A Novel Preprocessing Algorithm for Fingerprint

Jin-Moon Nam, Member, KIMICS

Abstract—This paper proposes a fingerprint image processing algorithm to accurately extract minutiae in the process of fingerprint recognition. We improved the matching accuracy of low quality fingerprint images by using effective ridge vector and ridge probability. The proposed algorithm improves the clarity of ridge structures and reduces undesired noise. We collected thumb print images from 10 individuals 5 separate times each, in total using 50 thumbprints. We registered one of the five thumbprint images from each individual to match the registered one with the other four thumbprint images, and alternated the registered thumbprint image. We matched thumbprints 20 times for each individual. In total, we conducted 200 matches for the thumbprints from the 10 individuals. We improved the verification accuracy and reliability compared to conventional methods.

Index Terms—fingerprint, ridge, valley, minutiae, enhancement, preprocessing, orientation

I. INTRODUCTION

A fingerprint is a collection of ridge and valley patterns of a human fingertip. Each fingerprint has its own unique characteristics. The uniqueness of a fingerprint is exclusively determined by the local ridge characteristics and their relationships. The most important and remarkable ridge characteristics are called minutiae. A ridge ending is defined as the point where a ridge ends abruptly. A ridge bifurcation is defined as the point where a ridge forks or diverges into branch ridges[1].

Fingerprint image matching systems use these local ridge characteristics and their relationships in order to make a personal identification. The most important and critical step in fingerprint matching is for the algorithm to extract highly reliable minutiae. The performance of a minutiae extraction algorithm relies on the quality of the fingerprint images [2]. With corrupted but recoverable fingerprint images, it is essential to accurately estimate ridge orientation and frequency, as well as ridge structures [1].

Figure 1 shows fingerprint images collected from a thumbprint two times and which include corrupted but recoverable regions. The ridge structures are not well-defined. During the image processing, many spurious minutiae may be created, and genuine minutiae may be ignored due to unclear ridges [1]. An image enhancement algorithm, which makes ridge structures clear, is necessary for the minutiae extraction algorithm to accurately process minutiae on corrupted regions.

Fig.1. Corrupted but recoverable fingerprint

In ideal fingerprint images, ridges are shown clearly, and the location and the type of minutiae can be accurately extracted. However, in reality, due to variations in impression conditions (movements or pressures at the moment of fingerprinting) or different skin conditions, a significant percentage of fingerprint images is of too poor quality to be used in fingerprint recognition [1],[3]. In cases mentioned above, the ridge structures are unclear and many disconnected ridges could be created. This is also true in cases where ridges have many wrinkles; there will be
disconnected ridge structures. When a valley is wide between ridges, small dots that are defined as noise are created.

With such a low quality fingerprint image, ridge orientation and frequency estimation can be distorted. Specifically, images are seriously distorted due to large pores or noise dots between ridges. In order to correctly estimate the ridge orientation and frequency from a low quality fingerprint image, undesired noise must be removed before any further image processing is done. Figure 2 shows distorted ridges and minutiae. Two fingerprints from figure 1 are processed with a conventional algorithm to produce the thinned image. Below the center, we can find many disconnected ridges, and the area within the two small circles shows two distorted minutiae, which indicate that the types do not match.

Even though Gaussian low-pass filter removes some noise, the estimation of orientation and frequency may have some errors caused by disconnected ridges and noises [4]. Especially rapid variations in orientation and frequency of ridges around minutiae make severe amount of distortion. Unfortunately, most minutiae have such sudden curves, and the performance of a fingerprint recognition system relies heavily on the accurate estimation of ridges around minutiae. An additional vector-type filter can be used because the Gaussian low-pass filter is not enough to solve this problem [5]. The Gabor filter also needs accurate estimations of the orientation and frequency of ridges around minutiae.

The goal of an enhancement algorithm is to improve the clarity of ridge structures of fingerprint images in corrupted regions. A fingerprint algorithm should not result in any spurious ridge structures. This is very important because spurious ridge structures may change the individuality of fingerprint images [1],[3],[6].

The algorithm proposed in this paper focuses on corrupted but recoverable regions. In the following sections, we will describe the fingerprint enhancement algorithm in detail and how this uses information on local ridge orientation and frequency to improve the clarity of ridge and valley structures. In order to evaluate the performance of this algorithm, tests we have designed use some fingerprint images, which was collected in a same way as in Figure 1. These tests are described in section 3, and section 4 contains our summary and conclusion.

**II. FINGERPRINT ENHANCEMENT**

As seen in Figure 1 and 5, false orientation could be estimated in corrupted regions due to noise and disconnected ridges. In order to deal with this problem, conventional methods use a Gaussian low-pass filter[7], but an accurate orientation is not estimated. The filter merely disperses orientation noise to surrounding blocks. It weakens the high frequency characteristics around minutiae.

