A Comparative Study of Fingerprint Thinning Algorithms

N.P. Khanyile
University of KwaZulu-Natal
Faculty of Engineering;
Council for Scientific and Industrial Research
Telephone: (012) 841–3372
Email: PKhanyile@csir.co.za

J.R. Tapamo
University of KwaZulu-Natal
Faculty of Engineering
Email: tapamoj@ukzn.ac.za

E. Dube
Council for Scientific and Industrial Research
Email: EDube@csir.co.za

Abstract—**Thinning plays a very important role in the preprocessing phase of automatic fingerprint recognition/identification systems. The performance of minutiae extraction relies heavily on the quality of skeletons used. A good fingerprint thinning algorithm can depress image noise and promote the robustness of the minutiae extraction algorithm which helps improve the overall performance of the system. Many thinning algorithms have been devised and applied to a wide range of applications including, Optical Character Recognition (OCR), biological cell structures and fingerprint patterns. With so many thinning algorithms available, deciding which one is appropriate for a particular application has become very difficult. In an effort to assist fingerprint biometrics developers choose an appropriate thinning algorithm, a study was taken to compare performance of four different thinning algorithms. These four algorithms are implemented and their performance evaluated and compared. The algorithms are compared in terms of the quality of the skeletons they produce (i.e. connectivity and spurious branches) as well as the time complexity associated with each algorithm. 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.**

I. Introduction

Thinning is a process of extracting a skeleton from an object in a digital image. A skeleton of an image can be thought of as a one-pixel thick line through the middle of an object which preserves the topology of that object. Thinning is a fundamental preprocessing step in many image processing and pattern recognition algorithms [1]. Thinned images (skeletons) are easier to process and they reduce processing time for the subsequent operations. Many thinning algorithms have been developed in the past three decades [1]-[10]. Two major approaches of thinning digital patterns can be categorized into iterative boundary removal algorithms and non-iterative distance transformation algorithms (Fig. 1).

Iterative boundary removal 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 [2].

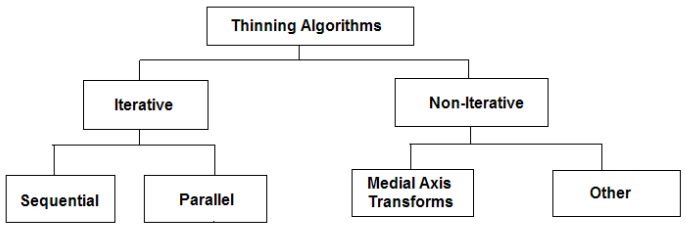


Fig. 1. Classification of thinning algorithms

In sequential algorithms, the pixels are examined for deletion in a fixed sequence in each iteration, and the deletion of pixel p in the n^{th} iteration depends on all operations performed so far, i.e. on the results of $(n-1)^{th}$ iteration; as well as on the current pixel in the n^{th} iteration. In a parallel algorithm,the deletion of pixels in the n^{th} iteration depends only in the results of the n^{th} iteration; therefore, all pixels are examined independently in the parallel manner in each iteration [3]. The behavior of a thinning algorithm is determined by its structuring element. Structuring elements are policies which define the situations at which foreground pixels will be set to background and hence deleted. Thinning is used in but not limited to applications that process handwritten and printed characters, fingerprints and palm prints, chromosomes and biological cell structures, and circuit diagrams.
Generally, fingerprint recognition systems work by matching minutiae extracted from probe data, to reference minutiae and it consists of the following stages: fingerprint acquisition, image pre-processing (fingerprint segmentation, enhancement, and orientation field estimation), fingerprint classification, minutiae detection and matching [4]. Fingerprint thinning is an important image enhancement processing step in an Automatic Fingerprint Identification System (AFIS). It plays an equally significant role with fingerprint classification and enhancement in practical AFIS. It can significantly improve the recognition performance of an AFIS [5].

Binary image thinning has been studied extensively in literature. While some researchers have developed sequential

algorithms [6]–[9], the main focus is in parallel thinning algorithms [2], [10]–[13], which are efficient and fast. Raju and Xu [14] in their study of parallel thinning algorithms compared Zhang-Suen, Guo-Hall and One Pass Thinning Algorithm (OPTA) for character recognition. They found that Guo-Hall outperformed the two other algorithms in terms of skeleton quality. While OPTA is faster than the other two algorithms, its skeleton quality is not as good compared to those of the other two algorithms. Gupta and Kaur [3] compared Zhang-Suen, Abdulla et al and a multipass iterative boundary removal algorithm based on [15], [16]. They found that the mutlipass algorithm produced better results than Zhang-Suen and Adbulla et al with regards to connectivity and spurious branches of numerical patterns.
This paper compares the application of four fingerprint thinning algorithms (Zhang-Suen's [17], Guo-Hall's [1], Abdulla et al's [10] and Hall's [18] algorithm) on fingerprints based on iterative boundary removal. These algorithms were implemented, their performance was evaluated and compared based on skeleton quality.

