Applied Mathematics

Face Recognition using PCA

Group Members : Shubham Wagh, Pamir Ghimire

Submitted to : Prof. Désiré Sidibé

Dated : 06/01/2017

Centre Universitaire Condorcet - Master Computer Vision

Master in Computer Vision

Vibot

Contents

1 Methodology **2**
1.1 Normalization 2
1.2 Recognition using PCA 3
1.2.1 Training data set 3
1.2.2 Test data set 5

2 Experiments **6**
2.1 Face Recognition 6

3 Conclusion **9**

4 Appendix **10**

List of Figures

1 320 x 320 image to 64 x 64 image 3
2 EigenFaces of first 30 Principal Components 4
3 Correct Face Recognition 6
4 Wrong Face Recognition 7
5 Accuracy plot for different values of k 8
6 For $k = 3$, best matches 8
7 Face Recognition GUI 10
8 Face Recognition GUI example 11

1 Methodology

1.1 Normalization

Once a face has been detected in an image, a normalization step is required prior to recognition to account for scale, orientation and location variations. The face images obtained from the original ones should be reshaped all in the same size with the face in the middle. This will also help for a faster computation.

The data set used is composed of size 320×320 images, with feature coordinates, an iterative scheme based on affine transformations from feature coordinates to some predefined coordinates is performed in a 64×64 blank image in order to normalize them to this size.

The predefined coordinates in a 64×64 blank imaga are included in the following matrix:

$$\overline{F} = \begin{bmatrix} 13 & 20 \\ 50 & 20 \\ 34 & 34 \\ 16 & 50 \\ 48 & 50 \end{bmatrix}$$

The feature coordinates in a 320×320 original image, for instance, this one:

$$f = \begin{bmatrix} 137 & 151 \\ 174 & 146 \\ 155 & 172 \\ 141 & 210 \\ 176 & 202 \end{bmatrix}$$

The affine transformation has therefore this shape:

$$\overline{F} = A * f + b$$

This transformation admits 10 equations and 6 unknowns i.e an over-determined system which can be solved using SVD. The equation can be reshaped in order to perform SVD on it:

$$\overline{F} = f_1 * A'$$
$$A' = f_1^+ * \overline{F}$$

where f_1 being the f coordinates with an added column of ones on the right, and A' being the A unknown matrix with the 2 b unknown coefficients added under A, so that we get a (5×3) times (3×2) multiplication resulting in the (5×2) predefined coordinates.
Computing the coefficients for all images and the corresponding normalized feature coordinates in 64×64 pixels, we are now able to change the predefined coordinates by the average of normalized feature coordinates. Updating $\overline{F}$ until it reaches a difference $|\overline{F_t} - \overline{F_{t-1}}| \leq 1pxl$

(in around 6 or 7 iterations), we get a proper transformation which allows us to normalize our entire image.
The coefficients of A and b needs to be retrieve from the matrix A' and ap- plying the inverse transform on pixel coordinates of the output image, we can find the corresponding pixel coordinates in the original one :

$$Pxl = A^{+} * \left(\begin{bmatrix} x \\ y \end{bmatrix} - b \right)$$

Once we fetch all values inside the original image, we get normalized image, for instance in Fig.1. As we have 5 images per student we send 3 of them in the training images directory

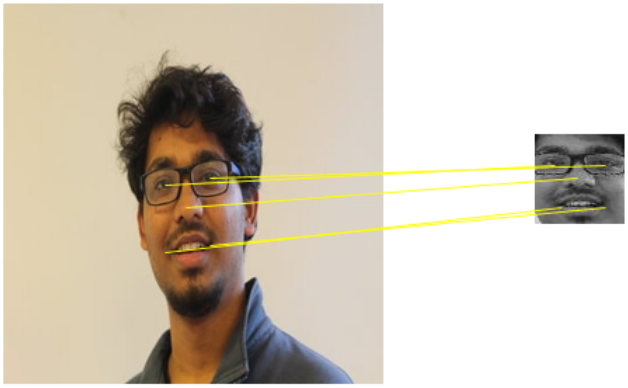


Figure 1: 320 x 320 image to 64 x 64 image

and the rest in the test images directory.