We need to implement new filter based on vectors. In this paper, in order to solve these problems, we first make a rough estimation of the local ridge orientation, and then we adopt a new directional filter:

1) **Rough Estimation of Local Orientation:**

After fingerprint images are obtained from a sensor, they are normalized [1] and divided into \( w \times w \) sized window \( B_{\text{raw}}(i,j) \) which is not overlapped. We estimate one orientation from a center pixel of the window \( B_{\text{raw}}(i,j) \) [1]. We use a least mean square algorithm to estimate a ridge orientation \( \theta(i,j) \) at a center pixel of each window, which covers all the pixels of the orientation window \( B_{\text{raw}}(i,j) \) [1],[3].

Using the pixel as a center, we define the image window \( G_{N\times N} \), which is normal to the local orientation \( \theta(i,j) \), and its size is \( N \times N \). This window, rotated from the normalized original fingerprint image, consists of all the pixels corresponding to
those of the original image [1],[3]. The orientation on the left in Figure 5 shows the result of this calculation. We can see that there are errors in the ridge orientation estimation.

2) Computation of ridge value: Generally, a ridge is expressed as a black line. If a ridge value is high for a pixel of fingerprint images, then the ridge is remarkable and clear. On the contrary, a valley is described as bright areas in the fingerprint images. The formula (1) shows the relationship between ridge value \( r(i,j) \) and pixel brightness \( w(i,j) \). The \( r_{\text{max}} \) means the maximum ridge value of a pixel. In 8 bit gray images, this indicates 255.

3) Computation of Referenced Ridge Value: We assume that a pixel, which is at a distance from the center pixel of window \( G_{N\times N} \), has a ridge value \( r \) whose portion is collected into the center point. The longer the distance between the two pixels is, the lower the portion of the ridge value added up into the center pixel. We define the referenced ridge value as the fraction of the neighboring ridge value collected into the point of the center pixel. A reference ratio, which decreases as the distance grows, is shown in table 1, and is defined as the reference coefficient \( K \). A referenced ridge value \( r_{\text{ref}} \) is defined as a multiplication of a ridge value \( r \) and a reference coefficient \( K \), which depends on distance.

4) Computation of Effective Ridge Vector: The center pixel in the window \( G_{N\times N} \) has a left-side line which is perpendicular to its local orientation. The line starts from the center pixel and ends at the edge of the window \( G_{N\times N} \). The sum of referenced ridge values of all the pixels on left-side line means the effective ridge \( m_{\text{left}}(i,j) \) from the left-side pixels at the center pixel. If this value is high, it represents a high ridge density within the valid left-side distance. In the same manner, we can calculate the right-side effective ridge \( m_{\text{right}}(i,j) \), which was calculated from the pixels on the right side at the center pixel of the window. The sum of left-side and right-side effective ridges are defined as an effective ridge vector. This vector plays an important role in ridge orientation estimation, finding minutiae, and elimination of noise when ridges are unclear and a fingerprint image has many noise dots.

5) Computation of Ridge Probability: A self-referenced ridge value is calculated by multiplying a ridge value \( r(i,j) \) of a center pixel with a self-reference coefficient \( K_0 \). After finding a self-referenced ridge value from a pixel, which is added by two effective ridge values, \( m_{\text{left}}(i,j) \) and \( m_{\text{right}}(i,j) \), to produce the sum, ridge probability \( p(i,j) \). A ridge probability \( p(i,j) \) of a center pixel references all the pixels on line, which are perpendicular to the window orientation of a center pixel. We limit the pixels that contribute to the referenced ridge value and the ridge probability to internal pixels of window \( G_{N\times N} \), which belongs to the center pixel.

6) Noise Reduction: The ridge probability plays an important role in fingerprint image enhancement, along with a previously calculated effective ridge vector. An effective ridge vector around noise dots is arranged as a formation towards the center of that dot. We can easily find the center of the noise dots and remove them. Corrupted ridge structures can be made clearer by using both effective ridge vector and ridge probability. When noise dots between ridges are eliminated, we can estimate a very clear ridge pattern through Gabor filtering. Figure 7 shows that noise dots between ridges were eliminated and ridge pattern was improved.

7) Ridge Emphasis by Filtering: An effective ridge vector shows the extent of its break away from the center of a ridge. When a pixel of a fingerprint image has a maximum ridge probability value, the left and right-side effective ridge vectors have almost the same scalar values that are bound by opposite orientation. Therefore, the
effective ridge vector is almost zero. Conversely, at the minimum point of a valley, a ridge probability value and effective ridge vectors are almost zero. After emphasizing ridge structures by using a ridge probability, a ridge and a valley are well defined. Figure 6 shows that the effective ridge vector improved the clarity of ridge structures.