The rest of the paper is organized as follows: section 2 presents some preliminary concepts that will be used throughout the paper. Section 3 is devoted to the evaluation and comparison of results obtained and section 4 evaluates and compares their performance. Section 5 concludes the paper, and Section 6 discusses future works.

II. Preliminary Concepts

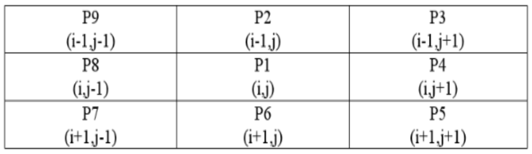
Given N, M and g three positive integers, a gray scale image could be defined as

$$I = \{(i, j, x_{ij}) | 0 \leq i \leq N-1 \wedge 0 \leq j \leq M-1 \wedge 0 \leq x_{ij} \leq g-1\} \quad (1)$$

where i, j, x_{ij} is a pixel of the image I, with (i, j) being the position of the pixel and x_{ij} being its gray value. More often the pixel (i, j, x_{ij}) is represented by (i, j), which we will adopt in this paper. A special case where $g = 2$ defines a binary image. An image is physically represented by a matrix, where entries are gray levels of the image. A binary image consists of only black and white pixel values. It typically represents an image in a more compact way, with obviously a lost of information. In this paper, a black pixel is represented by a pixel value of 1, and a white pixel is represented by a pixel value of 0.

A neighbor of a pixel is any pixel that is at a distance 1 in any direction from the pixel in question. Assume that the neighbors of the pixel (i, j) are $(i-1, j)$, $(i-1, j+1)$, $(i-1, j-1)$, $(i, j+1)$, $(i, j-1)$, $(i+1, j)$, $(i+1, j+1)$, and $(i+1, j-1)$, as shown in Fig. 2.

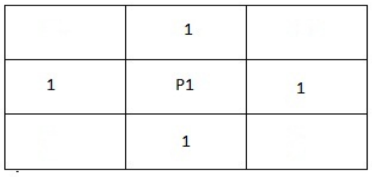
Connectivity is determined by the number of pixels connected to other pixels. By connected pixels we mean pixels that are neighbors. The algorithms discussed in this paper make use of 4-connectivity and 8-connectivity.



P9 (i-1,j-1)	P2 (i-1,j)	P3 (i-1,j+1)
P8 (i,j-1)	P1 (i,j)	P4 (i,j+1)
P7 (i+1,j-1)	P6 (i+1,j)	P5 (i+1,j+1)

Fig. 2. 3x3 window showing the 8-neighborhood of a pixel (i, j)

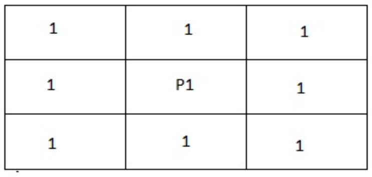
Pixels are 4-connected if they are connected to every horizontal and vertical neighbor. This relationship is better illustrated by a diagram, view Fig 3.



	1	
1	P1	1
	1	

Fig. 3. 4-connectivity

Pixels are 8-connected if they are connected to every horizontal, vertical and diagonal neighbor. For a better illustration of 8-connectivity, view Fig 4.



1	1	1
1	P1	1
1	1	1

Fig. 4. 8-connectivity

For an input binary image, let the object to be thinned be represented by a set S, and the background and holes in the image be represented by a set $\bar{S}$.

III. Thinning Algorithms

A. Zhang-Suen

In Zhang-Suen's algorithm, a binary image is represented as a matrix *IT* where each pixel value *IT(i,j)* is either 1 or 0. The object/pattern to be thinned consists of those pixels valued 1. Each stroke in the pattern is more than one pixel thick. Iterative transformations are applied to the matrix pixel by pixel, based on values of a small set of neighboring pixels shown in Fig. 2. This method removes all the contour points of the picture except those that belong to the skeleton [17]. Fully parallel thinning algorithms which are restricted to operators with 3x3 support have difficulty preserving the connectivity of an image [1]. To preserve connectivity most parallel algorithms divide each iteration two subiterations [1], [10], [17], [18].

