2022

Core Point Pixel-Level Localization by Fingerprint Features in Spatial Domain

Xueyi Yi

Yuzhong Shen
Old Dominion University, yshen@odu.edu

Maosheng Zeng

Yirui Liu

Huahua Chen

See next page for additional authors

Original Publication Citation

Ye, X., Shen, Y., Zeng, M., Liu, Y., Chen, H., & Zhao, Z. (2022). Core point pixel-level localization by fingerprint features in spatial domain. *Mathematical Biosciences and Engineering*, *19*(1), 707-737. https://doi.org/10.3934/mbe.2022032

Mathematical Biosciences and Engineering

MBE, 19(1): 707–737.
DOI: 10.3934/mbe.2022032
Received: 21 September 2021
Accepted: 09 November 2021
Published: 19 November 2021

http://www.aimspress.com/journal/MBE

Research article

Core point pixel-level localization by fingerprint features in spatial domain

Xueyi Ye[1], Yuzhong Shen[2], Maosheng Zeng[1,*], Yirui Liu[1], Huahua Chen[1] and Zhijing Zhao[1]

[1] Lab of Pattern Recognition and Information Security, Hangzhou Dianzi University, Hangzhou 310018, China
[2] Department of Modeling, Simulation and Visualization Engineering, Old Dominion University, Commonwealth of Virginia VA 23529, USA

* **Correspondence:** Email: zms_0310@163.com.

Abstract: Singular point detection is a primary step in fingerprint recognition, especially for fingerprint alignment and classification. But in present there are still some problems and challenges such as more false-positive singular points or inaccurate reference point localization. This paper proposes an accurate core point localization method based on spatial domain features of fingerprint images from a completely different viewpoint to improve the fingerprint core point displacement problem of singular point detection. The method first defines new fingerprint features, called furcation and confluence, to represent specific ridge/valley distribution in a core point area, and uses them to extract the innermost Curve of ridges. The summit of this Curve is regarded as the localization result. Furthermore, an approach for removing false Furcation and Confluence based on their correlations is developed to enhance the method robustness. Experimental results show that the proposed method achieves satisfactory core localization accuracy in a large number of samples.

Keywords: accurate core point location; furcation; confluence; image spatial domain

1. Introduction

Singular point (SP) is an important global feature of fingerprints. By Henry's definition [1], it is classified as Core, the topmost point of the innermost curving ridge, and Delta, the center of a triangular region where three different ridges meet.

In modern research of automatic fingerprint identification system (AFIS), there are two main

applications of SP usually. SPs are firstly used to index fingerprint types [2,3] by narrowing down the search space when it needs to match fingerprint samples in a large-scale database. In this case, the SP type is the only concern. Secondly SPs are also used to align a registered fingerprint sample with the input one to decrease the computational cost [4,5] and here SP locations are much more important than the first application, even decide the matching performance. Whether a SP location is accurate or not at pixel-level will directly result in fingerprint's alignment perfect or not. To some big extent, it decides a difficult problem which the image rotation or translation is negative to fingerprint recognition to be resolved perfectly or not. Because Deltas are often located on the lower parts of fingerprints, they are not guaranteed to appear in the corresponding fingerprint image even if they indeed exist in a fingerprint. Therefore, accurate determination of core locations at pixel-level is especially important to fingerprint alignment and a big challenge to SP detection.

Among existing methods of SP detection, Poincare Index (PI), introduced by Kawagoe and Tojo [6], is a classical one, and has been widely used because of the advantages of its simple design, the robustness against image rotation, and the ability to distinguish SP types. For instance, Fan et al. [7] and Jin and Kim [8] proposed improved methods that built upon PI, respectively. However, PI-based methods are sensitive to noise. A decrease in fingerprint image quality will degrade the detection rate significantly. To deal with this problem, some researchers attempted to combine PI with other methods [9]. Meanwhile, non-PI-based methods were proposed such as orientation curvature based approach [7,10].

Although SP detection has been extensively researched, some problems and challenges still exist. For example, there still exist big discrepancies between automatically detected SPs and those manually identified by Henry's definition. Mostly existing methods of SP detection are based on block-level orientation field [4,5,7,9,10], and consequently the center of a block is usually regarded as a SP's location. Bazen and Gerez [11] combined Principal Components Analysis(PCA) and PI to compute high resolution Orientation Field (OF) and further detect SPs at pixel-level, but there are often more false-positive SPs in detection results because of linear filters used for computing high resolution OF. Jin and Kim [8] improved the method and decreased the number of false-positive SPs, using multi-scale Gaussian filters and Nested-PI to estimate OF and detect SPs respectively. However, but for high-quality fingerprint images, the detected SP's locations are inconsistent with Henry's definition because the high-curvature area of fingerprint is degraded by Gaussian filters. It is known that SPs are always located in the high-curvature area.

In fact, there are also some methods that use singularity or pseudo-singularity points for classification/authentication. C. Militello et al. [12] proposed a fingerprint recognition approach based on core and delta SPs detection. V. Conti et al. [13] proposed a fingerprint recognition approach based on SPs detection and singularity regions analysis. Both of their proposed systems are based on core and delta position, their relative distance and orientation to perform both classification and matching tasks. And the approach proposed by V. Conti et al. [13] enhances the performance of SPs based methods introducing pseudo-SPs when the standard SPs can not be extracted.

In addition, M. Sabir et al. [14] proposed an approach to perform alignment of fingerprints followed by their matching in fewer computations. Ridges are not included in the matching process to avoid redundant computations, which is where the approach proposed by M. Sabir et al. [14] differs from the traditional cross-correlation based matching algorithms. And the real-time image filtering technique that allows for faster implementation of fingerprint image enhancement proposed by T. M. Khan et al. [15] circumventing the hurdle of expensive hardware implementation of fingerprint filtering techniques.

To improve SP detection and provide more accurate reference points for fingerprint alignment, this paper proposes a novel core point pixel-level localization method only based on fingerprint features of spatial domain. The proposed method does not consider OF and frequency-domain (including filters) of fingerprint. The specific ridge/valley distribution in a core point area is used firstly, which is called Furcation and Confluence characteristics, to extract the innermost Curve of ridges. Then the summit of this Curve is regarded as the location of a core. Furthermore, a Furcation and Confluence correlation based schedule to remove false Furcation and Confluence is designed to enhance the method's robustness against noise. Experimental results indicate that the proposed method achieved better core localization accuracy for 40.8% of the samples, similar accuracy for 55% the samples, and less accuracy for 4.2% of the samples compared with the method of Jin and Kim [8] on the database of FVC2000-DB2.

The remainder of this paper is organized as follows. Section 2 describes how to extract Furcation and Influence from fingerprint images. Section 3 introduces the anti-noise measures to remove false Furcation and Confluence from a candidate set based on the correlation of Furcation and Confluence. Section 4 describes the final step to extract the innermost curve of ridges and locate accurately a core point. Section 5 presents experimental results and analyses. Finally, Section 6 draws conclusion.

2. Furcation and Confluence extraction

Both Furcation and Confluence (FC) are spatial-domain features of fingerprint ridges/valleys. They are directly and natured ridges/valleys presentation of fingerprint images and different with representation of extracted features which are transformed by filters or operators. It is the reason why FC is used in here.

2.1. Definition of judge lines, Furcation and Confluence

The extraction of Furcation or Confluence is based on the distribution of ridges and valleys. In other words, the pixel values will not be directly taken into account. Therefore, binary images are used for FC extraction.

Figure 1 shows the binary image of an input fingerprint with a resolution of 500 dpi, where the binary image is obtained by the short time Fourier transform(STFT) [16]. Let C_n be the nth core that is detected by a specific algorithm [8] (as shown in Figure 1) and θ_n be the opening direction of C_n calculated by the local OF around it [17] (as shown the purple line in Figure 1, where the direction indicated by the arrow is the opening direction). Then, according to C_n and θ_n, a single-pixel-wide line l with length of 60 pixels can be drawn (as shown the blue line in Figure 1). Treating each pixel of l as the midpoint, 60 line segments can be constructed that are perpendicular to l with length of 41 pixels. These lines, shown the red lines in Figure 1, are called Judge Lines (JLs). Note here two number parameters (one is 60 which defines the total number of judge line segments; another is 41 which defines the length of each judge line segment) are determined by the resolution

of the input fingerprint image. Basically, the higher the resolution is, the larger the parameters will be. Specifically, take Figure 1 as an example, the width of ridges and valleys is roughly 5 pixels at a resolution of 500 dpi of the input fingerprint image, and the four ridges or valleys are approximately 20 pixels wide, so the line segment l is 20 pixels on each side and the length of JL is 41 pixels. Similarly, since the ridges and valleys are interconnected, the length of l is 60 pixels in order to allow line segment l to pass through up to 6 pairs of ridges and valleys.