1.2 Recognition using PCA

1.2.1 Training data set

The training images are now stored in a training data matrix D where each row contains an image vector of our data set. This vector is obtained by concatenating the rows of the image in 1×4096. So we get a matrix of $p \times 4096$, where p is the number of images in our training data set.

$$D = \begin{bmatrix} I_1(1,1) - \overline{D_1} & \cdots & I_1(1,N) - \overline{D_N} & \cdots & I_1(M,N) - \overline{D_M} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ I_p(1,1) - \overline{D_1} & \cdots & I_p(1,N) - \overline{D_N} & \cdots & I_p(M,N) - \overline{D_M} \end{bmatrix}$$

We then apply PCA algorithm on the matrix. First, we remove the mean to each variable in every row.

$\overline{D_i}$ is mean of intensities at each pixel locations. From this new matrix, get the covariance matrix:

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

Σ is a $d \times d$ variable since we have d $= M \times N = 4096$ variables. Now, we have a very large number of variables ($d = MN$) but a fewer number of images p. This means that the number of nonzero eigenvalues of the covariance matrix is limited to p. Therefore, instead of computing the eigenvectors of the very large covariance matrix $\Sigma = \frac{1}{p-1} D^T D$ which is a $d \times d$ matrix, we can just compute the eigenvectors of $\Sigma' = \frac{1}{p-1} D D^T$ which is $p \times p$ matrix. Then putting the eigenvectors of Σ' as columns of a matrix Φ', the eigenvectors of Σ are given by $D^T \Phi'$ denoted by Φ. The eigenvectors are called EigenFaces as Φ is a $MN \times p$ matrix and



Figure 2: EigenFaces of first 30 Principal Components

every column of Φ, if represented as an image, represents the features of faces. We will only keep the k principal components of them. k is taken from the k largest eigen values, and is

obtained from this condition:

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i} \geq threshold$$

The threshold value is often equal to 0.95. For experimental purpose we keep all the the components and store them in the projection matrix Φ. Thus, any image I_i represented as a vector X_i in the the d-dimensional space, can be projected into the PCA space by computing $F_i = X_i\Phi$, where F_i is the feature vector that represents the initial image I_i.

1.2.2 Test data set

Read one test image I_q and concatenate it in row. Compute the feature vector Φ_q of this test image row X_q multiplying it by Φ. Compute the distance between σ_q and all training feature vectors

$$dist_i = ||F_i - \Phi_q||$$

and find the best match I_z with the minimum distance between the feature vectors. We take the minimum norm of the $dist$ vector : $dist_z$.

2 Experiments

2.1 Face Recognition

For a training data set of 97 images, a test data of 57 images and an obtained threshold values of 0.95, we get an accuracy of 53%



Figure 3: Correct Face Recognition

When some faces in training images have the same orientation as the test face, then the face recognition fails

