

# A Low Complexity Fingerprint Verification Algorithm with Reduced Hardware Resources<sup>\*</sup>

JIA-HONG DAI AND CHING-HAN CHEN<sup>+</sup>

*Department of Electrical Engineering*

*I-Shou University*

*Kaohsiung, 840 Taiwan*

<sup>+</sup>*Department of Computer Science and Information Engineering*

*National Central University*

*Taoyuan, 320 Taiwan*

A low complexity fingerprint verification scheme with ultra-compact hardware resources requirement is presented in the paper. Firstly, the fingerprint image is captured and performed noise canceling by a binary low-pass filter. Secondly, the displacement of the fingerprint pattern is estimated by means of directional image matching, and then circular profile features of the fingerprint are extracted for verification matching. Finally, the proposed method is implemented in an 8-bits microcontroller run at 25MHz, equipped only with 12 Kbytes program code memory and 35 Kbytes data memory. It is very favorable for prototyping the low-cost fingerprint verification SoC (System-On-Chip).

**Keywords:** biometrics, fingerprint, verification, embedded system, SoC, 8051

## 1. INTRODUCTION

The biometric system is a pattern recognition system, which automatically differentiates people on the basis of individual biometric features, such as voice, hand geometry, fingerprint and iris patterns of the eyes. It generally includes three components: data acquisition, feature extraction and decision-making. The biometric data may come from specific sets of physiological (like fingerprint, face, *etc.*) or behavioral characteristics (voice, gait, *etc.*). The integration of biometrics for person authentication system provides the advantages of increasing the system's security and reliability [1].

A biometric system can be run in two modes [2]: identification or verification. Identification is the process of finding out an identity from a set of enrolled template patterns. A pattern may be identified through matching the input pattern with every enrolled template. According to the similarity between the pattern and the enrolled template, a matching score is obtained. The template with the maximal matching score is assigned as the identified person.

As for the verification, it is the process to verify the input pattern with a template of which the person's identity has been claimed a priori. Like the identification, the similarity between the input pattern and the enrolled template is checked to provide the exclusive Yes/No output.

---

Received October 3, 2006; revised February 7 & April 20, 2007; accepted June 27, 2007.

Communicated by Pau-Choo Chung.

<sup>\*</sup> The earlier version of this paper was presented at IEEE International Symposium on Consumer Electronics (ISCE), Macau, June 14-16, 2005.

Among different biometrics techniques, fingerprint is the most widely used because of its uniqueness, ease of collection and public acceptance [3]. Since 1960, a large number of criminal and civilian fingerprint-based applications are studied and developed. In these cases, the unknown fingerprint has to be recognized among numerous fingerprint images stored in the database. These systems generally adopted identification mode. On the contrast, for most portable security applications like pen drive, mobile phone, PDA and smart card, the verification mode might be more attractive than identification mode due to the nature of individualized use. The fingerprint could serve as a secret key which is not likely to be stolen.

Most fingerprint recognition systems rely on minutiae-based representation. Minutiae are specific points in a finger image. There are two main types, known as ridge endings and bifurcations [4]. An excellent review of minutiae-based fingerprint verification method can be found in Yager and Amin [5]. This method is characterized by high recognition precision as well as high algorithmic complexity. Therefore, it is generally implemented in a desktop PC or a high performance processor. Its integration in a portable application system constitutes a big challenge because of the strict constraints related to size, power and performance of the processor. But we still can find embedded fingerprint verification systems. Yang and Verbauwhede [7] have developed an embedded fingerprint verification system for a "ThumbPod" device. They implement the complex signal preprocessing step on a powerful card reader, and only the matching step is executed on the embedded device. With specific hardware acceleration and memory optimization, the minutiae extraction and the matching process can be executed in a 50MHz LEON-2 processor with 483KB memory used. In the work of Pan *et al.* [8], the fingerprint verification is implemented in a 32-bit ARM7TDMI with 64KB ROM, 8KB RAM and 32KB EEROM for smart card application. Tang *et al.* [9] use a 206 MHz StrongARM-based PDA to implement the fingerprint verification algorithm. Rikin *et al.* [10] developed a 100MHz DSP board with 32KB memory.

In this paper, we devote to elaborate a new fingerprint verification algorithm in which the computational complexity and memory requirement are extremely reduced. Aiming at the portable security applications, the proposed algorithm is more adequate than conventional methods for implementation with restricted hardware resources.

The rest of this paper is organized as follows. Section 2 reviews the conventional minutiae-based method as a comparative fingerprint verification algorithm. Section 3 explains the proposed fingerprint verification algorithm. Section 4 describes the implementation details using an 8051 microcontroller with reduced memory requirement. The performance evaluation and comparisons are presented as well in this section. The conclusions are given in section 5.

