# Face Recognition for Single and Different Facial Expressions

GJCST Computing Classification
I.5.4 , I.4.6

Rashedur M Rahman[1] Anirban Das[2] M Russel[3] Md. Shazzad Maruf[4]

***Abstract*- This paper presents and analyzes the performance of Principle Component Analysis (PCA) based technique for face recognition. We consider recognition of human faces with two facial expressions: single and differential. The images that are captured previously constitute the training set. From these images eigenspaces/eigenfaces are calculated. The image that is going to be recognized through our system is mapped to the same eigenspaces. Next two classification techniques, namely distance based and neural network based classifier are used to classify the images as recognized or non-recognized. In this research we categorize the face images into six different test cases and discuss the performance of each test case with various performance metrics. Our experimental results demonstrate that the neural network technique outperforms the distance based classifier in most of the test cases.**

*Keywords*- principal component analysis, neural network, face recognition, image processing.

## I. INTRODUCTION

Face is an essential element of focus of our daily life. We convey our identity and emotions through our face and different expressions of faces respectively. Though human faces are complex in shape, face recognition is not difficult for a human brain whereas for a computer this job is not easy. The complexity of recognition is prominent and several algorithms are reported in literature [2,4-7] that could achieve the recognition with high degree of accuracy. Face recognition system is widely used in different areas [9] that include a) criminal record and identification, b) Robot vision, c) security system, d) human computer interaction, e) image and field processing.

Face recognition system is divided into two categories, i) appearance based and ii) component based. For appearance based, we consider the holistic feature or the whole face image as our feature for recognition. On the other hand, in component based face recognition, we consider geometrical relationship of different components of face such as eye, nose, lip etc as the features of a recognition system. Principal Component Analysis (PCA) [3,4] is a fast and efficient technique that is widely used for appearance based face recognition. This technique is also used for dimensionality reduction in different areas that include image processing, signal processing and data mining. This technique is sometimes also called eigenfaces [3].

*About -[1] Rahman, R.M. is with the Department of Electrical Engineering and Computer Science, North South University, Dhaka, Bangladesh (telephone: 8802-8852000-1507 email: rashedur@northsouth.edu)*
*About- [2]Das, A., [3]Russel, M, [4]Maruf, S. Department of Electrical Engineering and Computer Science, North South University, Dhaka, Bangladesh.*

The eigenfaces approach is chosen for this study considering its capability of recognizing real time images where orientation, illumination and distortion are continuously changing. This work focuses on how the images with real time attributes affect the recognition feature of eigenfaces technique. Our primary objective for this research is to minimize the complexity in calculation for bigger matrices. For example, if we have 120 pictures with the size of (180 × 200), we will have a very big number while calculating the one dimensional vector from 2D matrix (by calculating 180 × 200 × 120) which is a very big number. By using the eigenvectors, we could minimize the use of all the images and reduce them for example 40 pictures which will also bring down our total calculation to (180 × 200 × 40). Though, we are using lesser amount of data, we will still get the same level of accuracy. Besides, we could make the size even smaller by changing the order of matrix multiplication which in turn reduces the principal components, and the end we could work only on (40×120) matrix with the same level of accuracy.

Rest of the paper is organized as follows. Section 2 describes the methodologies used in this research in detail. Section 3 describes about the system and execution flows of different components in the system. Section 4 presents and analyzes the result. Finally Section 5 concludes and gives direction of future research.

## II. METHODOLOGIES

Our face recognition system consists of several steps. Each of the steps is described in detail in below:

*Initialization and Finding Principal Components*

At first we take images. These images are nothing but the matrix which has pixel intensity at different rows and columns. This image could be viewed as a vector also. If an image has height, *h* and width, *w*, then we could formulate this image as *w* vectors, where each vector has *h* dimensions. The rows of the images are placed one after another like the Figure1 below:

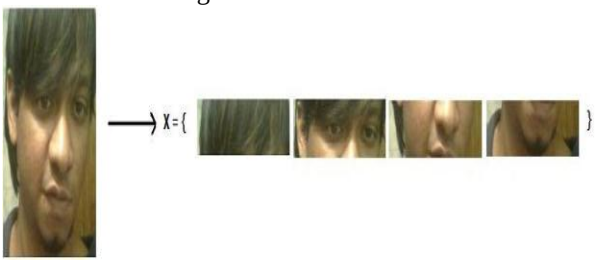

Fig 1: Formation of the face's vector from the face's images

The vector which is $w \times h$ represents our image and this image has a certain space so this is called image space. If we have $N$ images, we have image space dimension as $N \times w \times h$. In this image space all images are represented by $w$ by $h$ pixels. These images under same image space look like each other. They all have two eyes, a nose, a mouth etc located at the same image space.