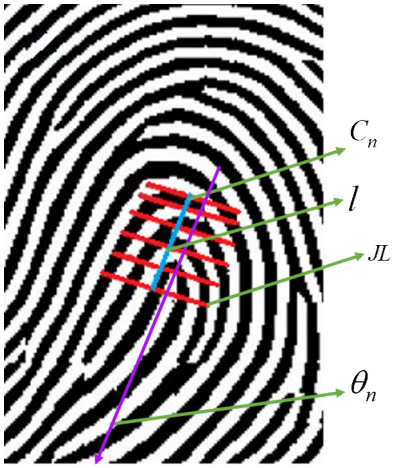


Figure 1. The formation of judge lines.

Figure 2 shows the local region of a binary fingerprint image, in which the red lines are JLs. Consider the number of fingerprint lines: including ridges and valleys, intersected by these judge lines. If the lower JL intersects more fingerprint lines than does the adjacent upper one, the corresponding position is called a Furcation. And according to the exact difference of intersections between these two JLs, a Furcation is further divided into four categories: two-Furcation (as shown in Figure 2a), four-Furcation, six-Furcation and eight-Furcation. Using the same rule but in the opposite direction, Confluence can be defined and divided into four classes: two-Confluence (as shown in Figure 2b), four-Confluence, six-Confluence and eight-Confluence. In particular, six-Furcation, eight-Furcation, six-Confluence and eight-Confluence are caused by noise.

(a) (b)

Figure 2. The definition of Furcation and Confluence: (a) Furcation; (b) Confluence.

Curves (its definition and extraction will be discussed in Section 4) inside the local region of

C_n can be extracted based on Furcation and Confluence. In this paper, the summit of the innermost curve is regarded as the core point. Hence, the extraction of FC is the first step of the proposed method.

2.2. Furcation and Confluence extraction

Through experiments and observation, the following statements can be made for FC: if two adjacent JLs of a fingerprint start at the same ridge/valley and end at another same ridge/valley, then

Statement 1: Their intersection number will be same without furcation or confluence on the two adjacent JLs, and

Statement 2: The widths of corresponding intersections between two adjacent JLs will be nearly same without furcation or confluence on the two JLs.

However, as shown in Figure3, when both Furcation and Confluence appear on the same JL, the number of intersections of two adjacent JLs could be same, and make Statement 1 inapplicable for FC extraction. Consequently, Only Statement 2 is adopted. Two kinds of FC extractors namely Furcation extractor, cz_0, and Confluence extractor, cz_1, are introduced. According to the definition of FC in Section 2.1, the number and width of fingerprint lines intersecting adjacent JLs can be used to determine Furcation or Confluence, and the extractor cz_0 and cz_1 (both of them are two-dimensional matrices) are based on this principle. Specifically, let JL_i and JL_{i+1} be adjacent JL. The value in cz_0 is defined as the result of comparing the total width of the first k intersections of JL_i with the fingerprint line (from left to right, including ridges and valleys) with the first $k+1$ intersections of JL_{i+1}, which corresponds to Furcation if it is less than 0. Similarly, the value in cz_1 is defined as the result of comparing the total width of the first k intersections of JL_{i+1} with the fingerprint line with the first $k+1$ intersections of JL_i, which corresponds to Confluence if it is greater than 0. The mathematical expressions are given in Eq (7) and Eq (9), respectively. Take Figure 4 as an example, the total width of the first 6 intersections of JL_2 is less than the first 5 intersections of JL_1, and the value of cz_0 is less than 0 at this time, indicating that the extraction to Furcation.

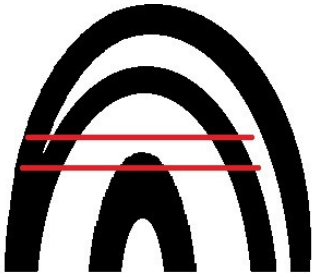


Figure 3. Furcation and Confluence located in the same judge line.

2.2.1. Construction of matrix E, YS and Z

To better explain the formation of cz_0 and cz_1, some matrices are required to be constructed. The matrices are described as follows: E_i, YS_i and Z_i.

1) E_i, the ith row of E, records the width of fingerprint line: For the ith JL. The width of every intersection is counted and recorded from left to right, padding 0 at empty positions, and then the E_i can be obtained.

2) YS_i, the ith row of YS, records the type of every intersection: For every nonzero element of E_i. Let 1 represent intersections located in valleys and -1 represent intersections located in ridges, padding 0 if $E_i(j)=0$.

3) Z_i, the ith row of Z, records the number of intersections of the ith JL: For each row of E, the number of positive integers is counted, then the vector Z_i is obtained.

2.2.2. Furcation and Confluence extractor in ideal situation

As discussed above, Statement 2 works under the premise that the adjacent two JLs start at same fingerprint line and end at another same fingerprint line, and that is called an ideal situation. We will explain how the Statement 2 is used for FC extractor formation in an ideal situation.

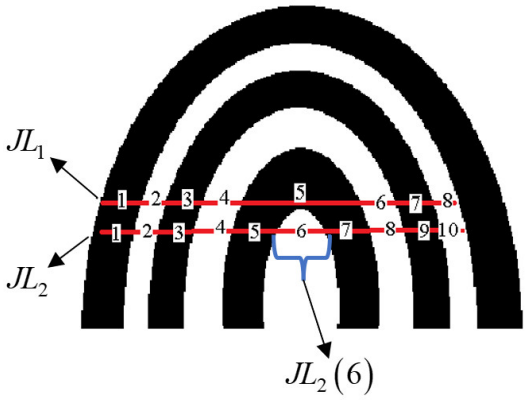


Figure 4. FC extractor formation of an ideal situation.

As shown in Figure 4, let JL_1 be the upper JL, JL_2 be the adjacent one below JL_1, and $JL_i(j)$ be the width of j th intersection on the i th JL. Then we have

$$JL_1(j) \approx JL_2(j), \text{when } j \in [1,4] \tag{1}$$

Equation (1) is the rule that is described in Statement 2. But the rule is broken while $j=5$. When a Furcation appears in Figure 4, we have

$$JL_1(5) > JL_2(5)$$

and even

$$JL_1(5) > JL_2(5) + JL_2(6) \tag{2}$$

Equation (2) is satisfied only if a Furcation or Confluence appears. The general format of Eq (2) can be expressed as

$$JL_1(k) > JL_2(k) + JL_2(k+1) \tag{3}$$

where k denotes the intersection that Furcation appears.

If we use JL_i and JL_{i+1} to represent two adjacent JLs, then Eqs (1) and (3) can be written in E respectively as

$$\sum_{j=1}^{k-1} E(i,j) \approx \sum_{j=1}^{k-1} E(i+1,j) \tag{4}$$

and

$$E(i,j) > E(i+1,j) + E(i+1,j+1) \tag{5}$$

Adding Eq (4) to Eq (5), we obtain Eq (6). It is true only that k is an intersection where Furcation appears.

$$\sum_{j=1}^{k} E(i,j) > \sum_{j=1}^{k+1} E(i+1,j) \tag{6}$$

According to the definition of cz_0 above, then we have

$$cz_0(i,k) = \sum_{j=1}^{k+1} E(i+1,j) - \sum_{j=1}^{k} E(i,j) \tag{7}$$

where $cz_0(i,k)$ denotes the Furcation extractor in ideal situation. And then $cz_0(i,k) < 0$ denotes the appearance of a Furcation while traverse E using Eq (7).

Similarly, while a Confluence appears, the intersection's relationship between two adjacent JLs can be expressed as

$$E(i+1,j) > E(i,j) + E(i,j+1) \tag{8}$$

Then the Confluence extractor in ideal situation can be written as

$$cz_1(i,k) = \sum_{j=1}^{k} E(i+1,j) - \sum_{j=1}^{k+1} E(i,j) \tag{9}$$

where $cz_1(i,k) > 0$ denotes the appearance of a Confluence.

However, the ideal situation is not always satisfied. The two adjacent JLs may start or end at different fingerprint lines, a situation called "Edge Dislocation"(ED) as shown in Figure 5. Besides, the emerging of more than two FCs in the same JL will generate mistakes if Eqs (7) and (9) are used directly. These two problems can be solved by introducing the "Edge Compensating Factor" (ECF) and the "Jumper Factor" (JF) respectively.

