Embedded Fingerprint Verification System

Gwo-Cheng Chao*, Shung-Shing Lee**, Hung-Chuan Lai***, Shi-Jinn Horng***
*Graduate Institute of Networking and Multimedia National Taiwan Univ., Taipei, Taiwan
**Department of Electrical Engineering Ching Yau University, Jung-Li, Taiwan
***Department of Computer Science and Information Engineering National Taiwan Univ. of Science & Technology, Taipei, Taiwan

ABSTRACT

Fingerprint verification is one of the most reliable personal identification methods in biometrics. In this paper, an effective fingerprint verification system is presented. We describe an enhanced fingerprint verification system consisting of image pre-processing, feature extraction and matching processes. Improved image pre-processing and broken ridge reconnection methods are proposed here. In this paper, we also describe the design and implementation of a fingerprint verification system on SoC.

1. Introduction

Fingerprint recognition has been researched for many years. Most automatic systems for fingerprint comparison are based on minutiae matching. Recent detection methods of minutiae still use gray-scale fingerprint image. The gray level image can be transformed to a binary image through a binarization process. Most minutiae detection methods that have been proposed in the literature are based on image binarization [1] [2]. Figure 1 shows the processing flow of our proposed methods. In section 2, we will illustrate how to calculate the orientation field of a fingerprint. Section 3 describes the pre-processing steps of a fingerprint image. In section 4, a singular point detection method and a minutiae matching algorithm are introduced. Section 5 addresses the embedded system design. Section 6 some results of our approaches are presented. Finally, a conclusion is given.

2. Calculation of the orientation field

Several methods for estimating image directional information have been proposed in the literature [3] [4]. The orientation field, θ, is defined as a $M \times N$ image, where $\theta(i, j)$ represents the local ridge orientation at pixel (i, j). Local ridge orientation is usually specified for a block rather than at every pixel. In this paper, we calculate the orientation of each image pixel. In this way, the direction locality of an image can be preserved and the orientation field of an image can be more smoothly recalculated. Finally, with pixels neighborhood, Gaussian smoothing operator is used to smooth the orientation fields of an image [5].

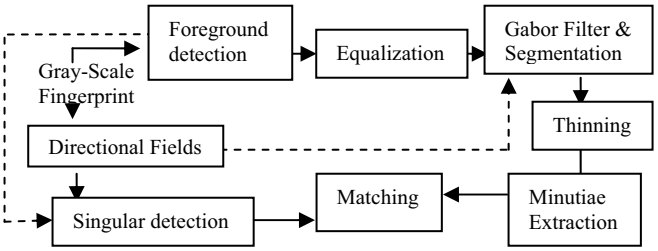


Figure 1. The processing flow of fingerprint verification system.

3. Fingerprint pre-processing

There are several pre-processing methods used in a gray-scale image and a binarization image [6] [1]. The pre-processing procedures will affect the result of extracting fingerprint minutiae. In this section, we will describe foreground (object) detection, image equalization, and Gabor Filter.

3.1. Fingerprint Foreground detection

After investigating a fingerprint image, we know that the ratio of ridges to valleys is nearly equal in a fingerprint. We use this characteristic of a fingerprint to deal with this problem.

A foreground detection procedure is illustrated as follows:

1. First, we set a rough threshold (T) value for a fingerprint image using image histogram.
2. The image is divided into four equal regions, $A_i, i = 1,2,3,4$, by dividing the image vertical and horizontal equally (Figure 2).
3. Using a window block B ($m \times m$), a gray-scale fingerprint image ($n \times n$) is divided into n/m blocks. In each block, the

Proceedings of the 2005 11th International Conference on Parallel and Distributed Systems (ICPADS'05)
0-7695-2281-5/05 $20.00 © 2005 IEEE

gray value is bigger than threshold (T) indicating the foreground (value 1), otherwise, background (value 0). A block of an image will be considered as foreground region if the ratio of count-1 to count-0 in this block is less than a threshold value.

4. Apply the dilation operation to the eliminate these holes and make the boundary area smooth. Then we get the final foreground image region, foreground mask. Figure 3 shows some examples.

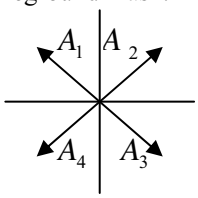


Figure 2. Four regions of an image.

(a) (b) (c) (d)

Figure 3. Examples of foreground extraction. (a, c) Original thinned binary fingerprint (b, d) the tracing result.

After retrieving the foreground of an image, we apply an image equalization procedure in the foreground region. Doing so helps to avoid the influence of background noise during the equalization procedure.

