Fingerprint Feature Extraction Using Midpoint ridge Contour method and Neural Network

Bhupesh Gour
Asst. Prof. Dept. of Computer Sc. & Engg.
All Saints' College of Technology, Bhopal.

T. K. Bandopadhyaya
Professor, Bansal Institute of
Science and Technology, Bhopal.

Sudhir Sharma
Professor, RGPV,
Bhopal.

Abstract

Automatic minutiae detection is an extremely critical process especially in low quality fingerprints. Most automatic systems for fingerprint comparison are based on minutiae matching. Minutiae are essentially terminations and bifurcations of the ridge lines that constitute a fingerprint pattern. A minutia extraction approach has been presented using the midpoint ridge contours. Our method do not requires thinning of fingerprint image but it can work with actual size ridges as it has been acquired. On the fingerprint image segmentation, normalization and contrast enhancement operations are required to be perform for clear identification of minutiae points. Color coding scheme has been used so that each ridge line should be scanned only once. Feature point locations that are being identified by midpoint ridge contours method are stored and passed to multi layer Perceptron trained with backpropagation algorithm. The result achieved is compared with those obtained through a method based on image thinning. The method proposed takes less time and do not detect any false minutiae.

Keywords: Minutiae, Segmentation, Image Normalization, Contrast Enhancement, midpoint ridge contour, multilayer Perceptron.

1 Introduction

An accurate representation of the fingerprint image is critical to automatic fingerprint identification systems, because most deployed commercial large-scale systems are dependent on feature-based matching, even though correlation-based matching methods may have high performance when additional computation time is available. Among all the fingerprint features, minutia point features maps are unique enough to provide robust discriminative capacity [1], the minutiae feature representation reduces the complex fingerprint recognition problem to a point pattern matching problem. In order to achieve high-accuracy minutiae with varied quality fingerprint images, segmentation algorithm needs to separate foreground from noisy background without excluding any ridge-valley regions and not include meaningless background. Image enhancement algorithm needs to keep the original ridge flow pattern without altering the singularity, join broken ridges, clean artifacts between pseudo-parallel ridges, and not introduce false information. Finally minutiae detection algorithm needs to locate efficiently and accurately the minutiae points. There are a lot of minutiae extraction methods available in the literature. Based on image detection domain, there are roughly four categories of detection algorithms. First category of methods extracts minutiae directly from the gray-level image [2, 3] without using binarization and thinning processes. Second category of methods extracts minutiae from binary image profile patterns [4, 5]. Third category of methods extracts minutiae via machine learning methods [6, 7]. The fourth categories of methods extract minutiae from binary skeletons [8, 9]. Most feature extraction algorithms uses thinning for extraction of minutiae from a fingerprint image [10]. Thinning is a lossy and computationally expensive operation and the accuracy of the output skeletal representation varies for different algorithms. In this paper we introduce the use of midpoint ridge contour representation as an efficient alternative for extraction of minutiae from fingerprint images. The first step is segmentation, to separate foreground from background of fingerprint image. A 64 X 64 region is extracted from fingerprint image. The grayscale intensities in this 64 X 64 region are normalized to a constant mean and variance to remove the effects of sensor noise and grayscale variations due to finger pressure differences. After the normalization, we enhance the contrast of the ridges by filtering this 64 X 64 normalized window by appropriately tuned Gabor filter [11]. Processed fingerprint image is then scanned from top to bottom and left to right, and transitions from white (background) to black (foreground) are detected. The length vector is calculated in all the eight directions of contour is calculated. Each contour element represents a pixel on the contour, contains fields for the x, y coordinates of the pixel. Midpoint ridge contour algorithm is described in section 3.

Manuscript received July 5, 2008.
Manuscript revised July 20, 2008.

2 Fingerprint Image Processing

Processing of fingerprint image is necessary to: (i) improve the clarity of ridge structures of fingerprint images (ii) maintain their integrity, (iii) avoid introduction of spurious structures or artifacts, and (iv) retain the connectivity of the ridges while maintaining separation between ridges. Fingerprint image processing operation [12] are- image segmentation, image normalization, image contrast enhancement.

