A systematic method for fingerprint ridge orientation estimation and image segmentation

En Zhu*, Jianping Yin, Chunfeng Hu, Guomin Zhang

School of Computer Science, National University of Defense Technology, Changsha 410073, China

Received 16 June 2005; received in revised form 25 February 2006; accepted 1 March 2006

Abstract

This paper proposes a scheme for systematically estimating fingerprint ridge orientation and segmenting fingerprint image by means of evaluating the correctness of the ridge orientation based on neural network. The neural network is used to learn the correctness of the estimated orientation by gradient-based method. The trained network is able to distinguish correct and incorrect ridge orientations, and as a consequence, the falsely estimated ridge orientation of a local image block can be corrected using the around blocks of which orientations are correctly estimated. A coarse segmentation can also be done based on the trained neural network by taking the blocks of correctly estimated orientation as foreground and the blocks of incorrectly estimated orientation as background. Besides, following the steps of estimating ridge orientation correctness, a secondary segmentation method is proposed to segment the remaining ridges which are the afterimage of the previously scanned fingers. The proposed scheme serves for minutiae detection and is compared with VeriFinger 4.2 published by Neurotechnologija Ltd. in 2004, and the comparison shows that the proposed scheme leads to an improved accuracy of minutiae detection.

© 2006 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Ridge orientation; Neural network; Fingerprint segmentation; Remaining ridges; Secondary segmentation; Segmentation revision; Minutiae detection

1. Introduction

Fingerprint recognition is one of the technologies of biometric authentication. Fingerprint image consists of interweaving ridges and valleys. Ridge termination and bifurcation, uniformly called minutiae, are the characteristic features of fingerprint. Automatic fingerprint recognition typically goes through a series of processes, mainly including ridge orientation estimation, segmentation, enhancement, minutiae detection and matching [1]. Minutiae detection and matching are heavily dependent on ridge orientation estimation and image segmentation, since ridge orientation is inevitably used for detecting, describing and matching minutiae, and correct image segmentation helps to avoid detecting spurious minutiae in low quality image regions.

There are many methods for ridge orientation estimation in literatures. Most of them are based on gray-scale relationship between pixels [2–13]. Kawagoe and Tojo [2] compare each pixel block of size $2 \times 2$ with four types of edge templates to estimate the coarse orientation of the block, and then averages the orientation within a larger region. Mehtre et al. [3] compute gray consistency along 16 directions, respectively, at each pixel. And the orientation of the best consistency is taken as the ridge orientation for the corresponding pixel. Hung [4] binarizes the pixels into ridge pixels and non-ridge pixels, and evaluates ridge orientation by computing the consistency of pixel type, ridge and non-ridge, along a discrete number of orientations. The orientation estimation method proposed by Kovacs-Vajna et al. [5] is similar to that proposed in Ref. [4]. Sherlock et al. [6] project a local image block along different orientations, and the orientation associated with the maximum variance of the projected...
signal is taken to be the local ridge orthogonal orientation. Nagaty [7] uses a hierarchical neural network to estimate ridge orientation. The most popular method for ridge orientation estimation is the gradient-based method [8–13]. It computes the gradient vector at each pixel. Along the direction of the gradient vector, the pixel-intensity changes most rapidly and the length of the vector indicates the speed of this change: larger length, faster change. It is the evidence for gradient-based method that the hypothetical edge, which is parallel to ridge orientation for good ridge structure, at an edge pixel is orthogonal to its gradient vector direction. That is to say that the gradient vector direction of an edge pixel is orthogonal to the ridge orientation of the local block which is of good structure. And the local ridge orientation is estimated by means of averaging gradient vector direction of the pixels in the local block. However, if the edge pixels, whose gradient is not orthogonal to the ridge orientation, dominate the local image block, a falsely estimated ridge orientation will result. Fig. 1 shows an example of false estimation of local ridge orientation. Fig. 1(a) is the original image. Fig. 1(b) is the orientation estimation by gradient-based method. And Fig. 1(c) is the enlarged sub-image to clearly show the edge pixels. The enlarged sub-image shows us that the gradient direction of most of the edge pixels is not orthogonal to the local ridge orientation, and as a consequence, the local ridge orientation is falsely estimated. In order to revise the falsely estimated orientation, Hong et al. [8] filter the orientation field using a low-pass filter. Low-pass filtering is able to smooth the orientation and is very effective when the local image block of falsely estimated orientation is relatively isolated as shown in Fig. 2(a1)–(c1). However, if the falsely estimated orientations dominate the local region, the correctly estimated orientation will be falsely revised by low-pass filtering as shown in Fig. 2(a2)–(c2). Besides, the location of the core point will shift slightly away from the true location after low-pass filtering as shown in Fig. 2(a3)–(c3).

There is another class of quite different methods for ridge orientation estimation, which globally predict orientation in noisy area by modeling the orientation field [14–19]. Sherlock and Monro [14] generate the orientation field from the positions of singular points including core and delta. Obviously, it is possible that two fingerprints have the same positions of singularities, while they have different orientation pattern. Sherlock–Monro model cannot distinguish the difference and is improved by Vizcaya and Gerhardt [15] and Araque et al. [16]. However, Sherlock–Monro model is very useful for fingerprint generation as proposed by Capelli et al. [17]. The orientation from these models does not represent the true ridge orientation of a real fingerprint image, and the model is collapsed when the fingerprint contains no singularities or the positions of singularities cannot be correctly located. Gu et al. [18] and Zhou and Gu [19] propose a model method combining gradient-based method, which computes the coarse orientation field based on gradient method and detects the singularities, and then the orientation field is globally smoothed using LMS model.
Fingerprint ridge orientation estimation, especially for low quality image, is still a challenging problem in automatic fingerprint recognition and new creative methods for orientation estimation and correction are expected to be proposed and investigated.

Fingerprint segmentation is mainly to remove non-ridge regions and unrecoverable ridge regions so as to avoid detecting spurious features and losing true features and to decrease the time of image processing. The usual method for segmentation divides the fingerprint image into unoverlapped blocks for each of which several features are computed to determine whether it belongs to foreground or background [3,8,20–22]. Besides, there are some methods segmenting fingerprint image pixel-wise [23]. The features used for segmentation mainly include gray-scale statistical features, local directionality, statistics of direction image, consistency of orientations, ridge frequency and ridge projection, etc. Refs. [3,24,20,21] use gray variance and the histogram of gradient direction for segmentation. Ratha et al. [9] compute the variance of ridge projection along a discrete number of directions for each image block. For blocks containing ridges, the ridge projection is of large variance along the direction parallel to the ridge and small along the orthogonal. And for block containing no ridges, the projection is of small variance along all the directions. Hong et al. [8] also use the features, including frequency, variance and amplitude, of the projection signal along the direction parallel to the ridge orientation for segmentation. Klein et al. [22] use gray mean, gray variance, gradient consistency and Gabor responding for segmentation based on HMM. And Bazen and Gerez [23] use gradient consistency, gray mean and variance to segment fingerprint image pixel-wise.