Now we will build the face space from the image space. The main task of building a face space is to describe the face images. The basis vector of this space is called principal component. The dimension of the face space will be $M \times w \times h$. In the face space all pixel is not relevant and each pixel depends on the neighbors. So the dimension of face space is less than the dimension of the image space. We could find the principle components of the faces by finding the eigenvectors of the covariance matrix of the set of face images. This eigenvectors are basically a set of features which characterize to the maximum variations between face images. Each of this images that comes from the image space contribute more of less to the eigenfaces. So we can display eigenvector as a sort of ghostly faces which we call eigenfaces. Actually eigenfaces do not exist in real world. We could not say we can build or create eigenface of a particular image face which is in the image space. Eigen face actually is an imaginary face which is a combination of all the images with in a particular image space. Figure 2 presents eight eigenfaces from a sample image database presented in Figure 5.

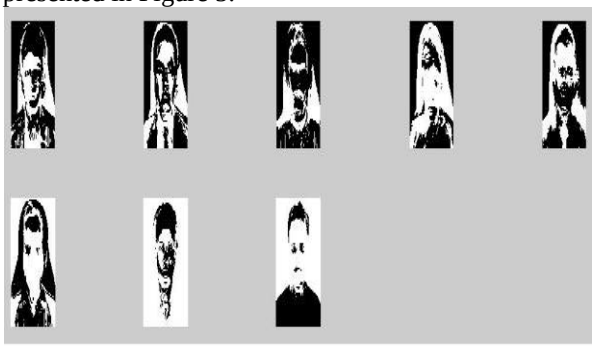

Fig2: Eigenfaces from the image database.

We present the mathematical formulation of eigenfaces below. More details of the formulation could be found elsewhere [5,6].

1. We obtain $N$ training images $I_1, I_2 \ldots\ldots I_N$. Each of these images have dimension $w \times h$. Convert these images into vector space by concatenation. After the concatenation a matrix is converted to a vector. An example of concatenation is given in the following page.

Represent each image $I_i$ with its corresponding vector $\lambda_i$ .

$$\begin{bmatrix} B_{11} & B_{12} & . & B_{1h} \\ . & . & . & . \\ . & . & . & . \\ B_{w1} & B_{w2} & . & B_{wh} \end{bmatrix} \xrightarrow{\text{Concatenation}} \begin{bmatrix} B_{11} \\ . \\ . \\ B_{1h} \\ . \\ . \\ . \\ B_{wh} \end{bmatrix} \equiv \lambda_i$$

3. Calculate the mean face vector $\varpi$ by the following equation

$$\varpi \equiv \frac{1}{N}\sum_{i=1}^{N}\lambda_i$$

Subtract the mean face, $\varpi$ from each face vector, $\lambda_i$ to get a set of vectors, $\mu_i$ .

$$\mu_i = \lambda_i - \varpi$$

The purpose of subtracting the mean image from each image vector is to keep only the distinguishing features from each face by removing the common information.

Find the covariance matrix $C$ by the following equation:

$$C = A^T A \qquad \text{where,} \qquad A = [\mu_1, \mu_2, \ldots\ldots\ldots \mu_N]$$

Find the eigenvalues and eigenvectors for the covariance matrix, $C$. Sort the eigenvectors according to the eigenvalues. Take the first $M$ eigenvectors that have higher eigenvalues. Now each eigenvector will have $N \times 1$ dimension. Let us name those eigenvectors as $\eta_i$ for $i=1,2,\ldots M$.