2.2.3. The Edge Compensating Factor and the Jumper Factor

Let JL_i be the upper JL in Figure 5(a), and JL_{i+1} be the lower one. As Edge Dislocation occurs between these two JLs, the extra intersection, which is $E_i(1)$ in this example, needs to be cut off prior to subsequent processing.

(a)

$$\begin{matrix} E_i \\ E_{i+1} \end{matrix}\begin{bmatrix} 1 & 3 & 1 & 3 & 10 & 4 & 3 & 1 \\ 7 & 5 & 2 & 6 & 4 & 3 & 0 & 0 \end{bmatrix}$$

(b)

$$\begin{matrix} YS_i \\ YS_{i+1} \end{matrix}\begin{bmatrix} 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & -1 & 1 & 0 & 0 \end{bmatrix}$$

(c)

Figure 5. The case of a non-ideal situation: (a) a binary image with two adjacent JLs; (b) the width of corresponding intersections in E_i and E_{i+1}; (c) the types of corresponding intersections in YS_i and YS_{i+1}.

Through observation, there are two properties that can be used to identify the ED:
Property 1 The first intersections of two adjacent JLs are of different types when an ED appears;
Property 2 The difference of the widths between the first intersections of two adjacent JLs is greater than a threshold when an ED appears.

Equations (10) and (11) below can be obtained by using these properties.

$$cys(1) = YS_i(1) \times YS_{i+1}(1) \tag{10}$$

$$d(1) = E_{i+1}(1) - E_i(1) \tag{11}$$

where $cys(1) = -1$ indicates the satisfaction of Property 1, and $|d(1)| \geq d_T$ indicates the satisfaction of Property 2, where d_T denotes the average width of ridge/valley in a binary fingerprint image. An ED emerges when both Properties are satisfied.

Then, the values of ECFs need to be assigned. Let cw_0 and cw_1, with initial values of 0, be the ECF of JL_i and JL_{i+1} respectively. The assignment rules are as follows:

1) $\begin{cases} cw_0 \leftarrow 1 \\ cw_1 \leftarrow 0 \end{cases}$, if d(1) > 0

2) $\begin{cases} cw_0 \leftarrow 0 \\ cw_1 \leftarrow 1 \end{cases}$, if d(1) ≤ 0

Incorporating these ECFs into Eqs (7) and (9), we obtain Eqs (12) and (13), which are capable of cutting off the extra intersections.

$$zc_{01}(j,w) = \sum_{k=1+cw_1}^{w+1+cw_1} E_i(j+1,k) - \sum_{l=1+cw_0}^{w+cw_0} E_i(j,l) \tag{12}$$

$$zc_{11}(j,w) = \sum_{l=1+cw_0}^{w+1+cw_0} E_i(j,l) - \sum_{k=1+cw_1}^{w+cw_1} E_i(j+1,k) \tag{13}$$

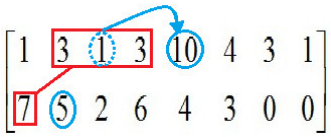


Figure 6. Add marks to Figure 5(b).

To clarify the necessity of introducing JF, we redraw Figure 5(b) by adding some marks, as shown in Figure 6, and then deal with the JLs in Figure 5(a) by using Eqs (12) and (13).

When $w = 1$, there are

$$zc_{11}(j,1) = -3 < 0$$

which indicates an appearance of a Confluence. As this is a two-Confluence (the way of discriminating the type of FC will be discussed later), the upper three intersections, consisting of $E_i(j,2)$, $E_i(j,3)$ and $E_i(j,4)$, correspond to the $E_i(j+1,1)$, as the red rectangles marked in Figure 6. Then if we continue to traverse E_i by increasing w by 1, it is

$$zc_{11}(j,2) = -5 < 0$$

which indicates a mistakenly detected Confluence. The JF need to be introduced to eliminate this kind of error by adjusting w in Eqs (12) and (13). In this example, JF equals to 2 and we only adjust w in $\sum E_i(j,l)$. As show in Figure 6, when the two-Confluence is detected, w of $\sum E_i(j,l)$ jumps two more steps from the dotted blue circle to the solid blue one. And the w in

$\sum E_i(j+1,k)$ stays at the solid blue circle in the second row. After this process, the next Furcation or Confluence can be extracted accurately.

The value of JF is determined according to the type of FC, so we will discuss the assignment method of JF along with the total procedure of FC extraction in Section 2.2.5.

2.2.4. Furcation and Confluence type determinator

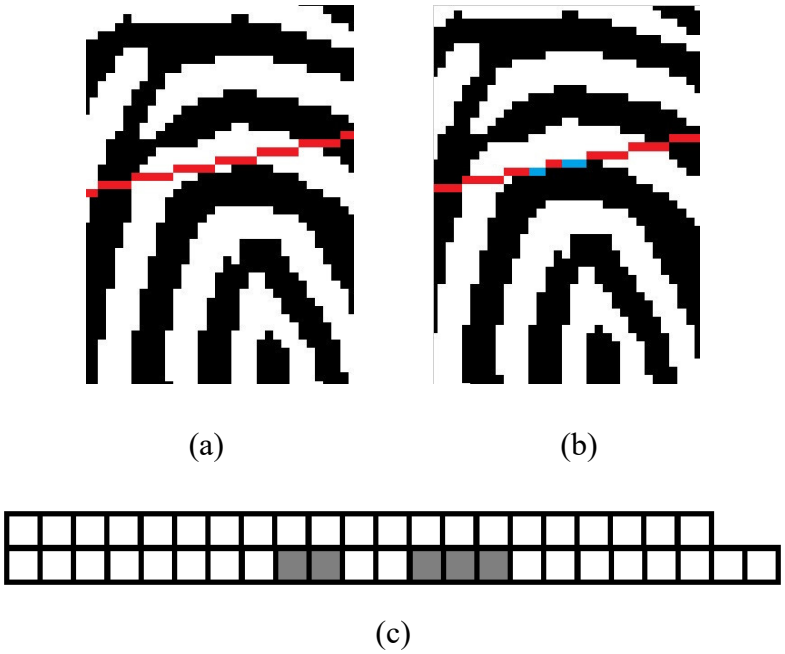


Figure 7. The determination of Furcation type: (a) the first JL; (b) the second JL; (c) pixels on JLs of the Furcation area in (a) & (b).

Through observation, we found that the FC type can be determined by calculating the width of intersections on JLs. Take Figure 7 as an example to interpret how FC types can be determined. The red line in Figure 7(a),(b) are two adjacent JLs, the blue pixels in Figure 7(b) represent the intersections of the JL with the curved ridge. Figure 7(c) shows the pixels on JLs of the Furcation area, where the white squares indicate the pixels on the valleys and the dark-gray ones indicate the pixels on the ridges. It can be easily seen from Figure 7(c) that the sum of lower five intersections nearly equal the width of the upper intersection. This quantitative relation can also be found in other types of Furcation or Confluence and is therefore useful for the determination of FC types.

Take Furcation as an example, the above quantitative relation can be expressed as

$$E_i(j,x) \approx \sum_{k=x}^{x+N} E_i(j+1,k) \tag{14}$$

where x denotes column coordinates of a Furcation and N denotes the type of a Furcation ($N=2$ for a two-Furcation, $N=4$ for a four-Furcation, and so on). Then the problem of determining the Furcation type is converted to finding the optimal solution for N satisfying Eq (14).

The determination of Confluence type is almost the same.

2.2.5. The procedure of Furcation and Confluence extraction

Take ECF and JF into consideration, the FC extractors in a practical application are as follows

$$cz_0(j,w)=\sum_{k_0=1}^{w_1+1}E_i(j+1,k_0)-\sum_{l_0=1}^{w_0}E_i(j,l_0) \tag{15}$$

$$cz_1(j,w)=\sum_{l_1=1}^{w_0+1}E_i(j,l_1)-\sum_{k_1=1}^{w_1}E_i(j+1,k_1) \tag{16}$$

where

$$w_0=w+cw_0+a_0$$

$$w_1=w+cw_1+a_1$$

a_0 and a_1 are cumulative values of JF (the assignment method of them will be discussed later).

The procedure of FC extraction is as follows:

Step 1: Traverse E_i by using Eqs (15) and (16) (Note here that Eqs (10) and (11) will be used to calculate the corresponding cw_0 and cw_1, and that both a_0 and a_1 need to be initialized to 0 once j is increased by 1). $cz_0(j,w)<0$ indicates the appearance of a Furcation, and $cz_1(j,w)<0$ indicates the appearance of a Confluence. Then, execute Step 2) to determine FC type.

