

1. INTRODUCTION

1.1 BIOMETRICS

Biometrics is used in authentication of a person by verifying or identifying that a user requesting for access is who he, she, or it claims to be, and vice versa. It uses human features associated with a person itself like structure of palm, finger, face details etc. By comparing the existing data with the incoming data we can verify the identity of a particular person.

There are many types of biometric system like iris recognition, ear recognition, fingerprint recognition, face detection and recognition etc. These are used for human identification in surveillance system, criminal identification, at airports and in many digital devices. Advantages of using these features are that they cannot be destroyed, forgotten or lost. The features which are unique for each human being are used for identification.

1.2 FACE RECOGNITION

Face recognition means to identify the person from still image or from video based on facial features from face image dataset. In today's unsecure world authentication is required everywhere. Biometrics is an automatic method of recognizing a person or checking the identity of a person based on its unique features. Many biometric techniques like iris recognition, ear recognition, finger print recognition, face recognition etc have left their impression in the area where authentication or security is the prime concern. Each of the techniques has some advantages and disadvantages. Biometric uses the features which are always with the person and it will not change with time. Idea of swipe card has become obsolete as there will be so many risks in using that. Cards might be lost or forgotten or stolen, at that time it creates a problem. The necessity for person recognition in the fields of security systems made face recognition one of the main fields among other biometric technologies. The importance of face recognition increases from the fact that a face recognition system does not require the cooperation of the individual while the other systems need such cooperation. Face is very rich which contains many

unique features for each person like eyes, eyebrow, lips, nose and many more other features. Of course, there are many dimensions of problem in using face for the system. Facial features may also change with age, illumination, pose of the face, dimension etc. Many algorithms are there for face recognition.

As shown in figure, 1 most of the face recognition techniques are divided into the four categories: holistic approach, feature based approach, model based approach and hybrid method.

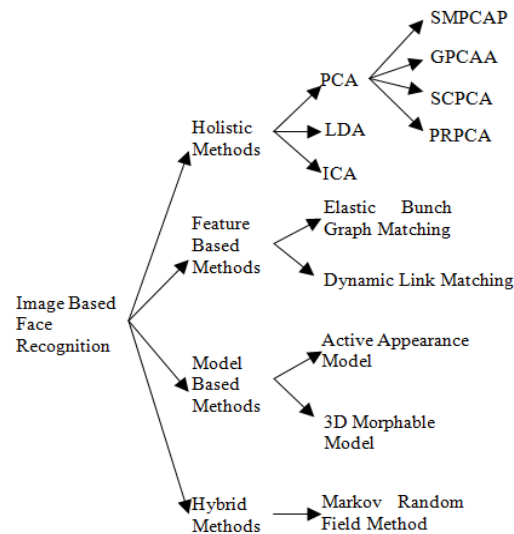


Fig.1.1 Taxonomy of face recognition methods

Feature based methods only emphasis on facial features like eyes, eyebrow, nose , lips etc and their geometric relations. Whereas, holistic methods use entire face and represent face as a code point in higher dimensional image space [9]. As holistic methods are applying entire face region for the training, it is quite complicated to train image against it compared to straightforward facial features. PCA is a holistic method. PCA is a

variable reduction procedure and useful when obtained data have some redundancy. This will reduce number of variables then the original variables. These variables are called principle component.

1.3 AIM AND OBJECTIVE

Objective of this project is to provide a program which identifies a correct person related to shown input image.

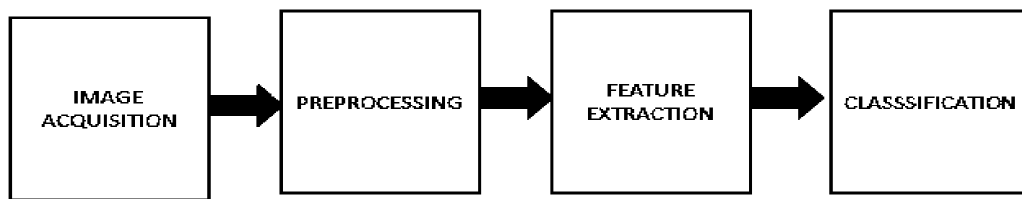


Fig 1.2 Face recognition system

Face recognition system recognizes face from the image. This application uses the face image of the user. This image is processed and compared with dataset images, find the matching person and recognized person name is displayed. The main phases of the recognition are Image acquisition, preprocessing, segmentation, feature extraction and classification. When these operations are performed on real time video it takes more time. So cluster is used for faster response. OpenCV(*Open Source Computer Vision Library*) library is used for face recognition. MPI and MPICH library is used for making cluster.

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision. It is developed by Intel. The library is cross-platform so it is used on many different platforms. It focuses mainly on real-time image processing and computer vision related applications. If the library finds Intel's Integrated Performance Primitives on the system, In this Project Open CV is used for many image related transformations.

Cluster will provide higher performance to system. If program run on many processors simultaneously then it provide faster execution using resources of all cluster nodes. MPI provide a good way to execute a program parallelly.

There are many methods for face recognition, But holistic methods encode entire image and represent image as a code point in higher dimensional image space. PCA and FDA are the holistic methods which projects $N \times N$ image into some less number of key features. In this project PCA (Principal component analysis) is used for Feature extraction.

1.4 Project feasibility

A feasibility study aims to answer a number of questions like:

- Does the system contribute to the overall objective our project?
- Can the system be implemented using the currently available technology and within given cost and schedule conditions?
- Can the system be integrated with system which is already implemented?

1.4.1 Operational Feasibility

Operational feasibility measures how well the solution will work in the organization and how will the end-user feel about our system. Proposed system is helpful for the analyzing and decision making related to material requirement, planning and scheduling. The following conclusion could be derived by studying the operational feasibility of the project:

- Developed system will provide satisfactory throughput to the users.
- It will provide reliable services.
- System will function properly under the same if installed under an environment of limited resources.
- The proposed system makes best efforts to combine the possible features required for proper interaction with the client & hence satisfy the requirements of the users.

So the proposed system is operationally feasible.

1.4.2 Technical Feasibility:

For checking technical feasibility of system following questions should be answered.

- The software used in application or tools for building or running the application are easily available or not?
- Softwares used in system are compatible to other software or not?
- Are developers of this system aware of these chosen technologies?
- What are the alternate technologies of the chosen technology in current system?

The answers to the above mentioned questions are given as follows:

The tools used in system are easily available from the internet and anyone can download or buy easily. Company also provides their trial & licensed versions.

The proposed project has Linux(Ubuntu) as its developing operating system, & our chosen technologies are well compatible with this operating system.

As developers, we are well aware of the technologies chosen to implement this project.

Hence, the proposed project is technically feasible.

1.4.3 Economical Feasibility:

Economic feasibility addresses the following issues:

- Is the organization having suitable budget to develop the proposed system?
- How much profit can be earned by using this system by an organization?
- Would it be cost-effective to develop this system or it is worthwhile to remain with current system.

We would to give the answer the above question as below:

- As development tools are downloaded freely from internet or you can purchase those softwares, So there is not any burden of buying new system.

- The profit will be good enough according to our team has seen as far as market is concerned.

2. LITERATURE REVIEW

2.1 Facial Recognition using Eigenfaces by PCA [1]

This paper refers different face recognition methods and focuses on Principal Components Analysis. Scilab is the free software for the analysis and the implementation. This system detects the faces in the image captured by the webcam. And then these images are checked with training images based on descriptive features.

