American Journal of Software Engineering and Applications
2013; 2(1) : 1-6
Published online February 20, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20130201.11

A Modified fingerprint image thinning algorithm

Davit Kocharyan

Digital Signal and Image Processing Laboratory
Institute for Informatics and Automation Problems of NAS RA, Yerevan, Armenia

Email address:
david.kocharyan@gmail.com (D. Kocharyan)

To cite this article:
Davit Kocharyan. An Efficient Fingerprint Image Thinning Algorithm, *American Journal of Software Engineering and Applications*. Vol. 2, No. 1, 2013, pp. 1-6. doi: 10.11648/j.ajsea.20130201.11

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

Keywords: Image Thinning; Fingerprint Recognition; Minutiae; Image Enhancement

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 [1], as thinned images are easier to process, and reduce operations processing time. As thinning does not change the structure of the fingerprint image and preserves the locations of the fingerprint ridge and valley features, it 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 [2].

An example of thinned fingerprint image is shown in Figure 1 below:

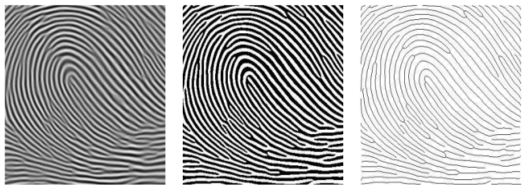

Fig. 1. *From left to right: original fingerprint image, binarized image and corresponding thinned image.*

An effective and accurate thinning algorithm directly affects the fingerprint feature extraction and matching accuracy and results.

Most known thinning algorithms fall into the following two categories [3]:

- Iterative
- Non-iterative

Iterative algorithms delete pixels on the boundary of a pattern repeatedly until only 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 (T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns" *Communications of the ACM*, vol. 27, pp. 236–239, Mar 198),Guo-Hall(Z. Guo and R. Hall, "Parallel thinning with two-subiteration algorithms"*Communications of the ACM*, vol. 32, pp. 359–373, Mar 1989.), Abdulla et al(W. Abdulla, A. Saleh, and A. Morad, "A preprocessing algorithm for hand-written character recognition," *Pattern Recognition Letters 7*, pp. 13–18, 1988.), R. W. Hall(R. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation*," Communications of the ACM*, vol. 32, pp. 124–129, Jan 1989.), understand their strengths and weaknesses and propose a modified and more efficient algorithm.

2. Concepts

The binary image I is described as a matrix MxN, 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 (Fig. 2).

	1	
1	P1	1
	1	

Fig. 2. *4-connectivity: P1 is is connected to every horizontal and vertical neighbor.*

- 8-connectivity: The pixel is connected to every horizontal, vertical and diagonal neighbor (Fig. 3).

1	1	1
1	P1	1
1	1	1

Fig. 3. *8-connectivity:P1 is connected to every horizontal, vertical and diagonal neighbor.*

3. Known Thinning Algorithms

In this chapter some known fingerprint thinning algorithms are described.

3.1. Zhang-Suen's Algorithm

The algorithm works using a 3x3 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 divides into two sub-iterations [4].

The algorithm is describes below:

```
1. While points are deleted, do
2. for all p(i, j) pixels, do
3. if(a)
```

$$2 \leq \mathrm{B}(P_1) \leq 6$$

```
(b) A(P1) = 1
(c) One of the following is true:
1. P2 x P4 x P6 = 0 in odd iteration,
2. P2 x P4 x P8 = 0 in even iteration,
(d) One of the following is true:
1. P4 x P6 x P8 = 0 in odd iteration,
2. P2 x P6 x P8 = 0 in even iteration,
then
4. Delete pixel p(i, j).
```

where A(P1) is the number of 0 to 1 transitions in the clockwise direction from P9, B(P1) is the number of non-zero neighbors of P1:

$$B(P_1) = \sum_{i=2}^{9} P_i$$

P1is not deleted, if any of the above conditions are not met.

The algorithm is fast, but fails to preserve such patterns that have been reduced to 2x2 squares. They are completely removed. It also has problems preserving connectivity with diagonal lines and identifying line endings.

3.2. Guo-Hall's Algorithm

The algorithm works using a 2x2 sized block. C(P1) is defined as the number of distinct 8-connected components of P1. [2]B(P1)is defined as the number of non-zero neighbors of P1. ¯ , □ and □ symbols are defined as logical completing, AND and OR, respectively. N(P1) is defined as:

$$N(P_1) = MIN[N_1(P_1), N_2(P_1)]$$

where:

$$N_1(P_1) = (P_9P_2) + (P_3P_4) + (P_5P_6) + (P_7P_8)$$

$$N_2(P_1) = (P_2P_3) + (P_4P_5) + (P_6P_7) + (P_8P_9)$$

$N_1(P_1)$and$N_2(P_1)$ divideneighbors of P_1 into four pairs and calculated the number of pairs that contain one or two non-zero elements.

