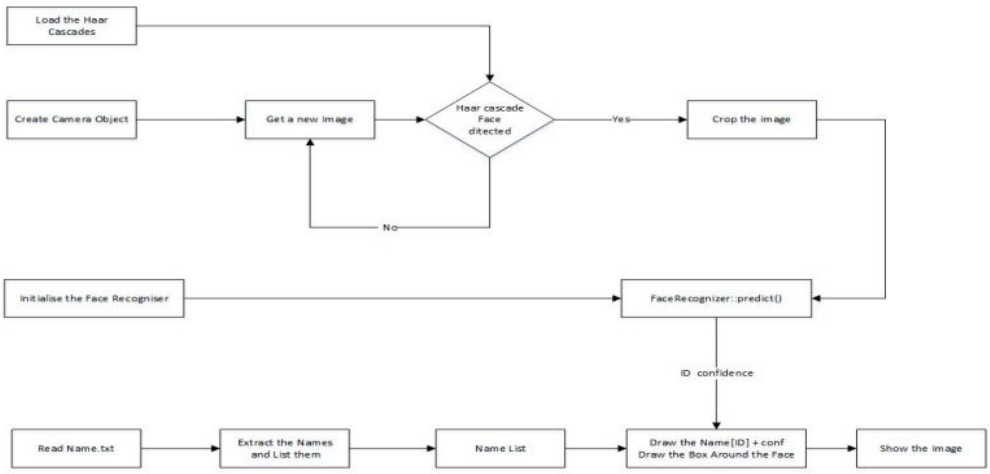


Figure 4.12: Flowchart of the face recognition application

Results

The collected images are shown below. Each face has 50 images. Three applications were written to iterate 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. The applications are :

Test Data ColltorEigenFace.py

Test Data Collector FisherFace.py

Test Data Collector LBPH.py.

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

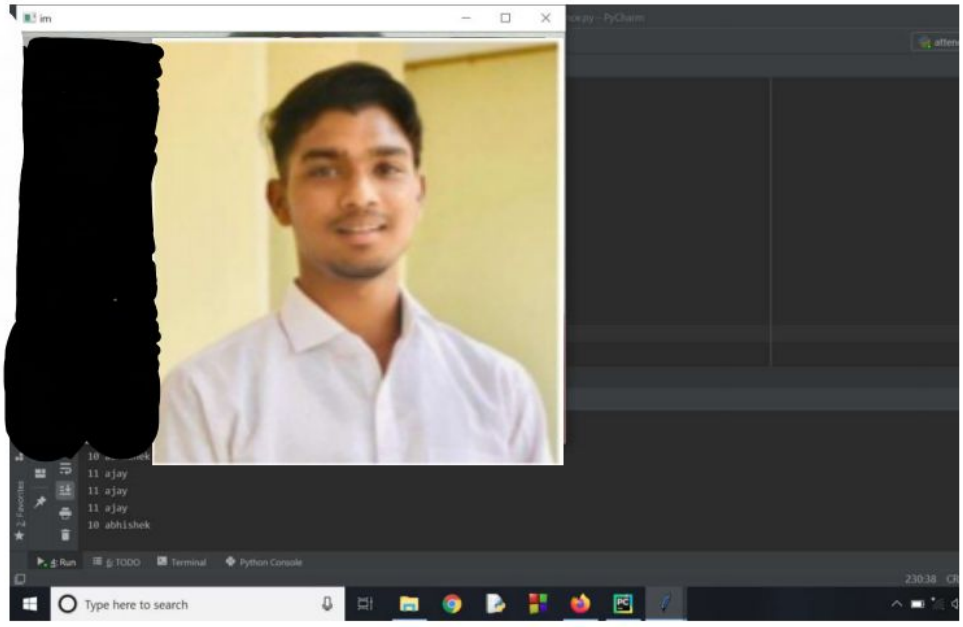


Figure 5.1: Initial Testing

The first test image is shown in figure 5.2 and the plots are analyzed below.

The resulting ID change is plotted below in figure 5.3. Note when components were 1, it identified

Figure 5.2: Image used for this test

the face as ID-17 and the rest are between ID-20 and ID-21, which is the same person.