2.2 Face Recognition using eigenface [2]

This paper presents an approach of detection and identification of human faces. It is near real time face recognition system which detects a person's head and then recognizes the person by comparing characteristics of the face.

2.3 3D FACE RECOGNITION TECHNIQUES - A REVIEW [3]

This paper gives analysis of new face recognition technology, 3D face recognition. It gives discussion of the various approaches which are accepted. 2D face recognition technology has faced many challenges. These challenges are solved through various approaches. Each process in the face recognition has sub process and the sub process is categorized into registration, representation, extraction of discriminative features.

2.4 A high-performance, portable implementation of the MPI message passing interface standard [4]

MPI forum, a broadly based group of parallel computer vendors, library writers has defined Message passing interface. It is a specification for a standard library of message passing. There are multiple implementations developed for MPI. This paper describes MPICH which is unique in its design goal of combining portability with high performance among the other existing implementations.

2.5 Learning from the Success of MPI [5]

The Message Passing Interface has become extremely successful as a portable way to program high-performance parallel computers. Instead of the different views that message passing is difficult, it has been successful. The paper argues that MPI has succeeded because it addresses all the important issues for a parallel programming model.

3. DESIGN APPROACH AND IMPLEMENTATION

3.1 Cluster

3.1.1 Introduction

Very often applications need more computing power than a sequential computer can provide. One way of overcoming this limitation is to improve the operating speed of processors and other components so that they can offer the power required by computationally intensive applications. Even though this is currently possible to a certain extent, future improvements are constrained by the speed of light, thermodynamic laws, and the high financial costs for processor fabrication. A viable and cost-effective alternative solution is to connect multiple processors together and coordinate their computational efforts. The resulting systems are popularly known as parallel computers, and they allow the sharing of a computational task among multiple processors.

As Pfister points out, there are three ways to improve performance:

- Work harder,

- Work smarter, and
- Get help.

In terms of computing technologies, the analogy to this mantra is that working harder is like using faster hardware (high performance processors or peripheral devices). Working smarter concerns doing things more efficiently and this revolves around the algorithms and techniques used to solve computational tasks. Finally, getting help refers to using multiple computers to solve a particular task.

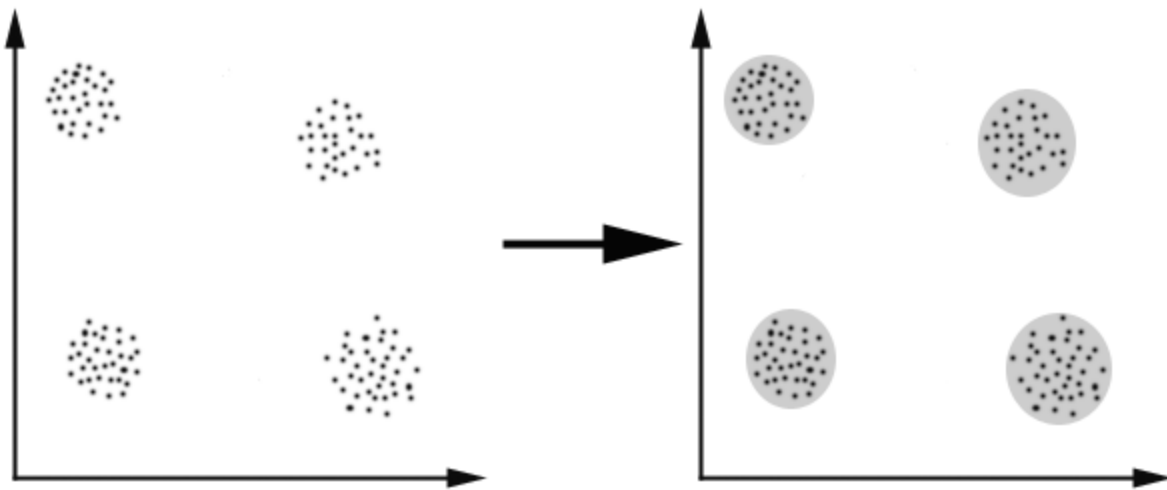


Figure 3.1

Goals of clustering

The main goal of clustering is to determine the inherent grouping in a set of unlabeled data. But what parameters decide what constitutes a good clustering? It can be shown that there is no absolute “best” criteria which would be independent of the final aim of the clustering. Therefore, it is the user which must supply this criteria, in such a way that the result of the clustering will fulfil their needs.

For instance, we would be interested in finding representatives for homogeneous groups (groups with same features) (data reduction), in finding “natural clusters” and describe their unknown properties (“natural” data types), in finding useful and suitable groupings (“useful” data classes) or in finding unusual data objects (outlier detection).

Possible Applications

Clustering algorithms can be applied in many fields, for instance:

- Clustering is used in marketing for finding groups of consumers with similar behaviour given a large database of consumer data containing their properties, past choices and past buying records.
- It is used in biology in classification of plants and animals from their features.
- It is libraries for book ordering;
- Insurance Companies also use clustering in identifying groups of people who have highest claims for motor insurance policy.
- City-planning uses clustering to identify groups of houses according to their house type, value and geographical location.
- Clustering observed earthquake centres to identify dangerous zones.
- WWW (World Wide Web) use clustering for document classification and to discover groups of similar access patterns.

Requirements

The main requirements that a clustering algorithm should satisfy are:

- Scalability, usability and interpretability are the main requirements.
- It should deal with different types of attributes.
- Algorithm should discover clusters with random shape.
- It should contain minimum requirements for domain knowledge to determine input parameters.

- It has ability to deal with noise.
- It should be insensitive to order of input records.
- high dimensionality is also one requirement.
- Interpretability and usability.
- Scalability.

Problems

There are a number of problems with clustering. Some problems are shown here:

- Currently used clustering techniques do not address all the requirements sufficiently and simultaneously.
- Because of time complexity, dealing with large number of dimensions and large number of data items can be problematic.
- For distance-based clustering the effectiveness of the method depends on the definition of “distance” .
- If a noticeable distance measure doesn’t exist we must “define” it, which is not always easy, especially in multi-dimensional spaces.
- The result of the clustering algorithm (that can be arbitrary itself) can be explained in many different ways.

3.1.2 Cluster setup

For cluster setup we have used MPICH2.

Steps for installation of MPI in UNIX

Following steps are used for installation of MPI.

- 1) Create a directory MPI or any other name in the home directory.
`$ cd $HOME`
`$ mkdir MPI`

- 2) Then unpack the tar file in MPI folder.

```
$ tar xzf mpich2-1.0.5p3.tar.gz
```

Now the MPI folder has a sub-directory named mpich2-1.0.5p3.

- 3) Now choose an installation directory.

```
$ mkdir mpich2-install
```

- 4) Then choose a build directory.

```
$ mkdir mpich2-1.0.5
```

After these steps the MPI directory will have three subdirectories, mpich2-1.0.5p3, mpich2-1.0.5 and mpich2-install.

- 5) To configure MPICH2, specify the installation directory and run the configure script in the source directory.

```
$ cd $HOME
```

```
$ cd MPI/mpich2-1.0.5
```

```
$/home/mpi1/MPI/mpich2-1.0.5p3/configure -prefix=/home/mpi1/MPI/mpich2-install
```

- 6) Now build MPICH2.

```
$ make
```

- 7) Then install the MPICH2 commands.

```
$ make install
```

- 8) Then add the bin directory to your path.

```
$ export PATH=/home/mpi1/MPI/mpich2-install/bin:$PATH
```

(To add this path permanently in the system add these two lines in the `.bash_profile` file present in the home directory

```
$ cd $HOME
```

```
$ vi .bash_profile
```

Then append the above command of step 8.)

