Fingerprint Recognition using MatLab

Rajab Challoo and Avinash Ramalingegowda
Department of Electrical Engineering
MSC 192, 700 University Blvd.
Texas A&M University, Kingsville, Texas

ABSTRACT

In this paper, a new method for fingerprint recognition with wavelet transform and Principal Component Analysis (PCA) is developed using MatLab. Fingerprint images used for this purpose are taken in grey-scale without any pre-processing (i.e., smoothing, minutiae extraction, ridge thinning and ridge segmentation). Fingerprint images are chosen in such a way that the core point is located at the center of the image. The proposed algorithm is tested on a small fingerprint database using PCA and recognition rate of more than 90% is obtained. This algorithm can be used on a desktop computer to recognize small group of fingerprints effectively and efficiently.

Keywords: Wavelet Transform, Fingerprint, DWT, PCA, Eigenfingers.

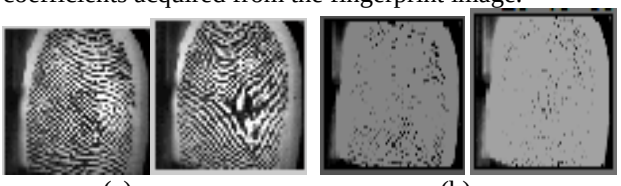
1. INTRODUCTION

Fingerprints are imprints formed by friction ridges of the skin in fingers and thumbs. Fingerprint recognition remains as one of the most prominent and the oldest biometric identification methods. The pattern of ridges and furrows as well as the minutiae points in the fingerprint are believed to be unique for each and every human being. The ridge characteristics that occur at a ridge bifurcation or at ridge ending are called as Minutiae points [1].

Fingerprint matching can be classified into two categories: minutae based and image based. Minutiae based technique is the oldest one which is employed almost everywhere. This technique first finds the minutiae points and their corresponding position on the finger and this information is stored as features of the fingerprint which later used for matching purpose. Sometimes it is very difficult to extract the minutiae points from a low quality fingerprint and preprocessing of the image is a must in order to effectively extract the features. Minutiae based approach also uses more memory and it is a time consuming process. While the image based technique has solved some of the issues of the minutiae based approach. Image based approach offers much higher computation efficiency with minimum pre-processing and proves also effective even when the image quality is low [2]. However, this approach is vulnerable to shape distortions as well as variation in position, scale and orientation angle.

2. DISCRETE WAVELET TRANSFORM

Discrete Wavelet Transform (DWT) is the recent advancement in the signal processing field. Taking DWT of a signal gives important characteristics of the signal [3]. DWT can be applied to both one and two dimensions. In this paper, fingerprint is taken as two dimensional grey-scale image and each row of the image is represented as a one dimensional signal. One dimensional DWT is applied to each row of the grey-scale image. Based on the simulation results obtained only the approximate coefficients are retained while the detail coefficients which are insignificant and do not contribute much to the recognition accuracy were neglected. After applying DWT to all the rows and down-sampling the resulting approximate coefficients, a compressed version of the image is obtained. Similarly same method is applied to each columns of the resultant image. This process is known as one level wavelet decomposition of the image. In our approach we have continued this process to achieve two level decomposition. The final image is much smaller version of the fingerprint image that was initially taken. This method not only simplifies the computations involved to extract features but also consumes less memory to store the approximate coefficients acquired from the fingerprint image.



(a) (b)

Figure 1: Fingerprint image acquired from different person (a), and their corresponding two level DWT approximate image (b).

3. PRINCIPAL COMPONENT ANALYSIS

Principal Components Analysis (PCA) also known as Karhunen and Leove (KL) transform, is a way of identifying patterns in data, and expressing the data in such a way as to highlight their similarities and differences. Since patterns in data can be hard to find in data of high dimension, where the luxury of graphical representation is not available, PCA is a powerful tool for analyzing data [4]. The other main advantage of PCA is that once you have found these patterns in the data, and you compress the data, ie. by reducing the number of dimensions, without much loss of information. Here an attempt to use this method to recognize fingerprint images has been made and satisfactory results were obtained.