1. Algorithm:

```
1: while points are deleted do
2:     for all pixels $p(i,j)$ do
3:         if (a) $2 \leq B(P_1) \leq 6$
               (b) $A(P_1) = 1$
               (c) Apply one of the following
                 1. $P_2 \times P_4 \times P_6 = 0$ in odd iterations
                 2. $P_2 \times P_4 \times P_8 = 0$ in even iterations
               (d) Apply one of the following
                 1. $P_4 \times P_6 \times P_8 = 0$ in odd iterations
                 2. $P_2 \times P_6 \times P_8 = 0$ in even iterations
           then
4:             Delete pixel $p(i,j)$
5:         end if
6:     end for
7: end while
```

where $A(P_1)$ is the number of 0 to 1 transitions in a clockwise direction from P_9 back to itself, and $B(P_1)$ (2) is the number of non-zero neighbors of P_1

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

If any of the conditions is not satisfied, P_1 is not deleted from the image.
Conditions 3(c) and 3(d) of the first subiteration remove only the south-east boundary points and north-west corner points which do not belong to an ideal skeleton. Condition 3(a) preservers the end-points of a skeleton line and condition 3(b) prevents deletion of points that lie between the end-point of a skeleton line [17]. This algorithm has been criticized by [19] for its failure to eliminate noise and preserve some of the structures such as patterns which have been reduced to 2x2 squares which are eventually completely eroded. The next algorithm discussed modifies this algorithm [17] to preserve connectivity in all images and produce thinner results.

B. Guo-Hall

In this algorithm [1] conductivities of $S\&\bar{S}$ (defined in section 2) are defined with 8- connectivity and 4-connectivity respectively in an effort to avoid connectivity paradoxes. Thinned objects should be a curve or a union of curves which are referred to as medial curves. A set of pixels G, is curve-like if most of the pixels of G have exactly two 8-neighbors in G and a few pixels in G are end-points (with one 8-neighbors in G) or branch points (more than two 8-neighbors in G) [1]. This algorithm modifies the two-subiteration algorithm presented in [17] and improved in [19] to preserve connectivity properties and produce thinner results. The algorithm uses operators with a 3x3 support as defined in Fig. 2. $C(P_1)$ is defined as the number of distinct 8-connected components of 1s in P_1's 8-neighborhood. $B(P_1)$ is defined as the number of 1s in P_1's neighborhood (1). Symbols $^-$, $\wedge$ and $\vee$ refer to logical complement, AND and OR, respectively; and the reserve + and • for arithmetic addition and multiplication respectively [1]. A variable $N(P_1)$, defined in (3) helps with the detection of end-points as well as achieving thinner results [1].

$$N(P_1) = MIN[N_1(P_1), N_2(P_1)] \tag{3}$$

where

$$N_1(P_1) = (P_9 \vee P_2) + (P_3 \vee P_4) + (P_5 \vee P_6) + (P_7 \vee P_8) \tag{4}$$

and

$$N_2(P_1) = (P_2 \vee P_3) + (P_4 \vee P_5) + (P_6 \vee P_7) + (P_8 \vee P_9) \tag{5}$$

$N_1(P_1)$ and $N_2(P_1)$ each break the ordered set of P_1's neighboring pixels into four pairs of adjoining pixels and count the number of pairs which contain one or two 1s [1].

2. Algorithm:

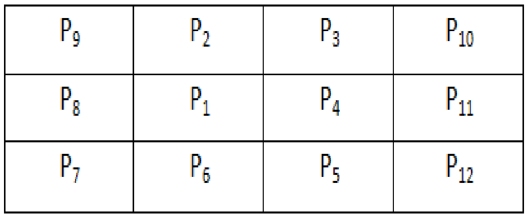
```
1: while points are deleted do
2:     for all pixels $p(i,j)$ do
3:         if (a) $C(P_1) = 1$;
               (b) $2 \leq N(P_1) \leq 3$;
               (c)Apply one of the following:
                 1. $(P_2 \vee P_3 \vee \bar{P_5}) \vee P_4 = 0$ in odd iterations
                 2. $(P_6 \vee P_7 \vee \bar{P_9}) \wedge P_8 = 0$ in even iterations
           then
4:             Delete pixel $p(i,j)$
5:         end if
6:     end for
7: end while
```

Condition 3(a) is necessary for preservation of local connectivity when P_1 is deleted and avoids deletion of pixels in the middle of medial curves. $C(P_1)$ allows some of the 1s in the middle of two-width diagonal lines to be deleted which in [17], [19] were preserved [1]. The variable $N(P_1)$ provides an end-point check replacing $B(P_1)$ which is used in [17], [19]. When $B(P_1) = 1$, P_1 is an obvious end-point and $N(P_1) = 1$. But when $B(P_1) = 2$, P_1 may or may not be an end-point. The definition of $N(P_1)$ allows end-points to be preserved while deleting many redundant pixel in the middle of the curve [1].

C. Abdulla et al

The first attempt to solve the problem of amplification of noise and the production of non-unitary skeletons incurred by [17], was made by Abdulla et al [10] for extracting skeletons of characters. Abdulla et al proposed a modified algorithm [10] based on [17] which achieves unitary skeletons. Deciding whether or not a pixel is skeletal depends on its 8-neighborhood. Thus, a window of 3x3 pixels shown in Fig. 2 is used. It is a two-subiteration algorithm. In the first iteration, the skeleton in scanned horizontally by the 3x4 pixel window shown in Fig. 5. Any two points which are horizontally adjacent to each other and horizontally isolated from other pixels are detected. With P_1 and P_4 representing these points, apply the following test to check whether one of them is redundant [10].
In the second iteration the skeleton is scanned vertically by the 4x3 pixel window shown in Fig. 6. Any two points which are vertically adjacent to each other and vertically isolated from

other points are deleted. With P_1 and P_6 representing these points, apply the following tests to check whether one of them is redundant [10].

