Applied Maths Project

Facial Recognition Algorithm using PCA

Master in Medical Imaging and Applications

Presented by:

Benjamin Lalande
Katherine Sheran

1 Introduction

Facial recognition is one of the most known topic of Analysis and Computer Vision by its recurrent apparition in the cinematic world. But most important, facial recognition has developed many applications in our current life like identification of a person in the context of security (Airport, University, Governmental infrastructures) or direct interactions with electronic devices.

It exists several method of facial recognition as holistic matching methods or features based methods. The method we will present you belongs to this second category. It consists to extract a set of defined discriminative features from a image where is present a face. However, in order to deal with large database, it's important to reduce at the maximum the time computation by reducing the the number of variables. The method of the Principal Components Analysis (PCA), which reduce the dimensionality, has been chosen in this purpose.

In this report will be presented the methodology of the facial recognition method based on PCA, a brief description of the implemented algorithm and an analyse of the practiced experiments and their results

2 Methodology

In this section, we will approach the theoretical aspect behind the method. The method is composed of 3 main steps: the normalization of the face images, the computation of the features vectors for each normalized images and the comparison between the given face image and the training data set to find the best match.

2.1 Normalization

As explained in the second part of the paper, the face images you have to obtain from the original ones should be reshaped all in the same size with the face in the middle and the discriminative features at specific coordinates, this is why we have to normalize them. Furthermore, it will help us to perform a faster and more accurate matching later.
The set of images, composed of 320 x 320 face images from each student of the promotion, associating to a set features coordinates. We will perform an affine transformation from our feature coordinates to some predefined coordinates in a 64 x 64 blank image in order to normalize them to this size.
The predefined coordinates are included in the following matrix:

$$\bar{F} = \begin{pmatrix} 13 & 20 \\ 50 & 20 \\ 34 & 34 \\ 16 & 50 \\ 48 & 50 \end{pmatrix}$$

The different set of feature coordinates have the same shape but got in the 320 x 320 frames like, for instance, this one :

$$F = \begin{pmatrix} 102 & 94 \\ 226 & 106 \\ 158 & 156 \\ 100 & 234 \\ 213 & 242 \end{pmatrix}$$

The affine transformation we will perform has therefor this shape :

$$\bar{F} = A * f + b \tag{1}$$

This affine transformation will admit 10 equations (two for each feature) and 6 unknowns. The equation can be reshaped in order to perform Least Mean Square on it :

$$\bar{F} = f_1 * Ab \tag{2}$$

f_1 being the f coordinates with an added column of ones on the right, and Ab being the A unknown matrix with the 2 b unknowns coefficient added under A, so that we get a (5 x 3) times (3 x 2) multiplication resulting in the (5 x 2) predefined coordinates.

f_1 is singular, so the Least Mean Square solution will be computed using the pseudo-inverse thanks to Singular Value Decomposition (SVD). The solution Ab is therefor :

$$Ab = f_1^+ * \bar{F} \tag{3}$$

f_1^+ is the pseudo-inverse of the matrix f_1.
$f_1^+ = V * \Sigma^+ * U^+$ (which are the SVD components corresponding to the eigenvalues and eigenvectors of f_1)

By computing the coefficients for all images and the corresponding normalized feature coordinates in 64 x 64 pixels, we are now able to change the predefined coordinates by the average of normalized feature coordinates.
Updating until it reaches a difference $|\bar{F}_t - \bar{F_{t-1}}| < 1$ pxl (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 Ab and applying the inverse transform on pixel coordinates of the output image , you can find the corresponding pixel coordinates in the original one :

$$Pxl = A^+ * \left(\left| \begin{matrix} x \\ y \end{matrix} \right| . - B \right) \tag{4}$$

Once you have fetched all values inside the original image, you have got your normalized image (Figure 1).

In order to have a sufficient training data set, we dedicate 60 % of the images to this database and let the rest to complete the training data set.

Figure 1: Normalization of the face image

2.2 Computation of the features vectors using PCA

The computation of the features vectors of each normalized images from both training and testing database represent the second step.

The training images are now stored in a training data matrix D where each row contains an image of our data set concatenated in 14096 . So we get a matrix of p * 4096 where p is the number of images in our training data set. You have then to apply PCA algorithm on this matrix. First, remove the mean to each variable in every row.

$$D = \begin{pmatrix} I_1(1,1) - & \dots & I_1(1,N) - & \dots & I_1(M,N) - \\ & . & & . & \\ & . & & . & \\ & . & & . & \\ I_p(1,1) - & \dots & I_1(1,N) - & \dots & I_1(M,N) - \end{pmatrix}$$

From this new matrix, get the covariance :

$$\Sigma = \frac{1}{p-1} D^T D \tag{5}$$

Σ is a d * d variable since we have d = M * N = 4096 variables.

