IEICE TRANS. INF. & SYST., VOL.E94–D, NO.9 SEPTEMBER 2011

PAPER

Practical Orientation Field Estimation for Embedded Fingerprint Recognition Systems

Yukun LIU[†a)], ***Nonmember***, **Dongju LI**[†], **Tsuyoshi ISSHIKI**[†], ***Members***, ***and*** **Hiroaki KUNIEDA**[†], ***Fellow***

SUMMARY As a global feature of fingerprint patterns, the Orientation Field (OF) plays an important role in fingerprint recognition systems. This paper proposes a fast binary pattern based orientation estimation with nearest-neighbor search, which can reduce the computational complexity greatly. We also propose a classified post processing with adaptive averaging strategy to increase the accuracy of the estimated OF. Experimental results confirm that the proposed method can satisfy the strict requirements of the embedded applications over the conventional approaches.

key words: *fingerprint recognition, orientation field, binary pattern, embedded system*

1. Introduction

A fingerprint is an orientated texture pattern constructed by ridges and valleys on the tip of an individual's finger. Its unique feature can be represented by the global pattern of ridges and valleys in forms of block-wise orientation (Orientation Field OF). In fingerprint recognition systems, OF estimation has a critical impact on almost all subsequent processes [1], [2].

Many remarkable OF estimation algorithms have been studied. The filter bank-based approach [3] can improve the OF at a local region. Ji et al. [4] proposed a binary based approach to estimate the block orientation by calculating a projective variance of a major ridge in a block ("primary ridge") against four discrete orientations as reference. However, their algorithms cannot derive a perfect OF because limited discrete orientations (8 in [3] and 4 in [4]) are not sufficient to represent a smooth ridge flow, and their computation time increases monotonically in proportional to the predefined orientation number. The model-based approach [5] considers the global constraint and regularity of the OF. However, it needs to find a tradeoff between the miss-estimation in good quality areas and prediction in low quality areas. The gradient-based approach [6] is the most broadly used method because it can provide continuous values by estimating the orientation of an image block via averaging the squared gradients to avoid directional ambiguity. However, it brings high computational costs and workspace memory requirements.

Furthermore, to compensate incorrect estimation of OF which is caused by noise, a post smoothing process is indispensable. Ji et al. [4] smooth the OF by estimating the orientations of both ridges and valleys in a image block and find the blocks with different estimated orientations, but this method cannot eliminate the noise effects and is only executable in [4] because the gradients of both ridges and valleys correspond to the same local orientations. Wang et al. [7] choose the best orientation estimated from four overlapping neighborhoods of every image block. Jain et al. [8] utilize a hierarchial scheme to adjust the estimation resolution of local OF via iterative steps. Both [7] and [8] take the coherence measure [9] as the only reliability measurement for deciding the correct orientation. They are fundamentally equivalent to averaging filters, whose drawbacks are: 1. they do not distinguish the high curvature areas and noise areas, both of which have the low coherence; 2. if the noise distribution is not symmetric with zero mean, the estimated OF obtained from averaging over the same area still suffers from strong noise effects [7]. Such high curvature areas are the neighborhood of the Singular Points (SPs) labeled with the white squares, as shown in Fig. 2 (a). The SPs can be detected from the OF using the Poincare Index Analysis [2].

This paper proposes a practical OF estimation algorithm for the embedded systems. The proposed approach contains 3 steps: 1. The preprocessing on the original fingerprint images in Sect. 2; 2. The block level OF estimation on binary images with a cost-effective nearest-neighbor search in Sect. 3, which has low cost computation without floating calculation and low workspace memory requirement for online processing under a desirable directional resolution; 3. The post processing on estimated OF with an averaging strategy which is adaptive to the ridge curvature and thus improves the accuracy of OF estimation in Sect. 4. In Sect. 5, we give a theoretical analysis in the computational aspects between the methods mentioned in the literature and the proposed method. Section 6 conducts several experiments to evaluate the accuracy, robustness and the computational complexity of the conventional methods and the proposed method.

2. Preprocessing

The original fingerprint image should be preprocessed before the OF estimation. Our preprocessing consists of image enhancement, segmentation and binarization. The image enhancement includes a high pass filter to sharpen the ridge profile and a low pass filter to remove the high frequency

Manuscript received November 26, 2010.
Manuscript revised May 6, 2011.

[†]The authors are with the Dept. of Communications and Integrated Systems, Tokyo Institute of Technology, Tokyo, 152–8550 Japan.
a) E-mail: liu.y.ad@m.titech.ac.jp

DOI: 10.1587/transinf.E94.D.1792

noise. The segmentation [10] extracts the foreground region from the filtered image to avoid unnecessary calculation on the background. The binarization converts the segmented gray image into a binary one with:

$$Bin_{i,j}(x,y) = \begin{cases} 1 & \text{if } B_{i,j}(x,y) \geq BinTH(i,j) \\ 0 & \text{else} \end{cases} \quad (1)$$

where $[x, y]$ denotes a pixel of block $B(i, j)$, output $Bin_{i,j}(x, y) = 1$ indicates the pixel value exceeds the dynamic binary threshold $BinTh(i, j)$, otherwise the output value is 0. The dynamic binary threshold $BinTh(i, j)$ means the sum of mean pixel intensity of the current block, which is obtained via:

$$BinTH(i,j) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} B_{i,j}(x,y) \quad (2)$$

where N denotes the block size for binarization.

