Minutiae-based Algorithm for Automatic Fingerprint Identification

Edin H. Mulalić[1], Stevica S. Cvetković[2] and Saša V. Nikolić[3]

Abstract - **This paper describes a complete fingerprint identification algorithm that uses minutiae for representation and matching of fingerprints. The presented algorithm consists of three main steps: 1) Image enhancement, 2) Minutiae extraction and 3) Minutiae matching. Principal description of each step is given concluded with brief description of developed software with test examples.**

Keywords – **Fingerprint identification, Image processing.**

I. Introduction

Among all the biometric characteristics, fingerprints are one of the oldest, the most widely used and highly reliable. The most comprehensive explanation of all aspects of fingerprint identification could be found in [1].

The most of the methods for automatic fingerprint identification presented in literature has the similar global structure as our algorithm. At the beginning, image enhancement is applied in order to improve the quality of the input fingerprint image. Then, algorithm extracts local characteristics of the fingerprint – minutiae. Finally, minutiae matching is performed by comparing minutiae form input fingerprint with a set of one or more template fingerprints stored in database. The rest of the paper will give principal description of each step of the algorithm.

II. Image Enhancement

The performance of minutiae extraction algorithms relies heavily on the quality of the input fingerprint images. Due to skin conditions, sensor noise and incorrect finger pressure, a significant percentage of fingerprint images (approximately 10%) are of poor quality. In general, there are two types of degradation that could be observed in fingerprint images:

- The ridges are not strictly continuous i.e. the ridges have small breaks (gaps),
- Due to the presence of noise which links ridges, parallel ridges are not well separated.

The most widely used technique for fingerprint image enhancement is based on *contextual filters*. In conventional image filtering, only a single filter kernel is used for convolution throughout the image. In contextual filtering, the filter characteristics change according to the local context, where the context is often defined by the local ridge orientation and local ridge frequency. Appropriate contextual filter should posses the following characteristics:

- It should provide a low pass (averaging) effect along the ridge direction with the aim of linking small gaps and filling impurities due to pores or noise.
- It should have a band pass (differentiating) effect in the direction orthogonal to the ridges in order to increase the discrimination between ridges and valleys and top separate parallel linked ridges.

One of the first using of contextual filters for fingerprint image enhancement was performed by O'Gorman [2] and Mehtre [3]. Since then, numerous filtering methods have been proposed in literature, both in spatial and frequency domain. Hong et al. [4] proposed an effective method based on Gabor filters and reported to achieve good performance. In order to improve performances of the previous method Yang et al. [5] modified method described in [4]. Although they reported better accuracy, computational complexity of their algorithm is too high.

Method which is used in this paper is similar in principle to the one described in [4]. The method assumes that parallel ridges and valleys exhibit some ideal sinusoidal plane waves associated with some noise. In other words, the 1-D signal orthogonal to the local orientation is approximately a digital sinusoidal wave. After convolving the image with corresponding Gabor filter which is tuned to a specific local ridge orientation and frequency, image could be efficiently enhanced. An even symmetric two dimensional Gabor filter has the following form in the spatial domain:

$$G_{\theta,f,\delta_x,\delta_y}(x,y) = \exp\left[-\frac{1}{2}\left(\frac{{x_\theta}^2}{\sigma_x^2}+\frac{{y_\theta}^2}{\sigma_y^2}\right)\right]\cos(2\pi f x_\theta),$$

$$x_\theta = x\cos(90-\theta)+\sin(90-\theta) = x\sin\theta + y\cos\theta,$$
$$y_\theta = -x\sin(90-\theta)+\cos(90-\theta) = -x\cos\theta + y\sin\theta \qquad (1)$$

where $[x_\theta, y_\theta]$ are the coordinates of $[x,y]$ after a clockwise rotation of the Cartesian axes by an angle of $(90^\circ-\theta)$. The parameters of the previous equation are:

a) θ is the local orientation of the ridges. It is an angle of the ridge with the x-axis computed by examining the gradients of pixel intensities in the x and y directions within the local block (16×16)of the image.

