Mathematical Problems of Computer Science 39, 21--30, 2013.

New Fingerprint Image Thinning Algorithm

Davit A. Kocharyan

Institute for Informatics and Automation Problems of NAS of RA
e-mail: david.kocharyan@gmail.com

Abstract

Minutiae-based fingerprint recognition systems rely heavily on efficient and fast image enhancement algorithms. An image thinning is a very important stage of the image enhancement. A good thinning algorithm preserves the structure of the original fingerprint image, reduces the amount of data needed to process and helps to improve the feature extraction accuracy and efficiency. In this paper we describe and compare some of the most used fingerprint thinning algorithms. The results show that the faster algorithms have difficulty in preserving connectivity. Zhang and Suen's algorithm gives the least processing time, while Guo and Hall's algorithm produces the best skeleton quality. In this paper we propose a modified Zhang and Suen's algorithm that is efficient and fast. Some test results show that the proposed modification better preserves the structure and connectivity of the original fingerprint image.

Keywords: Fingerprint Recognition, Image Enhancement, Image Thinning; minutiae.

1. Introduction

Fingerprint image thinning is a very important step in fingerprint recognition algorithms. In this step the ridge lines of the fingerprint image are transformed to a one-pixel thickness. This process is fundamental for fingerprint recognition algorithms [2], because the thinned images are easier to process and reduce the processing time. Thinning does not change the structure of the fingerprint image: it preserves the locations of the fingerprint ridge and valley features, which makes easier to identify the global and local features of the fingerprint image (such as Core, Delta, Minutiae points) that are used for fingerprint classification, recognition and matching [1]. An example of thinned fingerprint image is shown in Figure 1 below:

Fig. 1. From left to right: Original fingerprint image; Binarized image and the corresponding thinned image.

An effective and accurate thinning algorithm directly affects the fingerprint feature extraction, the matching accuracy and the results. The best known thinning algorithms fall into the following two categories: Iterative and Non-iterative [3].

Iterative algorithms delete pixels on the boundary of a pattern repeatedly until only the unit pixel-width thinned image remains. Non-iterative distance transformation algorithms are not appropriate for general applications since they are not robust, especially for patterns with highly variable stroke directions and thicknesses. Thinning, based on iterative boundary removal, can be divided into sequential and parallel algorithms. Thinning is mostly done on the binarized image of the fingerprint. The mostly discussed and described thinning algorithms are based on parallel thinning, as they are fast and efficient. In this paper we intend to describe and compare the most used iterative fingerprint thinning algorithms (Zhang-Suen [4], Guo-Hall [2], Abdulla et al [5], R. W. Hall [6]), to understand their strengths and weaknesses and propose a modified and more efficient algorithm.

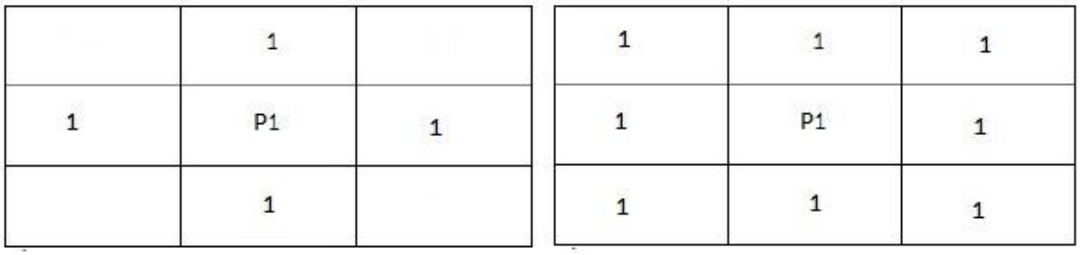
2. Concepts

The binary image I is described as a matrix $M \times N$, where $x(i, j)$ represents the binary value of the pixel (i, j), equal to 1, if the pixel is black, or 0, if the pixel is white.

Any pixel which is at distance of 1 from the pixel (i, j) is considered a neighbor for that pixel.

Connectivity is defined as the number of neighbors to which the pixel is connected:

- *4-connectivity:* The pixel is connected to every horizontal and vertical neighbor (see Figure 2a).
- *8-connectivity:* The pixels are connected to every horizontal, vertical and diagonal neighbor (see Figure 2b).