3. Fast Orientation Estimation

The proposed orientation estimation in the block level is well calculated by employing a fast gradient estimation in the pixel level on a binary image and averaging the derived OF in the block. The result is represented by the averaged slope value of the ridge in the block. Searching for the slope values of the designated directions, which are nearest to the result of the proposed orientation estimation, the orientation result with any directional resolution is given with significant low computational expense.

3.1 Pixel Level Binary Gradient Estimation

According to the definition of the gradient in gray scale [1], if we regard the binary image as a special case of the gray one, in which the maximum range of pixel-intensity change is from 0 to 1, instead of 0 to 255, the binary pattern based gradient vector $g(x_i, y_j)$ at point $[x_i, y_j]$ is a two dimensional vector $[g_x(x_i, y_j), g_y(x_i, y_j)]$, where g_x and g_y components are the derivatives of the local block (i, j) with respect to the x and y directions, respectively. In the following of this paper, we name this vector as a binary gradient vector.

If 0 represents the ridge and 1 represents the valley, a 2×2 neighborhood binary image has $2^4 = 16$ binary pattern configurations. These configurations can be categorized into 9 types, corresponding to the phase angles 0°, 45°, 90°, 135°, 180°, 225°, 270°, 315° and *Don't care*, as shown in column 1, 2, 3 of Table 1. Since type 9 may be regarded as ridges of 45°, 135° or noise, this type is discarded. When taking the upper left pixel in a 2×2 window as the pixel of interest, the binary gradient vector of each configuration can be represented by a 4-bit binary data. By matching the binary data against the 16 binary patterns', the binary gradient vector of the given pixel with unit length can be achieved as shown in 4th column of Table 1. By overlapping the 2×2 window, all the pixels in the binary image can be labeled with one identical value from 1 to 9 indicating an identical binary gradient vector or *Don't care*.

Table 1 Properties of binary pattern configurations.

Type	Binary pattern configurations	Phase angles	Binary gradient vectors
1		0°	[1, 0]
2		45°	[1, 1]
3		90°	[0, 1]
4		135°	[-1, 1]
5		180°	[-1, 0]
6		225°	[-1,-1]
7		270°	[0,-1]
8		315°	[1,-1]
9		*Don't care*	[0, 0]

3.2 Block Level Orientation Estimation

An averaging process is necessary to obtain the block level orientations. Three problems need to be noticed before this process: 1. due to the non-linearity and discontinuity around 90°, computing θ as $\arctan(\frac{g_y}{g_x}) + 90°$ generates problems; 2. simply averaging the pixel binary gradients is not possible due to the circularity of angles, such as the average orientation of 5° and 175° is not 90° but 0°; 3. since a ridge line has two edges, the binary gradient vectors on both sides of the ridge are opposite to each other and are likely to cancel each other during the averaging process. [9] proposed an elegant solution to these problems by doubling the phase angles before averaging.

The squared binary gradient vectors $[g_{s,x}, g_{s,y}]^T$ can be derived by:

$$\begin{aligned} \begin{bmatrix} g_{s,x} \\ g_{s,y} \end{bmatrix} &= \begin{bmatrix} g\cos 2\theta \\ g\sin 2\theta \end{bmatrix} \\ &= \begin{bmatrix} g^2(\cos^2\theta - \sin^2\theta) \\ g^2 2\sin\theta\cos\theta \end{bmatrix} = \begin{bmatrix} g_x^2 - g_y^2 \\ 2g_x g_y \end{bmatrix} \end{aligned} \quad (3)$$

where g denote the pixel binary gradient vectors and 2θ is used instead of θ to discount the effect of the mentioned problems. The average squared binary gradient vectors $[\overline{g}_{s,x}, \overline{g}_{s,y}]^T$ in a window size of W can be derived by averaging the two components of the squared binary gradient vectors, respectively:

$$\begin{aligned} \begin{bmatrix} \overline{g}_{s,x} \\ \overline{g}_{s,y} \end{bmatrix} &= \begin{bmatrix} \sum_W g_{s,x} \\ \sum_W g_{s,y} \end{bmatrix} \\ &= \begin{bmatrix} \sum_W g_x^2 - g_y^2 \\ \sum_W 2g_x g_y \end{bmatrix} = \begin{bmatrix} g_{xx} - g_{yy} \\ 2g_{xy} \end{bmatrix} \end{aligned} \quad (4)$$

Commonly, the block level gradients are estimated at discrete positions to reduce the computational efforts. By dividing the binary image into equal-sized blocks of $n \times n$ pixels, the average squared binary gradient vectors can be calculated over each block independently.