To check that everything is in order, use following commands.

```
$ which mpd
```

```
$ which mpiexec
```

```
$ which mpicc
```

```
$ which mpirun
```

Now we have to perform the following steps to make a cluster using MPI.

- 1) We must add the IP address of home machine and other machine in `/etc/hosts` file on each PC.

```
$ vi /etc/hosts
```

In this file append all machine names and IP address.

- 2) Now create a new file **mpd.hosts** which consist the list of machine names. You have to add single machine name on each line.

```
$ cd $HOME  
$ vi mpd.hosts
```

Now we have to perform following steps before starting the programs.

- 1) Only once we have to execute these commands on each single system.

```
$ touch .mpd.conf  
$ chmod 600 .mpd.conf  
$ vi .mpd.conf
```

And

Add
MPD_SECRETWORD=<_____>

For secret word we have to use any word without spaces. It should be same for each user in all the systems.

This has to be done so that other user cannot communicate with our daemon. And without this our MPI interface would not start.

- 2) To synchronize the activities between different systems, each user will be running a daemon named 'mpd' on each single system. Among them one daemon will be master daemon. Master daemon is to be initiated at first. Then other mpd daemons are started later as will be shown in step 3.

```
$ mpd &  
$ mpdtrace -l
```

It gives hostname_portno (IP address) as the output. This port number is required to start the mpd on other systems.

- 3) Then login into other systems, and start mpd daemons as such

```
$ mpd -h <hostname of master daemon> -p <portno> &
```

- 4) Then run the following command.

```
mpdtrace -l
```

This command is used to check how many systems are included in our loop.

- 5) At the end, You have to check your personal multi-processor environment by

```
mpiexec -n 4 /bin/hostname
```

Now we can write MPI programs and compile any program using command given below.

```
mpicc -o <object file> <source code>.c
```

3.2 PCA (Principal Component Analysis) based approach

3.2.1 Why use PCA?

There were many issues to consider when choosing a face recognition method. The main issues were:

- Accuracy
- Time limitations
- Process speed
- Availability

Considering them in mind PCA method is selected for this project because:

- It is simplest and easiest method to implement as a beginners.
- It has very fast computation time.
- It is accurate enough. This method is definitely not the most accurate of face recognition algorithms but considering the requirements of this project it was taken to be accurate enough.
- PCA is supported within the OpenCV (Computer Vision) library – this was key because it made integration with the face detection program very easy.

3.2.2 Theory of PCA

Principal Component Analysis is a process that extracts the most relevant information from face and then tries to build a computational model that best describes it. The basic theory of PCA can be explained by the following steps:

1. Eigenvectors or eigenfaces of the covariance matrix are found. This is done by training a set of face images.
2. These eigenvectors become the eigenspace (a multi-dimensional subspace comprised of the Eigenvectors) in which every face is projected on.

3. Recognition is performed by comparing the location of a face in the eigenspace with the location of known users. In other words calculating the Euclidean distance.

This is only the basic theory of PCA. To understand PCA it needs to be explained mathematically which is done below.

3.2.3 Mathematics of PCA

The mathematics of PCA describes how an image is broken down, how the eigenvectors are calculated and eventually how images can be compared to each other by calculating the Euclidean distance.

First a 2D face image is broken down into a 1D vector by concatenating each row into one large vector. Say M equals the number of input images there are and N (= rows of image x columns of image) equals the size of the images. Then an image can be represented by the vector:

$$x_i = [p_1 \dots p_N]^T, i = 1, \dots, M \quad (1)$$

where the p_j 's represent an individual pixel value. Before analysis these image vectors have to be mean centred by subtracting the mean image from the image vector. Let m be the mean image and w_i be the mean centred image:

$$m = \frac{1}{M} \sum_{i=1}^M x_i \quad w_i = x_i - m \quad (2)$$

The next step is to find a set of eigenvectors that best describe the principal information in each of the mean centred images. The set of eigenvectors (e_i) with the highest eigenvalues (λ_i) are the ones needed. Where the eigenvalues are equal:

$$\lambda_i = \frac{1}{M} \sum_{n=1}^M (e_i^T w_n)^2 \quad (3)$$

It has been proven that these eigenvectors and eigenvalues are obtained by getting the eigenvectors and eigenvalues of the covariance matrix. The equation of covariance matrix is given the below:

$$C = WW^T \quad (4)$$

W is the matrix composed of the mean centred image vectors (w_i) and W^T is its transpose. The eigenvectors of c cannot be solved directly because of its size. E.g. a 100 x 100 image would create a mean centred image 10000 coefficients long, so the covariance matrix would be 10000 x 10000 in size. There is another way of calculating them using the $M \times M$ matrix $W^T W$. Let d_i and μ_i be the eigenvectors and eigenvalues of $W^T W$, respectively:

$$W^T W d_i = \mu_i d_i \quad (5)$$

Then multiply both sides by W :

$$WW^T (W d_i) = \mu_i (W d_i) \quad (6)$$

Now the first $M - 1$ eigenvectors (e_i) of the covariance matrix (WW^T) are represented by the normalised eigenvectors $W d_i$. Eigenvalues (λ_i) of the covariance matrix (WW^T) are given by the eigenvalues μ_i . The covariance matrix can only be of rank $M - 1$ because

there is a finite amount of input image vectors and the ‘-1’ comes from the initial subtraction of the mean image.

“The eigenvectors corresponding to nonzero eigenvalues of the covariance matrix produce an orthonormal basis for the subspace within which most image data can be represented with a small amount of error.” So these eigenvectors contain the most relevant information from the set of input images and are used to construct an eigenspace within which other images can be analysed. They are arranged from high to low according to their matching eigenvalues. The reason for this is because the eigenvector with the highest eigenvalue exhibits the greatest variance in the image.

A face image can then be projected onto this eigenspace and the location of the image in this subspace is determined by the formula:

$$\Omega = [v_1 v_2 \dots v_{M'}]^T \quad (7)$$

Where $v_i = e_i^T \times w_i$ and M' is the number of dimensions (smaller than M). Ω describes the input of each eigenvector in the representation of the input face image. Face recognition can be implemented by projecting an input face image (Ω) onto the eigenspace and then finding a face in the database (Ω_k) that is closest to it. The distance between the faces in the subspace is called the Euclidean distance (ϵ_k).

$$\epsilon_k = \|\Omega - \Omega_k\| \quad (8)$$

A threshold, θ_e , is predefined so if the Euclidean distance between an input face and a user’s face is smaller than this threshold, it is recognised as a face belonging to that user.

We have performed PCA algorithm to recognize the face.

3.2.4 Dataset description

3.2.4.1 Cambridge face dataset

We have implemented PCA for Cambridge face dataset. It contains 10 images of 40 people, total 400 images. The files are in PGM format. The size of each image is 112 X 92 pixels, with 256 grey levels per pixel. Figure 1 shows the one image of each person.



Fig 3.2. Sample images of ORL dataset

3.2.4.2 Faces 94

This dataset contains images of 20 images of 153 individual-20 female, 113 male, and 20 male staff. The size of each image is 180 X 200 pixels. All the images are colored and the background is plain green. Figure 2 shows the sample images for one person.

