On-Line Fingerprint Verification

Anil Jain, *Fellow, IEEE*, Lin Hong, and Ruud Bolle, *Fellow, IEEE*

Abstract—Fingerprint verification is one of the most reliable personal identification methods. However, manual fingerprint verification is so tedious, time-consuming, and expensive that it is incapable of meeting today's increasing performance requirements. An automatic fingerprint identification system (AFIS) is widely needed. It plays a very important role in forensic and civilian applications such as criminal identification, access control, and ATM card verification. This paper describes the design and implementation of an on-line fingerprint verification system which operates in two stages: minutia extraction and minutia matching. An improved version of the minutia extraction algorithm proposed by Ratha et al., which is much faster and more reliable, is implemented for extracting features from an input fingerprint image captured with an on-line inkless scanner. For minutia matching, an alignment-based elastic matching algorithm has been developed. This algorithm is capable of finding the correspondences between minutiae in the input image and the stored template without resorting to exhaustive search and has the ability of adaptively compensating for the nonlinear deformations and inexact pose transformations between fingerprints. The system has been tested on two sets of fingerprint images captured with inkless scanners. The verification accuracy is found to be acceptable. Typically, a complete fingerprint verification procedure takes, on an average, about eight seconds on a SPARC 20 workstation. These experimental results show that our system meets the response time requirements of on-line verification with high accuracy.

Index Terms—Biometrics, fingerprints, matching, verification, minutia, orientation field, ridge extraction.

1 Introduction

FINGERPRINTS are graphical flow-like ridges present on human fingers. They have been widely used in personal identification for several centuries [11]. The validity of their use has been well established. Inherently, using current technology fingerprint identification is much more reliable than other kinds of popular personal identification methods based on signature, face, and speech [11], [3], [15]. Although fingerprint verification is usually associated with criminal identification and police work, it has now become more popular in civilian applications such as access control, financial security, and verification of firearm purchasers and driver license applicants [11], [3]. Usually, fingerprint verification is performed manually by professional fingerprint experts. However, manual fingerprint verification is so tedious, time-consuming, and expensive that it does not meet the performance requirements of the new applications. As a result, automatic fingerprint identification systems (AFIS) are in great demand [11]. Although significant progress has been made in designing automatic fingerprint identification systems over the past 30 years, a number of design factors (lack of reliable minutia extraction algorithms, difficulty in quantitatively defining a reliable match between fingerprint images, fingerprint classification, etc.) create bottlenecks in achieving the desired performance [11].

An automatic fingerprint identification system is concerned with some or all of the following issues:

- *Fingerprint Acquisition*: How to acquire fingerprint images and how to represent them in a proper format.
- *Fingerprint Verification*: To determine whether two fingerprints are from the same finger.
- *Fingerprint Identification*: To search for a query fingerprint in a database.
- *Fingerprint Classification*: To assign a given fingerprint to one of the prespecified categories according to its geometric appearance.

A number of methods are used to acquire fingerprints. Among them, the inked impression method remains the most popular. It has been essentially a standard technique for fingerprint acquisition for more than 100 years [3]. The first step in capturing an inked impression of a fingerprint is to place a few dabs of ink on a slab then rolling it out smoothly with a roller until the slab is covered with a thin, even layer of ink. Then the finger is rolled from one side of the nail to the other side over the inked slab which inks the ridge patterns on top of the finger completely. After that, the finger is rolled on a piece of paper so that the inked impression of the ridge pattern of the finger appears on the paper. Obviously, this method is time-consuming and unsuitable for an on-line fingerprint verification system. Inkless fingerprint scanners are now available which are capable of directly acquiring fingerprints in digital form. This method eliminates the intermediate digitization process of inked fingerprint impressions and makes it possible to build an on-line system. Fig. 1 shows the two inkless fingerprint scanners used in our verification system. Fingerprint images captured with the inked impression method and the inkless impression method are shown in Fig. 2.

- A. Jain and L. Hong are with the Pattern Recognition and Image Processing Laboratory, Department of Computer Science, Michigan State University, East Lansing, MI 48824. E-mail: {jain, honglin}@cps.msu.edu.
- R. Bolle is with the Exploratory Computer Vision Group, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. E-mail: bolle@watson.ibm.com.

Manuscript received Feb. 2, 1996; revised Oct. 21, 1996. Recommended for acceptance by B. Dom.
For information on obtaining reprints of this article, please send e-mail to: transpami@computer.org, and reference IEEECS Log Number P96113.

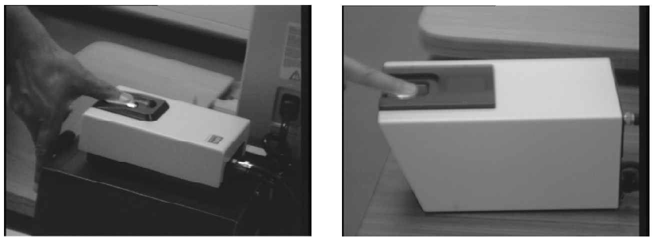


Fig. 1. Inkless fingerprint scanners: (a) Manufactured by Identix. (b) Manufactured by Digital Biometrics.

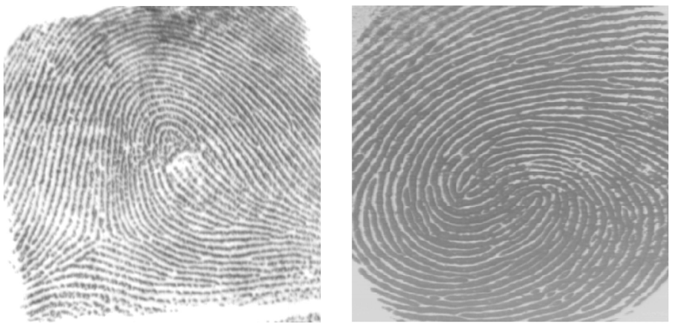


Fig. 2. Comparison of fingerprint images captured by different methods. (a) Inked impression method (from NIST database). (b) Inkless impression method (with a scanner manufactured by Digital Biometrics).

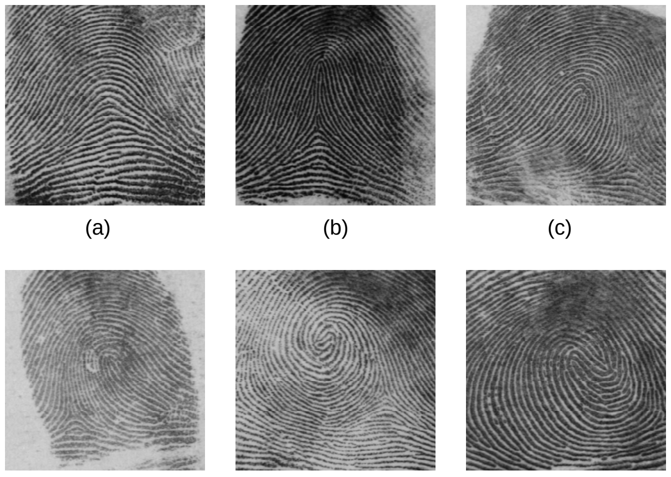


Fig. 3. A coarse-level fingerprint classification into six categories: (a) Arch. (b) Tented arch. (c) Right loop. (d) Left loop. (e) Whorl. (f) Twin loop.

The goal of fingerprint classification is to assign a given fingerprint to a specific category according to its geometric properties (Fig. 3 shows a coarse-level fingerprint classification). The main purpose of fingerprint classification is to facilitate the management of large fingerprint databases and to speedup the process of fingerprint matching. Generally, manual fingerprint classification is performed within a specific framework such as the well-known Henry system [3]. Different frameworks use different sets of properties. However, no matter what type of framework is used, the classification is based on ridge patterns, local ridge orientations and minutiae. Therefore, if these properties can be described quantitatively and extracted automatically from a fingerprint image then fingerprint classification will become an easier task. During the past several years, a number of researchers have attempted to solve the fingerprint classification problem [11], [3], [9], [10], [26]. Unfortunately, their efforts have not resulted in the desired accuracy. Algorithms reported in the literature classify fingerprints into five or six categories with about 90 percent classification accuracy on a medium size test set (several thousand images) [9], [10], [26]. However, to achieve a higher recognition accuracy with a large number of categories still remains a difficult problem.

