Fingerprint Classification Using Singularities Detection

Liu Wei

***Abstract*—**Applying the Automatic Fingerprint Classification technology, the search of a fingerprint can be restricted to a limited scope while performing fingerprint identification, so that the retrieval effectiveness of the whole system can be improved greatly. Singularities are the most important and reliable features in classification. This paper describes an improved rapid singularity searching algorithm which employees delta field Poincare index and a rapid classification algorithm to classify the fingerprints into 5 classes. The detection algorithm only searches the direction field which has the larger direction changes to get the singularities. Then a singularities detection post-processing method is used to increase the accuracy. The classification algorithm uses the delta direction and the singularities to partition the similar classes. The algorithms were tested on NIST-4 database and got a good performance.

***Keywords*—**fingerprint, classification, singularities, Poincare, direction field

I. INTRODUCTION

Among all the biometric indicators, fingerprint has one of the highest levels of reliability and has been extensively used. In an automatic fingerprint identification system (AFIS), the goal is to find a match for a probe fingerprint in the database of enrolled prints, possibly numbering millions. Classification is used in an AFIS to reduce the size of the search space to fingerprints of the same class before attempting exact matching. It is very important to detect singularities (core and delta) accurately and reliably for classification and matching of fingerprints. See Figure 1.

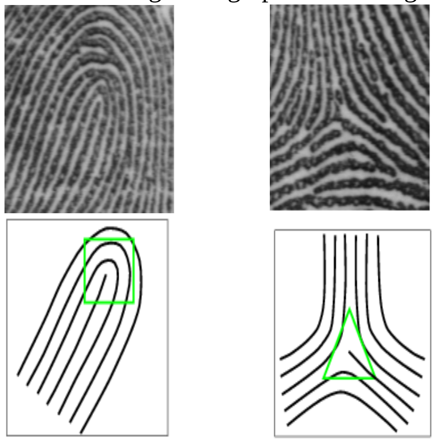


Figure 1 Sample of core and delta

A number of approaches have been developed for singularities detection in fingerprint classification. [1][2] select high curvature blocks as searching fields. In [3], minimum variance is employed to detect singularities. Nero network computing on direction field is introduced in [5]. All these approach get good performances, but these algorithms can't justify whether the singularities exists in some position directly, and the complex post processing algorithms need to be used to get the final results.

The approach represented in this paper is based on Poincare index which can search the singularities speedy and directly. In the direction field, counterclockwise rounding a singular point along a close curve will influence the following angle changes. The angle change will be 180° while the center of the curve is core, -180° for delta and 0° for other points. We use this feature to search and detect the singularities.

The paper will be arranged as follows. Section 2 is about the Poincare index. Section 3 introduced the singularities detection. Section 4 is about some postprocessing. Section 5 describe the classification algorithm using the singularities. Section 6 gives the experimental results and the performance. We draw conclusion in section 7.

II. POINCARE INDEX

Poincare index is defined as:

$$Poincare(x,y)=\frac{1}{2\pi}\lim_{\varepsilon\to 0}\left\{\int_0^{2\pi}\frac{\partial}{\partial}\nu(x+\varepsilon\cos\theta, y+\varepsilon\sin\theta)d\theta\right\} \quad (1)$$

o' respresents the direction field, and the Poincare index of point (i, j) is defined as:

$$Poincare(i,j)=\frac{1}{2\pi}\lim_{\varepsilon\to 0}\left\{\int_0^{2\pi}\frac{\partial}{\partial}o'(i+\varepsilon\cos\theta, j+\varepsilon\sin\theta)d\theta\right\} \quad (2)$$

with

$$\frac{\partial}{\partial}o'(i+\varepsilon\cos\theta, j+\sin\theta)=\begin{cases} d\delta, if|d\delta|<\pi/2 \\ \pi+d\delta, ifd\delta\le-\pi/2 \\ \pi-d\delta, otherwise \end{cases}$$

Manuscript received February 5, 2008: Revised version received May 28, 2008.

Liu Wei is with School of Computer Science, Hubei University of Technology, Wuhan, Hubei 430068, China (phone: 86-27-88026670; fax: 86-27-88026670; e-mail: liuwei001@vip.sina.com).

