Minutiae Detection Through Classifier Fusion and Clustering

Benjamin P. Carlson(2), George Bebis(1,2), and Carl Looney(2)
(1)Computer Vision Laboratory
(2)Department of Computer Science
University of Nevada, Reno
{benjamin,bebis,looney}@cs.unr.edu

correspondence should be addressed to:
Dr. George Bebis
Phone: (775) 784-6463
Fax: (775) 784-1877
Email: bebis@cs.unr.edu

Abstract: *Despite recent advances in the area of fingerprint identification, fingerprint matching continues to be a challenging pattern recognition problem. The first step to this problem is the extraction of landmarks known as minutiae points from a print. Once extracted, these points are then compared to all sets on file in search of a match. The accurate extraction of minutiae from an image is the basis for the entire matching process. Various minutiae extraction approaches have been proposed in the literature, each with its own merits and degree of success. The most common approach is to extract the ridges in the fingerprint image through skeletonization, apply ridge-following, and use rule-based classification for minutiae detection. Our emphasis in this paper is on extracting the minutiae from the original gray-scale images, without any image preprocessing. In particular, we have implemented and compared three methods based on eigenspace representations and neural network classifiers. Moreover, we present preliminary results of an attempt to fuse the outputs of these three methods using a clustering algorithm unique to this type of problem.*

Keywords: fingerprint minutiae extraction, eigenspaces, neural networks, classifier fusion.

1. Introduction

Due to steady increases in computing power and the advent of unobtrusive, easy-to-use fingerprint sensors, fingerprints are used more frequently as a biometric (identification based on a physiological or behavioral characteristic) for identification and recognition [1][6]. Since fingerprints are unique, even between identical twins, they are perfect for various security uses. The primary technique for matching newly-acquired prints is the extraction and matching of landmarks known as minutiae. Minutiae are areas where the ridges of the print either terminate to form a ridge ending, or split into two new ridges, forming a ridge bifurcation [2] (see Figure 1 (a-d)).

Numerous procedures have been proposed for the extraction of minutiae points from fingerprint images. Most of them, however, involve extensive pre-processing of the fingerprint image. A common approach for example, involves the thinning of the ridges, also known as skeletonization or ridge extraction [3] [4]. The process of ridge extraction requires extensive pre-processing, which is time consuming. In this paper, we compare four techniques which we apply without any pre-processing of the fingerprint images. Two representation schemes are considered, the first based on raw data and the second based on eigenspaces. Each technique improves upon the previous, and the final approach combines the results of the initial three in an attempt to classify the data through clustering. Each technique classifies an extracted frame into one of three groups: (1) a ridge ending, (2) a bifurcation, or (3) a plain ridge (or simply, no minutia point). All four techniques and the data used in our experiments are described below. First, we present a brief overview of the theory of eigenspace representations that form the basic representation scheme for two of our methods.

2. Review on Eigenspace Resepresentations

Eigenspace representations of images use Principal Components Analysis (PCA) [5] to linearly project an image in a low-dimensional space. This space is spanned by the principal components (i.e., eigenvectors corresponding to the largest eigenvalues) of the distribution of the training images. After an image has been projected in the eigenspace, a feature vector containing the coefficients of the projection is used to represent the image. We refer to these features as eigen-features. Here, we just summarize the main ideas. Representing each image $I(x,y)$ as an NxN vector Γ_i , first, the average face Ψ is computed:

$$\Psi = \frac{1}{R}\sum_{i=1}^{R}\Gamma_i$$

where R is the number of faces in the training set. Next, the difference Φ of each face from the average face is computed: $\Phi_i = \Gamma_i - \Psi$. Then the covariance matrix is estimated by:

$$C = \frac{1}{R}\sum_{i=1}^{R}\Phi_i\Phi_i^T = AA^T ,$$

where, $A = [\Phi_1 \quad \Phi_2 \quad \cdots \quad \Phi_R]$. The eigenspace can then be defined by computing the eigenvectors m_i of C . Since C is very large (N^2xN^2) , computing its eigenvectors will be very expensive. Instead, we can compute v_i , the eigenvectors of $A^T A$, an RxR matrix. Then, m_i can be computed from v_i as follows:

$$m_i = \sum_{j=1}^{R} v_{ij}\Phi_j, \quad j = 1 \cdots R$$

Usually, we only need to keep a smaller number of eigenvectors R_k corresponding to the largest eigenvalues. Given a new image, Γ , we subtract the mean ($\Phi = \Gamma - \Psi$) and compute the projection:

$$\tilde{\Phi} = \sum_{i=1}^{R_k} w_i u_i ,$$

where $w_i = u_i^T\Phi$ are the coefficients of the projection. The projection coefficients allow us to represent images as linear combinations of the eigenvectors. It is well known that the projection coefficients define a compact image representation and that a given image can be reconstructed from its projection coefficients and the eigenvectors (i.e., basis). The eigenspace representation of images is very powerful and has been used in various applications such as image compression and face recognition.

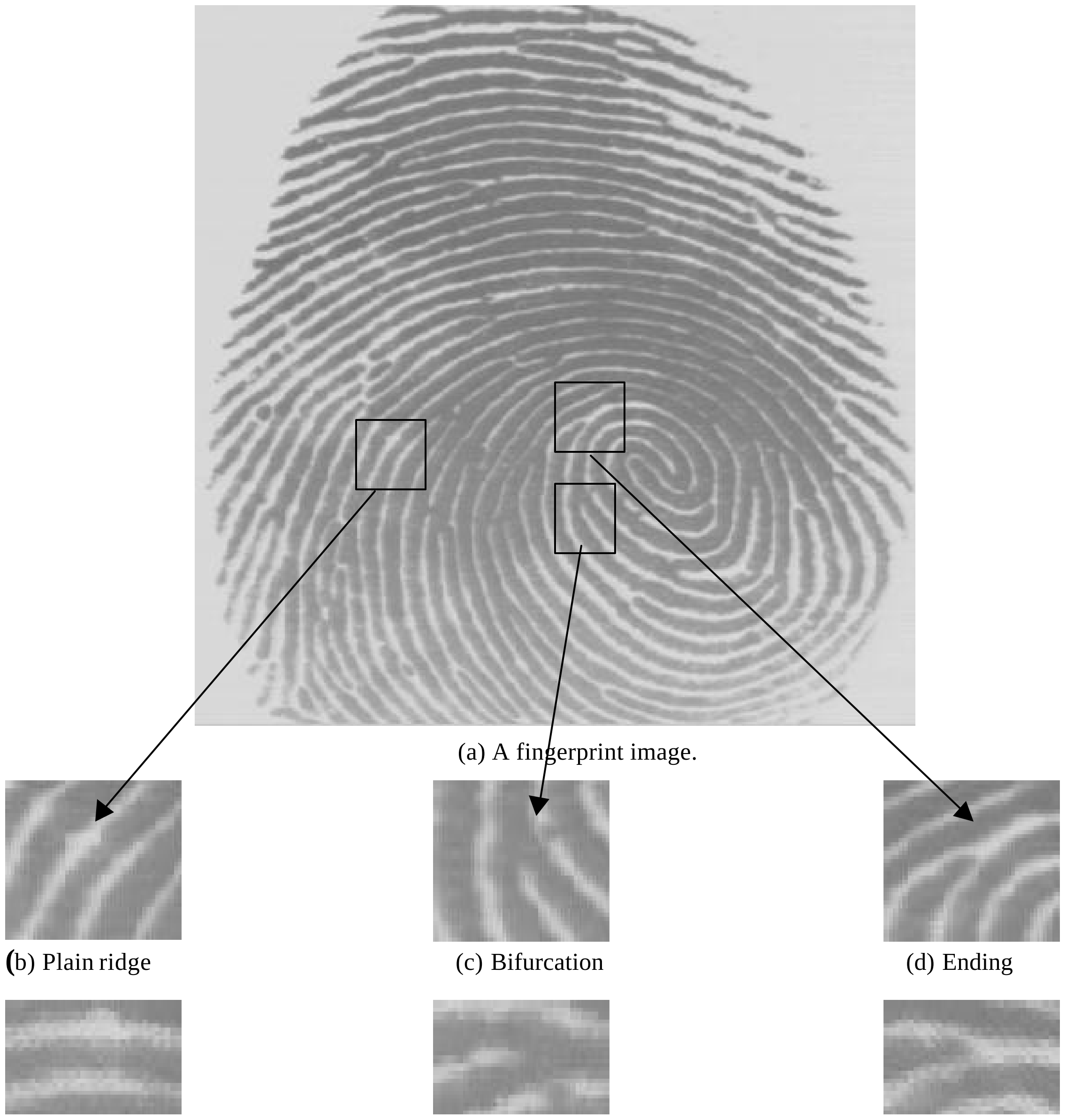
2. Methodology

Initially, we considered the eigenspace approach that has been used successfully for face recognition [5].
We used three different eigenspaces to represent (i) bifurcations, (ii) endings, and (iii) other. Classification was based on a minimum distance classifier. This approach produced reasonably good results. However, it was obvious that further improvements were possible. For comparison purposes, we fed the raw, standardized frames directly to a neural network. We were surprised to find that this simpler approach produced better results than the eigenspace technique. With this improvement, we then used PCA features which were fed to a neural network. This technique produced even better results than either of the prior two. However, the two neural network techniques produced results that were quite close. This led to the idea of combining the results of all three. Due to the overlap of many missed frames, we determined that a simple voting system would not suffice. We decided that a form of clustering may be the key to improvement over the best of these three algorithms.

3. Experiments and Results

3.1. Dataset and normalization