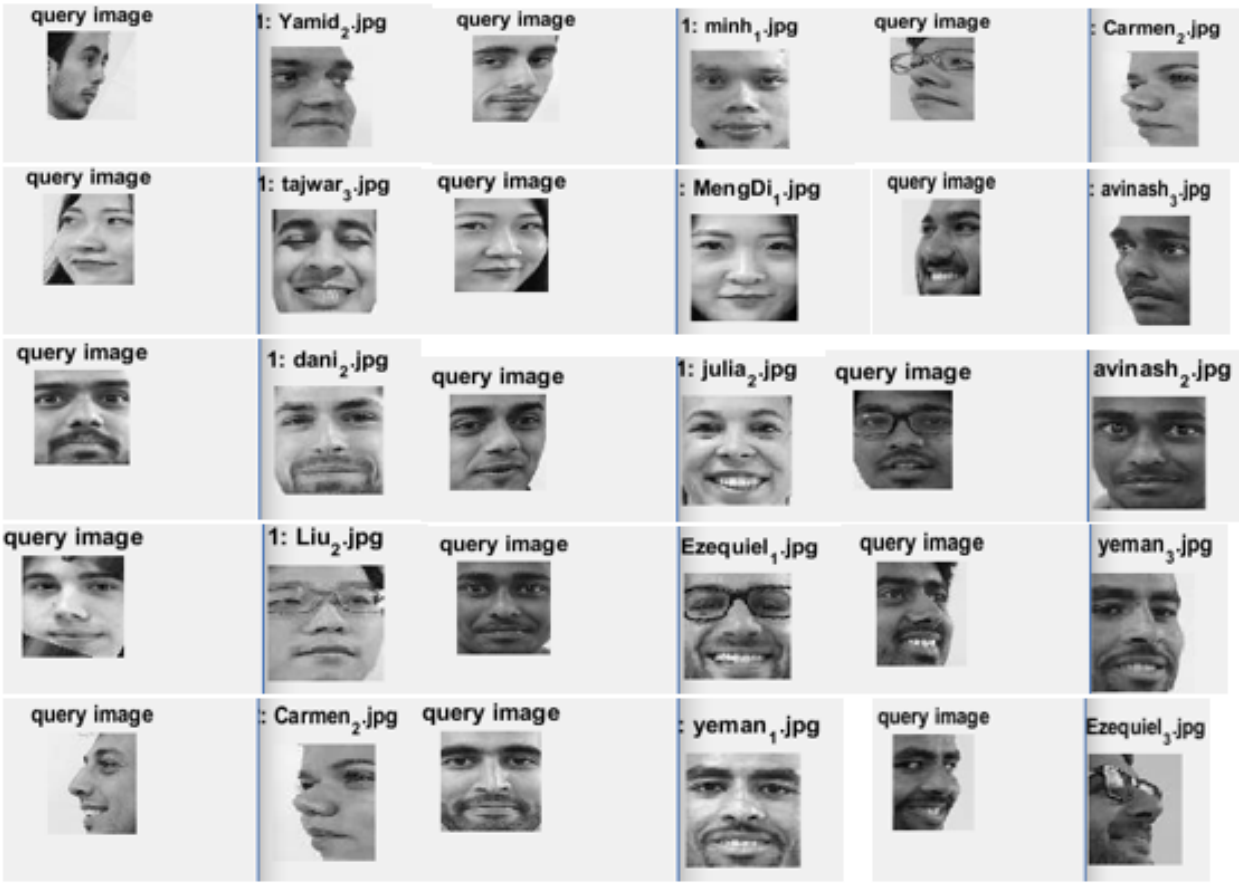


Figure 4: Wrong Face Recognition

We also checked if the correct face is among the first k faces and computed the accuracy for different values of k.