The challenging problem in fingerprint segmentation is the acceptance or rejection of low quality ridge block as foreground. The decision should depend on the capability of ridge restoration algorithm: if the low quality ridge structure is not recoverable for the algorithm, the corresponding block should be regarded as background to avoid detecting spurious features, else as foreground. The existing segmentation methods [3,8,9,20–24] segment fingerprint image from the visual aspect to remove visually unrecoverable ridge regions and non-ridge regions. However, some visually recoverable ridge regions are not recoverable for a specific algorithm. Fig. 1 shows an example in which the circled region is of falsely estimated ridge orientations and it is difficult to extract true ridge structure for the enhancement based on context filtering. Thus it is better to remove the circled region. Visually recoverable region is not necessarily recoverable for a specific algorithm, since the algorithms cannot do better than human vision on recovering ridge
structure. And constantly, there are some remaining ridges, which are the afterimage of previously scanned fingers and are expected to be removed. Thus a fingerprint image can be gradually partitioned. Firstly, it can be partitioned into non-ridge region (type A) and ridge region (type A'). The latter can be partitioned into remaining ridge region (type B) and real ridge region (type B'). Region of type B' can be partitioned further into visually unrecognizable region (type C) and visually recoverable region (type C'). And type C' region can be partitioned into algorithmically unrecognizable region (type D) and algorithmically recoverable region (type E). Fig. 3(a) gives the partition tree. A fingerprint image usually contains one or more of these types of regions. Fingerprint segmentation should remove types A–D while remain type E. Of course, fingerprint region can be partitioned by other order as shown in Fig. 3(b) which is an alternative partition order of Fig. 3(a). Each partition stage of Fig. 3(a) and (b) is a binary classifying problem. We call the segmentation as in Fig. 3(a) and (b) multiple layered partition (MLP) as in Fig. 3(e), which simply uses one binary classifier to partition the fingerprint image into background and foreground. MLP-based segmentation is able to more accurately segment fingerprint than OLP-based method, since the classifying space at each partition stage of MLP-based method is smaller than OLP-based method.

Correct fingerprint ridge orientation estimation and image segmentation play an important role in detecting fingerprint features, typically minutiae, since they can help to improve minutiae detection. It is difficult to correctly detect minutiae from low quality fingerprint images, since it is difficult to correctly compute their ridge orientations and exactly segment them. There are two kinds of errors in minutiae detection: (1) losing true minutiae, and (2) detecting spurious minutiae. In order to avoid errors, we should compute the ridge orientation as accurate as possible, and falsely estimated orientation should be corrected according to its surrounding correct orientation, and those blocks of uncorrectable orientation should be removed. Therefore we propose a systematic scheme for distinguishing between correct and incorrect orientations by means of machine learning using neural network. The trained network is used to identify incorrect orientation and make a coarse segmentation of fingerprint image. And a secondary segmentation method for removing remaining ridges, which is the after image of the previously scanned fingers, is proposed. Following the secondary segmentation, the segmentation is revised based on heuristic rules, and some incorrect orientations can be corrected during the segmentation revising. The proposed scheme serves for minutiae detection and leads to an improved accuracy.

Section 2 defines some notations and describes the main steps of the proposed systematic scheme. The detailed algorithms are described in Sections 3–6. Section 7 is the Experimental results. And Section 8 makes a conclusion.

2. Notations and main steps

Let us begin with defining some notations. Suppose \( I \) is a fingerprint image of size \( m \times n \), and \( I(x, y)(0 \leq x < m, 0 \leq y < n) \) is the intensity of the pixel \((x, y)\). The fingerprint image is to be segmented block-wise, since pixel-wise segmentation consumes more time than block-wise segmentation. We divide the image into unoverlapped blocks of size \( w \times w (15 \times 15) \) as shown in Fig. 4.

Without loss of generality, we suppose both \( m \) and \( n \) are multiple times of \( w \), since the image margins can be cut to meet this. Therefore, there are \((m/w) \times (n/w)\) unoverlapped blocks, denoted as \( W_i(i, j) \) \((0 \leq i < m/w, 0 \leq j < n/w)\), where \((i, j)\) indicates the position of the block \( W_i(i, j) \). And \( W_i(i, j) \) centers at \((i \cdot w + w/2, j \cdot w + w/2)\). All these blocks are to be classified into background or foreground which are to be marked by \( M(W_i(i, j)) : W_i(i, j) \) is a foreground block if \( M(W_i(i, j)) = 1 \) and background if \( M(W_i(i, j)) = 0 \). Let \( O(W_i(i, j)) \) be the ridge orientation of block \( W_i(i, j) \). There is no difference between \( \theta \) and \( \theta + \pi \) for ridge orientation, and therefore, the ridge orientation is regulated into \([0, \pi)\). In practice, the ridge orientation can be computed, at each pixel, as the orientation of the local region centered at the pixel. Let \( O(x, y) \) be the ridge orientation of the local region centered at the pixel \((x, y)\).

Fingerprint segmentation is to remove uninterested blocks to improve the accuracy of minutiae detection and decrease the time consumption. Correct ridge orientation estimation is the foundation of minutiae detection. Blocks of incorrectly estimated orientation will introduce spurious minutiae. First,
fingerprint image can be easily segmented into non-ridge regions and ridge regions. Second, we are to segment ridge regions based on the correctness of the estimated ridge orientation: classifying ridge blocks of correctly estimated orientation into foreground and other blocks into background, during which a neural network is trained to compute the orientation correctness. However, it is possible that the image contains remaining ridges of clear structure and these ridges may be classified into foreground, since the ridges of clear structure will have correctly estimated orientation. The remaining ridges falsely classified into foreground are expected to be revised to background by means of secondary segmentation. This segmentation process is like Fig. 3(c). For non-ridge block, the ridge orientation makes no sense and can be taken as negative input, incorrect orientation, when training the network, and thus the segmentation process can be simplified like in Fig. 3(d). During the process, some algorithms are needed: (1) an algorithm coarsely computing the ridge orientation, (2) a scheme computing the correctness of the estimated orientation by the prior step, (3) coarse segmentation based on orientation correctness and secondary segmentation removing remaining ridges which are falsely classified into foreground in the coarse segmentation, (4) segmentation revision and orientation correction. As a whole, the proposed systematic methods for ridge orientation estimation and image segmentation include the following steps:

(1) Orientation estimation: computing the orientation field \(O(W(i,j)))\) \((0 \leq i < m/w, 0 \leq j < n/w)\) of the fingerprint image.

