Fingerprint Verification Using Direction Images and Local Features

Edward K. Wong*[1], Yao Wang*, Syng-Yup Ohn+
*Polytechnic Institute of NYU, Brooklyn, NY 11201, USA
+ Korea Aerospace University, Goyang-city, Gyeonggi-do. 412-791, South Korea

ABSTRACT

In this paper, we present a novel method for fingerprint verification. A unique characteristic of our method is the use of direction images and local features in the matching process. A direction center is computed from the direction image and used as a reference point for aligning fingerprints. Fingerprint matching is performed in two stages. In the first stage, we compute the correlation between the direction images of the two fingerprints. In the second stage, we compare various features derived from fingerprint minutiae. The first stage acts as a filtering procedure that rejects fingerprints based on the global directional patterns of the ridges. The second stage verifies the local characteristics of the fingerprint minutiae. The two-stage matching process results in a robust procedure that minimizes verification errors.

Keywords: fingerprint verification, direction images, fingerprint minutiae, biometrics, ridge detection

1. INTRODUCTION

Fingerprint verification involves matching an input fingerprint against a reference fingerprint for the purpose of identity authentication. In the past, fingerprint verification is done manually by fingerprint experts. But as computer technology advances, fingerprint verification can be more efficiently done by using computer algorithms. The input fingerprint image could be scanned from an ink fingerprint on paper, or it could be captured live using a fingerprint scanner. Fingerprint verification has wide ranging applications, including controlling access to computer systems and networks, and controlling access to office buildings and restricted facilities. When combined with a PIN (personal identification number), a very high degree of security can be achieved. The advantage of using fingerprint verification over conventional ID cards or bank cards is that it truly verifies the identity of a person, whereas an ID card or a bank card can be stolen and used by someone else other than the card holder. Traditionally, automated fingerprint technology is used only by law enforcement agencies. Their interest is to identify a person (e.g. a criminal) using an automated fingerprint identification system (AFIS). In an AFIS system, the fingerprint of an unknown person is matched against hundreds or thousands of fingerprints in a database. This is different from fingerprint verification, which we seek to authenticate a person through a one-to-one match of the fingerprints. As computers become faster and more affordable, we begin to see the deployment of automated fingerprint technology in everyday use. For example, some brands of notebook computers have a built-in fingerprint scanner used for authenticating a user in the login process. We can buy door locks that use fingerprint verification technology today at affordable price. In a fingerprint verification system, the fingerprints of authorized individuals can be stored in a decentralized manner (e.g. on a smartcard) or in a centralized location (e.g. main database on a computer.) A disadvantage of storing fingerprints in a centralized manner is that privacy and security for all the fingerprints will be compromised if a hacker breaks into the database server. On the other hand, a centralized server offers convenience in the management of a large fingerprint database. In a decentralized system, fingerprint of individuals are stored at different locations or devices. In case of a hacker attack, the privacy and security of only one (or a small number) of the fingerprints will be compromised. In addition, verification can be performed locally in a distributed manner instead of at a central server location. For a more extensive discussion of privacy and

[1] wong@poly.edu

Image Processing: Machine Vision Applications II, edited by Kurt S. Niel, David Fofi, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7251, 725109 · © 2009 SPIE-IS&T · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.806391

security issues in fingerprint systems (and biometric systems in general), the readers are referred to reference [1].

A fingerprint pattern is caused by elevated ridges on the skin of a finger. In-between the ridges are the valleys of the fingerprint. Fingerprints can be categorized into different classes based on the global patterns of their ridges and valleys. Fingerprint categorization is useful for indexing and it facilitates searching in large fingerprint databases. In [2], Galton categorizes fingerprints into the three general classes of *arch, whorl,* and *loop.* In [3], Henry expands the categorization to the eight classes of *whorl, plain arch, right loop, left loop, central pocket, tented arch, twin loop,* and *accidental,* and this has become the most commonly used classification scheme. Further details about fingerprint classification can be found in [4].