(3)

$$d\delta = \lim_{v \to 0} \frac{\delta'(i+\varepsilon\cos(\theta+v), j+\varepsilon\sin(\theta+v)) - \delta'(i+\varepsilon\cos\theta, j+\varepsilon\sin\theta)}{v}$$

(4)

Define $\psi_x(\cdot)$, $\psi_y(\cdot)$ as a close curve with ψ pixels in an image, the Poincare index will be represented as:

$$Poincare\ (i,\ j) = \frac{1}{2\pi} \sum_{k=0}^{\psi} \Delta(k) \quad (5)$$

with

$$\Delta(k) = \begin{cases} d\delta(k), if\, |d\delta(k)| < \pi/2 \\ \pi + d\delta(k), if d\delta(k) \le -\pi/2 \\ \pi - d\delta(k), otherwise \end{cases} \quad (6)$$

$$d\delta(k) = \delta'(\psi_x((i+1) MOD\,\psi), \psi_y((i+1) MOD\,\psi)) - \delta'(\psi_x(i), \psi_y(i)) \quad (7)$$

III. Singularities Detection

The singularities detection method will refer the following flow.

1) compute the local direction of blocks with different size.

Typically, only 8×8 and 16×16 blocks will be used. And the local direction will be θ_1 and θ_2.

2) acquire the candidate field

As the angle of candidate field exists singluarities will change biggish, so the difference of θ_1 and θ_2 will be higher. Use this feature, we can define:

$$d = |\theta_1 - \theta_2| \quad (8)$$

And a threshold is deined as T_θ. We will only search the blocks while $d > T_0$. This will increase the detection speed.

3) compute the Poincare index.

d_0	d_7	d_6
d_1	(i, j)	d_5
d_2	d_3	d_4

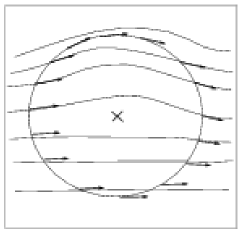
Figure 2. Poincare index chart

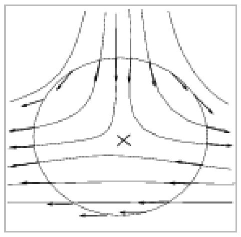
See figure 2, in close curve $d_0 d_1 d_2 d_3 d_4 d_5 d_6 d_7 d_0$, the Poincare index equals:

$$Poincare(i, j) = \frac{1}{2\pi} \sum_{k=0}^{7} \left| d_k - d_{(k+1)\bmod 8} \right| \quad (9)$$

4) singularities detection

In section 1, we use the feature that the angle changes is 180°, -180° and 0° for core, delta and others,. See figure 3. We will use this to detect the singularities.


normal points


delta point

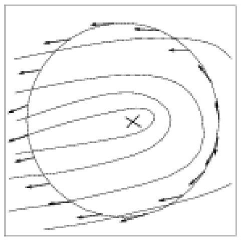

core point

Figure 3 angle changes and point

We can get the relationship conclusions below.

The relationship between angle changes and Poincare Index:

If the angle changes equals 180°, the Poincare index is $1/2$;

If the angle changes equals -180°, the Poincare index is $-1/2$;

If the angle changes equals 0°, the Poincare index is 0;

So the points and their Poincare index is concluded as followed.

Core: the Poincare index is $1/2$;

Delta: the Poincare index is $-1/2$;

Others: the Poincare index is 0.

Scanning the candidate blocks, we will get the singularities' positions and numbers after computing their Poincare index.

IV. Postprocessing of Detection

The postprocessing step will eliminate the false singularities. The processing includes 3 steps: probability analysis of singularities, Pocincare index computation with different circle curves, Neighborhood scan of the singularities.

4.1 Probability analysis

The core is more adjacent to the center of the fingerprint, so the probability function $P_c(i, j)$ of the core is:

$$P_c(i, j) = \frac{\exp[-dis[(i, j), (i_c, j_c)] / 2\sigma^2]}{\sqrt{2\pi\sigma}} \quad (10)$$

But the $P_c(i,j)$ can't work properly for delta point, for the delta is not adjacent to the center and the quality of the fingerprint and the background is always low. So we employee the coherence parameter to compute the $P_c(i,j)$ of delta.

$$P_d(i,j)=C_{oh}(i,j)\cdot\frac{\exp[-dis((i,j),(i_c,j_c))/2\sigma^2]}{\sqrt{2\pi}\sigma} \tag{11}$$

4.2 Circle curves

In our experiment, we always choose radius 3 and 5 to confirm there's no other singularities in the close curve. Figure 4 is the sketch map of close curve.