To measure the reliability of the block level binary gradients, a metric called *coherence* is introduced [9], which

calculates the strength of the average squared binary gradients in the distribution of local binary gradient vectors. The coherence of a window w can be calculated as:

$$Coh = \frac{|\sum_w g_{s,x} g_{s,y}|}{\sum_w |g_{s,x} g_{s,y}|} \tag{5}$$

3.3 Nearest-Neighbor Search for Orientation Quantization

With the estimated block level binary gradients, the local orientation with rather high directional resolutions in continuous values can be achieved by the inverse trigonometric function:

$$\theta = \left[\frac{1}{2}\arctan\left(\frac{G_y}{G_x}\right) + 90^\circ\right] \mod 180^\circ \tag{6}$$

$$G_x = \sum_{i=1}^{n}\sum_{j=1}^{n} g_{s,x}^2(i,j) - g_{s,y}^2(i,j)$$

$$G_y = \sum_{i=1}^{n}\sum_{j=1}^{n} 2g_{s,x}(i,j)g_{s,y}(i,j)$$

The higher directional resolution we employ, the smoother OF we can obtain. However, high directional resolutions require high computational expenses. Furthermore, the floating point calculation involved in formula (6) cannot be applied to the embedded systems directly. To derive a high directional resolution with a low computational cost, instead of implementing formula (6) with the Look Up Table (LUT) technique, which is not so flexible and requires additional workspace memory, we propose a nearest-neighbor search for orientation quantization, which can quantize the block level orientations to any number of discrete orientations with a low computational effort.

Suppose that we quantize a vector $[u, v]$, which is orientating the continuous angle φ within $[0^\circ, 180^\circ)$, to K discrete orientations, which are represented by a reference vector $[x(k), y(k)]$ at the same lengths, where $k = 0, \ldots, K-1$. The inner product of the vector $[u, v]$ and the reference vector $[x(k), y(k)]$ corresponds to the angle differences between φ and the reference angles as:

$$\begin{aligned} V(k) &= \cos\left(\varphi - \frac{k}{K}\times 180^\circ\right) \times \sqrt{u^2+v^2} \times \sqrt{x^2(k)+y^2(k)} \\ &= u \times x(k) + v \times y(k) \end{aligned} \tag{7}$$

The desired orientation corresponds to $\max(V(0), \ldots, V(K-1))$. The quantized result can be achieved by:

$$\theta' = \left(k_{max} + \frac{K}{2}\right) \mod K \tag{8}$$

where k_{max} corresponds to the maximum $V(k)$, θ' corresponds to one discrete orientation of $0, 1, \ldots, K-1$.

Comparing with the LUT of formula (6), in which all possible values of G_x and G_y should be listed and results in a table of maximum size of $(2 \times n^2)^2 = 4 \times n^4$, our method only requires a table with the size of $2 \times K$, which lists the K predefined reference vectors.

4. Classified Post Processing

A three-step post processing scheme is proposed to smooth the estimated OF by classifying the fingerprint foreground into high curvature areas and plain areas, and applying the filters with different window size over these areas. This processing can successfully reach a good tradeoff between the accuracy in high curvature areas and the correctness in noise areas.

Step 1 Over-averaged OF estimation

The fingerprint image is divided into a series of non-overlapped blocks size of $n \times n$ pixels. When averaging the squared binary gradient vectors $[g_{s,x}, g_{s,y}]^T$ for a image block, the square blocks are equally enlarged from $n \times n$ to $n_1 \times n_1$ and formulae (4) (7) (8) are applied for the consequent calculations. A sufficiently large window size n_1 is utilized to average $[g_{s,x}, g_{s,y}]^T$, and obtain an over-averaged OF. Two major objectives can be achieved in the over-averaged OF: 1. the accuracy of the high curvature areas is low, but the SP patterns are reserved; 2. the noise effects are depressed, and the orientations are smoothed, although they might not be so accurate yet.

Step 2 Adaptive averaging

The squared binary gradients are averaged again with different window sizes for the high curvature areas and the plain areas. In the high curvature areas, the orientation of a block of interest is not consistent with its surrounding neighbors. The orientation of a block here can be taken as a gradient vector of unit length. Thus, Eq. (5) can be applied to calculate a coherence map for the over-averaged OF using a window size of 3×3 blocks. The value of coherence is a quantitative representation of the orientation consistency. We specify the high curvature areas by setting a coherence threshold TH. Those areas in which the coherence value are lower than TH are defined as the high curvature areas, other areas are considered as the plain areas. In our experiments, TH is set to 0.8.

The averaging window size $n_2 \times n_2$ is applied to the high curvature areas and $n_3 \times n_3$ is applied to the plain areas, where $n < n_2 < n_3$. This adaptive averaging process eliminates the zero-mean noise effects and reserves the accuracy of the high curvature areas.

Step 3 One dimensional low pass filter

The non-zero mean noise effects in the adaptive averaged OF appear as mutational orientations. It means that the orientation of a certain block is significantly different from its neighboring blocks. The simple enlargement of the window size will affect the neighboring blocks, and cannot eliminate the noise effects completely. Therefore, we introduce a one dimensional low pass filter to smooth the mutational orientations without affecting the neighboring blocks by minimizing the orientation variances from the block of interest to its neighborhoods, as shown in Fig. 1.