*Projection of new face to eigenfaces*

When a new image is encountered, calculate the set of weights based on the new or input image and the $M$ eigenfaces by projecting the input image onto each of the eigenfaces. The mathematical formulation is given below:

Let us consider the new image as $I_{new}$

Find out the $M$ eigenface components, $\psi_l$ , by projecting the new image

$$\psi_l = \gamma_l^T (I_{new} - \varpi) \text{ for } l=1,2,\ldots M$$

where,

$$\gamma_l = \sum_{k=1}^{N} \eta_{lk} \mu_k \text{ for } l=1,2,..M$$

Create a new feature vector, $\Omega_{new}$ for the new image by concatenating eigenface components, $\psi_l$

$$\Omega_{new} = [\psi_1 \quad \psi_2 \quad . \quad . \quad \psi_M]$$

*Face Recognition by classification algorithms*

The last step of the face recognition system is to identify the new face to be recognized or not recognized. If the face is recognized the system will tell the person's name for whom the face has been recognized. In the other word, if we have *N* persons in the image database, we say that there are *N* classes where each individual person representing a class. There are two algorithms used for classification, one is distanced based and the other one is neural network based classification.

The distance based classifier works in the following way:

For each image in the image database, find out the feature vector $\Omega_i$ for *N* persons where $i=1,2,...N$ . The procedure will be same that is discussed for the new image in the earlier sub section.

Classification is performed by comparing the feature vector of new image, $\Omega_{new}$, with the feature vector of the images in the image database.

Comparison is done by the Euclidian distance between two features, $\Omega_{new}$ and $\Omega_i$, if the distance is less than some predefined threshold, *t*, we say that the image is recognized. The class of the new image will be one that has the least Euclidian distance with the new image, providing this distance is less than the threshold.

We also use Neural Network (NN) for classification of the new image to the image database. The idea used in this research is inspired by the research paper[8]where the authors proposed PCA based neural network technique for face recognition. A fully connected, layered, feed-forward network is depicted in Figure 3, where $x_i, h_i, o_i$ represent unit activation levels of input, hidden, and output units, respectively. Weights on the connections between the input and hidden layers are denoted by $w1_{ij}$, while weights on connections between the hidden and output layers are denoted by $w2_{ij}$. The neurons marked with "1" are threshold neurons and their activation value is set to 1. Figure 3's network has three layers, although it is possible and sometimes useful to have more layers. Each unit in one layer is connected in the forward direction to

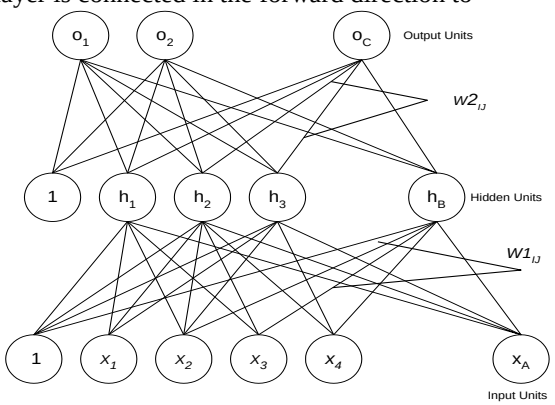


Fig. 3: A Multilayer Feed-Forward Neural Network

every unit in the next layer. Activations flow from the input layer, through the hidden layer, to the output layer. The knowledge of the network is encoded in the weights on connections between units. A backpropagation network typically is initialized with a random set of weights. The network adjusts its weights each time it processes an input-output pair. More details about neural network and its training algorithm could be found elsewhere [8,10,13].

Our neural network algorithm for classification work in the following way:

Randomize the weights to small arbitrary values and initialize the activations neurons. The neurons marked as "1" in Figure 3 are activation neurons. Their values are set to 1 in this step.

Select a training pair from the training set. A training pair consists of an input and output vectors. The input will be the projected feature vector $\Omega_i$, and the output will be the class label. For $\Omega_i$ input vector class label will be *i*. In other word this projected vector represents person *i* in the image database.

Apply the input vector to the network input neurons.

Propagate the activations from the input neurons of the input layer to the hidden neurons of the hidden layer using the activation function.

Propagate the activations from the neurons of the hidden layer to the neurons of the output layer.