b) f is the local frequency of the sinusoidal plane wave. It corresponds to the reciprocal value of inter-ridge distance in fingerprints images. Although some algorithms in literature

[1]Edin H. Mulalić is student at the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: edinmulalic@yahoo.com

[2]Stevica S. Cvetkovic is PhD student at the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: stevica_cvetkovic@yahoo.com

[3]Saša V. Nikolić is with the Faculty of Electronic Engineering, University of Niš, Aleksandra Medvedeva 14, 18000 Niš, Serbia, E-mail: caci@elfak.ni.ac.yu

calculate f for each block of the image, empirical value can be used [6]. Therefore, the frequency is set to $f = \frac{1}{8}$.

c) σ_x, σ_y are the standard deviations of the Gaussian envelope along the x and y-axes, respectively. Based on empirical data [4], these values were set to $\sigma_x = \sigma_y = 4$.

To apply Gabor filters to an image, the four parameters $\theta, f, \sigma_x, \sigma_y$ must be specified for each pixel. Values of f, σ_x, σ_y are based on empirical data, thus only θ should be calculated for each block of the image. To make the enhancement faster, instead of “online” computing the best-suited contextual filter for each pixel, a set of eight filters (named “filter bank”) for eight discrete values of $\theta_k = k\frac{\pi}{8}, \quad k = 0,1,...,7$ are a priori created and stored. Then, after calculating local orientation which is discretized to the closest value of θ_k, each pixel of the image is convolved with corresponding filter from precomputed Gabor filter bank.

III. MINUTIAE EXTRACTION

Most of proposed minutiae extraction algorithms operate on binary images. The binary images obtained by the binarization process are usually submitted to a thinning phase of algorithm which reduces ridge line thickness to one pixel. Some authors have proposed minutiae extraction approaches that work directly on the gray-scale images, without binarization and thinning, mainly because of following reasons [1]:

- a significant amount of information may be lost during the binarization process;
- binarization and thinning are time consuming;
- thinning may introduce a large number of spurious minutiae;
- in the absence of an a priori enhancement step, most of the binarization techniques do not provide satisfactory results when applied to low-quality images.

But most of these drawbacks can be avoided by using efficient and effective algorithms for image enhancement, binarization, thinning, false minutiae recognition and matching. On the other side, binarization and thinning provide numerous advantages in minutiae recognition phase. Obvious motivation to apply thinning process is to simplify process of recognizing minutiae and minutiae’s position. Method for minutiae extracting presented in this paper is based on images processed by binarization and thinning algorithms. For binarization, global threshold algorithm is used. Among many thinning algorithms described in [7], we chose to implement Nagendraprasad-Wang-Gupta iterative algorithm [8].

The most popular method for minutiae recognition is the Crossing Number (*CN*) approach. The in a binary image is defined by Arcelli and Baja in 1984. Mathematically, crossing number *CN(p)* of a pixel *p*, can be calculated as half the sum of the differences between pairs of adjacent pixels in the 8-neighborhood of *p*:

$$CN(p) = 0.5 * \sum_{i=0}^{7} \left| val(p_i) - val(p_{(i+1) \bmod 8}) \right| \tag{2}$$

where p is central pixel, $p_0, p_1, ..., p_7$ are ordered set of pixels which describes 8-neighborhood of p and $val(p_i) \in \{0,1\}$. Using *CN(p)*, each pixel can be classified into one of five categories described in Table 1.

CN(p)	Type of pixel
0	Isolated point
1	End of ridge
2	Continuing ridge point
3	Bifurcation point
4	Crossing point

Table 1. Classification of pixels according to its crossing number

False Minutiae Rejection

The presence of noise in poor-quality images and imperfect thinning are two main reasons that cause a large number of extraction errors - dropping of true minutiae and production of false minutiae. Therefore, additional post processing is necessary to filter extracted minutiae and keep only set of true minutiae.

