Face Recognition using PCA

Applied Mathematics Project

Supervised by

Associate Prof. Dr. Desire SIDIBE

UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Deogratias LUKAMBA NSADISA
Macaulay SADIQ OLAYIWOLA
Master in Computer Vision & Robotics
University of Burgundy
France

January 2019

Contents

List of Figures

Resume

These days, the application for face recognition systems has grown exponentially particularly for Security and Authentication purposes. Numerous systems were utilized for this reason, one of them is face recognition. Face recognition is an effective means of identifying a person.

The advantage of this technique is that, it facilitates us to detect changes in the face pattern of an individual to an appreciable extent. Hence face recognition can be used as a key factor in crime detection mainly to identify fugitive. There are several techniques to face recognition of which Principal Component Analysis (PCA) is used. This concept of Applied Math is essential in this time to make the technology robust and efficient. Just as Google's page ranking using Eigenvectors and eigenvalues, Face recognition is using PCA and SVD. In this project, we use the Principal Components Analysis (PCA) and Singular value decomposition (SVD) for face recognition. The system consists of a database of a set of facial patterns for each individual. The characteristic features called eigenfaces are extracted from the stored images using which the system is trained for subsequent recognition of new images.

Chapter 1

Introduction

This project mainly consist of implementation of the face recognition FR using Principal Component Analysis (PCA). PCA is a statistical approach used for reducing the number of variables in face recognition. However, the idea is to represent all the training set images as a linear combination of the eigenfaces. These eigenfaces are obtained from covariance matrix of a training image set. The eigenfaces are found out after selecting a set of most relevant Eigenfaces. Recognition is performed by projecting a test image onto the subspace spanned by the eigenfaces and then classification is done by measuring the Euclidean distance [1]. A number of experiments were done to evaluate the performance of the face recognition approach. In this project, we are going to use a training database of students of Master in computer Vision and Medical imaging and Applications from University of Burgundy.

The principal objective of this project on applied mathematics course is to familiarize with SVD and PCA through creation of face detection system

1.1 Face recognition

The Eigenface technique is one of the easiest and most effective PCA techniques utilized in face recognition. In this technique,the faces are transformed into a small set of essential characteristics, eigenfaces. Eigenface is the main fundamental of the initial set of training data. Recognition is done by projecting a new image in the eigenface subspace, after which the person is classified by comparing its position in eigenface space with the position of known individuals [2]. This technique has one advantage over other face recognition systems since it is rapid , robust to sensativity to small change on the face and simple. Its main problem is limitation of the files that can be used to recognize the face is considered as the main problem. In addition, the views of human faces of images are set to be vertically frontal.

1.2 Principal Component analysis

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.

1.3 Singular Value Decomposition

The Singular Value Decomposition (SVD) is a widely used technique to decompose a matrix into several component matrices, exposing many of the useful and interesting properties of the original matrix. The decomposition of a matrix is often called Factorization. In SVD, we decompose a real matrix $A \in R$ mxn $(m < n)$ can be written as the product of three matrices as follows:

$$A = USV^T$$

The two matrices $U \in R$ mxn and $V \in R$ nn are orthonormal. The diagonal elements of $S \in R$ nxn are the singular values. The column vectors of U and V are the corresponding singular vectors of each singular value.

Face recognition is one example where principal component analysis has been extensively used. The choice of the Singular Value Decomposition (SVD) is very important due to its vast use in the Principal Component Analysis and because it largely defines its performance.

Chapter 2

Methodology

We subdivide the project into two parts:

1. Normalization

2. Recognition

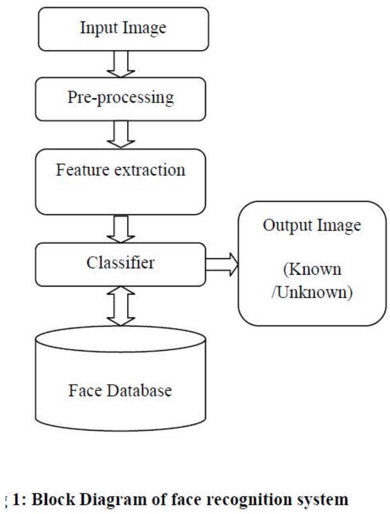

Figure 2.1: Block Diagram of face recognition systems

2.1 Block Diagram of face recognition systems

We need to perform the normalization steps to ensure that all face images are approximately aligned for the scale, orientation, and location variation adjustment to all the images with some predefined features and the feature of images. Basically, we set all the images to a 64x64 window taking some of the important facial features using the following facial features:

1. left eye center
2. right eye center
3. tip of nose
4. left mouth corner
5. right mouth corner.