D_0	D_{15}	D_{14}	D_{13}	D_{12}
D_1	d_0	d_7	d_6	D_{11}
D_2	d_1	(i,j)	d_5	D_{10}
D_3	d_2	d_3	d_4	D_9
D_4	D_5	D_6	D_7	D_8

Figure 4 sketch map of close curve radius 3 and 5

And the Poincare index of close curve $d_0d_1d_2d_3d_4d_5d_6d_7d_0$ radius 3 is:

$$Poincare_1(i,j)=\frac{1}{2\pi}\sum_{k=0}^{7}\left|d_k-d_{(k+1)\bmod 8}\right| \tag{12}$$

Similarly, the Poincare index of close curve $D_0D_1D_2\ldots D_{15}D_0$ radius 5 is:

$$Poincare_2(i,j)=\frac{1}{2\pi}\sum_{k=0}^{15}\left|D_k-D_{(k+1)\bmod 16}\right| \tag{13}$$

Only when $Poincare_1(i,j)=Poincare_2(i,j)$, the singluar point detected is effective, otherwise, the point candidate will be deleted.

4.3 Neighborhood scanning

The algorithm is:

① For every candidate singularities, save their coordinates and types, the type refers to core or delta.

② Scanning along x-axis of every candidates. If there is any other candidate, go to step ③, else scanning the next candidate.

③ If the types of the candidates in one 8-neighborhood are same, only the last one will be saved and the others will be marked false. If the types are not same, all the candidates will be marked false.

④ While ending, eliminate all the false candidates and re-scan the image.

In the postprocessing step, the different circle curves will work first, and the neighborhood scanning follow. Using these two methods, the noise will be limited to a low level. The detection samples will be shown in section 7.

V. Classification algorithm

The singularity models of 6 fingerprint class are shown in Fig. 5. In brief, the arch has no singularity, the tented arch and the loop have one core and one delta, the whorl and the double loop have two cores and two deltas. We employ these to identify the different classes.

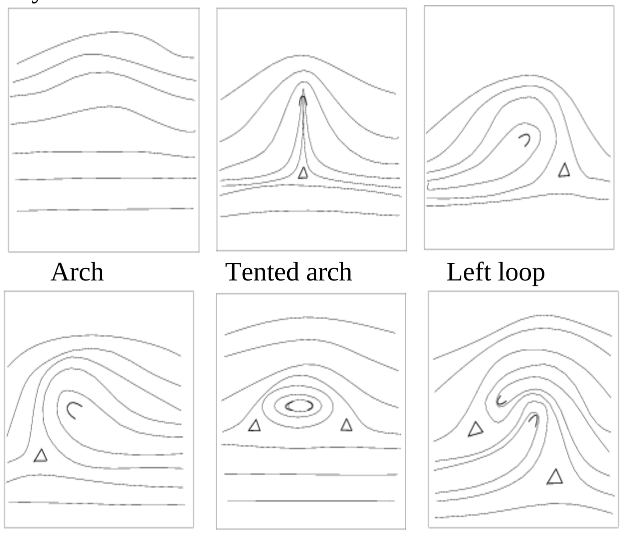
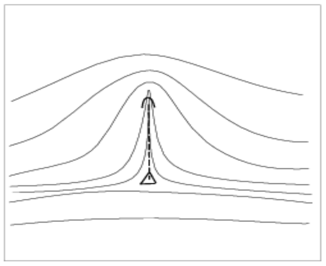


Figure 5 Singularity models in 6 classes

① Partition of arch and tented arch

The images of arch class or tented arch class all have 2 singularities: one core and one delta. We use position difference to separate them. See Fig. 6, the angle of line between core and delta in arch class is consistent with the local direction, and the angle in tented class makes a large separation angle with the delta field. We use this to separate these two class.

β is the slope of the line L between core and the delta, $\alpha_1,\alpha_2,\ldots,a_n$ represent the local direction of the points in ridgeline and L. If $\frac{1}{n}\sum_{i=1}^{n}\sin(\alpha_i-\beta)$ is less than a given threshold (always 0.2 in experiments), the image belongs to the arch class, else the image is in the tented arch class.