## 2. MINUTIAE-BASED FINGERPRINT VERIFICATION

The typical minutiae-based fingerprint verification system generally consists of three fundamental modules [4, 11, 12]: (1) pre-processing module which includes image enhancement, binarization, and thinning process for the refinement of acquired fingerprint image; (2) minutiae extraction module which locates all the branching points and bifurcation points in the fingerprint image; and (3) minutiae matching module which

compares the extracted minutiae set with those of enrolled template. At the module of pre-processing, the directional band-pass Gabor filter-bank approach is one of the most effective and mathematically elegant techniques to date for fingerprint image enhancement. It was proposed for the first time by Hong, Wan, and Jain [11]. Gabor filters have both frequency-selective and orientation-selective properties in frequency domain. It is composed of a set of coefficients in two-dimensional structure to remove effectively the noise and preserve parallel ridge/valley pattern. Minutiae are extracted generally from thinned skeleton images [13, 14]. Before this process, the binarization on enhanced gray image and thinning operation have to be executed first. The information of extracted minutiae includes location, orientation and type of minutiae. It will be used for the follow-up matching procedure.

Minutiae matching consists of two processes which are alignment and minutiae pairing [15]. In order to match the extracted minutiae set with unknown direction and displacement, we must try to find the differences of direction and displacement between the template fingerprint and the one to be verified. In this stage, the affine transforms such as translation and rotation are estimated. According to the estimated parameters, two minutiae sets are aligned [16]. Once the alignment is completed accurately, matching stage consists in computing the Euclidian distance of two point patterns. A matching score is then obtained.

We resume the minutiae-based fingerprint verification scheme in Fig. 1.

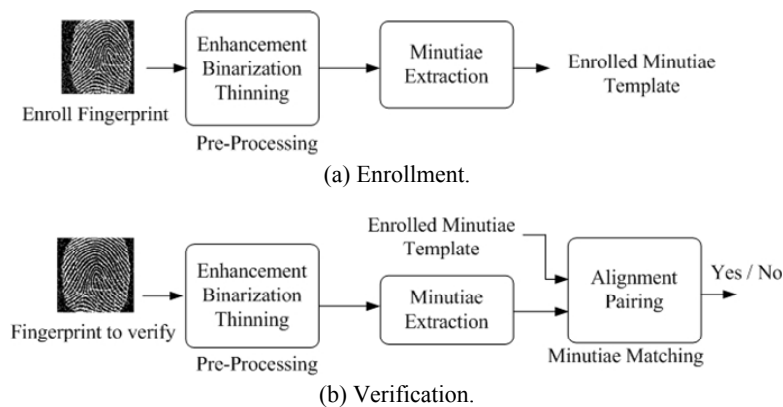


Fig. 1. Minutiae-based fingerprint verification scheme.

### 3. LOW COMPLEXITY FINGERPRINT VERIFICATION SCHEME

The minutiae-based fingerprint verification algorithm described in section 2 possesses high computational complexity and hardware resources requirement. It is disadvantage for resource-constraint portable application. Aiming at this problem, we propose a low complexity fingerprint verification scheme as shown in Fig. 2. It has the objectives of reducing the computational complexity and minimizing the memory requirement at sufficient recognition performance.

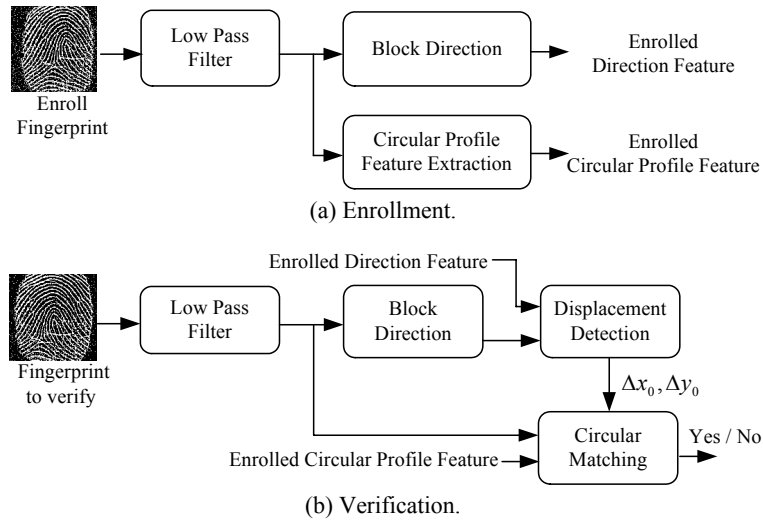


Fig. 2. Proposed fingerprint verification scheme.

We use the fingerprint sensor AFS2 from AuthenTec Co. [18] whose resolution is 250 DPI with image size  $128 \times 128$  pixels, and each pixel has eight gray-levels represented by three bits. The sensor's output can be adjusted as one-bit output for the purpose of efficiency and complexity reduction.