With the goal of finding a vector $\bar{F}$ that corresponds to the average location to map each features $\bar{F_i}$. Therefore the algorithm for Normalization is presented as follows:

1. Initialized features coordinates $\bar{F_i}$.
2. For every $\bar{F_i}$. Then compute the affine transformation given by the equation:

   $$\bar{F_i^p} = A * \bar{F_i} + b$$

   . such that A and b are unknowns.
3. For every iteration , update $\bar{F}$ by using SVD
4. Compute the average of all the aligned feature locations for each face image and update $\bar{F}$ with this value.
5. Compare $\bar{F_t}$ and $\bar{F_{t-1}}$. If the differential is less than a threshold, then stop meaning that the final converged gives the affine transformation matrix A and vector b; otherwise, repeat step 2.

2.2 Recognition

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

$$D = \begin{bmatrix} I_1(1,1) & I_1(1,2) & \cdots I_1(1,N) & \cdots I_1(M,1) & I_1(M,2) & \cdots I_1(M,N) \\ I_2(1,1) & I_2(1,2) & \cdots I_2(1,N) & \ldots I_2(M,1) & I_2(M,2) & \cdots I_2(M,N) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ I_p(1,1) & I_p(1,2) & \cdots I_p(1,N) & \cdots I_p(M,1) & I_p(M,2) & \cdots I_p(M,N) \end{bmatrix}$$

So, we calculate the covariance of matrix D by:

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

After, we find the eigenvalues and eigenvectors of D and save the eigenvectors corresponding to the largest eigenvalues. Notice that Each eigenvector has the same dimensionality (number of components) as the original images, and thus can itself be seen as an image. The eigenvectors of this covariance matrix are called Eigenfaces. Here we have few samples and many variables. This means that the number of nonzero eigenvalues of the covariance matrix is limited to training images. Therefore, instead of computing the eigenvectors of the very large covariance matrix, we can just compute the eigenvectors of $\Sigma' = \frac{1}{p-1} D \cdot D^T$ which is a p x p matrix.

In PCA, we basically try to find eigenvalues and eigenvectors of the covariance matrix, D. We showed that D $= \frac{AA^T}{(n-1)}$, and thus finding the eigenvalues and eigenvectors of D is the same as finding the eigenvalues and eigenvectors of AA^T. In SVD, decompose a matrix A as follows:

$$A = USV^T$$

This can be proved that: (see the lecture note [3])

$$AA^T = US^2U^T$$

Where the columns of U contain the eigenvectors of AA^T and the corresponding eigenvalues are the squares of the singular values in S. Thus SVD gives us the eigenvectors and eigenvalues that we need for PCA.

2.3 Recognizing a face

The face is expressed in the face space by its Eigenface coefficients which are considered as the weights. By taking the eigenface ,We can manipulate a large input vector now, facial image.

In the recognition phase, we give an image of a known person. The image is labeled with name of the person. First, image is represented by a vector X_j as in the training phase and this vector is projected into the PCA space to get a corresponding feature vector.

$$\Phi_j = X_j \cdot \Phi$$

To identify, we have to compute a similarity, for example Euclidean distance, between corresponding feature vector and all feature vectors in the training set. The image whose feature vector is most similar to the corresponding feature vector is the output of the face recognition system. If the Euclidean distance between them is equal, then we have correctly identified the person. Otherwise, we have to misclassify the person.

the algorithm of the recognize face is as follows:

1. Set the number of Principal components
2. Build a training data matrix D from training images
3. Assign name as the label for each training image and form a labels matrix Lt
4. Remove the mean and compute covariance matrix (D). Then compute the PCA of D, which gives PCA transformation matrix.
5. Feature vectors of all training images using transformation matrix and store them in a matrix Ft
6. Read test images, and for each test image : Calculate the feature vector q using the PCA transformation matrix.

   Compute the distance between q and all training feature vectors, which are stored in Ft.

Find the best match I_i (minimum distance between feature vectors).

Check if the label of I_i is the same as the label of I_j. Otherwise, increment an error count ϵ by 1.