2.1 Image Normalization:

Let I(x; y) denote the grayscale value at pixel (x; y), M and V , the estimated mean and variance of grayscale values in this 64 X 64 window, respectively, and N(x; y), the normalized grayscale value at pixel (x; y). For all the pixels in the window, the normalized image is defined as:

$$N(x,y) = \begin{cases} M_0 + \sqrt{\frac{V_0 \times (I(x,y)-M)^2}{V}}, & if \ \ I(x,y) > M \\ M_0 - \sqrt{\frac{V_0 \times (I(x,y)-M)^2}{V}}, & otherwise, \end{cases}$$

where *M*0 and *V*0 are the desired mean and variance values, respectively. Normalization is a pixel-wise operation and does not change the clarity of the ridge and valley structures. For our experiments, we set the values of both *M*0 and *V*0 to 100. The values of *M*0 and *V*0 should be the same across all the training and test sets.

2.2 Image Contrast Enhancement:

We enhance the contrast of the ridges by filtering this 64 X 64 normalized window with an appropriately tuned Gabor filter. An even symmetric Gabor filter has the following general form in the spatial domain:

$$\begin{aligned} G(x,y;f,\theta) &= exp\left\{\frac{-1}{2}\left[\frac{x'^2}{\delta_x^2} + \frac{y'^2}{\delta_y^2}\right]\right\} cos(2\pi f x'), \\ x' &= x sin\theta + y cos\theta, \\ y' &= x cos\theta - y sin\theta, \end{aligned}$$

where *f* is the frequency of the sinusoidal plane wave along the direction θ from the *x*-axis, and δ_x and δ_y are the space constants of the Gaussian envelope along *x* and *y* axes, respectively. We set the frequency *f* of the Gabor filter to the average ridge frequency (1=*K*), where *K* is the average inter-ridge distance. The average inter-ridge distance is approximately 10 pixels in a 500 *dpi* fingerprint image. The values of parameters δ_x and δ_y for Gabor filters were empirically determined and each is set to 4 (about half the average inter-ridge distance). Since the extracted region is in the direction of the potential minutia, the filter is tuned to 0^o direction. We perform the filtering in the spatial domain with a mask size of 33£33. The filter values smaller than 0.05 are ignored and the symmetry of the filter is exploited to speed up the convolution. We extract a 32 X 32 region from the center of the 64 X 64 filtered regions.

3 Minutiae Detection

Most automatic systems for fingerprint comparison are based on minutiae matching [12, 22]. The American National Standards Institute has proposed minutiae classification based on four classes: terminations, bifurcations, trifurcations (or crossovers) and undetermined [l5]. In this work we adopt the identification model used by the Federal Bureau of Investigation. This model, adopted in most automatic systems, is based on a two-class minutiae classification: termination and bifurcation. When a ridge line terminates or intersects another ridge line (originating a minutia) the algorithm stops and gives the characteristics (coordinates location) of the minutia found. To extract all the ridge lines in the image and, consequently, detecting all the minutiae it is needed to examine each ridge line only once and locating the intersections and terminations of ridge lines. Our technique uses color coding method to trace each ridge line only once. When any ridge line is being traced out, all of its pixels between starting and ending points are colored red, so that when we again search for next black ridge line in the fingerprint image no previously traced ridge line come into the picture. Fingerprint image is divided into small portions of 16 x 16 pixels, so each square area will be having 256 pixels. For extracting feature points from a 16X16 portion we have to first initialize current scanning position (x,y) to (0,0), until end of image portion, scan the image from current position left to right and top to bottom. Place contour on current pixel (black) found in the image, store that point as starting of ridge minutiae position. Calculate the length vector in each contour direction. Find the maximum and minimum length vector in each direction. Direction of minimum length vector will be width of the ridge line, count the pixels along the width. Store all pixel positions along the width of the ridge in to an array and check contour positions for all pixels in that array. If any of these pixels are white in color then store that pixel as minutiae position, if minutiae shows ending of ridge line then store that point as ending of ridge position; change the color of current starting and ending of ridge to red color and otherwise move to the pixel at the center in ridge and move to the next pixel into the given direction.