(2) Orientation correctness evaluation: compute the correctness of \(O(W(i,j))\).

(3) Coarse segmentation: classifying each image block into foreground or background according its orientation correctness.

(4) Secondary segmentation: classifying the foreground output by the coarse segmentation into remaining ridge blocks and true ridge blocks.

(5) Orientation correction and segmentation revision: post-processing the segmentation to change the assigned class of isolated foreground blocks and background blocks. Some background blocks, which have visually recoverable ridge structure but their estimated ridge orientations are not correct, should be revised to foreground blocks, as long as their ridge orientation is able to be corrected by their neighbor blocks of correct orientations.

3. Computing ridge orientation

The initial ridge orientation is estimated using gradient-based method [8,10]. The local ridge orientation of a block of size \(w \times w\) centered at pixel \((i, j)\), denoted as \(\theta^I(i, j)\), is computed using Eqs. (1)–(6), where \((G_x(u, v), G_y(u, v))\) is the gradient vector of the pixel \((u, v)\). Eq. (6) regulates the orientation into \([0, \pi)\). And Eq. (7) sets orientation of block \(W^I(k, l)\) to \(O^I(k \cdot w + w/2, l \cdot w + w/2)\)

\[
G_x(u, v) = \frac{\partial I(u, v)}{\partial u}, \quad G_y(u, v) = \frac{\partial I(u, v)}{\partial v}, \quad \theta^I(i, j) = \tan^{-1}(V_y/V_x)/2, \quad O(W^I(k, l)) = \left\{
\begin{array}{ll}
\pi/4, & V_x = 0, V_y < 0, \\
3\pi/4, & V_x = 0, V_y \geq 0,
\end{array}
\right. \quad O^I(i, j) = \left\{
\begin{array}{ll}
\theta^I(i, j) + \pi/2, & V_x > 0, \\
\theta^I(i, j), & V_x < 0, V_y \leq 0,
\end{array}
\right. \quad O(W^I(k, l)) = O^I(k \cdot w + w/2, l \cdot w + w/2).
\]

4. Distinguishing correct and incorrect orientation

This section is to present and explain the features used to classify ridge blocks of correctly estimated orientations into foreground and else into background including non-ridge blocks and ridge blocks of incorrectly estimated orientation. The remaining ridges of the foreground will be revised to background in Section 5. And the correctable orientations of some ridge blocks of incorrectly estimated orientations will be corrected in Section 6. We regard the orientation of a non-ridge block as incorrect, since for a non-ridge block, its orientation makes no sense. For each image block, 11 features, denoted as \(\langle C_1, C_2, \ldots, C_{11} \rangle\), are computed for distinguishing correct and incorrect orientations. The first fours features are independent of orientation and are used for distinguishing ridge blocks and non-ridge blocks. The last seven features are dependent on orientation and are used for distinguish between correct and incorrect orientations of ridge blocks. Let \((i, j)\) be the center of block \(W(k, l)\). The feature vector \(\langle C_1, C_2, \ldots, C_{11} \rangle\) for image block \(W(k, l)\) is computed as follows:

(1) \(C_1\), the normal of the sum of the squared gradient vector in the block \(W(k, l)\), is computed as Eq. (8). For ridge blocks of clear ridge pattern, the value of \(C_1\) will be large. And for noisy blocks and non-ridge blocks,
the value will be small

$$C_1 = \left( \sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} (G_x(u, v) - G_y(u, v)) \right)^2$$

$$= \left( \sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} (G_x^2(u, v) - G_y^2(u, v)) \sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} (2 \cdot G_x(u, v)G_y(u, v)) \right)^2$$

(2) $C_2$, the sum of the normal of the squared gradient vector in the block $W(k, l)$, is computed as Eq. (9). For non-ridge blocks, the value of $C_2$ is usually small. And for ridge blocks, the value of $C_2$ is usually large

$$C_2 = \sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} \left( \frac{(G_x(u, v) - G_y(u, v))^2}{(G_x(u, v) - G_y(u, v))^2 + (2 \cdot G_x(u, v) \cdot G_y(u, v))^2} \right)^2 \sqrt{\left( \sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} \left( \frac{(G_x(u, v) - G_y(u, v))^2}{(G_x(u, v) - G_y(u, v))^2 + (2 \cdot G_x(u, v) \cdot G_y(u, v))^2} \right)^2 \right)}$$

(3) $C_3$, the mean gray of the block $W(k, l)$, is computed as Eq. (10). The mean gray value of ridge blocks is usually smaller than non-ridge blocks

$$P_h[k] = \frac{\sum_{h=-H/2}^{H/2} I(i+h \cdot \cos(O^I(i, j))) + k \cdot \sin(O^I(i, j))) + j \cdot \sin(O^I(i, j))) - k \cdot \cos(O^I(i, j)))}{H}$$

$$C_3 = \frac{\sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} I(u, v)}{w \cdot w}$$

(4) $C_4$, the gray variance of the block $W(k, l)$, is computed as Eq. (11). The variance of a ridge block is usually larger than a non-ridge block

$$C_4 = \frac{\sum_{u=-w/2}^{i+w/2} \sum_{v=-w/2}^{j+w/2} (I(u, v) - C_3)^2}{w \cdot w - 1}$$

(5) $C_5$ is the variance of the projected signal along the direction parallel to the estimated local ridge orientation as shown in Fig. 5(a). The projection window is of size $L \times H (21 \times 15)$ and with the side $L$ parallel to the estimated local orientation. The center of the window and the center of the block $W(k, l)$ are overlapped. $C_5$ is computed as Eqs. (12) and (13)

$$P_v[k] = \frac{\sum_{h=-H/2}^{H/2} (i-h \cdot \sin(O^I(i, j))) + k \cdot \cos(O^I(i, j))) + j \cdot \sin(O^I(i, j))) + k \cdot \sin(O^I(i, j)))}{H}$$

$$C_5 = \frac{\sum_{k=-L/2}^{L/2} P_v[k]^2}{L'}$$

(6) $C_6$ is the variance of the projected signal along the direction orthogonal to the estimated local ridge orientation as shown in Fig. 5(b). The projection window is of size $L \times H (41 \times 15)$ (side $H$ is parallel to the estimated local orientation) and its center is overlapped with the center of lock $W(k, l)$. $C_6$ is computed using Eqs. (14) and (15). For a ridge block of correctly estimated orientation, $C_6$ is large while $C_5$ is small

$$C_6 = \frac{\sum_{k=-L/2}^{L/2} (P_v[k] - \sum_{k=-L/2}^{L/2} P_v[k]/L)^2}{L - 1}$$