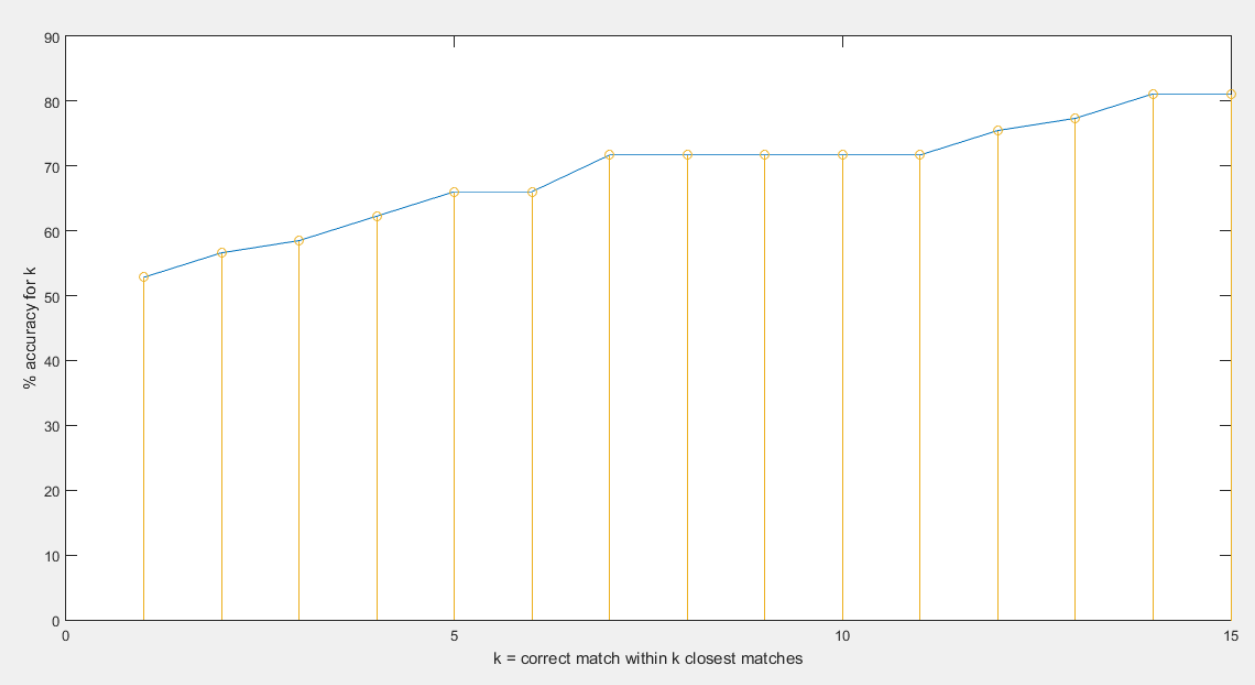


Figure 5: Accuracy plot for different values of k

We see that accuracy gradually increases as the value of k increases. For best 5 matches, we get an accuracy of 66%. An accuracy of 66% is obtained for best 10 matches and 81% for best 15 matches.

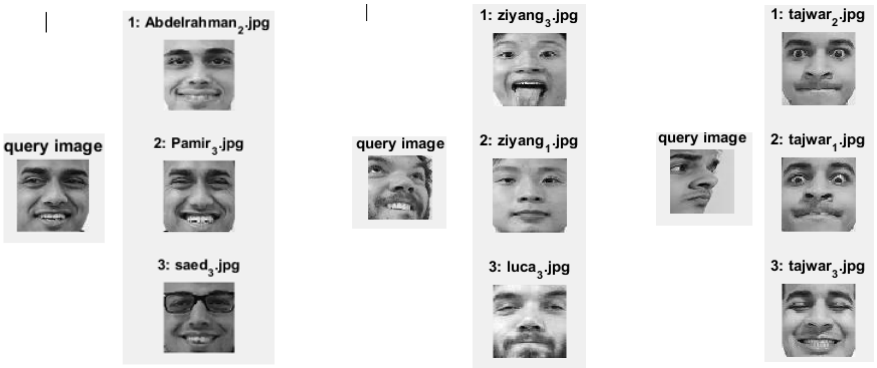


Figure 6: For $k = 3$, best matches

3 Conclusion

Face Recognition using PCA is not 100% accurate when using small number of training images, it tends to recognize other people who have the same face orientation. The realization of this algorithm allowed us, first to understand solving affine transformation using Least Mean Square method and SVD, and finally to understand the aim of PCA.

Overall, this activity was a good learning experience embedding us with real life applications of the things we learned in Linear Algebra classes. It was a good activity as we get to learn a lot in terms of MATLAB scripting and its memory constraints.

We hope to get more of these activities in future.

