Fingerprint-Based Verification System
A Research Prototype

Uroš Klopčič, Peter Peer
Computer Vision Laboratory
Faculty of Computer and Information Science
University of Ljubljana
Ljubljana, Slovenia
uros.klopcic@gmail.com, peter.peer@fri.uni-lj.si

Abstract— The design and implementation of a fingerprint verification system is presented. The system is capable of segmentation, fingerprint enhancement using Gabor filters, thinning, minutiae extraction, classification and matching. The system has been tested on four sets of fingerprint images used at Fingerprint Verification Competition 2002 (FVC2002). All steps except enhancement were found to be effective. Especially the sensitivity to fingerprint quality should be taken care of in the future. Processing time of 3 seconds on average meets the response time requirements of a simple fingerprint verification system.

Keywords-Fingerprints, fingerprint verification, FVC2002, fingerprint reader.

I. INTRODUCTION

Fingerprints are found to be one of the most reliable and accurate biometric characteristics. Used for access control and criminal identification, they are the most widely used biometric features. With the spread of computer technology the time-consuming manual fingerprint verification has been replaced with fast and reliable automated fingerprint identification systems (AFIS).

In this paper we will present a fingerprint verification system capable of identifying people in real-time. The main intention was to build a working prototype not necessarily 100% reliable, but providing a good, modularly designed framework on which further improvements and optimizations could be simply applied.

At first digital fingerprint image acquisition is performed. For this purpose two fingerprints reader were used. However in this paper fingerprint acquisition will not be discussed. More emphasis will be on next step, which is the minutiae extraction where minutiae feature vector is obtained. Finally minutiae matching algorithm is executed. Minutiae from the input finger are matched against the template minutiae.

The paper is organized as follows: Section 2 discusses segmentation algorithm, followed by fingerprint image enhancement algorithm in Section 3. In Section 4 binarization, thinning and minutiae extraction algorithms are presented. Section 5 describes classification of the fingerprint using singular points, followed by matching algorithm in Section 6. Experimental results on FVC2002 [3] databases are presented in Section 7. Section 8 contains summary and discussion.

II. SEGMENTATION

We have implemented a segmentation algorithm which is a slightly modified version of the method presented in [4]. Firstly a value representing the image background is detected. On a negative of a fingerprint image local histogram is calculated. Observing the sums for each color values in histogram from 0 to 255 the very first value higher than threshold *ThrF* is selected as the background color. This value is subtracted from every pixel *(i, j)* in the image. Image is then stretched as in:

$$neg\_streched(i,j) = \frac{neg\_subtracted(i,j)}{\max_{i,j}(neg\_subtracted(i,j))} \cdot 255 \quad (1)$$

With (1) the dynamic range between ridges and background is increased. Next, image is divided into blocks of equal height and width *W* (9 in our case). For each block centered at pixel *(i, j)* four parameters are calculated: mean, gradient, variance and ROI (2).

$$ROI(i,j) = \sqrt{\frac{1}{W^2} \cdot \frac{(V_x(i,j)^2 + V_y(i,j)^2)}{V_e(i,j)}}, \quad (2)$$

where

$$V_x(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} 2 \cdot \partial_x(u,v) \cdot \partial_y(u,v), \quad (3)$$

$$V_y(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} (\partial_x^2(u,v) - \partial_y^2(u,v)), \quad (4)$$

$$V_e(i,j) = \sum_{u=i-\frac{W}{2}}^{i+\frac{W}{2}} \sum_{v=j-\frac{W}{2}}^{j+\frac{W}{2}} (\partial_x^2(u,v) + \partial_y^2(u,v)) \quad (5)$$

∂_x and ∂_y are gradients in *x* and *y* direction obtained using Sobel operator [10]. If mean is higher and all other parameters are lower than defined thresholds then the block is marked as a background.

However some blocks can be misclassified for foreground, others for background. Additional processing is needed using heuristic rules seen in Fig. 1.