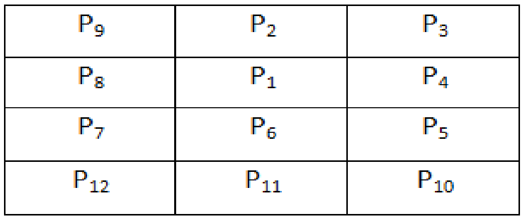


P_9	P_2	P_3	P_{10}
P_8	P_1	P_4	P_{11}
P_7	P_6	P_5	P_{12}

Fig. 5. 3x4 pixel window

3. Algorithm:

```
 1: while points are deleted do
 2:   for all pixels $p(i, j)$ do
 3:     Iteration 1 :
 4:     if (a) $\overline{SP}_{1.1} \wedge P6 = 1OR$
          (b) $\overline{SP}_{1.2} \wedge P2 = 1OR$
            (c) $[(P_2 \wedge \bar{P_3}) \vee (P_3) \wedge \bar{P_2} \vee \bar{P_9}] \wedge [(\bar{P_5}) \wedge P_6) \vee (P_5 \wedge \bar{P_6} \wedge P_7]$
      then
 5:       Delete pixel $P_1$
 6:       where $SP_{1.1} = P_3 \vee P_2 \vee P_9$, $SP_{1.2} = P_6 \vee P_5 \vee P_7$
          and $(^{-})$, $\vee$, $\wedge$ are complement, logical OR and
          logical AND respectively.
 7:     end if
 8:     if $P_1$ is not redundant
      then
 9:       if $(\bar{P_3} \wedge P_{10}) \vee (\bar{P_5} \wedge P12)$
          then
10:         Delete $P_4$
11:       end if
12:     end if
13:     Iteration 2 :
14:     if (a) $\overline{SP}_{2.1} \wedge P4 = 1OR$
          (b) $\overline{SP}_{2.2} \wedge P8 = 1OR$
          (c) $[(P_8 \wedge \bar{P_7}) \vee (P_7) \wedge \bar{P_8} \vee \bar{P_9}] \wedge [(\bar{P_4}) \wedge P_5) \vee (P_5 \wedge \bar{P_4} \wedge P_3]$
      then
15:       Delete pixel $P_1$
16:       where $SP_{2.1} = P_9 \vee P_8 \vee P_7$, $SP_{2.2} = P_3 \vee P_4 \vee P_5$
          and $(^{-})$, $\vee$, $\wedge$ are complement, logical OR and
          logical AND respectively.
17:     end if
18:     if $P_1$ is not redundant
      then
19:       if $(\bar{P_7} \wedge P_{12}) \vee (\bar{P_5} \wedge P10)$
          then
20:         Delete $P_6$
21:       end if
22:     end if
23:   end for
24: end while
```



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. 6. 4x3 pixel window

D. Hall

Fully parallel thinning algorithms can have difficulty preserving connectivity of an image and researchers have attempted to overcome this problem by partially serializing their algorithms by breaking a given iteration of their algorithm into distinct subiterations, or by partitioning the image space into distinct subfields [18]. The algorithm proposed in [18] 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 works as follows:

4. Algorithm:

```
1:  while points are deleted do
2:      for all pixels $p(i,j)$ do
3:          Determine the deletability of pixel $p(i,j)$
4:          if  (a) $1 < B(P_1) < 7$;
                 (b) $P_1$'s 8-neighborhood contains exactly one 4-
            connected component (connected set) of 1s.
             then
5:              $p(i,j)$ is deletable
6:          end if
7:      end for
8:      for all pixels $p(i,j)$ do
9:          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
10:             Do not delete pixel $p(i,j)$
11:         end if
12:     end for
13: end while
```

Condition 4(b) guarantees that local connectivity is not disrupted by removal of P_1 alone. Conditions 4(a) and 4(b) together guarantee that P_1 is 4-connected to $\bar{S}$ (defined in section 2) and condition 4(a) alone attempts to preserve end-points of thin lines. Condition 8(a) preserves pixels in a vertical two-width rectangle, condition 8(b) preserves pixels in a horizontal two-width rectangle , and 8(c) preserves a pixel in a 2x2 square [18].

IV. Performance Evaluation and Comparison

The four thinning algorithms were applied to thin five fingerprint images shown in Fig. 7. Input images are filtered binary images with the following sizes:

1) input image 1: 276x408 pixels
2) input image 2: 408x480 pixels
3) input image 3: 264x264 pixels
4) input image 4: 336x336 pixels
5) input image 5: 420x600 pixels

Performance is evaluated in terms of connectivity, spurious branches, convergence to unit width and data reduction efficiency/computational cost.

