Realization and optimization of face recognition system based on MATLAB

Xi Yan, Gaqiong Liu
Department of Computer, Jiangsu University of Science and Technology, Zhenjiang, China

Abstract. In this dissertation, several techniques used in face recognition have been investigated. Besides, a critical problem in face recognition which is illumination variation has been considered. In this dissertation, a face recognition system has been built using MATLAB. In this system, after the Eigenface has been achieved, two techniques to deal with illumination changing have been implemented, shadow compensation and local binary pattern (LBP) respectively. After the experiment, LBP shows a better performance than shadow compensation on Yale B cropped database.

Keywords: Face recognition, Eigenface, Shadow compensation, Local Binary Pattern

1. Introduction
The term “Face Recognition” generally means to implement a system with face recognition ability by electronic devices. It is a key technology in digital image processing and computer vision which is used in systems for detecting and identifying faces from captured or given digital images or video frames.

Nonlinear analysis which is more complicated to implement than linear ones due to the projection between the facial images and the points in the feature space is nonlinear. Kernel Principal Component Analysis (KPCA) [3] is the most common used one in nonlinear analysis.

Model-based face recognition which involves models to achieve the recognition function. The model used typically consisted of a geometric mesh with 44 facial muscles, the points to attach the muscles to skin and the elastic properties of the skin [4]. After the model is constructed, the input facial image is warped into this physical model for testing. Feature-based elastic bunch graph matching [5] is a widely used technique belongs to model-based face recognition, which uses nodes and edges to represented faces for comparison.

2. Proposed Method

2.1. Technique used for Feature Recognition
Face recognition is the last stage of the entire system and also the most important one. There are two stages involved in a traditional recognition process, training and testing respectively. Training is the key of feature recognition process and it requires statistical algorithm or other techniques to fulfill the task.

In this dissertation, the Eigenface approach is used for dimensional reduction and several encoding techniques are available. The most common one is PCA and is also used in this dissertation as the technique to reduce the dimension of the database.
In this case, the principal components are the eigenvectors that projected to the feature space. A detailed procedure of PCA is shown below.

1) Construct Training Set

Start with the training stage, the first thing is to construct the training database. Given a 2D image \( I_i \) which has a dimension of \( m \times n \), in order to load it into the training database, it need to be concatenated into a 1-D vector with length \( N \) which is equal to the total number of the pixel in \( I_i \), i.e. \( m \times n \). Therefore, for a training set with \( M \) images, the dimension is \( N \) by \( M \) and each image vector is represented by \( \Gamma_i \). An example for the concatenation is shown below.

![Figure 1. Example for image concatenation.](image)

2) Face Normalization

Face Normalization is the necessary step in PCA. The normalized face \( \Phi_i \) is obtained by subtracting the mean face \( \Psi \) from \( \Gamma_i \). The mean face can be calculated as follow.

\[
\Psi = \frac{1}{M} \sum_{i=1}^{M} \Gamma_i
\]

The normalized face is obtained by,

\[
\Phi_i = \Gamma_i - \Psi
\]

A new matrix is constructed by normalized faces given by \( A = (\Phi_1, \Phi_2, \Phi_3\ldots \Phi_M) \). This step adjusts the mean and unit variance of the input images.

3) Calculate the Covariance Matrix

The covariance matrix is given by:

\[
Q = AA^T
\]

Where \( A \) is the matrix formed by normalized faces and \( AT \) is the transpose of \( A \)

4) Compute the Eigenvalues and Eigenvectors

At this stage, the problem is commonly solved as a linear least-square problem by singular value decomposition (SVD). More specifically, the eigenvectors and eigenvalues can be computed from covariance matrix by using SVD. Moreover, eigenvalues and eigenvectors are sorted in descending order. The ordered eigenvectors \( \lambda_i \) forms a new matrix \( W \) which is given by:

\[
W = [\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_M]
\]

In order to robust the compute efficiency, the eigenvectors with the largest \( K \) eigenvalues could be choose from the \( M \) eigenvectors, where \( \theta \) is the threshold value defined by the program and \( K \) is changing with the variation of \( \theta \). The formula is represented as:
5) Projection of Training Images

This is the last step for training in which all training faces are projected into the feature space. This can be done by calculating the product of the image with the corresponding eigenvectors as shown in the equation below.

\[ Y = W^T A \]  

(6)

Where WT is the transpose of W.