7. We can now find the Accuracy as :

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

Chapter 3

Implementation

We created a student database and which contains their photographs and here we are working with 81 training images. We have taken the mean of all the 81 training faces. We have subtracted these values from each face. We concatenate all the subtracted faces in a single matrix D.

$$D = \begin{bmatrix} \mathrm{I}_1(1,1) & \mathrm{I}_1(1,2) & \mathrm{I}_1(1,3) & \cdots & \cdots & \mathrm{I}_1(1,64) \\ \mathrm{I}_2(1,1) & \mathrm{I}_2(1,2) & \mathrm{I}_2(1,3) & \cdots & \cdots & \mathrm{I}_2(1,64) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathrm{I}_{81}(1,1) & \mathrm{I}_{81}(1,2) & \mathrm{I}_{81}(1,3) & \cdots & \cdots & \mathrm{I}_{81}(64,64) \end{bmatrix} \$$$

We just compute the eigenvectors of $\Sigma = \frac{1}{p-1} DD^T$. This reduces the dimension to 81x81 in this case. The Eigenvector computed from this matrix will be of size 81x81. Next, we find the Eigenfaces. Each face in the training set can be represented as a linear combination of these eigenvectors.

$$\Phi_i = X_i \cdot \Phi$$

Where Φ is the principal components and X_i is the training images.

Now, we normalize the image and project this normalized image into the PCA space to get a corresponding feature vector.

Now, we calculate the Euclidean Distance between Φ_j and $\Phi_i's$ all feature vectors in the training set. The image whose, feature vector is similar and, distance is minimum to Φ_j will be the output of the face recognition system.

Chapter 4

Results

4.1 Normalization

The Accuracy of Normalization is proportional to the Input images. However, Normalization is fundamental in Face Recognition. Here, it will take all the features average to proceed further. The main problem faced during the normalization process is that some images were not captured from the same distance meaning distance camera and face due to this reason when performing normalization for the entire set of images, some of them were removed because of various reasons: distance face and camera was far this means the coordinate points were concentrated in the center; distance face and camera get closer this implies that coordinates points are stretched . In this case , $||\bar{F_t} - \bar{F_{t-1}}|| < \epsilon$ $(\epsilon = Treshold)$ will converge slowly.

In addition , other problem might affect the normalization is that the coordinate system and format of the image have not been respected and correctly set . So, Normalized images is showing below :

Figure 4.1: Normalized Image 1

Figure 4.2: Normalized Image 2

Figure 4.3: Normalized Image 3

4.2 Face Recognition

The main factor affecting Face Recognition is Normalization. The following are the output of Face Recognition by using PCA Algorithm.