8) Compensation of Orientation and Frequency: As seen in figures 1, corrupted regions include many errors in local ridge orientation and frequency estimation. We implemented the proposed algorithm into our fingerprint recognition system. In the above-mentioned steps 6 and 7, the algorithm eliminated noise and made ridge structures clear by using effective ridge vector and ridge probability. The orientation on the right in Figure 5 shows that the orientation of ridges is estimated correctly. The figures 8 and 9 show well-defined ridges and minutiae after noises were eliminated by using effective ridge vector and ridge probability.

Table 1. Normalized reference coefficient.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00134</td>
<td>0.002681</td>
<td>0.005362</td>
<td>0.010724</td>
<td>0.021448</td>
</tr>
<tr>
<td>2</td>
<td>0.042895</td>
<td>0.08311</td>
<td>0.128686</td>
<td>0.134048</td>
<td>0.13941</td>
</tr>
<tr>
<td>3</td>
<td>0.134048</td>
<td>0.128686</td>
<td>0.08311</td>
<td>0.042895</td>
<td>0.021448</td>
</tr>
<tr>
<td>4</td>
<td>0.010724</td>
<td>0.005362</td>
<td>0.002681</td>
<td>0.00134</td>
<td></td>
</tr>
</tbody>
</table>

In figure 6, the picture on the left is an scalar image of an effective ridge vector. The picture on the right shows the difference in ridge frequency.
after it was processed through a proposed directional filter. The color black indicates that compensated quantity is great.

Figure 7 shows two fingerprints from figure 1 processed using a proposed enhancement algorithm. The ridges below the center are comparable to those broken in Fig. 2, and two minutiae inside the two small circles show that two minutiae match in type.

In figure 7, after applying a proposed directional filter to the fingerprint image on the left, we can see that noise dots between ridges were eliminated and ridge pattern was improved. Especially around core areas, ridges are clearly well-defined.

In figure 8, the picture on the left shows the image on the left in picture 1 processed through a conventional algorithm, then Gabor filtered. The picture on the right is an image that used a proposed directional filter before Gabor filtering. This is a result of the elimination of noise dots and compensation of ridges.

In this paper, we evaluated the performance of the proposed fingerprint image preprocessing, excluding the extraction of minutiae and the elimination of spurious minutiae. The matching of types, ridge angles, and distances are all used to define the matching between two minutiae. We excluded the relative importance of each minutia in relation to each other. We judged that angles of two ridges and distances between two minutiae match if they are within a pre-described allowable error range. For types of minutiae, two minutiae have to be the same in order to consider them as matching.

III. EXPERIMENT RESULTS

The performance of the fingerprint recognition system may rely on extraction of minutiae and elimination of spurious minutiae. A fingerprint sensor also plays important role to improve the system performance, but these topics are out of the scope of this paper.

In this paper, we evaluated the performance of the proposed fingerprint image preprocessing, excluding the extraction of minutiae and the elimination of spurious minutiae. The matching of types, ridge angles, and distances are all used to define the matching between two minutiae. We excluded the relative importance of each minutia in relation to each other. We judged that angles of two ridges and distances between two minutiae match if they are within a pre-described allowable error range. For types of minutiae, two minutiae have to be the same in order to consider them as matching.
Minutiae type, a ridge orientation of minutiae, distance between minutia are sets of elements of minutia. In the same manner, a set of minutiae of a fingerprint we try to identify is defined as , and its set is also defined as . A matching rate of two fingerprint images is defined as shown in equation (13) [1],[8].

After enrolling one fingerprint image, the same fingerprint can be used in order to identify it as that of a person. In this case, its matching rate becomes 100, a maximum value.

The Images used in this test were 50 fingerprints extracted from 10 individuals, five separate times per person. An arbitrary image of one individual’s five fingerprints was enrolled, and the rest of the individual’s four fingerprints were identified separately. Following this method, 20 tests was conducted using the individual’s five fingerprints. In the same manner, 200 tests were conducted by rotating the fingerprints of all 10 individuals.

\[
f^i_r = (k^i_r, o^i_r, d^i_r)
\]

\[
M_r = (f^1_r, f^2_r, f^3_r, ..., f^n_r) N_r = n
\]

\[
f_q^j = (k_q^j, o_q^j, d_q^j)
\]

\[
M_q = (f^1_q, f^2_q, f^3_q, ..., f^m_q) N_q = m
\]

\[
\delta_{rq}(i,j) = \begin{cases} 1, & \text{if } k^i_r = k_q^j, o^i_r = o_q^j, d^i_r = d_q^j \\ 0, & \text{otherwise} \end{cases}
\]

\[
N_{\text{match}}(r,q) = \sum_{i=1}^{n} \sum_{j=1}^{m} \delta_{rq}(i,j)
\]

\[
S_{\text{match}}(r,q) = \frac{N_{\text{match}}(r,q)}{\max[N_r, N_q]} \times 100
\]

Table 2 shows the results of matching scores resulting from the above experiment. The proposed algorithm produced higher average matching scores by an average of 5.51% over the scores of the conventional one. Table 3 is the approximate amount of time that each step of the algorithm takes when they are run on the IBM personal computer.