Step 2: The numerical implementation of FC type determination will be discussed in this step. Since the determination of the Furcation type and the Confluence type are theoretically the same, we take Furcation as an example to show the numerical implementation.

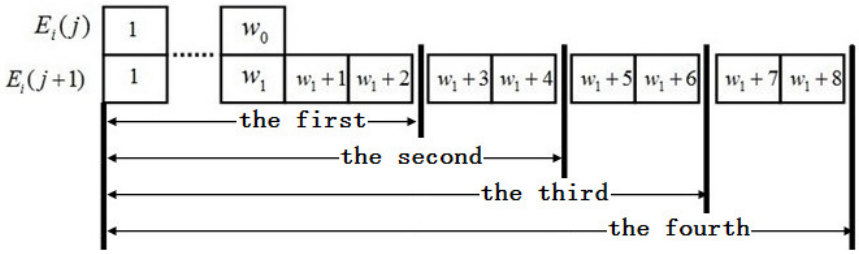


Figure 8. The determination of Furcation type.

As shown in Figure 8, let a Furcation appear on the intersection marked with w_0, four intervals

correspond to four different values of N in Eq (14) ranging from $N=2$ to $N=8$. It can be easily seen that to determine the optimal solution for N that satisfies Eq (14) is equivalent to finding the interval in Figure 8 whose $\left|\sum E_i(j+1)-\sum E_i(j)\right|$ value is the minimal. This can be achieved iteratively, which means that the $\left|\sum E_i(j+1)-\sum E_i(j)\right|$ values for all four intervals do not need to be calculated, which can reduce the computational cost.

Equations (17) and (18) are used to determine the Furcation type.

$$g_{01}=\left|c_0+\sum_{k_{01}=w_1+1}^{w_1+2+2bc_0}E_i(j+1,k_{01})\right| \tag{17}$$

$$g_{02}=\left|c_0+\sum_{k_{02}=w_1+1}^{w_1+4+2bc_0}E_i(j+1,k_{02})\right| \tag{18}$$

where $c_0=cz_0(j,w)-E_i(j+1,w_1+1)$ and the initial value of bc_0 is 0.

Algorithm 1 the process of determining the Furcation type

$bc_0 \leftarrow 0$
while true **do**
  $g_{01} \leftarrow \left|c_0+\sum_{k_{01}=w_1+1}^{w_1+2+2bc_0}E_i(j+1,k_{01})\right|$
  $g_{02} \leftarrow \left|c_0+\sum_{k_{02}=w_1+1}^{w_1+4+2bc_0}E_i(j+1,k_{02})\right|$
  $b_0 \leftarrow w_1+4+2bc_0$
  if $b_0 > Z_i(j+1,1)$ or $g_{01} > g_{02}$ **then**
    $N=-2bc_0-2$ /*N indicates the Furcation type*/
    break
  else
    $bc_0 \leftarrow bc_0+1$
  end if
end while

We use pseudocode formatting to depict the process of determining the Furcation types, as shown in Algorithm 1.

Similarly, the equations for Confluence type determination are as follows

$$g_{11} = \left| c_1 + \sum_{k_{11}=w_0+1}^{w_0+2+2bc_1} E_i\left(j, k_{11}\right) \right| \tag{19}$$

$$g_{12} = \left| c_1 + \sum_{k_{12}=w_0+1}^{w_0+4+2bc_1} E_i\left(j, k_{12}\right) \right| \tag{20}$$

where $c_1 = cz_1\left(j, w\right) - E_i\left(j, w_0+1\right)$, and the initial value of bc_1 is 0.

The process of Confluence type determination is also described using a pseudocode format, as shown in Algorithm 2.

Algorithm 2 the process of determining the Confluence type

$bc_1 \leftarrow 0$
while true **do**
    $g_{11} = \left| c_1 + \sum_{k_{11}=w_0+1}^{w_0+2+2bc_1} E_i\left(j, k_{11}\right) \right|$
    $g_{12} = \left| c_1 + \sum_{k_{12}=w_0+1}^{w_0+4+2bc_1} E_i\left(j, k_{12}\right) \right|$
    $b_1 = w_0 + 4 + 2bc_1$
    if $b_1 > Z_i\left(j,1\right)$ or $g_{11} > g_{12}$ **then**
        $N \leftarrow -2bc_1 - 2$ /*N indicates the Furcation type*/
        break
    else
        $bc_1 \leftarrow bc_1 + 1$
    end if
end while

FC types need to be stored once they are determined. A zero matrix FJ_i with the same dimensions as E_i is defined and it can be updated as,

$$FJ_i\left(j+1, w_1 + bc_0 + 1\right) \leftarrow 2bc_0 + 2 \tag{21}$$

$$FJ_i\left(j+1, w_1\right) \leftarrow -2bc_0 - 2 \tag{22}$$

Each positive value in FJ_i corresponds to a Furcation type, and a negative value corresponds

to a Confluence type.

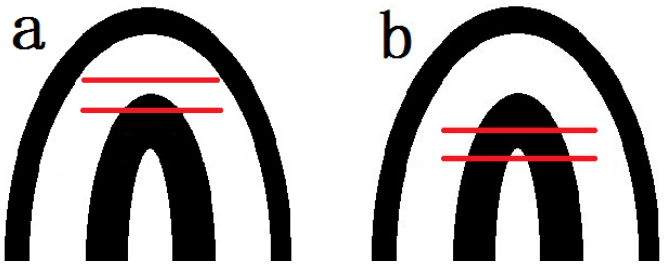


Figure 9. Two kinds of two-Furcation.

Note that there is another trait of Furcation that should be recorded. Figure 9 shows two different two-Furcation. In Figure 9a, the intersection lines of the Furcation in the upper JL lines in valley, on the contrary, as shown in Figure 9b, the intersection lines in ridge. This trait is recorded by a matrix LX_i which has the same dimension as E_i and each element has an initial value of zero, and is updated in the following way

$$LX_i\left(j+1, w_1+bc_0+1\right) \leftarrow YS_i\left(j, w_0\right) \tag{23}$$

Equation (23) is used once a Furcation is detected.

Now, Step 2 is finished. Then, go to Step 3 to assign value of JF.

Step 3: The value of JF is bounded up with the FC type. Let jf_0 be the JF of a Furcation, jf_1 be the JF of a Confluence, a_0 and a_1 be the cumulative value of JF. The equations or assignment of these factors are as follows:

$$LX_i\left(j+1, w_1+bc_0+1\right) \leftarrow YS_i\left(j, w_0\right) \tag{24}$$

$$jf_1 \leftarrow 2bc_1+2 \tag{25}$$

$$a_0 \leftarrow a_0+jf_1 \tag{26}$$

$$a_1 \leftarrow a_1+jf_0 \tag{27}$$

where a_0 and a_1 are cumulative values of JF, whose initial values are both 0. Once a Furcation is detected and the Furcation type is determined, the jf_0 will be calculated and the corresponding a_1 will be updated; Confluences are processed in a similar way.

Steps 1–3 are repeated and executed for the entire E_i. The final result FJ_i contains all Furcations and Confluences that have been extracted.

3. Study on false FC and FC validation

After FC extraction, matrix FJ_i contains all candidate FCs of the local region of core C_n. False FCs exist in FJ_i due to two kinds of noise (noise will be introduced in section 3.1). We analyzed the effect of noise on FCs and proposed an algorithm to remove false FCs.

3.1. False Furcation and Confluence

As shown in Figure 10, island, lake, independent ridge, and the insufficiency of Binarization algorithm are regarded as the first kind of noise, which will result in noisy binarized images (as shown in Figure10e). False FCs will be detected in these noisy regions (as shown in Figure 10f). This kind of noise can be solved by developing a special Binarization algorithm to obtain an ideal binary image, as shown in Figure 11. However, this idea has two main drawbacks: one is that the special Binarization algorithm is difficult to design; the other is that the second kind of noise cannot be removed by adopting any Binarization algorithm.

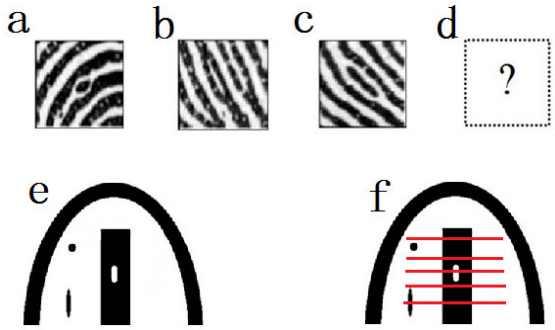


Figure 10. The first kind of noise and the corresponding effects on FCs: (a) island; (b) lake; (c) independent ridge; (d) the insufficient of Binarization algorithm; (e) the effects of (a) (b) (c) (d) on a binary image; (f) the occurrence of false FCs.