The proposed fingerprint verification scheme ignores, at the first stage, the ridge and rotation characters to find out a coarse displacement from the direction feature. According to the displacement, the circular matching module considers the details of fingerprint ridge to perform more accurate matching.

### 3.1 Low Pass Filter

The original image  $I(x, y) \in \{0, 1\}$ ,  $0 \leq x, y < W$  from fingerprint sensor AFS2 is a binary image (see Fig. 3 (a)). A multiplierless low-pass filter is used to reduce high frequency noises and to enhance the ridge pattern of a fingerprint image. The filtered image is obtained by:

$$L(x, y) = \begin{cases} 1, & \text{if } (\text{sum}(x, y) > T) \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

$$\text{sum}(x, y) = \sum_{x_h = -w_L}^{w_L} \sum_{y_h = -w_L}^{w_L} I(x + x_h, y + y_h) \quad (2)$$

where  $(2w_L + 1)$  is the filter size,  $T$  is a given threshold, and  $L(x, y)$  is the output filtered image (see Fig. 3 (b)). The '1' represents the ridge (marked black), and '0' the valley (marked white).

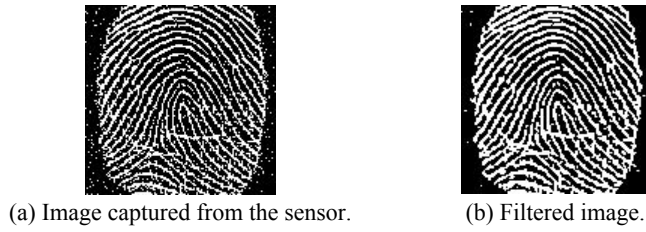


Fig. 3. Low pass filtering.

### 3.2 Feature Extraction

In the proposed scheme, the fingerprint feature is represented by block direction image and circular profile in order to reduce computational complexity of conventional minutiae extraction procedure.

#### 3.2.1 Block direction computing

This process separates  $L(x, y)$  into sub-blocks with  $b \times b$  pixels. Considering a pixel block structure where the target pixel  $X = L(x, y)$  with eight nearest surrounded  $X_0, X_1, \dots, X_7$  are arranged as shown in Fig. 4.

$X_0$	$X_1$	$X_2$
$X_7$	$X$	$X_3$
$X_6$	$X_5$	$X_4$

Fig. 4. Pixel block structure.

We use a set of logic Eqs. (3)-(6) to compute the directions of each target pixel.

$$d_0 = X \cap X_7 \cap X_3 \tag{3}$$

$$d_1 = X \cap X_1 \cap X_5 \tag{4}$$

$$d_2 = X \cap X_2 \cap X_6 \tag{5}$$

$$d_3 = X \cap X_0 \cap X_4 \tag{6}$$

$d_i \in \{0, 1\}$  represents whether there is direction  $i$  of pixel.  $i \in \{0, 1, 2, 3\}$  is the index of four directions ( $0^\circ, 90^\circ, 45^\circ, -45^\circ$ ). Then direction of each block can be obtained by:

$$D(u, v) = direction(\max A_i(u, v)), \quad 0 \leq u < w, \quad 0 \leq v < w, \quad w = \left\lfloor \frac{W}{b} \right\rfloor \tag{7}$$

and

$$A_i(u, v) = \sum_{u_b=0}^{b-1} \sum_{v_b=0}^{b-1} d_i(u \cdot b + u_b, v \cdot b + v_b) \tag{8}$$

where  $direction(\cdot)$  is the direction  $i$  of the block.

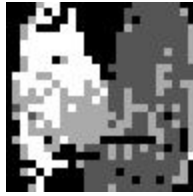


Fig. 5.  $32 \times 32$  pixels block direction image.

Fig. 5 shows the block direction image obtain from Fig. 3 (b) when  $b = 4$ . Because the block size is  $b \times b$ , the direction image area is reduced to  $1/b^2$  of the source image area. So it benefits the reduction of the computational operation of the succeeding displacement detection procedure.

### 3.2.2 Circular profile feature extraction

This process extracts circular features of a fingerprint image by  $m$  co-centric circles. The circular features can be manipulated as a 1-D array in memory. It can therefore avoid complex memory addressing operation such as multiply-addition or shifting.