3.2. Fingerprint Image Enhancement

After foreground extraction and image equalization, we use the Gabor filters to enhance the fingerprint image. An even symmetric Gabor filter has the following general form in the spatial domain:

$$G(x, y, f, \theta) = \exp\{-\frac{1}{2}\left\{\frac{x'^2}{\delta_{x'}^2} + \frac{y'^2}{\delta_{y'}^2}\right\}\}\cos(2\pi f x') \quad (1)$$

$$x' = x\sin\theta + y\cos\theta \quad (2)$$

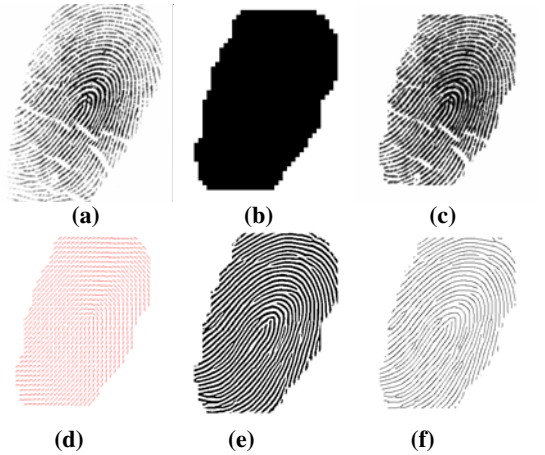
$$y' = x\cos\theta - y\sin\theta \quad (3)$$

where f is the frequency of the sinusoidal plane wave along the direction θ from the x-axis, the filter frequency can be set to the average ridge frequency ($1/K$), where K is the average inter-ridge distance. If f is too large, spurious ridges are created in the filtered image whereas if f is too small, nearby ridges are merged into one. The values $\delta_{x'}$ and $\delta_{y'}$ are the space constants of the Gaussian envelope along x' and y' axes, respectively. If $\delta_{x'}$ and $\delta_{y'}$ (standard deviations of the Gaussian envelope) values are too large, the filter is more robust to noise, but is more likely to smooth the image to the extent that the ridge and valley details in the fingerprint are lost. If $\delta_{x'}$ and $\delta_{y'}$ values are too small, the filter is not effective in removing the noise. The values for each were empirically determined and set to 4.0 (about half the average inter-ridge distance) [6] [7]. We perform the filtering on an image in the spatial domain with a mask size of $18*18$. However, to speed up the filtering process, we designed a circuit, named Gabor redundancy circuit (GRC) which will be discuss in section 6.

region A_i, $i = 1,2,3,4$, then scan the image from the center toward the boundary and put the foreground blocks together. So, the entire foreground region can be obtained.

5. The foreground region obtaining from step 4 may have some small holes inside the region; we need to

According to each pixel's orientation, the corresponding Gobal filter is used to enhance this pixel. By setting a proper threshold value to the enhanced image, a binary image can be obtained. An example of fingerprint image processing is shown in Figure 4.



(a) (b) (c) (d) (e) (f)

Figure 4. (a) Original fingerprint (b) the foreground of the original fingerprint (c) the gray-scale image of the original fingerprint after equalization (d) the direction field of the original fingerprint (e) the binary image of the original fingerprint after Gabor filter and segmentation (f) the thinning image of the original fingerprint.

4.Singular detection and Matching algorithm

4.1 Singular detection

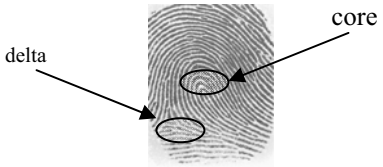


Figure 5. A core and a delta in a fingerprint.

There are two kinds of singular points, core and delta. Figure 5 shows the shapes of core and delta. A point in the directional image is classified as an ordinary point, core or delta by computing the Poincaré index [4] along a small closed curve around the point. The Poincaré index is computed by summing up the changes in the direction angle around the curve. When making a full counter-clockwise turn around the curve in a directional image, we see that the direction angle turns 0, ± 180 and ± 360 degree during this trip. A point is termed ordinary if the angle has turned 0 degree, core if it has turned 180 degree and delta if it has turned -180 degree.

In a block image (each block contains 9x9 pixels), we compute the Poincaré index at every block (i, j) in a 2 x 2 rectangle, where the upper left corner is placed at the pixel of interest. The rectangle is traversed in the counterclockwise direction:
$(i, j) \rightarrow (i+1, j) \rightarrow (i+1, j+1) \rightarrow (i, j+1) \rightarrow (i, j)$
when computing the difference between two angles (which is determined up to $\pm$ 180), we take the difference that is smallest in the absolute value.