Eigenvalues and eigenvectors of this covariance matrix are then computed and we will only keep the k principal components of them. k is taken from the k largest eigenvalues.

The k principal components are stored into a d * k projection matrix ϕ. Thus, any image I_i represented as a vector X_i in the d-dimensional space, can be projected into the PCA space by computing

$$F_i = X_i * \phi \tag{6}$$

This new k-dimensional vector F_i is the feature vector that represents the initial image I_i.

2.3 Matching between training and testing data sets

After computing a feature vector for each image from training and testing data sets, it is possible to perform the facial recognition by searching a matching for a given image.

The distance between the feature vector of the given image to each element of the training data set is computed.

In theory, the image from the training with the highest probability of matching is the one with the smallest distance between the feature vectors. However in our case, considering the limited size of our training set and the environmental variables, it happens that the closest match is a false positive.

To evaluate the accuracy of a matching, first, we sort the elements of the training data set in ascending order using the distance. Then we compare the labels of the input image to the training data set following this order until we found a true positive match.Each time the labels differs, we increment the error count ϵ.

Then we compute the accuracy following the formula below with ϵ equivalent to the number of false positive detected before.

$$Accuracy = \left(1 - \frac{\epsilon}{totalnumberoftestimages}\right) * 100 \tag{7}$$

3 Implementation

The facial recognition algorithm has been developed under the software Matlab.

3 mains functions have been implemented, respectively corresponding to each of the section of the previous chapter:

- **Normalization.m**: Normalize the initial set of image from the dimensions 320*320 to the dimensions 64*64 while performing a affine transformation to align the facial features of each images to the predefined ones. Then divide the images into two data sets.
- **Do_PCA.m**: For each image of the training and testing data sets, compute their features vectors.
- **Recognition.m**: Compare the given image to the elements of the training data set, compute the accuracy and show the given image, closest match and closest positive match with the accuracy.

These 3 functions are independents. The transfer of data between them is doing by saving and loading matrix files.
Recognition.m is the only function which required an input: the name of the image from the testing data set to compare without the format suffix.

4 Results

As explained previously, the results, on figure 2 and 3, present from left to right the input image, the closest match in term of distance between the features vectors but not necessary a positive match, and the closest positive match among the training data set.

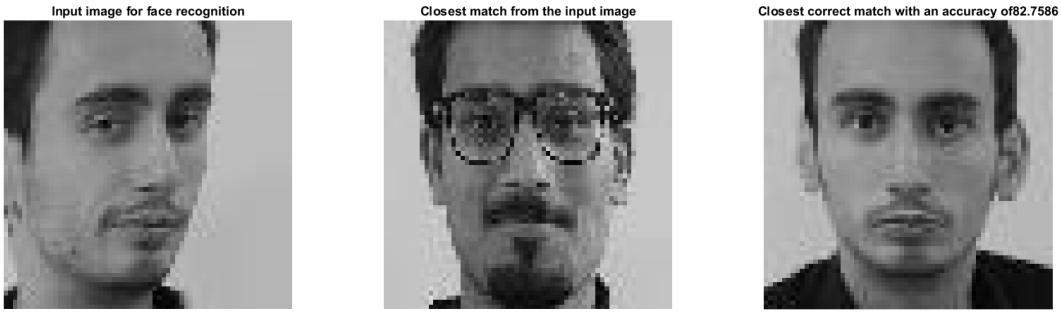

Figure 2: Facial recognition for Benjamin with an accuracy of 82.7 %

Figure 3: Facial recognition for Katherine with an accuracy of 94.2 %

We observe on the figure 2 and 3 that the closest matches are false positive matches, even if the accuracy of the closest positive matches are highs.

5 Conclusion

The algorithm of Face Recognition using the method of the Principal Component Analysis is not 100 % accurate.
This can be explain by the smaller number of samples in the training data set, which increase the possibility to have two different persons with the same facial orientation.

This realization of this project has an opportunity to practice the different methods seen in course such as Singular Value Decomposition, Least Mean Square or Principal Component Analysis.
It has been also the opportunity to learn how to build classifiers and their mechanisms, and develop our abilities concerning the proclamation on Matlab.