Fingerprint verification determines whether two fingerprints are from the same finger or not. It is widely believed that if two fingerprints are from the same source, then their local ridge structures (minutia details) match each other topologically [11], [3]. Eighteen different types of local ridge descriptions have been identified [11]. The two most prominent structures are ridge endings and ridge bifurcations which are usually called minutiae. Fig. 4 shows examples of ridge endings and ridge bifurcations. Based on this observation and by representing the minutiae as a point pattern, an automatic fingerprint verification problem may be reduced to a point pattern matching (minutia matching) problem. In the ideal case, if

1) the correspondences between the template and input fingerprint are known,
2) there are no deformations such as translation, rotation and nonlinear deformations, etc. between them, and
3) each minutia present in a fingerprint image is exactly localized, then fingerprint verification consists of the trivial task of counting the number of spatially matching pairs between the two images.

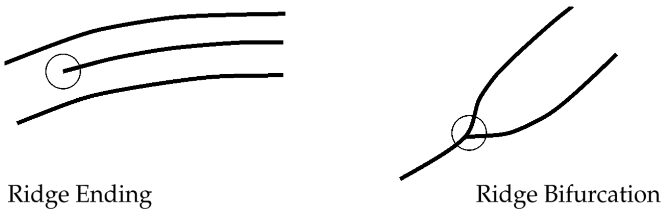


Fig. 4. Ridge ending and ridge bifurcation.

However, in practice

1) no correspondence is known beforehand,
2) there are relative translation, rotation and nonlinear deformations between template minutiae and input minutiae,
3) spurious minutiae are present in both templates and inputs, and
4) some minutiae are missed.

Therefore, in order for a fingerprint verification algorithm to operate under such circumstances, it is necessary to automatically obtain minutia correspondences, to recover deformations, and to detect spurious minutiae from fingerprint images. Unfortunately, this goal is quite difficult to achieve. Fig. 5 illustrates the difficulty with an example of two fingerprint images of the same finger.

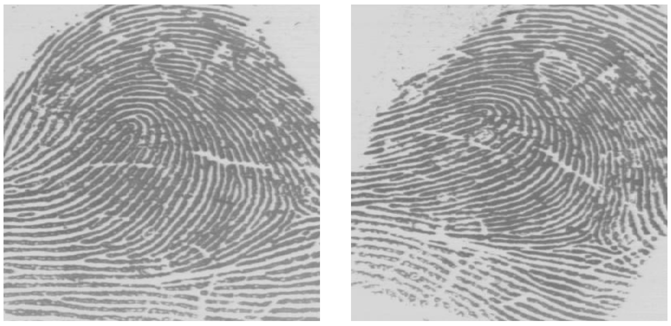


Fig. 5. Two different fingerprint images from the same finger. In order to know the correspondence between the minutiae of these two fingerprint images, all the minutiae must be precisely localized and the deformations must be recovered.

Fingerprint identification refers to the process of matching a query fingerprint against a given fingerprint database to establish the identity of an individual. Its goal is to quickly determine whether a query fingerprint is present in the database and to retrieve those which are most similar to the query from the database. The critical issues here are both retrieval speed and accuracy. In fact, this problem relates to a number of techniques studied under the auspices of computer vision, pattern recognition, database, and parallel processing. Operational fingerprint retrieval systems are being used by various law enforcement agencies [11].

In this paper, we will introduce an on-line fingerprint verification system whose purpose is to capture fingerprint images using an inkless scanner and to compare them with those stored in the database in "real time." Such a system has great utility in a variety of personal identification and access control applications. The overall block diagram of our system is shown in Fig. 6. It operates as follows:

1) Off-line phase: Several impressions (depending on the specification of the system) of the fingerprint of a person to be verified are first captured and processed by a feature extraction module; the extracted features are stored as templates in a database for later use;
2) On-line phase: The individual to be verified gives his/her identity and places his/her finger on the inkless fingerprint scanner, minutia points are extracted from the captured fingerprint image; these minutiae are then fed to a matching module, which matches them against his/her own templates in the database.

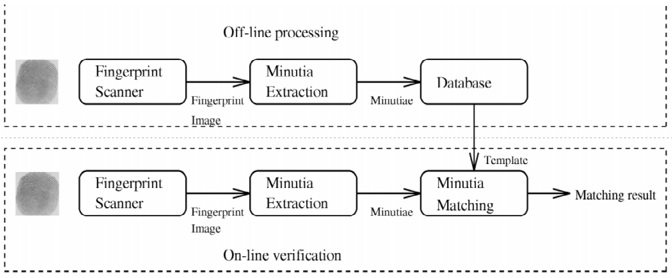


Fig. 6. Overview of our on-line fingerprint verification system.

The following two modules are the main components of our on-line fingerprint verification system:

- *Minutiae extraction*. Minutiae are ridge endings or ridge bifurcations. Generally, if a perfect segmentation can be obtained, then minutia extraction is just a trivial task of extracting singular points in a thinned ridge map. However, in practice, it is not always possible to obtain a perfect ridge map. Some global heuristics need to be used to overcome this limitation.
- *Minutia matching*. Minutia matching, because of deformations in sensed fingerprints, is an elastic matching of point patterns without knowing their correspondences beforehand. Generally, finding the best match between two point patterns is intractable even if minutiae are exactly located and no deformations exist between these two point patterns. The existence of deformations makes the minutia matching much more difficult.

For segmentation and minutia extraction, a modified version of the minutia extraction algorithm proposed in [18] is implemented which is much faster and more reliable for minutia extraction. We propose a hierarchical approach to obtain a smooth orientation field estimate of the input fingerprint image, which greatly improves the performance of minutia extraction. For minutia matching, we propose an alignment-based elastic matching algorithm. This algorithm is capable of finding the correspondences between minutiae without resorting to an exhaustive search and has the ability to adaptively compensate for the nonlinear deformations and inexact pose transformations between different fingerprints. Experimental results show that our system achieves excellent performance in a real environment.

In the following sections we will describe in detail our on-line fingerprint verification system. Section 2 mainly discusses the fingerprint feature extraction module. Section 3 presents our minutia matching algorithm. Experimental results on two fingerprint databases captured with two different inkless scanners are described in Section 4. Section 5 contains the summary and discussion.

2 MINUTIA EXTRACTION

It is widely known that a professional fingerprint examiner relies on minute details of ridge structures to make fingerprint identifications [11], [3]. The topological structure of the minutiae of a fingerprint is unique, and invariant with aging and impression deformations [11], [3]. This implies that fingerprint identification can be based on the topological structural matching of these minutiae. This reduces the complex fingerprint verification to minutia matching process which, in fact, is a sort of point pattern matching with the capability of tolerating, to some restricted extent, deformations of the input point patterns. Therefore, the first stage in an automatic fingerprint verification procedure is to extract minutiae from fingerprints. In our on-line fingerprint verification system, we have implemented a minutia extraction algorithm which is an improved version of the method proposed by Ratha et al. [18]. Its overall flowchart is depicted in Fig. 7. We assume that the resolution of input fingerprint images is 500 dpi.

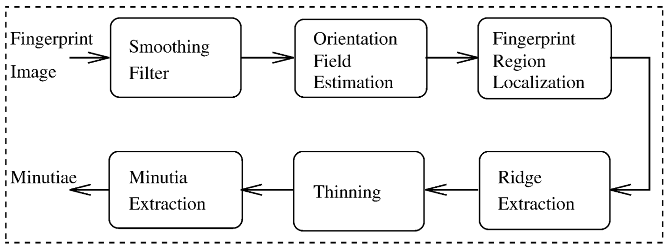


Fig. 7. Flowchart of the minutia extraction algorithm.

2.1 Estimation of Orientation Field

A number of methods have been proposed to estimate the orientation field of flow-like patterns [17]. In our system, a new hierarchical implementation of Rao's algorithm [17] is used. Rao's algorithm consists of the following main steps:

1) Divide the input fingerprint image into blocks of size $W \times W$.
2) Compute the gradients G_x and G_y at each pixel in each block.
3) Estimate the local orientation of each block using the following formula:

$$\theta_o = \frac{1}{2}\tan^{-1}\left(\frac{\sum_{i=1}^{W}\sum_{j=1}^{W} 2G_x(i,j)G_y(i,j)}{\sum_{i=1}^{W}\sum_{j=1}^{W}\left(G_x^2(i,j) - G_y^2(i,j)\right)}\right) \tag{1}$$

where W is the size of the block, and G_x and G_y are the gradient magnitudes in x and y directions, respectively.

