Pattern Analysis and Applications
https://doi.org/10.1007/s10044-021-01039-y

SHORT PAPER

A novel fully parallel skeletonization algorithm

Jun Ma[1] · Xunhuan Ren[1] · Viktar Yurevich Tsviatkou[1] · Valery Kanstantinavich Kanapelka[1]

Received: 30 September 2020 / Accepted: 23 September 2021

Abstract
A Skeleton that is extracted by a skeletonization algorithm from a binary image is useful for object description, matching, recognition and compression. The parallel thinning algorithm, one of the skeletonization algorithms is well known to have computational effciency. The main contribution of this paper is that we proposed a novel fully parallel thinning algorithm based on a comprehensive investigation of the well-known Zhang-Suen (ZS)-series algorithms and the one-pass thinning algorithm (OPTA)-series algorithms, which not only has good performance in terms of (8,4) connectivity preservation and single-pixel thickness, but also has the following qualities: it is more robust to the boundary noise than the OPTA-series algorithms and it is faster than the ZS-series algorithms in terms of thinning speed, as confirmed by the experiments presented in this paper.

Keywords Skeletonization · Noise immunity · Fully parallel · Connectivity · Thinning rate · Thinning speed

1 Introduction

Skeletonization is an elementary processing step in pattern recognition. The result of the skeletonization of a binary image is called the skeleton or medial axis, and it preserves the topological and geometric properties of the original image. The foundation of skeletonization was originally introduced by Blum [1] through an analogy with grassfires. An image object can be seen as a field of dry grass; fires will start simultaneously propagating inside the object at a uniform velocity, and the fires will quench at the point where they meet. These quenching points constitute the skeleton of the object [2–4]. Skeletonization methods can be divided into three major approaches [5, 6]: geometric, curve propagation and digital approaches.

Methods based on a geometric approach compute the skeleton of an object by focusing on the geometric properties of Blum's medial axis. Under this category, the method based on Voronoi diagrams [7–9] is popular. The drawback of the Voronoi method is that it produces a large number of unwanted branches.

Methods based on curve propagation approaches [2, 3], in which the evolution process is modeled by partial differential equations, simulate the process of generating Blum's skeleton. The flaw of continuous curve propagation approaches is that the result of such methods may not be topologically connected [10].

The digital approach, which is also called the thinning algorithm, uses iterative removal on a digital grid under predefined geometric and topological rules to simulate the propagation process of Blum's grassfire [5]. Digital approaches can be further divided into full kernel-based iterative algorithms, iterative boundary peeling algorithms under geometric and topological rules and distance transform algorithms [11]. The main merit of the distance transform algorithm is that it does not require repetitive image scans, which leads to high computational efficiency. However, this approach may be hard to parallelize.

Skeletonization algorithms may also be classified into parallel and sequential algorithms [12] in terms of computational efficiency. The computational efficiency of the iterative digital approach can be increased by using the technique

✉ Xunhuan Ren
renxunhuan@bsuir.by

Jun Ma
majun1313@hotmail.com

Viktar Yurevich Tsviatkou
vtsvet@bsuir.by

Valery Kanstantinavich Kanapelka
volos@bsuir.by

[1] Belarusian State University of Informatics and Radioelectronics, 220013 Minsk, Belarus

Published online: 12 November 2021

of parallelization, and such iterative algorithms are referred to as parallel thinning algorithms. Parallel thinning algorithms can be divided into three major partitions according to the different parallelization strategies they use: fully parallel algorithms [13–20], sub-iteration parallel algorithms [21–33] and subfield parallel algorithms [34–36]. The fully parallel algorithm applies the same deletion criteria at every iteration. The sub-iterative parallel algorithm alternately uses a small set of removal conditions rather than using the same removal condition. Subfield parallel algorithms partition the image into subsets. In each iteration, a parallel deletion condition is applied to only one subfield of the partition.

A good parallel thinning algorithm should have the following important properties [22, 42]: the skeleton results must be one-pixel thick; the skeleton results must approximate the medial axis of the original image; the thin curves and endpoints of the original image must be preserved; the connectivity of the original image must be preserved; the parallel speed should be as fast as possible; slight noise appearing near the boundary should not greatly affect the resulting skeleton.