The algorithm is describes below:

```
1. While points are deleted, do
2. for all p(i, j) pixels, do
3.if(a) C(P1) = 1
(b)
```

$$2 \leq \mathrm{N}(P_1) \leq 3$$

```
(c) One of the following is true:
1.
```

$$(P_2P_3\bar{P}_5)P_4 = 0$$

in odd iteration,
2.

$$(P_6P_7\bar{P}_9)P_8 = 0$$

in even iteration,
then
4. Delete pixelp(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_1could also be a non-ending point. The definition of$N(P_1)$ preserves the ending points and remove redundant pixel in the middle of the curve.

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

3.3. Abdulla et al's Algorithm

The algorithm uses a 3x3 sized block and consists of two sub-iterations [5]. The first sub-iteration scans the image horizontally using a 3x4 sized block (Fig. 4). 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 4x3 sized block (Fig. 5). 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}

Fig. 4. *3x4 sized block.*

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

Fig. 5. *4x3 sized block.*

The algorithm is describes below:
1. While points are deleted, do
2. for all pixels p(i, j) do
3.First iteration:
4.if(a)

$$\overline{SP_{1.1}}\ P_6 = 1$$

or
(b)

$$\overline{SP_{1.2}}\ P_2 = 1$$

or
(c)

$$[(P_2\ \bar{P}_3)\ (P_3)\ \bar{P}_2\ \bar{P}_9]$$

$$[(\bar{P}_5\ P_6)\ (P_5\bar{P}_6P_7)] = 1$$

then
5.Deletepixel P1.
6.where

$$SP_{1.1} = P_3P_2P_9,\ SP_{1.2} = P_6P_5P_7$$

$\bar{}$, □ and □ are defined as logical completing, AND and OR, respectively.
7. ifP1 is not deleted
then
8.if

$$(\bar{P}_3\ P_{10})\ (\bar{P}_5\ P_{12}) = 1$$

then
9. Deletepixel P_4.
10. Second iteration:
11. if(a)

$$\overline{SP_{2.1}}\ P_4 = 1$$

or
(b)

$$\overline{SP_{2.2}}\ P_8 = 1$$

or

$$(q)\ [(P_8\ \bar{P}_7)\ (P_7)\ \bar{P}_8\ \bar{P}_9]$$

$$[(\bar{P}_4\ P_5)\ (P_5\bar{P}_4P_3)] = 1$$

then
12. Delete pixel P_1.
13.where

$$SP_{2.1} = P_9P_8P_7,\ SP_{2.2} = P_3P_4P_5$$

$\bar{}$, □ and □ are defined as logical completing, AND and OR, respectively:
14. ifP1 is not deleted
then
15.if

$$(\bar{P}_7\ P_{12})\ (\bar{P}_5\ P_{10}) = 1$$

then
16. Delete pixel P_6.

3.4. R. W. Hall's Algorithm

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

The algorithm is describes 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 1s.
then
5. p(i, j) should be deleted
6. for all p(i, j) pixels, do
7.if(a)

$$P_2 = P_6 = 1$$

andP_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 pixelp(i, j).
The above mentioned conditions preserve local connectivity, end-points and 2x2 sized patterns.

4. Comparison

During the comparison the evaluation is 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 crucial fingerprint recognition, as 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 6:

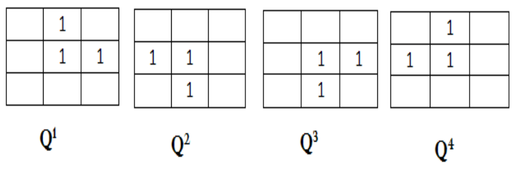

Fig. 6. *Patterns of non-unitary skeletons.*

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

$$m_t = 1 - \frac{Area[\cup_{1 \le k \le 4} S_m Q_k]}{Area[S_m]}$$

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

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

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

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 the value, the higher the efficiency [7].

To compare the above described algorithms, they have been applied to thin five different images, shown in Figure 7.

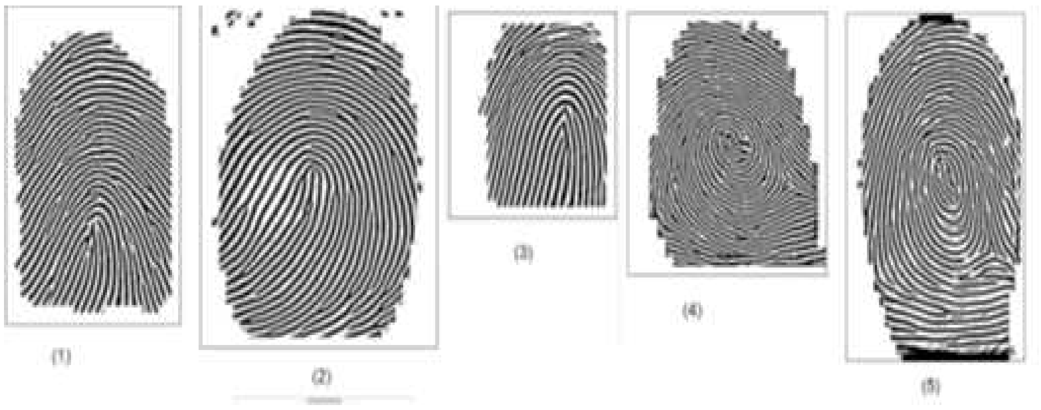

Fig. 7. *Five different fingerprint images used for comparing the thinning algorithms.*

1) 276x408 pixels

2) 408x480 pixels

3) 264x264 pixels

4) 336x336 pixels

5) 420x600 pixels

The results of the values m_t and $m_d$are given in the table below:

Table 1. *Results of the tests.*

Image	Algorithm	Results	
		m_t	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 is low, giving the result of m_d = 0.062, a comparatively low value.

