UNIVERSITY OF BURGUNDY

APPLIED MATHEMATICS PROJECT

FACE RECOGNITION USING PCA

Submitted by

Gopikrishna Erabati

Nayee Muddin Khan Dousai

Supervisor:

Dr. Désiré Sidibé

CONTENTS

List Of Figures

1. Introduction

Principal Component Analysis (PCA) is a way to identify and find the similar patterns from a given set of data. From the past years, 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.

The main objective of our project is to recognize faces from the collected set of face data. We have collected the five pictures of every student and the idea here is to extract few features from the faces with the goal of reducing the number of variables used to represent the faces. In the next step, we divide the faces 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. The locations of five facial features are extracted from every image: left eye, right eye, tip of nose, left mouth corner and right mouth corner as presented in below figure 1.

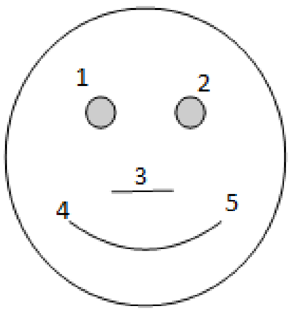


Figure 1 Five facial features used

But, the problem here is, an image has high dimensionality space (each image is a point in a space of dimension d = MN, M and N being image size) as each pixel is considered as a variable of an image. So, we can reduce the dimensionality by using PCA to simplify recognition problem, which can be considered as the core concept.

2. Methodology

Our face recognition system consists of three modules. The first module normalizes the input images. The second module performs the PCA decomposition of training faces, which produces Eigen vectors and Eigen values. The third module identifies the face. The three modules are briefly described below.

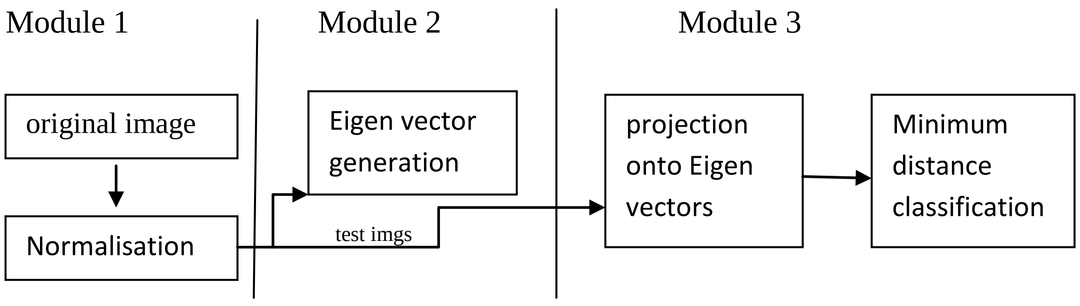


Figure 2 Block diagram of PCA-based Face Recognition system

2.1 Normalization

Once all the locations are extracted for all 145 images, we need to normalize the images prior to recognition. This step places the face in the standard geometric position by affine transformations, of the predefined feature locations. The goal of this step is to remove variations in size ,orientation and location of faces. From the provided facial features, we have the predefined values for 64×64 image as left eye (13, 20), right eye (50,20), Tip of nose (34,34), mouth left corner (16,50), mouth right corner (48,50). Affine transformation is represented by six parameters. We must find the affine transform for the first image now with respect to the predefined facial values.

$$\mathrm{x} = a_{11}x^1 + a_{12}y^1 + b_1$$

$$y = a_{21}x^1 + a_{22}y^1 + b_2$$

From the above equation x and y are the values of feature locations of predefined image, x^1 and y^1 are the values of feature locations of first image. The above equations can be written for all the five features of the first image into a matrix form, represented as:

$$PC_1 = P_x$$

$$PC_2 = P_y$$

$$\text{where } C_1 = [a_{11} \quad a_{12} \quad b_1]^T \text{ and } C_2 = [a_{21} \quad a_{22} \quad b_2]^T$$

We can find C_1 and C_2 by using SVD. C_1 and C_2 values are substituted in the above equation to find the new transformation equation which gives the new values for x and y. By using this x and y values we will find C_1 and C_2 for all the 145 images and apply on x^i and y^i of all images to get new x and y for every image.