The minutiae in figure 9 are extracted from the images, figure 1, which are created by using the proposed algorithm. The figure shows that the minutiae match better than those from figure 2 which are created from the same image, figure 1 by using conventional algorithm. Table 2 shows matching score improvements resulting from the proposed algorithm.

Table 2. Matching score

<table>
<thead>
<tr>
<th>Image Group</th>
<th>Matching Score Without Enhancement</th>
<th>Matching Score With Enhancement</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>AL1~AL5</td>
<td>46.41</td>
<td>55.73</td>
<td>+9.32</td>
</tr>
<tr>
<td>AR1~AR5</td>
<td>41.67</td>
<td>54.01</td>
<td>+12.35</td>
</tr>
<tr>
<td>BL1~BL5</td>
<td>41.81</td>
<td>53.17</td>
<td>+11.36</td>
</tr>
<tr>
<td>BR1~BR5</td>
<td>34.70</td>
<td>37.48</td>
<td>+2.78</td>
</tr>
<tr>
<td>CL1~CL5</td>
<td>46.42</td>
<td>46.74</td>
<td>+0.32</td>
</tr>
<tr>
<td>CR1~CR5</td>
<td>14.86</td>
<td>19.55</td>
<td>+4.69</td>
</tr>
<tr>
<td>DL1~DL5</td>
<td>43.66</td>
<td>44.53</td>
<td>+0.87</td>
</tr>
<tr>
<td>DR1~DR5</td>
<td>29.71</td>
<td>34.52</td>
<td>+4.81</td>
</tr>
<tr>
<td>EL1~EL5</td>
<td>61.62</td>
<td>65.70</td>
<td>+4.09</td>
</tr>
<tr>
<td>ER1~ER5</td>
<td>38.53</td>
<td>43.07</td>
<td>+4.54</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>39.939</strong></td>
<td><strong>45.45</strong></td>
<td><strong>+5.51</strong></td>
</tr>
</tbody>
</table>

Table 3. Proposed algorithm time

<table>
<thead>
<tr>
<th>Algorithm Step</th>
<th>Time (mS)</th>
<th>Percentage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normalization</td>
<td>0.160</td>
<td>16.3</td>
</tr>
<tr>
<td>Rough Orientation</td>
<td>0.020</td>
<td>2.0</td>
</tr>
<tr>
<td>Image Enhancement</td>
<td>0.125</td>
<td>12.9</td>
</tr>
<tr>
<td>Orientation Enhancement</td>
<td>0.010</td>
<td>1.0</td>
</tr>
<tr>
<td>Frequency</td>
<td>0.020</td>
<td>2.0</td>
</tr>
<tr>
<td>Gabor Filtering</td>
<td>0.644</td>
<td>65.8</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>0.979</strong></td>
<td><strong>100</strong></td>
</tr>
</tbody>
</table>

IV. CONCLUSIONS

The proposed fingerprint image preprocessing algorithm, which uses effective ridge vector and ridge probability for low quality images, shows greater performance. A lot of noises get removed and ridges are estimated more clearly. A local ridge orientation and frequency can be more accurately extracted. This result makes it possible to extract minutiae accurately through fingerprint post-processing algorithm. It reduces the probability of the appearance of spurious minutiae. The matching rate collected from 200 tests shows good results compared to results using the conventional method.

Generally, effective ridge vectors around minutiae, as shown in figure 4, is uniquely arranged. They provide a very important clue in accurate extraction of minutiae. Accurately extracting minutiae using effective ridge vectors will be a topic of next research.
REFERENCES


Jin-Moon Nam was born in Pohang, Korea, on April 26, 1964. He received the B.S. in electronics engineering from Kyungbook National University, Daegu, Korea. He received the M.S. and Ph.D. degrees in school of electrical and electronics engineering from Yonsei University, Seoul, Korea, in 2002 and 2006, respectively. He is CEO of RiverLogics Company, Bucheon, Korea. His research interests include CMOS sensor, image algorithm and embedded microcontroller design.