A. Connectivity

Connectivity preservation of a fingerprint pattern is crucial in AFISs, as disconnected patterns produce false minutiae.

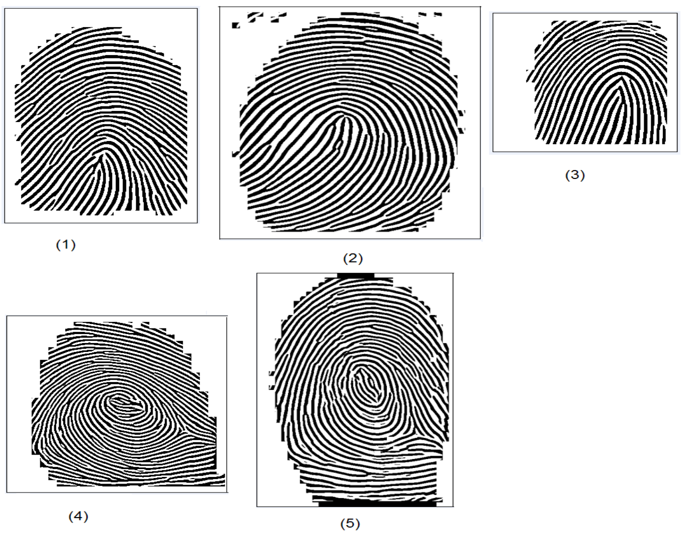


Fig. 7. Input images

Patterns often loose their topology features if they become disconnected.

B. Spurious Branches

Like disconnectivity, spurious branches lead to false minutiae. Although post processing operations for spurious branch removal exist, it is usually not the preferred approach since the extra post processing operations add extra complexity.

C. Convergence to unit width

A perfect skeleton must be unitary. If a skeleton S_m does not contain any one of the patterns Q^k (for k = 1 to 4) given in Fig 8 , it is unitary [20].
To measure width of the resulting skeleton, Jang and Chin [20] introduced a measure m_t to compute the width of the extracted skeleton. m_t is defined as:

$$m_t = 1 - \frac{Area[\cup_{1 \leq k \leq 4} S_m Q^k]}{Area[S_m]} \tag{6}$$

where Area[•] is the operation that counts the number of 1-pixels. This measures a non-negative value less than or equal to 1, with $m_t = 1$ if S_m is a perfect unit-width skeleton [20].

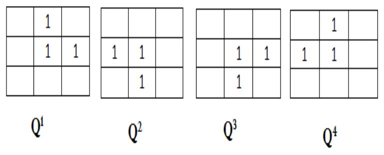


Fig. 8. Templates used to examine the width of the converged skeleton

D. Computational Cost

A fingerprint recognition system can only be realized when it is fast and efficient. Thorough and efficient preprocessing techniques lead to more efficient and accurate systems because they improve the minutiae extraction algorithm which

minimizes the number of false minutiae points, thus improving quality and speed of matching.
A measure to evaluate both the data reduction efficiency and the computational cost was defined by Jang and Chin as

$$m_d = min[1, \frac{Area[S] - Area[S_m]}{n \times Area[S]}] \quad (7)$$

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; a large value indicates high efficiency.

E. Experimental Results and Comparison

Fig 10-14 show skeletons of the 5 test images resulting from the 4 thinning algorithms and Table 1 shows the results of the two measures m_t & m_d (Eq 6 & Eq 7 respectively) for each skeleton.

TABLE I
EXPERIMENTAL RESULTS: m_t & m_d READINGS

Image	Algorithm	Quality Measure	
		m_t	m_d
1	(a) Abdulla et.al	0.996	0.117
	(b) Guo-Hall	0.998	0.062
	(c) Hall	0.991	0.083
	(d) Zhang-Suen	0.698	0.129
2	(a) Abdulla et.al	0.974	0.120
	(b) Guo-Hall	0.997	0.065
	(c) Hall	0.988	0.085
	(d) Zhang-Suen	0.790	0.137
3	(a) Abdulla et.al	0.997	0.122
	(b) Guo-Hall	0.998	0.061
	(c) Hall	0.999	0.084
	(d) Zhang-Suen	0.864	0.130
4	(a) Abdulla et.al	0.978	0.105
	(b) Guo-Hall	0.993	0.056
	(c) Hall	0.993	0.079
	(d) Zhang-Suen	0.747	0.115
5	(a) Abdulla et.al	0.985	0.118
	(b) Guo-Hall	0.997	0.064
	(c) Hall	0.993	0.085
	(d) Zhang-Suen	0.695	0.134

The table has been graphed below (Fig 9) for a better illustration.