2.2. Techniques for Illumination Compensation

The methods to compensate illumination variation have been under researched over years and some valid solutions have been developed. Two successful methods, shadow compensation and LBP respectively, are introduced in this section and both techniques are implemented in this dissertation.

The general aim for the method to solve the illumination problem is to pre-process the image in order to find the information which covered by the shadow.

The task for this method is to compensate the distorted magnitude components. The procedure is shown below.

1). Generate an auxiliary magnitude by averaging the magnitudes components of the images used in the face database. The auxiliary magnitude is represented by \( |F_{IA}^a| \), which indicates the magnitudes of general human faces.

2). Implement 2-D Fourier transform to the facial images in the used database.

3). Obtain the magnitude components and phase components of those images in frequency domain.

4). Replace the magnitude components \( |F_{IA}(u,v)| \) by a modified magnitude \( |F_{IM}(u,v)| \) which is the mean of \( |F_{IA}(u,v)| \) and \( |F_{IA}^a| \).

5). Reconstruct the images by using inverse 2-D Fourier transform in (7).

\[ I_{Re}(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F_{IM}(u,v) e^{j2\pi I_{IA}(ux/M + vy/N)} \]  

(7)

Figure 2 illustrates the above procedure by using block diagram.

![Figure 2. Procedure for shadow compensation method](image)

Examples are shown below which are the results after implementing shadow compensation on some faces covered by shadows.
The first row in Figure 3 is the faces covered by shadow and the second row is the faces after compensation which is part of the results of this dissertation. As it is shown, shadow compensation is not have the strong effects on the face with front light such as Figure 3 (a). However, it recovers the faces in the shadow efficiently such as Figure 3 (b) and (d).

Different from shadow compensation, LBP is a pixel-based pre-processing technique. It summarizes local structures of images by comparing each pixel with its neighboring pixels. The comparison is mainly based on the value of gray-level of corresponding pixel. Because of this processing philosophy, this technique is robust the accurate of face recognition when dealing with illumination problem. However, also due to the way that LBP process an image, the compute time required is longer than shadow compensation.

LBP encode the image by using LBP operator which labels the pixels in the image with decimal numbers. An example of an original LBP operator is shown in Figure 4.

![Figure 4. Example of LBP operator](image)

In Figure 4, each pixel is compared with its neighbors in a 3 × 3 neighborhood block. If a neighbor pixel is greater than or equal to the center pixel, it is encoded with binary number 1. Otherwise, a neighbor pixel is encoded with a binary 0. For each center pixel, the binary number is obtained by

![Figure 5. Example of implementation of LBP.](image)
concatenating all these binary values in a clockwise direction which starts from the top-left corner as shown in Figure.4. Then, the center pixel is labelled by this generated binary value. Examples of implement LBP with standard LBP operator as shown in Figure.5.

As shown in Figure.5, after implement LBP, the texture of the faces are clearly obtained. However, the results are still affected by the orientation of the light source in Figure.5 (f) and (g). The second row in Figure.5 also picked from the results of this dissertation.

3. Analysis and Results

3.1. Performance of Eigenface on Yale B Cropped Database

With respect to the Eigenface method implemented on Yale B database, for the selected 30 subjects, 5 images are for the training and 5 for testing like the previous case. Thus, there are 150 images in total for training and 150 images for testing.

In Table 1, the accuracy for the threshold value of 1 is 14.7% and the number of eigenvectors involved is 150. Moreover, the number of eigenvectors involved is decreased dramatically with the threshold value drop from 1 to 0.9.