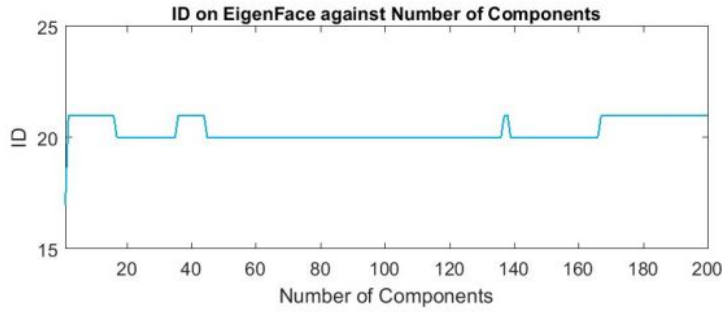


Figure 5.3: The ID from the face recognizer changes between two classes of the same person

The change of confidence is plotted in figure 5.4, increasing with components. From this plot it appears the best is when components are below 20.

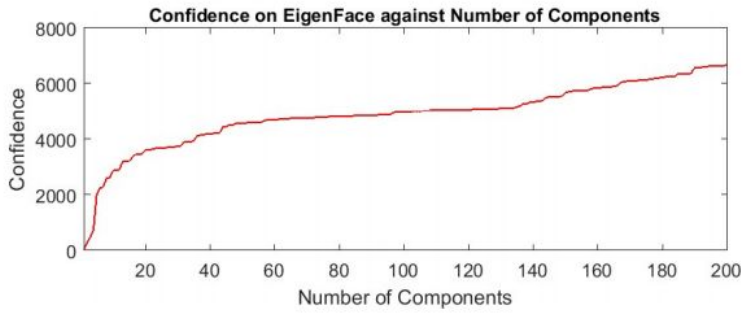


Figure 5.4: The Confidence increasing with No. of components

The ID results from Fisherface are more stable than Eigenface and is on ID-21 as seen in figure 5.5.

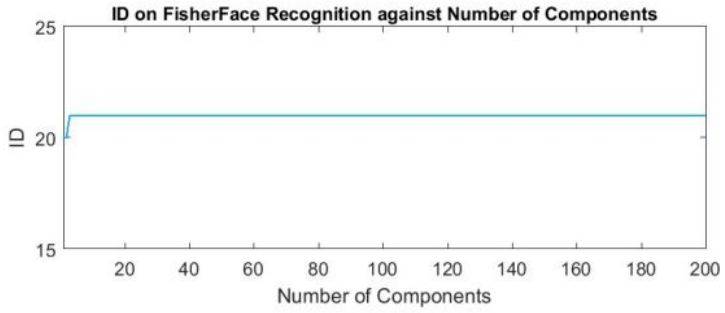


Figure 5.5: Fisherface ID is stable

Fisherface confidence increase in figure 5.6 until the number of components is 10 and will be used as the ideal value. LBPH has more than one parameter to change. All are incremented to the maximum limit and the results are shown below.

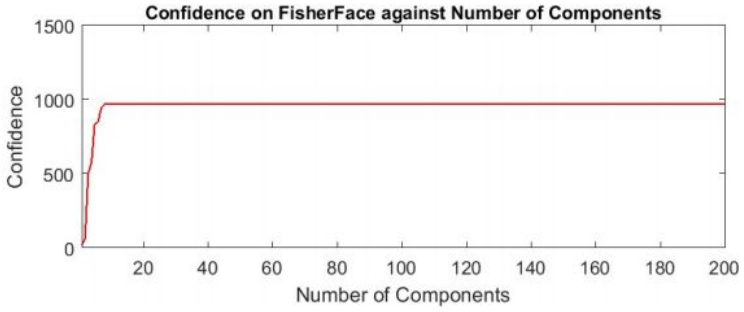


Figure 5.6: Stable confidence after 10 components

The first is the radius from the centre pixel and since the image size is 110 X 110, maximum radius is 54. The ID is steady all the way to 50 as can be seen in figure 5.7.