The ZS algorithm [21] is one of the most widely used sub-iterative parallel algorithms and is also considered a directional method in some studies. The ZS algorithm is well known for its ease of implementation since it uses only simple logical conditions. Additionally, the ZS algorithm performs very well with respect to both insensitivity to contour noise and connectivity. However, the ZS algorithm also has some potential problems, which are described in later sections. Many algorithms have been proposed to address the problems remaining in the ZS algorithm, such as those described in [22–33]. The OPTA algorithm [13] is a fully parallel thinning algorithm that accelerates the thinning speed by reducing the number of iterations but suffers from the influence of edge noise. As a result, many modified versions have been proposed, such as those proposed in [14–20].

In this paper, a novel fully parallel algorithm is proposed that combines robustness (one of the merits of the ZS series) and high speed (one of the merits of the OPTA series). In addition, the skeleton produced by this algorithm is exactly one-pixel thick and provides a good visual effect.

The remainder of this paper is organized as follows: Sect. 2 introduces some of the basic notations used in this paper. Section 3 reviews four recent ZS-series algorithms and three recent OPTA-series algorithms. The proposed algorithm is presented in Sect. 4, and the proof of topology preservation of the proposed algorithm is given in Sect. 5. Section 6 introduces some measures used to evaluate the algorithms. The three conducted experiments are described in Sect. 7, and the results of the different algorithms are compared. Finally, Sect. 8 concludes this article.

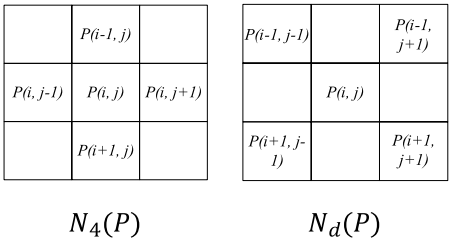


Fig. 1 The definition of $N_4(P)$ and $N_d(P)$

2 Basic notations

In this paper, a *pixel* (point) is a closed square in the Euclidean plane, which has four edges and four vertices. The edges of one pixel are parallel to the coordinate axis, whose lengths are 1. If two pixels are distinct and they share at least one vertex, they are said to be *adjacent*. One pixel is said to be a *neighbor* of another pixel if they are adjacent. For a pixel in a 2D image, there are most 8 neighbors. The *4-neighborhood* of pixel $P(i,j)$, whose row coordinate is i and column coordinate is j, comprises the 4 neighboring pixels that share a common edge with pixel P. This neighborhood set is denoted by $N_4(P)$. The *D-neighborhood* of pixel P comprises four diagonal neighbors that share a common vertex with it and are denoted by $N_d(P)$. The points of $N_4(P)$ and $N_d(P)$ (as shown in Fig. 1) are determined based on the concept of the *8-neighborhood* of pixel P, denoted as $N_8(P)$.

In a binary image, the value of a pixel can only be 0 or 1; 0-valued pixels are called *0's* and constitute the *background*. Similarly, 1-valued pixels are called *1's* and constitute the *foreground*.

A foreground point can be considered an *edge pixel* when at least one of its 8-neighborhood belongs to the background. An *end point* refers to a foreground pixel that has only one foreground neighbor in its 8-neighborhood.

3 Literature review

3.1 ZS-series algorithms

3.1.1 The ZS algorithm

In 1984, Zhang and Suen proposed the ZS algorithm [21], which operates on an 8-neighborhood, as shown in Fig. 2.

The ZS algorithm has two sub-iterations: the first sub-iteration removes southeast edge pixels and northwest corner points, while the second sub-iteration removes northwest edge pixels and southeast corner points.

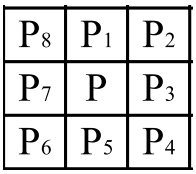


Fig. 2 The 8-neighborhood of the ZS algorithm

During the first sub-iteration, a pixel P will be deleted from the original digital image if it meets the following conditions:

$$2 \le B(P) \le 6 \tag{1}$$

$$A(P) = 1 \tag{2}$$

$$P_1 \times P_3 \times P_5 = 0 \tag{3}$$

$$P_3 \times P_5 \times P_7 = 0 \tag{4}$$

In the second sub-iteration, the last two conditions are modified as follows:

$$P_1 \times P_3 \times P_7 = 0 \tag{5}$$

$$P_1 \times P_5 \times P_7 = 0 \tag{6}$$

$A(P)$ is the number of pairs (0,1) in the ordered set $P_1, P_2, \cdots, P_8$ that are the eight neighbors of P, and $B(P)$ denotes the sum of foreground pixels in $N_8(P)$.

$$A(P) = \sum_{a=1}^{4} \left(\overline{P_{2a-1}} \times P_{2a} + \overline{P_{2a}} \times P_{(2a+1) mod 8} \right) \tag{7}$$

$$\overline{\mathrm{P_a}} = 1 - \mathrm{P_a} \tag{8}$$

$$\mathrm{B(P)} = \sum_{a=1}^{8} \mathrm{P_a} \tag{9}$$

Due to the simple conditions mentioned above and the separate processing of boundaries with different directions in each sub-iteration, the ZS algorithm is both fast and insensitive to boundary noise. However, it has three main drawbacks [22]:

(1) The complete disappearance of 2×2 square patterns prevents the algorithm from extracting some small image details.
(2) The output skeleton does not always have a single-pixel thickness, increasing the difficulty of directly using the results in some applications.
(3) The ZS algorithm suffers from excessive erosion of diagonal lines.