Zhang-Suen's algorithm is the most 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 to improve and preserve structure of the image and stop unwanted removal of lines and end-points.

End-points are detected by the $A(P_1)$ = 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:

1. P4 x P6 = 1 and P9 = 0 or
2. P4 x P2 = 1 and $\bar{P}_3\ x\ \bar{P}_7\ x\ \bar{P}_8 = 1$

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

1. P2 x P8 = 1 and P5 = 0 or
2. P6 x P8 = 1 and $\bar{P}_3\ x\ \bar{P}_4\ x\ \bar{P}_7 = 1$

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 $2 \le B(P_1) \le 6$
4. if $A(P_1) = 1$ and
(a) One of the following is true:
        1. $P_2$ x $P_4$ x $P_6$ = 0 in odd iteration,
        2. $P_2$ x $P_4$ x $P_8$ = 0 in even iteration,
(b) One of the following is true:
        1. $P_4$ x $P_6$ x $P_8$ = 0 in odd iteration,
        2. $P_2$ x $P_6$ x $P_8$ = 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. $P_4$ x $P_6$ = 1 and $P_9$ = 0,in odd iteration,
        2. $P_2$ x $P_8$ = 1 and $P_5$ = 0,in even iteration,
(b) One of the following is true:
    1. P4 x P2 = 1 and $\bar{P}_3\ x\ \bar{P}_7\ x\ \bar{P}_8 = 1$in odd iteration,
    2. P6 x P8 = 1 and $\bar{P}_3\ x\ \bar{P}_4\ x\ \bar{P}_7 = 1$, 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:

$$B(P_1) = \sum_{i=2}^{9} P_i$$

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

After adding the above mentioned conditions, Zhang Suen's algorithm preserved structure and fairly maintains connectivity. A comparison of skeletons produces by the original algorithm and the modified version is shown in Figure 8, where the corresponding minutiae points are also shown:

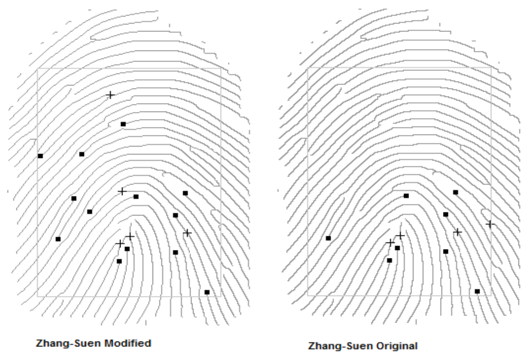

Fig. 8. *Left to right: Modified and original versions of Zhang Suen's algorithm.*

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 7.

The results of the values m_t and $m_d$are given in the table below:

Table 2. *Results of the tests.*

Image	Results	
	m_t	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 mt= 0.929 and an average $m_d =$ 0.130.

6. Conclusion and Future Work

In this paper we discussed the most used fingerprint thinning algorithms and showed their comparisons. Zhang Suen's [4]. algorithm proves to be the most efficient and with the proposed modification shows the best result among all with regards to the comparison criteria.

As the next step, creation of a fingerprint recognition software solution based on minutiae matching and the proposed thinning algorithm is planned.

Fingerprint recognition is the most widely used biometric authentication and identification technology[10], and heavily relies on efficient image processing algorithms and techniques[11]. It has many applications, from consumer to commercial sectors.

References

[1] Davide Maltoni, Dario 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, Mar 1989.

[3] R. Gupta and R. Kaur, "Skeletonization algorithm for numerical patterns", International Jornal of Signal Processing, Image Processing andPattern Recognition, vol. 01, pp. 63–72, Dec 2008.

[4] T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns," Communications of the ACM, vol. 27, pp. 236–239, Mar 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, Jan 1989.

[7] B. Jang and T. Chin, "One-pass parallel thinning: Analysis, properties, and quantitative evaluation," IEEE Transactions on Pattern Analysis andMechine 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, Dec 1991.

[9] S. Prabhakar, A. K. Jain and S. Pankanti, "Learning fingerprint minutiae location and type", Pattern Recognition, 36(8): 1847–1857, 2003.

[10] Kalyani Mali, Samayita Bhattacharya, Fingerprint Recognition Using Global and Local Structures, International Journal on Computer Science and Engineering, Vol. 3 No. 1 Jan 2011.Ratha, Nalini; Bolle, Ruud, "Automatic Fingerprint Recognition Systems", Springer, XVII, 458 p. 135.

[11] James L. Wayman (Editor), Anil K. Jain, "Biometric Systems", Springer, 2004.

[12] Bir Bhanu,Xuejun Tan, "Computational Algorithms for Fingerprint Recognition", Springer, 2004.