Hindawi Publishing Corporation
Journal of Biomedicine and Biotechnology
Volume 2012, Article ID 324249, 9 pages
doi:10.1155/2012/324249

Research Article

Finger Vein Recognition Based on $(2D)^2$ PCA and Metric Learning

Gongping Yang, Xiaoming Xi, and Yilong Yin

School of Computer Science and Technology, Shandong University, Jinan 250101, China

Correspondence should be addressed to Yilong Yin, ylyin@sdu.edu.cn

Received 22 February 2012; Accepted 19 March 2012

Academic Editor: Sabah Mohammed

Finger vein recognition is a promising biometric recognition technology, which verifies identities via the vein patterns in the fingers. In this paper, $(2D)^2$ PCA is applied to extract features of finger veins, based on which a new recognition method is proposed in conjunction with metric learning. It learns a KNN classifier for each individual, which is different from the traditional methods where a fixed threshold is employed for all individuals. Besides, the SMOTE technology is adopted to solve the class-imbalance problem. Our experiments show that the proposed method is effective by achieving a recognition rate of 99.17%.

1. Introduction

Finger vein recognition is a promising biometric recognition technology which verifies identities through finger vein patterns. Medical studies have shown that the finger vein pattern is unique and stable. In detail, the finger veins of an individual are different from the others', and even the veins captured from a single individual are quite different from one finger to another. Furthermore, the finger veins are also invariant for healthy adults.

Compared with fingerprints, finger veins are hard to be forged or stolen as they are hidden inside the fingers. The contactless captures of finger veins also ensure both convenience and cleanliness, and they are user-friendly. Furthermore, Finger veins are less affected by physiology and environment factors such as dry skin and dirt.

A typical finger vein recognition process is composed of the following four steps. Firstly, the finger vein images are obtained via the finger vein capturing devices. Secondly, the finger vein images are preprocessed. Thirdly, the features are extracted. Finally, the finger vein images are matched based on the extracted features.

The preprocessing procedure includes image enhancement, normalization, and segmentation. For image enhancement, Yang and Yan incorporated directional decomposition and Frangi filtering to enhance the image quality [1]. Yu et al. proposed an enhancement algorithm based on multi-threshold combination [2]. Yang and Yang introduced multi-channel Gabor filter to enhance the images and obtained better performance [3]. Finger vein segmentation is also a very important step, and there are some typical methods including line tracking [4], mean curvatures [5], and region growth-based feature [6]. A detailed description of these approaches is beyond the scope of this paper. However, a summary of these approaches with the typical references is provided in Table 1.

PCA is a popular linear dimensionality reduction and feature extraction technology. It has extensive applications in image processing. Wu and Liu extracted the PCA features and then trained a neural network for matching, which results in a high recognition rate [7]. Since PCA transforms the 2-dimensional image matrix to a 1-dimensional vector, the covariance matrix is always large in size and it is time intensive to obtain the projection matrix which is composed of the covariance matrix's eigenvectors. Yang et al. proposed 2DPCA to reduce the size of the covariance matrix and save time for computing projection matrices [8]. In order to represent the characteristics of the 2-dimensional images more accurately, Zhang and Zhou introduced $(2D)^2$ PCA which can reflect the information of the image in row and column directions, respectively, use less time to compute the projection matrix, and get better experimental results on face recognition [9].

Table 1: Methods for personal authentication using finger vein recognition.

References	Method	Database fingers × samples per each	Performance
[4]	Linetracking	339 × 2 images	EER: 0.145%
[5]	Mean curvature	125 × 9 images	EER: 0.25%
[13]	Wide line detector	10, 140 × 5 images	EER: 0.87%
[14]	Statistical vein energy	100 × 10 images	CCR: 98.7%
[15]	Moment invariants	50 × 4 images	EER: 8.93%
[16]	Sliding window matching	76 × 6 images	EER: 0.54%
[10]	Manifold learning	164 × 70 images	EER: 0.8%
[7]	PCA + BP network	10 × 10 images	CCR: 99%
[11]	PCA + LDA + SVM	10 × 10 images	CCR: 98%

Recently, more and more researchers apply machine learning methods to finger vein recognition. Liu et al. introduced manifold learning to finger vein recognition [10]. Wu and Liu used PCA and LDA to extract features and train a SVM model for recognition [11]. Measuring the distance of the two samples is the premise of machine learning. For example, KNN requires a distance metric to find the neighbors of the target instance and then conducts classification or regression based on the distance metric. Typical distance metrics, such as Euclidean distance, make significant contribution in some application domains. In some conditions, these metrics cannot satisfy the assumption that the distances between instances from the same class are small while those from different classes are large. It limits the utilities of most machine learning methods.

There are two challenges for finger vein recognition: (1) how to efficiently extract distinguishing features and (2) how to design a strong classifier with high recognition rate and fast recognition speed to make the system more practical in real-world applications.

To overcome these two challenges, in this paper we apply $(2D)^2$ PCA to extract the features from finger vein images. In order to address the shortcoming of traditional distance-metric-based classifiers, we build a classifier for each individual based on metric learning. With regard to training samples of each classifier, the number of positive samples is inadequate as compared to the negative samples. Thus, we use SMOTE technology to oversample the positive samples to balance the two classes before training the classifier. The experimental results show that the proposed method has good performance on finger vein recognition.