(a) (b)

Fig. 2. Pixel connectivity: (a) 4-connectivity; (b) 8-connectivity.

3. Known Thinning Algorithms

In this chapter some known fingerprint thinning algorithms are described.

1. ***Zhang-Suen's Algorithm***

The Zhang-Suen's algorithm works using a 3×3 sized block. It is an iterative algorithm and it removes all the contour points of the image except those that belong to the skeleton. The algorithm is divided into two sub-iterations [4].
The algorithm is described below:

1. **While** points are deleted, **do**
2. **For** all $p(i, j)$ pixels, **do**
3. **If** (a) $2 \le B(P_1) \le 6$
   (b) $A(P_1) = 1$
   (c) One of the following is true:
      1. $P_2 \cdot P_4 \cdot P_6 = 0$ in odd iteration,
      2. $P_2 \cdot P_4 \cdot P_8 = 0$ in even iteration,
   (d) One of the following is true:
      1. $P_4 \cdot P_6 \cdot P_8 = 0$ in odd iteration,
      2. $P_2 \cdot P_6 \cdot P_8 = 0$ in even iteration,

   then
4. **Delete** pixel $p(i, j)$,

where $A(P_1)$ is the number of 0 to 1 transitions in the clockwise direction from P_9, $B(P_1)$ is the number of non-zero neighbors of P_1:

$$B(P_1) = P_2 + P_3 + \Lambda + P_9.$$

P_1 is not deleted, if any of the above conditions is not met.
The algorithm is fast, but fails to preserve such patterns that have been reduced to 2×2 squares: they are completely removed. It also has problems preserving connectivity with diagonal lines and identifying line endings.

2. ***Guo-Hall's Algorithm***

The algorithm works using a 2×2 sized block. $C(P_1)$ is defined as the number of distinct 8-connected components of P_1 [2]. $B(P_1)$ is defined as the number of non-zero neighbors of P_1: $\bar{}$, $\wedge$ and $\vee$ symbols are defined as logical Completing, AND, and OR, respectively. $N(P_1)$ is defined as:

$$N(P_1) = \min\{N_1(P_1), N_1(P_1)\},$$

where

$$N_1(P_1) = (P_9 \vee P_2) + (P_3 \vee P_4) + (P_5 \vee P_6) + (P_7 \vee P_8),$$

$$N_2(P_1) = (P_2 \vee P_3) + (P_4 \vee P_5) + (P_6 \vee P_7) + (P_8 \vee P_9).$$

$N_1(P_1)$ and $N_2(P_1)$ divide the neighbors of P_1 into four pairs and calculate the number of pairs that contain one or two non-zero elements.
The algorithm is described below:

1. **While** points are deleted, **do**
2. **for** all $p(i, j)$ pixels, **do**
3. **if** (a) $C(P_1) = 1$
   (b) $2 \le N(P_1) \le 3$
   (c) One of the following is true:
      1. $(P_2 \vee P_3 \vee \overline{P_5}) \vee P_4 = 0$ in odd iteration,
      2. $(P_6 \vee P_7 \vee \overline{P_9}) \wedge P_8 = 0$ in even iteration,

then

4. **Delete** pixel $p(i, j)$.

When $B(P_1) = 1$, P_1 is an ending point and $N(P_1) = 1$. But when $B(P_1) = 2$, P_1 could also be a non-ending point. The definition of $N(P_1)$ preserves the ending points and removes the redundant pixel in the middle of the curve.

Guo-Hall [5] algorithm is more precise than Zhang-Suen's [4] algorithm, but it needs more computational time to execute.

3. *Abdulla et al's Algorithm*

The algorithm uses a 3×3 sized block and consists of two sub-iterations [5]. The first sub-iteration scans the image horizontally using a 3×4 sized block (Fig. 3a). Any two points, which are horizontally adjacent to each other and horizontally isolated from other pixels, are deleted. The second sub-iteration scans the image vertically using a 4×3 sized block (Fig. 3b). Any two points, which are vertically adjacent to each other and vertically isolated from other points, are deleted.