In the case of horizontal processing, the orientation variances from the block of interest (i, j) to its previous block $(i-1, j)$ and next block $(i+1, j)$ are denoted as D_p and D_n, which are calculated with:

$$\begin{pmatrix} D_p \\ D_n \end{pmatrix}_{ij} = \begin{pmatrix} \text{Difference}(\theta_{ij}, \theta_{(i-1)j}, K) \\ \text{Difference}(\theta_{(i+1)j}, \theta_{ij}, K) \end{pmatrix} \tag{9}$$

where D_p and D_n are in the range of $(-\frac{K}{2}, \frac{K}{2}]$. The orientation variance between two quantized angles p and q in the congruence class modulo K can be derived by:

$$\text{Difference}(p, q, K) = \begin{cases} p-q & \text{if } -\frac{K}{2} < p-q \le \frac{K}{2}; \\ p-q+K & \text{if } p-q \le -\frac{K}{2}; \\ p-q-K & \text{else.} \end{cases} \tag{10}$$

The result local orientation can be achieved with:

$$\theta'' = \begin{cases} \left(\theta' + \frac{D_n - D_p}{2}\right) \mod K & \text{if } D_p < 0, D_n \ge 0 \\ & \text{or } D_p \ge 0, D_n < 0; \\ \left(\theta' + \frac{D_n + D_p}{2}\right) \mod K & \text{else.} \end{cases} \tag{11}$$

The one dimensional filter is processed twice on the estimated OF from left to right, up to down, respectively.

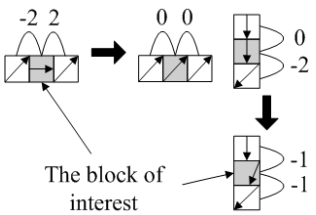


Fig. 1 One dimensional low pass filter.

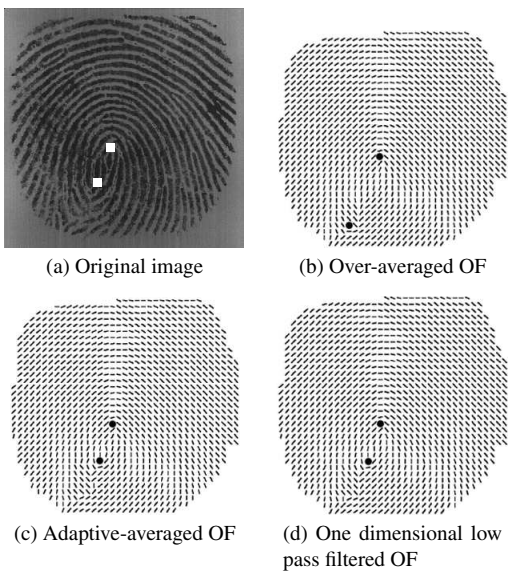


Fig. 2 An example of the classified post processing. $n = 8$, $n_1 = 72$, $n_2 = 24$, $n_3 = 56$.

An example of the classified post processing is shown in Fig. 2. In Fig. 2 (a), the manually inspected SP locations are highlighted with white squares. In Fig. 2 (b) (c) (d), the black circles are used to represent the SP locations. By comparing these figures, it can be observed that the over-averaged OF is smooth, while there is significant position shift of the SP location, see Fig. 2 (b). By applying the adaptive averaging process, the position shift becomes minor in Fig. 2 (c). However, some blocks remain unsmooth. The one dimensional low pass filter solves the unsmooth blocks without affecting the SP locations and the neighboring blocks as shown in Fig. 2 (d).

5. Computational Aspects

This section gives a theoretical analysis of computational complexity for two literature mentioned methods [4], [7] and the proposed method that may be suitable for the embedded systems.

Assume that [4], [7] and the proposed method are using the same preprocessing technique, the binarization is a byproduct of highpass filtering, in which the mean pixel intensity of each block will be calculated. This is beneficial for the two binary image based methods. For a fingerprint with the size of R rows and C columns, the computational complexity of the proposed algorithm is approximately $O(R \times C)$ in LUT searching for binary gradient estimation. The quantization and each step in the post processing takes $O(R \times C \times K/n^2)$, which sums up to about $O(R \times C \times (1 + 5 \times K/n^2))$ for the proposed algorithm, where K denotes the predefined discrete orientation for quantization and n denotes the block size for OF estimation. The gradient-based method [7] has the similar complexity with respect to R and C for pixel gradient calculation. However, it is carried out with summation and subtraction on pixel intensity values, which results in a much higher computational cost. In the implementation of this paper, the pixel gradient is calculated with the Sobel mask for the method proposed in [7], with a computational complexity of $O(3^2 \times R \times C)$ after parallel optimization and totally $O(R \times C \times (9 + 2 \times K/n^2))$. For the algorithm proposed in [4], the computational complexity is $O(R \times C \times m_1 \times m_2)$ for primary ridge determination, where m_1 and m_2 are the sizes of the two dimensional pulse coupled neural network, and $O(R \times C \times K^2/n^2)$ for ridge projection. These will sum up to $O(2 \times R \times C \times (m_1 \times m_2 + K^2/n^2))$ because it processes twice (ridge and valley) for the purpose of orientation correction. A comparison of the computational complexity of each step for the three methods is listed in Table 2 for fixed parameters of $m_1 = m_2 = n = 8$ and

Table 2 The computational complexity comparison of 3 methods. $K = 32$, $m_1 = m_2 = n = 8$.