The orientation field of a good quality fingerprint image can be reasonably estimated with this algorithm. However, the presence of high-curvature ridges, noise, smudges, and breaks in ridges leads to a poor estimate of the local orientation field. A postprocessing procedure needs to be applied to overcome this limitation. In our system, the following iterative steps are added to improve an inconsistent orientation field:

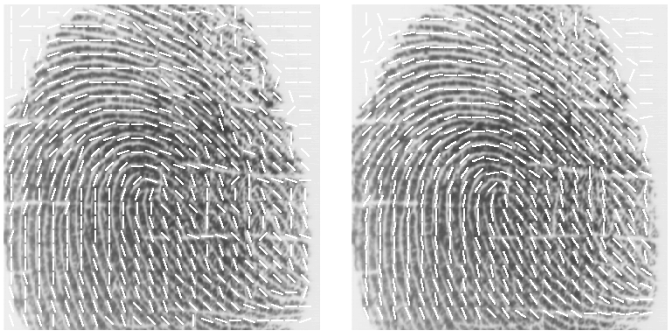
- Compute the *consistency level* of the orientation field in the local neighborhood of a block (i, j) with the following formula:

$$C_o = \frac{1}{N}\sqrt{\sum_{(i',j')\in D}\left|\theta(i',j') - \theta(i,j)\right|^2} \tag{2}$$

$$|\theta' - \theta| = \begin{cases} d & \text{if } \left(d = (\theta' - \theta + 360) \bmod 360\right) < 180 \\ d - 180 & \text{otherwise} \end{cases} \tag{3}$$

where D represents the local neighborhood around the block (i, j) (in our system, the size of D is 5×5); N is the number of blocks within D; $\theta(i',j')$ and $\theta(i,j)$ are local ridge orientations at blocks (i',j') and (i, j), respectively.

- If the *consistency level* (2) is above a certain threshold T_c, then the local orientations around this region are reestimated at a lower resolution level until it is below a certain level. With this post-smoothing scheme, a fairly smooth orientation field estimate can be obtained. Fig. 8 shows the orientation field of a fingerprint image estimated with our new algorithm.



(a) Rao's method. (b) Hierarchical method.

Fig. 8. Comparison of orientation fields by Rao's method and the proposed hierarchical method; the block size ($W \times W$) is 16×16 and the size of D is 5×5.

After the orientation field of an input fingerprint image is estimated, a segmentation algorithm which is based on the local variance of gray level is used to locate the region of interest from the fingerprint image. In our segmentation algorithm, we assume that there is only one fingerprint present in the image.

2.2 Ridge Detection

After the orientation field of the input image is estimated and the fingerprint region is located, the next step of our minutia exaction algorithm is ridge detection. The most salient property corresponding to ridges in a fingerprint image is the fact that gray level values on ridges attain their local maxima along the normal directions of local ridges. Therefore, pixels can be identified to be ridge pixels based on this property. In our minutia detection algorithm, a fingerprint image is first convolved with the following two masks, $h_t(x, y; u, v)$ and $h_b(x, y; u, v)$, of size $L \times H$ (on an average 11×7 in our system), respectively. These two masks are capable of adaptively accentuating the local maximum gray level values along the normal direction of the local ridge direction:

$$h_t(x,y;u,v) = \begin{cases} -\frac{1}{\sqrt{2\pi}\delta}e^{-\frac{u}{\delta^2}} & \text{if } u = \left(v\,\mathrm{ctg}\left(\theta(x,y)\right) - \frac{H}{2\cos\left(\theta(x,y)\right)}\right), v \in \Omega \\ \frac{1}{\sqrt{2\pi}\delta}e^{-\frac{u}{\delta^2}} & \text{if } u = \left(v\,\mathrm{ctg}\left(\theta(x,y)\right)\right), v \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$h_b(x, y; u, v) = \begin{cases} -\dfrac{1}{\sqrt{2\pi}\delta} e^{-\frac{u}{\delta^2}} & \text{if } u = \left(v \operatorname{ctg}(\theta(x, y)) + \dfrac{H}{2\cos(\theta(x, y))} \right), v \in \Omega \\ \dfrac{1}{\sqrt{2\pi}\delta} e^{-\frac{u}{\delta^2}} & \text{if } u = \left(v \operatorname{ctg}(\theta(x, y)) \right), v \in \Omega \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\Omega = \left[-\left| \frac{L \sin(\theta(x, y))}{2} \right|, \left| \frac{L \sin(\theta(x, y))}{2} \right| \right] \tag{6}$$

where $\theta(x, y)$ represents the local ridge direction at pixel (x, y). If *both* the gray level values at pixel (x, y) of the convolved images are larger than a certain threshold T_{ridge}, then pixel (x, y) is labeled as a ridge. By adapting the mask width to the width of the local ridge, this algorithm can efficiently locate the ridges in a fingerprint image.

However, due to the presence of noise, breaks, and smudges, etc. in the input image, the resulting binary ridge map often contains holes and speckles. When ridge skeletons are used for the detection of minutiae, the presence of such holes and speckles will severely handicap the performance of our minutia extraction algorithm because these holes and speckles may drastically change the skeleton of the ridges. Therefore, a hole and speckle removal procedure needs to be applied before ridge thinning.

After the above steps are performed on an input fingerprint image, a relatively smooth ridge map of the fingerprint is obtained. The next step of our minutia detection algorithm is to thin the ridge map and locate the minutiae.

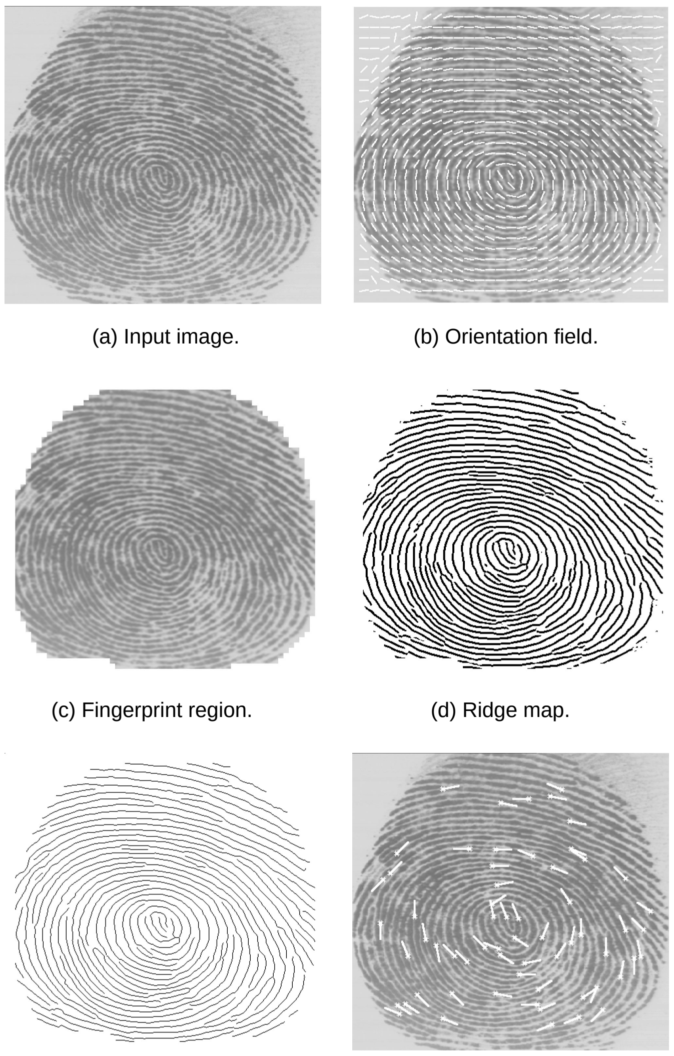
2.3 Minutia Detection

Minutia detection is a trivial task when an ideal thinned ridge map is obtained. Without a loss of generality, we assume that if a pixel is on a thinned ridge (eight-connected), then it has a value 1, and 0 otherwise. Let (x, y) denote a pixel on a thinned ridge, and $N_0, N_1, ..., N_7$ denote its eight neighbors. A pixel (x, y) is a ridge ending if $\left(\sum_{i=0}^{8} N_i\right) = 1$ and a ridge bifurcation if $\left(\sum_{i=0}^{8} N_i\right) > 2$. However, the presence of undesired spikes and breaks present in a thinned ridge map may lead to many spurious minutiae being detected. Therefore, before the minutia detection, a smoothing procedure is applied to remove spikes and to join broken ridges. Our ridge smoothing algorithm uses the following heuristics:

- If a branch in a ridge map is roughly orthogonal to the local ridge directions and its length is less than a specified threshold T_b, then it will be removed.
- If a break in a ridge is short enough and no other ridges pass through it, then it will be connected.