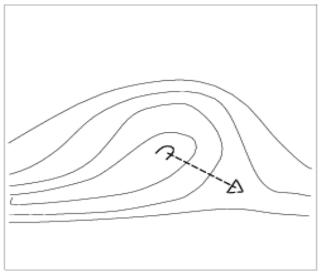


Figure 6 arch and tented arch

② Partition of whorl and double loop

The images of arch class or tented arch class all have 4 singularities: two core points and two delta points, See Fig. 7. Here we use the similar method to separate these two classes. The only difference of method ① and ② is that the line L in method ② is a line between two cores.

β is the slope of the line L between the cores, $\alpha_1, \alpha_2, ..., a_n$ represent the local direction of the points in ridgeline and L. If $\frac{1}{n}\sum_{i=1}^{n}\sin(\alpha_i - \beta)$ is less than a given threshold, the image belongs to the whorl, else the image is in the double loop class.

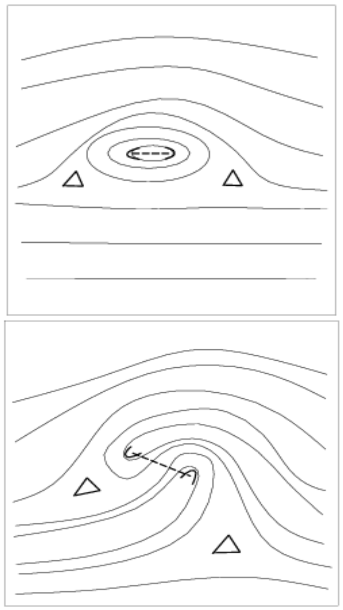


Figure 7 arch and tented arch

③ Partition of left loop and right loop

The algorithm is: moving along the local direction from the core, if the delta stays left, the image belongs to the left loop class, else it's in the right loop class. More accurately, points B, C, D represent the boundary point, the core and the delta. The algorithm is: moving along the local direction from point C towards B. See Fig. 8.

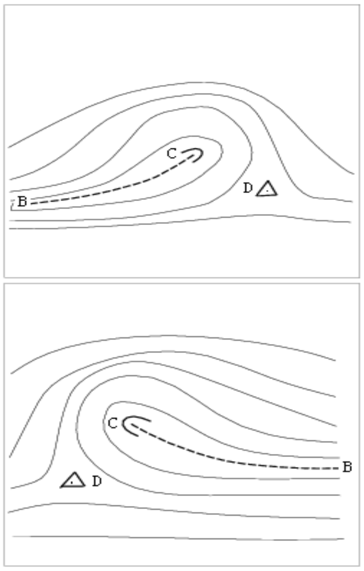


Figure 8 left loop and right loop

Here we use a ridgeline tracing method to get point B.

The starting point (x_0, y_0) is the core (C_x, C_y).

The next point (x_1, y_1) is computed by this:

$$\begin{cases} x_1 = x_0 - BlockSize \times \cos\theta_0 \\ y_1 = y_0 - BlockSize \times \sin\theta_0 \end{cases}$$

, θ_0 is the local direction of the starting point.

$$\begin{cases} x_n = x_{n-1} - BlockSize \times \cos\theta_{n-1} \\ y_n = y_{n-1} - BlockSize \times \sin\theta_{n-1} \end{cases} \quad (12)$$

Finally, the last point in the border block is the point B.

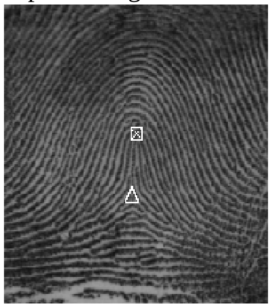
While we get the point B, we compute the following formula.

$$T = (B_x - C_x)(D_y - C_y) - (B_y - C_y)(D_x - C_x) \quad (14)$$

If $T > 0$, the image belongs to the left loop class, else the image belongs to the right loop class.