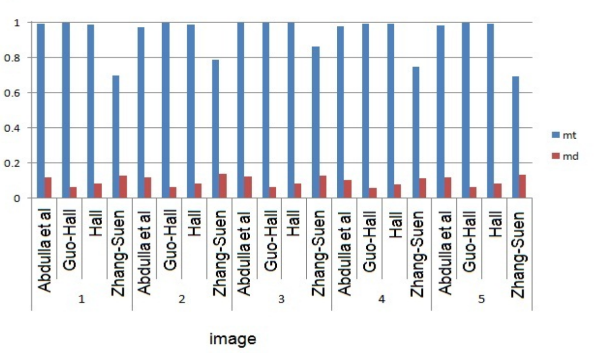


Fig. 9. m_t and m_d readings

Experimental results show that Zhang-Sue's algorithm effectively thins the image, but creates undesirable artifacts. The resulting skeletons (Figures 11(d), 12(d), 13(d), 14(d) and 15(d)) are non-unitary, giving an average value of $m_t = 0.698$ (lowest amongst the four algorithms and some images have gaps between edges, in particular there is an entire ridge missing from the skeleton in Fig. 11 (d). This is due to the fact that end-points are detected by $A(P_1) = 1$. This condition works in many cases, but not for 2-pixel thick diagonal lines because in such cases $A(P_1) = 2$. In this case end-points are deleted as they satisfy all the the deletion conditions in the algorithm. One way around this was discussed by Raju and Xu [14] on their study of parallel thinning algorithms. In order to obtain 1-pixel thick skeleton and avoid deleting diagonal lines, some additional conditions are added to the Zhang-Suen algorithm.
In odd iterations, when $A(P_1) = 2$, the following conditions are checked:
1) $P4 \times P6 = 1$ and $P9 = 0$ or
2) $P4 \times P2 = 1$ 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') $P2 \times P8 = 1$ and $P5 = 0$ or
2') $P6 \times P8 = 1$ and $\bar{P}3 \times \bar{P}4 \times \bar{P}7 = 1$

The resulting skeleton is not perfect, but it is significantly better than the skeleton produced by the original algorithm. This is shown by the minutiae points detected (Fig. 10) after introducing the conditions mentioned above. It can be observed from Fig. 10 that after adding the extra conditions, the quality of the skeleton improves. Zhang-Suen's algorithm is the most efficient of the four algorithms, giving an average of $m_d = 0.235$. The algorithm fairly maintains connectivity and does not produce spurious branches.

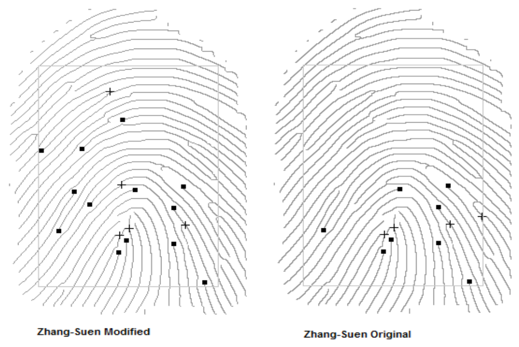


Fig. 10. Skeleton produce by Zhand-Suen's algorithm compared to modified version

In [10] Abdulla et al stated that their algorithm produces unitary skeletons and in that manner it is not affected by noise, nor does it amplify it. Although the algorithm works on characters [10], it does not work so well for fingerprint patterns.

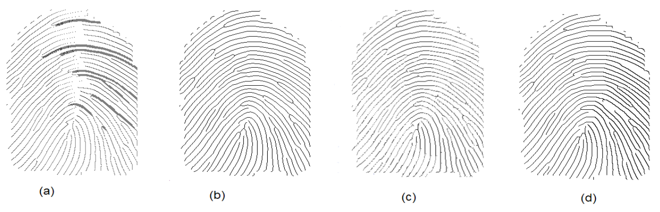


Fig. 11. (a) Skeleton produced by Adbulla et al's algorithm from image 1; (b) Skeleton produced by Guo-Hall's algorithm from image 1; (c) Skeleton produced by Hall's algorithm from image 1 ; (d) Skeleton produced by Zhang-Suen's algorithm from image 1

Figures 11(a), 12(a), 13(a), 14(a) and 15(a) show that extra noise has been added to the skeleton. Even with its high value of $m_t = 0.986$ on average and a data reduction efficiency value of $m_d = 0.116$, Abdulla's algorithm fails to maintain connectivity and produces clusters of spurious branches.

Like Zhang-Sue's, Hall's algorithm [18] effectively thins the fingerprint pattern and unlike [17] it preservers all connectivity and does not leave gaps. Figures 11(c), 12(c), 13(c), 14(c) and 15(c) show the skeletons produced by this algorithm. The average one-pixel width measure is $m_t = 0.993$, which means the algorithm produces fairly unitary skeletons. The major concern with this algorithm [18] for fingerprint patterns is the amount of spurious branches in the resulting skeletons. Fingerprint recognition relies heavily on minutiae extraction and spurious skeletons often lead to false minutiae detection and hence depress the performance of the system.