Once we have all the data of x and y values, we will calculate the average of all the values of all images and compare with the x and y values of the first image. If the error between this images is less than the threshold we will stop or else, we will continue by finding the new affine transformations again.

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. After the normalization of 145 images we will proceed to face recognition. All the required code for normalization is attached as normalization.m file.

2.2 PCA Decomposition of training set

For face recognition, all the 145 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 93 train images and the rest are named as test images. All the train images are converted to form a row vector of dimension 4090 (64 x 64) and all train images are kept in a data matrix labeled D. Data matrix is represented as:

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

where p is number of images and d is number of variables

Once we have the data matrix of train images we will find the mean vector of all images which gives 1×d matrix and the mean matrix is represented as $p \times d$ by repeating the mean vector. Before computing the covariance, we will subtract the mean matrix from the data matrix. Now we compute the covariance of the matrix D as,

$$\Sigma = {}^{(D^T D)}\!\big/_{(p-1)}$$

But here, we have many variables and few images. If we compute $D^T D$ or DD^T the matrix dimension we get is as follows,

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

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

As in our case the number of variables is more than number of images p. The number of nonzero eigen values of covariance matrix is limited to p. From the above equations, it shows $D^T D$ is a very large matrix, so, we can just compute DD^T, which is relatively a small matrix to compute. So, we will consider this matrix and find the eigen vectors.

Actually, we require eigen vectors of $D^T D$.

Let X be eigen vector of DD^T then (DD^T) X = λX

Multiply both sides with D^T to get

(DD^T) D^TX = λ(D^TX)

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 we will find the PCA transformation matrix. This projection of train images into PCA space gives p×p matrix. Finding the feature vector of train images

$D^T X$ = (d×p) (p×p) = d×p = Φ

Feature vector = DΦ = (p×d) (d×p) = p×p

From the above equation, the columns of p can be varied depending on principal components required. The feature vectors of all images are labeled with the names of images.

For, the number of principal components required, we have done analysis of eigen values (vs) number of eigen values as given below.