Figure 11. An ideal binary image.

Figure 12 shows the second kind of noise, which is caused by the characteristic of the digital

image. Figure 12a shows an ideal binary image with three adjacent JLs. By magnifying the blue rectangular area, the spatial relationship between each JL and the ridge can be illustrated clearly, as shown in Figure 12b–d respectively. Note that in Figure 12b–d, each square represents a pixel, with the red ones indicating pixels on the JLs, the light gray ones indicating pixels on the ridges, and the dark gray ones indicating pixels that intersect between the JLs and the ridges. Based on Figures 12b,c, a four-Furcation can be detected, while from Figures 12c,d, a two-Confluence can be detected. However, by definition, only a two-Furcation should be detected from these three JLs.

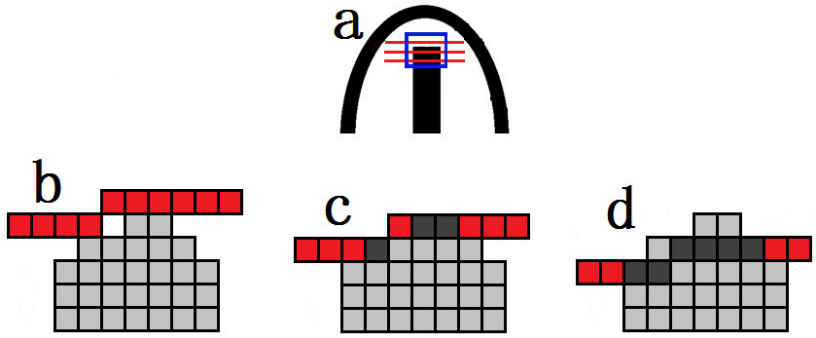


Figure 12. The second kind of noise and the corresponding effect on FC.

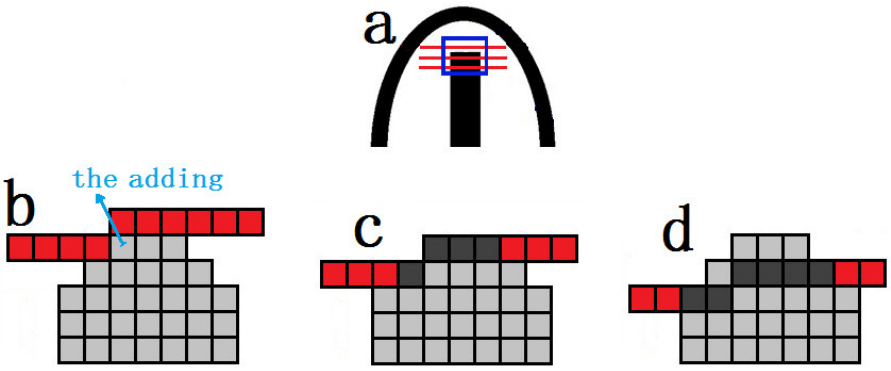


Figure 13. The method of adding pixels to remove the second kind of noise.

This kind of misdetection can be eliminated by adding or padding some pixels at the specific position in the binary image. For the case of Figure 12, the corresponding added pixel is shown in Figure 13. It can be seen that only a two-Furcation will be detected from Figure 13b,c. Theoretically, this method can achieve our goal. But technically, it is difficult to implement. We have not found a unified rule to deal with this specific situation.

3.2. Correlations between false Furcation and Confluence

Either developing a special Binarization algorithm or designing a pixel padding method is of tremendous difficulty. So we have experimented alternative ways for FC validation. Through experiment, we found some correlations between false FCs that can be used to eliminate them.

The first correlation is that a pair of Furcation and Confluence is always detected from a given

noisy region. Take Figure 14 as an example, which contains a lake like noisy region. One can see this correlation clearly that a false two-Furcation and a false two-Confluence appear together. The same phenomenon can also be found in Figure 12 that a false four-Furcation is followed by a false two-Confluence.

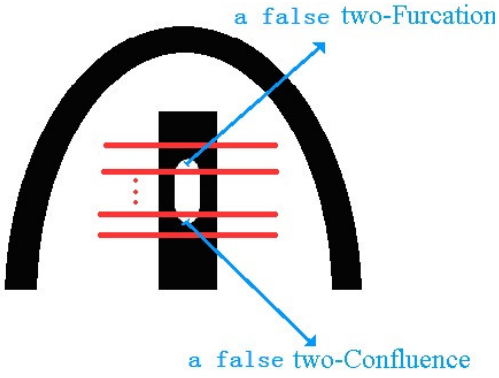


Figure 14. False FCs in a lake like noise region.

Therefore, the first correlation of false FCs can be stated as:

Correlation 1. False FCs always appear in pairs.

The second correlation is about the coordinates of paired false FCs in FJ_i. Let fc_t be a false Furcation and (f_t, c_t) be its coordinate in FJ_i. Let $jh_t(j_t, h_t)$ be the paired false Confluence. Two quantitative relations between them can be expressed as:

1) $|f_t - j_t| < T$. ($T = 12$ for an image with resolution of 500 dpi, and the value of T should vary with the image resolution);

2)
$$\begin{cases} lb(jh_t) \geq lb(fc_t) - \dfrac{FJ_i(f_t, c_t)}{2} \\ lb(jh_t) \leq lb(fc_t) + \dfrac{FJ_i(f_t, c_t)}{2} - 2 \end{cases} \tag{28}$$

where $lb(z)$ denotes the normalized column coordinate of z. So, the second correlation can be concluded as:

Correlation 2. Paired false FCs satisfy the above two quantitative relations.

If let

$$\begin{cases} a_t = lb(fc_t) - \dfrac{FJ_i(f_t, c_t)}{2} \\ b_t = lb(fc_t) + \dfrac{FJ_i(f_t, c_t)}{2} - 2 \end{cases} \tag{29}$$

then Eq (28) can be written as

$$lb\left(jh_t\right)\in\left[a_t,b_t\right] \tag{30}$$

The validity of Eq (30) will be explained that by the following examples.

Let's reconsider Figure 14 as the first example and let the top red line be the j_1th JL, the bottom red line be the $\left(j_1+m_1\right)$th JL, the false Furcation be fc_{t1}, the false Confluence be jh_{t1}. Then the coordinates of the Furcation in FJ_i can be written as $fc_{t1}\left(j_1,3\right)$, and the Confluence can be written as $jh_{t1}\left(j_1+m_1,2\right)$. The type of fc_{t1} is $FJ_i\left(j_1,3\right)=2$. Note that each normalized column coordinate of these two FCs is equal to its column coordinate that

$$\begin{cases} lb\left(fc_{t1}\right)=3 \\ lb\left(jh_{t1}\right)=2 \end{cases}$$

Then it can be directly calculated by Eq (29) that

$$\begin{cases} a_{t1}=2 \\ b_{t1}=2 \end{cases}$$

which means that Eq (30) is satisfied that

$$lb\left(jh_{t1}\right)\in\left[a_{t1},b_{t1}\right]$$

Take Figure 12 as the second example, let fc_{t2} be the false Furcation and jh_{t2} be the false Confluence. As normalized column coordinate is also equal to the column coordinate, and it would be calculated similarly that

$$lb\left(jh_{t2}\right)=2$$

$$\left[a_{t2},b_{t2}\right]=\left[1,3\right]$$

which means Eq (30) is satisfied.

Equation (30) is still valid for some special examples such as Figures 15 and 16. Let the false Furcation in Figure15 be fc_{t3} and the false Confluence be jh_{t3}. It can be seen that $lb\left(jh_{t3}\right)=1$ and $\left[a_{t3},b_{t3}\right]=\left[1,3\right]$, which satisfy Eq (30). For Figure 16, let the false Furcation and false Confluence be fc_{t4} and jh_{t4}, respectively. There are $lb\left(jh_{t4}\right)=3$ and $\left[a_{t4},b_{t4}\right]=\left[1,3\right]$ which

also satisfy Eq (30).

FC validation is based on these two correlations discussed above. Note that the concept of normalized column coordinate should be introduced and it will be explained in the following section.