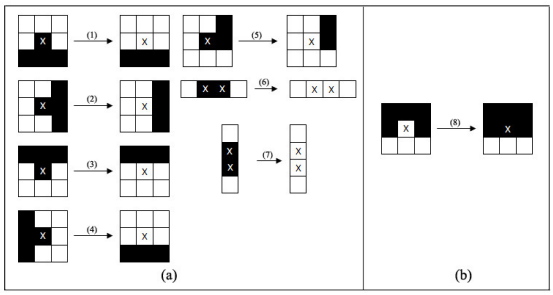

Figure 1. Heuristic rules for removing (a) foreground and (b) background blocks

III. FINGERPRINT IMAGE ENHANCEMENT

The purpose of this step is to eliminate any presence of noise to improve the matching result. Image is enhanced using Gabor filters as used in [5]. At first ridge orientation is calculated using gradient vectors. Due to noise and any other deformations on a fingerprint orientations must be smoothed. Algorithm presented in [6] is used. This method wisely adapts size s of the smoothing neighborhood $\Omega(s)$ according to consistency of orientation:

$$Cons(s) = \frac{\sqrt{(\sum_{(i,j)\in\Omega(s)} \cos(2\theta(i,j)))^2}}{O(s)} \qquad (6)$$

where $\theta(i, j)$ is the orientation of the given block and $O(s)$ represents number of orientations $\theta(i, j)$ in $\Omega(s)$.

When consistency is low, larger smoothing neighborhood is used to suppress the influence of noise, however when consistency is high, small neighborhood is used. Consistency is calculated for every block according to outer $8 \cdot s$ blocks from it's $(2 \cdot s + 1) \times (2 \cdot s + 1)$ neighborhood. The steps of orientation smoothing algorithm are:

1. Calculate unit vector with doubled orientation $[cos(2\theta(i, j), sin(2\theta(i, j)]$. Set $s = 1$ and use unit vector values when computing $Cons(s)$.
2. Increase s by 1. Compute $Cons(s)$ for $\Omega(s)$, where $\Omega(s)$ represents 8 outer neighboring blocks.
3. $Cons(s)$ is defined on an interval [0, 1]. If $Cons(s)$ is less than threshold (0.5 in our case) or is less then $Cons(s-1)$, then go to step 2. Keep returning to step 2 until you reach the maximum of neighborhood size s (5 in our case).
4. If s is equal to maximum, set $\Omega(s)$ to neighborhood size 3×3, otherwise go to step 5.
5. Compute smoothed orientation according to:

$$\overline{\theta}(i_c, j_c) = \frac{1}{2}\arctan\left(\frac{\sum_{(i,j)\in\Omega(s)} \sin(2\theta(i,j))}{\sum_{(i,j)\in\Omega(s)} \cos(2\theta(i,j))}\right) \qquad (7)$$

where i_c and j_c are the coordinates of the center point of the currently processed block.

Ridge frequency is calculated as defined in [5]. Finally filtering is performed using Gabor filters.

IV. BINARIZATION, RIDGE THINNING AND MINUTIAE EXTRACTION

Binarization is performed using AForge .NET framework [2]. The result is a binary image where ridges are presented with bit 1 and background with 0.

The purpose of thinning is to obtain ridge skeleton with 1 pixel wide ridges. An algorithm must not produce spurious ridges and must preserve the shape and location of the ridges. A thinning method [8] was used that applies 21 thinning rules and 4 diagonal rules to the binary image. Additionally 12 pre-thinning rules are used to prepare the image for the thinning process.

$$cn(\mathbf{p}) = \frac{1}{2}\sum_{i=1}^{8} |\mathrm{val}(\mathbf{p}_{i \bmod 8}) - \mathrm{val}(\mathbf{p}_{i-1})|, \qquad (8)$$

In the step of minutiae extraction we define 8 neighboring pixels of the given pixel (x, y) as $N_0, N_1, \ldots N_7$. For detecting ridge ending the Crossing Number method [12] was used. In (8) p represents the currently processed pixel and $val(p)$ represents the value of that pixel. If the output of (8) is equal to 1, then given pixel is marked as ridge ending. The detection of ridge bifurcations is carried out using a set of 24 bifurcation masks. However, the presence of spikes, ridge breaks and other undesired formations may lead to many spurious minutiae being detected. Therefore every found minutia must be removed or validated in the postprocessing step [1]. Neighborhood of every minutia is inspected to detect and suitably remove structures such as ridge breaks, spikes, bridges, ladders, short ridges and holes.