The rest of this paper is organized as follows. The technical background is briefly introduced in Section 2. The proposed method is described in Section 3. Experimental results are provided in Section 4. Finally, this paper is concluded in Section 5.

2. Technical Background

2.1. $(2D)^2$ PCA. PCA is a typical linear dimensionality reduction and feature extraction method. Due to the transformation from the 2-dimensional image matrix into a 1-dimensional column vector, PCA often makes the size of the corresponding covariance matrix too large, and computing the eigenvectors and eigenvalues becomes time-consuming. In order to solve this problem, Yang et al. proposed 2DPCA to extract the features [8]. 2DPCA directly uses the image matrix to compute PCA features without transforming the 2-dimensional image matrix into a 1-dimensional column vector. Therefore, it reduces the size of corresponding covariance matrix and obtains the feature projection matrix with less time. However, 2DPCA works only for the row direction of images. To address the problem, Zhang and Zhou proposed $(2D)^2$ PCA which captures the image information from not only the row direction but also the column direction [9]. The experimental results show that $(2D)^2$ PCA outperforms 2DPCA and PCA in terms of both recognition rate and running time. The process of $(2D)^2$ PCA is described as follows.

Considering M finger vein images, which are denoted by $\mathbf{A}_1, \ldots, \mathbf{A}_M$, we compute the mean image matrix as $\overline{\mathbf{A}} = (1/M)\sum_j \mathbf{A}_j$ and the image covariance matrix $\mathbf{G}$ as

$$\mathbf{G} = \frac{1}{M}\sum_{j=1}^{M}\left(\mathbf{A}_j - \overline{\mathbf{A}}\right)^T\left(\mathbf{A}_j - \overline{\mathbf{A}}\right). \tag{1}$$

For a random image matrix $\mathbf{A}$, the key of obtaining the new features is to get the projection matrix $\mathbf{X} \in R^{n\times d}$, $n \geqslant d$. Then the new features are calculated as $\mathbf{Y} = \mathbf{AX}$. The total scatter of the projected samples is used to determine a good projection matrix $\mathbf{X}$, where the total scatter of the projected samples can be characterized by the trace of the covariance matrix of the projected feature vectors. From this point of view, we adopt the following criterion:

$$\begin{aligned} J(\mathbf{X}) &= \operatorname{trace}\left\{E\left[(\mathbf{Y} - E(\mathbf{Y}))(\mathbf{Y} - E(\mathbf{Y}))^T\right]\right\} \\ &= \operatorname{trace}\left\{E\left[(\mathbf{AX} - E(\mathbf{AX}))(\mathbf{AX} - E(\mathbf{AX}))^T\right]\right\} \\ &= \operatorname{trace}\left\{\mathbf{X}^T E\left[(\mathbf{A} - E(\mathbf{A}))^T(\mathbf{A} - E(\mathbf{A}))\right]\mathbf{X}\right\}. \end{aligned} \tag{2}$$

So,

$$J(\mathbf{X}) = \operatorname{trace}\left\{\mathbf{X}^T\mathbf{G}\mathbf{X}\right\}. \tag{3}$$

It has been proven that $J(\mathbf{X})$ gets the maximum when the projection matrix $\mathbf{X}$ is composed by the d orthonormal eigenvectors coupled to the d largest eigenvalues. In so

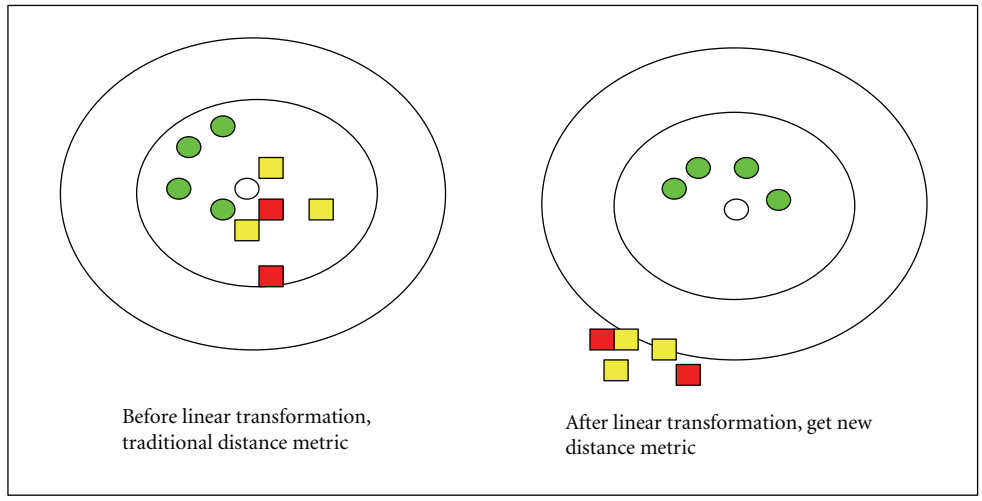

Figure 1: An example of LMNN.

saying, $\mathbf{X}$ obtains the optional value, and d can be controlled by setting a threshold as follows:

$$\frac{\sum_{i=1}^{d}\lambda_i}{\sum_{j=1}^{n}\lambda_j} \succcurlyeq \theta, \tag{4}$$

where θ is a user-specific threshold and $\lambda_1, \lambda_2, \ldots, \lambda_n$ is the top-n largest eigenvalues of $\mathbf{G}$.