In this paper, we present a novel method for fingerprint verification. Our method uses direction images and features derived from fingerprint minutiae for matching. The direction image is a global feature that represents the directions of the fingerprint ridges. We also compute a center point from the direction images for aligning the input and reference fingerprints. We describe the computation of direction images and their center points in Section 2 of this paper. We describe the extraction of fingerprint minutiae and the computation of features based on the extracted minutiae in Section 3. In Section 4, we describe the fingerprint matching process. Section 5 gives experimental results. In Section 6, we present our summary and discussion on the proposed method.

2. COMPUTATION OF DIRECTION IMAGES AND DIRECTION CENTERS

The direction image can be considered as a map that represents the directions of ridges on a fingerprint. To compute the direction image, we first apply the direction filters in [5] to determine a direction at every pixel location on the grayscale image. Pixel locations with no clearly defined directions are marked as background or irregular regions. We then quantize the computed directions into one of eight equally spaced directions. Next we smooth the direction image by using a quadtree-like approach. Starting with the entire image as a square, we count the number of pixels for each direction and identify the first dominant and second dominant directions. If the first dominant direction has pixel count larger than that of the second dominant direction by a certain threshold in term of ratio, we consider the current square has a single dominant direction and assign this direction to every pixel in the square. Otherwise, we divide the square into four smaller quadrants and repeat the above process for every quadrant recursively, until the size of a quadrant reaches a certain minimal size. At the end of this process, we count the total number of quadrants. If the number is above a certain threshold, we consider the image to be too noisy for further processing and another fingerprint scan will be requested. Figure 2(a) shows a grayscale fingerprint image and Figure 2(b) shows its computed direction image.

We then compute the direction center from the direction image. The direction center represents a point on the direction image where different directions meet. It will be used as a reference point to align the input and reference fingerprints during the matching process. The direction centers for the reference fingerprint and the input fingerprint are determined in a slightly different manner. We first describe how to determine the direction center for a reference fingerprint. Given the smoothed direction image, we identify the center location by using a quadtree-like procedure to recursively identifies quadrants with three or more salient directions. A salient direction is one that has a pixel count larger than a certain percentage of the size of the quadrant. At the end of this process, a set of quadrants with three or more salient directions and have a certain minimal size (3x3) will be identified. Next, we will choose one of the quadrants to contain the direction center. For a reference fingerprint, the quadrant to contain the direction center is chosen manually during the enrollment process. By inspecting the direction image and the grayscale fingerprint image, a human operator can choose the quadrant that is closest to the point where different directions meet. The center of the chosen quadrant is used as the direction center. The manual selection process is done only once for a reference fingerprint, and the computed direction center can be stored for later use. The direction center for an input fingerprint is determined in a similar but slightly different manner. We also use a quadtree-like procedure to determine a set of quadrants with three or more salient directions and have a

certain minimal (3x3) size. Then for each of such quadrants, we compare the smoothed direction pattern over a 25x25 square surrounding this quadrant with the direction pattern over a 25x25 square surrounding the direction center of the reference fingerprint to be matched to. The center of the quadrant that has the best matching direction pattern is chosen to be the center of the direction image.