P_9	P_2	P_3	P_{10}
P_8	P_1	P_4	P_{11}
P_7	P_6	P_5	P_{12}

(a)

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5
P_{12}	P_{11}	P_{10}

(b)

Fig. 3. (a) 3×4 sized block; (b) 4×3 sized block.

The algorithm is described below:

1. **While** points are deleted, **do**
2. **for** all pixels $p(i, j)$ **do**
3. First iteration:
4. **if** (a) $\overline{SP_{1,1}} \wedge P_6 = 1$ or

(b) $\overline{SP_{1,2}} \wedge P_2 = 1$ or

(c) $[(P_2 \wedge \overline{P_3}) \vee (P_3 \wedge \overline{P_2}) \vee \overline{P_9}] \wedge [(\overline{P_5} \wedge P_6) \vee (P_5 \wedge \overline{P_6} \wedge P_7] = 1$

then

5. **Delete** pixel P_1
6. Where $SP_{1,1} = P_3 \vee P_2 \vee P_9$, $SP_{1,2} = P_6 \vee P_5 \vee P_7$, $\overline{\ }$, $\wedge$ and $\vee$ are defined as logical Completing, AND, OR, respectively.
7. **if** P_1 is not deleted

then

8. **if** $(\overline{P_3} \wedge P_{10}) \vee (\overline{P_5} \wedge P_{12}) = 1$

then

9. **Delete** pixel P_4.
10. Second iteration:
11. **if** (a) $\overline{SP_{2,1}} \wedge P_4 = 1$ or

(b) $\overline{SP_{2,2}} \wedge P_8 = 1$ or

(c) $[(P_8 \wedge \overline{P}_7) \vee (P_7) \wedge \overline{P}_8 \vee \overline{P}_9] \wedge [(\overline{P}_4 \wedge P_5) \vee (P_5 \wedge \overline{P}_4 \wedge P_3)] = 1$

then

12. **Delete** pixel P_1.

13. where $SP_{2,1} = P_9 \vee P_8 \vee P_7$, $SP_{2,2} = P_3 \vee P_4 \vee P_5$, $\overline{\ }$, $\wedge$ and $\vee$ are defined as logical Completing, AND, OR, respectively:

14. **if** P_1 is not deleted

then

15. **if** $(\overline{P}_7 \wedge P_{12}) \vee (\overline{P}_5 \wedge P_{10}) = 1$

then

16. **Delete** pixel P_6.

4. ***R. W. Hall's Algorithm***

The algorithm [6] consists of two parallel sub-iterations, functions first identifying in parallel all deletable pixels and then deleting in parallel all those deletable pixels except certain pixels which should be maintained to preserve the connectivity in an image.

The algorithm is described below:

1. **While** pixels are deleted, **do**
2. for all pixels $p(i, j)$ **do**
3. Determine whether $p(i, j)$ should be deleted
4. **if** (a) $1 < B(P_1) < 7$

(b) $P_1's$ 8-neighborhood contains exactly one 4-connected component of 1's.

then

5. $p(i, j)$ should be deleted
6. for all $p(i, j)$ pixels, **do**
7. **if** (a) $P_2 = P_6 = 1$ and P_4 is deletable,

(b) $P_4 = P_8 = 1$ and P_6 is deletable,

(c) P_4, P_5, P_6 are deletable,

then

8. Do not delete pixel $p(i, j)$.

The above-mentioned conditions preserve local connectivity, end-points and 2×2 sized patterns.

4. Comparison of the Algorithms

During the comparison, the evaluation was based on the following criteria: connectivity, spurious branches, convergence to unit width and data reduction efficiency/computational cost.

Connectivity preservation of a fingerprint pattern is a crucial fingerprint recognition, as the disconnected patterns may produce false minutiae points.

Spurious branches also produce false minutiae points. Some post processing operations may be applied to remove spurious branches, but it will cost extra processing operations and execution time.

A perfect skeleton must be unitary, meaning that it does not contain any of the patterns given in Figure 4:

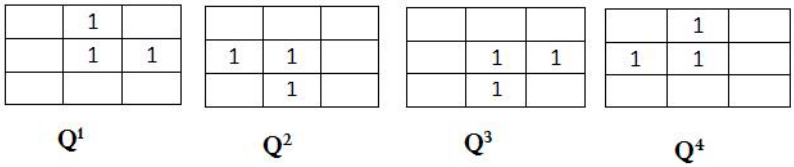


Fig. 4. Patterns of non-unitary skeletons.