3.1 Midpoint Ridge Contour (MRC) Algorithm.

1. Initialize current scanning position (x,y) to (0,0).
2. While x_end, y_end not comes; scan the image from current position left to right and top to bottom.
3. Place contour on current pixel (black) found in the image, store that point as start_ridge minutiae position.
4. Calculate the length vector in each contour direction.
5. Find the maximum (Max_ln) and minimum (Min_ln) length vector in each direction.
6. Direction of minimum length vector will be width of the ridge line, count the pixels along the width.
7. Store all pixel positions along the width of the ridge in to an array Wd (1 to n).
8. Check contour positions for all pixels in an array from Wd(2) to Wd(n-1).
9. If any of these pixels are white in color then store that pixel as minutiae position.
    a. If minutiae shows ending of ridge line then store that point as end_ridge position; change the color of current ridge (between start_ridge to end_ridge) to red color go to step 2.
10. Else move to the pixel at the center in the array and move to the next pixel in to direction Max_ln go to Step 4.

4. Feature Optimization.

Once all the possible feature points has been extracted form a fingerprint by using Midpoint ridge contour [16] method artifacts removal is also important. False feature points must be removed from it to improve extraction accuracy. Artifact removal algorithms typically look for anomalies like adjacent minutiae, intersecting perpendicular edges, and statistically impossible minutiae. After artifacts are detected and removed, the extraction of certain minutiae can begin. The ridge ending and bifurcation are the two features that are easier to pick out accurately. But it becomes complex job when correct feature points need to be selected from a bigger collection of feature points. In order to automate the recognition of these two features, the recognition program must be taught to accept an input pattern as a feature or a non-feature. This training process is done through the simulation of a neural network comprising of layers of Perceptron. In effect, this network of multilayered Perceptron guides a minutiae recognition program to make desirable scans.

4.1. Perceptron Model

A perceptron is an automatic system which can be taught to understand a concept. It learns by being exposed repeatedly to examples of what a concept is and what a concept is not. The teaching process requires the input of examples paired with desired outputs. The input examples are real values that signify a meaning in some context. Then, the training set can be defined as Dm= {(e1, d1), (e2, d2).., (em,dm)}, where m denotes the number of examples, e1 through em are the input examples and d1 through dm are the associated desired outputs. Furthermore, a vector of weights is defined as w1, w2 through wm. See a representation of a Perceptron in Figure 1.

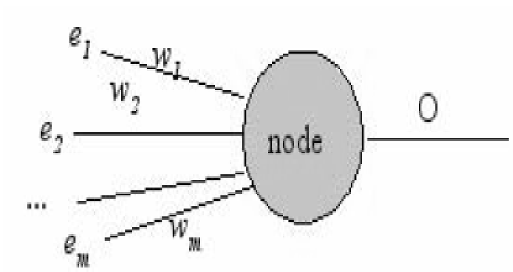

Fig. 1. A Perceptron Model.

4.2. Working of Perceptron Model

The Perceptron network used in minutiae recognition consists of 3 layers. There are nine neurons in the first layer, each processing an associated element from the input vector. This first layer is simply an input layer, where no computation takes place. The second layer, also known as the hidden layer, consists of five neurons. This second layer evaluates the results of the first nine neurons. Finally, the last layer has only one neuron, acting as a single output. The MLP is trained using Back propagation Neural Network [17], in off line mode, because training only needs to occur once. A set of known desirable and undesirable patterns for both minutiae are provided to the program simulating the network. Figure 2 contains training examples.