3.1.2 The Tarabek algorithm

To overcome the drawbacks of the ZS algorithm, the Tarabek algorithm was proposed in 2012 [24]. This algorithm not only avoids removing 2×2 squares and diagonal line patterns by adding several conditions but also generates a comprehensively single-pixel-thick skeleton by introducing a postprocessing step. The runtime of this method is approximately the same as that of the ZS algorithm. The author asserted that the Tarabek algorithm could potentially improve the accuracy of shape analysis.

3.1.3 The RIEPTA algorithm

The RIEPTA algorithm [32] was proposed by Rui Liu and Xiaoyu Zhang to address the defects of redundant branches caused by non-smooth contours. It eliminates the restrictions of the iterative scaling mechanism, avoids the branching phenomenon effect on the pixel points of off-smooth contours and ensures efficient refinement. RIEPTA has good robustness against noise, redundant branches and thin lines, it guarantees efficient refinement, and it has good robustness and practical application value. In addition, it solves the 3×4 rectangular erosion problem of the ZS algorithm. The RIEPTA algorithm yields a line with two pixels rather than the one-pixel produced by the ZS algorithm to represent that the shape is rectangular.

3.1.4 The MZS algorithm

Lynda Ben Boundaoud, Basek Solaiman and Abdelkamel Tari proposed a modified ZS thinning algorithm using a hybrid approach (called the MZS algorithm) that combines the directional approach used by the ZS algorithm and the subfield approach to avoid the ZS algorithm's problems of excessive erosion and at the same time ensure a one-pixel skeleton thickness [22]. The MZS algorithm adds the condition that the parity of the sum of the coordinates of each pixel needs to be checked in each sub-iteration of the ZS algorithm. Furthermore, it integrates an additional condition into the second sub-iteration to preserve the tiny square patterns, minimizing the appearance of spurious branches and avoiding the excessive erosion of slanting lines.

3.2 OPTA-series algorithms

3.2.1 The OPTA algorithm

To reduce the time consumed by the multi-pass methods, a one-pass parallel thinning algorithm called the OPTA algorithm was proposed by Roland T. Chin et al. This algorithm uses a set of 3×3 thinning templates (as shown in Fig. 3). Two other restoring templates (whose sizes are 4×1 and

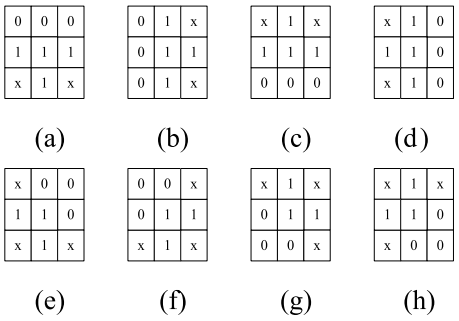


Fig. 3 Thinning templates

1 × 4) are applied to address the breakage and disappearance of horizontal and vertical limbs with double-pixel thicknesses [13] (as shown in Fig. 4), where the 'x' symbol denotes ignorable pixels, whose value can be either 0 or 1. Additionally, eight trimming templates (as shown in Fig. 4) are adopted to eliminate noise in each iteration to avoid producing spurious branches. However, this method can cause another problem in which the end points of object patterns are not preserved. In particular, in some extreme situations, if a limb with a one-pixel width is thinned so the object shape degenerates into a single pixel.