Jang and Chin [7] introduced a measure m_t to compute the width of the thinned S_m skeleton:

$$m_t = 1 - \frac{Area\left[\bigcup_{1 \le k \le 4} S_m Q_k\right]}{Area[S_m]},$$

where *Area*[ ] is the operation that counts the number of pixels with the value of 1. If $m_t = 1$, then S_m is a perfect unitary skeleton [7].

An effective thinning algorithm should also be **fast**. A measure to evaluate both the data reduction efficiency and the computational cost was defined by Jang and Chin [7] as:

$$m_d = \min\left[1, \frac{Area[S] - Area[S_m]}{n \cdot Area[S]}\right],$$

where *n* is the number of parallel operations required to converge, and S is the original input image. This measure has a value between 0 and 1. The larger value means the higher efficiency [7]. To compare the above described algorithms, they have been applied to thin five different images, shown in Figure 5. The results of the values m_t and m_d are given in Table 1.

Fig. 5. Five different fingerprint images used for comparing the thinning algorithms: (1) 276x408 pixels; (2) 408x480 pixels; (3) 264x264 pixels; (4) 336x336 pixels; (5) 420x600 pixels.

Table 1: Results of the tests.

Image	Algorithm	Results	
		$\overline{m_t}$	$\overline{m_d}$
1	▪ Abdulla et.al	0.996	0.117
	▪ Guo-Hall	0.998	0.062
	▪ Hall	0.991	0.083
	▪ Zhang-Suen	0.698	0.129
2	▪ Abdulla et.al	0.974	0.120
	▪ Guo-Hall	0.997	0.065
	▪ Hall	0.988	0.085
	▪ Zhang-Suen	0.790	0.137
3	▪ Abdulla et.al	0.997	0.122
	▪ Guo-Hall	0.998	0.061
	▪ Hall	0.999	0.084
	▪ Zhang-Suen	0.864	0.130
4	▪ Abdulla et.al	0.978	0.105
	▪ Guo-Hall	0.993	0.056
	▪ Hall	0.993	0.079
	▪ Zhang-Suen	0.747	0.115
5	▪ Abdulla et.al	0.985	0.118
	▪ Guo-Hall	0.997	0.064
	▪ Hall	0.993	0.085
	▪ Zhang-Suen	0.695	0.134

The results show that Guo-Hall's algorithm best preserves the structure of the image, but the efficiency and speed are low, giving the result of $m_d = 0.062,$ a comparatively low value. Zhang-Suen's algorithm is most frequently used in literature and shows an average $m_d = 0.129.$ But in some cases it does not preserve the structure of the image and even removes some ridges and end-points [8].

5. Proposed Modification

We propose a slight modification to the Zhang-Suen's algorithm, as it is the most efficient one. The proposed modification improves and preserves the structure of the image and stops an unwanted removal of lines and end-points. In the original algorithm the end-points are detected by $A(P_1) = 1,$ but it does not apply to diagonal ridges that have 2 pixel thickness, as in that case $A(P_1) = 2.$ The following conditions can be added to Zhang-Suen's algorithm to eliminate those problems: In odd iterations: when $A(P_1) = 2,$ the following conditions are checked:

In odd iterations: when $A(P_1) = 2,$ the following conditions are checked:

1. $P_4 \times P_6 = 1$ and $P_9 = 0$ or
2. $P_4 \times P_2 = 1$ and $\bar{P}_3 \times \bar{P}_7 \times \bar{P}_8 = 1$

In even iterations: when $A(P_1) = 2,$ the following conditions are checked:

1. $P_2 \times P_8 = 1$ and $P_5 = 0$ or
2. $P_6 \times P_8 = 1$ and $\bar{P}_3 \times \bar{P}_4 \times \bar{P}_7 = 1$

These conditions are added to avoid deleting diagonal lines and preserve connectivity.