Although the above heuristics do delete a large percentage of spurious minutiae, many spurious minutiae still survive. The reason is that the above processing relies on local ridge information. If this information itself is unreliable, then the above heuristics have no way of differentiating false minutiae from true minutiae. Therefore, a refinement which is based on structural information is necessary. Our refinement algorithm eliminates the spurious minutiae based on the following rules:

- If several minutiae form a cluster in a small region, then remove all of them except for the one nearest to the cluster center.
- If two minutiae are located close enough, facing each other, but no ridges lie between them, then remove both of them.


(a) Input image. (b) Orientation field.

(c) Fingerprint region. (d) Ridge map.

(e) Thinned ridge map. (f) extracted minutiae.

Fig. 9. Results of our minutia extraction algorithm on a fingerprint image (512 × 512) captured with an inkless scanner. (a) Input image. (b) Orientation field superimposed on the input image. (c) Fingerprint region. (d) Extracted ridges. (e) Thinned ridge map. (f) Extracted minutiae and their orientations superimposed on the input image.

After the above refinement procedure is performed, the surviving minutiae are treated as true minutiae. Although the above heuristics can not ensure a perfect location of each minutia, they are able to delete several spurious minutiae. For each surviving minutia, the following parameters are recorded:

1) x-coordinate,
2) y-coordinate,
3) orientation which is defined as the local ridge orientation of the associated ridge, and
4) the associated ridge.

The recorded ridges are represented as one-dimensional discrete signals which are normalized by the average inter-ridge distance. These recorded ridges are used for alignment in the minutia matching phase. Fig. 9 shows the results of our minutia extraction algorithm on a fingerprint image captured with an inkless scanner.

3 Minutia Matching

Generally, an automatic fingerprint verification/identification is achieved with point pattern matching (minutiae matching) instead of a pixel-wise matching or a ridge pattern matching of fingerprint images. A number of point pattern matching algorithms have been proposed in the literature [23], [1], [21], [16]. Because a general point matching problem is essentially intractable, features associated with each point and their spatial properties such as the relative distances between points are often used in these algorithms to reduce the exponential number of search paths.

The relaxation approach [16] iteratively adjusts the confidence level of each corresponding pair based on its consistency with other pairs until a certain criterion is satisfied. Although a number of modified versions of this algorithm have been proposed to reduce the matching complexity [23], these algorithms are inherently slow because of their iterative nature.

The Hough transform-based approach proposed by Stockman et al. [22] converts point pattern matching to a problem of detecting the highest peak in the Hough space of transformation parameters. It discretizes the transformation parameter space and accumulates evidence in the discretized space by deriving transformation parameters that relate two point patterns using a substructure or feature matching technique. Karu and Jain [8] proposed a hierarchical Hough transform-based registration algorithm which greatly reduced the size of accumulator array by a multiresolution approach. However, if the number of minutia point is less than 30, then it is very difficult to accumulate enough evidence in the Hough transform space for a reliable match.

Another approach to point matching is based on energy minimization. This approach defines a cost function based on an initial set of possible correspondences and uses an appropriate optimization algorithm such as genetic algorithm [1] and simulated annealing [21] to find a possible suboptimal match. These methods tend to be very slow and are unsuitable for an on-line fingerprint verification system.

In our system, an alignment-based matching algorithm is implemented. Recognition by alignment has received a great deal of attention during the past few years [12], because it is simple in theory, efficient in discrimination, and fast in speed. Our alignment-based matching algorithm decomposes the minutia matching into two stages:

1) *Alignment stage*, where transformations such as translation, rotation and scaling between an input and a template in the database are estimated and the input minutiae are aligned with the template minutiae according to the estimated parameters; and
2) *Matching stage*, where both the input minutiae and the template minutiae are converted to polygons in the polar coordinate system and an elastic string matching algorithm is used to match the resulting polygons.

3.1 Alignment of Point Patterns

Ideally, two sets of planar point patterns can be aligned completely by two corresponding point pairs. A true alignment between two point patterns can be obtained by testing all possible corresponding point pairs and selecting the optimal one. However, due to the presence of noise and deformations, the input minutiae cannot always be aligned exactly with respect to those of the templates. In order to accurately recover pose transformations between two point patterns, a relatively large number of corresponding point pairs need to be used. This leads to a prohibitively large number of possible correspondences to be tested. Therefore, an alignment by corresponding point pairs is not practical even though it is feasible.

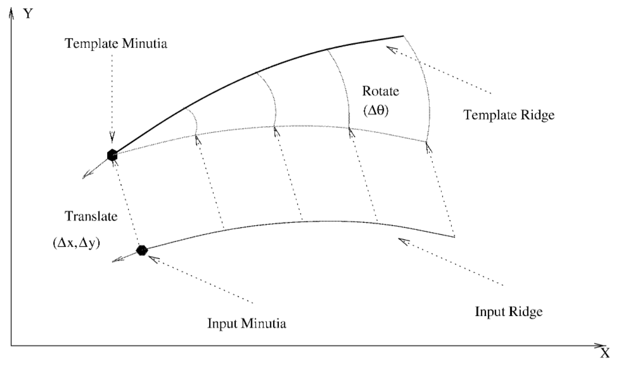


Fig. 10. Alignment of the input ridge and the template ridge.

It is well known that corresponding curve segments are capable of aligning two point patterns with a high accuracy in the presence of noise and deformations. Each minutia in a fingerprint is associated with a ridge. It is clear that a true alignment can be achieved by aligning corresponding ridges (see Fig. 10). During the minutiae detection stage, when a minutia is extracted and recorded, the ridge on which it resides is also recorded. This ridge is represented as a planar curve with its origin coincident with the minutia and its x-coordinate being in the same direction as the direction of the minutia. Also, this planar curve is normalized with the average inter-ridge distance. By matching these ridges, the relative pose transformation between the input fingerprint and the template can be accurately estimated. To be specific, let R^d and R^D denote the sets of ridges associated with the minutiae in input image and template, respectively. Our alignment algorithm can be described in terms of the following steps:

1) For each ridge $d \in R^d$, represent it as an one-dimensional discrete signal and match it against each ridge, $D \in R^D$ according to the following formula:

$$S = \frac{\sum_{i=0}^{L} d_i D_i}{\sqrt{\sum_{i=0}^{L} d_i^2 D_i^2}} \tag{7}$$

where L is the minimal length of the two ridges and d_i and D_i represent the distances from point i on the ridges d and D to the x-axis, respectively. The sampling interval on a ridge is set to the average inter-ridge distance. If the matching score S $(0 \le S \le 1)$ is larger than a certain threshold T_r, then go to step 2, otherwise continue to match the next pair of ridges.

2) Estimate the pose transformation between the two ridges (Fig. 10). Generally, a least-square method can be used to estimate the pose transformation. However, in our system, we observe that the following method is capable of achieving the same accuracy with less computation. The translation vector $(\Delta x, \Delta y)^T$ between the two corresponding ridges is computed by

$$\begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} = \begin{pmatrix} x^d \\ y^d \end{pmatrix} - \begin{pmatrix} x^D \\ y^D \end{pmatrix} \tag{8}$$

where $(x^d, y^d)^T$ and $(x^D, y^D)^T$ are the x and y coordinates of the two minutiae, which are called reference minutiae, associated with the ridges d and D, respectively. The rotation angle $\Delta\theta$ between the two ridges is computed by

$$\Delta\theta = \frac{1}{L}\sum_{i=0}^{L}\left(\gamma_i - \Gamma_i\right) \tag{9}$$

where L is the minimal length of the two ridges d and D; γ_i and Γ_i are radial angles of the ith point on the ridge with respect to the reference minutia associated with the two ridges d and D, respectively. The scaling factor between the input and template images is assumed to be one. This is reasonable, because fingerprint images are captured with the same device in both the off-line processing phase and the on-line verification phase.