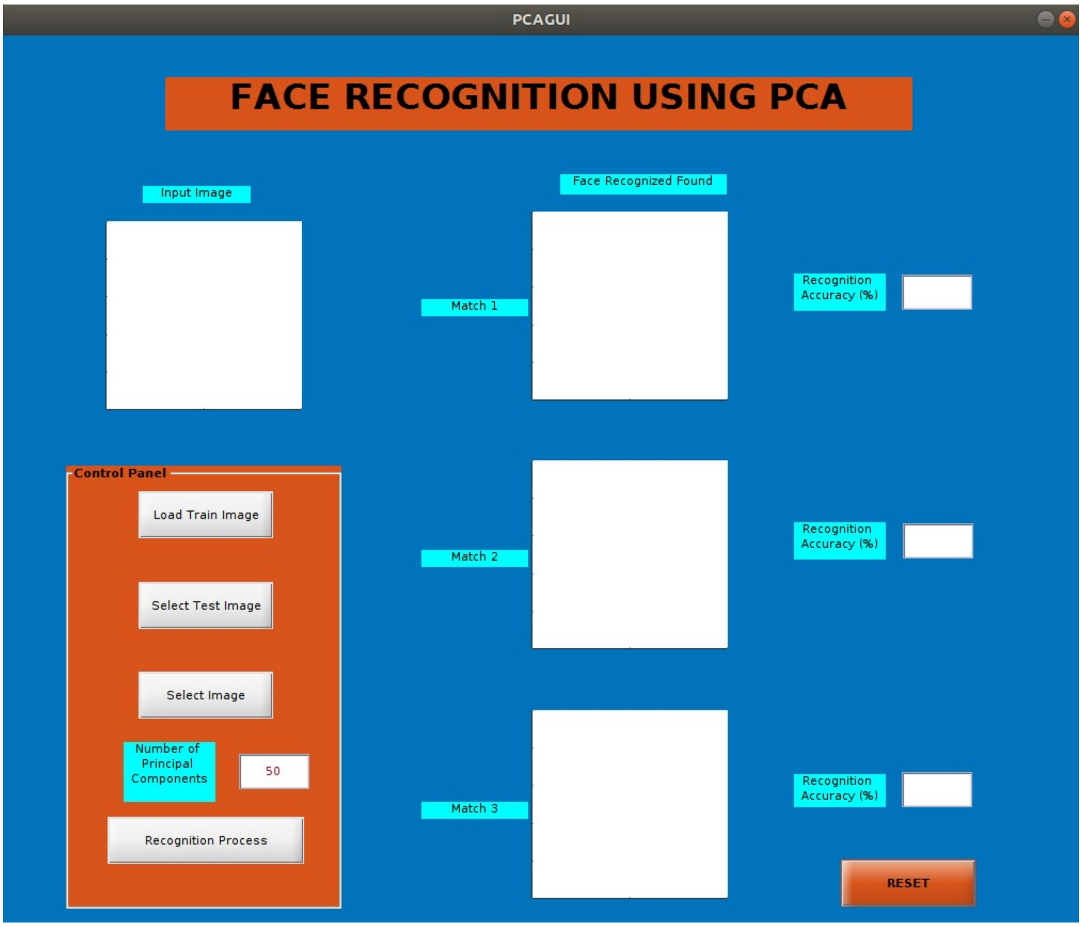

Figure 4.4: GUI Window

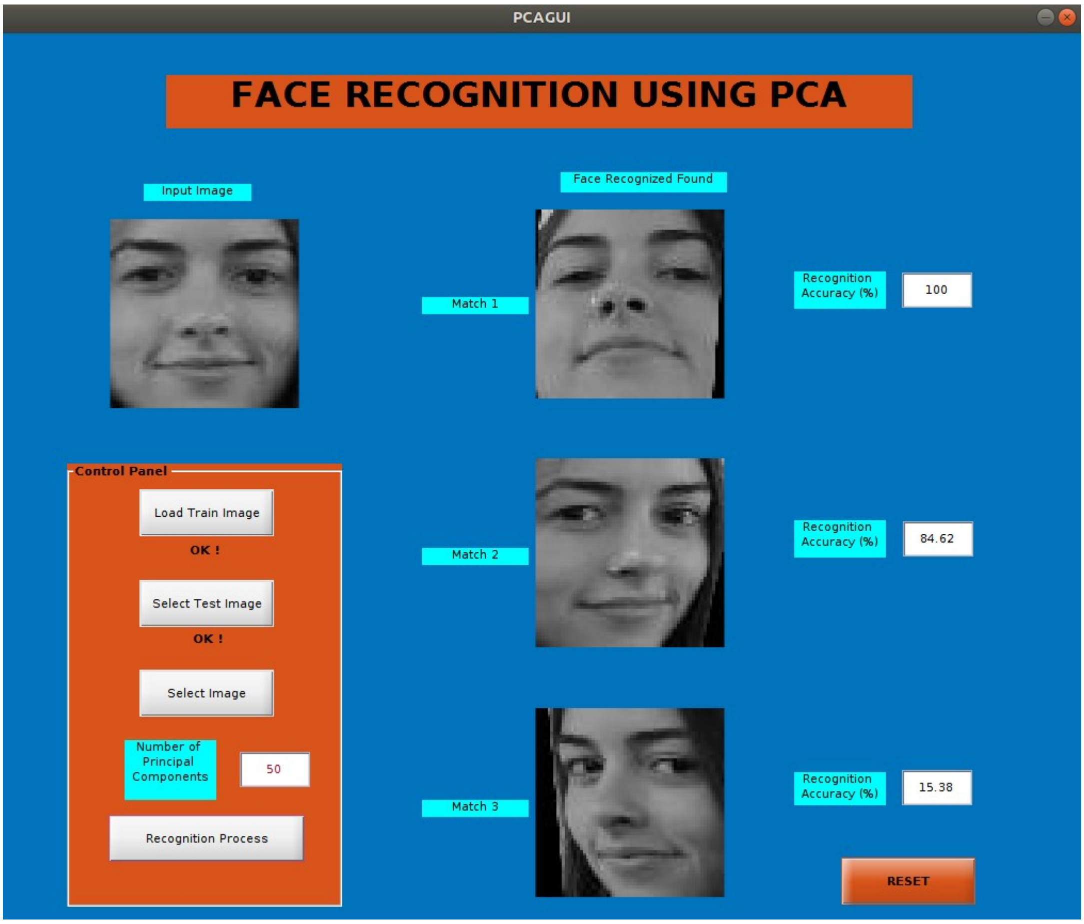

