UNIVERSITY OF BURGUNDY

APPLIED MATHEMATICS PROJECT

MASTERS IN COMPUTER VISION AND ROBOTICS

FACE RECOGNITION USING PCA

Submitted by

KODIPAKA VAMSHI

FABIO CYRO RIBEIRO

Submitted to

Dr. Désiré Sidibé

CONTENTS

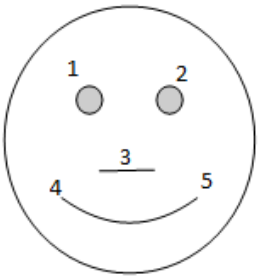
ABSTRACT

In recent years, the demand for face recognition systems has increased very much for identification and the verification purposes. There are different systems like Fingerprint recognition, Iris recognition, Retina recognition, voice recognition and so on. Face recognition, in particular has received a considerable attention in recent years for identification and the verification purposes because of its applications. At present, there are many methods for face recognition.

Applied Math's concepts play crucial role in this time to make the technology easy and efficient. For instance google page ranking using linear algebra, for face recognition using PCA and SVD. Here in our project, we used Principal Components Analysis (PCA) and Singular value decomposition (SVD) for face recognition. In PCA, we find eigenvalues and eigenvectors of the covariance matrix of the set of face images by using the Eigen vectors. We are able to detect the face of the person.

1. Introduction

Principal Component Analysis (PCA) is an approach to recognize and locate the similar patterns from a given set of data. PCA has been implemented in various applications like face recognition, hand written text matching and in image compression. PCA plays a vital role in image compression as we compress the data by reducing dimensions with no loss of data.



Principal Components Analysis (PCA) is a practical and standard statistical tool in modern data analysis that has found application in different areas such as face recognition, image compression and neuroscience. It has been called one of the most precious results from applied linear algebra. The Principal Component Analysis (PCA) is one of the most successful techniques that have been used in image recognition and compression. PCA can be used for many reasons such as Simplification, data Reduction, modeling, outlier detection, variable selection, classification, prediction and so on. But in here the purpose of PCA is to reduce the large dimensionality of the data space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which is needed to describe the data economically.

The main aim of project is for the face recognition gathered arrangement of face information. We have gathered the five pictures of every student and separate them into two categories of train and test images. The train images should be selected as one facing the camera and the other two of side face. Every image is selected as five facial features: left eye, right eye, tip of nose, left mouth corner and right mouth corner as presented in below.

2. Methodology

PCA has 3 main concepts:

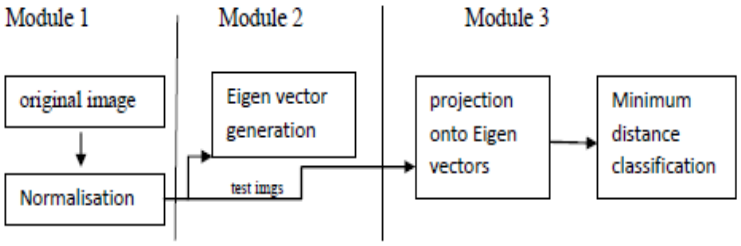


Figure 2 Block diagram of PCA-based Face Recognition system

Step1:: Normalization

We need to normalize the images as required to scale the image in the proper way as per the needs of affine transformation. From the provided facial features, we have the predefined values for 64×64 image as left eye (13, 20), righteye (50,20), Tip of nose (34,34), mouth left corner (16,50), mouth right corner (48,50). We must find the affine transform ::

$$f_i^p = Af_i + b$$

Algorithm for Normalization

Step 1: We take the predefined feature co-ordinates

Step 2: We consider all the features f_i and then we calculate the affine transformation by the equation

$$f_i^p = Af_i + b$$

Where A and b are 6 unknowns

Step 3: We update **F_bar** each time by using SVD

Step 4: We take the average of all the aligned feature locations for each face image and update **F_bar** with this value

Step 5: we compare $\mathbf{F_t}$ and $\mathbf{F_{t-1}}$. If the difference is less than a threshold, then stop and come out of the loop and the final converged **F_bar** gives the affine transformation matrix A and vector b; otherwise, go to step 2.

After calculating the affine transforms, we will apply transform to images and save them in a folder. After the transformation of the images we will convert them into 64×64 images.

Step2:: Face Recognition

For face recognition, all the images are divided into two parts: train and test images. Train images should be selected as one front view and two other side facial views which makes train images and the rest are named as test images.