The features from $C_7$ to $C_{11}$ are computed using the smoothed signal, by Gauss filter as in Eq. (16), of $P_v[k] (L \leq k \leq L/2)$. Fig. 6 gives out an example of computing from $C_7$ to $C_{11}$

$$P[k] = \sum_{i=-3}^{3} Gauss(i) \cdot P_v[k+i], \quad \left( -\frac{L}{2} \leq k \leq \frac{L}{2} \right).$$

(7) $C_7$ is average inter-ridge distance which is within a certain range.

$$C_7 = \frac{\sum_{k=-L/2}^{L/2} (P_v[k] - \sum_{k=-L/2}^{L/2} P_v[k]/L)^2}{L - 1}$$

(8) $C_8$ is the variance of the peak heights of the signal.

(9) $C_9$ is the variance of the valley heights of the signal. Both $C_8$ and $C_9$ are small for a block of high quality ridges and large for a block of noisy ridges.

$$C_8 = \frac{\sum_{k=-L/2}^{L/2} P_v[k]^2}{L'}$$

$$C_9 = \frac{\sum_{k=-L/2}^{L/2} (P_v[k] - \sum_{k=-L/2}^{L/2} P_v[k]/L)^2}{L' - 1}$$
Fig. 5. Projection of ridges.

C10 is the average amplitude of the signal which is bigger in ridge blocks than in non-ridge blocks.

C11 is the variance of inter-peak-valley distance. The variance is small for a block of high quality ridges and large for a block of noisy ridges.

5. Segmenting image

5.1. Coarse segmentation

The coarse segmentation is to segment the fingerprint image according to the orientation correctness of each image block, regarding the orientation of a non-ridge block as incorrect. The orientation correctness is learned by a BPNN as shown in Fig. 7(b). The neural network uses the feature vector \( C_1, C_2, \ldots, C_{11} \) in Section 4 as its input. At the stage of training, we take ridge blocks of correctly estimated orientation as positive samples and take the ridge blocks of incorrectly estimated orientation as negative samples. Fig. 8 gives out some positive and negative samples. The neural network outputs 1 for positive inputs, and outputs 0 for negative inputs. The correctness of block orientation is estimated using the trained neural network as shown in Fig. 7(a) and (c). At the stage of estimation, the output of the neural network is between \([0, 1]\). The network will respond with a large value to correct orientation and a small value to incorrect orientation. Large responding value indicates that the corresponding block contains ridges and is of correctly estimated orientation. And small responding value indicates that the corresponding block either contains noisy ridges of incorrectly estimated orientation or contains no ridges. Coarse segmentation is to take blocks of large responding value over a threshold \( t_F \) (0.5, and all the threshold values in this paper are empirically selected through trial and error) as foreground and take blocks of small responding value below the threshold \( t_F \) as background. The foreground and background are marked using \( M \): if block \( W^I(i, j) \) is foreground, then \( M(W^I(i, j)) = 1 \); and if \( W^I(i, j) \) is background, then \( M(W^I(i, j)) = 0 \). Fig. 9 gives out an example of coarse segmentation according to the responding values.

5.2. Secondary segmentation

The coarse segmentation takes the ridge blocks of correct orientation as foreground. As a consequence, the remaining ridges which are the afterimage of the previously scanned fingers will be classified into foreground by the coarse segmentation, if they are of good quality of ridge structure which will result in correctly estimated orientation. Remaining ridges are to produce spurious features. The secondary segmentation is proposed to remove the remaining ridges. Constantly, remaining ridges are of clear structure. And for two fingerprint image, \( I_1 \) and \( I_2 \), containing remaining ridges, it is possible that the remaining ridges of \( I_1 \) have the similar characteristics with the non-remaining ridges (true ridges) of \( I_2 \). Therefore, it is difficult to distinguish remaining ridges from the interested ridges by just one-layered segmentation as like in Fig. 3(e) which is the typical scheme by existing methods. Within a fingerprint image containing remaining ridges, there are two typical differences, \( A \) and \( B \), between the true ridges and the remaining rides as follows:

(A) The pixel gray value of the remaining ridges is usually bigger than the true ridges.
(B) The amplitude of the remaining ridge blocks is usually smaller than that of the true ridge blocks. That is to say the gray difference between the ridges and valleys for remaining ridge blocks is usually smaller than for true ridge blocks.
Therefore, at the stage of the secondary segmentation, four features are computed for each foreground block $W$ of the coarse segmentation to be used for distinguishing remaining ridges and true ridges within the same fingerprint image. The four features to be computed for a foreground block $W$ are as follows:

1. Local mean gray of the block $W$, denoted as $LG$ (local gray), $LG = C_3$.

2. Mean gray of all the foreground of the coarse segmentation, denoted as $MG$ (mean gray). $MG$ is computed using Eq. (17):

   $$ MG = \frac{\sum_{W^I(i,j)} LG(W^I(i,j))}{\sum_{W^I(i,j)} M(W^I(i,j))}. \quad (17) $$

3. Local amplitude of the block $W$, denoted as $LA$ (local amplitude), $LA$ is computed as the difference between the average peak height and average valley height of projected signal $P[k]|(-L/2 \leq k \leq L/2)$ corresponding to the block $W$.

4. Average amplitude of all the foreground blocks of the coarse segmentation. Denoted as $MA$ (mean amplitude), $MA$ is computed using Eq. (18)

   $$ MA = \frac{\sum_{W^I(i,j)} LA(W^I(i,j))}{\sum_{W^I(i,j)} M(W^I(i,j))}. \quad (18) $$