V. CLASSIFICATION

Classification was done using five most used classes from Galton-Henry system: arch, tented arch, left and right loop, whorl [6]. To effectively determine the class of the fingerprint the singular points need to be detected. Therefore a method using curvature map was used as suggested in [9].

Image is divided into blocks of size $W \times W$ ($W = 7$ in our case). Blocks curvature is computed regarding the direction of 8 surrounding blocks

$$B_k = \frac{|O(B_1) - O(B_2)|}{2}, \qquad (9)$$

where B_1 and B_2 are blocks whose direction of selected block shows to. Singular points are obtained from curvature map using the following steps:

1.) Curvature threshold T_c and distance threshold T_d are defined (0.35 and 4 in our case).

2.) Let S_c define the set of singular points whose curvature is higher than T_c.

3.) Choose any two blocks B_i and B_j from set S_c. If distance between chosen blocks is less than T_d then the block with the smaller curvature is removed from the set S_c.

4.) Return to step 3 until size of set S_c is stable.

5.) Set S_c contains true singular points.

Although fingerprints of arch type don't contain any singular points a singular point is selected also for these fingerprints for the purpose of setting a reference point in the later stage. In this case the block with the highest curvature is taken as a reference point. Still, selected singular point is not used when determining the class of the fingerprint.

Type of the singular point is determined using Poincare Index method [11]. By knowing the number of cores N_c and deltas N_d fingerprint can be classified. Following heuristics are used:

1.) If $N_c = 2$ then class of the fingerprint is whorl.

2.) If $N_c = 0$ and $N_d = 0$ then class of the fingerprint is arch.

3.) If $N_c = 1$ and $N_d = 1$ then class of the fingerprint is tented arch or loop. Additionally analysis is needed. A narrow line is drawn between core and delta. The orientations along the line are parallel with the line at tented arch and are squared at loop. Let λ be the angle between drawn line and μ_1, $\mu_{2...}$ μ_n angles of local orientations along the line. If average sum

$$\sum_{i=1}^{n} \sin(\eta_i - \lambda) \qquad (10)$$

is less than a threshold (0.3 in our case) then fingerprint is of type tented arch, otherwise loop. Left loops are distinguished from right using (11) – if the result is less than zero, determined class is left loop, otherwise right loop.

$$(B_x - C_x)(D_y - C_y) - (B_y - C_y)(D_x - C_x), \qquad (11)$$

where indices x and y denote the x and y coordinate of the given singular point (Fig. 2).

4.) Otherwise class cannot be determined.

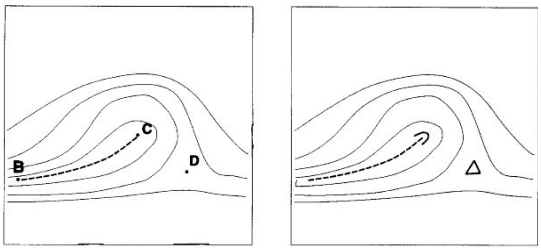

Figure 2. Distinguishing between left and right loops.

VI. MATCHING

Nowadays many different methods exist for matching two fingerprints. However, the most widely used method is based on comparing location, angle, direction and type of the minutiae. In our algorithm, minutiae are firstly converted to the polar coordinate system with respect to the corresponding reference point on which alignment is performed.

Last missing datum is the orientation of the reference point. If two cores exist, orientation is equal to the orientation of the line drawn through these points. Otherwise 16 directions (Fig. 3) around the singular point are defined, separated by step $\pi/8$. The reference point orientation is the direction, where the difference between the angle of the chosen direction and angle of local orientations around this direction is the smallest.

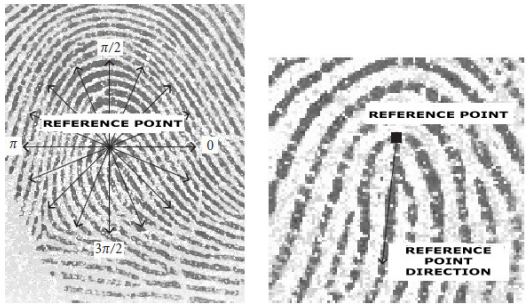

Figure 3. Determining the reference point direction.