4 Appendix

GUI Manual

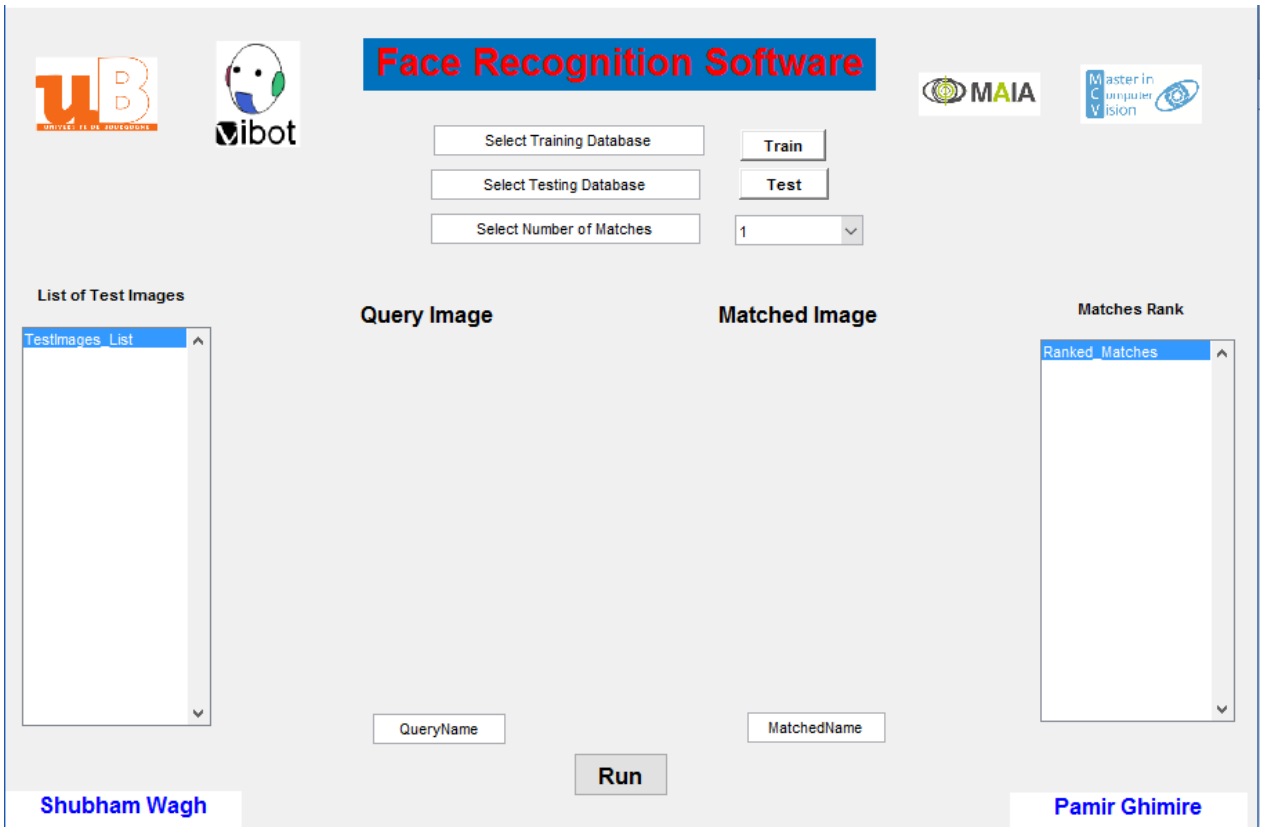


Figure 7: Face Recognition GUI

- Add the project **faceRecognition** to the current directory and set it up as the active project.
- Open **FaceRecognition.m** file and run the script file.
- It will run the program and GUI will open up.
- Now click on **Train** button to select the training images database path.
- Click on **Test** button to select the test images database path.
- Once test images database path is selected, all the list of test images appears on the left hand side listbox in the GUI under **List of Test Images**.