3.2.2 The OPTA4 algorithm

A new one-pass parallel thinning algorithm for binary images called OPTA4 was proposed by Rei-Yao Wu et al. [15]. They elaborated a new set of matching templates derived from the idea of asymmetry that eliminated the need

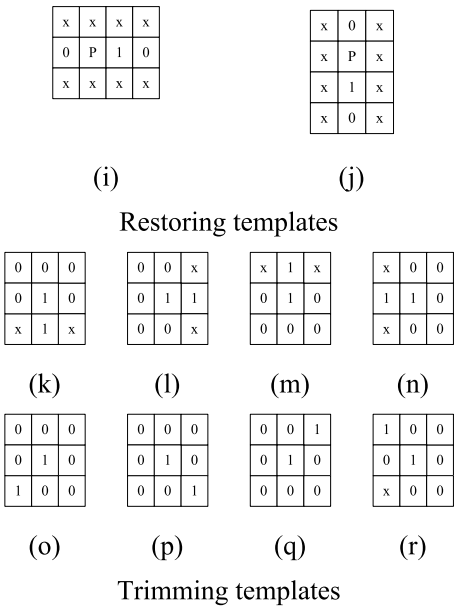


Fig. 4 Restoring and trimming templates

to distinguish between restoring templates, thinning templates and trimming templates. When an object is thinned, the pixels on one side of the object are deleted according to preset preferences; however, the pixels on the other side are preserved. Wu and Tsai's algorithm is both unique and fast; it is one of the fastest parallel algorithms. However, the resulting skeletons do not always retain the structures of the initial objects in some of the produced patterns (e.g., they completely remove 2-by-2 squares).

3.2.3 The FCTA algorithm

The FCTA algorithm proposed by Lei Guan et al. [19] is an improved text image thinning algorithm based on a study of the classical OPTA algorithm; it addresses the defects of breakdowns and thinning incompleteness. The FCTA algorithm deploys eight thinning templates and four restoring templates. All these templates work on a 14-pixel neighborhood field. The authors reported that their method has a very fast thinning speed compared with other algorithms and that it ensures both the feature integrity and the connectivity of the original image.

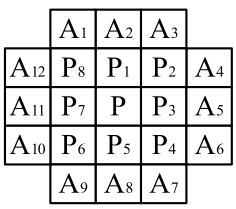
4 The proposed algorithm

A novel fully parallel algorithm is proposed in this section that combines the advantages of the ZS-series and OPTA-series algorithms to address the previously mentioned drawbacks. The proposed algorithm uses a 20-pixel neighborhood that extends the standard 3 × 3 neighborhood, as shown in Fig. 5.

The new algorithm includes two procedures: a basic thinning procedure and a postprocessing procedure. The thinning procedure applies 13 templates in total, of which two are restoring templates, one is a compulsory deletion template (shown in Fig. 6), and the others are extra deletion templates (shown in Fig. 7).

In the thinning process of the proposed algorithm, our restoring templates, which are modified based on templates (i) and (j) of the OPTA algorithm, and the compulsory deletion template work together in the case that the first two conditions (see formulas (1) and (2)) of the ZS algorithm hold. Therefore, in our algorithm, $A(P)$ and $B(P)$ also need to be calculated according to formulas (7) and (9), respectively.

Fig. 5 The 20-pixel neighborhood used in the proposed algorithm



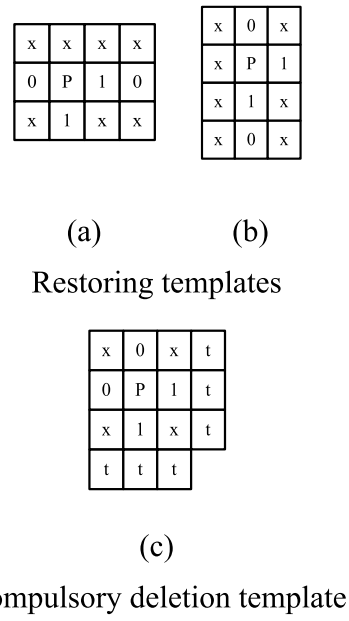


Fig. 6 Restoring and compulsory deletion templates

The restoring templates are designed to prevent the complete removal of patterns with a small fraction. This works well in the OPTA algorithm; however, the drawback of deploying restoring templates is that they may retain more pixels than we need. As a result, a compulsory deletion template is necessary to delete excessive pixels that may be retained by the restoring templates. Thus, here, the priority of a compulsory template is higher than that of a restoring template. A candidate foreground pixel is removed when its neighbors do not match any of the restoring templates or match one restoring template but also match the compulsory deletion template.

By modifying formulas (1) and (2), two more conditions are given, as follows:

$$4 \leq B(P) \leq 5 \tag{10}$$

$$A(P) = 2 \tag{11}$$

The extra deletion templates will be used to delete foreground pixels when these two conditions hold. In fact, formulas (10) and (11) mainly work as filters, which is helpful in selecting foreground pixels that may succeed in matching the extra deletion templates. This can effectively avoid comparing the neighborhood of every pixel that does not satisfy formulas (1) and (2) with 10 extra deletion templates, which may accelerate the thinning speed.