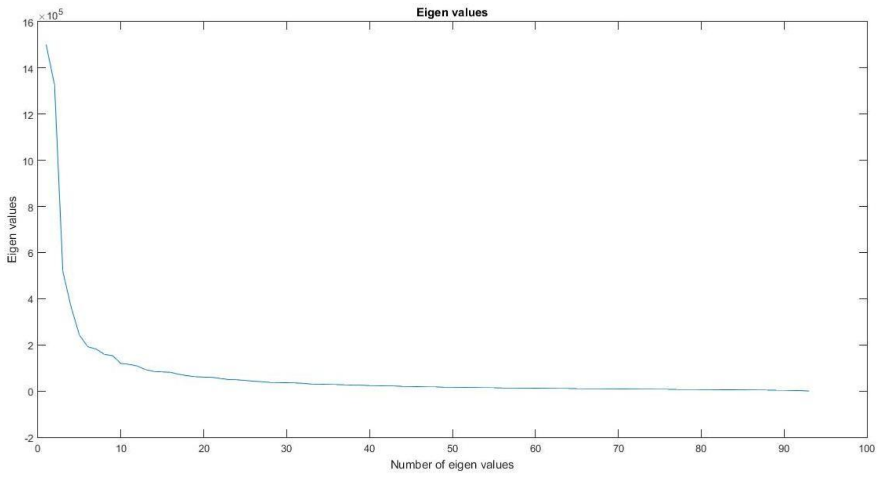


Figure 3 Eigen value vs number of eigen values

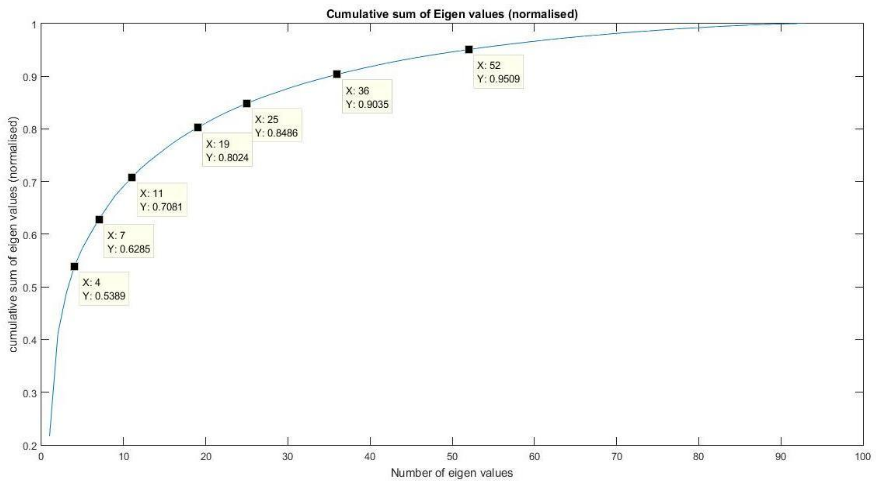


Figure 4 Cumulative sum of eigen values vs no. of eigen values

Figure 3 shows the plot between eigen values and number of eigen values of covariance matrix. Figure 4 shows plot between the cumulative sum of eigen values (normalized) and number of eigen values. From the above plot we can clearly see that for 52nd eigen values the normalized cumulative sum is greater than 95%. So we can take 52 principal components to get almost accurate results.

The code for this module is given in createTrainDataBase.m and featurevector.m.

2.3 Identification of test faces

We have find the feature vector of all the train images and we should check with the test images now. The feature vector for test images (I) is given as

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

We will take every single test image and compute the Euclidean distance between test image with the feature matrix of all trained images. The distances are arranged in ascending order and if the label of test image matches with the label of train image with minimum distance then the test image is detected otherwise we increment the error. We perform this operation on all test images to detect the accuracy of algorithm. The accuracy is given by, Accuracy = (1 – error / no. of test images) × 100

Accuracy = (1 – error / 52) × 100 (52 test images in our case)

We also made provision to check the correct face among the first *k* faces returned by our algorithm. We have found accuracy for each value of *k* as given below.

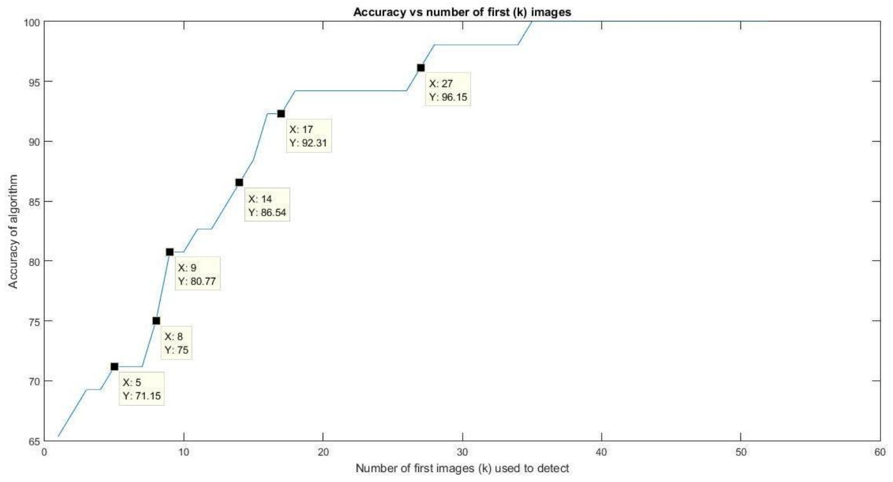


Figure 5 Accuracy of algorithm vs first *k* faces returned by algorithm

We can see clearly from the figure 5, for only one face returned by algorithm we got an accuracy of 65.38% .

The code for computing accuracy can be found in recognize_accuracy.m. Tho code for recognition module is given in recognize.m.