The modified algorithm is described below:

1. **While** points are deleted, **do**
2. **for** all p(i, j) pixels, **do**
3. **if** $2 \leq B(P_1) \leq 6$
4. **if** $A(P_1) = 1$ and
   (a) One of the following is true:
   1. $P_2 \times P_4 \times P_6 = 0$ in odd iteration,
   2. $P_2 \times P_4 \times P_8 = 0$ in even iteration,
   (b) One of the following is true:
   1. $P_4 \times P_6 \times P_8 = 0$ in odd iteration,
   2. $P_2 \times P_6 \times P_8 = 0$ in even iteration,
   then
5. **Delete** pixel p(i, j).
6. **else if** $A(P_1) = 2$ and
   (a) One of the following is true:
      1. $P_4 \times P_6 = 1$ and $P_9 = 0$, in odd iteration,
      2. $P_2 \times P_8 = 1$ and $P_5 = 0$, in even iteration,
   (b) One of the following is true:
      1. $P_4 \times P_2 = 1$ and $\bar{P}_3 \times \bar{P}_7 \times \bar{P}_8 = 1$ in odd iteration,
      2. $P_6 \times P_8 = 1$ and $\bar{P}_3 \times \bar{P}_4 \times \bar{P}_7 = 1$, in even iteration,
   then

5. **Delete** pixel p(i, j).

where $A(P_1)$ is the number of 0 to 1 transitions in the clockwise direction from P_9, $B(P_1)$ is the number of non-zero neighbors of P_1:

$$B(P_1) = P_2 + P_3 + \Lambda + P_9.$$

P_1 is not deleted, if any of the above conditions is not met.

After adding the above-mentioned conditions, Zhang Suen's algorithm preserves the structure and fairly maintains connectivity. A comparison of skeletons produces the original algorithm and the modified version shown in Figure 6, where the corresponding minutiae points are also shown:

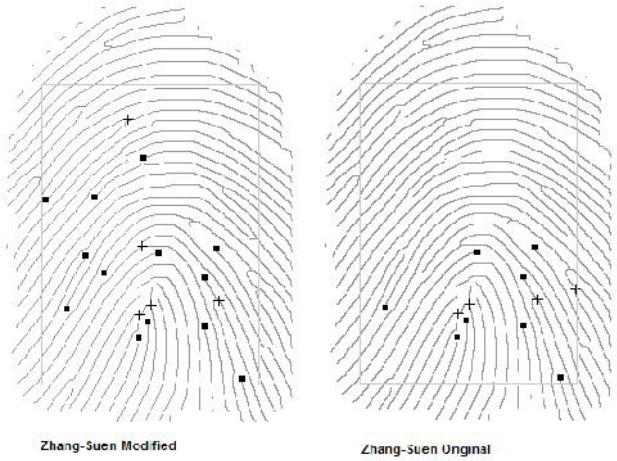


Fig. 6. Left to right: Modified and original versions of Zhang Suen's algorithm

A noticeable improvement in maintaining structure and connectivity can be seen. The modified algorithm has been applied to thin five different images, shown in Figure 5. The results of the values m_t and m_d are given in the table below:

Table 2: Results of the tests

Image	Results	
	$\overline{m_t}$	$\overline{m_d}$
1	0.897	0.130
2	0.943	0.132
3	0.976	0.143
4	0.935	0.113
5	0.896	0.136

The modified algorithm shows a noticeable improvement with average m_t= 0.929 and an average m_d = 0.130.

6. Conclusion and Future Work

In this paper we discussed the most frequently used fingerprint thinning algorithms and showed their comparisons. Zhang Suen's [4] algorithm proves to be the most efficient one, and with the proposed modification it shows the best result among all with respect to the comparison criteria. As to the next step, a creation of a fingerprint recognition software solution based on minutiae matching and the proposed thinning algorithm are planned.