Calculate the error, the difference between the network output, and the desired output. The desired output is the output vector from the training pair and the network output is calculated by activation of output neurons. These errors are the errors of the neurons in the output layer.

Compute the errors of the neurons in the hidden layer

Adjust the weights of the network between the hidden layer and output layer.

Adjust the weights between the input layer and the hidden layer. The error adjustment in steps 8-9 use the gradient decent method [13].

Repeat Steps 2-9 for each pair of input-output vectors (we have *N* images so *N* vectors) in the training set until the error for the entire system is acceptably low.

When the network is trained the weights are adjusted accordingly. If we now present the testing image data $I_{new}$ through its projected vector $\Omega_{new}$ to the input unit of the neural network, at most one of the output neurons, for example, *i* neuron will fire that corresponds to the highest match to the input image. We classify the image to be recognized as person *i* .

## III. SYSTEM DESCRIPTION

We have developed our system by using MATLAB 2008a ( version7.6).because we found that the performance of MATLAB 2008a (version 7.6) [14] is better than other programming language. Besides, MATLAB is a high-level language for technical computing development environment for managing code, files, and data interactive tools for iterative exploration, design, and problem solving. It also supports Mathematical functions for linear algebra,

statistics, Fourier analysis, filtering, optimization, and numerical integration 2-D and 3-D graphics functions [1,11,12] for visualizing data. It has also tools for building custom graphical user interfaces and neural network toolbox that we use for classification. For our system we have used certain method and techniques that are offered explicitly by MATLAB. These are as follows:

A) *Image reading*: MATLAB can easily read an image and convert the image in a certain matrix. Later on we can use the image matrix for our related work. MATLAB can read an image of 8 bit up to 32 bit .

B) *Image conversion from RGB to GRAYSCALE*: MATLAB can convert an image from RGB to GRAYSCALE. This computational task can be done by MATLAB command. If RGB image is 32 bit it represent RED for 8 bit, GREEN for 8 bit, BLUE for 8 bit.

C) *Image resize*: MATLAB command can be used to resize a certain image in to any size that MATLAB allow.

D) *Convert MATRIX to 1 dimensional VECTOR*: We can use certain technique to convert a matrix to 1 dimensional vector which helps us to compute the desired out put.

E) *Matrix transpose*: In MATLAB we can easily transpose a certain matrix.

In our system, at first of the system take the images. These images are captured by webcam or other image capturing source. We calculate the mean image and eigenfaces.




Figure 4: Basic flow diagram of the face recognition system

Then, we take the input image that will be detected later on. We process the Input image and compare the input image with training image set if any match below the threshold value than we can say it is recognized other wise not. The detail flow of execution is given in Figure 5

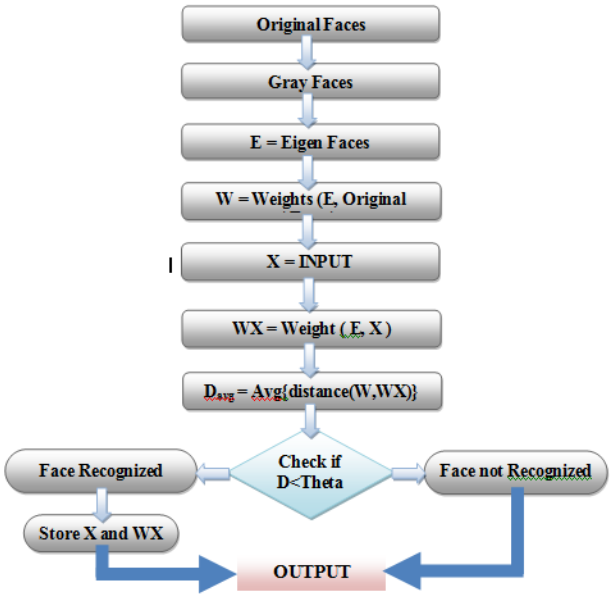


Figure 5: Detail flow of execution in the system

## IV. Results and Analysis of Results

In total, six test cases were chosen to test the performance of face recognition system. All the images were in same size. The training set images were captured such that the person faces were frontal with minimal head tilt and had decent varied expressions. Tests were taken in well illuminate area to recognize, at least, the known images. We have to do 2 types of testing

1.Face image with single facial expression.