In mathematical terms, the objective is to find the principal components of the distribution of fingerprint, or the eigenvectors of the covariance matrix of the set of fingerprint images. These eigenvectors can be thought of as a set of features which together characterize the variation between fingerprint images. Each image location contributes more or less to each eigenvector, so that we can display the eigenvector as a sort of complex fingerprint called an eigenfinger.

Each fingerprint image in the training set can be represented exactly in terms of a linear combination of the eigenfinger. The number of possible eigenfinger is equal to the number of fingerprint images in the training set. However, the fingerprint images can also be approximated using only the "best" eigenfingers—those that have the largest eigenvalues, and which therefore account for the most variance within the set of fingerprint images. The primary reason for using fewer eigenfingers is computational efficiency.

The eigenfinger approach for fingerprint recognition involves the following initialization operations:

1. Acquire a set of training images.
2. Calculate the eigenfingers from the training set, keeping only the best *M* images with the highest eigenvalues. These *M* images define the "finger space". As new fingerprint images are experienced, the eigenfingers can be updated.
3. Calculate the corresponding distribution in *M*-dimensional weight space for each known individual (training image), by projecting their fingerprint images onto the finger space.

Having initialized the system, the following steps are used to recognize new fingerprint images:

1. Given an image to be recognized, calculate a set of weights of the *M* eigenfingers by projecting it onto each of the eigenfingers.
2. Calculate the characteristic weight pattern of the new fingerprint image.
3. Using Euclidian distance method to classify the image.
4. (Optional) Update the eigenfingers and/or weight patterns.

4. CALCULATING EIGENFINGERS

Let a fingerprint image Γ(x,y) be a two-dimensional *N* by *N* array of intensity values after taking wavelet transform. An image may also be considered as a vector of dimension N^2, so that a typical image of size 64 by 64 becomes a vector of dimension 4096, or equivalently, a point in 4096-dimensional space. An ensemble of images, then, maps to a collection of points in this huge space.

Fingerprint images, being similar in overall configuration, will not be randomly distributed in this huge image space and thus can be described by a relatively low dimensional subspace. The main idea of the principal component analysis is to find the vector that best account for the distribution of these images within the entire image space. These vectors define the subspace of fingerprint images, which we call "finger space". Each vector is of length N^2, describes an *N* by *N* image, and is a linear combination of the original fingerprint images. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original fingerprint images, and because they are similar to fingerprint in appearance, they are referred to as "eigenfingers".

Let the training set of fingerprint images be Γ_1, Γ_2, Γ_3, …, Γ_M. The average image of the set if defined by $\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n$. Each image differs from the average by the vector $\Phi_n = \Gamma_n - \Psi$. An example training set is shown in Figure 1a, with the average image Ψ shown in Figure 1b. This set of very large vectors is then subject to principal component analysis, which seeks a set of *M* orthonormal vectors, μ_n, which best describes the distribution of the data. The *k*th vector, μ_k is chosen such that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}(\mu_k^T \Phi_n)^2 \qquad (1)$$

is a maximum, subject to

$$\mu_l^T \mu_k = \begin{cases} 1, l = k \\ 0, otherwise \end{cases} \quad (2)$$

The vectors μ_k and scalars λ_k are the eigenvectors and eigenvalues, respectively, of the covariance matrix [4]

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi_n \Phi_n^T = AA^T \quad (3)$$

where the matrix $A = [\Phi_1 \Phi_2 ... \Phi_M]$. The matrix *C*, however, is N^2 by N^2, and determining the N^2 eigenvectors and eigenvalues is an intractable task for typical image sizes [4]. A computationally feasible method is needed to find these eigenvectors.