3) Denote the minutia $(x^d, y^d, \theta^d)^T$, based on which the pose transformation parameters are estimated, as the reference minutia. Translate and rotate all the N input minutiae with respect to this reference minutia, according to the following formula:

$$\begin{pmatrix} x_i^A \\ y_i^A \\ \theta_i^A \end{pmatrix} = \begin{pmatrix} \Delta x \\ \Delta y \\ \Delta\theta \end{pmatrix} + \begin{pmatrix} \cos\Delta\theta & \sin\Delta\theta & 0 \\ \sin\Delta\theta & -\cos\Delta\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x_i - x^d \\ y_i - y^d \\ \theta_i - \theta^d \end{pmatrix} \tag{10}$$

where $(x_i, y_i, \theta_i)^T$, $(i = 1, 2, ..., N)$, represents an input minutia and $\left(x_i^A, y_i^A, \theta_i^A\right)^T$ represents the corresponding aligned minutia.

3.2 Aligned Point Pattern Matching

If two identical point patterns are exactly aligned with each other, each pair of corresponding points is completely coincident. In such a case, a point pattern matching can be simply achieved by counting the number of overlapping pairs. However, in practice, such a situation is not encountered. On the one hand, the error in determining and localizing minutia hinders the alignment algorithm to recover the relative pose transformation exactly, while on the other hand, our alignment scheme described above does not model the nonlinear deformation of fingerprints which is an inherent property of fingerprint impressions. With the existence of such a nonlinear deformation, it is impossible to exactly recover the position of each input minutia with respect to its corresponding minutia in the template. Therefore, the aligned point pattern matching algorithm needs to be elastic which means that it should be capable of tolerating, to some extent, the deformations due to inexact extraction of minutia positions and nonlinear deformations. Usually, such an elastic matching can be achieved by placing a bounding box around each template minutia, which specifies all the possible positions of the corresponding input minutia with respect to the template minutia, and restricting the corresponding minutia in the input image to be within this box [18]. This method does not provide a satisfactory performance in practice, because local deformations may be small while the accumulated global deformations can be quite large. We have implemented an adaptive elastic matching algorithm with the ability to compensate the minutia localization errors and nonlinear deformations.

Let

$$P = \left(\left(x_1^P, y_1^P, \theta_1^P\right)^T, \ldots, \left(x_M^P, y_M^P, \theta_M^P\right)^T\right)$$

denote the set of M minutiae in the template and

$$Q = \left(\left(x_1^Q, y_1^Q, \theta_1^Q\right)^T, \ldots, \left(x_N^Q, y_N^Q, \theta_N^Q\right)^T\right)$$

denote the set of N minutiae in the input image which is aligned with the above template with respect to a given reference minutia point. The steps in our elastic point pattern matching algorithm are given below:

1) Convert each minutia point to the polar coordinate system with respect to the corresponding reference minutia on which the alignment is performed:

$$\begin{pmatrix} r_i \\ e_i \\ \theta_i \end{pmatrix} = \begin{pmatrix} \sqrt{\left(x_i^* - x^r\right)^2 + \left(y_i^* - y^r\right)^2} \\ \tan^{-1}\left(\dfrac{y_i^* - y^r}{x_i^* - x^r}\right) \\ \theta_i^* - \theta^r \end{pmatrix} \tag{11}$$

where $\left(x_i^*, y_i^*, \theta_i^*\right)$ are the coordinates of a minutia,

$(x^r, y^r, \theta^r)^T$ are the coordinates of the reference minutia, and $(r_i, e_i, \theta_i)^T$ is the representation of the minutia in polar coordinate system (r_i represents the radial distance, e_i represents the radial angle, and θ_i represents the orientation of the minutia with respect to the reference minutia).

2) Represent the template and the input minutiae in the polar coordinate system as symbolic strings by concatenating each minutia in the increasing order of radial angles:

$$P_p = \left(\left(r_1^P, e_1^P, \theta_1^P\right)^T, \ldots, \left(r_M^P, e_M^P, \theta_M^P\right)^T\right) \quad (12)$$

$$Q_p = \left(\left(r_1^Q, e_1^Q, \theta_1^Q\right)^T, \ldots, \left(r_N^Q, e_N^Q, \theta_N^Q\right)^T\right) \quad (13)$$

where $\left(r_*^P, e_*^P, \theta_*^P\right)$ and $\left(r_*^Q, e_*^Q, \theta_*^Q\right)$ represent the corresponding radius, radial angle, and normalized minutia orientation with respect to the reference minutia, respectively.

3) Match the resulting strings P_p and Q_p with a dynamic-programming algorithm [4] to find the edit distance between P_p and Q_p which is described below.

4) Use the edit distance between P_p and Q_p to establish the correspondence of the minutiae between P_p and Q_p. The matching score, M_{pq}, is then computed according to the following formula:

$$M_{pq} = \frac{100 N_{pair}}{\max\{M, N\}} \quad (14)$$

where N_{pair} is the number of the minutiae which fall in the bounding boxes of template minutiae. The maximum and minimum values of the matching score are 100 and 1, respectively. The former value indicates a perfect match, while the later value indicates no match at all.

Minutia matching in the polar coordinate has several advantages. We have observed that the nonlinear deformation of fingerprints has a radial property. In other words, the nonlinear deformation in a fingerprint impression usually starts from a certain point (region) and nonlinearly radiates outward. Therefore, it is beneficial to model it in the polar space. At the same time, it is much easier to formulate rotation, which constitutes the main part of the alignment error between an input image and a template, in the polar space than in the Cartesian space. The symbolic string generated by concatenating points in an increasing order of radial angle in polar coordinate uniquely represents a point pattern. This reveals that the point pattern matching can be achieved with a string matching algorithm.

A number of string matching algorithms have been reported in the literature [4]. Here, we are interested in incorporating an elastic criteria into a string matching algorithm. Generally, string matching can be thought of as the maximization/minimization of a certain cost function such as the edit distance. Intuitively, including an elastic term in the cost function of a string matching algorithm can achieve a certain amount of error tolerance. Given two strings P_p and Q_p of lengths M and N, respectively, the edit distance, $C(M, N)$, in our algorithm is recursively defined with the following equations:

$$C(m,n) = \begin{cases} 0 & \text{if } m = 0, \text{or } n = 0 \\ \min\begin{Bmatrix} C(m-1,n)+\Omega \\ C(m,n-1)+\Omega \\ C(m-1,n-1)+w(m,n) \end{Bmatrix} & \begin{matrix} 0 < m \le M, \\ \text{and } 0 < n \le N \end{matrix} \end{cases} \quad (15)$$

$$w(m,n) = \begin{cases} \alpha\left|r_m^P - r_n^Q\right| + \beta\Delta e + \gamma\Delta\theta & \begin{matrix}\text{if } \left|r_m^P - r_n^Q\right| < \delta, \\ \Delta e < \epsilon \text{ and } \Delta\theta < \varepsilon \end{matrix} \\ \Omega & \text{otherwise} \end{cases} \quad (16)$$

$$\Delta e = \begin{cases} a & \text{if } \left(a = \left(e_m^P - e_n^Q + 360\right) \bmod 360\right) < 180 \\ a - 180 & \text{otherwise} \end{cases} \quad (17)$$

$$\Delta\theta = \begin{cases} a & \text{if } \left(a = \left(\theta_m^P - \theta_n^Q + 360\right) \bmod 360\right) < 180 \\ a - 180 & \text{otherwise} \end{cases} \quad (18)$$

where α, β, and γ are the weights associated with each component, respectively; δ, ϵ, and ε specify the bounding box; and Ω is a pre-specified penalty for a mismatch. Such an edit distance, to some extent, captures the elastic property of string matching. It represents a cost of changing one polygon to the other. However, this scheme can only tolerate, but not compensate for, the adverse effect on matching produced by the inexact localization of minutia and nonlinear deformations. Therefore, an adaptive mechanism is needed. This adaptive mechanism should be able to track the local nonlinear deformation and inexact alignment and try to alleviate them during the minimization process. However, we do not expect that this adaptive mechanism can handle the "order flip" of minutiae, which, to some extent, can be solved by an exhaustive reordering and matching within a local angular window.

