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 image profile patterns [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.
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 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:

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

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:

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

$$x' = x \sin \theta + y \cos \theta,$$

$$y' = x \cos \theta - y \sin \theta,$$

where $f$ is the frequency of the sinusoidal plane wave along the direction $\theta$ from the x-axis, and $\delta_x$ and $\delta_y$ 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 $(1\approx K)$, 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 $\delta_x$ and $\delta_y$ 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 $0^\circ$ direction. We perform the filtering in the spatial domain with a mask size of 33X33. 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 [15]. 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](image)

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. 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.
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 $X_i, i = 1, 2, 3, \ldots n$ receives the input signal, $x$, and broadcast it to the next layer.

Step 5: For each hidden layer neuron denoted as $Z_j, j = 1, 2, 3, \ldots p$,

$$ Z_{inj} = v_{oj} + \sum x_i v_{ij} $$

$$ Z_j = f(Z_{inj}) $$

Broadcast $z_j$ to the next layer.

Step 6: For each output neuron $Y_k, k = 1, 2, \ldots m$

$$ y_{ink} = w_{ok} + \sum z_j w_{jk} $$

$$ y_k = f(y_{ink}) $$

Step 7: Compute $\delta_k$ for each output neuron, $Y_k$

$$ \delta_k = (t_k - y_k) f'(y_{ink}) $$

$$ \Delta w_{jk} = \alpha \delta_k z_j $$

$$ \Delta v_{ok} = \alpha \delta_k $$ since $z_o = 1$

Step 8: For each hidden neuron,

$$ \delta_{oij} = \sum \delta_k w_{jki} $$

$$ \delta_{ij} = \delta_{oij} f'(Z_{oij}) $$

$$ \Delta v_{ij} = \alpha \delta_{ij} $$

$$ \Delta v_{ij} = \alpha \delta_{ij} $$

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

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

Step 10: 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.

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

Table 1. Comparison between Thinning and proposed method.
References