After direction of reference point is obtained each minutia is presented with four parameters *(r, e, s, Φ)*:

- *r* – radius (distance from ref. point to minutia)
- *e* – polar angle
- *s* – type of minutia
- *Φ* – normalized minutia orientation with respect to the reference point, respectively.

Let $\mathbf{P}_i = \{(r_{i1}, e_{i1}, s_{i1}, \Phi_{i1})^T, \ldots (r_{iM}, e_{iM}, s_{iM}, \Phi_{iM})^T\}$ and $\mathbf{Q}_j = \{(r_{j1}, e_{j1}, s_{j1}, \Phi_{j1})^T, \ldots (r_{jN}, e_{jN}, s_{jN}, \Phi_{jN})^T\}$ denote the M minutiae in the template and N minutiae in the input image, respectively. The matching is performed as follows:

1.) Each minutia in $\mathbf{P}_i$ is matched against every minutia in $\mathbf{Q}_j$.

2.) Let M_0 and N_0 denote number of minutiae in template and input image and M_N number of successful matches. Stop matching when $M_N / \sqrt{N_0 M_0}$ is greater than threshold β (0.3 is our case).

3.) Take next template vector $\mathbf{Q}_{j+1}$.

Due to the presence of noise and nonlinear deformations of fingerprints it is impossible to exactly recover the position of each minutia with respect to the reference point. Therefore minutia based matching algorithm must be able to tolerate, to some extent, the deformations due to inexact extraction of minutiae.

In our matching algorithm this tolerance is achieved using Variable Bounding Box (Fig. 4). Let angle α_b and radius r_b denote the size of the bounding box. When the radius of a given minutia is small, a small deformation can cause extensive changes in minutia angle while at the same time it has little or no effect on radius. In this case α_b is increased and r_b is decreased. On contrary, when radius is large, small change in angle of minutia can cause big changes in its location. Then α_b is decreased and r_b is increased to negate the sum of all deformations between reference point and minutia.

VII. RESULTS

We have tested our system on four sets of fingerprint images used at FVC2002. Each set contains 8 images per finger from 100 individuals for a total of 800

images, which were captured with optical (DB1, DB2) and capacitive (DB3) scanner and the last set (DB4) was computer generated.

Each fingerprint in the set was matched against the remaining samples of the same finger to compute the False Non Match Rate (FNMR). Additionally, the first sample of each finger in the set was matched against the first sample of the remaining fingers to compute the False Match Rate (FMR).

Each step except enhancement performed satisfactory in our test. Fingerprint enhancement step failed to provide distinctive ridges when processing fingerprint images of poor quality. This had consequently an effect on the whole system. Moreover in some cases the algorithm was not able to provide correct location and number of singular points, which reflected in inaccurate location of a reference point. Consequently, the classification also failed. The behavior was mostly observed in images of poor quality. Overall results in Table 1 show that our prototype system is not completely reliable. Assuming poor quality images are reason for bad result, a custom test comprising of 64 fingerprint images of good quality was performed. Table 2 shows the FNRM and FMR indicators for images of good quality. FNMR has immensely dropped at sets DB1 and DB4 and a half at sets DB2 and DB3 as they contained mostly images of poor quality. In order for the verification system to be acceptable in practice, the response time of the system needs to be within a few seconds. Table 3 shows that our system does meet the response time requirements. Figure 4 shows Receiver Operating Characteristic (ROC) curves for all four sets. With empty rectangle and filled circle Equal Error Rate of last but one and first algorithm from FVC2002 is marked.

TABLE I. FNMR AND FMR INDICATORS AT THRESHOLD $\beta = 0.3$

Indicator	TEST SETS			
	DB1	***DB2***	***DB3***	***DB4***
FNMR (%)	89.3	88.6	91.2	81.3
FMR (%)	1.7	3.7	2.4	0.9

TABLE II. FNMR AND FMR INDICATORS AT THRESHOLD $\beta = 0.3$ USING ONLY FINGERPRINT IMAGES OF GOOD QUALITY

Indicator	TEST SETS			
	DB1	***DB2***	***DB3***	***DB4***
FNMR (%)	6.1	41.1	42.6	8.2
FMR (%)	0	0	0	0

VIII. CONCLUSION

The system alone is not yet reliable enough for commercial purposes (e.g. access control). Most errors occur due to poor quality of fingerprint images. Main improvement should be focused on determining ridge frequency in the fingerprint enhancement step which should be robust and resistant to noise. Determining the location of singular points also needs to be taken into consideration. We are currently investigating several approaches for singular point detection. To conclude, the developed system established a good framework and provides several opportunities for further enhancements.