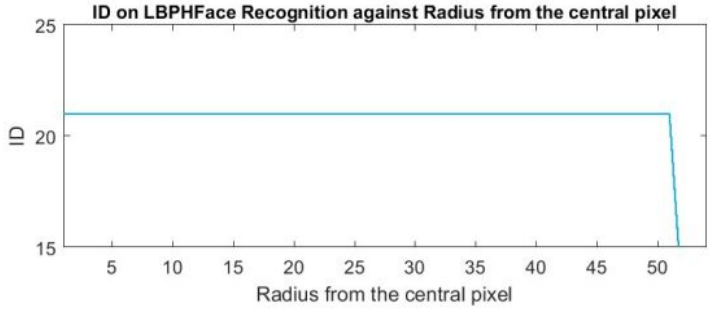


Figure 5.7: The ID returned from LBPH

Confidence level is graphed against the radius in figure 5.8. The confidence is fluctuating after 40.The lowest confidence level is at 2.

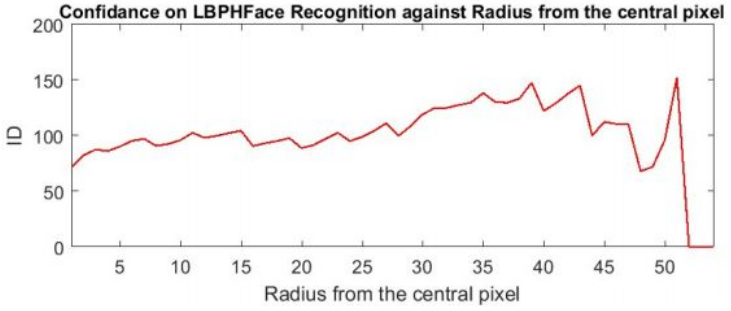


Figure 5.8: The confidence returned from LBPH

The number of neighbours are changed from 1 - 13. Further increase caused the computer to stall. The returned ID is plotted below in figure 5.9. ID steady until 9 neighbours and changed to ID-20.

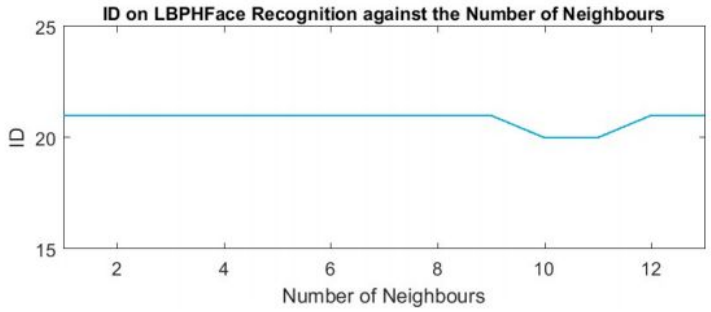


Figure 5.9: The ID returned from LBPH changing neighbours

The confidence continuously increased as can be seen in figure 5.10 and 1 neighbour will be included to the next test.

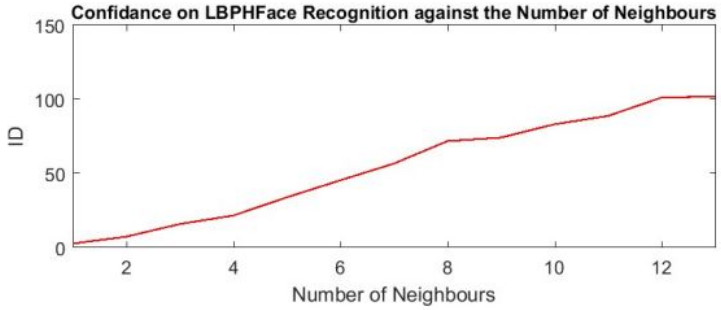


Figure 5.10: The Confidence returned from LBPH changing neighbour

The number of cells in X and Y directions are changed simultaneously. ID return is plotted below in figure 5.11. The ID changed from ID-20 to ID -21 and stay steady.

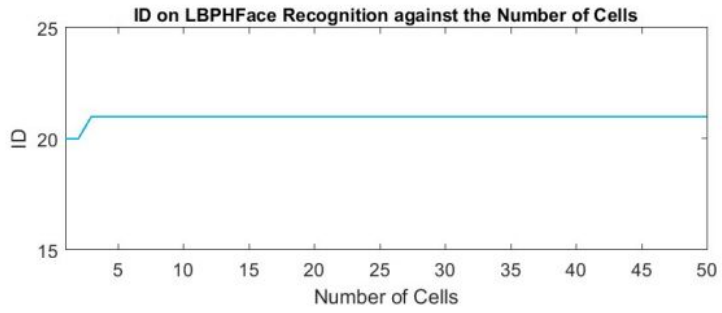


Figure 5.11: The ID returned from LBPH changing the number of cells

The returned confidence is plotted in figure 5.12. The confidence was low when cells were less than 8 per side and increased rapidly after 10 ending up more than 1700 at 50.