Although the main goal of the filtering process is to remove false minutiae, keeping true minutiae is more important for reliable minutiae matching. There are numerous methods to achieve that goal and most of them are based on set off heuristic rules. Examples of those rules are:

- if the break in the ridges is less than threshold value and no other pixel passes through it, then the break is connected;
- if the angle between the branch and the trunk is greater than 70º and less than 110º, then the branch is removed
- short ridges are removed on the basis of the relationship between the ridge length and the average distance between ridges;
- a ridge ending point that is connected to a bifurcation point, and is below a certain threshold distance is eliminated;
- two bifurcations are eliminated if distance between them is less than threshold value (10-15 pixels).

One effective and efficient method based on set off heuristic rules is presented in [9]. The setting of parameters, mostly some thresholds, should be obtained dynamically. Parameters adaptive to the image usually perform better than fixed values.

Experimental results given in this paper are obtained by using an algorithm for filtering based on examining local neighborhood around potential minutiae point, proposed by Tico and Kousmanen [10].

IV. MINUTIAE MATCHING

Fingerprint matching is a process of comparing an input fingerprint with a set of one or more template fingerprints. The return value is either a degree of similarity (for example, a score between 0 and 1) or a binary decision (matched/not matched). There are three main families of fingerprint matching techniques: correlation based matching, minutiae based matching and ridge-feature based matching [1]. Since it is widely accepted that minutiae set is the most discriminating and reliable feature, in modern systems for automatic fingerprints identification, that approach is the most popular. But, those are not the only reasons for popularity of minutiae based approach. Prior to the matching process, feature information has to be extracted from all the template images. The amount of extracted information from templates has to be low because of saving memory. Also, it has to be possible to efficiently traverse the structure which describes extracted information in order to check for necessary similarities. Those additional demands also favor minutiae based approach. In some applications it is possible to use hybrid approach, combining two or even all three approaches. This paper is focused on minutiae based approach.

Each minutia may be described by a number of attributes, including its location in the fingerprint image, orientation, type (e.g., ridge termination or ridge bifurcation), a weight based on the quality of the fingerprint image in the neighborhood of the minutia, and so on. *Local minutia structure* can be described in many different ways. One simple and most obvious would be a triplet (x, y, Ψ), where x and y are minutia location coordinates and Ψ is the local ridge direction in given coordinate system. Yang and Verbauwhede [11] proposed derived local structure as well as an algorithm based on it and experimental results given in this paper are based on implementation of that algorithm. The algorithm is described in the rest of the section.

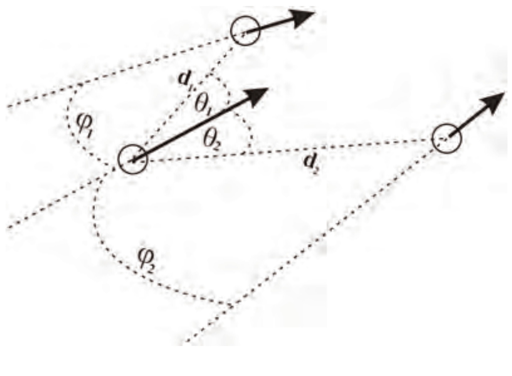

Fig. 1. Example of minutiae local structure

Local structure L_M can be described by Eq. 3:

$$L_M = \{(d_1,\varphi_1,\vartheta_1),(d_2,\varphi_2,\vartheta_2),...(d_N,\varphi_N,\vartheta_N),\Psi\} \tag{3}$$

where N is the number of nearest neighbors of minutiae *M*, d_i (*i* = 1,2,...*N*) describes the distance between the selected minutia *M* and its i^{th} nearest neighbor, φ_i (*i* = 1,2,...*N*) is the related radial angle between *M* and its i^{th} nearest neighbor, ϑ_i (*i* = 1,2,...*N*) represents the related position angle of the i^{th} nearest neighbor and Ψ is the local ridge direction of minutia *M.* An example is illustrated in Fig 1.