First we put all the images in the training set and the training set in stored in a single matrix D

$$D = \begin{bmatrix} I_1(1,1) & I_1(1,2) & \dots & I_1(1,N) & \dots & I_1(M,1) & I_1(M,2) & \dots & I_1(M,N) \\ I_2(1,1) & I_2(1,2) & \dots & I_2(1,N) & \dots & I_2(M,1) & I_2(M,2) & \dots & I_2(M,N) \\ & & \vdots & & \vdots & & \vdots & & \\ I_p(1,1) & I_p(1,2) & \dots & I_p(1,N) & \dots & I_p(M,1) & I_p(M,2) & \dots & I_p(M,N) \end{bmatrix}_{p \times d}$$

Now, we compute the covariance matrix of D by

$$\sum = \frac{1}{p-1} D^T D$$

We have find the eigen vectors of all the train images and compare them with test images to find the most accuracy match. All the train images are labelled in a row which is denoted by ‘p’ and the variables are labelled in columns of matrix are denoted as ‘d’ which makes 4096 (96×96).

Names of the persons are denoted in label matrix which takes the first three characters of a name. Data matrix is represented as:

$$D = [\quad]_{p \times d}$$

We will now find the mean of each image which gives 1×d matrix and the mean matrix is represented as $p \times d$. Then, subtract the mean with the data matrix. Now we compute the covariance of the matrix D

$$D^T D = (d \times p)(p \times d) = d \times d$$

$$DD^T = (p \times d)(d \times p) = p \times p$$

Here $D^T D$ is a very large matrix, so it will be tough to compute while DD^T is relatively a small matrix to compute. So, we will consider with this matrix and find the eigen vectors.

We require eigen vectors of $D^T D$.

Let X be eigen vector of DD^T then $(DD^T)\,X = eX$

Multiply both sides with D^T to get :: $(DD^T)\,D^T X = e(D^T X)$

From above $D^T X$ is eigen vector of $D^T D$

After finding all the eigen vectors we will arrange them into decreasing order of eigen values and then find the PCA transformation matrix. This projection of train images into PCA space gives p×p matrix. Finding the feature vector of train images

$$D^T V = (d \times p)\,(p \times p) = d \times p = \Phi$$

$$\text{Feature vector} = D\Phi = (p \times d)\,(d \times p) = p \times p \qquad \Sigma = {(DD^T)}/{(p-1)}$$

From the above equation, the columns of p can be varied

We have trained and check with the test images now. The feature vector for test images is given as:

$$\text{Feature vector} = I\Phi = (1 \times d)\,(d \times p) = 1 \times p$$

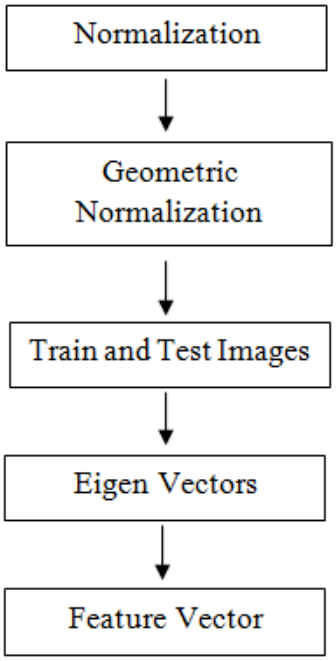
From the above equation, the p is variable. Taking every single test images & compare with the feature matrix of trained images unless we find the most least Euclidean distance between them. The distances are arranged in ascending order & represents k as an

integer to find for the given no. of images. It keeps checking for the matched image and error is increased every image it checks & display the accuracy when it is matched with similar facial features.

Accuracy is given as:

$$\text{Accuracy} = (1 - \text{error}/\ \text{TestImages}) \times 100$$

Algorithm for Recognition



Step 1: Set the number of Principal components

Step 2: Build a training data matrix D from training images

Step 3: We assign name as the label for each training image and form a labels matrix Lt

Step 4: Remove the mean and compute covariance matrix (D). Then compute the PCA of D, which gives PCA transformation matrix

Step 5: Feature vectors of all training images using transformation matrix and store them in a matrix Ft

Step 6: Read test images, and for each test image Iq

– Calculate the feature vector ϖq using the PCA transformation matrix

– Compute the distance between ϖq and all training feature vectors, which are stored in Ft

– Find the best match Iz (minimum distance between feature vectors)

– Check if the label of Iq is the same as the label of Iz.