	[4]	[7]	Proposed
Pixel Level	—	$O(9R \times C)$	$O(R \times C)$
Block Level	$O(64R \times C)$	$O(0.5R \times C)$	$O(0.5R \times C)$
Post Processing	$O(0.5R \times C)$	$O(0.5R \times C)$	$O(2R \times C)$
Overall	$O(129R \times C)$	$O(10R \times C)$	$O(3.5R \times C)$

$K = 32$. It is obvious that method in [4] requires much more computational efforts than the other two methods mentioned above although its processing is carried on binary images.

6. Experimental Results

In this section, the proposed approach is compared with [4], [7] by three general criterions: the average computation time and workspace memory requirement, the orientation estimation accuracy and the performance improvement of fingerprint recognition [2], [11]. Three experiments are carried out. The first experiment applies the proposed algorithm to several fingerprint images and evaluates the accuracy and robustness of orientation estimation. The second experiment evaluates the OF estimation in terms of a fingerprint recognition system. In the third experiment, the computational expenses of different algorithms are evaluated according to the requirements of the embedded applications.

6.1 Accuracy and Robustness

The proposed method is applied to fingerprint images of FVC2002 DB3 [1], which is constructed with 100 fingers, 8 impressions per finger. Three of the estimated OF results in different stages with normal, low and extremely low quality images including whorl, arch and loop are shown in column 1–4 of Fig. 3. The estimated OF results are quantized to 16 orientations. For the normal quality image 35_5 and low quality image 99_4, the proposed algorithm can produce accurate OF after the post processing. For the extremely low quality image 95_8, the coarse OF is with lots of unrecoverable noise effects. The proposed post processing cannot produce accurate OF. The corresponding OF results of [4] are shown in column 5 of Fig. 3. Since the OF block size implemented in this experiment is 8×8, while Ji et al. use 16×16, [4]'s results are worse because the primary ridge cannot be robustly detected in such a small block. Furthermore, four directional resolution is not sufficient to reserve the precision of a continuous ridge flow.

Since there is no ground truth exists for fingerprint OF, the OF estimation algorithms are often evaluated by human inspection [5]. In this experiment, we regard the human inspected orientation as the ground truth orientation. To evaluate the orientation estimation accuracy, we derive the orientation difference between the human inspected results θ''' and the algorithm-estimated results θ'' by:

$$\Delta\theta = |\text{Difference}(\theta''', \theta'', K)| \tag{12}$$

Figure 4 shows the statistical distribution of $\Delta\theta$ for the images listed in Fig. 3. It can be observed that the orientation differences in all images are reduced after the proposed post processing. The average $\Delta\theta$ of different images are