If (x_M, y_M) denotes x and y coordinates of minutiae M, (x_i, y_i) denotes x and y coordinates of i^{th} nearest neighbor, and $diff(\alpha,\beta)$ calculate the difference between angles α and β and converts the result to range $[0,2\pi)$, than parameters $(d_i,\varphi_i,\vartheta_i)$ (*i* = 1,2,..,N) can be calculated using following formula:

$$\begin{aligned} d_i &= \sqrt{(x_i - x_M)^2 + (y_i - y_M)^2}, \\ \varphi_i &= diff(\Psi_i - \Psi), \\ \vartheta_i &= diff(\arctan(\frac{y_i - y_M}{x_i - x_M}), \Psi) \end{aligned} \tag{4}$$

Proposed matching algorithm is based on calculating *minutiae similarity factor* and *image similarity factor* using vector of threshold values $(Th_d, Th_\varphi, Th_\vartheta, Th_\Psi, Th_{MSF}, Th_{ISF})$.

Let *M* be a minutiae from input (query) image with its local structure $L_M = \{(d_1,\varphi_1,\vartheta_1),(d_2,\varphi_2,\vartheta_2),...(d_N,\varphi_N,\vartheta_N),\Psi\}$, and *M'* be a minutiae from template (database) image described by its local structure $L_{M'} = \{(d'_1,\phi_1,\vartheta_1),(d'_2,\phi_2,\vartheta_2),...(d'_N,\phi_N,\vartheta_N),\Psi'\}$. *Minutia similarity factor (MSF)* represents degree of similarity between two minutiae M and M'. It can be calculated by observing their local structures L_M and $L_{M'}$. First of all, if $\left|\Psi - \Psi'\right| > Th_\Psi$, than *MSF* = 0, M and M' are not matched and another pair of minutiae can be checked. Otherwise, it is necessary to investigate similarity of neighborhoods of minutiae M and M'. If i^{th} neighbor of minutiae M and j^{th} neighbor of minutiae M' satisfy set of conditions described by Eq. 5:

$$\begin{aligned} \left|d_i - d'_j\right| &< Th_d \\ \left|\varphi_i - \varphi'_j\right| &< Th_\varphi \\ \left|\vartheta_i - \vartheta'_j\right| &< Th_\vartheta \end{aligned} \tag{5}$$

than those two neighbors can be marked as "matched neighbors". *MSF* is equal to the total number of "matched neighbors". If $MSF > Th_{MSF}$, than minutiae M and M' represent "matched minutiae pair". After comparing all minutiae from input image with minutiae from template image, total number of minutiae matched pairs (NUM_{MMP}) is obtained. *Image similarity factor* is calculated using Eq. 6:

$$ISF = \frac{NUM_{MMP}}{\max(NUM_{input}, NUM_{template})} \tag{6}$$

where NUM_{input} and $NUM_{template}$ are total number of minutiae in the input and template images, respectively. Two images are considered to represent the same fingerprint if $ISF > Th_{ISF}$.

V. Experimental Results

For implementation of the presented algorithm we used VS.NET (C#). All experiments have been done on a Pentium IV (3GHz, 1GB RAM) machine. Instead of using images obtained by our own sensors, we used FVC2002 DB_2 database [12]. We used 4, 5 and 6 and 7 nearest neighbors for minutiae matching step of the algorithm. FAR (False Acceptance Rate) and FRR (False Rejection Rate) factors had satisfying values only when 5 and 6-neighborhoods where used. Those results are consistent with results presented in [11]. Figure 2. presents some results of created testing software.