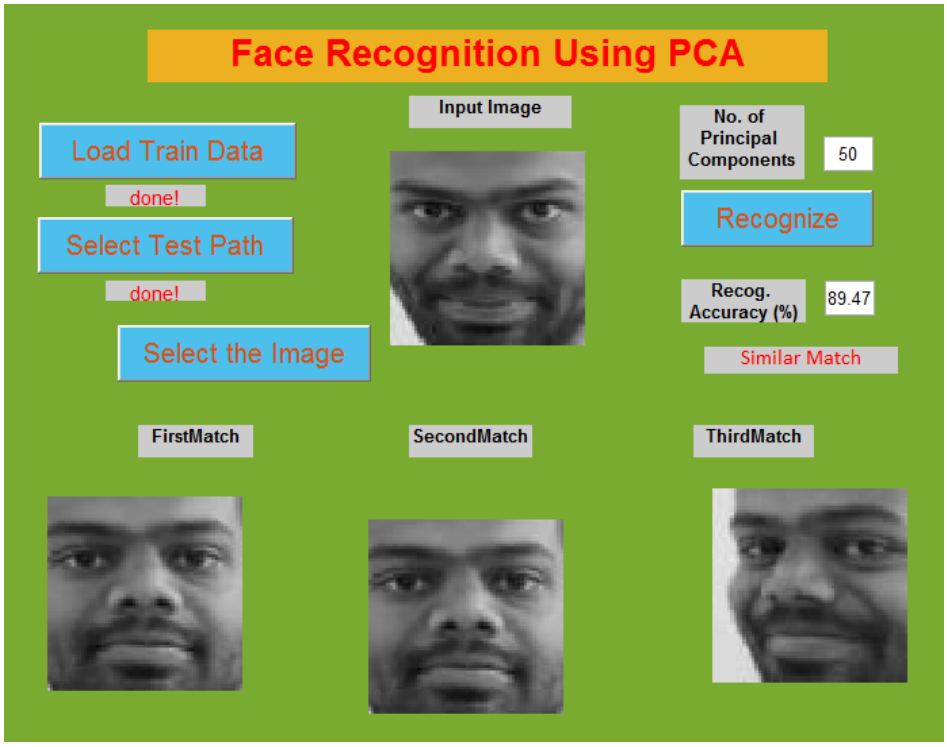
Otherwise, increment an error count € by 1.

Step 7: We find the Accuracy

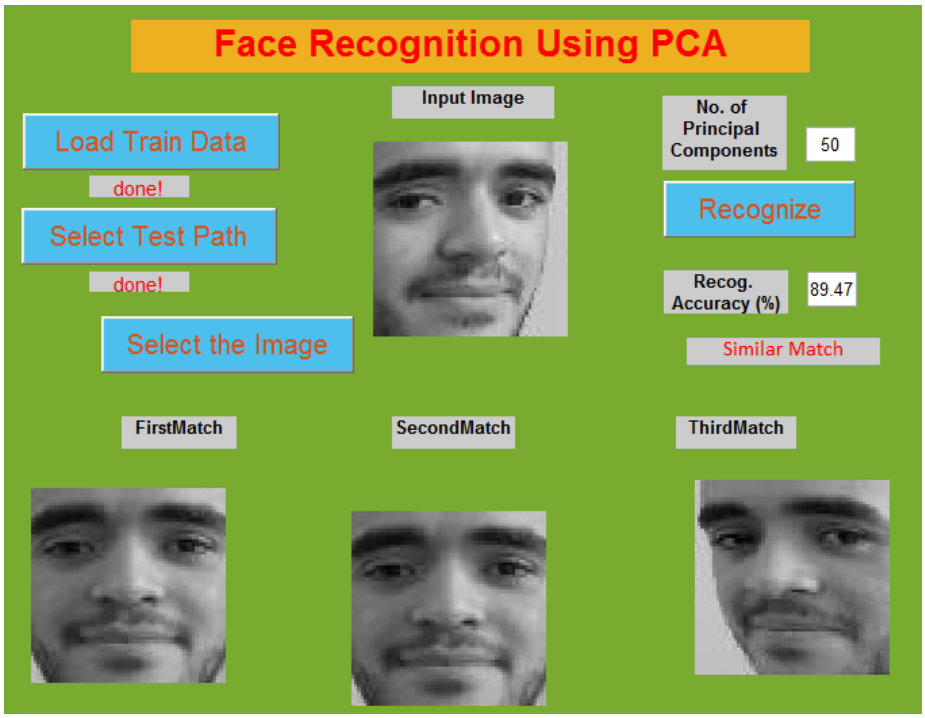
$$accuracy = \left(1 - \frac{\epsilon}{total\ number\ of\ test\ images}\right) * 100$$