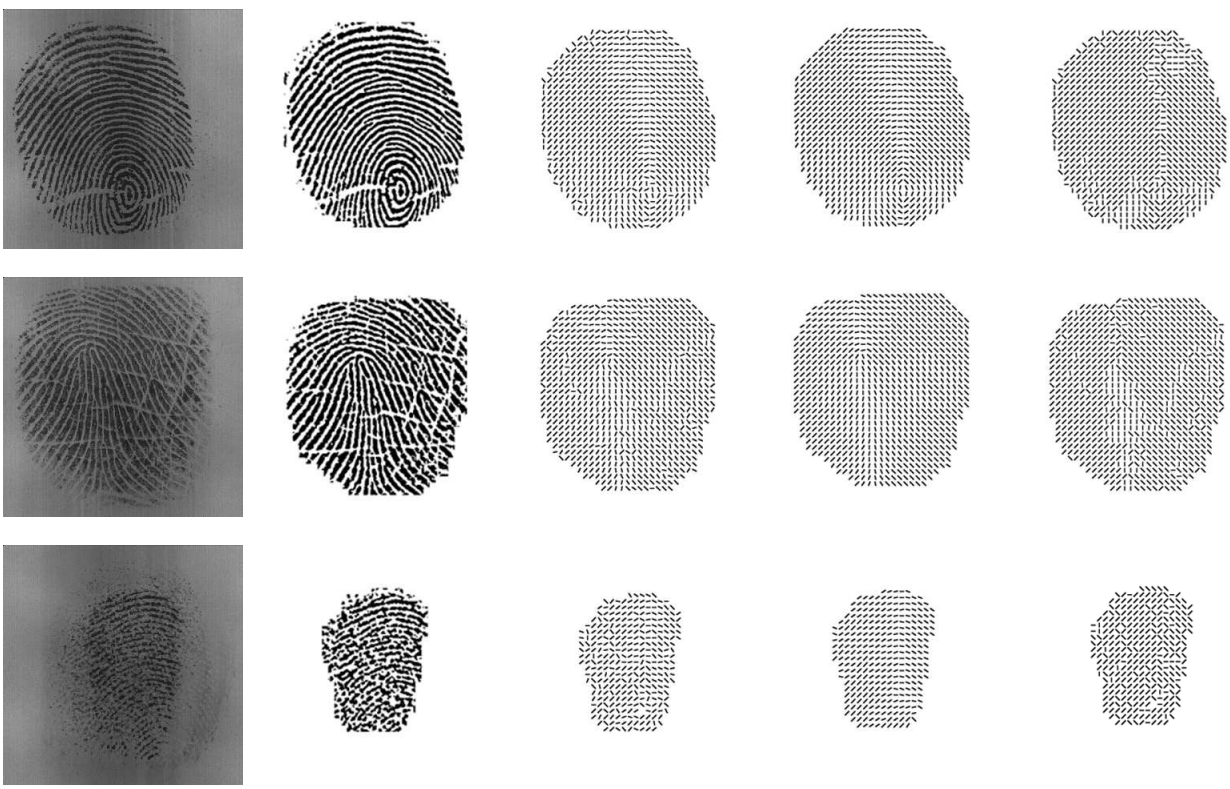
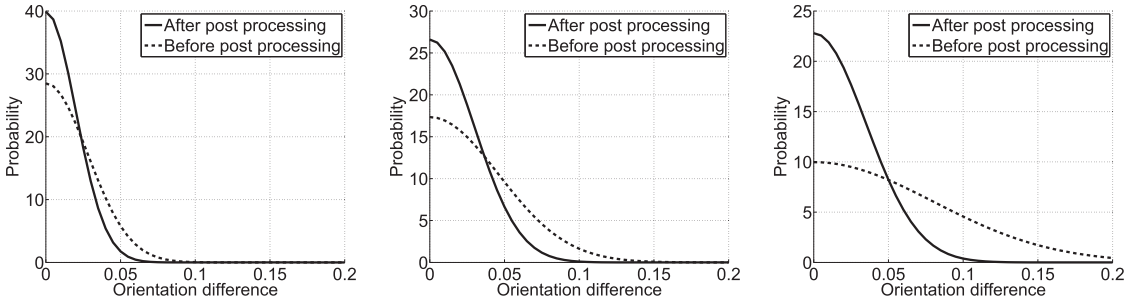
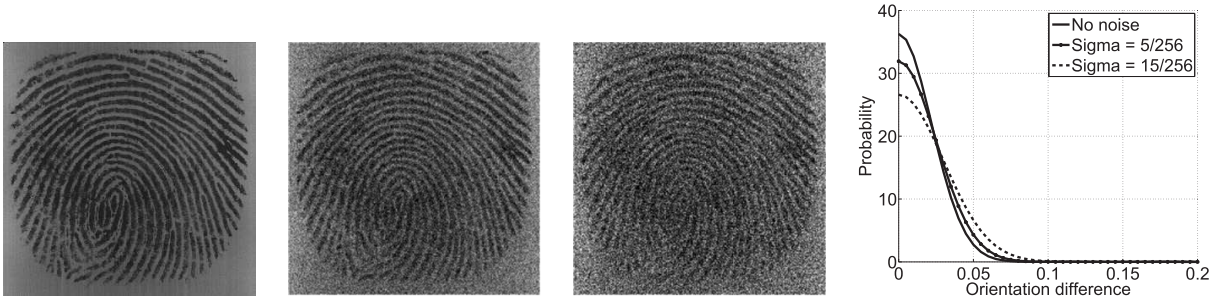


Fig. 3 Processing results of [4] algorithm and the proposed method on fingerprints with different quality levels. Each row shows normal, low and extremely low quality images, respectively. Each column corresponds to original, binary, coarse block level OF, smoothed OF with 16 directional resolution and [4]'s result with 4 directional resolution, respectively. $n = 8$, $n_1 = 72$, $n_2 = 24$, $n_3 = 56$, $TH = 0.8$.



(a) 35_5 (b) 99_4 (c) 95_8

Fig. 4 The statistical distributions of the orientation difference before and after the proposed post processing.



(a) Original image (b) $\sigma = 5/256$ (c) $\sigma = 15/256$ (d) The statistical distributions of orientation difference

Fig. 5 The Gaussian noise fingerprint samples and the statistical distributions of orientation difference against variant Gaussian noises.

2.55°, 4.20° and 7.02° before the proposed post processing, 1.82°, 2.74° and 3.19° after the proposed post processing, respectively.