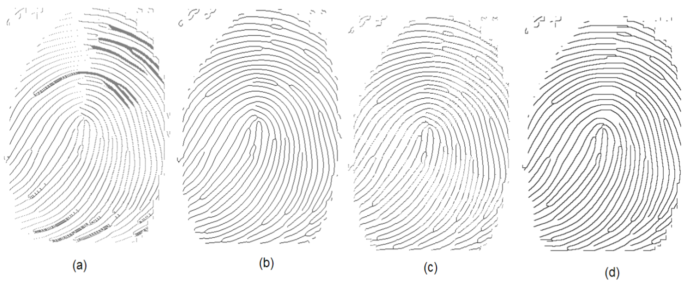


Fig. 12. (a) Skeleton produced by Adbulla et al's algorithm from image 2 ; (b) Skeleton produced by Guo-Hall's algorithm from image 2; (c) Skeleton produced by Hall's algorithm from image 2 ; (d) Skeleton produced by Zhang-Suen's algorithm from image 2

However, post processing algorithms can be applied to eliminate spurious branches and smooth skeletons [21]–[23]. This approach is not recommended because Hall's algorithm is not efficient, giving an average of $m_d = 0.083$, adding extra operations required for removing spurious branches would worsen the efficiency of this algorithm which might not be ideal for real time applications.

Lastly [1] gives the best results. The skeletons are non-spurious and preserve connectivity (see Figures 11(b), 12(b), 13(b), 14(b) and 15(b)), thin and no ridges are missing. Minutiae features are clear and there is no apparent noise. Guo-Hall is better than Zhang-Suen at detecting end-points. $N(P_1)$ is able to detect end-points whether or not they have one or two 8-neighbors. 2-pixel thick diagonal lines are not deleted. It produces fairly unitary skeletons, with $m_t = 0.997$ on average. The only concern with this algorithm is its data reduction efficiency. The algorithm gave a value of $m_d = 0.062$, the lowest amongst the four algorithms.

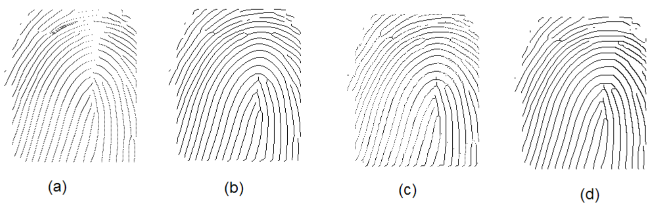


Fig. 13. (a) Skeleton produced by Adbulla et al's algorithm from image 3; (b) Skeleton produced by Guo-Hall's algorithm from image 3; (c) Skeleton produced by Hall's algorithm from image 3 ; (d) Skeleton produced by Zhang-Suen's algorithm from image 3

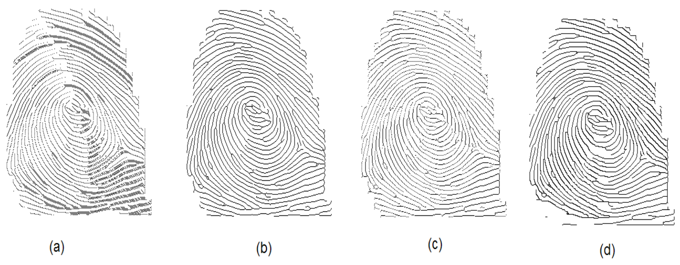


Fig. 14. (a) Skeleton produced by Adbulla et al's algorithm from image 4; (b) Skeleton produced by Guo-Hall's algorithm from image 4 ; (c) Skeleton produced by Hall's algorithm from image 4 ; (d) Skeleton produced by Zhang-Suen's algorithm from image 4

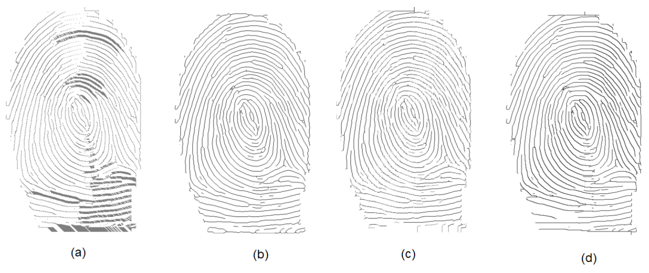


Fig. 15. (a) Skeleton produced by Adbulla et al's algorithm from image 5 ; (b) Skeleton produced by Guo-Hall's algorithm from image 5 ; (c) Skeleton produced by Hall's algorithm from image 5 ; (d) Skeleton produced by Zhang-Suen's algorithm from image 5

V. Conclusion

There are many thinning algorithms available, and all have their own advantages and disadvantages. The choice of the thinning algorithm should depend on the application, as not all thinning algorithms will be suitable for a certain application. As with most computing systems, a trade off usually has to be made between accuracy and execution time. Fast parallel thinning algorithms often suffer from loss of connectivity, as shown earlier with Zhang-Suen's algorithm. The choice should depend on the nature of the application at hand. For example an application for an airport boarding gate, would have to trade accuracy for execution time, whereas a high security location or a transaction between business needs to ensure high accuracy, and can trade the execution time. This paper has shown Guo-Hall's thinning algorithm works best for fingerprint pattern. Even with the short-comings Zhang-Suen's algorithm is still the most used thinning algorithm in literature for pattern recognition applications. Most algorithms modify Zhang-Suen's algorithm either to address the connectivity issue or to adapt it for specific application.