Within a fingerprint image, the difference of type A can be indicated by $LG$ and $MG$: $LG - MG$ is big for remaining
ridge blocks and small for true ridge blocks. The difference of type B can be indicated by $LA$ and $MA$: $LA-MA$ is small for remaining ridge blocks and big for true ridge blocks. However, certain blocks with small $LG$ and big $LA$ should be kept still be foreground without regarding the values $LG-MG$ and $LA-MA$, since small $LG$ value and big $LA$ value indicates the corresponding block is a true ridge block. And contrarily, blocks with big $LG$ and small $LA$ should be reclassified to background blocks without regarding the values of $LG-MG$ and $LA-MA$, since big $LG$ value and small $LA$ value indicates that the corresponding block is a remaining ridge block. The secondary segmentation uses $\langle LG, MG, LA, MA \rangle$ to reclassify the foreground blocks of the coarse segmentation by means of linear classifier, $LMS$ [25]. 400 instances, each 200 for positive (true ridge block), and negative (remaining ridge block), respectively, are used to estimate the parameters of the $LMS$ model. Let the instances be denoted as $\{\langle LG(W_i), MG(W_i), LA(W_i), MA(W_i), y(W_i) \rangle | 1 \leq i \leq N \}$, and $y(W_i) = 1$ if $W_i$ is a positive instance, and $y(W_i) = -1$ if $W_i$ is a negative instance. The $LMS$ model in this case is described as Eq. (19), where $\langle a_0, a_1, a_2, a_3, a_4 \rangle$ are the parameters to be estimated through all the 400 instances

$$
\begin{pmatrix}
1 & LG(W_1) & MG(W_1) & LA(W_1) & MA(W_1) \\
1 & LG(W_2) & MG(W_2) & LA(W_2) & MA(W_2) \\
1 & LG(W_3) & MG(W_3) & LA(W_3) & MA(W_3) \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
1 & LG(W_N) & MG(W_N) & LA(W_N) & MA(W_N)
\end{pmatrix} \times
\begin{pmatrix}
a_0 \\
a_1 \\
a_2 \\
a_3 \\
a_4
\end{pmatrix} =
\begin{pmatrix}
y(W_1) \\
y(W_2) \\
y(W_3) \\
\vdots \\
y(W_N)
\end{pmatrix}.
$$

(19)

At the stage of the secondary segmentation, for a block $W_i$ which is foreground block of the coarse segmentation, $\hat{y}(W_i)$ is computed as Eq. (20). And $W_i$ would be reclassified
into background (setting $M(W_i) = 0$) if $\text{sign}(\hat{y}(W_i)) = -1$,
\[
\hat{y}(W_i) = a_0 + a_1 \cdot LG(W_i) + a_2 \cdot MG(W_i) + a_3 \cdot LA(W_i) + a_4 \cdot MA(W_i).
\] (20)

Fig. 10 shows an example of secondary segmentation. The remaining ridge blocks as in Fig. 10(c) are reclassified to background as in Fig. 10(d) in which the holes can be filled through segmentation revision in Section 6. Fig. 10(b) is the segmentation result of VeriFinger 4.2 published by Neurotechnologija Ltd. in 2004.

6. Revising segmentation and correcting orientation

Except for the remaining ridges, there are two kinds of errors in coarse segmentation according to the orientation correctness computed by the trained network: (1) some blocks, which are of incorrectly estimated orientations but of large network responding value, are falsely classified into foreground; (2) some blocks, which are of correctly estimated orientations but of small network responding values, are falsely classified into background. The segmentation revision, following the secondary segmentation, is to correct the two kinds of errors. At the same time, some ridge blocks, which are of incorrectly estimated orientation and are classified into background in the coarse segmentation, can have their orientations corrected according to their neighbouring blocks, and then can be revised to foreground.

6.1. Segmentation revision

Falsely classified blocks are typically and relatively isolated. These blocks can be revised according to their neighbors using heuristic rules. Fig. 11 lists the rules used in the proposed scheme. There are two classes of heuristic rules: one revising background block to foreground (rules A–D) and another revising foreground to background (rules E and F) as shown in Fig. 11. Segmentation revision will go through the image block-by-block from bottom to top and from left to right to check each block whether it meets the condition of anyone of the rules and to revise the block if it satisfies one of the rules. The collision between the rules, that an image block satisfies more than one rule, will be coped with late in this section. Suppose that the block to be checked for the moment is $W(i, j)$, $i$th column and $j$th row. The detailed descriptions of the heuristic rules are as follows.

Rule A: If the block $W(i, j)$ is a background block, and there are two symmetric neighbor blocks, $W_1$ and $W_2$, with regard to the block $W(i, j)$ as shown in Fig. 11 (rule A), and the orientation difference between $W_1$ and $W_2$ is below a threshold $t_1$, then the block $W(i, j)$ is revised into foreground and its orientation is computed as the average orientation of $W_1$ and $W_2$. The computation of the average orientation is to be described in Section 6.2. The orientation difference between the two blocks, $W_1$ and $W_2$, is computed using Eq. (21)

\[
\text{Dis}(O(W_1), O(W_2)) = \begin{cases} 
|O(W_1) - O(W_2)|, & |O(W_1) - O(W_2)| \leq \pi/2, \\
\pi - |O(W_1) - O(W_2)|, & |O(W_1) - O(W_2)| > \pi/2,
\end{cases}
\] (21)

Rule B: If the block $W(i, j)$ is a background block, and there are more than four neighbor foreground blocks around
$W(i, j)$, and the responding value of trained network to the input features corresponding to the average orientation of these neighbor foreground blocks is larger than threshold $t_{CN}$ (set to 0.1 in experiments), then the block $W(i, j)$ is revised into foreground and its orientation is computed as the average orientation of its neighbor foreground blocks.

**Rule C:** If $W(i, j)$ and $W(i, j + 1)$ are background blocks, and both $W(i, j - 1)$ and $W(i, j + 2)$ are foreground blocks, and the orientation difference between $W(i, j - 1)$ and $W(i, j + 2)$ is below the threshold $t_2$, then both $W(i, j)$ and $W(i, j + 1)$ are revised into foreground and their orientations are computed as the average orientation of $W(i, j - 1)$ and $W(i, j + 2)$. Similarly, if $W(i, j)$ and $W(i + 1, j)$ are background blocks, and both $W(i - 1, j)$ and $W(i + 2, j)$ are foreground blocks, and the orientation difference between $W(i - 1, j)$ and $W(i + 2, j)$ is below the threshold $t_2$, then both $W(i, j)$ and $W(i + 1, j)$ are revised into foreground block, and their orientations are computed as the average orientation of $W(i - 1, j)$ and $W(i + 2, j)$.

**Rule D:** If $W(i, j)$, $W(i, j + 1)$ and $W(i, j + 2)$ are background blocks, and both $W(i, j - 1)$ and $W(i, j + 3)$ are foreground blocks between which the orientation difference is below the threshold $t_3$, then $W(i, j)$, $W(i, j + 1)$ and $W(i, j + 2)$ are revised into foreground blocks and their orientation are computed as the average orientation of $W(i, j - 1)$ and $W(i, j + 3)$. Similarly, if $W(i, j)$, $W(i + 1, j)$ and $W(i + 2, j)$ are background blocks, and both $W(i - 1, j)$ and $W(i + 3, j)$ are foreground blocks between which the orientation is below the threshold $t_3$, then $W(i, j)$, $W(i + 1, j)$ and $W(i + 2, j)$ are revised into foreground blocks, and their orientations are computed as the average orientation of $W(i - 1, j)$ and $W(i + 3, j)$.