Because $\mathbf{X}$ only reflects the information in the row direction, Zhang and Zhou proposed alternative 2DPCA which reflects the information in the column direction and combines 2DPCA with alternative 2DPCA to obtain a new method called $(2D)^2$ PCA [9]. Here is the process of alternative 2DPCA.

Let the image matrix $\mathbf{A}_j = [(\mathbf{A}_j^{(1)})^T, \ldots, (\mathbf{A}_j^{(n)})^T]^T$, and the mean image matrix $\overline{\mathbf{A}} = [(\overline{\mathbf{A}}^{(1)})^T, \ldots, (\overline{\mathbf{A}}^{(n)})^T]^T$, where $\mathbf{A}_j^{(i)}$ and $\overline{\mathbf{A}}^{(i)}$, denote the ith row vector of $\mathbf{A}_j^{(i)}$ and $\overline{\mathbf{A}}^{(i)}$ respectively. The image covariance matrix can be rewritten as

$$\mathbf{G}_1 = \frac{1}{M}\sum_{j=1}^{M}\sum_{k=1}^{m}\left(\mathbf{A}_j^{(k)} - \overline{\mathbf{A}}^{(k)}\right)^T\left(\mathbf{A}_j^{(k)} - \overline{\mathbf{A}}^{(k)}\right). \tag{5}$$

Similarly, to achieve the projected matrix $\mathbf{X}$ in 2DPCA, we can obtain the projection matrix $\mathbf{Z} \in R^{m\times q}$ from (2) and (5). We can also compute q in the same manner as we compute d in 2DPCA.

Using the projected matrix $\mathbf{X}$, $\mathbf{Z}$ in 2DPCA and alternative 2DPCA, respectively, we can obtain the new feature

$$\mathbf{C} = \mathbf{Z}^T\mathbf{A}\mathbf{X}. \tag{6}$$

We can see from (6) that the new feature $\mathbf{C}$ reflects more information of the image than the features obtained by 2DPCA and alternative 2DPCA. Furthermore, the dimension of $\mathbf{C}$ is smaller, and thus $(2D)^2$ PCA costs less time than 2DPCA and alternative 2DPCA for image processing.

2.2. Metric Learning. Most machine learning methods use distance metrics to measure the dissimilarity of instances. Metric learning is able to learn an appropriate distance metric. The main task of the metric learning is to find a better distance metric, based on which the distances between the samples from same class become small while those from different classes become large. This helps to improve the performance of the machine learning methods.

To overcome the shortage of the KNN classifier using Euclidean distance, Weinberge et al. proposed a metric learning method called LMNN (Large Margin Nearest Neighbor) [12] which learns a distance metric to improve the performance of KNN classifiers. The metric is obtained by learning a linear transformation matrix $\mathbf{L}$. With this distance metric, the distance between the same-class instances becomes smaller, and they are separated from the other instances by a large margin. The details are as follows.

Let $\mathbf{X}_i \in R^d$ $(i = 1, \ldots, n)$ denote the feature vector of training instances and let y_i be the corresponding label. The essence of LMNN is to obtain a new distance $D(x_i, x_j) = \|\mathbf{L}(x_i - x_j)\|^2 = (x_i - x_j)^T\mathbf{L}^T\mathbf{L}(x_i - x_j)$ after learning a linear transformation $\mathbf{L}$ matrix. With this distance metric, the distance between the instance and its k nearest neighbors will be minimized and the distance between the instances in different classes will be larger. Figure 1 shows an example of LMNN.

In Figure 1, green circles denote instances from the first class, yellow squares denote instances from the second class, and red squares denote instances from the third class. Consider the instance denoted by the white circle, which is treated as a test instance from the first class, in our following analysis. Based on Euclidean distance, we find 4 nearest neighbors, and this test instance is misclassified into the second class. However, using the LMNN-learned metric; this instance is separated from the second and the third instances. The distance between this instance and its neighbor is small. Now it is correctly classified into the first class.