4.2 Matching algorithm

In this work an alignment-based matching algorithm is implemented. This alignment-based matching algorithm is decomposed into two stages:

1) ***Alignment stage***:

We use the minutiae of a template image as base and align them to the minutiae of the input image. A transformation function $Y_i = F(X_i)$ between input image Y_i and template image X_i can be written as:

$Y_i = s \cdot R \cdot X_i + T$,where s is the scale factor, set to value 1, in our fingerprint verification system because the same size of fingerprints are used.

The value θ is the rotation angle between two fingerprints. We use the core point as center and the distance ***r*** from core point as the radius, and then we construct a local structure of each fingerprint. Then we find the most corresponding point pair (P_1, P_2) through structure matching. We extract the orientations (θ_1, θ_2) of the point pair (P_1, P_2) , and then compute the difference parameter $\theta = \theta_2 - \theta_1$ as the rotation angle. The rotation matrix $R = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, is used to align both images minutiae in the same orientation. The translation vector $T = [t_x, t_y]^T$ is used to adjust the distance ***r*** of template image minutiae, we calculate the difference of the distance δ between the core points of the two fingerprints, and then move all minutiae of template image by adding the parameter δ.

2) ***Matching stage***:

After alignment each pair of corresponding minutiae points is completely coincident if two identical munitiae are exactly aligned with each other.

Let $I = \{(x_1^I, y_1^I, \theta_1^I)^T, \ldots, (x_M^I, y_M^I, \theta_M^I)^T\}$ denote the set of M minutiae in the input image and $P = \{(x_1^P, y_1^P, \theta_1^P)^T, \ldots, (x_N^P, y_N^P, \theta_N^P)^T\}$ denote the set of N minutiae in the template image. We denote the reference point (core point) as $(x_I^r, y_I^r, \theta_I^r)$ and $(x_P^r, y_P^r, \theta_P^r)$ for each input image and template image. Then we construct match tables M_I and M_P for matched minutiae pair I_M , P_N , and we compute the following parameters and store them in each match table M_I and M_P .

1. The distance of x and y coordinate from reference point to others in an image.
2. The angle difference of the input image and the template image.
3. The type of the minutiae in the input point patterns and the template point patterns.
4. The quadrant using the core point as a center point.

After creating match tables, we compare these two tables and find the set of points that are best matched between input and template image.

5. Embedded System Design

In this section, we illustrate our SoC architecture, design flow and verification flow. System partition is the first step in Hardware and Software Co-design. We can't make decisions without a particular system analysis. Every embedded processor possesses individual operations capability. Therefore we must proceed to a case by case system analysis. We implement our fingerprint algorithm in software and execute it using a Nios processor. Thereby we can obtain accurate information and determine the appropriate system partition method. There are four points for attending to the system partition.

The part of hardware:

- We must take care of hardware complexity, which affects the system cost.

The part of system software:

- System execution time must fit in specification.
- Instruction ROM size is fixed by specification.
- Memory size is fixed, so system software in execution time cannot waste memory space.

5.1. SoC Architecture

Figure 6 shows our SoC architecture which consists of Nios CPU, memory, sensor controller, gradient fields, Gabor redundancy circuit (GRC), thinning hardware, Avalon bus and Infineon FingerTIP sensor. We use SOPC Builder to generate

all hardware except flash memory, a SRAM module, a sensor controller and the Infineon FingerTIP sensor. The Infineon FingerTIP sensor is connected to SoC by parallel port. The sensor provides an 8-bit representation of the gray levels for each pixel. A fingerprint image consists of 224 columns and 288 lines, yielding 64,512 bytes plus the end of line marks. The sensor controller is implemented in Verilog HDL code. Using a custom instruction and memory mapping method, all system hardware can be connected together. Using the thinning hardware reduces complexity as it implements the algorithm by custom instruction. The circuits of gradient fields and GRC are implemented by memory mapping method. The thinning, gradient fields and GRC hardware architectures are shown in Figure 7 and Figure 8. Because the gradient fields and GRC circuit use the same circuit to perform their functions, we use one circuit to implement both functions (Figure 9). In this way, the area of the hardware can be greatly reduced.