TABLE III. AVERAGE CPU TIME FOR PROCESSING FINGERPRINT ON A SEMPRON 2000MHZ, 1500MB DDR2 SYSTEM

Indicator	TEST SETS			
	DB1	***DB2***	***DB3***	***DB4***
Time (s)	~ 2	~ 4	< 2	~ 3

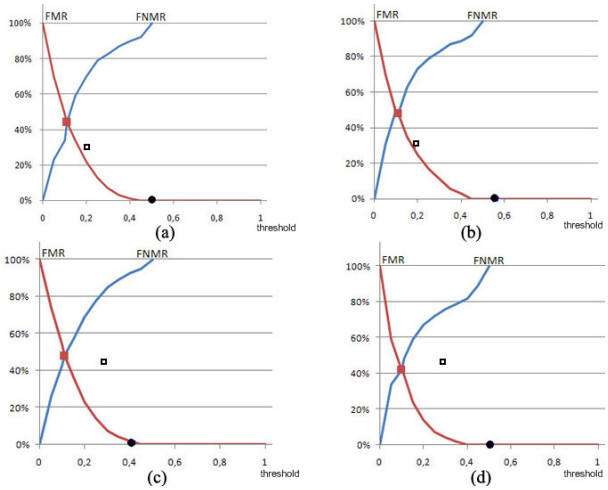

Figure 4. ROC curve for sets (a) DB1, (b) DB2, (c) DB3, (d) DB4

REFERENCES

[1] M.U. Akram, A. Tariq, S.A. Khan, S. Nasir, "Fingerprint image: pre and post-processing," International Journal of Biometrics 2008, Volume 1, Issue 1, pp. 63-80, 2008

[2] AForget .NET, http://www.aforgenet.com/framework/

[3] FVC2002, http://bias.csr.unibo.it/fvc2002/

[4] M.M. Hadhoud, W.S. El Kilani, M.I. Samaan, "An adaptive algorithm for fingerprints image enhancement using gabor filters," ICCES '07. International Conference on Computer Engineering & Systems, pp. 227-236, November 2007

[5] L. Hong, Y. Wan, A. Jain, "Fingerprint image enhancement: algorithm and performance evaluation," IEEE Transactions Pattern Analysis and Machine Intelligence, 20(8), pp. 777-789, 1998

[6] International Biometric Group, "The Henry Classification System", http://www.biometricgroup.com/Henry%20Fingerprint%20Classification.pdf

[7] M. Liu, X. Jiang, A.C. Kot, "Fingerprint Reference Point Detection," EURASIP Journal on Applied Signal Processing, Volume 2005, January, pp. 498-509, 2005

[8] P.M. Patil, S.R. Suralkar, F.B. Sheikh, "Rotation invariant thinning algorithm to detect ridge bifurcations for fingerprint identification," 17th IEEE International Conference on Tools with Artificial Intelligence, pp. 641-649, 2005

[9] J. Qi, M. Xie, "A robust algorithm for fingerprint singular point detection and image reference direction determination based on the analysis of curvature map," IEEE Conference on Cybernetics and Intelligent Systems, pp. 1051-1054, 2008

[10] J. Qi, M. Xie, "Segmentation of fingerprint images using the gradient vector field," IEEE Conference on Cybernetics and Intelligent Systems, pp. 543-545, September 2008

[11] T. Tang, X. Wu, M. Xiang, "An improved fingerprint singular point detection algorith based on continuous orientation field," International Symposium on Computer Science and Computational Technology ISCCST, 2nd Edition, pp. 454-457, 2008

[12] F. Zhao, X. Tang, "Preprocessing and postprocessing for skeleton-based fingerprint minutiae extraction", Pattern Recognition, Volume 40, Issue 1,pp. 1270-1281, 2007