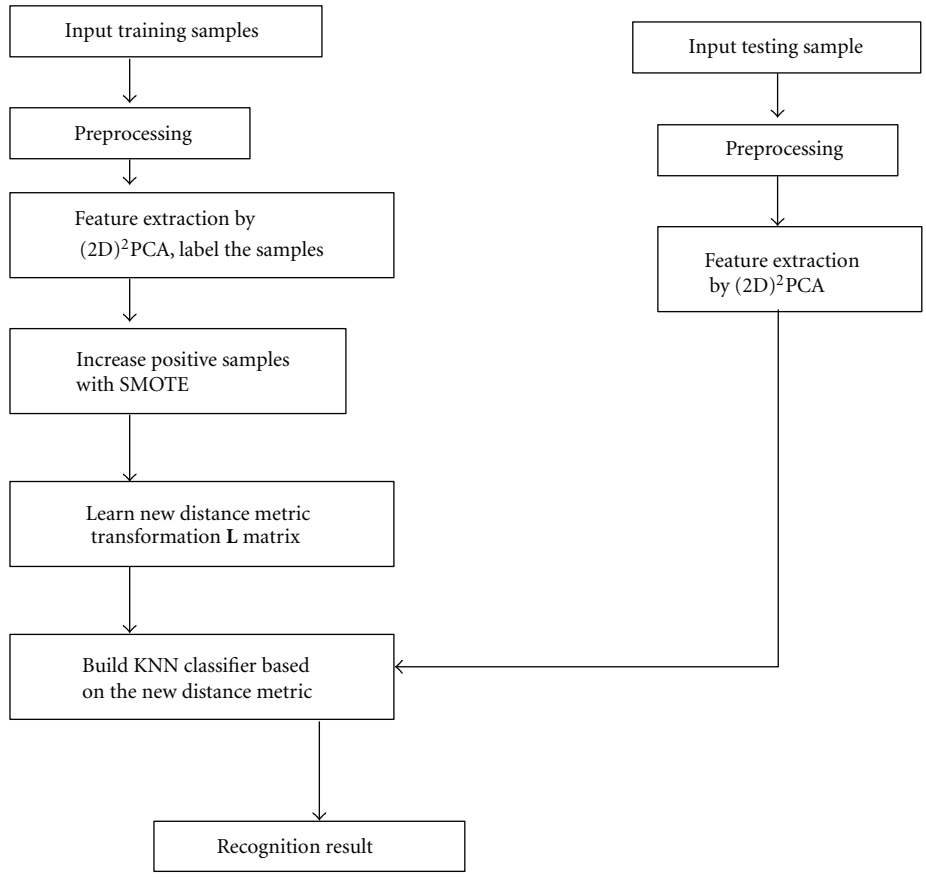

Figure 2: The proposed framework for finger vein recognition.

2.3. SMOTE. The performance of machine learning algorithms is typically evaluated by prediction accuracy. However, this is not applicable when the data is imbalanced. Existing solutions to the class imbalance problem can be divided into two categories. One is to assign distinct costs to training examples. The other is to resample the original dataset, either by oversampling the minority class and/or undersampling the majority class.

Chawla et al. proposed an oversampling approach called SMOTE where the minority class is oversampled by creating "synthetic" examples [17]. The minority class is oversampled by taking each minority class sample and introducing synthetic examples along the line segments joining any/all of the k minority class nearest neighbors. Depending upon the amount of oversampling required, neighbors from the k nearest neighbors are randomly chosen.

3. The Proposed Method

The proposed method includes training process and recognition process. As shown in Figure 2, a classifier is built for each individual, and the samples from a certain individual are treated as positive and others are negative. In the verification mode, we input a test sample to corresponding classifier to verify whether the sample comes from this individual based on the classification result. In the identification mode, we input a test sample to every classifier and identify which individual this sample belongs to.

In the training process, it is necessary to preprocess the infrared images of the finger veins. Preprocessing includes grayscale, ROI selection, and normalization (e.g., size normalization and gray normalization). After the preprocessing, we apply $(2D)^2$ PCA to extract the features of the training samples. Then we label the samples as positive and negative class accordingly and oversample the positive samples with SMOTE. We learn a new distance metric, that is, the transformation matrix L, with LMNN. Finally, we build the individual KNN classifier based on this new distance metric.

The preprocessing and feature extraction in the recognition process are similar to that in the training process. After that, we input the features of the samples to train classifier to verify the individual based on the classification result.

3.1. Preprocessing. The preprocessing includes image grayscale, ROI selection, size normalization, and gray normalization.

3.1.1. Image Grayscale. The original image (an example is shown in Figure 3(a)) is a 24-bit color image with a size of 320×240. In order to reduce the computational complexity, we transform the original image to an 8-bit image based on the gray-scale equation $Y = R * 0.299 + G * 0.588 + B * 0.114$, where R, G, and B denote the value of red, green, and

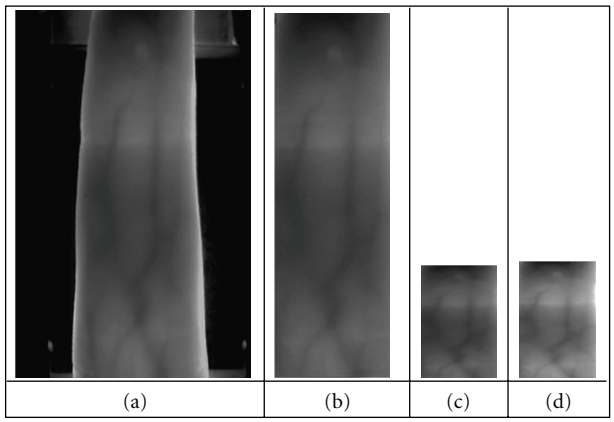

Figure 3: Examples of preprocessing.

blue. These three color components are coded by 8 bits. Y is the value of pixel after gray-scale transformation.

3.1.2. ROI Selection. As the background of finger vein region might include noise, we employ an edge-detection method to segment the finger vein region from the gray-scale image. A Sobel operator with a 3×3 mask $\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ is used for detecting the edges of fingers. The width of the finger region can be obtained based on the maximum and minimum abscissa values of the finger profile, and the height of the finger region is similarly detected. A rectangle region can be captured based on the width and height. This rectangle region is called ROI (as shown in Figure 3(b)).