To evaluate estimation robustness against low quality images, we add Gaussian noise into the original fingerprint images and test the accuracy of OF estimation. The variances of the zero-mean Gaussian noise adopted are $\sigma = 5/256$ and $\sigma = 15/256$. The noise samples are illustrated in Fig. 5 (b) (c). Figure 5 (d) gives the experimental results with respect to different Gaussian noise. It can be observed that Gaussian noise would depress the accuracy of OF estimation; however, the influence of noise on the proposed algorithm is insignificant. The average $\Delta\theta$ for (a) (b) (c) of Fig. 5 are 2.01°, 2.27° and 2.71°, respectively.

In order to obtain the statistical performance comparison results on large database, we construct a subset with the first fingerprint image of each finger in FVC2002 DB3, which contains 100 images. The average $\Delta\theta$ of different algorithms in [4], [7] and our method are 18.31°, 5.16° and 2.53°, respectively.

6.2 System Performance

For the system performance of fingerprint recognition, generally better OF estimation algorithm can cause more recognition performance improvement. To investigate the performance improvement of different algorithms, the algorithms of [4], [7] and the proposed method are implemented into a minutiae alignment based fingerprint recognition system [12], respectively. As shown in Fig. 6, during the fingerprint recognition, three systems share the same smoothing, segmentation, ridge enhancement, thinning, minutiae extraction and minutiae matching process. Since only the OF estimation step is different in these systems, the system performance comparison can be fairly derived from their recognition results. The algorithms are applied to fingerprint images of FVC2002 DB3. Each matching process generates $8\times8\times100/2 = 3{,}200$ genuine and $800\times99\times8/2 = 316{,}800$ impostor pairs. Since the image size is 300×300 pixels, and the OF block size is 8×8 pixels, there are 37×37 blocks in the estimated OF. The estimated OF of the proposed method is quantized to resolution 8, 16 and 24, respectively. The system evaluation results are presented in forms of the receiver operating character (ROC) curves. The ROC curves are plotted as the false accept rate (FAR) against the false reject rate (FRR).

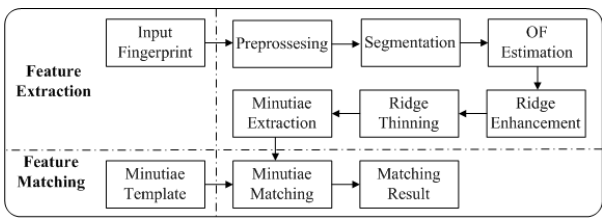


Fig. 6 The flow chart of the fingerprint recognition system.

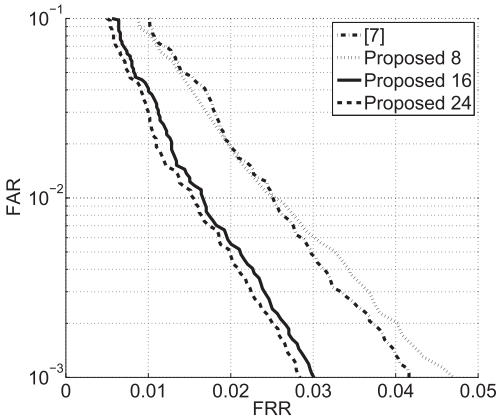


Fig. 7 System performance comparison on FVC2002 DB3.

Since [4]'s method is not so robust in a small block size and four directional resolution is not sufficient to represent a smooth ridge flow, the Equal Error Rate (EER) result of [4] system is up to 8.26%. Therefore, we only list the ROC curves of [7] system and the proposed one with directional resolution 8, 16 and 24, respectively. As shown in Fig. 7, the proposed algorithms with quantization resolution of more than 16 produce similar results to each other because the fixed block size limits the actual directional resolutions. As the directional resolution decrease from 16 to 8, the recognition result became worse. This is because the lower directional resolution lost precision of OF representation. It can be observed that for the same database, the proposed algorithm with directional resolution 24 causes the most improvement. On a whole, with a same FAR, the proposed algorithm can help the system to obtain the lowest FRR among the algorithms. Statistically, compared with the other systems, the EER improves from 8.26% in [4] and 1.87% in [7] to 1.42% and 1.35% in the proposed method with directional resolution 16 and 24, respectively.

6.3 Computation Time and Memory Capacity

As we analyzed in Sect. 5, the algorithms of [4], [7] require much more computational costs than the proposed one. We give an overall comparison between [4], [7] and our algorithm under a PC platform as listed in Table 3. The results listed in the table coincide with the theoretical analysis of computational complexity. Furthermore, it can be observed that the proposed method with directional resolution 16 is the most promising one for the embedded implementation, when the accuracy of orientation estimation is also taken into consideration.

We set up a fingerprint SoC with bit serial FPGA engine in our previous work [11]. The system chip includes a 64 KB ROM in which the fingerprint recognition algorithm is embedded. The 32-bit RISC processor works at 200 MHz frequency with 8 KB data cache, 8 KB instruction cache and memory protection unit. The calculation of 8×8 pixels OF requires less than 3 k words workspace memory, which allows the proposed algorithm to execute in the data cache of the processor. The average computation time of the OF estimation in the embedded environment is 15 ms.

Table 3 Average computation time and workspace memory requirement in Pentium 4, 2.33 GHz, 2 GB memory PC. Proposed 8, 16, 24 stand for the proposed algorithm with directional resolutions of 8, 16 and 24, respectively.