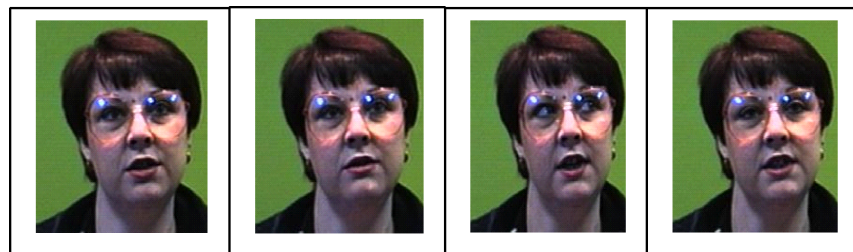


Fig 3.3. Sample images of one female

3.2.5 Steps for Face Recognition using PCA:

Let a face image $I(x,y)$ be a two-dimensional $M \times N$ array of 8-bit intensity values. An image may also be considered as a vector of dimension $N \times M$, so that a typical image of size 112×92 becomes a vector of dimension 10,304, or equivalently a point in 10,304-dimensional space. An ensemble of images, then, maps to a collection of points in this huge space.

1. Take K input images.

$$Y_1, Y_2, Y_3, \dots, Y_k$$



Figure3.2. Images of ORL Database

2. Reduce size of image by $\frac{1}{4}$ to make faster computation. So our new dimensions of images are $M/4 \times N/4$
3. Reshape image and convert the image into column major matrix. If image size is $M \times N$ then the reshaped image will be $(M \times N) \times 1$. So dimensions of reshaped image are 644×1 .
4. Now find mean of all images by adding all images and dividing them by total no of images.

$$\mu = Y_1 + Y_2 + Y_3 + Y_4 + \dots + Y_n / K.$$

Where K = Total number of Images

$$\mu = \sum_{i=1}^k Y_i$$



Figure 3.3. Mean Image of all Images

5. Now each of the training images must be centered. Subtracting the mean image from each of the training images centers the training images. The mean image is a column vector such that each entry is the mean of all corresponding pixels of the training images. We'll subtract mean μ from all the images.

$$Y_i = Y_i - \mu;$$

6. Now Collect all column major matrices into one matrix to create data matrix called MS whose dimension will be

$$MS(M \times N, K)$$

Where K = Total number of training images (variable).

So, dimensions of MS are $644 \times K$.

7. We find Covariance which is calculated by multiplying Mean subtracted image and its transpose.

$$\epsilon = MS \cdot MS^T$$

where the dimensions of matrix MS are $644 \times K$. Hence, the dimensions of covariance matrix ϵ are 644×644 real symmetric matrix, and determining the 644^2 eigenvectors and eigenvalues is an intractable task for typical image sizes. We need a computationally feasible method to find these eigenvectors.

If the number of data points in the image space is less than the dimension of the space ($K < M*N$), there will be only $K-1$, rather than $M*N$, meaningful eigenvectors. The remaining eigenvectors will have associated eigenvalues of zero. We can solve for the $M*N$ dimensional eigenvectors in this case by first solving the eigenvectors of a $K \times K$ matrix and then, taking appropriate linear combinations of the face images.

With this analysis, the calculations are greatly reduced, from the order of the number of pixels in the images $M*N$ to the order of the number of images in the training set (K). In practice, the training set of face images will be relatively small ($K \ll M*N$), and the calculations become quite manageable. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness in characterizing the variation among the images.

So, we calculate ϵ by this equation

$$\epsilon = MS^T * MS$$

8. Now We will find Eigen values and Eigen Vectors of Covariance ϵ using inbuilt function in openCV. Then arrange Eigen values in descending order and then take first n number of Eigen vectors which are enough for face recognition.

9. Now we'll find U which is calculated by multiplying mean subtracted image and selected Eigen vectors' matrix and using U we'll calculate Eigen Faces \mathcal{F} by,

$$U = MS * \epsilon$$

$$\mathcal{F} = U^T * MS$$

We have completed training procedure now testing will be started.

10. Now for testing we'll be taking an image and subtract mean from that image and reshape it to get column major matrix.



Figure 3.4 Input Image for testing

11. Now we'll multiply this result with transpose of U.

$$\text{Test} = U^T * \text{TestImage}$$

12. We'll compare this Test matrix with number of Eigen faces and find Euclidean distance and find minimum distance which is our recognized image.



Figure 3.5 Recognized output Image

Example:

Here for example we have taken four training images and calculated mean is given below.

$Y_1 =$	225	$Y_2 =$	10	$Y_3 =$	196	$Y_4 =$	255	$\mu =$	171.5
	229		219		35		223		176.5
	48		24		234		224		135.5
	251		255		232		255		248.25

33	18	59	0	27.50
238	247	244	255	246
0	17	243	249	127.25
255	255	57	255	205.5
217	2	226	235	170

Now as step-5 we have subtracted mean image from each images so we'll get,

$Y_1' =$	53.5	$Y_2' =$	-161.5	$Y_3' =$	24.5	$Y_4' =$	83.
	52.5		42.5		-141.5		46.5
	-84.5		-108.5		101.5		91.5
	2.75		6.75		-16.25		6.75
	5.5		-9.5		31.5		-27.5
	-8		1		-2		9
	127.25		-110.2		115.75		121.75
			5				
	49.5		49.5		-148.5		49.5
	47		-168		56		65

Now as step-6 collect all matrix into one matrix and call it MS.

$$MS = \begin{bmatrix} 53.5 & -161.5 & 24.5 & 83.5 \\ 52.5 & 42.5 & -141.5 & 46.5 \\ -84.5 & -108.5 & 101.5 & 91.5 \\ 2.75 & 6.75 & -16.25 & 6.75 \\ 5.5 & -9.5 & 31.5 & -27.5 \\ -8 & 1 & -2 & 9 \\ 127.25 & -110.2 & 115.75 & 121.75 \\ 49.5 & 49.5 & -148.5 & 49.5 \\ 47 & -168 & 56 & 65 \end{bmatrix}$$

-84.5	-108.5	101.5	91.5
2.75	6.75	-16.25	6.75
5.5	-9.5	31.5	-27.5
-8	1	-2	9
127.25	-110.25	115.75	121.75
49.5	49.5	-148.5	49.5
47	-168	56	65

Calculate the covariance matrix:

$$\epsilon = MS' * MS.$$

36517	-3639	23129	-778	304	113	24000	-4851	36446
-3639	226747	-19155	3045	-5851	324	-22083	28017	-9574
23129	-19155	37587	-1997	1247	1188	45603	-20097	25888
-778	3045	-1996	363	-746.5	78	-2153	3217	-1476
304	-5851	1247	-747	1869	-364	645	-6237	1831
113	324	1188	78	-364	150	1772	396	-71
24000	-22083	45603	-2153	150	1772	56569	-22919	26937
-4851	28017	-20097	3218	1772	396	-22919	29403	-11088
36446	-9574	25888	-1476	396	-71	26937	-11088	37794