3.1.3. Size Normalization. The size of the selected ROI is different from image to image due to personal factors such as different finger size and changing location. Therefore it is necessary to normalize the ROI region to the same size before the feature extraction process by $(2D)^2$ PCA. We use bilinear interpolation for size normalization in this paper, and the size of the normalized ROI is set to be $96 * 64$ (as shown in Figure 3(c)).

3.1.4. Gray Normalization. In order to extract efficient features, gray normalization is used to obtain a uniform gray distribution (as shown in Figure 3(d)). Formally, we have

$$p(i,j) = \frac{p'(i,j) - G_1}{G_2 - G_1}, \tag{7}$$

where $p'(i,j)$ is the pixel value of the original image, G_1 is the min pixel value of original image, G_2 is the max pixel value of original image, and $p(i,j)$ is the pixel value of image after gray normalization.

3.2. Training Process. After the preprocessing, we extract the features for each image by $(2D)^2$ PCA and assign labels for them. A classifier is trained for every individual, where the samples belonging to this individual are treated as positive and others are negative. We oversample the positive samples by SMOTE to obtain an augmented training set which achieves class balance in general. LMNN is then used on the augmented training set to obtain a transformation matrix **L**. With this new distance metric, a KNN classifier is built for classification.

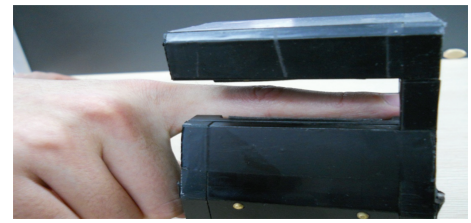

Figure 4: The finger vein capture device.

3.3. Recognition Process. In the verification mode, we input the feature vector of a test sample to a classifier which represents a certain individual, and then we verify whether the sample belongs to this individual based on the classification result. In the identification mode, we employ all classifiers to classify the test sample. If only a classifier C classifies it as positive class, this sample belongs to the individual which corresponds to the classifier C. If there are many classifiers classifying the sample as positive class, then we use the training accuracy rate for decision making: the sample belongs to the individual that corresponds to the classifier with the best training accuracy.

4. Experimental Result and Analysis

4.1. Database. The experiments were conducted using our finger vein database which is collected from 80 individuals' (including 64 males and 16 females, Asian race) index fingers of right hand, where each index finger contributes 18 finger vein images. Each individual participated in two sessions, separated by two weeks (14 days). The age of the participants was between 19 and 60 years, and their occupations included university students, professors, and workers at our school. The capture device was manufactured by the Joint Lab for Intelligent Computing and Intelligent System of Wuhan University, China, which is illustrated in Figure 4.

The original spatial resolution of the data is 320×240. After ROI extraction and size normalization, the size of the region used for feature extraction is reduced to 96×64. Samples collected from the same finger belong to the same class. Therefore, there are 80 classes, where each class contains 18 samples in our database. Some typical finger vein images are shown in Figure 5.

4.2. Experimental Settings. All the experiments are implemented with MATLAB and conducted on a machine with 2.4 G CPU and 4 G memory.

We design three experiments to verify the efficiency of the proposed method. In Experiment 1, we extract features by $(2D)^2$ PCA and then compare the classification performance of the metric-learning-based method and the