Algorithm	Time (ms)	Memory (words)
Proposed 8	3	3 k
Proposed 16	3	3 k
Proposed 24	4	3 k
[4]	130	10 k
[7]	43	28 k

7. Conclusions

This paper proposes a practical binary pattern based OF estimation approach for embedded fingerprint recognition systems. Experimental results show that the proposed method can reduce the average computation time down to less than 7% of the existing methods, and the workspace memory requirement is reduced to less than one third as small as the other methods. Furthermore, the proposed post processing can achieve more accuracy and robustness of the OF estimation. These advantages of the proposed algorithm over the conventional methods make implementation easier for the embedded applications.

References

[1] D. Maltoni, D. Maio, A.K. Jain, and S. Prabhakar, Handbook of Fingerprint Recognition, Springer Verlag, New York, 2003.
[2] A.M. Bazen and S.H. Gerez, "Systematic methods for the computation of the directional fields and singular points of fingerprints," IEEE Trans. Pattern Anal. Mach. Intell., vol.24, no.7, pp.905–919, 2002.
[3] L. Ogorman and J.V. Nickerson, "An approach to fingerprint filter design," Pattern Recognit., vol.22, no.1, pp.29–38, 1989.
[4] L.P. Ji and Z. Yi, "Fingerprint orientation field estimation using ridge projection," Pattern Recognit., vol.41, no.5, pp.1491–1503, 2008.
[5] J. Zhou and J.W. Gu, "A model-based method for the computiation of fingerprints' orientation field," IEEE Trans. Image Process., vol.13, no.6, pp.821–835, 2004.
[6] A.R. Rao, A Taxonomy for Texture Description and Identification, Springer Verlag, New York, 1990.
[7] Y. Wang, J. Hu, and F. Han, "Enhanced gradient-based algorithm for the estimation of fingerprint orientation fields," Appl. Math. Comput., vol.185, no.2, pp.823–833, 2007.
[8] A. Jain and L. Hong, "On-line fingerprint verification," IEEE Trans. Pattern Anal. Mach. Intell., vol.19, no.4, pp.302–314, 1997.
[9] M. Kass and A. Witkin, "Analyzing oriented patterns," Comput. Vis. Graph. Image Process., vol.37, pp.362–385, 1987.
[10] A.M. Bazen and S.H. Gerez, "A segmentation of fingerprint images," Proc. ProRISC2001, 12th Ann. Workshop Circuits, Systems and Signal Processing, Nov. 2001.
[11] Y. Wang, D. Li, T. Isshiki, and H. Kunieda, "A novel fingerprint SoC with bit serial FPGA engine," IPSJ Digital Courier, vol.1, pp.226–233, 2005.
[12] S.C. Dass and A.K. Jain, "Fingerprint-based recognition," Technometrics, vol.49, no.3, pp.262–276, 2007.

Yukun Liu received the B.E. and M.E. degrees in from Harbin University of Science and Technology, China, in 2000 and 2003 respectively. She worked as a junior lecture in Harbin University of Science and Technology, China, from 2003–2007. She is presently pursuing Ph.D. degree in Dept. of Communications and Integrated Systems at Tokyo Institute of Technology. Her research interests are pattern recognition and machine learning.

Dongju Li received B.E. degree from LiaoNing University and M.E. degree from Harbin Institute of Technology, China, in 1984 and 1987, respectively. She worked as an IC design engineer in VLSI Design Laboratory of Northeast Micro-electronics Institute, Electronic Industry Bureau, China, from 1987–1993. She received Ph.D. degree from Tokyo Institute of Technology in 1998. She is now a Research Associate at Dept. of Communications and Integrated Systems at Tokyo Institute of Technology. Her research interests are fingerprint authentication, multiprocessor architecture and VLSI design.

Tsuyoshi Isshiki received the B.E. and M.E. degrees in electrical and electronics engineering from Tokyo Institute of Technology in 1990 and 1992, respectively. He received his Ph.D. degree in computer engineering from University of California at Santa Cruz in 1996. He is now an Assistant Professor at Department of Communications and Integrated Systems in Tokyo Institute of Technology. His research interests include high-level design methodology for configurable systems, bit-serial synthesis, FPGA architecture, image processing, fingerprint recognition, computer graphics and speech synthesis.

Hiroaki Kunieda received B.E., M.E. and D.E. degrees from Tokyo Institute of Technology in 1973, 1975 and 1978, respectively. He was Research Associate in 1978 and Associate Professor in 1985 at Tokyo Institute of Technology. He is now a Professor at Dept. of Communications and Integrated Systems in Tokyo Institute of Technology. He has been engaged in researches on distribute circus, switched capacitor circus, IC circus simulation, VLSI CAD, VLSI signal processing and VLSI design. His current research focus on fingerprint authentication, VLSI multimedia processing, design for system-on-chip, VLSI signal processing, VLSI architecture, and VLSI CAD. He is a member of IEEE CAS & SP society and IPSJ.