The dataset used in our experiments consists of 300 prints from ten people (10 prints from each). For evaluation purposes, we have manually extracted a large sample of frames (21x31 pixels each) from a subset of the fingerprints. Endings were extracted from 71 different prints from 27 people. Bifurcations came from 88 different prints from 27 people. Plain ridges came from 20 different prints from 14 people. Each of our extracted frames has the object of interest centered in it. We extracted a total of 820 endings, 715 bifurcations and 820 plain ridges for a total of 2,355 sample frames. These frames were then split into five groups in preparation for using five-fold cross-validation with 20% of the images in each test set (20% each of the three types), and 80% in each training set. Another set of 300 frames (100 each of endings, bifurcations and plain ridges) were used for validation of the neural networks.


(a) A fingerprint image.

(b) Plain ridge (c) Bifurcation (d) Ending

(e) Rotated plain ridge (f) Rotated bifurcation (g) Rotated ending

Figure 1. Examples of minutiae (b-d) and normalized minutiae (e-g).

When attempting to identify a minutia in a frame, one must orient the ridges in the same direction as the frames used to train the system. Many techniques perform this task by identifying flow or direction fields through applying gradients or other calculus-based techniques. We avoid the pre-processing these techniques require and simply determine the angle of rotation based on intensity variations along the two frame diagonals. This rotation algorithm requires that we start with a square frame, so we initially extract a frame of 38x38 pixels. Next, we calculate the intensity variations along the diagonals of this square frame. We compare the difference in these two values to calculate an angle of rotation. Obviously our technique has many additional details, but this description explains the basics of the algorithm. This technique works quite well and has proven more than adequate for our purposes. The examples in Figures 1(e-g) show the success of this algorithm.

For each of the initial three algorithms, we produced five results files, each containing results for 20% of the entire 2,355 frame set. These files were then combined to create a single file with the results of all three approaches applied to all 2,355 manually-extracted frames. Finally, this single file was clustered using our algorithm explained in section 3.5 below.