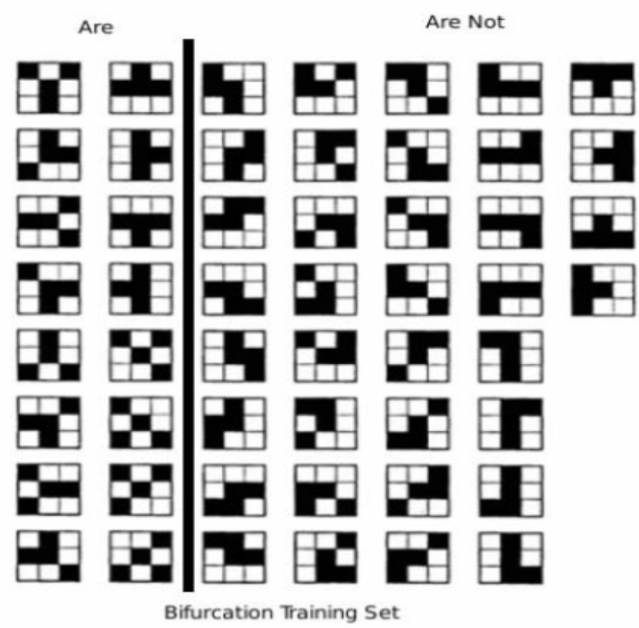

Fig. 2 A bifurcation training set grouped by desired output

Backpropagation training algorithm is as follows:

Step 1: Initialize the weights.

Step 2: While stopping condition is false, execute steps 3 to 10.

Step 3: For each training pair X:t, do step 4 to 9

Step 4: Each input unit X_i , I = 1,2,3,.n receives the input signal, x, and broadcast it to the next layer.

Step 5: For each hidden layer neuron denoted as $Z_{j,}$ j = 1,2,3,..,p.

$$Z_{inj} = v_{oj} + \Sigma x_I v_{ij}$$
$$Z_j = f(z_{inj})$$

Broadcast z_j to the next layer.

Step 6: For each output neuron Y_k, k=1,2..,m

$$y_{ink} = w_{ok} + \Sigma z_j w_{jk}$$
$$y_k = f(y_{ink})$$

Step 7: Compute δ_k for each output neuron, $Y_{k,}$

$$\delta_k = (t_k - y_k) f'(y_{ink})$$
$$\Delta w_{jk} = \alpha \delta_k z_j$$
$$\Delta w_{ok} = \alpha \delta_k \quad \text{since } z_o = 1$$

Step 8: For Each hidden neuron,

$$\delta_{inj} = \Sigma \delta_k w_{jk} \quad {}_{j-1,2,3\ldots} p$$
$$\delta_j = \delta_{inj} f'(Z_{inj})$$
$$\Delta v_{ij} = \alpha \delta_j x_i$$
$$\Delta v_{oj} = \alpha \delta_j$$

Step 9: $w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$

$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$

Step10**:** Test for stopping condition.

Once the recognition program has been fully trained, each 3x3 pixel scan of the fingerprint image of the locations identified by “Midpoint ridge Contour” algorithm is passed to the program to identify as either an accepted minutia or not. If the 3X3 scan has centered onto an accepted minutia, an output of “1” results, otherwise an output of “0”results. For each input scan that resembles minutiae, its position in relation to the core point of the fingerprint is recorded. This information is useful in the fingerprint matching process.

Experimental Result

Minutiae points from 100 fingerprints in database FVC2002 (DB1_a) has been extracted using thinning and our proposed technique (Midpoint ridge Contour method) and compared their performance. The comparison on 12 fingerprints out of 100 is given in the table 1. Table values shows that our proposed approach is much efficient and can extract minutiae points in much better way and in grater number then thinning based method. The method proposed takes less time and detect no false minutiae.

Fingerprint Image	**Actual No. of Minutiae**	**No. of Minutiae by thinning**	**No. of Minutiae by proposed method**
1_1.tif	42	36	36
2_1.tif	35	32	34
3_1.tif	38	34	36
4_1.tif	29	27	28
5_1.tif	45	40	43
6_1.tif	35	30	31
7_1.tif	44	39	41
8_1.tif	46	42	44
9_1.tif	37	34	36
10_1.tif	39	32	34
11_1.tif	32	28	30
12_1.tif	41	37	39

Table. 1. Comparison between Thinning and proposed method.