3. Graphic User Interface (GUI)

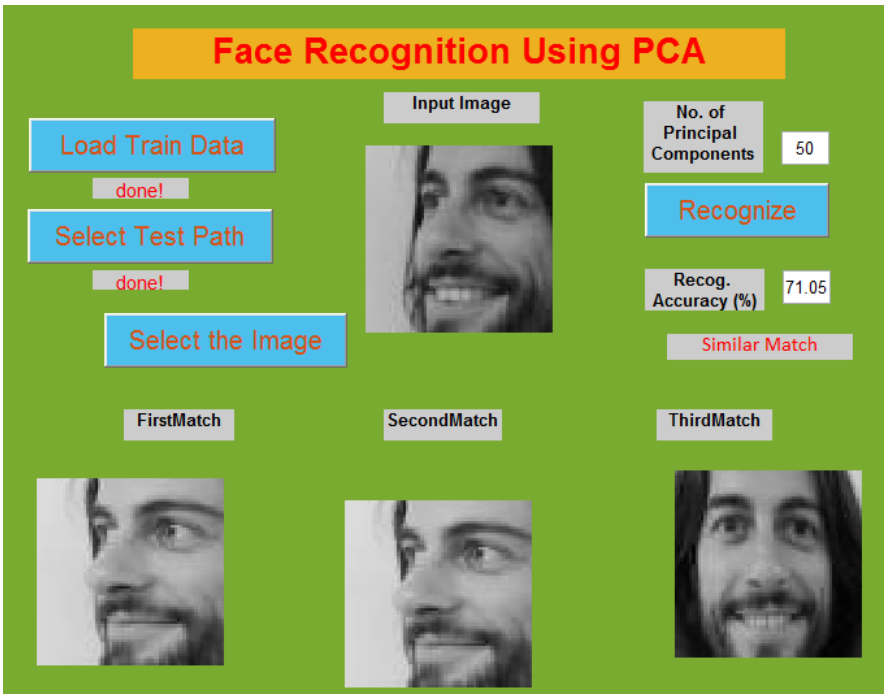
Once we are ready with the complete code, we design a GUI in Matlab to test our results