In our matching algorithm, the adaptation is achieved by adjusting the bounding box (Fig. 11) when an inexact match is found during the matching process. It can be represented as follows:

$$w'(m,n) = \begin{cases} \alpha\left|r_m^P - r_n^Q\right| + \beta\Delta e + \gamma\Delta\theta & \text{if} \begin{cases} \delta_l(m,n) < \left(r_m^P - r_n^Q\right) < \delta_h(m,n) \\ \epsilon_l(m,n) < \Delta e < \epsilon_h(m,n) \\ \Delta\theta < \varepsilon \end{cases} \\ \Omega & \text{otherwise} \end{cases} \quad (19)$$

$$\begin{pmatrix} \Delta r_a \\ \Delta e_a \end{pmatrix} = \begin{cases} \begin{pmatrix} r_m^P - r_n^Q \\ \Delta e \end{pmatrix} & \text{if} \begin{cases} \delta_l(m,n) < \left(r_m^P - r_n^Q\right) < \delta_h(m,n) \\ \epsilon_l(m,n) < \Delta e < \epsilon_h(m,n) \\ \Delta\theta < \varepsilon \end{cases} \\ 0 & \text{otherwise} \end{cases} \quad (20)$$

$$\delta_l(m+1, n+1) = \delta_l(m,n) + \eta\Delta r_a \quad (21)$$

$$\delta_h(m+1, n+1) = \delta_h(m,n) + \eta\Delta r_a \quad (22)$$

$$\epsilon_l(m+1, n+1) = \epsilon_l(m,n) + \eta\Delta e_a \quad (23)$$

$$\epsilon_h(m+1, n+1) = \epsilon_h(m,n) + \eta\Delta e_a \quad (24)$$

where $w'(m, n)$ represents the penalty for matching a pair of minutiae $\left(r_m^P, e_m^P, \theta_m^P\right)^T$ and $\left(r_n^Q, e_n^Q, \theta_n^Q\right)^T$, $\delta_l(m, n)$, $\delta_h(m, n)$, $\epsilon_l(m, n)$, and $\epsilon_h(m, n)$ specify the adaptive bounding box in the polar coordinate system (radius and radial angle); and η is the learning rate. This elastic string matching algorithm has a number of parameters which are critical to its performance. We have empirically determined the values of these parameters as follows: $\delta_l(0, 0) = -8$; $\delta_h(0, 0) = +8$; $\epsilon_l(0, 0) = -7.5$; $\epsilon_h(0, 0) = +7.5$; $\varepsilon = 30$; $\alpha = 1.0$; $\beta = 2.0$; $\gamma = 0.1$; $\Omega = 200(\alpha + \beta + \gamma)$; $\eta = 0.5$. The values of $\delta_l(0, 0)$, $\delta_h(0, 0)$, $\epsilon_l(0, 0)$, and $\epsilon_h(0, 0)$ depend on the resolution of fingerprint images. Fig. 12 shows the results of applying the matching algorithm to an input minutia set and a template.

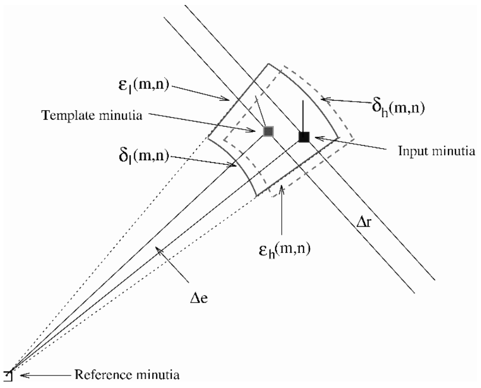


Fig. 11. Bounding box and its adjustment.

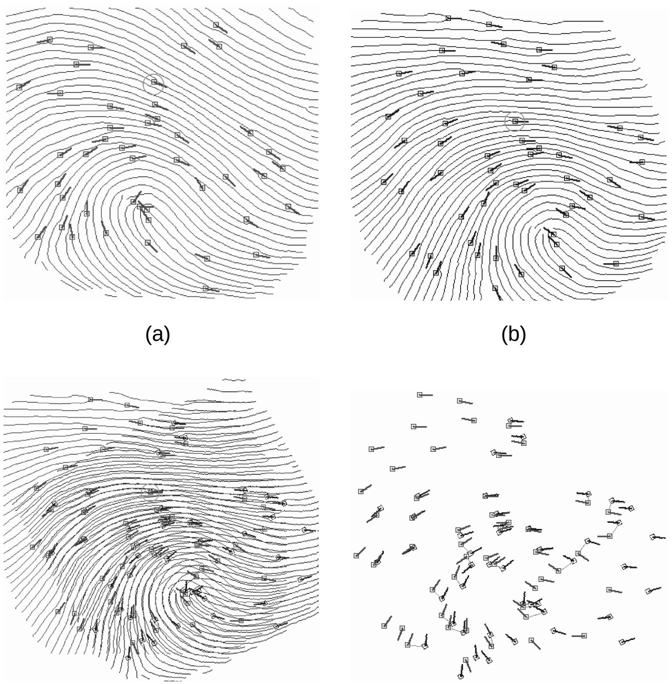


Fig. 12. Results of applying the matching algorithm to an input minutia set and a template. (a) Input minutia set. (b) Template minutia set. (c) Alignment result based on the minutiae marked with green circles. (d) Matching result where template minutiae and their correspondences are connected by green lines.

4 Experimental Results

We have tested our on-line fingerprint verification system on two sets of fingerprint images captured with two different inkless fingerprint scanners. Set 1 contains 10 images per finger from 18 individuals for a total of 180 fingerprint images, which were captured with a scanner manufactured by Identix. The size of these images is 380 × 380. Set 2 contains 10 images per finger from 61 individuals for a total of 610 fingerprint images, which were captured with a scanner manufactured by Digital Biometrics. The size of these images is 640 × 480. When these fingerprint images were captured, no restrictions on the position and orientation of fingers were imposed. The captured fingerprint images vary in quality. Figs. 13 and 14 show some of the fingerprint images in our database. Approximately 90 percent of the fingerprint images in our database are of reasonable quality similar to those shown in Figs. 13 and 14, while about 10 percent of the fingerprint images in our database are not of good quality (Fig. 15), which are mainly due to large creases and smudges in ridges and dryness of the impressed finger. First, we report some initial results on fingerprint matching, followed by fingerprint verification. The reasons why we did not use NIST-9 fingerprint database [25] to test the performance of our system are as follows:

1) we concentrate on live-scan verification, and
2) NIST-9 fingerprint database is a very difficult fingerprint database which contains a large number of fingerprint images of poor quality and no result has been reported from other on-line verification systems for comparison.

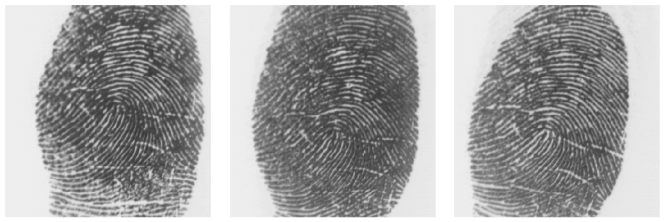


Fig. 13. Fingerprint images captured with a scanner manufactured by Identix; the size of these images is 380 × 380; all the three images are from the same individual's finger.

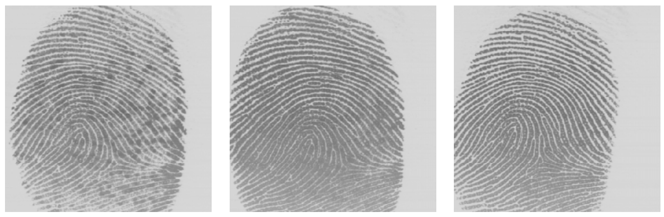


Fig. 14. Fingerprint images captured with a scanner manufactured by Digital Biometrics; the size of these images is 640 × 480; all the three images are from the same individual's finger.