3.2. Using Eigenspace Representations

The eigenspace approach [5] was our model for this first attempt. We start by building three separate eigenspaces using 80% of the extracted frames, one for endings, one for bifurcations and one for plain ridges. Each eigenspace consists of 20 eigenimages which are the principal components of the three sets of extracted fingerprint frames. We retained a large variation of principal components for the eigenspaces (from 5 to 150) and found that 20 seems to work best (this approach will be referred to as ES20). After the three spaces have been constructed, we project each of the test images into each space, then reconstruct three images from these three projections. We compare each reconstructed image with its original by calculating the Euclidean distance between each, producing three distance measures. The reconstructed image with the smallest distance will determine the classification for this frame (either ending, bifurcation or plain ridge). Using the five data sets (described in the data section) our final classification errors were **2.8%** for endings, **9.23%** for bifurcations, and **5.73%** for plain ridges. The average error was **5.77%** (see Table 1 below).

Table 1: Overall errors for the four techniques tried.

	Endings	**Bifurcations**	**Plain Ridges**	**Average Error**
ES20	2.8	9.23	5.73	5.77
PCA99+NN	1.83	5.31	1.95	2.93
Raw+NN	1.6	5.31	2.4	3.01
Clustering	1.1	5.31	2.44	2.845

3.3. Using Raw Standardized Images and Neural Networks

We wished to compare our eigenspace results with a standard approach so we implemented a neural network classifier with the raw image frames after standardization to [0, 1] (the Stuttgart Neural Network Simulator, SNNS, version 4.1 was used for all neural network simulations). All trials implemented a feed-forward neural network using back-propagation with momentum to classify the test data. Many different combinations of network architecture and settings were tried. However, 651 input units (21x31 pixels=651), 40 units in the first hidden layer, 18 in the second hidden layer, and 2 output units with a learning rate of 0.1, momentum of 0.8 and flat spot elimination of 0.01 proved the best combination. Our results were very good for raw data using many input features. The final error rates (averaged over the five test sets as with the other results) were **1.6%** for endings, **5.31%** for bifurcations, and **2.4%** for plain ridges. The average error was 3.01% (refer to Table 1).

3.4. Using PCA and Neural Networks

Next, we extracted the principal components from the training sets using PCA, standardized the values to [0, 1], and trained a feed-forward neural network using back-propagation with momentum to classify the test data. We retained 90%, 95% and 99% of the components on three separate trials but 99% proved best. We also attempted a wide variety of network structures (different number of hidden layers and different nodes in each hidden layer) as well as a wide variety of settings for the learning rate and momentum. The best results were found using 365 input units, 50 units in the first hidden layer, 20 units in the second hidden layer, and 2 output units. The settings were 0.1 for the learning rate, 0.7 for momentum and 0.01 for flat spot elimination. The final classification errors were **1.83%** for endings, **5.31%** for bifurcations, and **1.95%** for plain ridges. The average error was **2.93%** (refer to Table 1).

3.5. Classifier Fusion Using Clustering

We decided that some form of classification involving the results of the above three techniques could lower the error rate. After considering a simple voting approach, we decided that this would not help, since there was much overlap between the mis-classified frames in the three techniques. However, this overlap may not affect a clustering technique, since clustering would not operate on binary data but actual distance data produced by the other algorithms.

Each of the neural network approaches above produced two values between [0, 1] with ideal values being [1, 0] for an ending, [0, 1] a bifurcation and [0, 0] a plain ridge. The data from the ES20 approach, however, was in a three-number format with the upper value above 1,000. Due to this, we decided to standardize this data with the smallest of the 3*2355 values mapping to 0 and the largest mapping to 1. For the ES20 data a value of [0, 1, 1] was the ideal for an ending, [1, 0, 1] for a bifurcation and [1, 1, 0] for a plain ridge. Thus, the data used for clustering consisted of seven values between [0, 1] for each original frame.