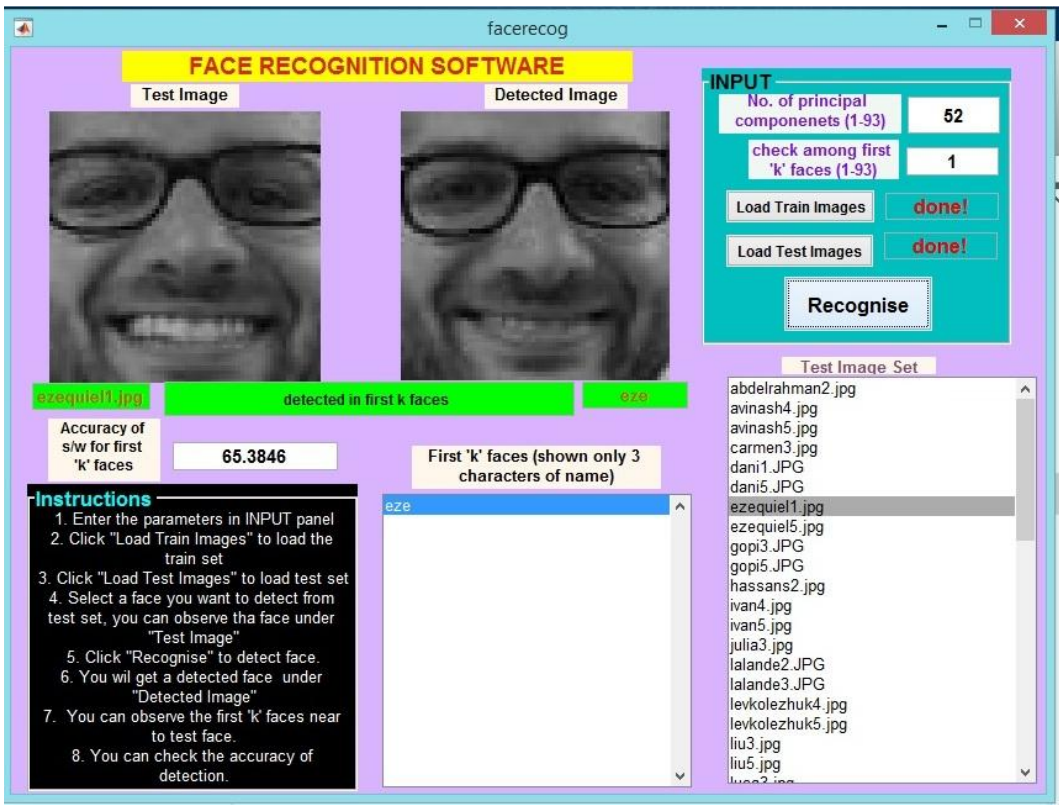
3. Graphic User Interface (GUI)

We have designed a GUI for our face recognition algorithm and used the modules which we defined above, in the GUI module. The code for GUI module can be found in facerecog.m. The GUI also shows the first *k* faces returned by the algorithm for a specific test image.

The usage of GUI and other codes are given in README.txt file in the zipper folder.

4. Results and discussions

The GUI model of our algorithm looks like below.



No. of principal componenets (1-93): 52
check among first 'k' faces (1-93): 1
Load Train Images: done!
Load Test Images: done!