5.2. Design Flow

The design flow begins with pre-design activity, which includes an analysis of the system requirements. In hardware design flow, we use the SOPC Builder and the Quartus II software to create and process our own Nios system module design that interfaces with components provided on the Nios development board. In software design flow, first we can begin coding device-independent C/C++ software, such as arithmetic algorithms or control programs. After we define the custom Nios processor hardware system using SOPC Builder, SOPC Builder generates a custom software development kit (SDK) that forms the foundation for the software development flow. With the SDK in hand, we can begin coding software that interacts at a low level with hardware components. The SDK defines the software view of the custom hardware, including the memory map and the data structures for accessing hardware components in the system. The SDK provides software routines for accessing standard peripherals such as UARTs, PIOs, and DMA controllers. We use the GNUPro Toolkit to compile and link software together with the SDK routines, header files, and other software libraries. Compilation results in an executable software image. After we prototype the basic Nios processor hardware working on the development board, we can download the executable software to the development board using an Altera ByteBlasterMV.

6. Experimental Result

In our fingerprint verification experiments, 1000 fingerprint images are captured from 200 different fingers, 5 images from the same finger as a template. Totally, 200 templates are contained in the database. Each fingerprint image in the database is verified with its own other four images of the same finger, and with the other 199 templates. There are totally 999000 (1000*999) times of matching in a verification process, and the *verification rate* is 97.32%, the *rejection rate* is 9.2%. The ***FAR*** is 0.0009255 and the ***FRR*** is 0.0275.Table 1 reports the average computational times (B,C,D,E)[8] in automatic minutiae extraction on a PC-80486-DX 50 Mhz. (A) is the average computational times of our SoC which contains a 33 Mhz Nios CPU.

Table 1. Average computational time.

Average computational time (sec.)	Directional	Segmentation	Smoothing	Thinning	Minutiae extraction	Matching	Total time
A (this paper) (Intel P4 1.8G, SW exe. only)	1.5	Gabor		1.58	0.0002	0.00067	8.34887
		5.138					
A (NIOS 33MHz SW exe. only)	10.34	40.52		4.965	0.05	0.148	57.23
A (NIOS &HW support)	1.2	2.82		0.003	0.05	0.148	5.431
B [8]	-	2.25	3.90	3.11	0.51	-	9.77
C [8]	-	3.08	3.90	3.11	0.67	-	10.76
D[8]	-	2.64	-	3.15	0.66	-	6.45
E [8]	0.51	15.73	-	2.45	0.33	-	19.03

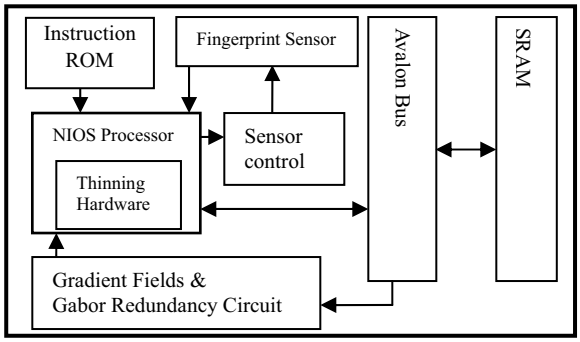


Figure 6. Our SoC architecture.

7. Conclusion

This paper describes the techniques used in a fingerprint

verification system as well as an implementation of such a system by means of a system-on-chip design. Several processing phases are applied to the fingerprint image like the fingerprint foreground detection, image equalization, enhancement by a Gabor filter and thinning to obtain an improved binary image of the given fingerprint. The matching algorithm is based on the detection of cores in the fingerprint, and a given fingerprint is rotated and translated to match another fingerprint.

The presented image processing algorithms and the matching algorithms are implemented with SoC technology using the SOPC Builder. A special hardware circuit is developed for the Gabor filter while the thinning process is implemented on chip as a special instruction of the microprocessor. The reason for this choice is that the hardware complexity of the thinning process is low in contrast to the Gabor filter. Moreover, the Gabor filter and the computation of the gradient field can use the same hardware circuit so that the area is reduced greatly. Finally, experimental results show that the developed system works well and the hardware support makes the system 10 times faster.

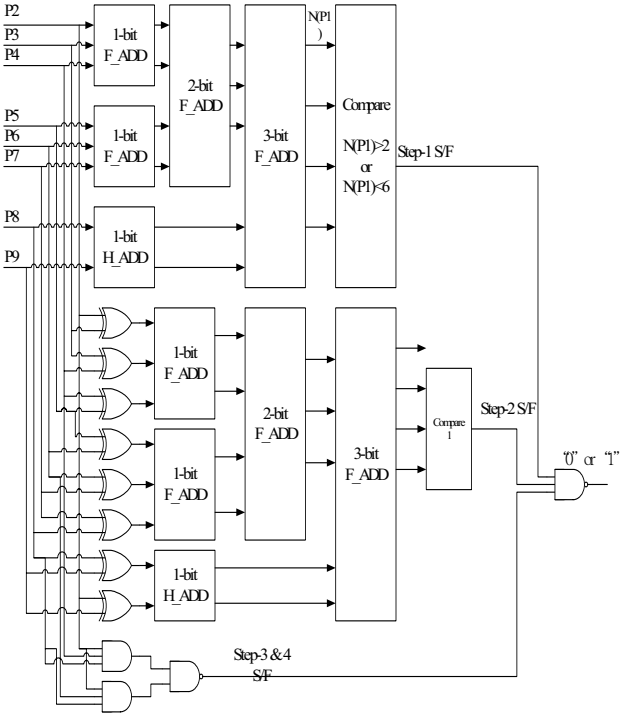


Figure 7. The thinning hardware architectures.