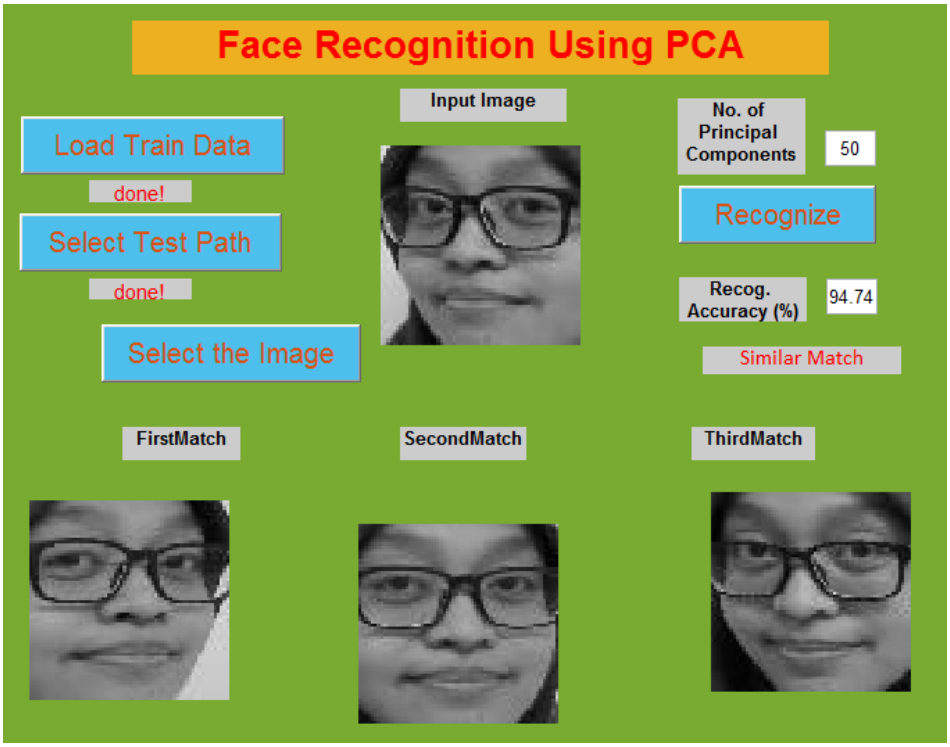
Output of GUI::Test-1



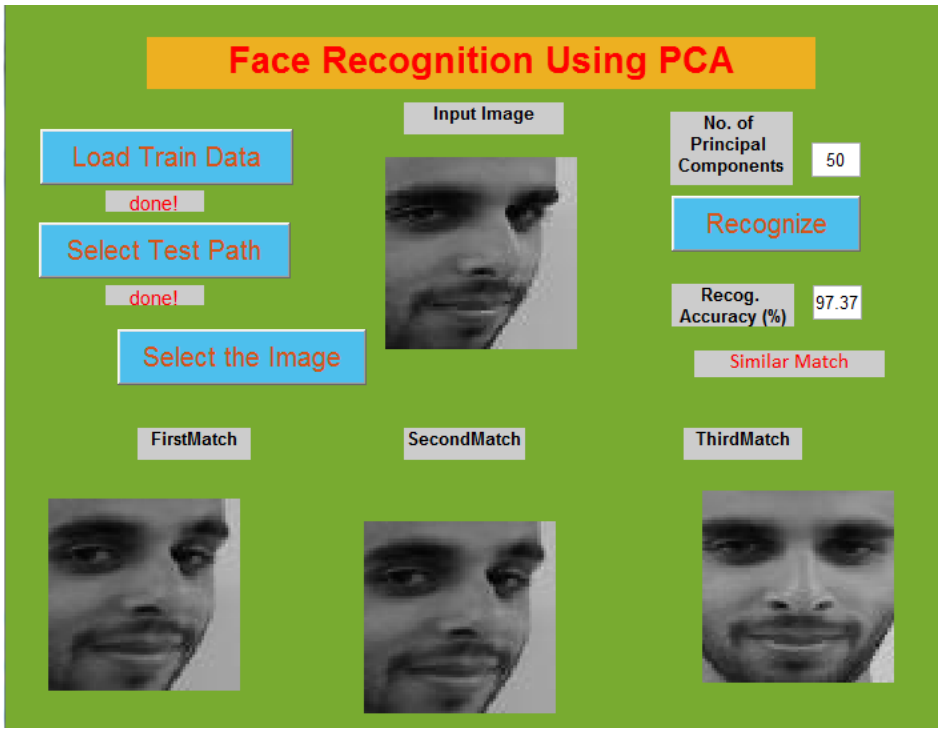
Output of GUI::Test-2



Output of GUI::Test-3



Output of GUI::Test-4



Output of GUI::Test-5

4. Results and conclusion

Normalization plays key role in face recognition. In this normalization it will take all the features average to proceed further. But here what we have observed is when taking the F'(F_bar) as initial features of the first image we are getting new normalized images. When trying to match in order obtain the accuracy we are getting different accuracy when we are changing initial features.

Principal component Analysis can be used for many purposes we found some of them are to decrease the computational complexity and measure of the covariance between the images. PCA reduces the complexity of computation when there is large number of database of images. These principal components of the Eigen vector of this covariance matrix when concatenated and converted gives the Eigen faces. These Eigen faces are the ghostly faces of the trained set of faces form a face space. This distance gives the location of the image in the Eigen space, which is taken as the output matched image.

What We Learned

1. If the Initial Features (F_bar) are changed accuracy changes

2. Affine Transformations are used to map images

3. Image can be represented in a vector form using reshape in MATLAB

5. If the intensity of the image pixels correlation increases it reduces its dimensions reduce by projecting along the vector using PCA

6. Data of the give images are easy to do face reorganization but some false data causes troubles

7. If we take covariance matrix as A*A' it leads to higher dimensions takes more time for computing

8. Learned many inbuilt matlab commands and their uses.

9. If there is any wrong data in the given images responsible for the accuracy.

10. Shows an example how applied math's is used in various fields.