The ordered non-zero eigenvectors of the covariance matrix and the corresponding eigenvalues are:

$$\begin{array}{c}
 V_1 = \begin{vmatrix} 0.356 \\ -0.279 \\ 0.480 \\ -0.031 \\ 0.035 \\ 0.009 \\ 0.56 \\ -0.296 \\ 0.402 \end{vmatrix}
 \end{array}
 \begin{array}{c}
 V_2 = \begin{vmatrix} -0.552 \\ -0.489 \\ 0.044 \\ -0.048 \\ 0.105 \\ -0.004 \\ 0.112 \\ 0.492 \\ -0.432 \end{vmatrix}
 \end{array}
 \begin{array}{c}
 V_3 = \begin{vmatrix} -0.264 \\ 0.347 \\ 0.309 \\ 0.064 \\ -0.222 \\ 0.078 \\ 0.585 \\ 0.401 \\ -0.391 \end{vmatrix}
 \end{array}$$

$$\lambda_1 = 153520 \quad \lambda_2 = 50696 \quad \lambda_3 = 22781$$

The eigenspace is defined by the projection matrix

$$V = \begin{vmatrix} 0.356 & -0.552 & -0.264 \\ -0.279 & -0.489 & 0.347 \\ 0.480 & 0.044 & 0.309 \\ -0.031 & -0.048 & 0.064 \end{vmatrix}$$

$$\begin{bmatrix} 0.035 & 0.105 & -0.222 \\ 0.009 & -0.004 & 0.078 \\ 0.56 & 0.112 & 0.585 \\ -0.296 & 0.492 & 0.401 \\ 0.402 & -0.432 & -0.391 \end{bmatrix}$$

The four centered training images projected into eigenspace are:

$$\begin{aligned} \mathbf{x}_1 = \mathbf{V}^T * \mathbf{X}_1' &= \begin{bmatrix} -103.09 \\ -117.31 \\ -96.57 \end{bmatrix} & \mathbf{x}_2 = \mathbf{V}^T * \mathbf{X}_2' &= \begin{bmatrix} -265.92 \\ 98.29 \\ 47.45 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \mathbf{x}_3 = \mathbf{V}^T * \mathbf{X}_3' &= \begin{bmatrix} 229.76 \\ 125.9 \\ -46.14 \end{bmatrix} & \mathbf{x}_4 = \mathbf{V}^T * \mathbf{X}_4' &= \begin{bmatrix} 139.24 \\ -106.88 \\ 95.26 \end{bmatrix} \end{aligned}$$

The test image viewed as a vector and the centered test image are:

$$\begin{aligned} \text{Test} &= \begin{bmatrix} 20 \\ 244 \\ 44 \\ 246 \end{bmatrix} & \text{Test Mean} &= \begin{bmatrix} -151.5 \\ 67.5 \\ -88.5 \\ -2.25 \end{bmatrix} \end{aligned}$$

21	-6.5
244	-2
4	-123.2
	5
255	49.5
2	-168

The projected test image is:

$$\begin{aligned} \text{Test}_{\mathcal{F}} &= V^T * \text{Test Mean}' \\ &= \begin{pmatrix} -266.6 \\ 5 \\ 80.75 \\ 50.6 \end{pmatrix} \end{aligned}$$

The L_2 norms are 296, 18, 508 and 449 of the test image Y_1 and the training images X_1 , X_2 , X_3 , X_4 respectively. By comparing the L_2 norms, the second training image X_2 is found to be closest to the test image Y_1 , therefore the test image Y_1 is identified as belonging to the same class of images as the second training image X_2 .

3.3 OpenCV

3.3.1 Overview

OpenCV stands for an open source computer vision library. It is originally developed by Intel. It runs on both operating system Windows and Linux. It is focused mainly towards real-time image processing. Other areas that the library covers are:

- Object Identification
- Face recognition
- Segmentation and recognition
- Motion understanding and tracking
- Mobile robotics
- Ego-motion
- Gesture recognition
- Structure from motion (SFM)
- Human-computer interface (HCI)

3.3.2 Steps for OpenCV

1. First you have to update ubuntu
`sudo apt-get update`
2. Then you have to upgrade ubuntu
`sudo apt-get upgrade`
3. Now execute following command on terminal. To install many dependencies. This dependencies are for reading image files, some important tools etc.
`sudo apt-get install build-essential libgtk2.0-dev libjpeg-dev libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev python-numpy python-tk libtbb-dev libeigen2-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev libx264-dev libqt4-dev libqt4-opengl-dev sphinx-common`

texlive-latex-extra libv4l-dev libdc1394-22-dev libavcodec-dev libavformat-dev
libswscale-dev

Following steps are for getting the OpenCV 2.4.1 source code:

4. `cd ~`
5. `wget`
<http://downloads.sourceforge.net/project/opencvlibrary/opencv-unix/2.4.1/OpenCV-2.4.1.tar.bz2>
6. `tar -xvf OpenCV-2.4.1.tar.bz2`
7. `cd OpenCV-2.4.1`

Now using `cmake` command you have to create Makefile. In this Makefile we have to define which parts of OpenCV we want to compile. Python, TBB, OpenGL, Qt, work etc are used in programs. So Makefile is the place where we have to set up Python, TBB, OpenGL, Qt, work with videos, etc.

8. `mkdir build`
9. `cd build`
10. `cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..`

Check that the above command produces no error and it reports FFMPEG as YES. If this is not there then you will not be able to read or write videos and images. Also, check that Python, TBB, OpenGL, V4L, OpenGL and Qt are detected which will be used in program.

This commands are for compiling and installing OpenCV 2.4.1 :

11. `make`
12. `sudo make install`

To configure OpenCV. First, open the `opencv.conf` file with the following code:

13. `sudo gedit /etc/ld.so.conf.d/opencv.conf`

Add the following line at the end of the file. This file may be empty but it is ok. and then save this file:

`/usr/local/lib`

For configuring library run the following code:

14. `sudo ldconfig`

Now open another file named `bash.bashrc` to add path:

15. `sudo gedit /etc/bash.bashrc`

Add these two lines at the end of the bash file and save this bash file:

```
PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig
export PKG_CONFIG_PATH
```

3.4 Implementation on cluster

Clustering is “the process of organizing objects into groups whose members are similar in some way”

A cluster is therefore a collection of objects which are “similar” between them and are “dissimilar” to the objects belonging to other clusters..

In proposed paper face recognition is executed on cluster for face recognition to improve the performance. When the dataset is large then on single PC face recognition takes more time. In proposed cluster based face recognition system, Person’s image dataset is divided on cluster of PCs. So there is less computation on single node. There are master are slaves. Master sends test image to slaves and slave find the nearest matching person using PCA from its dataset and send this result to master node. After slave send its result to master, on master

node results are compared and the image which has maximum confidence is selected as recognized image.

For this master slave communication MPI (*message passing interface*) is used. When whole dataset is on single PC, sometimes system will be slowed down or it stop working. In that situations cluster based face recognition is used. Execution time is less on cluster then single PC.