If the number of data points in the image space is less than the dimension of the space ($M < N^2$), there will be only $M - 1$, rather than N^2, meaningful eigenvectors (the remaining eigenvectors will have associated eigenvalues of zero). Fortunately, we can solve for the N^2-dimensional eigenvectors in this case by first solving for the eigenvectors of and *M* by *M* matrix—e.g., solving a 25 x 25 matrix rather than a 8464 x 8464 matrix—and then taking appropriate linear combinations of the fingerprint images Φ_n. Consider the eigenvectors ν_n of $A^T A$ such that

$$A^T A \nu_n = \lambda_n \nu_n \quad (4)$$

Premultiplying both sides by *A*, we have

$$AA^T A \nu_n = \lambda_n A \nu_n \quad (5)$$

from which we see that $A\nu_n$ are the eigenvectors of $C = AA^T$.

Following this analysis, we construct the *M* by *M* matrix $L = A^T A$, where $L_{mn} = \Phi_m^T \Phi_n$, and find the *M* eigenvectors ν_n of *L*. These vectors determine linear combinations of the *M* training set fingerprint images to form the eigenfingers μ_n:

$$\mu_n = \sum_{k=1}^{M} \nu_{nk} \Phi_k = A\nu_n, n = 1, ..., M \quad (6)$$

With this analysis the calculations are greatly reduced, from the order of the number of pixels in the images (N^2) to the order of the number of images in the training set (*M*). In practice, the training set of fingerprint images will be relatively small ($M < N^2$), and the calculations become quite manageable [4]. The associated eigenvalues allow us to rank the eigenvectors according to their usefulness in characterizing the variation among the images.

5. USING EIGENFINGERS TO CLASSIFY A FINGERPRINT IMAGE

The eigenfinger images calculated from the eigenvectors of *L* span a basis set with which to describe fingerprint images. As mentioned before, the usefulness of eigenvectors varies according their associated eigenvalues. This suggests we pick up only the most meaningful eigenvectors and ignore the rest, in other words, the number of basis functions is further reduced from *M* to *M'* (*M'*<*M*) and the computation is reduced as a consequence.

A new fingerprint image Γ is transformed into its eigenfinger components (projected onto "finger space") by a simple operation

$$\omega_n = \mu_n (\Gamma - \Psi) \quad (7)$$

for *n*=1,……,*M'*. This describes a set of point-by-point image multiplications and summations.

The weights form a vector $\Omega^T = [\omega_1, \omega_2, ..., \omega_{M'}]$ that describes the contribution of each eigenfinger in representing the input fingerprint image, treating the eigenfingers as a basis set for fingerprint images. The vector may then be used in a standard pattern recognition algorithm to find which of a number of predefined finger classes, if any, best describes the fingerprint. The simplest method for determining which finger class provides the best description of an input fingerprint image is to find the finger class *k* that minimizes the Euclidian distance

$$\varepsilon_k^2 = \left\| (\Omega - \Omega_k)^2 \right\| \quad (8)$$

where Ω_k is a vector describing the *k*th finger class. The finger classes Ω_k are calculated by averaging the results of the eigenfinger representation over a small number of fingerprint images (as few as one) of each individual. A fingerprint is classified as "unknown", and optionally used to create a new finger class.

Because creating the vector of weights is equivalent to projecting the original fingerprint image onto to low-dimensional finger space, many images (most of them looking nothing like a fingerprint) will project onto a given pattern vector. This is not a problem for the system, however, since the distance ε between the image and the finger space is simply the squared distance between the mean-adjusted input image $\Phi = \Gamma - \Psi$ and $\Phi_f = \sum_{i=1}^{M'} \omega_i \mu_i$, its projection onto finger space:

$$\varepsilon^2 = \left\| \Phi - \Phi_f \right\|^2 \qquad (9)$$

Thus there are four possibilities for an input image and its pattern vector: (1) near finger space and near a finger class; (2) near finger space but not near a known finger class; (3) distant from finger space and near a finger class; (4) distant from finger space and not near a known finger class.

In the first case, an individual is recognized and identified. In the second case, an unknown individual is present. The last two cases indicate that the image is not a fingerprint image. Case three typically shows up as a false positive in most recognition systems; in this framework, however, the false recognition may be detected because of the inclusion of no rejection option.