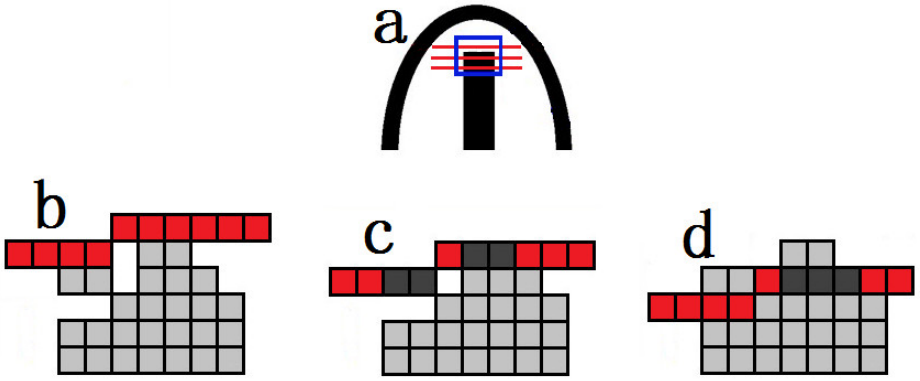


Figure 15. A special case of false Furcation and false Confluence.

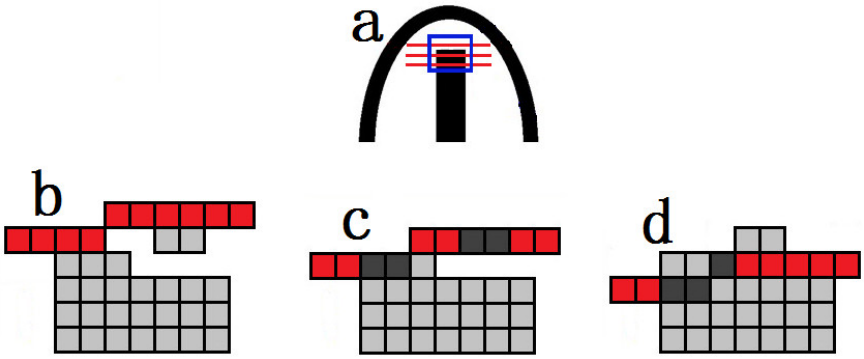


Figure 16. Another special case of false Furcation and false Confluence.

3.3. Column coordinates normalization

Usually, the normalization of columns is necessary because that the ridges/valleys around FCs affect the calculation of Eq (30). As shown in Figure 17, the coordinates of paired FCs A and B should satisfy Eq (30) due to the influence of ED and the left ridge, but they do not. This means the correlation-based FC validation discussed above will fail.

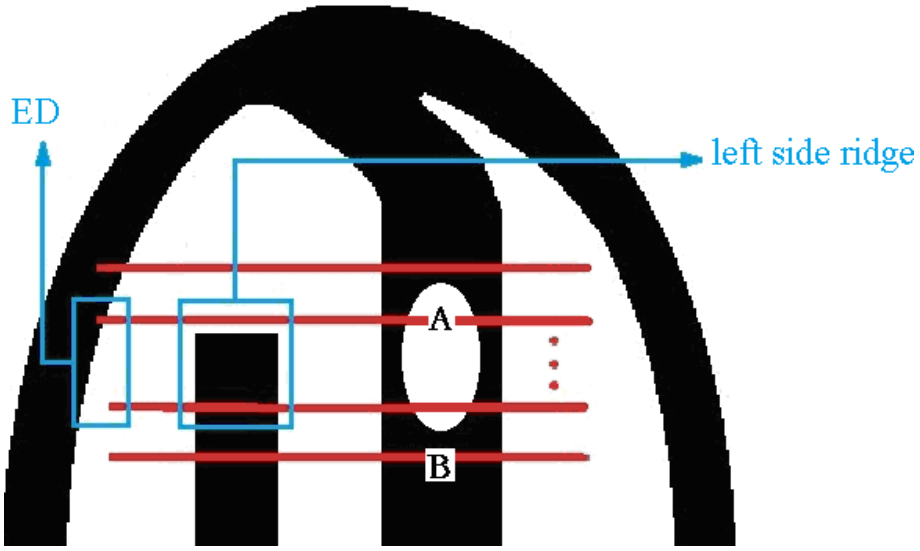


Figure 17. The effects of surrounding fingerprint lines on the coordinates of FCs.

The effect of ridges/valleys can be offset by a method called column coordinate normalization. One can see from Figure 17 that these effects can be divided into two categories:

1) The effect of ED.

2) The effect of ridges on the left side.

Column coordinates normalization aims at offsetting these two kinds of effects.

The effect of ED on adjacent JLs had been discussed in the previous section. Here, we need to offset the ED between any two JLs. The Edge Compensating Vector (ECV) is introduced to record the number of ED between any JL and the first JL, while the first JL will be used as a benchmark to offset the ED between any two JLs. The ECV can be calculated recursively from the ECF as

$$B_v(1, j+1) = cw_1 - cw_0 + B_v(1, j) \tag{31}$$

where B_v is the ECV with length of 60 and initial value of zero. cw_1 and cw_0 are ECF of adjacent JLs numbered $j+1$ and j respectively. After traversing the 60 JLs by Eq (31), the vector B_v can be obtained.

We will use Figure 17 as an example to express the idea of eliminating the effect of the left ridge and will discuss the numerical implementation of column coordinate normalization.

For the false FCs in Figure 17, it is easy to see that if the left side ridge marked by the blue rectangle is "erased", its effect on FCs' column coordinates will disappear. We don't really erase these ridges, but instead, adjust the column coordinates accordingly to achieve the effect of erasure. So the basic idea is: for Figure 17, traverse JL one by one from A to B, decrease or increase the column coordinate of A once ridges appear or disappear on the left side of it. Then the effect of left side ridges on A and B can be eliminated.

Algorithm 3 the process of column coordinates normalization

1: Calculate the absolute columns of FCs as

$$WF\left(fc_{t5}\right)=c_{t5}+B_v\left(1,f_{t5}\right)-\frac{FJ_i\left(f_{t5},c_{t5}\right)}{2}$$
$$WJ\left(jh_{t5}\right)=h_{t5}+B_v\left(1,j_{t5}\right) \quad (32)$$

/* where $WF(fc_{t5})$ is the absolute column of the Furcation and $WJ(jh_{t5})$ is the absolute column of the Confluence */

2: Assign $WF(fc_{t5})$ to wf, and traverse the elements of FJ_i between $\left(f_{t5}+1,1\right)$ and $\left(j_{t5},h_{t5}-1\right)$. For each non-zero element fj, whose coordinate is (x,y), using Eq (32) to calculate the absolute column of fj

3: **if** $WF\left(fj\right)<wf$ or $WJ\left(fj\right)<wf$ **then**

4: $wf \leftarrow wf+FJ_i\left(x,y\right)$ (33)

5: **end if** /* wf and $WJ(jh_{t5})$ are the normalized columns of the Furcation and the Confluence respectively */

Let $fc_{t5}\left(f_{t5},c_{t5}\right)$ be a Furcation and $jh_{t5}\left(j_{t5},h_{t5}\right)$ be a Confluence below it. We use pseudocode formatting to depict the process of column coordinates normalization, as shown in Algorithm 3.

3.4. Furcation and Confluence validation

We can use the two correlations discussed above to detect paired false FCs and correct or eliminate false FCs.

Let's take Figures 12 and 14 as examples to illustrate this rule. In Figure 14, the value of the false Furcation is $FJ_i\left(j_1,3\right)=2$, and the value of the false Confluence is $FJ_i\left(j_1+m,2\right)=-2$. Then the summation will be 0. By assigning the summation to $FJ_i\left(j_1,3\right)$ and to $FJ_i\left(j_1+m,2\right)$ respectively, the paired false FCs will be corrected. For Figure 12, the value of the false Furcation is $FJ_i\left(j_2,3\right)=4$, and the value of the false Confluence is $FJ_i\left(j_2+1,2\right)=-2$. Then the summation will be 2. By assigning the summation to $FJ_i\left(j_2,3\right)$ and assigning 0 to $FJ_i\left(j_2+1,2\right)$ respectively, the false FCs will be corrected.

Every paired false FCs can be corrected or eliminated by the above method. For a paired false FCs, let $FJ_i\left(x,y\right)$ be the false Furcation and $FJ_i\left(x',y'\right)$ be the false Confluence, the above method can be written as

$$\begin{cases} a \leftarrow FJ_i(x,y) + FJ_i(x',y') \\ b \leftarrow 0 \end{cases} \tag{34}$$

where

$$\begin{cases} a = FJ_i(x,y) \\ b = FJ_i(x',y') \end{cases}, \left|FJ_i(x,y)\right| \geq \left|FJ_i(x',y')\right|$$

$$\begin{cases} a = FJ_i(x',y') \\ b = FJ_i(x,y) \end{cases}, \left|FJ_i(x,y)\right| < \left|FJ_i(x',y')\right|$$

Then, take all the aspects discussed in Chapter 3.2 into account, the validation process is given in the form of pseudocode, as shown in Algorithm 4.