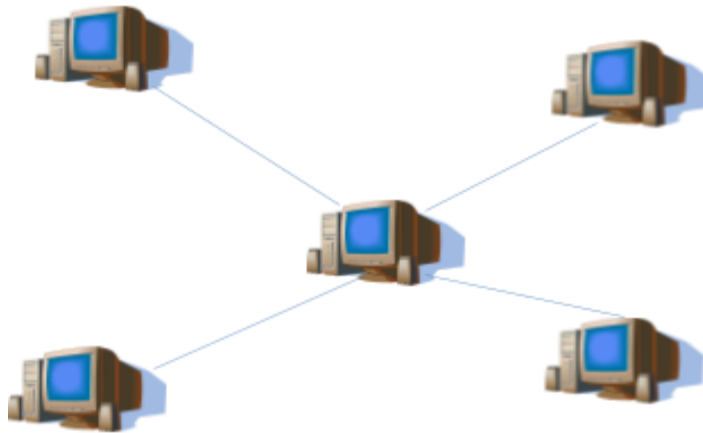


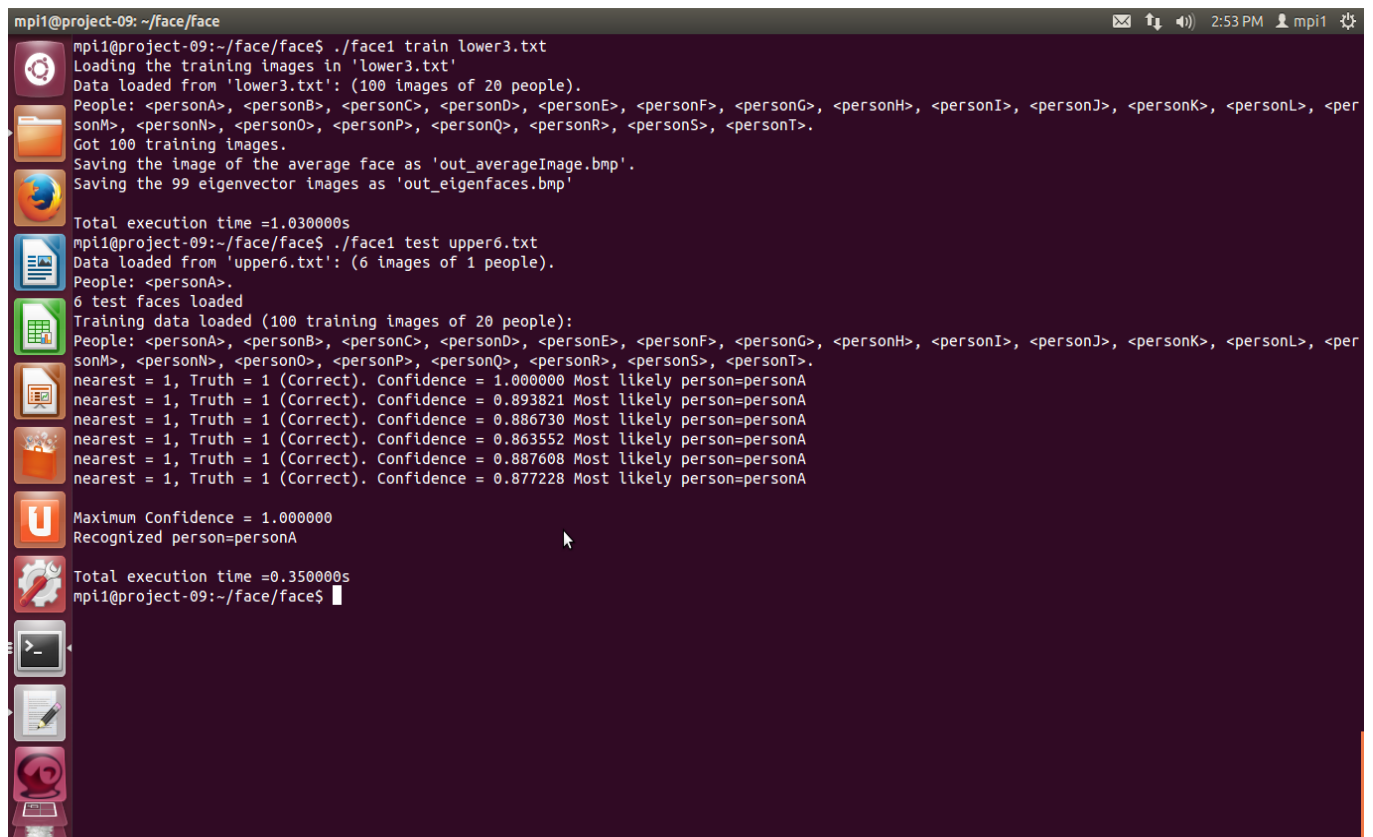
Fig 3.Cluster of five PCs

4. RESULTS & ANALYSIS

4.1 PCA on Single PC (100 images dataset)

We have performed PCA algorithm on single PC. We have used two different datasets of 100 images and 400 images.

Following fig. is the output for 100 images. In this dataset of 5 images of 20 people is used.

A screenshot of a Linux terminal window with a dark purple background. The terminal shows the execution of a PCA program. The user runs './face1 train lower3.txt', which loads 100 training images of 20 people and saves an average face and 99 eigenvector images. Then, the user runs './face1 test upper6.txt', which loads 6 test images of 1 person (personA). The program outputs the nearest neighbor and confidence for each test image, all correctly identifying personA with confidence values ranging from 0.863552 to 1.000000. The total execution time for training is 1.030000s and for testing is 0.350000s.

```
mpi1@project-09: ~/face/face
mpi1@project-09:~/face/face$ ./face1 train lower3.txt
Loading the training images in 'lower3.txt'
Data loaded from 'lower3.txt': (100 images of 20 people).
People: <personA>, <personB>, <personC>, <personD>, <personE>, <personF>, <personG>, <personH>, <personI>, <personJ>, <personK>, <personL>, <personM>, <personN>, <personO>, <personP>, <personQ>, <personR>, <personS>, <personT>.
Got 100 training images.
Saving the image of the average face as 'out_averageImage.bmp'.
Saving the 99 eigenvector images as 'out_eigenfaces.bmp'

Total execution time =1.030000s
mpi1@project-09:~/face/face$ ./face1 test upper6.txt
Data loaded from 'upper6.txt': (6 images of 1 people).
People: <personA>.
6 test faces loaded
Training data loaded (100 training images of 20 people):
People: <personA>, <personB>, <personC>, <personD>, <personE>, <personF>, <personG>, <personH>, <personI>, <personJ>, <personK>, <personL>, <personM>, <personN>, <personO>, <personP>, <personQ>, <personR>, <personS>, <personT>.
nearest = 1, Truth = 1 (Correct). Confidence = 1.000000 Most likely person=personA
nearest = 1, Truth = 1 (Correct). Confidence = 0.893821 Most likely person=personA
nearest = 1, Truth = 1 (Correct). Confidence = 0.886730 Most likely person=personA
nearest = 1, Truth = 1 (Correct). Confidence = 0.863552 Most likely person=personA
nearest = 1, Truth = 1 (Correct). Confidence = 0.887608 Most likely person=personA
nearest = 1, Truth = 1 (Correct). Confidence = 0.877228 Most likely person=personA

Maximum Confidence = 1.000000
Recognized person=personA

Total execution time =0.350000s
mpi1@project-09:~/face/face$
```

Figure 4.1 PCA on single machine(dataset of 100 images)