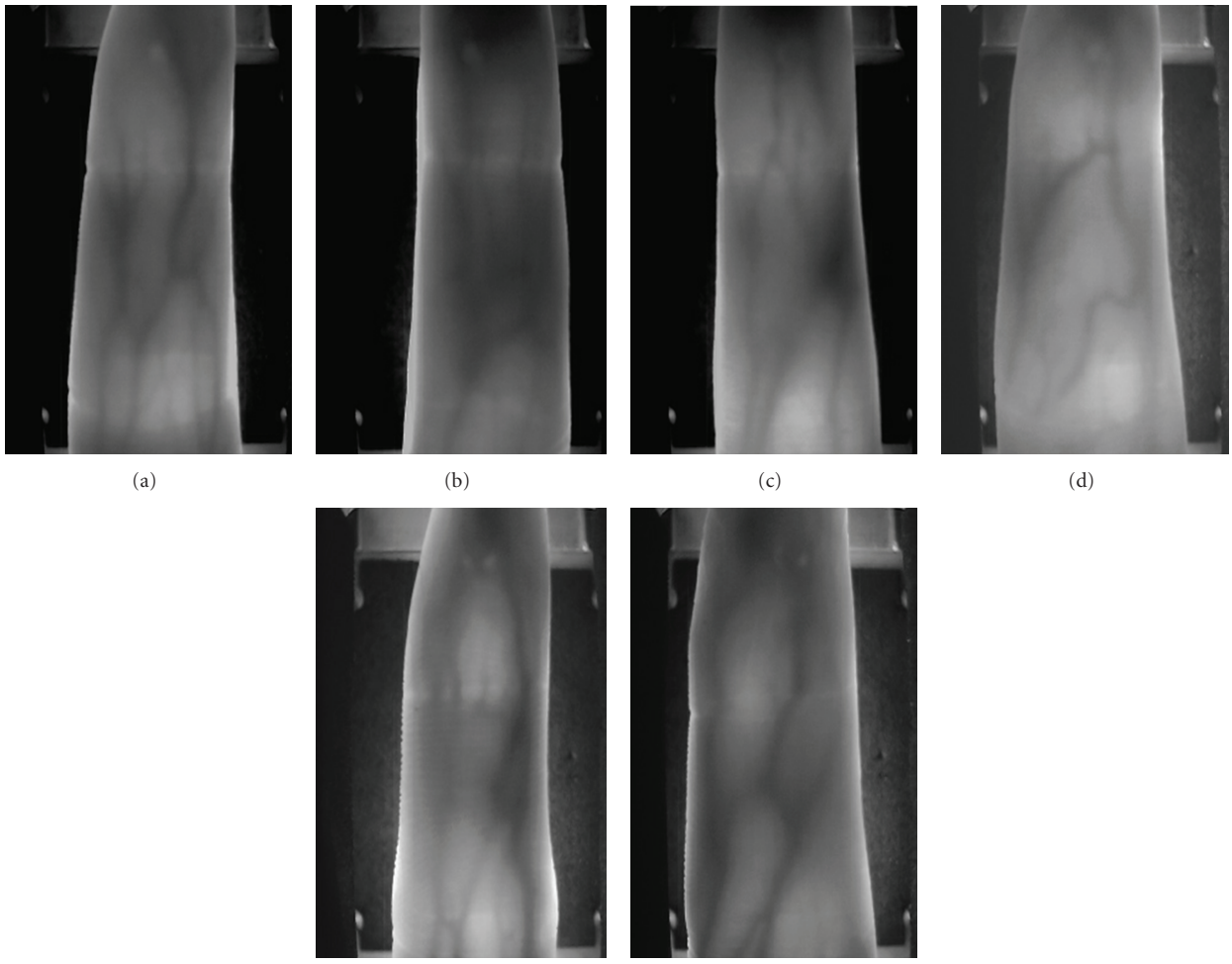

Figure 5: Sample finger vein images.

classic Euclidean-distance-based method. In Experiment 2, we compare the classification performance of KNN classifier combined with LMNN using different number of training samples. In Experiment 3, we employ the SMOTE technology to further boost the performance.

Experiment 1. In this experiment, we first generate four data sets as follows. We select 480, 720, 960, and 1200 images (i.e., 6, 9, 12, and 15 images for each individual) for training, and the rest of 960, 720, 480, and 240 images (i.e., 12, 9, 6, and 3 images for each individual) are left for testing, respectively. The Euclidean-distance-based recognition method works in the following way. We treat the training samples from each individual as the positive class and construct a center point for each class, where the ith feature of the center point is calculated by averaging the corresponding feature values of the training samples. As there are 80 individuals, we then obtain 80 center point $\overline{\mathbf{C}}_i$ $(i = 1, 2, \ldots, 80)$. For any testing sample $\mathbf{C}$, we estimate the Euclidean distances from sample $\mathbf{C}$ to each center point, $D(\mathbf{C}, \overline{\mathbf{C}}_i) = \|\mathbf{C} - \overline{\mathbf{C}}_i\|^2$ $(i = 1, 2, \ldots, 80)$. If $(\mathbf{C}, \overline{\mathbf{C}}_j) = \min D(\mathbf{C}, \overline{\mathbf{C}}_i)(i = 1, 2, \ldots, 80)$, then $\mathbf{C}$ goes to the jth class.

The metric-learning-based method works similarly as the Euclidean-distance-based method except for the usage of the learned distance metric $D(\mathbf{C}, \overline{\mathbf{C}}_i) = \|\mathbf{L}(\mathbf{C} - \overline{\mathbf{C}}_i)\|^2$ $(i = 1, 2, \ldots, 80)$. The recognition rates of these two methods are compared in Table 2.

It is clearly seen that the recognition rate of the metric-learning-based method is higher than the Euclidean-distance-based method. With distance metric transformation, two samples from different classes with small Euclidean distance are dragged farther. On the other hand, two samples from the same class with large Euclidean distance are pulled closer. Furthermore, the samples from different classes are separated by a large margin. Next we are going to provide an intuitive explanation based on the example shown in Figures 6 and 7.

These two figures show the data distribution of the data set with 480 training samples and 960 testing samples. We obtain 25 features for each sample using $(2D)^2$ PCA and select 2 features with the largest contribution to Euclidean distance metric. These two features constitute the vertical and horizontal coordinates of Figure 6. Similarly, these two

Table 2: The recognition rates of the compared methods.