2.Face image with different facial expression.

Figure 5 and Figure 6 presents some sample image for single and differential facial expressions respectively.



Fig 5. Sample dataset for single facial expression



Fig 6. Sample dataset for differential facial expression

The system performs relatively well in all test cases.
We apply our algorithm for six test cases. The test cases are described below:

**Test Case 1:** Test case 1 is measuring the impact of having large number of training images of a small group of persons. Altogether 30 images of 3 persons (10 images from each) were taken to create the training set. Different facial expression of each person is taken for this case.

**Test Case 2:** Test case 2 is measuring the impact of having large number of training images from a large group of persons. Altogether 50 images of 50 persons were taken to create the training set. The images are for single facial expression.

**Test Case 3:** Test case 3 is measuring the impact of having small number of persons in the training set. In total, 6 images from each of the 2 persons were considered to create the training set. So the total number of images is 12.

**Test Case 4:** Test case 4 is measuring the impact of having small number of images in the training set. In total, 20 images from 20 persons were considered to create the training set.

**Test Case 5:** Test case 5 is measuring the impact of having very small number of images in the training set. Altogether 3 images of 3 persons (per person image no is 1) were taken to create the training set.

**Test Case 6:** Test case 6 is measuring the impact of having small number of images in the training set. Altogether 3 images of 10 persons (in total 30 images) were taken to create the training set.

Table 1 summarizes all the test cases and the number of images in total used for recognition system. The accuracy reported here is the accuracy for distance based classifier.

*Table 1: Accuracy of Recognition*

| Test Case | Image/ person | Number of People | Total Images | Total Testing Images | Accuracy |
|---|---|---|---|---|---|
| Case1 | 3 | 10 | 30 | 50 | 80% |
| Case2 | 1 | 50 | 50 | 70 | 100% |
| Case3 | 6 | 2 | 12 | 24 | 37.5% |
| Case4 | 1 | 20 | 20 | 40 | 80% |
| Case5 | 1 | 3 | 3 | 5 | 95% |
| Case6 | 3 | 10 | 30 | 36 | 83% |

Figure 7 compares the accuracy between neural network and distanced based classifier. For most of the test cases, neural network outperforms the distance based classifer.

For the distance based classifier, the Euclidian distance between the new image and all the images in image database are calculated. Then, this distance has to be within two threshold values. The first threshold value is used to screen this image to be a valid face image or not. For example, if the image is for a flower or house then the Euclidian distance will be higher between flower and other facial images in the database.

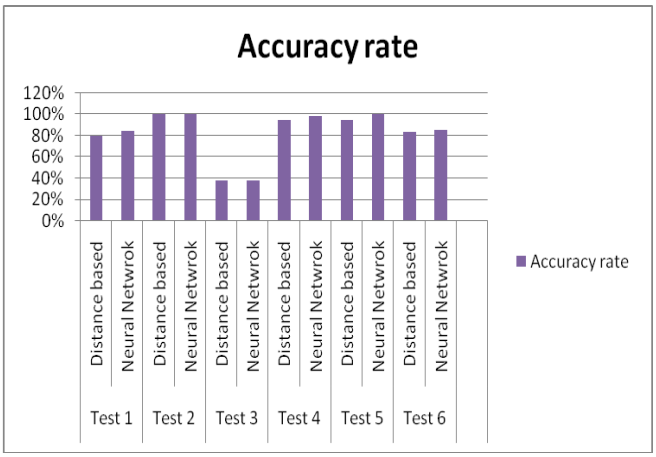


Fig 7. Comparison of accuracy between neural network and distance based classifier.

The system will let user know that this is not a valid face image as the distance is greater than the threshold. The next threshold is used to determine whether the image falls under the images in the database. The second threshold is chosen by trial and error method. Table 2 presents different threshold values and corresponding mismatch rates. If we increase the threshold value we could increase the probability of recognition as a face image. However, we should be careful about not to increase this value to much such that misclassification will increase due to non facial image classification to a facial image.