Algorithm 4 FC validation process

/* Assign initial values to some variables. */
$A \leftarrow [-8 \quad -6 \quad -4 \quad -2]$, $j \leftarrow 1$
$pb \leftarrow A(1,j)$, $\overline{FJ_i} \leftarrow FJ_i$
while there is any element in A that has not completed traversal of $\overline{FJ_i}$ **do**
    while $\overline{FJ_i}(x_1,y_1) = pd$ does not exist **do**
        /* Where $\overline{FJ_i}(x_1,y_1)$ represent any element of it */
        $j \leftarrow j+1$
        $pb \leftarrow A(1,j)$
        traverse $\overline{FJ_i}$
    end while
    find (x_2,y_2) satisfying $\overline{FJ_i}(x_2,y_2) > 0$, where $x_2 \in [max(x_1-12,1), x_1-1]$
    if find and that (x_1,y_1), (x_2,y_2) satisfy Eq (30) **then**
        use Eq.(34) to update $\overline{FJ_i}(x_1,y_1)$ and $\overline{FJ_i}(x_2,y_2)$
        $j \leftarrow j+1$
        $pb \leftarrow A(1,j)$
    else
        $j \leftarrow j+1$
        $pb \leftarrow A(1,j)$
    end if
end while /* Exiting the loop if all the elements in A finished traversal of $\overline{FJ_i}$ */
/* $\overline{FJ_i}$ is the result of FC validation. */

4. Curve extraction and accurate core point localization

A fraction of a crooked ridge is called a Curve, as shown the red part in Figure 18. It can be seen that the core point is the summit of the innermost Curve. As shown in Figure 19, the red ridge is the innermost curve, and the blue star is the core.

Figure 18. Curve.

(a) (b)

Figure 19. The summit of an innermost Curve and corresponding core point.

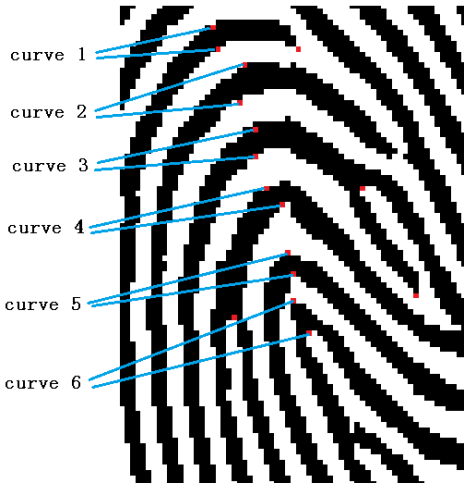


Figure 20. Curves and the corresponding Furcations.

As shown in Figure 20, red pixels denote the Furcation or Confluence. One can find two Furcations located on each side of a Curve, which can be used as marks of the Curve. For each

Curve, let fc_{h1} be the upper Furcation and fc_{h2} be the other one. Three properties can be used to identify them:

1) The absolute difference between their row coordinates is less than t_h; (t_h is determined by the resolution of input images. For example, $t_h = 12$ is for the input image of 500 dpi.)

2) After column coordinate normalization, the absolute difference between their columns is 1;

3) In LX_i, the value of fc_{h1} is 1, and the value of fc_{h2} is -1.

4.1. Innermost Curve extraction

All Curves can be identified by using the three properties above. In fact, only the innermost Curve is concerned. We depict the extraction process of the innermost curve in the form of pseudocode, as shown in Algorithm 5.

4.2. Accurate core point localization

The summit of the innermost Curve can be obtained by averaging $fc_{h1}(x_{h1}, y_{h1})$ and $fc_{h2}(x_{h2}, y_{h2})$. Since fc_{h1} and fc_{h2} are preserved in $\overline{FJ_i}$, a mapping is needed to transfer the coordinates of $\overline{FJ_i}$ to the coordinates of input fingerprint image. Let (x, y) be the coordinate of a FC in $\overline{FJ_i}$, $\overline{y}$ can be obtained by

$$\overline{y} = \sum_{j=1}^{y-1} E_i(x, j) + \left\lfloor \frac{E_i(x, y)}{2} \right\rfloor \tag{35}$$

where $(x, \overline{y})$ denotes the $\overline{y}$ th pixel of the x th judge line. Finally, the coordinate of a core can be obtained by

$$\begin{aligned} x_c &= \left\lfloor \frac{(x_{h1} + x_{h2})}{2} \right\rfloor \\ y_c &= \left\lfloor \frac{\left(\overline{y_{h1}} + \overline{y_{h2}}\right)}{2} \right\rfloor \end{aligned} \tag{36}$$

where (x_c, y_c) denotes the coordinate of the core.

Algorithm 5 innermost curve extraction process

while true **do**
  while true **do**
    Traverse $\overline{FJ_i}$ and LX_i from the bottom up to find out the element that satisfies

$$\begin{cases} \overline{FJ_i}(x_{h2}, y_{h2}) > 0 \\ LX_i(x_{h2}, y_{h2}) = -1 \end{cases} \tag{37}$$

    /* Element that satisfies Eq.(35) represents a Furcation lying in a ridge. Use fc_{h2} to denote this Furcation. */
    Traverse $\overline{FJ_i}$ and LX_i whose row coordinates are in the range of $[x_{h2} - t_h, x_{h2}]$ to find out the element that satisfies

$$\begin{cases} \overline{FJ_i}(x_{h1}, y_{h1}) > 0 \\ LX_i(x_{h1}, y_{h1}) = 1 \end{cases} \tag{38}$$

    if find **then**
      use fc_{h1} to denote this Furcation
      break
  end while
  Use Eq (32) and Eq (33) to calculate the normalized column coordinates between fc_{h1} and fc_{h2}
  if the absolute difference between their normalized columns is 1 **then**
    fc_{h1} and fc_{h2} are the two Furcations that mark the innermost Curve, which can be recorded by

$$\begin{cases} Hu(1,1) \leftarrow (x_{h1}, y_{h1}) \\ Hu(2,1) \leftarrow (x_{h2}, y_{h2}) \end{cases} \tag{39}$$

    break
  end if
end while

5. Experiments and analysis

For a direct comparison with the approach in [8], we use the same dataset(FVC2000-DB2) for our experiments. As manually defined cores are always below the initial cores detected by the algorithm proposed by [8], we only consider the area below the initial core. If no curve is extracted, the initial core is regarded as the final result; otherwise, the summit of the innermost Curve is regarded as the final result.

FVC2000-DB2 contains 880 fingerprint images from 110 individuals captured by a commercial capacitive sensor with a resolution of 500 dpi. Except for the initial cores that either cannot be detected or located near the boundary of the fingerprint images, the remaining 960 cores are suitable for our experiments.

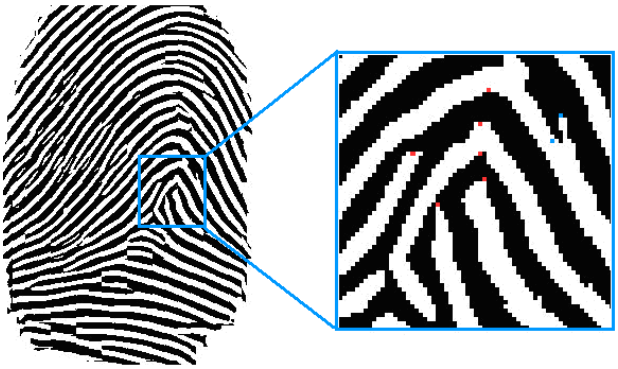
Figure 21. The initial core and the located core.

Figure 21 shows some examples of the experimental results. Pictures on the left column show the local regions of core points, in which black points (s_{black}) represent the initial cores that were detected by the method in [8], and red points (s_{red}) represent the final core located by the proposed method in this paper. Pictures on the right column are binary images whose innermost Curves are marked in red. These results can be divided into three categories: Figure 21(a)–(c) belong to the first

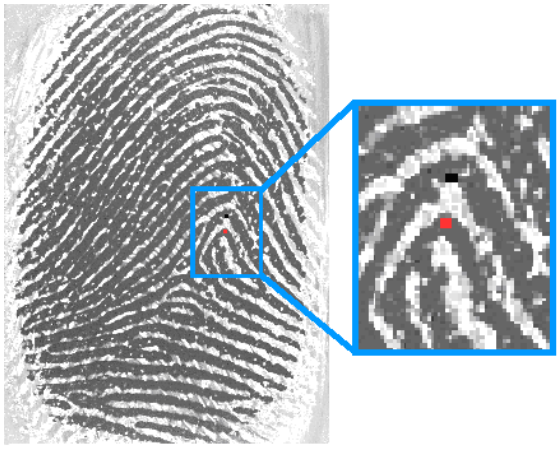
category; Figure 21(d)–(f) belong to the second category; Figure 21(g),(h) belong to the third category.