Initially we calculate a Euclidean distance matrix for all vectors. From this matrix, we find the distance from each vector to its closest neighbor. The average of these minimum distances is used in the first step for clustering of the vectors. We remove upper and lower outliers from the average minimum distance based on two numeric values, used as multipliers, from the user. We find that 3 for the upper and 2 for the lower work best (a larger value will cause more outliers to be included in the average minimum distance calculation). A third value entered by the user is used as a multiplier for the distance metric when it is determined that it must be increased in order for continued clustering (we find that 4 works well). After these matrices have been created and the distance metric calculated, we determine the best three vectors (or nodes) for seeds by determining which three have the largest minimum separation. These three will be used as the seeds for the first three clusters or classes. This approach successfully produces a seed node from each of the three groups we are attempting to cluster.

Next, we start the classification process. First we find the closest unclassified element to a classified element and, if it is within the average of the minimum distances, then it is included in this class. We continue this process until no new elements are classified. Then, we recalculate the average minimum distance using only elements that have not yet been classified (we again remove outliers from this calculation). If the new distance is not larger than the old one, then we multiply the old by the distance multiplier mentioned in the previous paragraph (this value must be larger than 1.0 to ensure that we do not get stuck in a loop), to ensure a larger distance metric for class inclusion. With this new distance, if no new classifications occur, we create a new class using the unclassified element that is the farthest from any classified element. For our specific problem of minutiae clustering, this procedure for the creation of new classes should never occur, since we start off with three classes. Next, we recalculate the distance measure but allow the possibility of the new distance to be smaller than the previous, so that the new group will be more likely to obtain members than the current groups. This process continues until all elements are classified. Even though new classes could have been created, none were. The three starting classes represented by the initial three seeds remained the only groups for the entire process. This situation is ideal, since we knew in advance that there were only three classes. This fact validates our algorithm's ability to determine the correct number of classes.

When all vectors have been included in a group, the classification of each group is determined by a simple check of the average of two of its seven values. Each of the two values from the raw neural network approach are averaged together for each group. Their Euclidean distance from the three ideal classes (ending=[1, 0], bifurcation=[0, 1], plain ridge=[0, 0]) determines the entire groups classification. The final results show a slight improvement over the best of the previous three approaches. The final errors were **1.01%** for endings, **5.31%** for bifurcations, and **2.44%** for plain ridges. The average error was **2.845%** (see Table 1).

4. Conclusions

We have considered the problem of minutiae detection without any preprocessing of the fingerprint images. Several different techniques were implemented and compared in this study. The first was based on eigenspace representations using a minimum-distance classifier. The second and third techniques used a neural network classifier with (i) raw data and (ii) eigen-features. The last technique, which produced the best results, was based on classifier fusion using clustering. While the improvement from the best of the three individual approaches (PCA99+NN) to the clustering approach was not great, there was a large decrease in the mis-classification of endings (from 1.83 to 1.1). This shows that the clustering approach was able to make an improvement over the best of the individual approaches for at least one cluster. This approach to classification shows great promise and will be further investigated.

Acknowledgements: This research was supported by a grant for undergraduate research from the UNR Office of Academic Affairs.

References

[1] Pankanti, S., Bolle, R. M., and Jain, A., "Biometrics: The Future of Identification". *IEEE Computer magazine*, February, 2000.

[2] Jain, A., and Pankanti, S., "Fingerprint Classification and Matching". *Handbook for Image and Video Processing*, A. Bovik (ed.), Academic Press, April 2000.

[3] Farina, A., Kovacs-Vajna, Z. M., and Leone, A., "Fingerprint Minutiae Extraction from Skeletonized Binary Images". *Pattern Recognition*, Vol. 32, No. 5, pp. 877-889, 1999.

[4] Ratha, N. K., Chen, S., and Jain, A. K., "Adaptive Flow Orientation Based Feature Extraction in Fingerprint Images". *Pattern Recognition*, Vol. 28, No. 11, pp. 1657-1672, 1995.

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

[6] Bebis G., Deaconu T., and Georgiopoulos M., "Fingerprint Identification Using Delaunay Triangulation". *IEEE International Conference on Intelligence, Information, and Systems (ICIIS)*, pp. 452-459, 1999.