*Table 2: Mismatch vs. Threshold*

| Mismatch | 69% | 32% | 12% | 1% |
|---|---|---|---|---|
| Threshold | 0.3 | 0.4 | 0.5 | 0.6 |

We also analyze the impact of number of eigenvectors on the accuracy. We select the test case 6 for this purpose. Figure 8 depicts this situation. By increasing the number of eigenvectors increases the performance of the classifier but up to an extent. If we plot the accuracy on *y* axis against the number of eigenvectors on *x* axis, we will get the following graph.

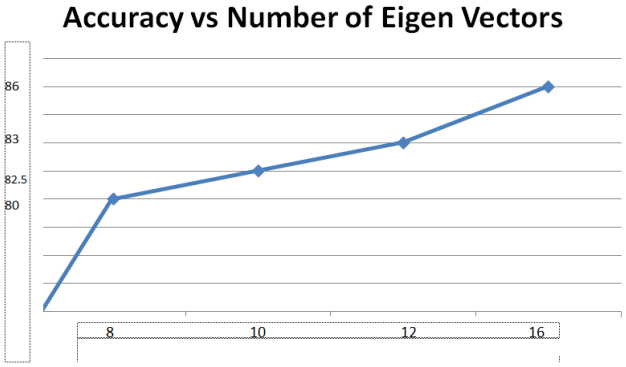


Fig 8: Accuracy vs. Number of eigenvectors.

## V. CONCLUSION

From the tests and analyses performed in this research, we conclude with the following remarks. Training set and test images need to be taken in good, comfortable illumination settings and need to be frontal faces with minimal head tilt. Number of images in the training set is a significant factor. It impacts on defining the correct threshold value for accepting true matches and rejecting false matches. The system performs relatively well with larger training sets and reflects similar behavior irrespective of the number of persons present in those larger training set. However, increase in the training set size increases the system performance and varies the acceptance or rejection rate depending on the person group size. Neural network performs better than the distance based classifier in most of the test cases studied in this research.

## VI. REFERENCES


1) Andrews, H.C. Computer Techniques in Image Processing, Academic Press, New York.
2) Bajcsy and A. Kovacic. Multiresolution elastic matching. Computer Graphics and Image Processing 46:1-21, 1989.
3) Baumberg, A. and D. Hogg. An adaptive eigenshape model. In proceedings of 6th British Machine Vision Conference, pages 87-96. BMVA Press, Sept. 1995.
4) Belhumuer, P., Hespanha J. Kreigman. Eigenface vs. Fisherface using class specific linear projection. IEEE transaction on pattern analysis and machine intelligence.19 711-720 , 1997.
5) Belongie, S., J. Malik, and C. Fuh. Matching shapes. In 8th International Conference on Computer Vision, volume 1, pages 454-463. IEEE Computer Society Press, July 2001.
6) Brunelli R, Poggio T . Face Recognition Features Vs. Templates. IEEE transaction on pattern analysis and machine intelligence.15 :1042-1052, 1993.
7) Delac and Mislav Grgic. Face Recognition Edited by Kresimir, I-Tech Education and Publishing, Vienna, Austria, 2007.
8) Eleyan, A., Demirel, H. Face Recognition System Based on PCA and Feedforward Neural Networks. Computational Intelligence and Bio-inspired Systems. Lecture Notes in Computer Science(LNCS), Volume 3512, pp.935-942, 2005.
9) Face Recognition Website. Retrieved on 28th. June, 2010. http://www.face-rec.org/
10) Fausett, L. Fundamentals of Neural Network: Architecture, Algorithms, and Applications. Prentice Hall, Upper Saddle River, NJ, 1994.
11) Gonzalez, R.C., Richard E. Woods. Digital image processing, 2nd edition. Pearson Education, 2002.
12) Gonzalez, R.C. Digital Image Processing by MATLAB, 3rd edition Pearson Education, 2004.
13) Ham, F. M and I. Kostanic. Principles of Neurocomputing for Science and Engineering, McGraw Hill Higher Education, New York, NY, 2001.
14) MATLAB User guide, Retrieved on 28th. June, 2010. http://www.mathworks.com/access/helpdesk/help/techdoc/.
15) Sereno, Santiago. Master of Science (M.Sc) Thesis, Drexel University, Retrieved on 28th. June, 2010. http://www.pages.drexel.edu/~sis26/Computer%20Vision.htm.