Figure 4.2 is the output of 400 image. In this dataset 10 images of 40 people are used.

```
mpi1@project-09: ~/singlenew db
mpi1@project-09:~/singlenew db$ cd singlenew\ db/
mpi1@project-09:~/singlenew db$ ./face train newlower.txt
Loading the training images in 'newlower.txt'
Data loaded from 'newlower.txt': (390 images of 39 people).
People: <personM1>, <personM2>, <personM3>, <personM4>, <personM5>, <personM6>, <personM7>, <personM8>, <personM9>, <personM10>, <personM11>, <personM12>, <personM13>, <personM14>, <personM15>, <personM16>, <personM17>, <personM18>, <personM19>, <personM20>, <personF1>, <personF2>, <personF3>, <personF4>, <personF5>, <personF6>, <personF7>, <personF8>, <personF9>, <personF11>, <personF12>, <personF13>, <personF14>, <personF15>, <personF16>, <personF17>, <personF18>, <personF19>, <personF20>.
Got 390 training images.
Saving the image of the average face as 'out_averageImage.bmp'.
Saving the 389 eigenvector images as 'out_eigenfaces.bmp'

Total execution time =36.700000s
mpi1@project-09:~/singlenew db$ ./face test newupper.txt
Data loaded from 'newupper.txt': (6 images of 21 people).
People: <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>, <personF1>.
6 test faces loaded
Training data loaded (390 training images of 39 people):
People: <personM1>, <personM2>, <personM3>, <personM4>, <personM5>, <personM6>, <personM7>, <personM8>, <personM9>, <personM10>, <personM11>, <personM12>, <personM13>, <personM14>, <personM15>, <personM16>, <personM17>, <personM18>, <personM19>, <personM20>, <personF1>, <personF2>, <personF3>, <personF4>, <personF5>, <personF6>, <personF7>, <personF8>, <personF9>, <personF11>, <personF12>, <personF13>, <personF14>, <personF15>, <personF16>, <personF17>, <personF18>, <personF19>, <personF20>.
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1
nearest = 21, Truth = 21 (Correct). Confidence = 1.000000 Most likely person=personF1

Maximum Confidence = 1.000000
Recognized person=personF1

Total execution time =4.380000s
mpi1@project-09:~/singlenew db$
```

Figure 4.2 PCA on single machine(dataset of 400 images)

4.2 PCA on cluster of three PCs

We have implemented PCA on cluster of 3 PCs. In cluster one master and two slaves are used. The dataset of 100 images is divided in two parts on slaves(50 images on each PC).

```

mpi1@project-09: ~/MPI/face new/3pc
6 test faces loaded
Training data loaded (50 training images of 10 people):
People: <personA>, <personB>, <personC>, <personD>, <personE>, <personF>, <personG>, <personH>, <personI>, <personJ>.
nearest = 1, Truth = 1 (Correct). Confidence = 1.000000 Most likely person: 'personA'
nearest = 1, Truth = 1 (Correct). Confidence = 0.796822 Most likely person: 'personA'
Training data loaded (50 training images of 20 people):
People: <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personL>, <per
sonM>, <personN>, <personO>, <personP>, <personQ>, <personR>, <personS>, <personT>.
nearest = 1, Truth = 1 (Correct). Confidence = 0.793404 Most likely person: 'personA'
nearest = 18, Truth = 1 (WRONG!). Confidence = 0.744399 Most likely person: 'personR'
nearest = 1, Truth = 1 (Correct). Confidence = 0.747735 Most likely person: 'personA'
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.741213 Most likely person: 'personM'
nearest = 1, Truth = 1 (Correct). Confidence = 0.809689 Most likely person: 'personA'
nearest = 19, Truth = 1 (WRONG!). Confidence = 0.751567 Most likely person: 'personS'
nearest = 1, Truth = 1 (Correct). Confidence = 0.784856 Most likely person: 'personA'
-----
recognized image::::::::::
Maximum Confidence = 1.000000 person =personA
-----
-----msg received from process 1 from project-10-----nearest=1 confidence=1.000000 person name=personA
nearest = 17, Truth = 1 (WRONG!). Confidence = 0.700634 Most likely person: 'personQ'
nearest = 18, Truth = 1 (WRONG!). Confidence = 0.724152 Most likely person: 'personR'
nearest = 17, Truth = 1 (WRONG!). Confidence = 0.752591 Most likely person: 'personQ'
-----
recognized image::::::::::
Maximum Confidence = 0.000000 person =personQ
-----
-----msg received from process 2 from project-09-----nearest=0 confidence=0.000000 person name=
-----
-----Recognized person name=personA-----
Total execution time=====0.534351
mpi1@project-09:~/MPI/face new/3pc$

```

In this the dataset of 400 images is divided in two slaves(200 images on each slave).

In this the dataset of 400 images is divided in two slaves(200 images on each slave).

[illegible]

37

4.3 PCA on cluster of five PCs

We have also implemented PCA on cluster of 5 PCs. In this cluster one master and four slaves are used. The dataset of 100 images is divided in 4 parts on slaves(25 images on each PC).

```
mpi1@project-09: ~/MPI/face new/SpC
Training data loaded (25 training images of 15 people):
People: <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personK>, <personL>, <personM>, <personN>, <personO>.
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.522108      Most likely person: 'personM'
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.564916      Most likely person: 'personM'
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.506146      Most likely person: 'personM'
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.508229      Most likely person: 'personM'
nearest = 13, Truth = 1 (WRONG!). Confidence = 0.487598      Most likely person: 'personM'
-----msg received from process 3 from project-09-----nearest=0 confidence=0.000000 person name=
nearest = 12, Truth = 1 (WRONG!). Confidence = 0.526317      Most likely person: 'personL'
-----
recognized image::::::::::
Maximum Confidence = 0.000000 person =personL
Training data loaded (25 training images of 20 people):
People: <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>, <personP>.
nearest = 18, Truth = 1 (WRONG!). Confidence = 0.528168      Most likely person: 'personR'
nearest = 18, Truth = 1 (WRONG!). Confidence = 0.530972      Most likely person: 'personR'
nearest = 19, Truth = 1 (WRONG!). Confidence = 0.567441      Most likely person: 'personS'
nearest = 17, Truth = 1 (WRONG!). Confidence = 0.463973      Most likely person: 'personQ'
nearest = 18, Truth = 1 (WRONG!). Confidence = 0.516605      Most likely person: 'personR'
nearest = 17, Truth = 1 (WRONG!). Confidence = 0.522218      Most likely person: 'personQ'
-----
recognized image::::::::::
Maximum Confidence = 0.000000 person =personQ
-----msg received from process 4 from Project-16-----nearest=0 confidence=0.000000 person name=
-----
-----Recognized person name=personA-----
Total execution time====0.259220
mpi1@project-09:~/MPI/face new/SpC$
```

Figure 4.5 PCA on cluster of 5 PC(dataset of 100 images)

In this the dataset of 400 images is divided in 4 slaves(100 images on each slave).

```
mpi1@project-09: ~/MPI/face new/newDB
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.757380      Most likely person: 'personM10'
nearest = 1, Truth = 1 (Correct). Confidence = -inf      Most likely person: 'personF1'
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.782653 Most likely person: 'personM20'

-----msg received from process 2 from project-09-----nearest=0 confidence=0.000000 person name=
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.758935      Most likely person: 'personM10'
-----
recognized image:::::::
Maximum Confidence = 0.000000 person =personM10
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.782455 Most likely person: 'personM20'
nearest = 1, Truth = 1 (Correct). Confidence = -inf      Most likely person: 'personF1'
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.783765 Most likely person: 'personM20'
nearest = 1, Truth = 1 (Correct). Confidence = -inf      Most likely person: 'personF1'

-----msg received from process 3 from Project-16-----nearest=0 confidence=0.000000 person name=
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.783224 Most likely person: 'personM20'
nearest = 10, Truth = 1 (WRONG!). Confidence = 0.000000 parson =personM20
maximum confidence=0.000000
final answer---matched image---0nearest = 1, Truth = 1 (Correct). Confidence = -inf Most likely person: 'personF1'
nearest = 1, Truth = 1 (Correct). Confidence = -inf      Most likely person: 'personF1'
-----
recognized image:::::::
nearest = 1, Truth = 1 (Correct). Confidence = 0.000000 parson =personF1

-----msg received from process 4 from Project-12-----nearest=0 confidence=0.000000 person name=
-----Recognized person name=personF1-----

Total execution time====4.785652
mpi1@project-09:~/MPI/face new/newDB$
```

Figure 4.6 PCA on cluster of 3 PC(dataset of 400 images)

4.4 ANALYSIS

Comparison of execution time (in seconds)

The following result table shows the time comparison of program execution on single PC and on cluster of different number of PCs.