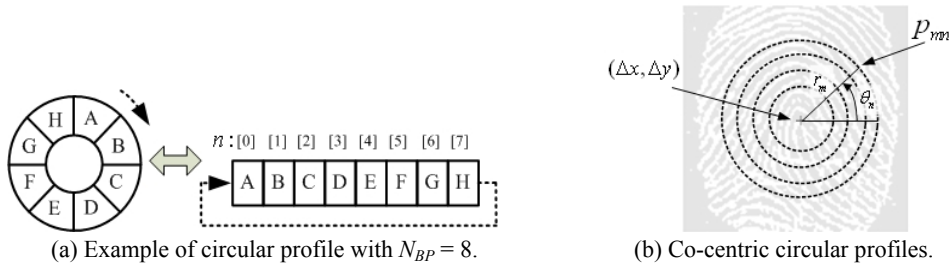


Fig. 6. Circular profile features.

Fig. 6 (a) shows an example of circular profile. All the sample pixels of the co-centric circular profile are the same. Assign a coordinate set  $(\Delta x, \Delta y)$  as the centre of co-centric circular profiles and the radius of circular profile  $r_m$  (see Fig. 6 (b)). The coordinate set  $O$  of circular profile is defined by:

$$O = \{(x_{mn}, y_{mn}) \mid r_m \in \mathfrak{R}; \theta_n \in \Theta; 0 \leq x_{mn}, y_{mn} < W\} \tag{9}$$

where,

$$\begin{cases} x_{mn} = r_m \cos \theta_n + \frac{W}{2} + \Delta x \\ y_{mn} = r_m \sin \theta_n + \frac{W}{2} + \Delta y \end{cases} \tag{10}$$

$$\mathfrak{R} = \{r_m \mid 1 \leq m \leq N_{BC}\} \tag{11}$$

$$\Theta = \{\theta_n \mid 1 \leq n \leq N_{BP}\}, \theta_n = n \cdot \frac{2\pi}{N_{BP}} \tag{12}$$

$(\Delta x, \Delta y)$  is the displacement between the centre of circular profile and the centre of fingerprint image. If there is no a priori knowledge or heuristic about the position of reference point of the fingerprint,  $(\Delta x, \Delta y)$  is assigned to  $(0, 0)$ .

$r_m$  is the radius of the  $m$ th circular profile.

$\theta_n$  is the sampling degree of the  $n$ th sample pixel of circular profile.

$N_{BC}$  is the number of circles.

$N_{BP}$  is the number of sample pixels on a circular profile.

The circular profile features (Fig. 6 (b)) of the fingerprint are defined by:

$$P = \{p_{mn} \mid p_{mn} = L(x_{mn}, y_{mn}), (x_{mn}, y_{mn}) \in O\}. \tag{13}$$

### 3.3 Feature Matching

#### 3.3.1 Displacement detection

The displacement can be detected by comparing two block direction images (see Fig. 7). We compute the mean error of the overlapped area of the enrolled block direction image  $D_E$  and the image to be verified  $D_C$ . The mean error is defined as:

$$MeanErr(\Delta u, \Delta v) = \frac{1}{(w - |\Delta u|)(w - |\Delta v|)} \cdot \sum_{u=f_1(\Delta u)}^{f_2(\Delta u)} \sum_{v=f_1(\Delta v)}^{f_2(\Delta v)} cmp(D_C(u, v), D_E(u - \Delta u, v - \Delta v)) \tag{14}$$

where,

$$f_1(z) = \begin{cases} z, & \text{if } (z > 0), \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

$$f_2(z) = \begin{cases} w - 1, & \text{if } (z > 0), \\ w + z - 1, & \text{otherwise.} \end{cases} \tag{16}$$

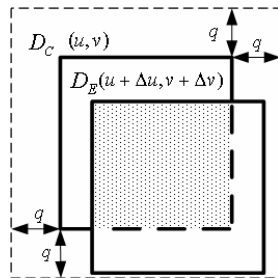


Fig. 7. Displacement detection.

$$\text{cmp}(D_C(u, v), D_E(u - \Delta u, v - \Delta v)) = \begin{cases} 0, & \text{if } (D_C(u, v) = D_E(u - \Delta u, v - \Delta v)), \\ 1, & \text{otherwise.} \end{cases} \quad (17)$$

The displacement  $(\Delta u_{\min}, \Delta v_{\min})$  between  $D_E$  and  $D_C$  is obtained by

$$(\Delta u_{\min}, \Delta v_{\min}) = \{(\Delta u, \Delta v) \mid \min(\text{MeanErr}(\Delta u, \Delta v)); -q \leq \Delta u, \Delta v \leq q\} \quad (18)$$

The  $(\Delta u_{\min}, \Delta v_{\min})$  is transformed to raw image scale by using:

$$\begin{cases} \Delta x_0 = b \cdot \Delta u_{\min} \\ \Delta y_0 = b \cdot \Delta v_{\min} \end{cases}. \quad (19)$$

We get the displacement  $(\Delta x_0, \Delta y_0)$  between the enrolled image  $L_E(x, y)$  and the image to be verified  $L_C(x, y)$ , which is the indispensable information for the circular matching procedure.