Figure 4.5: Output with 3-High Accuracy 100% ,

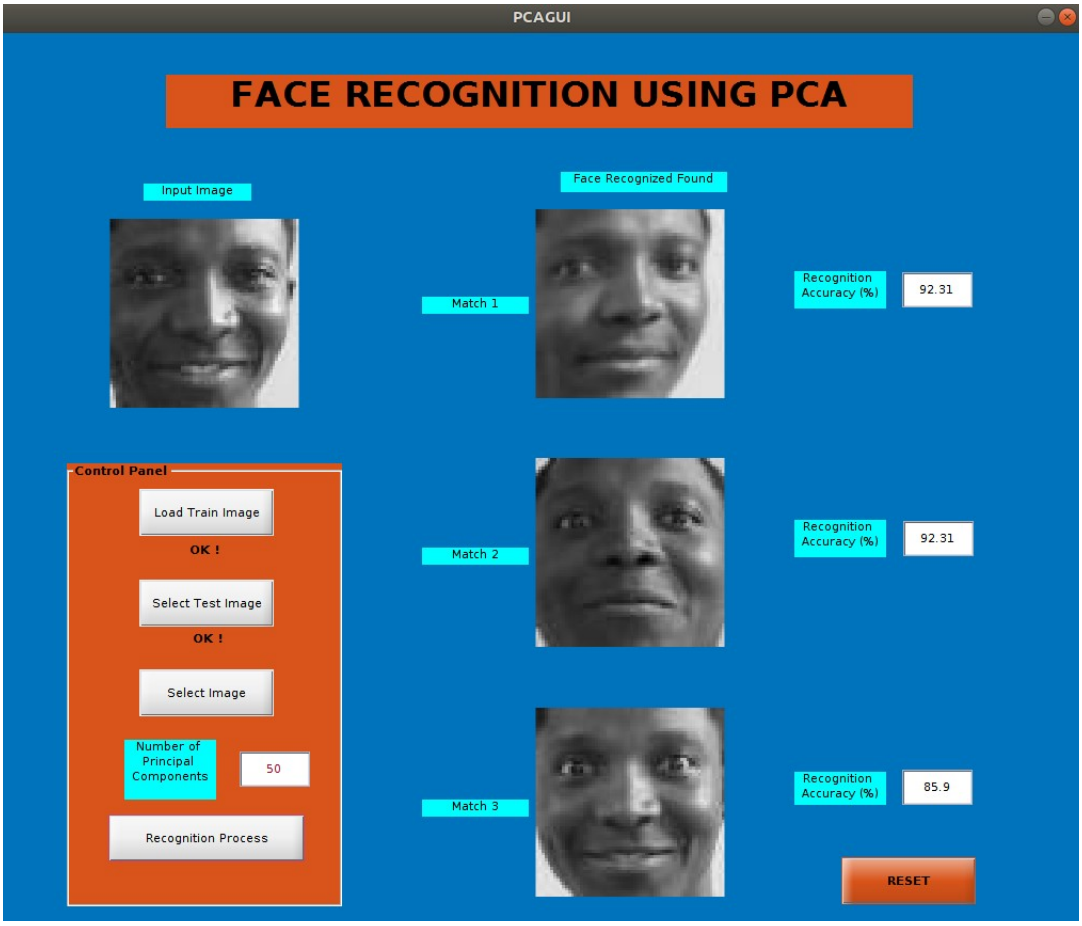

Figure 4.6: Output with Accuracy 33.3

[figure5]