No of training images in Database	Recognition time on single PC in seconds	Recognition time on cluster of 3 PCs in seconds	Recognition time on cluster of 5 PCs in seconds
100 images	1.38	0.5423	0.2547
400 images	41.12	13.350036	4.785652

Table 4.1 comparison of execution time

When we use dataset of 100 images at that time program takes 1.38 second on a single PC to recognize a person. Then we divide the dataset images on cluster of 3 PC in which one is master and other two are slaves. There are 50 images on each slave .This setup takes 0.5423 second to recognize a person which is 60.7% less than single PC. On cluster of 5 PC each slave contains 25 images. It takes 82% less time then single PC.

When we use dataset of 400 images at that time program takes 41.12 second on a single PC to recognize a person. Then we divide the dataset images on cluster of 3 PC in which one is master and other two are slaves. There are 200 images on each slave .This setup takes 013.350 second to recognize a person which is 67.5% less than single PC. On cluster of 5 PC each slave contains 100 images. It takes 88.3% less time then single PC.

Accuracy table of PCA(200 images)

No of testing images	No of training images(200)		
	True recognized image	False recognized image	Accuracy (%)
5	4	1	80
10	9	1	90
15	12	3	80

Table 4.2 Accuracy of PCA

Results of PCA

The result of Principal component analysis shows accuracy which is a recognition rate of input image. The parameters included in this result are number of training images used and number of testing images used in execution.

No of testing images	No of training images				
	Recognition rate(%)				
	100	150	175	200	400
5	60	80	80	80	100
10	50	60	70	90	100
15	46	53	66	80	100

Table 4.3 Results of PCA

For same number of testing images, as the training images are increased recognition rate is also increased.

5. CONCLUSION

What is trivial for a human eye may appear a very difficult task for the computer, but still computer vision can be very powerful and permits to perform very useful operations as the one we implemented in this project.

Principal component analysis approaches to the face recognition problem by means of information theory concepts. The most relevant information that is contained in a face image is extracted. Eigen faces method is a principal component analysis approach, where the eigenvectors of the covariance matrix of a small set of characteristic pictures are sought. These eigenvectors are called Eigen faces due to their resemblance of face images. Recognition is

performed by obtaining feature vectors from the eigenvectors space. We have implemented Euclidean distance method to recognize image from dataset.

We have implemented this face recognition algorithm on cluster of PCs which gives higher performance. It is useful when dataset of person's images are large then we can divide dataset on different machines. Execution time of this program is less on the cluster then the single computer. We used MPI interface for communication in cluster.

6. FUTURE WORK

We have implemented PCA but other methods like DCT (Discrete Cosine Transform), WHT (Walsh-Hadamard transformation) can also be implemented for extracting features from face.

This recognition system can also be extended for recognize face from video for that firstly face detection must be implemented. Once Face detected then we can easily recognize face using any of above algorithms and we can make application like video tracking. This Face Recognition system can also be extended for recognizing 3D-faces.

7. APPENDIX

Compilation steps for MPI are as follows:

- To compile mpi program

```
mpicc -o <object file> <source code>.c
```

- To run mpi program

```
mpiexec -n 4 /bin/hostname
```

Compilation steps for OpenCV are as follows:

- `gcc -ggdb `pkg-config --cflags opencv` -o first first.c `pkg-config --libs opencv``

Compilation step for OpenCV and MPI are as follows:

- `gcc -ggdb `pkg-config --cflags opencv` -I/home/mpi1/MPI/mpich2-install/include -o face face.cpp -L/home/mpi1/MPI/mpich2-install/lib -lmpich -lpthread -lrt `pkg-config --libs opencv` -lopenmp`

8. REFERENCES

1. Prof. Y. Vijaya Lata, Chandra Kiran Bharadwaj Tungathurthi, H. Ram Mohan Rao, Dr. A. Govardhan, Dr. L. P. Reddy, Facial Recognition using Eigenfaces by PCA

2. Matthew A. Turk , Alex P. Pent land , Vision and modeling group, the media laboratory, Massachusetts institute of technology, Face Recognition using eigenface
3. Preeti B. Sharma, Mahesh M. Goyani, 3D FACE RECOGNITION TECHNIQUES - A REVIEW
4. William Gropp a31, Ewing Lusk , Nathan Doss bq2, Anthony Skjellum , A high-performance, portable implementation of the MPI message passing interface standard
5. William. D. Gropp , Learning from the Success of MPI
6. R. Chellappa, C.L. Wilson, and S. Sirohey, “Human and machine recognition of faces: a survey”, *Proceedings of the IEEE*, Vol.83, No.5, 1995, pp.705–741.
7. P. S. Huang, C. J. Harris, and M. S. Nixon, “Human Gait Recognition in Canonical Space using Temporal Templates”, *IEE Proc.-Vis. Image Signal Process*, Vol. 146, No. 2, April 1999.
8. Ahmet Bahtiyar Gul, “Holistic Face Recognition by Dimension Reduction”, Master thesis, Dept. of Electrical and Electronics Engg., Sept – 2003.
9. Manojkumar Tewari, “Study of different algorithms for Face Recognition”, Bachelor thesis, Department of Electronics and Communication Engg., NIT, Rourkela, India, 2010.
10. G.R.G. Lancriet, L. El Ghoui, C. Bhattacharyya and M.I.Jordan, “Minmax probability machine”, In proc. Advances in Neural Information Processing System, 2002.

Web references

1. <http://msdn.microsoft.com/en-us/library/tt15eb9t.aspx>
2. [http://www.indusedu.org/IJRIESS/November2012\(Pdf\)/27.pdf](http://www.indusedu.org/IJRIESS/November2012(Pdf)/27.pdf)
3. <http://sourceforge.net/projects/opencvlibrary/>
4. <http://forrobot.blogspot.in/2012/08/installing-opencv-23-on-visual-studio.html>
5. http://isa.umh.es/pfc/rmvision/opencvdocs/ref/OpenCVRef_BasicFuncs.htm
6. http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html#
7. http://www.ijesit.com/Volume%202/Issue%203/IJESIT201303_17.pdf
8. [http://www.indusedu.org/IJRIESS/November2012\(Pdf\)/27.pdf](http://www.indusedu.org/IJRIESS/November2012(Pdf)/27.pdf)
9. http://en.wikipedia.org/wiki/Principal_component_analysis
10. http://en.wikipedia.org/wiki/Principal_component_analysis#Properties_and_limitations_of_PCA
11. http://doras.dcu.ie/17440/1/wu_hai_RR.pdf
12. http://www.bytefish.de/blog/pca_in_opencv/
13. http://isa.umh.es/pfc/rmvision/opencvdocs/ref/OpenCVRef_ObjectRecognition.htm
14. http://en.wikipedia.org/wiki/Three-dimensional_face_recognition
15. http://en.wikipedia.org/wiki/Active_appearance_model
16. http://www.cognotics.com/opencv/servo_2007_series/part_5/index.html
17. https://github.com/samvit/Face-Navigation/tree/master/Cambridge_FaceDB