3. LOCAL FEATURE EXTRACTION

This section describes our algorithm for extracting fingerprint minutiae and the computation of local features from the extracted minutiae. The grayscale fingerprint image is first enhanced by using the direction information contained in the smoothed direction image. The enhanced image is then thresholded into a binary image. We then perform binary smoothing to smooth the contours and fill in small holes on fingerprint ridges. Next we apply Deutch's thinning algorithm [6] to obtain a thinned image of the fingerprint. This thinning algorithm has the desirable characteristic that it eliminates isolated 1 pixels in the background. After thinning, we obtain a fingerprint image with ridges that are one-pixel thick. A pruning algorithm is then used to remove short parasite branches that come off the ridges. Figure 3(a) shows the smoothed binary image we obtained for the grayscale image in Figure 2(a), Figure 3(b) shows the thinned image before pruning, and Figure 3(c) shows the thinned image after pruning. Next we detect fingerprint minutiae. Our current algorithm detects the two basic types of minutiae: *ridge endings* and *ridge bifurcations*. A ridge ending is where a ridge terminates and a ridge bifurcation is where a ridge divides into two ridges. Figure 4 shows an example of a ridge ending and a ridge bifurcation. They are considered to be the two most important minutia features for matching fingerprints. The algorithm for detecting ridge endings and bifurcations from the thinned image is as follows:

1. Scan the entire thinned image pixel by pixel using a 3 x 3 window as shown in Figure 5
2. If the center pixel P_c is a background pixel, skip to the next pixel location
3. Else if the center pixel is a foreground (ridge) pixel then examine its 8-connected neighbors
    a. If only one of the 8-connected neighbors is a foreground pixel, then the center pixel is a ridge ending
    b. Visit the 8-connected neighbors in the order P_1 to P_8, counter clockwise, and count the number of background-to-foreground (0-to-1) transitions. If the number of transitions is greater than or equal to three, then the center pixel is a bifurcation.

For each minutia point detected, we compute the following features:

- Local orientation. For a ridge ending, this is the orientation of the ridge near the end point; for a bifurcation, this is the orientation of the ridge just right before it divides into two.
- Angle of the line segment between the direction center and the minutia point.
- Distance between the direction center and the minutia point.
- Number of ridges crossed by the line segment between the direction center and the minutia point.

Let us denote the above features as θ, α, *Radius*, and *RC,* respectively, and the minutia type (ridge ending or bifurcation) as *Type*. Note that the features *Radius* and α correspond to the polar coordinates of the minutia point when the direction center is used as the origin.

4. FINGERPRINT MATCHING

In the matching process, the input fingerprint and the reference fingerprint are aligned at their direction centers and we compute their similarity in two stages. In the first stage, we compute the correlation between the direction images of the two fingerprints. The computed correlation measures the overall fit between the general directions of the ridges of the two fingerprints. If the correlation is below a certain

threshold, the input fingerprint is rejected and we do not proceed to the second stage. Matching direction images is less sensitive to noise and minor distortions than matching local fingerprint features. In the second stage, we align the two thinned fingerprint images at their direction centers and identify the minutia points that lie inside the overlapping window. A minimum number of minutia points must be present from each image within the overlapping window; otherwise, we simply reject the input fingerprint. We then compare the minutiae using the features we described in the previous section. The minutia matching process is described in the following algorithm.

Minutia Matching Algorithm

1. For each pair of minutiae *i* and *j*, one from the input fingerprint and one from the reference fingerprint, we compute their distance *D* as a weighted sum of the differences between their types and their feature values; that is

$$D(i, j) = w_1 Type + w_2 \theta + w_3 \alpha + w_4 Radius + w_5 RC$$

   where the *w's* are the weight values. The weight w_1 for minutia type is assigned a high value since for two minutiae to match they should have the same type. But sometimes due to noise and processing errors, a bifurcation may become a ridge ending and vice versa.
2. Sort all pairs (*i, j*) in increasing distances and put them into a list.
3. Select the top pair (*k, l*) that has the smallest distance and mark it as a matching pair.
4. Remove all subsequent pairs in the list that have minutiae *k* or *l* in them.
5. Select the next pair (*k, l*) that has the next smallest distance and go to step 4. When all pairs are marked as matching pairs, continue to step 6.
6. Compute an affine mapping (or transformation) from the image coordinates of the matching pairs through least square fitting. This affine mapping will map minutiae from the input fingerprint into the reference fingerprint.
7. Remove pairs with poor fit between the fitted positions and the actual positions. A pair has poor fit if the distance between the fitted position and the actual position exceeds a certain threshold.