	Euclidean-distance-based method	Metric-learning-based method
480 training, 960 testing	78.96%	86.46%
720 training, 720 testing	82.08%	91.25%
960 training, 480 testing	86.25%	92.29%
1200 training, 240 testing	84.58%	93.75%

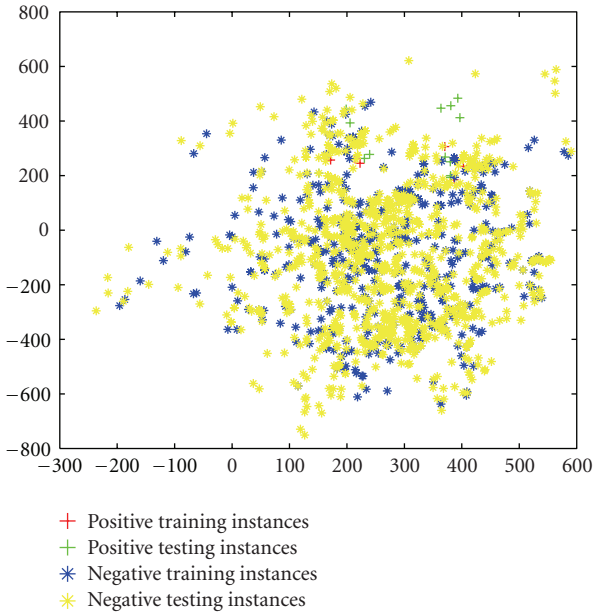

Figure 6: Samples distribution with Euclidean distance metric.

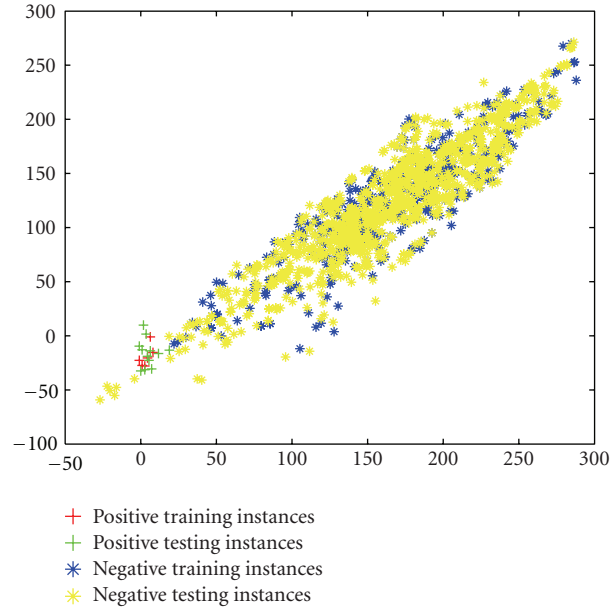

Figure 7: Samples distribution with new distance metric using LMNN.

features are transformed to a new metric space using LMNN, as shown in Figure 7. The samples of the first individual (including 6 training images and 12 testing images) are treated as positive, and the rest of them are considered as negative. In Figures 6 and 7, we use red plus to denote positive training samples, green plus for positive testing samples, blue star for negative training samples, and yellow star for negative testing samples. It is shown from Figure 6 that it is difficult to distinguish the first class from the others because the distances between samples in the first class and the other classes are indiscriminating. This inherent drawback of the Euclidean distance significantly reduces the recognition performance. However, by using LMNN, the samples in the first class are gathered together, as shown in Figure 7. In detail, the positive samples are located mainly in the area of abscissa value between 0 and 10. On the contrary, most negative samples are scattered out. This makes it easier to discriminate samples in the first class from samples in the other classes.

Experiment 2. In this experiment, we select 6, 9, 12, and 15 images from each individual as training samples to build a KNN classifier. The underlying distance metric for each individual is learned by LMNN. Here the number of neighbors, that is, k, is empirically set to be 3 in KNN. We obtain different recognition rates with different numbers of training samples, and the experimental results are shown in Figure 8.

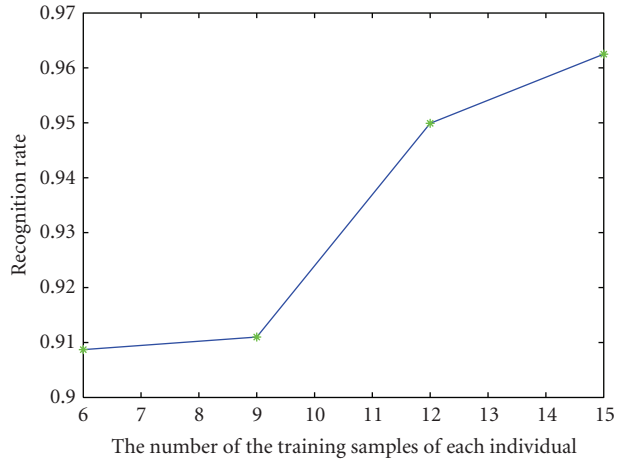

Figure 8: The recognition rates with different numbers of training images.

Overall, the recognition rate increases with the number of training images increases. When the number of the training images goes to 15, the recognition rate reaches 96.67%. It is also worth noting that, as compared to Table 2,

Table 3: The recognition rate by SMOTE.

Without SMOTE	96.67%
SMOTE-5	96.67%
SMOTE-10	96.67%
SMOTE-20	98.75%
SMOTE-30	98.33%
SMOTE-40	99.17%
SMOTE-50	99.17%

the KNN-based method outperforms the above-mentioned metric-learning-based method and the Euclidean-distance-based method, by considering the same number of training images.

Experiment 3. This experiment verifies that SMOTE can improve the classification performance. We select 1200 images (15 images for each individual) for training and 240 images (3 images for each individual) for testing. We use SMOTE to oversample the positive samples to be 5, 10, 20, 30, 40, and 50 times as large as the original set. The recognition result is shown in Table 3.