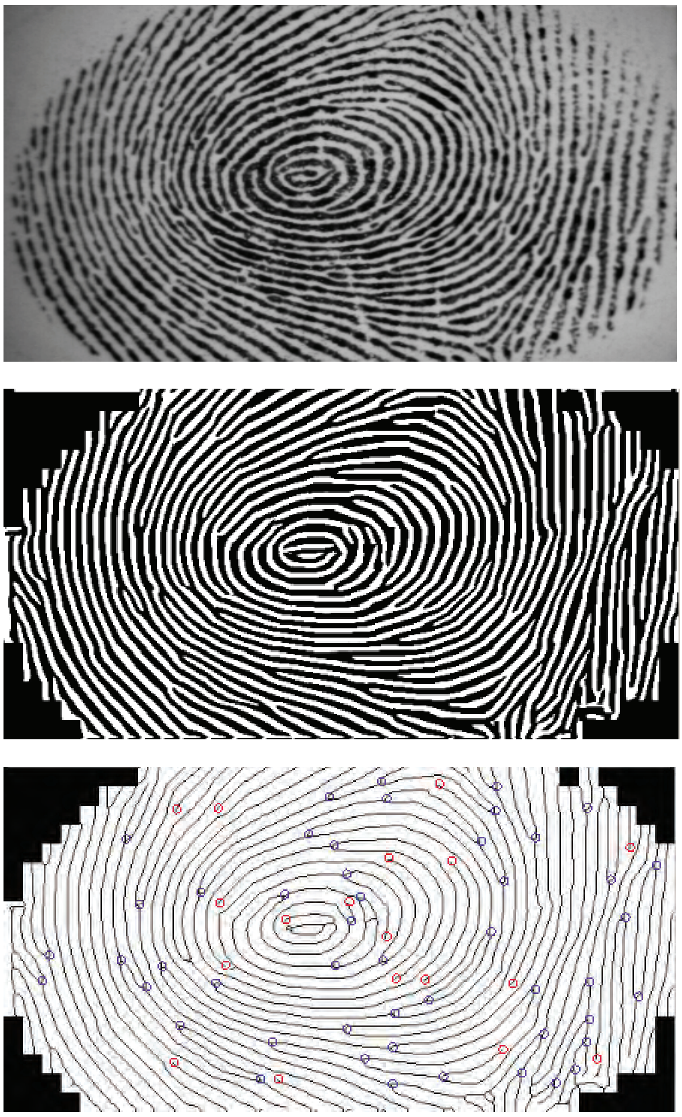

Fig. 2. Example of tested fingerprint image
(original, image after enhancement, image after minutiae extraction)

VI. Conclusion and Future Work

Created software achieved excellent results with images that had good quality, but had some problems in recognizing pour quality images. Those problems were identified and some ideas for their solving are:

a) using dynamically calculated frequency improves image enhancement process. Improving this part of the system will allows work with not so good quality images
b) grouping images by type of fingerprint improves matching process by reducing number of images which has to be compared with input image. This will improve speed of software.

Mentioned improvements will be focus of our future work.

References

[1] D. Maltoni, D. Maio, A. K. Jain, S. Prabhakar, “Handbook of fingerprint recognition”, New York: Springer-Verlag, 2003.

[2] O'Gorman, L., Nickerson, J.V., “An approach to fingerprint filter design”, Pattern Recognition 22 (1), 29–38, 1989.

[3] Mehtre, B.M., “Fingerprint image analysis for automatic identification”, Machine Vision, Appl. 22 (6), 124–139, 1993.

[4] L. Hong, Y. Wan, and A. Jain, "Fingerprint Image Enhancement: Algorithm and Performance Evaluation", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 20, No. 8, pp. 777-789, 1998.

[5] J. Yang, L. Liu, T. Jiang, and Y. Fan. “A modified gabor filter design method for fingerprint image enhancement”, Pattern Recognition Letters, 24:1805-1817, 2003.

[6] A. Ross, A. K. Jain and J. Reisman, "A Hybrid Fingerprint Matcher", Pattern Recognition, Vol. 36, No. 7, pp. 1661-1673, 2003.

[7] M. Couprie, “Note on fifteen 2D parallel thinning algorithms”, Internal Report, Institut Gaspard Monge, 2006.

[8] M.V. Nagendraprasad, P.S.P. Wang, and A. Gupta “Algorithms for thinning and rethickening binary digital patterns”, Digital Signal Processing 3, 97–102, 1993.

[9] H. Lu, X. Jiang and Wei-Yun Yau, "Effective and Efficient Fingerprint Image Postprocessing", Proc. ICARCV2002, Singapore, pp. 985-989, Dec. 2002.

[10] Tico, M., and Kuosmanen, P. “An algorithm for fingerprint image postprocessing”, in Proceedings of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers , vol. 2, pp. 1735–1739, November 2000.

[11] S. Yang and I. Verbauwhede, "A secure fingerprint matching technique", Proc. ACM Workshop on Biometrics: Methods and Applications, pp. 89-94, November 2003.

[12] FVC2002 - Fingerprint Verification Competition, http://bias.csr.unibo.it/fvc2002