**Rule E:** If $W(i, j)$ is a foreground block, and the number of its neighbor foreground blocks is less than 3, then $W(i, j)$ is revised into background.

**Rule F:** If $W(i, j)$ is a foreground block, and there are three neighbor foreground blocks, $W(i, j)$, $W(i, j + 2)$ and $W(i + 2, j)$, satisfying one of the four conditions: (1) $i1, i2, i3 < i$, (2) $i1, i2, i3 > i$, (3) $j1, j2, j3 < j$, (4) $j1, j2, j3 > j$, then $W(i, j)$ is revised into background.

Each of rules corresponds to a procedure $\text{Rev}(r, W(i, j))$, where parameter $r \in \{A, B, C, D, E, F\}$ indicates the rule number. $\text{Rev}(r, W(i, j))$ returns TRUE if the block $W(i, j)$ satisfies the precondition of rule $r$ and returns FALSE if the block $W(i, j)$ does not satisfy the precondition of rule $r$. The procedure is as follows:

```
BOOL Rev(r, W(i, j))
{
    if W(i, j) satisfies the precondition of rule r then
        1. revising the corresponding blocks of rule r;
        2. if r \in \{A, B, C, D\}, recompute the orientations of the revised blocks;
        3. return TRUE.
    else return FALSE.
}
```

It is possible that the block $W(i, j)$ to be checked satisfies more than one of the rules. This collision is able to be dealt with by one of two strategies, wide first order and deep first order, as shown in Fig. 12. The wide first order is to go through the image block-by-block from bottom to top and from left to right, and for each block, execute the rules according to a certain sequence until the current block satisfies a rule. The deep first order is to check each image block, for each rule, to revise blocks satisfying the current rule. Let $R[i](R[i] \in \{A, B, C, D, E, F\}(0 \leq i < L))$ denote the rule number, and then $\{R[0], R[1], \ldots, R[L-1]\}$ is the rule sequence to be executed for wide first order and deep first order. If $L > 6$, there is at least one repeated rule. Suppose that there are $(p + 1) \times (q + 1)$ blocks, $p + 1$ representing the number of columns and $q + 1$ representing the number of rows. The wide first order and deep first order are as follows, respectively.

![Fig. 12. Two strategies of executing rules for segmentation revision.](image-url)
Wide first order:
WideFirstRev()
{
    Flag = TRUE; //Flag is used for controlling the termination of the loop.
    While(Flag){
        For(j = 0; j <= q; j ++)
            For(i = 0; i <= p; i ++)
                For(k = 0; k < L; k ++){
                    Flag = Rev(R[k], W(i, j));
                    If(Flag == TRUE) break;
                }
    }
}

Deep first order:
DeepFirstRev()
{
    For(k = 0; k < L; k ++){
        Flag = TRUE; //Flag is used for controlling the termination of the loop.
        While(Flag){
            For(j = 0; j <= q; j ++)
                For(i = 0; i <= p; i ++)
                    Flag = Rev(R[k], W(i, j));
        }
    }
}

Different strategies and different rule executing sequence would produce different revision result. Fig. 13 shows the different results of the segmentation revision for Fig. 9(c), one result for wide first order with the sequence to be \( \langle A, E, F, C, D, B \rangle \) and two results for deep first order with the sequence to be \( \langle A, E, F, C, D, B \rangle \) and \( \langle E, F, A, C, D, B \rangle \), respectively. In Section 7 the deep first order is used and the sequence is set to be \( \langle A, E, F, C, D, B \rangle \), experimentally.

6.2. Orientation correction

At the segmentation revision, some blocks are revised from background to foreground and their orientations are computed from their neighbor foreground blocks. Most blocks satisfying the rules from background to foreground have their orientations falsely estimated at the orientation estimation stage (Section 3). Their orientations are correctable from their neighbors which have correct orientations. However, low-pass filtering for smoothing orientation field corrects the orientation for each block from the near blocks without regarding their correctness, and is possible to revise the orientation of a block from correct to incorrect as long as incorrect orientations dominate its near blocks. The orientation correction using the neighbor correct orientations can avoid revising the orientation from correct to incorrect. Rules A, C and D use blocks \( W_1 \) and \( W_2 \), as shown in Fig. 11, to compute the orientations of the revised blocks. And rule B uses the neighbor foreground blocks to compute the orientation of the revised block. Let \( \Omega_{\text{Mean}}(i) \) be the blocks whose average orientation is computed as the orientation of the revised blocks from background to foreground. The average orientation

![Fig. 13. Segmentation revision and orientation correction using different strategy and sequence.](image-url)
OMean of $W[i](0 \leq i < F)$ is computed as follows:

$$S_x = \sum_{0 \leq i < F} \cos(2 \cdot O(W[i])), \quad (22)$$

$$S_y = \sum_{0 \leq i < F} \sin(2 \cdot O(W[i])), \quad (23)$$

$$O' = \tan^{-1}(S_y/S_x)/2, \quad (24)$$

where Eq. (25) regulates the average orientation into $[0, \pi]$.

$$OMean = \begin{cases} \pi/4, & S_x = 0, \ S_y \geq 0, \\ 3\pi/4, & S_x = 0, \ S_y < 0, \\ O' + \pi/2, & S_x < 0, \\ O', & S_x > 0, \ S_y \geq 0, \\ O' + \pi, & S_x > 0, \ S_y < 0, \end{cases} \quad (25)$$

Fig. 14. Orientation field estimated by gradient-based followed by low-pass filtering. (a) Image sensed by SecuGen device, (b) FVC2000_DB1_110_5, (c) FVC2000_DB2_55_4, (d) FVC2004_DB2_97_8, (e) FVC2004_DB2_99_8, (f) FVC2004_DB2_99_5.
7. Experimental results

The experiments use six fingerprints as shown in Fig. 14. One of them is of high quality as shown in Fig. 14(c). Two of them are of medium quality as shown in Fig. 14(a) and (b). And three of them are of poor quality as shown in Fig. 14(d)–(f). The experiments are executed on PC Intel Celeron 1200 MHz. The orientation field in Fig. 14 is by gradient-based method followed by low-pass filtering. Fig. 14 shows us that high quality regions are of correct orientations and most low quality regions are of incorrect orientations and the position of the core shifts especially in Fig. 14(a).