ezequiel1.jpg — detected in first k faces — eze

Accuracy of s/w for first 'k' faces: 65.3846

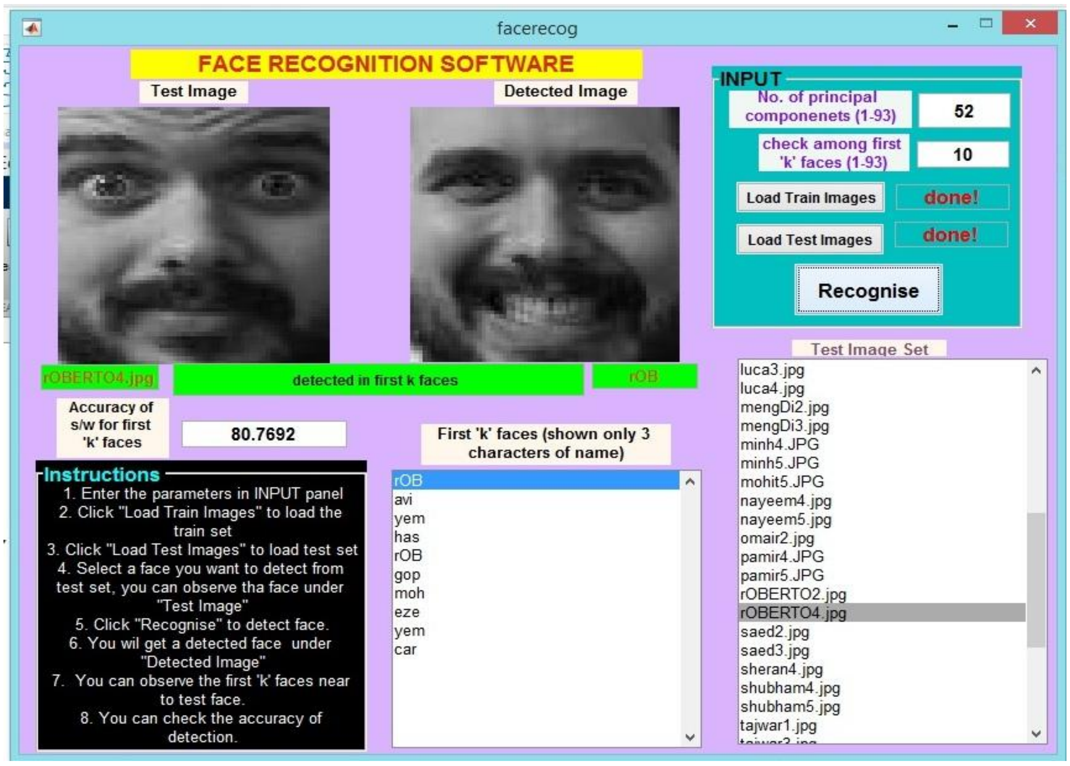
First 'k' faces (shown only 3 characters of name)

eze

Instructions
1. Enter the parameters in INPUT panel
2. Click "Load Train Images" to load the train set
3. Click "Load Test Images" to load test set
4. Select a face you want to detect from test set, you can observe tha face under "Test Image"
5. Click "Recognise" to detect face.
6. You wil get a detected face under "Detected Image"
7. You can observe the first 'k' faces near to test face.
8. You can check the accuracy of detection.

Test Image Set:
abdelrahman2.jpg
avinash4.jpg
avinash5.jpg
carmen3.jpg
dani1.JPG
dani5.JPG
ezequiel1.jpg
ezequiel5.jpg
gopi3.JPG
gopi5.JPG
hassans2.jpg
ivan4.jpg
ivan5.jpg
julia3.jpg
lalande2.JPG
lalande3.JPG
levkolezhuk4.jpg
levkolezhuk5.jpg
liu3.jpg
liu5.jpg

Figure 6 GUI - 1



No. of principal componenets (1-93): 52
check among first 'k' faces (1-93): 10
Load Train Images: done!
Load Test Images: done!

rOBERTO4.jpg — detected in first k faces — rOB

Accuracy of s/w for first 'k' faces: 80.7692

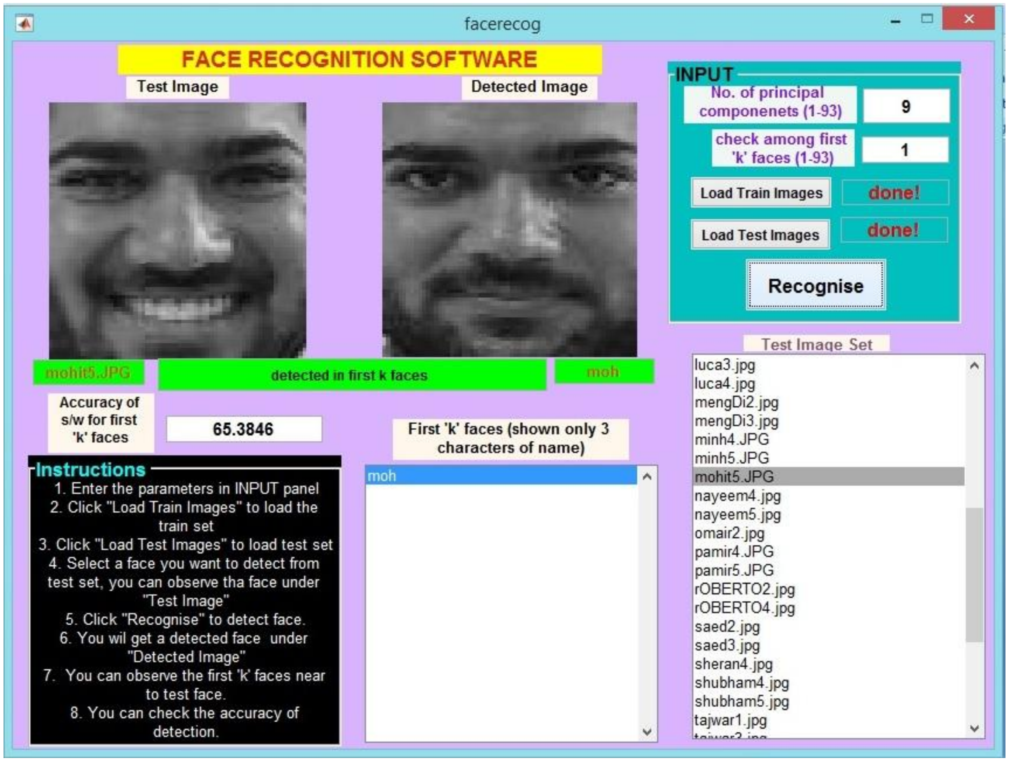
First 'k' faces (shown only 3 characters of name)