The purpose of steps 6 and 7 is to identify and eliminate false matching pairs through the determination of a global geometric mapping between the matching minutiae of the input and reference fingerprints. Using the mapping so find, we project the minutiae in the input image onto the reference image based on their coordinates in the input image. If the distance between the actual position of a minutia in the reference image and the projected position is larger than a preset threshold, the matching pair will be considered to be a false match and it will be deleted from the list of matching pairs. The mapping used in our implementation is the affine mapping, which covers most types of deformations between two scans of the same fingerprint, including rotation, scaling, and translation. When there are many false matching pairs in step 7 above, the mapping determined based on the initial set of all matching pairs may not reflect the true mapping. This problem is overcome by running the process twice. After removing those pairs which do not fit with the global mapping (step 7 above,) we use the remaining pairs to determine the mapping again, and delete those (among all the initial matching pairs) which do not fit with the mapping. Our test results have shown that this method is very effective in eliminating false matches. After running the minutia matching algorithm, if the final number of matching minutiae exceeds some threshold, we accept the input fingerprint; otherwise we reject the input fingerprint. Figure 6 shows the thinned images of a pair of matching fingerprints. On the left is the reference fingerprint and on the right is the input fingerprint. The direction center in both images is indicated by a square. Matching pairs of minutiae are indicated by matching numbers from the left and right images.

5. EXPERIMENTAL RESULTS

We performed experiments to evaluate the performance of our verification algorithm. We collected a set of 84 fingerprints using an external fingerprint scanner attached to a PC. Each image is cut into a size of 200 x 200 pixels to retain the center portion that contains the fingerprint. The set consists of nine fingerprints of type *arch*, 45 fingerprints of type *loop*, and 30 fingerprints of type *whorl*. From the 84 fingerprints, we selected 101 matching pairs and 100 non-matching pairs and use them as test data. Excellent performance was obtained when we apply our algorithm to the test data set. We obtained a false-rejection rate of 3.7% and a false-acceptance rate of 0% . For a more extensive evaluation of our algorithm, we plan to test our algorithm on a larger set of fingerprints, such as the NIST fingerprint data set [7].

6. SUMMARY AND DISCUSSION

We have presented a novel method for fingerprint verification. The method uses direction images and features derived from fingerprint minutiae in the matching process. We compute a direction center for aligning fingerprints and matching is done in two stages. In the first stage, we compute the correlation between the direction images of the input fingerprint and the reference fingerprint. This acts as a filtering procedure that rejects fingerprints based on global directions of the ridge patterns. This stage is less sensitive to local noise and errors introduced during the image acquisition process. The second stage of the matching process verifies the local characteristics of minutia points. The two-stage matching process results in a robust procedure that minimizes verification errors. Experimental result demonstrates the effectiveness of our method.

REFERENCES

[1] Ratha, N. K., Connell, J. H., and Bolle, R. M., "Enhancing security and privacy in biometrics-based authentication systems," IBM Sys. J. 49(3):614-634.

[2] Galton, F., Finger prints, McMillian, London (1892).

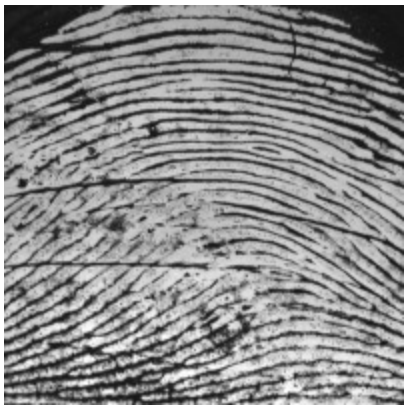
[3] Henry, E., Classification and uses of finger prints, Routledge, London (1900).

[4] Yager, N., and Amin, A., "Fingerprint classification: a review," Pattern Anal. Applic. 7:77-93 (2004).