References

[1] G. Parziale and E. Diaz-Santana. "The surround imager: A multi-camera touch-less device to acquire 3d rolled-equivalent fingerprints", *in Proc. of IAPR Int. Conf. on Biometrics, LNCS*, volume 3832, pages pp.244–250, 2006.

[2] L. Jinxiang, H. Zhongyang, and C. Kap Luk. Direct minutiae extraction from gray-level fingerprint image by relationship examination. *In International Conference on Image Processing(ICIP),* volume 2, pages 427–430 vol.2, 2000.

[3] D. Maio and D. Maltoni. Direct gray-scale minutiae detection in fingerprints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):27–40, 1997.

[4] B. Bir and T. Xuejun. "Fingerprint indexing based on novel features of minutiae triplets". Pattern Analysis and Machine Intelligence, *IEEE Transactions* on, 25(5):616–622, 2003.

[5] J. Xudong and Y. Wei-Yun. "Fingerprint minutiae matching based on the local and global structures*". In Proc. of International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1038–1041 vol.2, 2000.

[6] B. Bir and T. Xuejun. "Fingerprint indexing based on novel features of minutiae triplets". Pattern Analysis and Machine Intelligence*, IEEE Transactions on, 25(5):616–622*, 2003.

[7] S. Prabhakar, A. K. Jain, and S. Pankanti. Learning fingerprint minutiae location and type. *Pattern Recognition,* 36(8):1847–1857, 2003.

[8] S. A.M.Bazen, M. Van Otterlo and M. Poel. "Areinforcement learning agent for minutiae extraction from fingerprints". *In Proc. BNAIC*, pages pp.329–336, 2001.

[9]. A. K. Jain, L. Hong, and B. R. "On-line fingerprint verification". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):302–314, 1997.

[10]. Adrian Lim Hooi Jin et al, "Fingerprint Identification and Recognition Using Backpropagation Neural Network", *Students Conference on Research and Development Proceedings, IEEE*, 2002

[11] A. K. Jain, S. Prabhakar, L. Hong, S. Pankanti, Filterbank-based fingerprint matching*, IEEE Transactions on Image Processing* 9(5)(2000), 846-859.

[12] L. Hong, Y. Wan, A. K. Jain, Fingerprint image enhancement: algorithm and performance evaluation, *IEEE Trans. Pattern Analalysis and Machine Intelligence*, 20(8)(1998) 777-789.

[13]. J. Hollingum, "Automated Fingerprint Analysis Offers Fast Verification," *Sensor Review,* vol. 12, no. 3, pp. 12-15,1992.

[14]. B.M. Mehtre and N.N. Murthy, "A Minutia Based Fingerprint Identification System," *Proc. Second Int'l Conf. Advances in Pattern Recognition and Digital Techniques*, Calcutta 1986.

[15]. American National Standards Institute, Fingerprint Identification- Data Format for Information Interchange. New York, 1986.

[16]. Govindaraju, Zhixin Shi, "Feature Extraction Using a Chaincoded Contour Representation of Fingerprint Images", *EDAR, Department of Computer Science and Engineering*, New York 14226 March 24, 2003

[17]. Bhupesh Gour, Krishna Singh, "Fast Fingerprint Identification System Using Minutiae Matching with Back Propagation Neural Network and Self-Organizing Map.", *Proceedings of GLOGIFT*, Bhopal, December, 2005.

Bhupesh Gour received the B.E. degree in Computer Science & Engineering from Government Engineering College Jabalpur in 2000 and M-Tech in Computer Technology & Application from Rajiv Gandhi Technological University in 2005. He is having total eight years of industrial and academics experience. Presently he is working in All Saints' College of Technology, Bhopal as Head Department of Computer Science & Engineering.