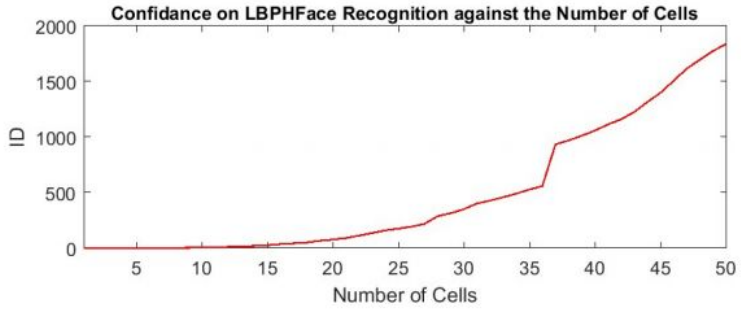


Figure 5.12: The confidence returned from LBPH changing the number of cells

5.1. Recognizing a different subject

For a secondary test, authors' colleague provided a photo which was 7 years old. Although the photo has two people, plots only contain data on Mr. Westlake for clarity. The idea was to observe if the program can identify a younger face from the data-set of an older face.

The same program is used as above and plots for Eigenface, Fisherface and three separate pairs for the LBPH are below in figures 5.13, 5.14.

Eigenface and Fisherface IDs and confidence was as expected. Note that Mr. Westlakes' ID is 17. The LBPH ID fluctuated when the pixels were above 38. Same can be noticed when cells are increased and low cells are unstable. The calibration details are detailed after the plots.

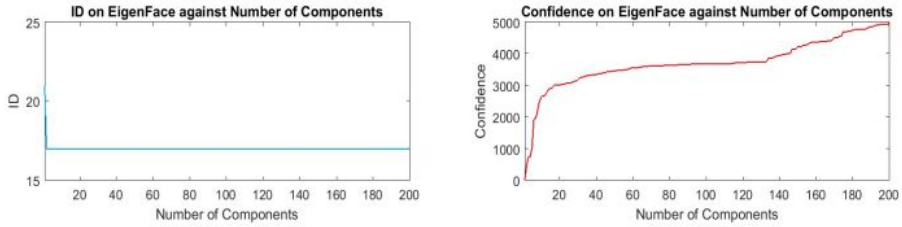


Figure 5.13: ID and Confidence for Eigenface. Note the ID is at 17 which is correct

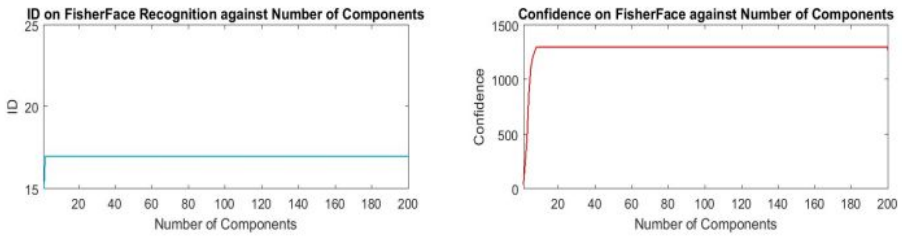


Figure 5.14: ID and Confidence for Fisherface. Note the ID is at 17 which is correct

The following plots show the data collected for LBPH parameters.