Fig. 15 shows the results of orientation estimation and image segmentation by the proposed systematic methods. The left column of Fig. 15 is the result of orientation estimation and correction. The medium column is the gray representation of the orientation correctness by the trained neural network. And the right column is the segmentation result with the foreground enhanced. Comparing Figs. 14 and 15, the proposed methods correct many orientations which are incorrect in Fig. 14, and the blocks with their incorrect orientations not corrected are correctly removed as background. The proposed method is a systematic scheme. The orientation computing and image segmenting operate by turns and they serve for each other. Thus it is not suitable to compare one of them separately with another corresponding method. Both orientation computation and fingerprint image segmentation serve for detecting fingerprint features. Incorrect orientation computation and image segmentation would produce spurious minutiae and lose true minutiae. The accuracy of minutiae detection is able to validate the orientation estimation and image segmentation. Therefore, we

Fig. 15. Orientation estimation and image segmentation by using the proposed scheme. (a) Image sensed by SecuGen device, (b) FVC2000_DB1_110_5, (c) FVC2000_DB2_55_4, (d) FVC2004_DB2_97_8, (e) FVC2004_DB2_99_8, (f) FVC2004_DB2_99_5. The circles mean that the circled regions have relatively better orientation estimation result than the method shown in Fig. 14 based on gradient-based method followed by low-pass filtering.
validate the proposed methods by detecting minutiae. Two minutiae detecting algorithms, A and B, are compared in the experiments. Algorithm A is based on the proposed methods followed by fingerprint enhancement in Ref. [26]. Algorithm B is the VeriFinger 4.2 published by Neurotechnologija Ltd. [27], which came up top in FVC2002 and FVC2004, in 2004. EI (error index) [26] is used for evaluating the accuracy of minutiae detection. Smaller EI value, more accurate the minutiae detection is.

\[ EI = \frac{\sum_i q_i \cdot (a_i + b_i)}{\sum_i q_i \cdot t_i} \]  

(26)

where \( q_i \) is the quality of the \( i \)th block, \( t_i \) is number of true minutiae contained in the \( i \)th block, \( a_i \) is the number of missed minutiae of the \( i \)th block, and \( b_i \) is the number of spurious minutiae, detected by algorithm, in the \( i \)th block. If regarding the quality of each block as the same, then EI is computed as

\[ EI = \frac{a + b}{t}, \]  

(27)

where \( t \) is the total number of true minutiae contained in the image, \( a \) is total number of missed minutiae, and \( b \) is the total number of spurious minutiae. In the experiments, we use the number of manually marked minutiae as the value of \( t \).

Figs. 16 and 17 show the minutiae detection results of the images in Fig. 14 using algorithms A and B, respectively. And EI values of the two algorithms to the images in Fig. 14 are listed in Table 1, and their average EI s for those images are 0.45 and 0.88, respectively. It shows that the proposed method leads to an improved accuracy in minutiae detection. Table 1 also lists the consumed time for each image in minutiae detection: algorithm A consumes a little more time than algorithm B.

It is hard to directly and quantitatively compare the orientation estimation result between algorithms A and B, for algorithm B (VeriFinger 4.2) does not provide the orientation estimation result. The estimated ridge orientation of algorithm B can be approximately inferred or suspected from the detected ridges, since the extracted ridges would certainly flow along the estimated ridge orientation. Yet the inferring
can operate only in the foreground identified by the algorithm rather than in the background, because all information is omitted in the background of the ultimate output ridge image. Fig. 18 shows an example of inferring estimated orientation from the extracted ridges which would flow along the estimated orientation: the circled region means that its estimated orientation deviates obviously away from the true ridge orientation, and this kind of deviation will introduce false ridges, and in turn, may very well produce spurious minutiae. It can be inferred that the circled regions of Fig. 17(a), (b), (e) and (f) have their ridge orientations falsely estimated, since their extracted ridges do not flow along the true orientation. Also, there are many false orientations in the output orientation image of algorithm A, as shown in Fig. 15 especially in (d)–(f). And fortunately, most those regions having the incorrectly estimated orientations are
Fig. 17. Minutiae detection using algorithm B, Neurotechnologija VeriFinger 4.2. (a) Image sensed by SecuGen device, (b) FVC2000_DB1_110_5, (c) FVC2000_DB2_55_4, (d) FVC2004_DB2_97_8, (e) FVC2004_DB2_99_8, (f) FVC2004_DB2_99_5. The circled regions are explained in Fig. 18.

Table 1
Error index of algorithms A and B to the images in Fig. 14

<table>
<thead>
<tr>
<th>Image</th>
<th>Manually labeled</th>
<th>Algorithm A</th>
<th>Algorithm B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>True</td>
<td>Spurious</td>
</tr>
<tr>
<td>(a)</td>
<td>25</td>
<td>19</td>
<td>3</td>
</tr>
<tr>
<td>(b)</td>
<td>25</td>
<td>21</td>
<td>2</td>
</tr>
<tr>
<td>(c)</td>
<td>32</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>(d)</td>
<td>22</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>(e)</td>
<td>15</td>
<td>9</td>
<td>2</td>
</tr>
<tr>
<td>(f)</td>
<td>22</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>Average</td>
<td>23.5</td>
<td>16.5</td>
<td>2</td>
</tr>
</tbody>
</table>
This kind of deviation will introduce false ridges, and in turn, may very overwhelmingly incorrect orientations, and processes only the
posed method retains the correct orientations unchanged re-
incorrect orientations after being filtered. However, the pro-
minority of correct orientations will follow the majority of
Fig. 15 (algorithm A) and Fig. 14 (algorithm C) shows that
pass filtering. It is reasonable to compare the orientation es-
low-pass filtering. Thus the errors in segmentation include two
algorithm A and algorithm B cannot be compared between al-
have been identified as background (uninterested regions). Let error_type1 denote the number of this kind of blocks, and the error_type1 values for each image and each algorithm (A and B) are listed in Table 2. For algorithm B, error_type1 value for each image is computed as the least number of blocks (of size 15 × 15) needed to cover the uninterested regions which are falsely marked as interested regions.