References

[1] D. Maltoni and D.Maio, *Handbook of Fingerprint Recognition*, Springer, 2009.
[2] Z. Guo and R. Hall, "Parallel thinning with two-subiteration algorithms,"*Communications of the ACM*, vol. 32, pp. 359–373, 1989.
[3] R. Gupta and R. Kaur, "Skeletonization algorithm for numerical patterns,"*International Jornal of Signal Processing, Image Processing and Pattern Recognition*, vol. 1, pp. 63–72, 2008.
[4] T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, pp. 236–239, 1984.
[5] W. Abdulla, A. Saleh, and A. Morad, "A preprocessing algorithm for hand-written character recognition," *Pattern Recognition Letters 7*, pp. 13–18, 1988.
[6] R. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation," *Communications of the ACM*, vol. 32, pp. 124–129, 1989.
[7] B. Jang and T. Chin, "One-pass parallel thinning: Analysis, properties, and quantitative evaluation," *IEEE Transactions on Pattern Analysis and Mechine Intelligence*, pp. 1129–1140, 1992.
[8] G. Raju and Y. Xu, "Study of parallel thinning algorithms," *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 01, pp. 661–666, 1991.
[9] S. Prabhakar, A. K. Jain and S. Pankanti, "Learning fingerprint minutiae location and type", *Pattern Recognition*, vol. 36, no. 8, pp. 1847–1857, 2003.
[10] K. Mali and S. Bhattacharya, "Fingerprint recognition using global and local structures", *International Journal on Computer Science and Engineering*, vol. 3, no. 1, pp. 161--172, 2011.

Submitted 15.11.2012, accepted 06.02.2013.

Նոր մատնահետքերի պատկերների բարակեցման ալգորիթմ

Դ. Քոչարյան

Ամփոփում

Հատուկ կետերի հիման վրա մատնահետքերի ճանաչման համակարգերում մեծ դեր ունեն արագ և արդյունավետ պատկերների բարելավման ալգորիթմները: Պատկերների բարելավման կարևոր փուլ է հանդիսանում պատկերի բարակեցումը: Բարակեցման արդյունքում պետք է պահպանվեն օրիգինալ պատկերի կառուցվածքային առանձնահատկությունները: Ալգորիթմի աշխատանքի ընթացքում նվազեցվում է մշակման համար պահանջվող տվյալների քանակը՝ դրանով բարձրացնելով մատնահետքի հատկանիշների դուրս բերման ճշգրտությունը և արդյունավետությունը: Այս աշխատանքում համեմատվում են հայտնի և հաճախ օգտագործվող մատնահետքերի բարակեցման ալգորիթմները: Արդյունքները ցույց են տալիս, որ արագագործ ալգորիթմները հիմնականում չեն պահպանում պատկերի կառուցվածքը և կապակցվածությունը: Zhang և Suen-ի ալգորիթմը ամենաարագագործն է, իսկ Guo և Hall-ի ալգորիթմը լավագույնն է պահպանում պատկերի կառուցվածքը: Առաջարկվում է ալգորիթմ, որը հանդիսանում է Zhang և Suen-ի ալգորիթմի ձևափոխությունը, արագագործ է և արդյունավետորեն է պահպանում մատնահետքի պատկերի կառուցվածքը: Փորձերի արդյունքները ցույց են տալիս, որ առաջարկվող ալգորիթմը զգալի առաջադիմություն է ցուցաբերում պատկերի կառուցվածքի և կապակցվածության պահպանման մեջ՝ միաժամանակ լինելով արագագործ և արդյունավետ:

Новый алгоритм утончения изображений отпечатков пальцев

Д. Кочарян

Аннотация

Системы распознавания отпечатков пальцев, в основном опираются на алгоритмы, которые эффективно и быстро улучшают изображение. Утончение изображения является очень важным этапом в улучшении изображения. Хороший утончающий алгоритм сохраняет структуру оригинального изображения отпечатков пальцев, уменьшает количество данных необходимых для обработки и улучшает точность и эффективность извлечения характерной черты отпечатка. В данной работе мы описываем и сравниваем некоторые из наиболее часто используемых алгоритмов утончения изображения отпечатков пальцев. Результаты показывают, что быстрые алгоритмы плохо сохраняют соединительность. Алгоритм Zhang и Suen обрабатывает в наикратчайший промежуток времени, а алгоритм Guo и Hall производит наилучшее качество скелета. В данной работе мы предлагаем модифицированный алгоритм Zhang и Suen, который работает эффективно и быстро. Результаты тестов показывают, что предлагаемая модификация лучше сохраняет структуру и соединительность оригинального изображения отпечатков пальцев.