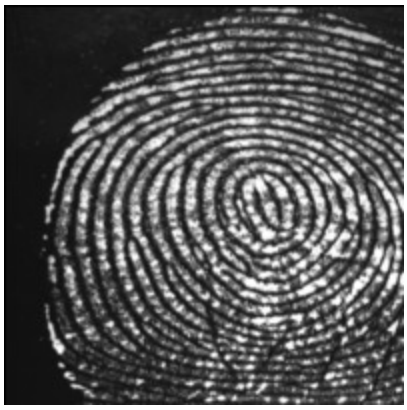
[5] Danielsson, P. E. and Ye, Q. Z., "Rotation-Invariant Operators Applied to Enhancement of Fingerprints," Proc. IEEE 8th International Conference on Pattern Recognition, Rome, pp. 329-333 (1988).

[6] Deutch, E. S., "Thinning algorithms on rectangular, hexagonal, and triangular arrays," Communications of the ACM, 15, 827-837 (1972).

[7] US National Institute of Standards and Technology (NIST). Scientific and Technical Databases. http://www.nist.gov/data/. Cited Jan 2009.



(a) Arch

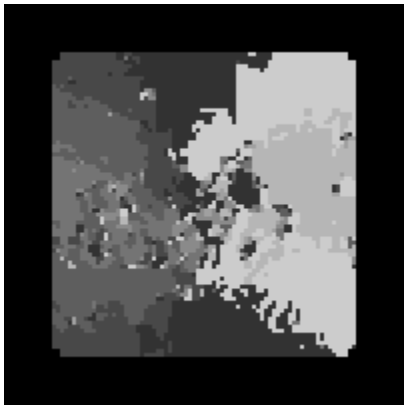


(b) Whorl

(c) Loop

Fig. 1. Examples of the arch, whorl, and loop fingerprint classes.

(a) Grayscale image



(b) Direction Image

Fig. 2. An input fingerprint and its direction image

(a) Smoothed binary image

(b) Thinned binary image before pruning

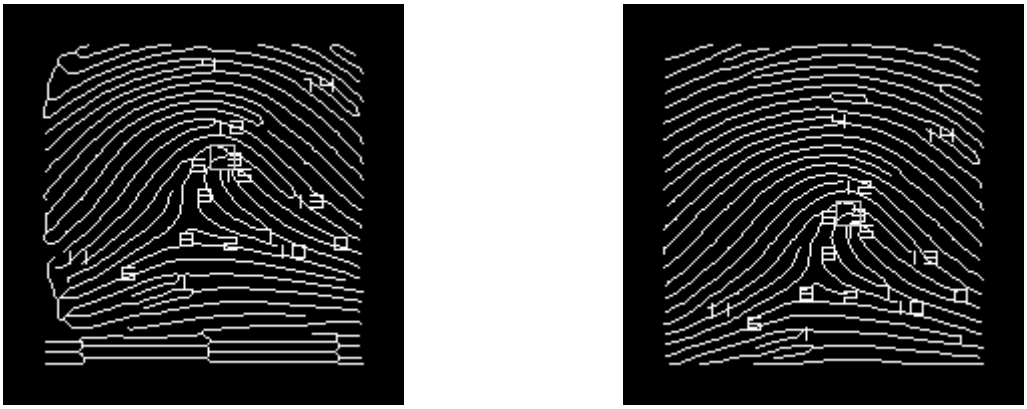
(c) Thinned binary image after pruning

Fig 3. The smoothed binary image and thinned binary image for the fingerprint in Fig. 2(a)

Fig 4. Example of a ridge ending and a bifurcation.

P_4	P_3	P_2
P_5	P_c	P_1
P_6	P_7	P_8

Fig. 5. The 8-connected neighbors of a center pixel in a 3x3 window.



(a) Reference fingerprint (b) Input fingerprint

Fig. 6. A pair of matching fingerprints.