We observe that the recognition rate does not improve by only increasing a small number of synthetic positive samples, as shown in SMOTE-5 and SMOTE-10. After that, the recognition rate increases by about 3%, and finally it achieves 99.17% with SMOTE-40 or SMOTE-50. With a sufficiently large set of synthetic positive samples, the recognition performance would not improve any more.

5. Conclusion

This paper proposes a new finger vein recognition method based on $(2D)^2$ PCA and metric learning. Firstly, we extract features by $(2D)^2$ PCA and then train a binary classifier for each individual based on metric learning. Furthermore, we address the class imbalance problem by using SMOTE oversampling before the classifier is trained. The experimental results show that the proposed method achieves a recognition rate of 99.17%. The contributions of this paper are as follows. (1) We apply $(2D)^2$ PCA to extract features of finger vein image, where $(2D)^2$ PCA reflects the information in both the row direction and the column direction, and it is more efficient for feature extraction as compared to PCA and 2DPCA. (2) We build the KNN classifier based on metric learning using LMNN which changes the sample distribution in the new metric space. LMNN makes the distance between the samples from the same class smaller and the distance between the samples from different classes larger. Furthermore, we also employ a maximum margin framework to improve the recognition performance. This is incorporated with individually trained classifiers which reflect the characteristics of the corresponding individuals. (3) We note the class-imbalance problem; that is, when building the classifier for an individual, the number of the samples from the other individuals is considerably large. We tackle it by oversampling the positive samples with SMOTE, and the experimental results validate the effectiveness. Promising future work includes the exploration of features with better discrimination as well as the processing finger vein images of low quality.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant nos. 61173069 and 61070097, and the Research Fund for the Doctoral Program of Higher Education under Grant no. 20100131110021. The authors would like to thank Shuaiqiang Wang and Guang-Tong Zhou for their helpful comments and constructive advice on structuring the paper. In addition, the authors would particularly like to thank the anonymous reviewers for their helpful suggestions.

References

[1] J. Yang and M. Yan, "An improved method for finger-vein image enhancement," in *Proceedings of the IEEE 10th International Conference on Signal Processing (ICSP '10)*, pp. 1706–1709, Beijing, China, October 2010.

[2] C.-B. Yu, D.-M. Zhang, and H.-B. Li, "Finger vein image enhancement based on multi-threshold fuzzy algorithm," in *Proceedings of the 2nd International Congress on Image and Signal Processing (CISP '09)*, pp. 1–3, Tianjin, China, October 2009.

[3] J. F. Yang and J. L. Yang, "Multi-channel gabor filter design for finger vein image enhancement," in *Proceedings of the 5th International Conference on Image and Graphics (ICIG '09)*, pp. 87–91, Xi'an, China, September 2009.

[4] N. Miura, A. Nagasaka, and T. Miyatake, "Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification," *Machine Vision and Applications*, vol. 15, no. 4, pp. 194–203, 2004.

[5] W. Song, T. Kim, H. C. Kim, J. H. Choi, H. J. Kong, and S. R. Lee, "A finger-vein verification system using mean curvature," *Pattern Recognition Letters*, vol. 32, no. 11, pp. 1541–1547, 2011.

[6] Q. Huafeng, Q. Lan, and Y. Chengbo, "Region growth-based feature extraction method for finger vein recognition," *Optical Engineering*, vol. 50, no. 2, pp. 281–307, 2011.

[7] J. D. Wu and C. T. Liu, "Finger-vein pattern identification using principal component analysis and the neural network technique," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5423–5427, 2011.

[8] J. Yang, D. Zhang, A. F. Frangi, and J. Y. Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[9] D. Zhang and Z. H. Zhou, "$(2D)^2$ PCA: two-directional two-dimensional PCA for efficient face representation and recognition," *Neurocomputing*, vol. 69, no. 1–3, pp. 224–231, 2005.

[10] Z. Liu, Y. Yin, H. Wang, S. Song, and Q. Li, "Finger vein recognition with manifold learning," *Journal of Network and Computer Applications*, vol. 33, no. 3, pp. 275–282, 2010.

[11] J. D. Wu and C. T. Liu, "Finger-vein pattern identification using SVM and neural network technique," *Expert Systems with Applications*, vol. 38, no. 11, pp. 14284–14289, 2011.

[12] K. Weinberge, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS '06)*, 2006.

[13] B. Huang, Y. Dai, R. Li, D. Tang, and W. Li, "Finger-vein authentication based on wide line detector and pattern normalization," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR '10)*, pp. 1269–1272, Istanbul, Turkey, August 2010.

[14] J. Yang and X. Li, "Efficient finger vein localization and recognition," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR '10)*, pp. 1148–1151, Istanbul, Turkey, August 2010.

[15] X. Qian, S. Guo, X. Li, F. Zhong, and X. Shao, "Finger-vein recognition based on the score level moment invariants fusion," in *Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE '09)*, pp. 1–4, Wuhan, China, December 2009.

[16] C. Liukui and Z. Hong, "Finger vein image recognition based on tri-value template fuzzy matching," in *Proceedings of the 9th WSEAS International Conference on Multimedia Systems and Signal Processing (MUSP '09)*, pp. 206–211, Hangzhou, China, May 2009.

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.