6. SUMMARY OF FINGERPRINT RECOGNITION PROCEDURE

The eigenfinger approach for fingerprint recognition is summarized as follows:

1. Collect a set of characteristic fingerprint images of the known individuals. This set should include a number of images for each person, with some variation in position (say five images of five people, so *M=25*).
2. Take the two-level DWT on each row and column of the image as explained earlier.
3. Calculate the (M x M) matrix *L*, find its eigenvectors and eigenvalues, and choose the *M'* eigenvectors with the highest associated eigenvalues (let *M'* = M in this example).
4. Combine the normalized training set of images according to Eq. (6) to produce the *M'* eigenfingers $\mu_k, k = 1,.....,M'$.
5. For each known individual, calculate the class vector Ω_k by averaging the eigenfinger pattern vectors Ω [from Eq. (8)] calculated from the original (five) images of the individual. Choose a threshold θ_ε that defines the maximum allowable distance from any finger class, and a threshold θ that defines the maximum allowable distance from finger space [according to Eq. (9)].
6. For each new fingerprint image to be identified, calculate its pattern vector Ω, the distance ε_k to each known class, and the distance ε to finger space. If the minimum distance $\varepsilon_k < \theta_\varepsilon$ and the distance $\varepsilon < \theta$, classify the input finger as the individual associated with class vector Ω_k.
7. (Optional) If the new image is classified as a known individual, this image may be added to the original set of familiar fingerprint images, and the eigenfingers may be recalculated (steps 1-4). This gives the opportunity to modify the finger space as the system encounters more instances of known fingerprints.

7. EXPERIMENTAL RESULTS

The proposed algorithm has been tested on a database of 168 fingerprint images of size 256 x 256 including 8 images per finger from 21 individuals. The images have been selected from the database [5] based on the position of the fingerprint pattern inside the image, selecting those images whose core point is located close to the centre of the image. The fingerprint database consists of 5 arches, 4 whorls and 12 loops. During the training phase 5 out of 8 images of each person i.e., 25 images are used and the rest 3 including the ones used for training are used during recognition phase with no rejection option.

Three experiments have been performed. In the first experiment fingerprints from 5 (N) individuals consisted of arch, whorl and loop as shown in the figure 2 having 8 (M) images each were selected from the database [5] and was tested with the fingerprint recognition algorithm and recognition accuracy for different wavelet filters used is listed in the table 1.

Figure 2: Fingerprint images used for experiment 1.

Table 1: Recognition rate vs wavelet filter used in experiment 1

Type of filter	Number of training fingerprints	Number of fingerprint recognized correctly	Percentage accuracy
Haar	25/40	39/40	97.5
Db1	25/40	39/40	97.5
Db2	25/40	40/40	100
Db4	25/40	39/40	97.5
Db5	25/40	40/40	100
Db6	25/40	40/40	100
Sym1	25/40	39/40	97.5
Sym2	25/40	40/40	100
Sym4	25/40	38/40	95
Sym6	25/40	40/40	100
Bior2.4	25/40	40/40	100
Bior4.4	25/40	40/40	100
Bior1.1	25/40	39/40	97.5
Coif1	25/40	39/40	97.5
Coif2	25/40	39/40	97.5
Average			98.5

The second experiment was conducted with the entire database [5] consisting all types of fingerprint images. Recognition accuracy for the experiment conducted is as shown in table 2.

Table 2: Recognition rate vs wavelet filter used in experiment 3

Type of filter	Number of training fingerprints	Number of fingerprint recognized correctly	Percentage accuracy
Haar	25/40	38/40	95
Db1	25/40	38/40	95
Db2	25/40	39/40	97.5
Db4	25/40	38/40	95
Db5	25/40	39/40	97.5
Db6	25/40	39/40	97.5
Sym1	25/40	38/40	95
Sym2	25/40	39/40	97.5
Sym4	25/40	37/40	92.5
Sym6	25/40	39/40	97.5
Bior2.4	25/40	38/40	95
Bior4.4	25/40	39/40	97.5
Bior1.1	25/40	38/40	95
Coif1	25/40	38/40	95
Coif2	25/40	38/40	95
Average			95.83

In the last experiment fingerprint images from 5 individual having only loops were selected from the database [5] having 8 (M) images each and the recognition algorithm was applied. Table 3 gives the recognition accuracy for different wavelet filters used.