References

[1] S. Greenberg, M. Aladjem, D. Kogan and I.Dimitrov, **"**Fingerprint image enhancement using filtering techniques**"** Proceedings of the 2000 IEEE Conference on Pattern Recognition , 2000,pp. 322 - 325

[2] M. Tico and P. Kuosmanen, "An algorithm for fingerprint image post-processing," Proceedings of the Thirty -Fourth Asilomar Conference on Signals, Systems and Computers, 2000, pp. 1735 – 1739.

[3] T. Chang, "Texture analysis of digitized fingerprints for singularity detection,"in Proc. 5th Int. Conf. Pattern Recognition, 1980, pp. 478–480.

[4] M. Kawagoe and A. Tojo, "Fingerprint pattern classification," Pattern Recognit., 1984, vol. 17, no. 3, pp. 295–303.

[5] Sen Wang,Student Member, IEEE, and Yangsheng Wang, "Fingerprint Enhancement in the Singular Point Area" IEEE Signal Processing Letters, Jan. 2004, vol. 11, no. 1, pp.16 - 19

[6] Anil K. Jain, Fellow, IEEE, Salil Prabhakar, Lin Hong, and Sharath Pankanti" Filterbank-Based Fingerprint Matching" IEEE Trans. on Image Processing, May 2000, vol. 9, no. 5, pp.846 - 859

[7] A. K. Jain, S. Prabhakar, and L. Hong, "A multichannel approach to fingerprint classification," IEEE Trans. Pattern Anal. Machine Intell., 1999, vol. 21, no. 4, pp. 348–359.

[8] *D. Maio and D. Malton,* "Direct gray-scale minutiae detection in fingerprints," IEEE Trans. on Pattern Analysis and Machine Intelligence, Jan. 1997, vol. 19, pp.27 – 40.

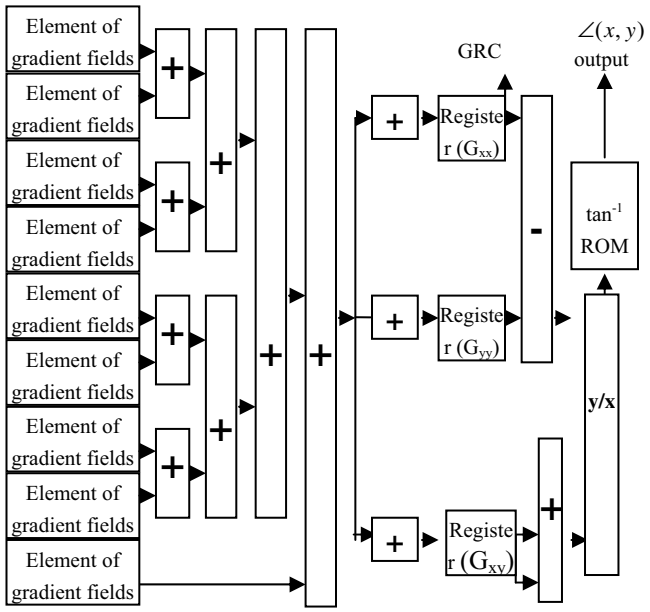


Figure 8. The gradient fields and GRC hardware architectures.

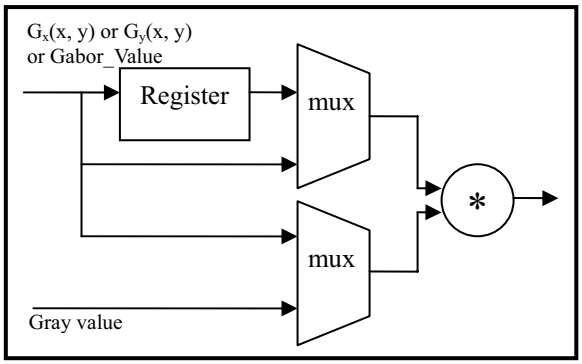


Figure 9. Element of gradient fields & GRC.