### 3.3.2 Circular matching

After the enrolled circular profile feature  $P_E$  and the displacement  $(\Delta x_0, \Delta y_0)$  are obtained separately, the following procedure will conduct to the final matching:

- Step 1: Initialize.** Let  $\Delta n$  ( $-\alpha \leq \Delta n \leq \alpha$ ) be the rotation parameter and  $\rho_x, \rho_y$  ( $-\beta \leq \rho_x, \rho_y \leq \beta$ ) be the shifted parameter,  $\alpha = 5, \beta = 5$  are adopted in our experiments.
- Step 2: Loop for  $\Delta n$  and  $(\rho_x, \rho_y)$ .** Change the rotation and the shifted parameters, then execute steps 3 and 4.
- Step 3: Compute  $P_C$ .** Extract the circular profile feature  $P_C$  from  $L_C$  using  $(\Delta x_0, \Delta y_0)$  and  $(\rho_x, \rho_y)$  by

$$P_C = \{p_{Cmn} \mid p_{Cmn} = L_C(x_{mn}, y_{mn}), (x_{mn}, y_{mn}) \in O, \Delta x = \Delta x_0 + \rho_x, \Delta y = \Delta y_0 + \rho_y\}. \quad (20)$$

- Step 4: Match  $L_C$  with enrolled  $P_E$ .**  $P_C$  and  $P_E$  are binary string, use Eq. (21) to calculate Hamming distance  $d_H(\Delta n, \rho_x, \rho_y)$ . The symbol  $\otimes$  denote the operation of Hamming distance calculation.

$$d_H(\Delta n, \rho_x, \rho_y) = \frac{\|P_E \otimes P_C\|}{\|P_C\|} \quad (21)$$

Keep the minimum Hamming distance  $d_H$  as  $d_{\min}$  for each cycle of loop. If all possible values of  $\Delta n$  and  $(\rho_x, \rho_y)$  are tested, execute step 5.

- Step 5: Compare  $d_{\min}$  with  $d_T$ .**  $d_T$  is a threshold. If  $d_{\min}$  is less than  $d_T$ , the  $I_C$  is accepted as  $I_E$  identity.

A rotated fingerprint image against the enrolled image can be matched by testing different  $\Delta n$ . Thanks to the rotation-invariant property of circular profile feature, the

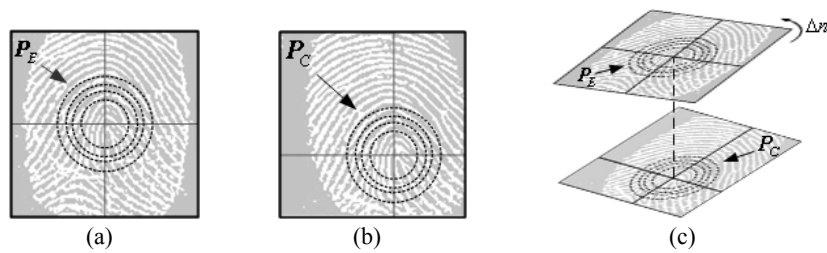


Fig. 8. Example of circular profile matching. (a) Enrolled circular profile feature; (b) Fingerprint to be verify and the extracted circular profile feature; (c) Feature matching with  $\Delta n = -1$ ,  $\rho_x = 3$ ,  $\rho_y = -4$ .

matching process has high flexibility to adjust between the matching time and rotation tolerance. The larger the range of  $\Delta n$  is, the longer the matching time will take, which, however, enables a larger rotation tolerance. Fig. 8 shows an example. When  $N_{BC} = 4$ ,  $N_{BP} = 90$  and the displacement detected  $(\Delta x_0, \Delta y_0) = (12, 14)$ , two fingerprints are matched by circular profile feature as shown in Fig. 8.

## 4. IMPLEMENTATION AND PERFORMANCE EVALUATION

### 4.1 Algorithmic Performance Evaluation

In view of validation of functionality and comparison, we implement firstly the conventional minutiae-based algorithm described in section 2 and the proposed algorithm in Pentium 4/2.5GHz desktop PC.

The performance of fingerprint verification is evaluated by means of FAR (False Acceptance Rate) curve, FRR (False Rejection Rate) curve and EER (Equal Error Rate) [17]. FAR is defined as the probability of an impostor being accepted as a genuine individual. FRR is defined as the probability of a genuine individual being rejected as an impostor. Both FAR and FRR are regarded as the functions of the threshold used to compare the matching score make the decision. The matching score is from the expected probability that a sample declared to match a template. Therefore, a small FRR usually results in a larger FAR, when a smaller FAR usually implies a larger FRR. In general, the term of FAR is very important. Because the value of FAR is zero, it implies that no impostor is accepted as a genuine individual. A specific point is obtained when FAR and FRR coincide, the so-called EER (equal error rate). The performance of a biometric verification system is estimated with the EER. A low equal error rate value reveals a high verification accuracy of the biometric system.