Figure 5.14: ID and Confidence for LBPA neighbours. Note the ID fluctuating at 37 pixels

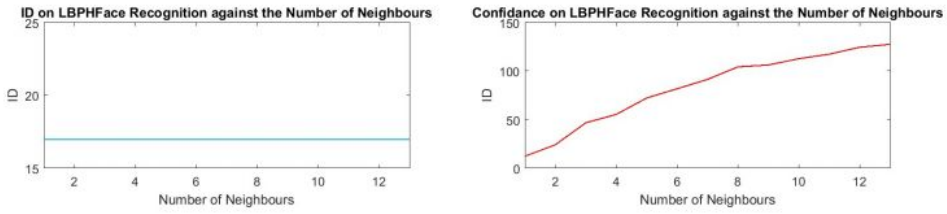


Figure 5.15: ID and Confidence for LBPA neighbours. Note the confidence increasing

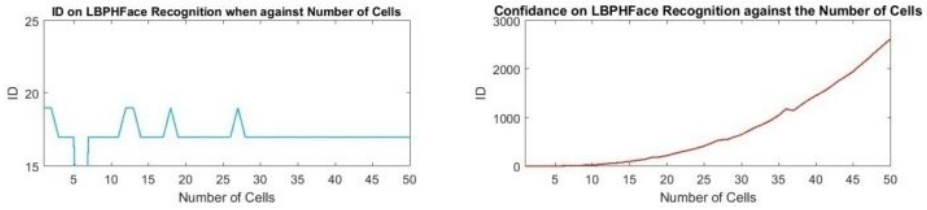


Figure 5.16: ID and Confidence for LBPA cells. Note the ID-17 is steady.

5.2. Calibrating the Trainer and Testing on Video

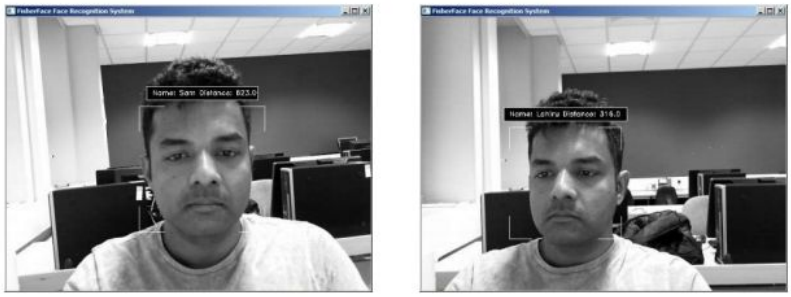
The Trainer (Trainer-All.py) application is used to train and recognizer (Recogniser-All.py) is used for testing the results. The image pairs shown below are with and without calibration of the trainer. All images are taken through video via Microsoft Life cam camera. The values used are as followed: **Eigenface = number of components 15, Threshold 4000**

Fisherface = number of components 5, Threshold 400

LBPH = Rac **ıres**

Before Calibration After Calibration

Figure 5.17: Before and after Eigenface Calibration threshold: 4000



Before Calibration After Calibration

Figure 5.18: Before and after Fisherface Calibration threshold:400

Before Calibration After Calibration

Figure 5.19: Before and after LBPH face Calibration threshold: 15

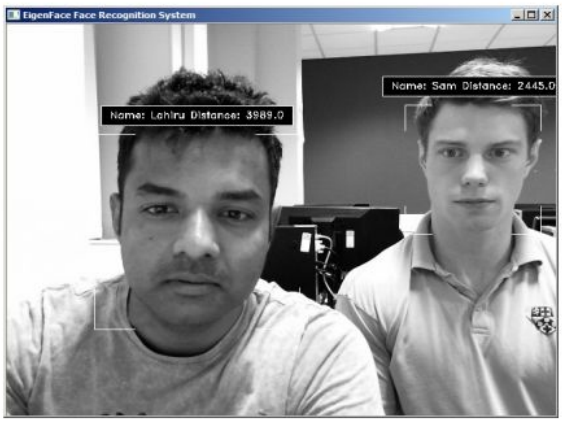


Figure 5.20: After calibration of Eigenface Note both faces are detected successfully

The following image shows the system stating that the face is not recognized. Mr. Wallkoetters' face was not included at the time.

Figure 5.21: A face that is not in the data-set not recognized.

5.3. Discussion

An early attempt on Eigenface and Fisherface was disappointing since the LBPH alone recognized 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. These 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.

6.1. 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.

6.2. Future Scope

The accuracy of text and face recognition is based upon pose, illumination, emotions, facial components and image quality. Certain features need to be incorporated in the system to process real time images at faster rate with high precision. Another aspect of the research includes, developing a model which if trained on given criminal record dataset can predict the face sketch of a criminal based upon features fed as input by a witness.