Aside from the above method for comparing segmentation, most existing methods compare the algorithmically segmented result with the manually segmented result to evaluate the segmentation. The manual segmentation usually classifies the visually or manually recoverable true ridge region to foreground and the rest region to background. Of course, manual segmentation is somewhat subjective. In spite of the subjectivity, it can be used as a reference evaluation method that compares the algorithmic result with the manual result, that is to count the falsely classified blocks with regard to the manual segmentation, including the algorithmic foreground blocks which are the background blocks in the manual segmentation and the algorithmic background blocks which are the foreground blocks in the manual result. We manually segment the six experimental images as in Fig. 19. Let error_type2 be defined as the number of this kind of blocks, and the error_type2 values for each image and each algorithm (A and B) are listed in Table 2. Also, we can use an alternative manual segmentation method which premises that a very competent fingerprint expert can recover ridges of very poor quality, and accordingly, simply classifies all the true ridge regions to foreground and the rest to background. Use error_type3 to represent the number of falsely classified blocks with regard to the second manual segmentation. And error_type3 values are listed in Table 2. Table 2 shows that algorithm A is better than algorithm B for error_type1, however is worse than B for error_type2 and error_type3. This is an effective evidence for supporting that fingerprint segmentation should consider the capability of feature detection algorithm and remove

![Figure 18](image_url) 

**Fig. 18.** Inferring estimated ridge orientation from the detected ridges based on the fact that detected ridges would definitely flow along the estimated orientation. The circled region means that its estimated orientation, which is inferred, deviates obviously away from the true orientation. This kind of deviation will introduce false ridges, and in turn, may very well produce spurious minutiae.

### Table 2

<table>
<thead>
<tr>
<th>Image</th>
<th>Error_type1</th>
<th>Error_type2</th>
<th>Error_type3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A B</td>
<td>A B</td>
<td>A B</td>
</tr>
<tr>
<td>(a)</td>
<td>4 6</td>
<td>42 46</td>
<td>73 2</td>
</tr>
<tr>
<td>(b)</td>
<td>0 23</td>
<td>34 52</td>
<td>25 60</td>
</tr>
<tr>
<td>(c)</td>
<td>0 0</td>
<td>16 0</td>
<td>16 0</td>
</tr>
<tr>
<td>(d)</td>
<td>0 3</td>
<td>52 33</td>
<td>112 50</td>
</tr>
<tr>
<td>(e)</td>
<td>0 16</td>
<td>120 121</td>
<td>184 170</td>
</tr>
<tr>
<td>(f)</td>
<td>9 27</td>
<td>150 90</td>
<td>170 108</td>
</tr>
<tr>
<td>Average</td>
<td>2 13</td>
<td>69 57</td>
<td>97 65</td>
</tr>
</tbody>
</table>

removal as background, except for the circled region in Fig. 16(a). If applying low-pass filtering on the background (and only on the background) of the output orientation field of algorithm A, the background blocks will have almost the same orientations in the corresponding regions of the orientation images which are shown in Fig. 14, since algorithm A has the orientation field initially computed by the gradient-based method. Let algorithm C represent the orientation estimation based on gradient-based method followed by low-pass filtering. It is reasonable to compare the orientation estimation result between A and C on foreground blocks identified by algorithm A, instead of on the whole image (including both foreground and background). Comparing between Fig. 15 (algorithm A) and Fig. 14 (algorithm C) shows that the orientation estimation result in the foreground identified by algorithm A is relatively better than in corresponding regions of algorithm C. The low-pass filtering causes that the minority of correct orientations will follow the majority of incorrect orientations after being filtered. However, the proposed method retains the correct orientations unchanged regardless of their minority or of their being surrounded by overwhelming incorrect orientations, and processes only the
those regions which are of false estimated orientations or are of so poor quality that correct ridges and minutiae cannot be extracted for the algorithm. Algorithm B keeps some manually recoverable regions not removed which, however, are not recoverable for the algorithm B, and consequently, it improves the error_type2 and error_type3 but produces more errors in minutiae detection. The smaller values of error_type1 of algorithm A than algorithm B indicate that A performs better to removes uninterested regions, especially of incorrectly estimated orientations. If it removed these regions, algorithm B would produce much less number of spurious minutiae and would be more accurate in minutiae detection.

Inevitably, manually labeling minutiae or segmenting on medium or poor quality images, especially the last three images, used in the above experiments is not strongly agreeable, for minutiae in poor quality region are often not obviously distinguishable. Therefore, we make another experiment by adding noise to high quality images, and detecting minutiae applying algorithms A and B, respec-
Fig. 21. Minutiae detection results of algorithms A and B on the noised images in Fig. 20.

Table 3
Error index of algorithms A and B to the images of Fig. 20(a1), (a2), (b1), (b2)

<table>
<thead>
<tr>
<th>Image</th>
<th>Manually labeled</th>
<th>Algorithm A</th>
<th>Algorithm B</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>True</td>
<td>Spurious</td>
<td>Missed</td>
</tr>
<tr>
<td>(a1)</td>
<td>34</td>
<td>32</td>
<td>1</td>
</tr>
<tr>
<td>(a2)</td>
<td>34</td>
<td>31</td>
<td>6</td>
</tr>
<tr>
<td>(b1)</td>
<td>35</td>
<td>35</td>
<td>4</td>
</tr>
<tr>
<td>(b2)</td>
<td>35</td>
<td>28</td>
<td>5</td>
</tr>
<tr>
<td>Average</td>
<td>34.5</td>
<td>31.5</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Applying algorithms A and B to the four noised images acquires the minutiae detection results as shown in Fig. 21, and the corresponding $EI$ comparison is listed in Table 3, where the manual labeling is done on the non-noised images, Fig. 20(a) and (b).

8. Conclusion

This paper proposes to estimate the correctness of orientation by machine learning using neural network. The correctness responded by the trained neural network to a block orientation indicates the quality of the block and whether there are ridges in the blocks or not. The correctness estimation can be used for segmentation and correcting falsely estimated orientations. A secondary segmentation method for removing the remaining ridges is also proposed. The experimental results show that the proposed systematic methods lead to an improved accuracy of minutiae detection.

References


About the Author—EN ZHU received his M.S. degree and Ph.D. degree in Computer Science from the National University of Defense Technology, China, in 2001 and 2005, respectively. He is now working in the School of Computer Science, National University of Defense Technology, China. His main research interests include pattern recognition, image processing, and information security.

About the Author—JIANPING YIN received his M.S. degree and Ph.D. degree in Computer Science from the National University of Defense Technology, China, in 1986 and 1990, respectively. He is a full professor of computer science in the National University of Defense Technology. His research interests involve artificial intelligence, pattern recognition, algorithm design, and information security.

About the Author—CHUNFENG HU is currently a graduate student in School of Computer Science, National University of Defense Technology, China. His main research interests include pattern recognition, and image processing.

About the Author—GUOMIN ZHANG is currently a Ph.D. student in School of Computer Science, National University of Defense Technology, China. His main research interests include pattern recognition, and image processing.