In our experiment, we use a semiconductor-type fingerprint sensor AFS2 from AuthenTec Co. [18] to capture 594 fingerprint images [20] from 54 fingers, 11 images for each finger. The acquisition of fingerprint image is under the normal working environment and normal user mode. As stated by Maio *et al.* [17], in this situation, the maximum rotation of image is approximately in the range  $[-15^\circ, 15^\circ]$ . Some sample images are shown in Fig. 9. For each finger, a fingerprint image is randomly selected as the registered image and the rest as the testing images. The verification is performed respectively

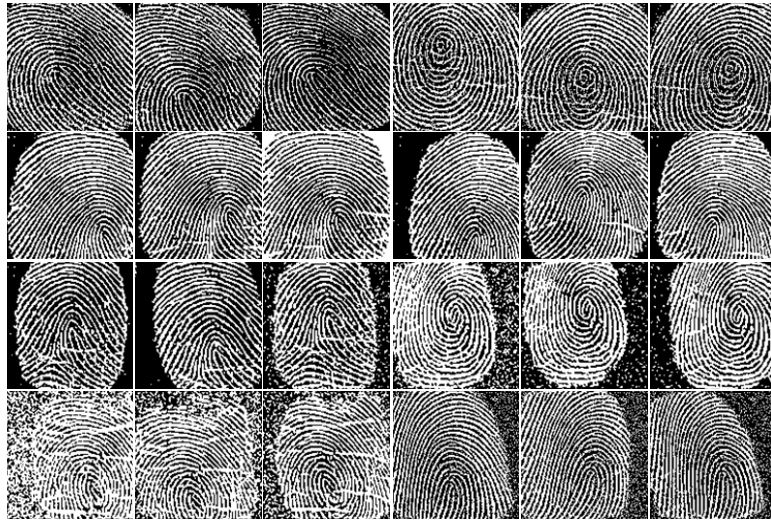


Fig. 9. Sample images extracted from experimental fingerprint image database (The first three columns are the right hand fingerprints. The first row is thumb. The second row and the third row are pointing finger. The last row is middle finger).

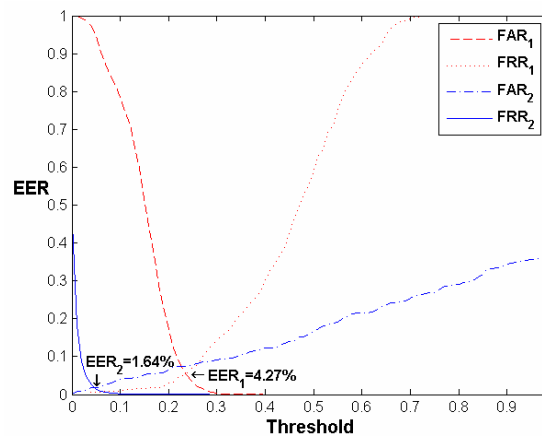


Fig. 10. Performance comparison between minutiae-based algorithm and the proposed algorithm. The subscript 1 represents the proposed algorithm; subscript 2 represents the minutiae-based algorithm.

**Table 1. Efficiency comparison between minutiae-base algorithm and the proposed algorithm.**

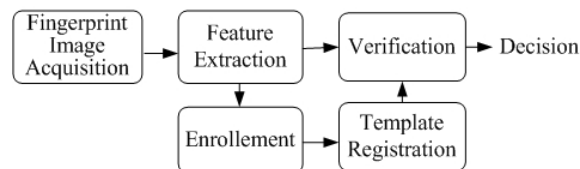
Execution time (in msec)	Preprocessing	Feature Extraction	Matching	Total
Minutiae-based	7.3	3.5	2.5	13.3
Proposed Algorithm	0.325	0.67	0.245	1.24

by the conventional minutiae-based algorithm and the proposed algorithm. The verification performance is shown in Fig. 10. Table 1 presents the comparison of their execution time. With a little sacrifice of verification performance (against minutiae-based), the proposed algorithm is far more efficient than minutiae-based fingerprint verification algorithm.

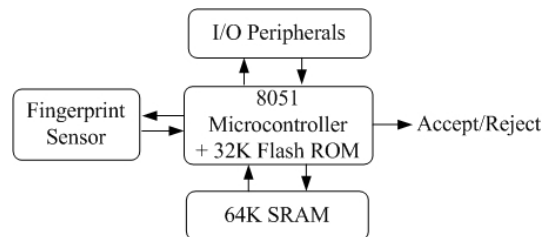
Some false accepted results (Fig. 11) are due to the displacement detection module assumed two fingerprint images from the same finger (but it is not real in fact), therefore the displacement value is wrong and two circular profile features are similar.