- Select the number of match images required from the the drop down box.
- Click on any one image from the listbox, on clicking, the image gets displayed under **Query Image**
- Click on **Run** button and it will find the best match from training database path and displays it under right hand side listbox in the GUI under **Matched Image**.

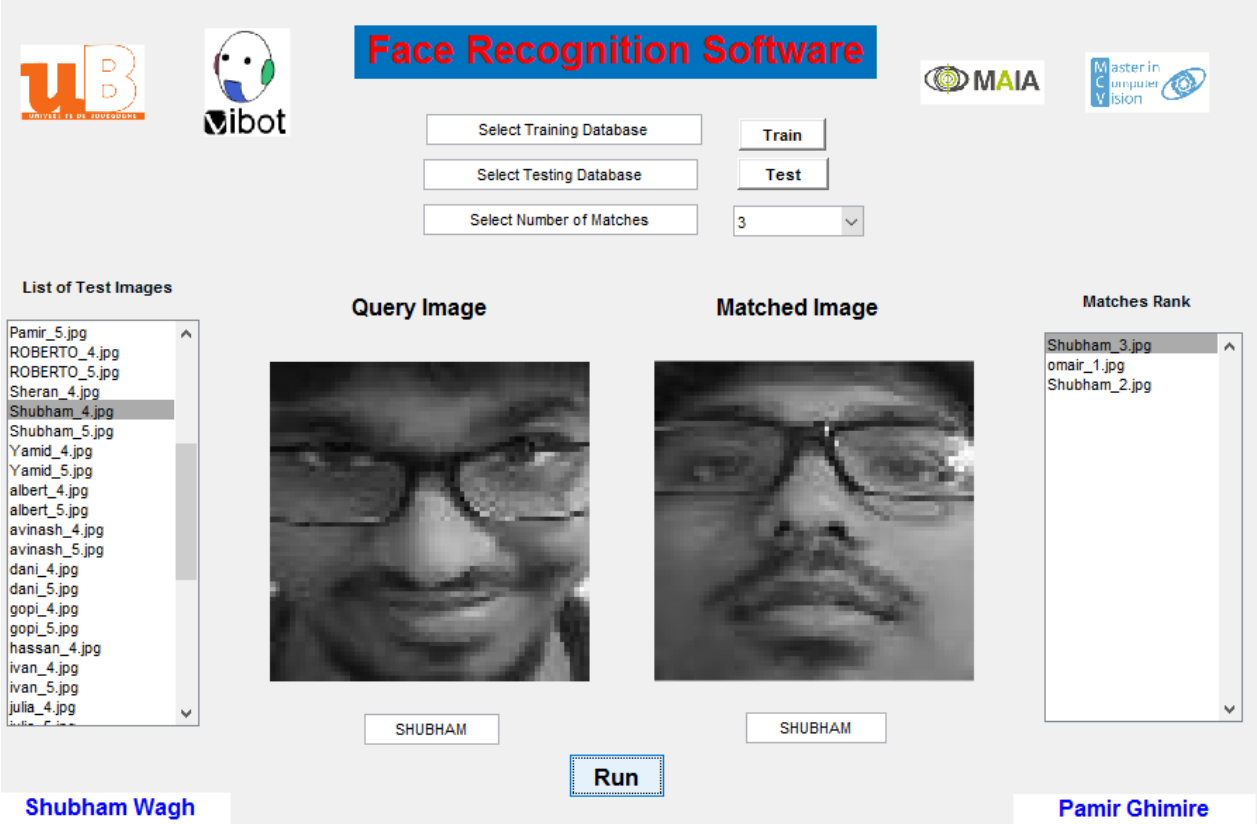


Figure 8: Face Recognition GUI example

- On the right hand side of GUI listbox **Matches Rank** will show the best matches images as per the number of matches selected by the user and are ordered as per their rank.
- Images from **Matches Rank** can be viewed by clicking it from the list.
- Follow the above steps everytime to find the best match image.