Table 1. Accuracy of Eigenface technique with different threshold value on Yale B cropped database.

| Threshold Value | Number of Eigenvector Involved | Accuracy |
|-----------------|-------------------------------|----------|
| 1               | 150                           | 14.7%    |
| 0.9             | 15                            | 10.0%    |
| 0.7             | 2                             | 6.0%     |
| 0.5             | 1                             | 8.0%     |
| 0.2             | 1                             | 8.0%     |

3.2. Performance of Eigenface Combined with Pre-Processing Techniques Yale B Cropped Database

The performances of implementing shadow compensation on Yale B cropped database in shown in Table.2.

Table 2. Accuracy of Eigenface + shadow compensation technique with different threshold value on Yale B cropped database.

| Threshold Value | Number of Eigenvector Involved | Accuracy |
|-----------------|-------------------------------|----------|
| 1               | 150                           | 62.7%    |
| 0.9             | 65                            | 54.7%    |
| 0.7             | 19                            | 44.0%    |
| 0.5             | 7                             | 27.3%    |
| 0.2             | 2                             | 9.3%     |

As shown above, the accuracy is largely robust. The accuracy for threshold value of 1 is increased from 14.7% to 62.7%, and for threshold value of 0.5 the accuracy is raised from 8% to 27.3%. However, with the low threshold value such as the last row in Table.1 and Table.2, the increment is not obvious (only 1.3%). In addition, with the pre-processing technique used, the number of eigenvectors involved for each threshold value except 1 is added.
Table 3. Accuracy of Eigenface + LBP technique with different threshold value on Yale B cropped database.

| Threshold Value | Number of Eigenvector Involved | Accuracy |
|-----------------|-------------------------------|----------|
| 1               | 150                           | 76.7%    |
| 0.9             | 117                           | 78.0%    |
| 0.7             | 72                            | 20.0%    |
| 0.5             | 38                            | 48.0%    |
| 0.2             | 8                             | 45.3%    |

Table 3 above gives the results of implementing LBP as the pre-processing technique. The result shows an impressive improvement compared with the data in Table 4.1, even in Table 4.2 under some threshold values. In this case, the accuracy with the threshold value of 1 and 0.9 is 76.7% and 78% respectively. The small incensement may be caused by some errors that can be tolerated. It is worth to be noticed that for the threshold value of 0.2, compared with these three case on the Yale B cropped database, the last one shows a great increase from the accuracy of less than 10% to almost 50%.

4. Conclusion

Overall, the system with LBP on Yale B cropped database has a higher accuracy than the one using shadow compensation. The peak value reaches 78% when the threshold value is 0.9. Moreover, even with the small energy used, i.e. low threshold value, the LBP still can give an eligible performance.

Moreover, Principal Feature Analysis (PFA) which is the extension version of PCA can be tested. PFA chooses a part of the features that contains the most useful information rather than all of the features and it can reduce the processing time which is important for the Android application. For the techniques deal with illumination problem, some extended LBP operators and variation LBP is worthy to be tested which may robust the accuracy.

References

[1] Chakraborty D, Saha SK, Bhuiyan MA. Face Recognition using Eigenvector and Principle Component Analysis. International Journal of Computer Applications. 2012, 50: 42-49.
[2] Delac K, Grgic M, Grgic S. Independent comparative study of PCA, ICA and LDA on the FERET data set. International Journal of Imaging Systems and Technology.2005, 15: 252-260.
[3] Kim KI, Jung K, Kim HJ. Face Recognition Using Kernel Principal Component Analysis. IEEE Signal Processing Letters.2002, 9: 40-42.
[4] Bartlett MS, Sejnowski TJ. Face Image Analysis by Unsupervised Learning. Kluwer Academic Publishers. 2001.
[5] Sarfraz M. Computer-Aided Intelligent Recognition: Techniques and applications. John Wiley & Sons Ltd. 2005.