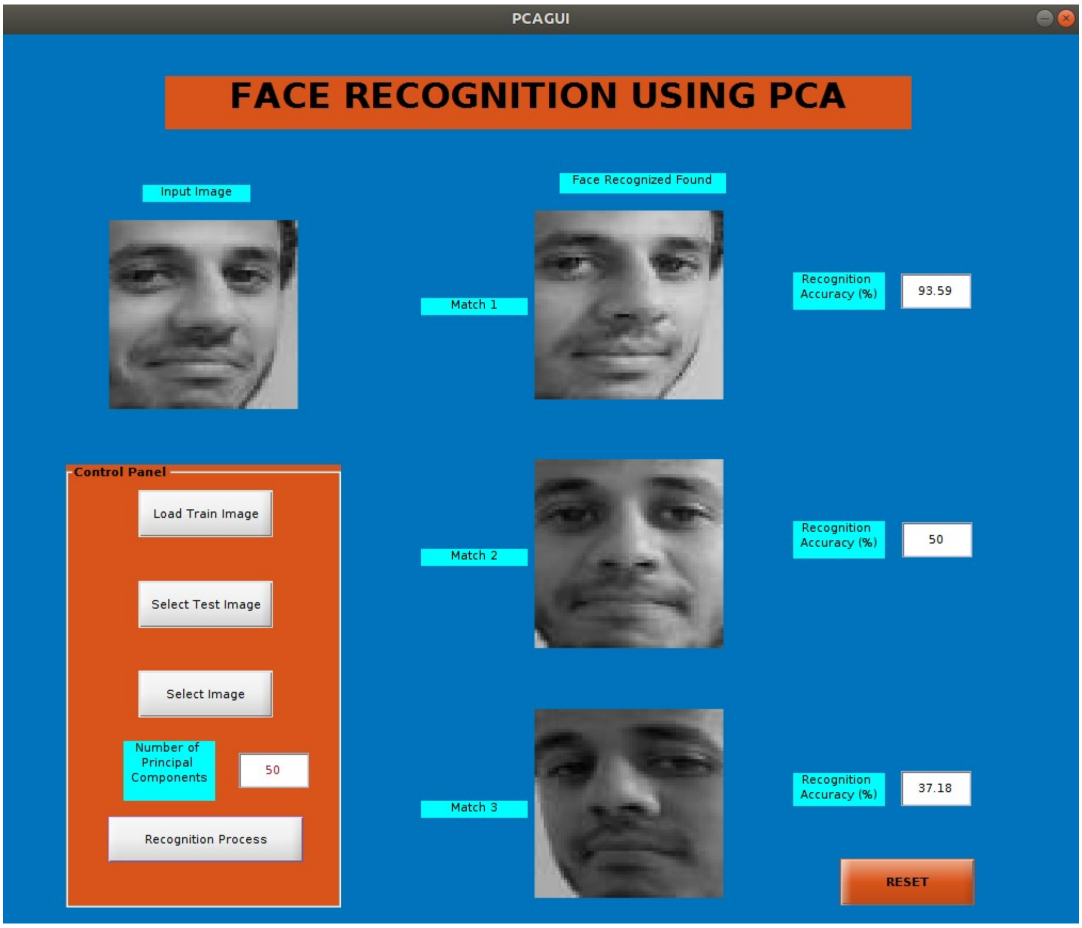

Figure 4.7: Output with Accuracy 33.3

[figure5]

Chapter 5

Conclusion

The project aimed to get familiar with SVD and PCA and this was accomplished successfully. Also, We have to notice that a good performance of a PCA requires that images should be normalized accurately. Any false measurement can seriously affect the results. Due to a lack of correlation in the training data, Normalization performance is inaccurate. Therefore the Higher is our Normalization accuracy, the Accuracy of our Face Recognition is.
As a future work, we plan to investigate how to get same accuracy for different predefined features.

5.1 What we learned

1. To Get familiar with Mathematical concept of PCA and SVD
2. to apply Affine Transformation which is a Geometric Transformations using in Face Recognition purpose.
3. To Learn and to use various in-built functions like pinv for computing the sudoinverse. Additional in-built functions such as : affine2d, imwarp . MATLAB-GUI , etc.
4. To get familiar with Latex for writing the project.

Bibliography

[1] A. A. S. Liton Chandra Paul, "Face recognition using principal component analysis method," *International Journal of Advanced Research in Computer Engineering Technology (IJARCET)*, vol. 1, p. 9, 2012.

[2] A. P. M. Turk, "Face recognition using eigenfaces," in *Conference on Computer Vision and Pattern Recognition3 6 , Maui, HI , USA, pp. 586 591.*, June 1991.

[3] M. Kaur and R. Vashisht, "Comparative Study of Facial Expression Recognition Techniques," in *International Journal of Computer Applications*, January 2011.