Table 3: Recognition rate vs wavelet filter used in experiment 4

Type of filter	Number of training fingerprints	Number of fingerprint recognized correctly	Percentage accuracy
Haar	105/168	150/168	89.29
Db1	105/168	150/168	89.29
Db2	105/168	150/168	89.29
Db4	105/168	148/168	88.09
Db5	105/168	150/168	89.29
Db6	105/168	149/168	88.69
Sym1	105/168	149/168	88.69
Sym2	105/168	149/168	88.69
Sym4	105/168	150/168	89.29
Sym6	105/168	151/168	89.88
Bior2.4	105/168	154/168	91.67
Bior4.4	105/168	150/168	89.29
Bior1.1	105/168	149/168	88.69
Coif1	105/168	150/168	89.29
Coif2	105/168	149/168	88.69
Average			89.21

A comparison of our method with the method proposed by Y. H. Fung and Y. H. Chan in [2] is shown in table 4. Here the results from the experiment 2 are compared with the results obtained using 2-NN in [2].

Table 4: Comparison of recognition rate of Y. H. Fung [2] and ours.

Type of filter	Number of training fingerprints	Number of fingerprint recognized correctly	Percentage accuracy	
			Ours	Method proposed in [4] using 2-NN
Db5	105/168	150/168	89.29	88.10
Db6	105/168	149/168	88.69	88.10
Db10	105/168	150/168	89.29	77.38
Sym6	105/168	151/168	89.88	91.67
Sym9	105/168	149/168	88.69	92.86
Sym10	105/168	150/168	89.29	85.71
Average			89.19	87.30

From table 4 it can be seen that the proposed method gives better accuracy for most of the wavelet filters than the one proposed by Y. H. Fung and Y. H. Chan in [2].

From tables 1 and 2, it is evident that for the database chosen the algorithm yields different results when different family of wavelet filters are used. When only fingerprints from 5 people were taken from the database, the average accuracy of recognition was found to be 98.5%. When the entire database was chosen the average accuracy of the system was found to be 89.21%. Based on the average accuracy obtained for different set of fingerprint images from the database [5], it can be concluded that as the number of people are increased, the accuracy of the system decreases. It can also be noted from the table 1 that for a small set of database at least few filters gave 100% recognition rate whereas for a larger database none of the filters yielded the same.

From tables 1 and 3, it can be concluded that the algorithm works better when different types of fingerprint images (whorl, loop and arch) are used rather than the same type (only loops, or whorls, or arches).

8. CONCLUSIONS

In this paper, we have proposed a simple method for fingerprint identification using wavelets and principal component analysis. This algorithm can be implemented on a desktop computer and used for identifying a small group of people. Since the images are in greyscale and carefully chosen in such a way as to eliminate the preprocessing, thus providing the computational efficiency to the system developed. Though the wavelet transform stage is slow because of single level wavelet decomposition used compared to the rest of the stages but overall speed of the system is comparable to any other algorithms that have been proposed. The method has been successfully compared against one of the methods recently proposed and results prove that our system is capable of doing the same task at a faster rate with more accuracy for small group of people. The high recognition rates achieved by our method as well as its low computational complexity reveal that the method can be used to efficiently solve a security problem involving a small number of fingerprint images.

9. REFERENCES

[1] D. Maltoni, D. Maio, A. K. Jain, S. Prabhakar, *Handbook of Fingerprint Recognition*, Springer, (2003).

[2] Y. H. Fung and Y. H. Chan, "Fingerprint Recognition using Improved Wavelet Domain Features", *Proc, of International Symposium on Intelligent Multimedia, Video and Speech Processing*, October 20-22, 2004.

[3] S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, (1998).

[4] M. Turk and A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol. 3, No. 1, 1991.

[5] Fingerprint Database "Fingdb.zip", Biometric Systems Lab, University of Bologna – Italy, available at website: http://www.csr.unibo.it/research/biolab/ (05/2005).