In the first category, s_{red} is closer to the manually defined core than s_{black}. This means that the core location accuracy increases. This category can be further divided into two kinds. The first kind, such as Figure 21(a),(b), has a relatively ideal binary image while the second kind, such as Figure 21(c), hasn't. Some mistakes in the binary image do not affect the location results, and this demonstrates the robustness of the proposed method.

In the second category, core point localization accuracy does not increase and s_{red} and s_{black} are nearly at the same position. This category can be further divided into two kinds. In the first kind, like Figure 21(d), s_{black} is closer to the manually defined core so that there's no space for accuracy improvement. In the second kind, like Figure 21(e),(f), no Curves can be extracted due to the less-than-ideal binarization results, and consequently, s_{black} is regarded as the core location result.



(a)



(b)

Figure 22. The located cores from the extracted Furcation and confluence: (a) The summits of curves extracted from Furcation and Confluence; (b) The initial core from Jin and Kim [8] and the located core by the propose method.

Table 1. The numbers of each of SP in SP_c.

Location accuracy	Promoted	Stayed the same	Declined
Number of cores	391	529	40
Percentage	40.8%	55%	4.2%

In the third category, localization accuracy declines because of the noisy binary image. Some false innermost Curves were extracted, making s_{red} further away from the manually defined core than s_{black}.

All experimental results can be classified into the above three categories. Table 1 shows the statistical results comparing with the method of Jin and Kim [8] using the database of FVC2000-DB2, in which 40.8% of cores' localization accuracy increased while 4.2% declined. By the result of Figures 22 and 23, it is clear that there is a significant increase in core localization accuracy for the proposed method.

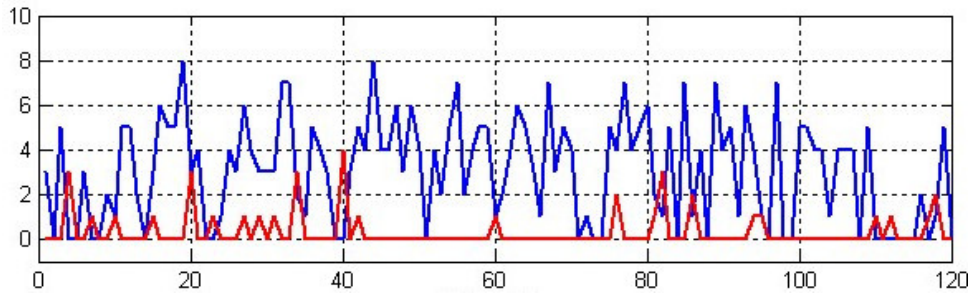


Figure 23. Overall performance comparison between the proposed (the blue curve) and the Jin and Kim [8] (the red curve) on FVC2000-DB2. The horizontal ordinate denotes different individual fingerprints. The vertical ordinate denotes the core number of fingerprint each individual. Each point of curves means that, to an individual fingerprint, the number of cores locating by the corresponding method same as the manually defined ones by Henry's definition.

6. Conclusions

For processing the pixel-level accuracy grade of singular point detection and providing more accurate reference points for the fingerprint alignment. This paper proposed a novel core point pixel-level localization method only based on fingerprint features of spatial domain. The method defined new fingerprint characteristics called Furcation and Confluence to represent ridge/valley distribution in a core point area and used them to extract the innermost Curve of ridges. Furthermore, the correlation between Furcation and Confluence was utilized to remove false Furcation and Confluence, enhancing the computation robustness against noise. Finally the summit of this Curve is regarded as the core point location. Experimental results comparing with the method of Jin and Kim [8] using the database FVC2000-DB2 demonstrated that the proposed method achieved better core

localization accuracy for 40.8% of the samples, similar accuracy for 55% samples, and less accuracy for 4.2% of the samples.

Acknowledgments

This work was supported by the National Key Natural Science Foundation of China(No. U19B2016) and the National Science and Foundation of China(No. 60802047) through the Pattern Recognition and Information Security Lab (PRIS) in Hangzhou Dianzi University.

Conflict of interest

All authors declare no conflict of interest in this paper.

References

1. E. R. Henry, Classification and Uses of Finger Prints, Routledge, London, 1913.
2. F. Wu, J. Zhu, X. Guo, Fingerprint pattern identification and classification approach based on convolutional neural networks, *Neural Comput. Appl.*, **32** (2020), 5725–5734. doi: 10.1007/s00521-019-04499-w.
3. H. W. Jung, J. H. Lee, Noisy and incomplete fingerprint classification using local ridge distribution models, *Pattern Recognit.*, **48** (2015), 473–484. doi: 10.1016/j.patcog.2014.07.030.
4. J. Khodadoust, A. M. Khodadoust, Fingerprint indexing based on minutiae pairs and convex core point, *Pattern Recognit.*, **67** (2017), 110–126. doi: 10.1016/j.patcog.2017.01.022.
5. G. A. Bahgat, A. H. Khalil, N. S. Abdel Kader, S. Mashali, Fast and accurate algorithm for core point detection in fingerprint images, *Egypt. Inf. J.*, **14** (2013), 15–25. doi: 10.1016/j.eij.2013.01.002.
6. M. Kawagoe, A. Tojo, Fingerprint pattern classification, *Pattern Recognit.*, **17** (1984), 295–303. doi: 10.1016/0031-3203(84)90079-7.
7. L. Fan, S. Wang, T. Guo, Global and local information combined to detect singular points in fingerprint images, *Sci. China Inf. Sci.*, **56** (2012), 1–13. doi: 10.1007/s11432-011-4516-0.
8. C. Jin, H. Kim, Pixel-level singular point detection from multi-scale Gaussian filtered orientation field, *Pattern Recognit.*, **43** (2010), 3879–3890. doi: 10.1016/j.patcog.2010.05.023.
9. N. T. Le, D. H. Le, J. W. Wang, C. C. Wang, Entropy-based clustering algorithm for fingerprint singular point detection, *Entropy*, **21** (2019), 786. doi: 10.3390/e21080786.
10. F. Belhadj, S. Akrouf, S. Harous, S. A. Aoudia, Efficient fingerprint singular points detection algorithm using orientation-deviation features, *J. Electron. Imaging*, **24** (2015), 033016.1–033016.13. doi: 10.1117/1.JEI.24.3.033016.
11. A. M. Bazen, S. H. Gerez, Systematic methods for the computation of the directional fields and singular points of fingerprints, *IEEE Trans. Pattern Anal. Mach. Int.*, **24** (2002), 905–919. doi: 10.1109/TPAMI.2002.1017618.
12. C. Militello, V. Conti, F. Sorbello, S. Vitabile, A novel embedded fingerprints authentication system based on singularity points, in *2008 International Conference on Complex, Intelligent and Software Intensive Systems*, (2008), 72–78. doi: 10.1109/CISIS.2008.56.
13. V. Conti, C. Militello, S. Vitabile, F. Sorbello, Introducing pseudo-singularity points for efficient

fingerprints classification and recognition, in *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, (2010), 368–375. doi: 10.1109/CISIS.2010.134.
14. M. Sabir, T. M. Khan, M. Arshad, S. Munawar. Reducing computational complexity in fingerprint matching, *Turk. J. Electr. Eng. Comput. Sci.*, **28** (2020), 2538–2551. doi: 10.3906/elk-1907-113.
15. T. M. Khan, D. G. Bailey, M. A. U. Khan, Y. Kong, Efficient hardware implementation for fingerprint image enhancement using anisotropic gaussian Filter, *IEEE Trans. Image Process.*, **26** (2017), 2116–2126. doi: 10.1109/TIP.2017.2671781.
16. S. Chikkerur, A. N. Cartwright, V. Govindaraju, Fingerprint enhancement using STFT analysis, *Pattern Recognit.*, **40** (2007), 198–211. doi: 10.1016/j.patcog.2006.05.036.
17. X. Wang, J. Li, Y. Niu, Definition and extraction of stable points from fingerprint images, *Pattern Recognit.*, **40** (2007), 1804–1815. doi: 10.1016/j.patcog.2006.10.012.

AIMS Press