VI. Future Work

Zhang-Suen, Gua-Hall and Hall algorithms are promising and we will investigate them further. To improve efficiency, new distributed algorithms based on these three will be developed. For Zhang-Suen's algorithm, the most important step will be to try and eliminate the artifacts, and for Hall's algorithm, the last step will be to apply post-processing techniques to remove spurious branches after distributing the processing.

References

[1] Z. Guo and R. Hall, "Parallel thinning with two-subiteration algorithms," *Communications of the ACM*, vol. 32, pp. 359–373, Mar 1989.

[2] N. Han, C. La, and P. Rhee, "An effecient fully parallel thinning algorithm," *Proceedings of the Fourth International IEEE Conference on Document Analysis and Recognition*, vol. 01, pp. 137–141, 1997.

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

[4] A. Saleh, A. Eldin, and A. Wahdan, "A modified thinning algorithm for fingerprint identification systems," *International Conference on Computer Engineering & Systems*, pp. 371–376, Dec 2009.

[5] L. Ji, Z. Yi, L. Shang, and X. Pu, "Binary fingerprint image thinning using template-based pcnns," *IEEE transactions on systems, man, and cybernetics -part B: Cybernetics*, vol. 37, pp. 1407–1413, Oct 2007.

[6] C. Arcelli and G. Baja, "A thinning algorithm based on prominence detection," *Pattern Recognition*, pp. 225–235, 1981.

[7] P. Kardos, G. Nemeth, and K. Palagyi, "An order-independant sequential thinning algrithm," pp. 162–175, 2009.

[8] H. Pu, J. Chen, and Y. Zhang, "Fingerprint thinning algorithm based onn mathematical morphology," *The Eighth International Journal on Eletriconic Measurements and Instruments*, vol. 01, pp. 618–621, 2007.

[9] R. Zhou, C. Quek, and G. Ng, "A novel single-pass thinning algorithm and an effective set of performance criteria," *Pattern Recogntion Letters 16*, pp. 1267–1275, 1995.

[10] W. Abdulla, A. Saleh, and A. Morad, "A preprocessing algorithm for hand-written character recognition," *Pattern Recognition Letters 7*, pp. 13–18, 1988.

[11] L. Haung, G. Wan, and C. Liu, "An improved parallel thinning algorithm," *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 01, pp. 780–783, Dec 2003.

[12] H. Lingga, S. Sudiro, and E. Wibowo, "Hardware implementation of fingerprint image thinning algorithm in fpga device," *IEEE International Conference on Networking and Information Technology*, pp. 187–191, 2010.

[13] H. Xu, Y. Qu, and F. Zhao, "Fpga based parallel thinning for binary fingerprint image," *IEEE Chinese Conference on Pattern Recognition*, pp. 1–4, Nov 2009.

[14] 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.

[15] M. Ahmed and R. Ward, "A rotational invariant rule-based skeletonization algorithm for character recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1672–1678, Jan 2003.

[16] M. Girgis, A. Sewisy, and R. Mansour, "Employing generic algorithms for precise fingerprint matching based on line extraction," *GVIP Journal*, vol. 07, pp. 51–59, 2007.

[17] T. Zhang and C. Suen, "A fast parallel algorithm for thinning digital patterns," *Communications of the ACM*, vol. 27, pp. 236–239, Mar 1984.

[18] R. Hall, "Fast parallel thinning algorithms: Parallel speed and connectivity preservation," *Communications of the ACM*, vol. 32, pp. 124–129, Jan 1989.

[19] H. L$\ddot{u}$ and P. Wang, "A comment on 'a fast parallel algorithm for thinning digital patterns'," *Communications of the ACM*, vol. 29, pp. 239–242, Mar 1986.

[20] 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.

[21] E. Virginia, "Fingerprint thinning algorithm," *IEEE AES Systems Magazine*, pp. 28–30, Sep 2003.

[22] E. Nel, J. Preez, and B. Herbst, *A Pseudo-Skeletonization Algorithm for Static Handwritten Scripts*. Springer-Verlag, 2009, vol. 01.

[23] D. Goldman and N. Bourbakis, "Well-shaped skeletons and fast computation of the (3,4) distance transform," *Journal of Electric Imaging*, vol. 11, pp. 404–413, Jul 2002.