Fig. 11. The false accepted samples in the fingerprint database.



(a) Functional architecture.



(b) Hardware architecture.

Fig. 12. The fingerprint verification system.

#### 4.2 Implementation of Low Cost Fingerprint Verification System

We implement a low cost fingerprint verification embedded system with the proposed low complexity algorithm. Fig. 12 represents the system architecture. Fig. 12 (a) shows the functional modules and data flow of the system; Fig. 12 (b) presents its hardware architecture.

The adopted sensor in Fig. 12 (b) is AFS2 from AuthenTec Co. [18] whose resolution is 250 DPI with image size  $W \times W = 128 \times 128$  pixels. A Silicon Lab's 8051 microcontroller C8051F020 [19] is selected for implementing the algorithm. The microcontroller runs at 25 MIPS (at 25MHz) equipped with 64K Flash ROM and 8K SRAM. The

embedded Flash ROM of microcontroller is used to store the program code as well as the template features of fingerprint. Beside, we add a 64K external SRAM into the embedded system as an image buffer. The output of verification module provides two alternatives: accept or reject.

The algorithm is transplanted to 8051 platform then is tuned for optimizing the code size and memory management. After a series of the fine tuning, the code size is reduced to 12 Kbytes and the data memory size occupies only 35 Kbytes. The photograph of the realized complete system is shown in Fig. 13.

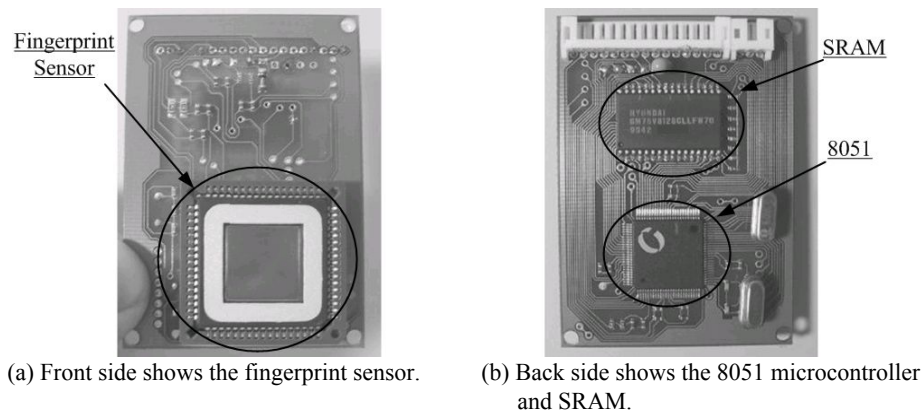


Fig. 13. Realization of low cost embedded fingerprint verification system.

**Table 2. Requirement of computational resources.**

Module	Addition	Comparator	Shifter	division
Low-Pass Filter	129541	96901	0	0
Block Direction	210011	228725	0	0
Displacement Detection	25828	10073	2	169
Circular Matching	23491	61393	21600	0
Total	568871	397092	21602	169

**Table 3. Execution time at 25 MIPS 8051.**

Module	Execution Time(sec)
Low-Pass Filter	0.01
Block Direction	0.13
Displacement Detection	0.15
Circular Matching	0.6
Total Match Time	0.89

The computational performance is evaluated by means of the number of required addition, comparison, shift and division operation, as well as the execution time run at 25 MIPS 8051. The result is shown in Tables 2 and 3.

Due to the utilization of integer data type and multiplierless operation, the execution speed is substantially accelerated and the code size and the memory size are simultaneously reduced. This compact implementation encourages the applications of the proposed fingerprint verification algorithm to hardware resources-constraint portable and mobile security. Furthermore, the realized embedded system provides a prototype for implementation of fingerprint verification SoC.

## 5. CONCLUSION

The proposed low complexity fingerprint verification scheme targets on low cost, hardware restricted embedded applications. Differing from conventional minutiae-based fingerprint verification, our approach uses block direction and circular profile as fingerprint feature representation. The purpose of the proposed method is to find out a coarse displacement from direction feature. Thus, the circular matching module will achieve more accurate matching by means of the direction feature. It reduces significantly image buffer requirement and accelerates the memory access speed. The matching procedure consists of displacement detection and circular matching that provide adjustable flexibility of rotation-invariant/matching speed. The proposed algorithm is verified on a desktop PC and compared with the minutiae-based fingerprint verification algorithm. With a little sacrifice of verification performance (EER = 4.27% against minutiae-based EER = 1.64%), the proposed algorithm achieves considerable reduction of computational complexity. The proposed algorithm is afterward implemented in an 8-bits 8051 microcontroller run at 25MHz. As a result of optimization in data type, arithmetic operation and memory management, the program code consumes only 12 Kbytes and the data memory size is 35 Kbytes. The evaluation experiment shows that the proposed algorithm has the advantages of high efficiency as well as very low computational complexity at satisfactory verification performance. The realized embedded fingerprint verification system provides a prototype for implementation of low cost fingerprint verification SoC for potential mobile security applications.