VI. Experiments

Figure 9 is a sample of singularities detection in NIST4.



arch

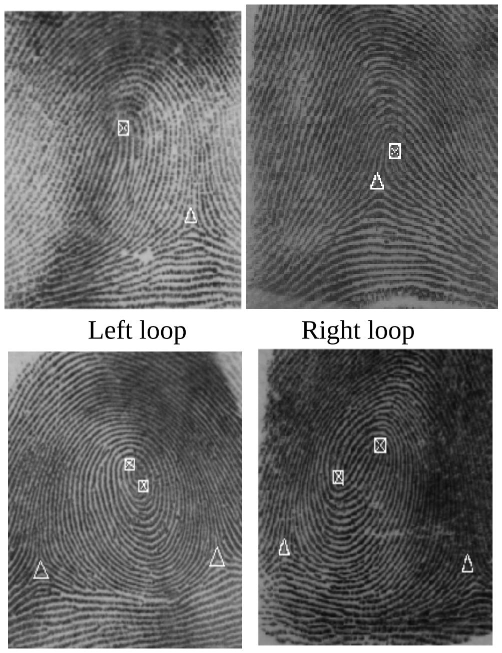


Figure 9 A detection sample

In figure 9, rectangle represents the core, and triangle stands for delta. From this sample, the singularites are accurate.

Table 1 shows the performance of classification on NIST4, the performance based on the singular searching algorithm in this paper and a ridgeline classification algorithm.

Table 1 Classification Performance

	Real classes				
	arch	tented arch	left loop	right loop	whorl
arch	921	17	6	9	0
tented arch	5	563	21	16	3
left loop	5	24	828	7	18
right loop	4	8	7	752	14
whorl	0	2	3	6	761

Table2 classification performance

	arch	tented arch	left loop	right loop	whorl
Perf. %	98.5	91.7	95.7	95.2	95.6
Avg.	95.6%				

From table1 and table 2, we can see this classification method works well especially on arch class and the classification accuracy is comfortable.

VII. Conclusion

In this article, we introduce a improved singularities detection algorithm, which can searching the singularities more accurately and rapidly constration with other algorithms in section 1. The experiments performance on NIST4 is satisfied too. But the relationship formula in section 3 is too simple, and this will partly influence the accurcy. The further work will improve this.

References

[1] O.Nakamura, K.Goto, T.Minami. "Fingerprint Classification by Directional Distribution", Patterns, Systems, Computers, Controls, vol.13. no.5,pp.81-89,1982.

[2] V.S.Srinivasan and N.N.Murthy. "Detection of Singular Points in Fingerprint Images", Pattern Recognition,vol.25,no.2,pp.139-153,1992

[3] A.R.Rao and R.C.Jain, "Computerized Flow Field Analysis Oriented Texture Fields", IEEE Trans. Pattern Analysis and Machine Intelli gence, vol.14,no.7, pp.693-709,July 1992.

[4] P.Perona, "Orientation Diffusions", IEEE Trans. Image Processing,vol .7, no.3, pp.457-467, Mar. 1998.

[5] G.A.Drets and H.G.Liljenstrom, "Fingerprint Sub classification: A Neural Network Approach, Intelligent Biometric Techniques in Fingerprint and Face Recognition", L.C.Jain, U. Halici, I. Hayashi,S.B.Lee, and STsutsui, eds., pp.109-134, Boca Raton, Fla.:CRC Press, 1999.

[6] A.K. Jain, S. Prabhakar, L. Hong,and S. Pankanti, "Filterbank-Based Fingerprint Matching", IEEE Trans. Image Processing, vol.9,no.5, pp. 846-859, May 2000.

[7] K. Karu and A. K. Jain. "Fingerprint classification". Pattern Recognition, 29(3), pp. 389–404, 2006.

[8] S. Minut and A. K. Jain. "Hierarchical kernel fitting for fingerprint classification and alignment". Proc. of International Conference on Pattern Recognition, Quebec City, August 11-15, 2002.

[9] Sarat C. Dass, Anil K. Jain. "Fingerprint Classification Using Orientation Field Flow Curves". Proceedings of ICVGIP, pp.650-655. 2004.

[10] Arun Ross, Sarat C. Dass, Anil K. Jain. "Fingerprint Warping Using Ridge Curve Correspondences". IEEE Trans. Pattern Anal. Mach. Intell. 28(1), pp. 19-30, 2006.

Liu Wei : was born in XiangFan, Hubei, China in Feb 26, 1977. He received the Ph. D degree in communication and information system from Wuhan University, hubei, China in 2007.

Currently he is a docent in School of Computer Science, Hubei University of Technology, Wuhan, Hubei, China. His current research interests include image analysis and pattern recognition.