rOB
avi
yem
has
rOB
gop
moh
eze
yem
car

Instructions
1. Enter the parameters in INPUT panel
2. Click "Load Train Images" to load the train set
3. Click "Load Test Images" to load test set
4. Select a face you want to detect from test set, you can observe tha face under "Test Image"
5. Click "Recognise" to detect face.
6. You wil get a detected face under "Detected Image"
7. You can observe the first 'k' faces near to test face.
8. You can check the accuracy of detection.

Test Image Set:
luca3.jpg
luca4.jpg
mengDi2.jpg
mengDi3.jpg
minh4.JPG
minh5.JPG
mohit5.JPG
nayeem4.jpg
nayeem5.jpg
omair2.jpg
pamir4.JPG
pamir5.JPG
rOBERTO2.jpg
rOBERTO4.jpg
saed2.jpg
saed3.jpg
sheran4.jpg
shubham4.jpg
shubham5.jpg
tajwar1.jpg

Figure 7 GUI - 2


facerecog

FACE RECOGNITION SOFTWARE

Test Image | Detected Image

mohit5.JPG — detected in first k faces — moh

Accuracy of s/w for first 'k' faces: 65.3846

First 'k' faces (shown only 3 characters of name): moh

INPUT
No. of principal componenets (1-93): 9
check among first 'k' faces (1-93): 1
Load Train Images: done!
Load Test Images: done!
Recognise

Instructions
1. Enter the parameters in INPUT panel
2. Click "Load Train Images" to load the train set
3. Click "Load Test Images" to load test set
4. Select a face you want to detect from test set, you can observe tha face under "Test Image"
5. Click "Recognise" to detect face.
6. You wil get a detected face under "Detected Image"
7. You can observe the first 'k' faces near to test face.
8. You can check the accuracy of detection.

Test Image Set
luca3.jpg
luca4.jpg
mengDi2.jpg
mengDi3.jpg
minh4.JPG
minh5.JPG
mohit5.JPG
nayeem4.jpg
nayeem5.jpg
omair2.jpg
pamir4.JPG
pamir5.JPG
rOBERTO2.jpg
rOBERTO4.jpg
saed2.jpg
saed3.jpg
sheran4.jpg
shubham4.jpg
shubham5.jpg
tajwar1.jpg

Figure 8 GUI - 3

We observed that for 9 principal components we almost have 70% of eigen values, so for 9 principal components our algorithm mostly detects all test images for even one (*k*=1) returned image. And for 18 returned images, our algorithm produces 95% accuracy and for 35 returned images our algorithm gives 100% accuracy for test image detection.

5. Conclusion

We have presented a face recognition system based on PCA. We had systematically varied the components and infer the impact of these variations on performance. The quality of images to be processed is the driving factor in determining the performance of PCA based face recognition system. The PCA plays an immense role in this system for dimensionality reduction and pseudo inverse using SVD played a significant role in normalization problem.