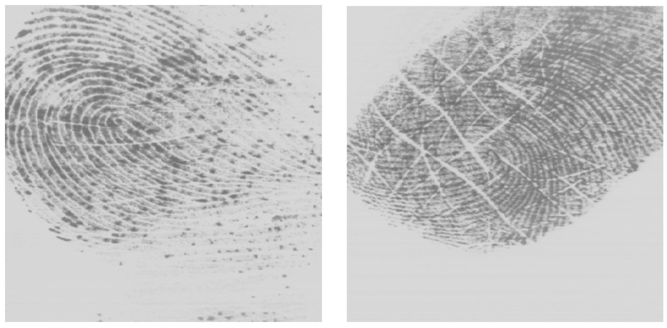
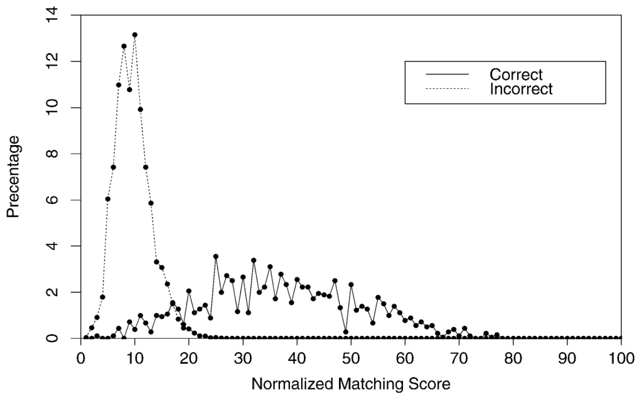


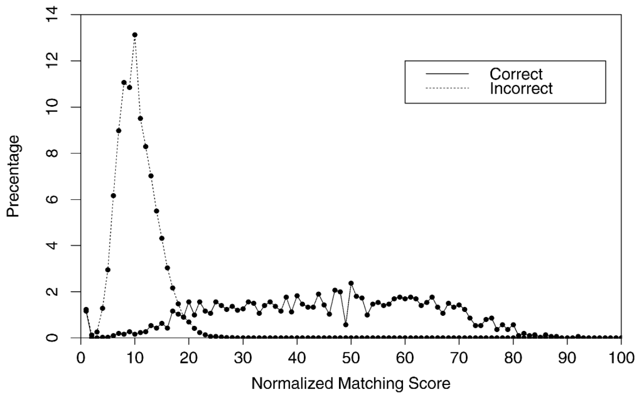
Fig. 15. Fingerprint images of poor quality.

4.1 Matching

Each fingerprint in the test set was matched with the other fingerprints in the set. A matching was labeled correct if the matched fingerprint was among the nine other fingerprints of the same individual, and incorrect otherwise. A total of 32,220 (180 × 179) matchings have been performed on test Set 1 and 371,490 (610 × 609) matchings on test Set 2. The distributions of correct and incorrect matching scores are shown in Fig. 16. It can be seen from this figure that there exist two peaks in the distribution of matching scores. One pronounced peak corresponds to the incorrect matching scores which is located at a value around 10, and the other peak which resides at a value of 40 is associated with the correct matching scores. This indicates that our algorithm is capable of differentiating fingerprints at a high correct rate by setting an appropriate value of the threshold. Table 1 shows the verification rates and reject rates with different threshold values. The reject rate is defined as the percentage of correct fingerprints with their matching scores below the threshold value. As we have observed, both the incorrect matches and the high reject rates are due to fingerprint images with poor quality such as those shown in Fig. 15. We can improve these matching results by ensuring that the database does not contain such poor quality fingerprint images.



(a) Identix



(b) Digital Biometrics

Fig. 16. Distributions of correct and incorrect matching scores; vertical axis represents distribution of matching scores in percentage. (a) Distribution of matching scores on test set 1 (180 images). (b) Distribution of matching scores on test set 2 (610 images).

TABLE 1
THE VERIFICATION RATES AND REJECT RATES ON TEST SETS WITH DIFFERENT THRESHOLD VALUES

Threshold Value	Verification Rate	Reject Rate
20	99.839 %	11.23 %
22	99.947 %	13.33 %
24	99.984 %	16.48 %
26	99.994 %	20.49 %
28	99.996 %	25.19 %
30	100 %	27.72 %

(a) Using Identix system (180 images).

Threshold Value	Verification Rate	Reject Rate
20	99.426 %	11.23 %
22	99.863 %	14.55 %
24	99.899 %	16.78 %
26	99.969 %	20.20 %
28	99.989 %	23.15 %
30	99.999 %	27.45 %

(b) Using Digital Biometrics system (610 images).

TABLE 2
MATCHING RATES ON TEST SETS USING THE LEAVE-ONE-OUT METHOD

Number of Best Matches	Matching Rate
1	91.17 %
2	94.72 %
3	96.89 %
4	98.17 %
5	98.89 %
6	99.39 %
7	99.72 %
8	99.83 %
9	99.94 %

(a) Using Identix system (180 images).

Number of Best Matches	Matching Rate
1	92.13 %
2	94.40 %
3	97.06 %
4	97.67 %
5	98.44 %
6	99.11 %
7	99.70 %
8	99.79 %
9	99.91%

(b) Using Digital Biometrics System (610 images).

TABLE 3
AVERAGE CPU TIME FOR MINUTIA EXTRACTION AND MATCHING ON A SPARC 20 WORKSTATION

Minutia Extraction (seconds)	Minutia Matching (seconds)	Total (seconds)
5.35	2.55	7.90

4.2 Verification

In on-line verification, a user indicates his/her identity. Therefore, the system matches the input fingerprint image only to his/her stored templates. To determine the verification accuracy of our system, we used each one of our database images as an input fingerprint which needs to be verified. An input fingerprint image was matched against all the nine other images of the same finger. If more than one half of the nine matching scores exceeded the threshold value of 25, then the input fingerprint image is said to be from the same finger as the templates and a valid verification is established. With this scheme, a 100 percent verification rate can be achieved with a reject rate around 16 percent on both test sets. Again, this reject rate can be reduced by preprocessing the database to remove the stored templates of poor quality. This demonstrates that, in practice, using a k-nearest neighbor type of matching is adequate for a successful verification. Table 2 shows the matching rate which is defined as the percentage of the correct fingerprints (of the same finger) present among the best *n* (*n* = 1, ..., 9) matches.

For an on-line fingerprint verification system to be acceptable in practice, its response time needs to be within a few seconds. Table 3 shows the CPU requirements of our system. The CPU time for one verification, including fingerprint image acquisition, minutia extraction and minutia matching, is, on an average, approximately eight seconds on a SPARC 20 workstation. It indicates that our on-line fingerprint verification system does meet the response time requirement of on-line verification.

The number of tests done on an automatic fingerprint identification system is never enough. Performance measures are as much a function of the algorithm as they are a function of the database used for testing. The biometrics community is slow at establishing benchmarks and the ultimate performance numbers of a fingerprint verification system are those which you find in a deployed system. Therefore, one can carry out only a limited amount of testing in a laboratory environment to show the anticipated system performance. Even in field testing, real performance numbers are not important—it's often the perceived performance which is crucial.

5 CONCLUSIONS

We have designed and implemented an on-line fingerprint verification system which operates in two stages: minutia extraction and minutia matching. A modified version of the minutia extraction algorithm proposed in [18] is used in our system which is much faster and more reliable. A new hierarchical orientation field estimation algorithm results in a smoother orientation field which greatly improves the performance of the minutia extraction. An alignment-based elastic matching algorithm is proposed for minutia matching. This algorithm is quite fast, because it is capable of finding the correspondences between minutia points without resorting to an exhaustive search. At the same time, this matching algorithm has a good performance, because it has the ability to adaptively compensate for the nonlinear deformations and inexact pose transformations between different fingerprints. Experimental results show that our system achieves excellent performance in a realistic operating environment. It also meets the response time requirement of on-line verification.

Based on the experimental results, we observe that the matching errors in our system mainly result from incorrect minutiae extraction and inaccurate alignment. We observe that a number of factors are detrimental to the correct location of minutia. Among them, poor image quality is the most serious one. Therefore, in the future, our efforts will be focused on global image enhancement schemes. Another issue related to minutia detection is to incorporate a structural-based model in minutia detection which extracts minutiae based on their local ridge formations. For elastic matching, an important aspect is to utilize additional information (e.g., neighboring ridges) about a minutia to increase the accuracy of alignment.