## REFERENCES

1. A. Jain, R. Bole, and S. Panakanti, *Biometrics: Personal Identification in Networked Society*, Kluwer Academic Publishers, Massachusetts, 1999.
2. P. Komarinski, *Automated Fingerprint Identification System*, Elsevier Academic Press, Burlington, MA, 2005.
3. A. K. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, 1997, pp. 302-314.
4. D. Maio and D. Maltoni, "Direct gray-scale minutiae detection in fingerprints," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, 1997, pp. 27-40.
5. N. Yager and A. Amin, "Fingerprint verification based on minutiae features: a review," *Pattern Analysis and Application*, Vol. 7, 2004, pp. 94-113.
6. Y. Gil, D. Moon, S. Pan, and Y. Chung, "Fingerprint verification system involving smart card, information security and cryptology," in *Proceedings of International*

- Calendar of Information Science Conferences*, 2002, pp. 510-524.
7. S. Yang and I. Verbauwhede, "A real time, memory efficient fingerprint verification system," in *Proceedings of IEEE Acoustics, Speech, and Signal Processing*, 2004, pp. V-189-192.
  8. S. B. Pan, D. Moon, Y. Gil, D. Ahn, and Y. Chung, "An ultra-low memory fingerprint matching algorithm and its implementation on a 32-bit smart card," *IEEE Transactions on Consumer Electronics*, Vol. 49, 2003, pp. 453-459.
  9. T. Y. Tang, Y. S. Moon, and K. C. Chan, "Efficient implementation of fingerprint verification for mobile embedded systems using fixed-point arithmetic," in *Proceedings of ACM Symposium on Applied Computing*, 2004, pp. 821-825.
  10. A. S. Rikin, W. Yiwen, T. Nakada, L. Dongju, T. Isshiki, and H. Kunieda, "Realization of fingerprint identification module on DSP board," in *Proceedings of Asia-Pacific Conference on Circuits and Systems*, Vol. 1, 2002, pp. 509-512.
  11. L. Hong, Y. Wan, and A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, 1998, pp. 777-789.
  12. K. C. Chan, Y. S. Moon, and P. S. Cheng, "Fast fingerprint verification using subregions of fingerprint images," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 14, 2004, pp. 95-101.
  13. R. S. Germain, A. Califano, and S. Colville, "Fingerprint matching using transformation parameter clustering," *IEEE Computational Science and Engineering*, Vol. 4, 1997, pp. 42-49.
  14. V. Espinosa-Duro, "Minutiae detection algorithm for fingerprint recognition," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 17, 2002, pp. 7-10.
  15. E. Zhu, J. Yin, and G. Zhang, "Fingerprint matching based on global alignment of multiple reference minutiae," *Pattern Recognition*, Vol. 38, 2005, pp. 1685-1694.
  16. N. Ratha, K. Karu, S. Chen, and A. Jain, "A real time matching system for large fingerprint databases," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 18, 1996, pp. 799-813.
  17. D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "FVC2000: fingerprint verification competition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 24, 2002, pp. 402-412.
  18. AuthenTec Co., <http://www.authentec.com>.
  19. Silicon Laboratories, Inc., <http://www.silabs.com>.
  20. [http://140.115.11.235/~chen/biometrics/database/AFS2\\_fingerprint\\_images.html](http://140.115.11.235/~chen/biometrics/database/AFS2_fingerprint_images.html).



**Jia-Hong Dai (戴嘉宏)** received his M.S. degree in Electrical Engineering from I-Shou University, Taiwan, R.O.C. in 2000. He currently is pursuing the Ph.D. degree in the department of Electrical Engineering at I-Shou University. His research interests include embedded system, computer vision, neural networks, and fuzzy theory.



**Ching-Han Chen (陳慶瀚)** was born in Kinmen, Fujian, R.O.C., on October 14, 1963. He received the B.S. degree and Master degree in Geophysics from National Central University, Taiwan, R.O.C. in 1986 and 1988 respectively, D.E.A (Diplôme d'Etudes Approfondies) in Informatique, Automatique et Productique in 1992 and Ph.D. degree in 1995 from the Franche-Comte University, France. From August 1995 to July 2006, he was an associate professor of the Department of Electrical Engineering, I-Shou University, Taiwan. He has been an associate professor in the Department of Computer Science and Information Engineering, National Central University, Taoyuan, Taiwan, since August 2006. His research interests include embedded system design, biometrics, and multimedia signal processing.