ACKNOWLEDGMENTS

We gratefully acknowledge our useful discussions with Nalina Ratha and Shaoyun Chen, and we would like to thank Nalina Ratha and Chitra Dorai for their careful comments on earlier versions of this paper.

REFERENCES

[1] N. Ansari, M.H. Chen, and E.S.H. Hou, "A Genetic Algorithm for Point Pattern Matching," Chapt. 13, B. Soucek and the IRIS Group, eds., *Dynamic, Genetic, and Chaotic Programming*. New York: John Wiley & Sons, 1992.

[2] P.E. Danielsson and Q.Z. Ye, "Rotation-Invariant Operators Applied to Enhancement of Fingerprints," *Proc. Eighth ICPR*, pp. 329-333, Rome, 1988.

[3] Federal Bureau of Investigation, *The Science of Fingerprints: Classification and Uses*. Washington, D.C.: U.S. Government Printing Office, 1984.

[4] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. New York: McGraw-Hill, 1990.

[5] D.C.D. Hung, "Enhancement and Feature Purification of Fingerprint Images," *Pattern Recognition*, vol. 26, no. 11, pp. 1,661-1,671, 1993.

[6] S. Gold and A. Rangarajan, "A Graduated Assignment Algorithm for Graph Matching," Research Report YALEU/DCS/RR-1062, Yale Univ., Dept. of Computer Science, 1995.

[7] L. O'Gorman and J.V. Nickerson, "An Approach to Fingerprint Filter Design," *Pattern Recognition*, vol. 22, no. 1, pp. 29-38, 1989.

[8] K. Karu and A.K. Jain, "Fingerprint Registration," Research Report, Michigan State Univ., Dept. of Computer Science, 1995.

[9] K. Karu and A.K. Jain, "Fingerprint Classification," *Pattern Recognition*, vol. 29, no. 3, pp. 389-404, 1996.

[10] M. Kawagoe and A. Tojo, "Fingerprint Pattern Classification," *Pattern Recognition*, vol. 17, no. 3, pp. 295-303, 1984.

[11] H.C. Lee and R.E. Gaensslen, eds., *Advances in Fingerprint Technology*. New York: Elsevier, 1991.

[12] D.P. Huttenlocher and S. Ullman, "Object Recognition Using Alignment," *Proc. First Int'l Conf. Computer Vision*, pp. 102-111, London, 1987.

[13] Z.R. Li and D.P. Zhang, "A Fingerprint Recognition System With Micro-Computer," *Proc. Sixth ICPR*, pp. 939-941, Montreal, 1984.

[14] L. Coetzee and E.C. Botha, "Fingerprint Recognition in Low Quality Images," *Pattern Recognition*, vol. 26, no. 10, pp. 1,441-1,460, 1993.

[15] B. Miller, "Vital Signs of Identity," *IEEE Spectrum*, vol. 31, no. 2, pp. 22-30, 1994.

[16] A. Ranade and A Rosenfeld, "Point Pattern Matching by Relaxation," *Pattern Recognition*, vol. 12, no. 2, pp. 269-275, 1993.

[17] A.R. Rao, *A Taxonomy for Texture Description and Identification*. New York: Springer-Verlag, 1990.

[18] N. Ratha, S. Chen, and A.K. Jain, "Adaptive Flow Orientation Based Feature Extraction in Fingerprint Images," *Pattern Recognition*, vol. 28, no. 11, pp. 1,657-1,672, 1995.

[19] A. Sherstinsky and R.W. Picard, "Restoration and Enhancement of Fingerprint Images Using M-Lattice-A Novel Non-Linear Dynamical System," *Proc. 12th ICPR-B*, pp. 195-200, Jerusalem, 1994.

[20] D.B.G. Sherlock, D.M. Monro, and K. Millard, "Fingerprint Enhancement by Directional Fourier Filtering," *IEE Proc. Vis. Image Signal Processing*, vol. 141, no. 2, pp. 87-94, 1994.

[21] J.P.P. Starink and E. Backer, "Finding Point Correspondence Using Simulated Annealing," *Pattern Recognition*, vol. 28, no. 2, pp. 231-240, 1995.

[22] G. Stockman, S. Kopstein, and S. Benett, "Matching Images to Models for Registration and Object Detection via Clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 4, no. 3, pp. 229-241, 1982.

[23] J. Ton and A.K. Jain, "Registering Landsat Images by Point Matching," *IEEE Trans. Geoscience and Remote Sensing*, vol. 27, no. 5, pp. 642-651, 1989.

[24] V.V. Vinod and S. Ghose, "Point Matching Using Asymmetric Neural Networks," *Pattern Recognition*, vol. 26, no. 8, pp. 1,207-1,214, 1993.

[25] C.I. Watson and C.L. Wilson, *NIST Special Database 4, Fingerprint Database*, National Institute of Standards and Technology, Mar. 1992.

[26] C.L. Wilson, G.T. Gandela, and C.I. Watson, "Neural-Network Fingerprint Classification," *J. Artificial Neural Networks*, vol. 1, no. 2, pp. 203-228, 1994.

[27] Q. Xiao and Z. Bian, "An Approach to Fingerprint Identification by Using the Attributes of Feature Lines of Fingerprint," *Proc. Seventh ICPR*, pp. 663-665, Paris, 1986.

Anil Jain received a BTech degree in 1969 from the Indian Institute of Technology, Kanpur, India, and the MS and PhD degrees in electrical engineering from the Ohio State University in 1970 and 1973, respectively. He joined the faculty of Michigan State University in 1974, where he currently holds the rank of University Distinguished Professor in the Dept. of Computer Science. Dr. Jain has made significant contributions and published a large number of papers on the following topics: statistical pattern recognition, exploratory pattern analysis, Markov random fields, texture analysis, interpretation of range images, neural networks, document image analysis, and 3D object recognition. Several of his papers have been reprinted in edited volumes on image processing and pattern recognition. He received the best paper awards in 1987 and 1991 and certificates for outstanding contributions in 1976, 1979, and 1992 from the Pattern Recognition Society. He also received the 1996 IEEE Transactions on Neural Networks Outstanding Paper Award. Dr. Jain was the editor-in-chief of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1990–1994), and he also serves as an associate editor for *Pattern Recognition*, *Pattern Recognition Letters*, *IEEE Transactions on Neural Networks, Applied Intelligence,* and the *Journal of Mathematical Imaging and Vision*. He is the coauthor of *Algorithms for Clustering Data* (Prentice Hall, 1988), has edited *Real-Time Object Measurement and Classification* (Springer-Verlag, 1988), and coedited *Analysis and Interpretation of Range Images* (Springer-Verlag, 1989), *Markov Random Fields* (Academic Press, 1992), *Artificial Neural Networks and Pattern Recognition* (Elsevier, 1993), and *3D Object Recognition* (Elsevier, 1993). He is a Fellow of the IEEE and IAPR.

Lin Hong received the BS and MS degrees in computer science from Sichuan University, China, in 1987 and 1990, respectively. He is currently a PhD student in the Dept. of Computer Science, Michigan State University. His current research interests include pattern recognition, image processing, biometrics, and computer graphics.

Ruud Bolle (S‘82-M’84-F’96) received the bachelor’s degree in analog electronics in 1977 and the master’s degree in electrical engineering in 1980, both from Delft University of Technology, Delft, The Netherlands. In 1983 he received the master’s degree in applied mathematics and in 1984 the PhD in electrical engineering from Brown University, Providence, Rhode Island. In 1984 he became a research staff member in the Artificial Intelligence Group, Computer Science Dept., at the IBM Thomas J. Watson Research Center. In 1988 he became manager of the newly formed Exploratory Computer Vision Group, which is part of IBM’s digital library effort. His current research interests are video database indexing, video processing, and biometrics. Dr. Bolle is a Fellow of the IEEE. He is on the Advisory Council of *IEEE Transactions on Pattern Analysis and Machine Intelligence*, and he is associate editor of *Computer Vision and Image Understanding*. He is the guest editor of a special issue on computer vision applications for network-centric computers in *Computer Vision and Image Understanding.*