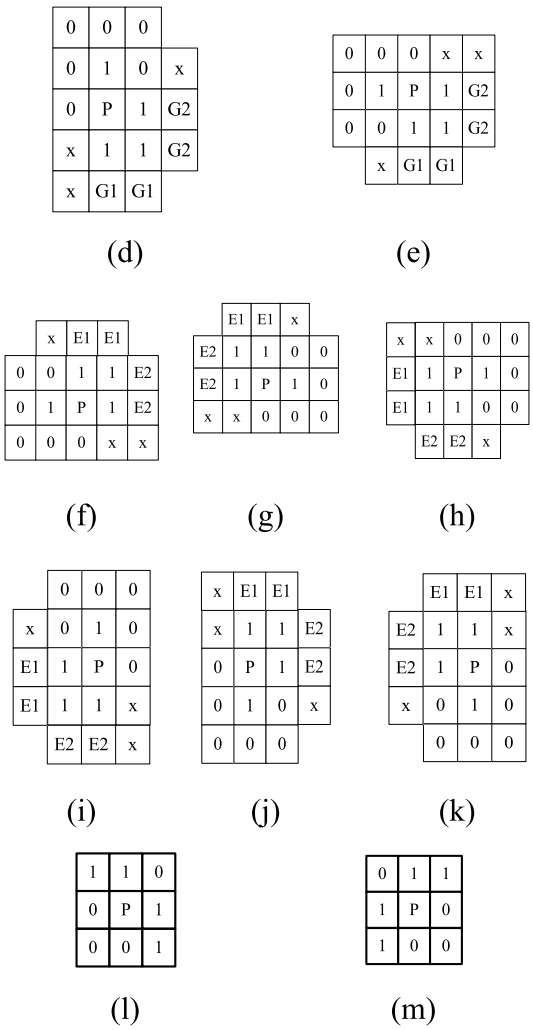


Fig. 7 Extra deletion templates

The extra deletion templates are supplementary templates, which are used to delete some pixels P that temporarily fail to satisfy formulas (1) and (2). These pixels have the common property that their parallel deletion does not break the original topology. They can constitute an 8-deletable set with adjacent foreground pixels that meet the deletion condition (the concept of the 8-deletable set will be given later).

In Fig. 7., the symbols '0', '1', 'P' and 'x' in these templates denote a background pixel, a foreground pixel, the currently tested pixel and an ignorable pixel, respectively, whereas 't' denotes that at least two of the pixels represented by the set of symbols should be a foreground pixel. The symbols 'E1', 'E2', 'G1' and 'G2' are defined as special symbols that should satisfy the following rule: the values of

two arbitrary pixels in a given template marked as identical special symbols should be equal. The difference between 'E1' and 'E2' and 'G1' and 'G2' is mainly that the value of a pixel marked as 'E1' and the value of a pixel marked as 'E2' are independent; however, the sum of the values of a pixel marked as 'G1' and another pixel marked as 'G2' should be greater than 1.

A postprocessing procedure is used after the thinning procedure. Four conditions are applied to examine each pixel to ensure a single-pixel skeleton thickness. A foreground pixel is deleted when its neighbors satisfy one of the following conditions:

$$P_1 \times P_3 \times \overline{P_6} = 1 \tag{12}$$

$$P_3 \times P_5 \times \overline{P_8} = 1 \tag{13}$$

$$P_5 \times P_7 \times \overline{P_2} = 1 \tag{14}$$

$$P_7 \times P_1 \times \overline{P_4} = 1 \tag{15}$$

The proposed algorithm is described in Algorithm 1.

Algorithm 1 Thinning Algorithm

Input: A binary image I with two layers zero padding
Output: Skeleton S

$Row, Col = Size(I)$
$DeleteMatrix = Zeros(Row, Col)$
// Thinning Processing Procedure
repeat
  $Flag = False$
  for $(r = 3; r \leq Row - 2; r++)$ **do**
    for $(c = 3; c \leq Col - 2; c++)$ **do**
      $P = I[r][c]$
      if (P is a foreground pixel) **then**
        Compute $A(P)$ and $B(P)$ according to formula (7) and (9)
        if $(A(P) == 1)$ AND $(2 \leq B(P) \leq 6)$ **then**
          if (P's neighbors not match any one of restoring templates) OR (P's neighbors do match compulsory deletion template) **then**
            $DeleteMatrix[r][c] = 1$
            $Flag = True$
          end if
        else
          if $(A(P) == 2)$ AND $(4 \leq B(P) \leq 5)$ **then**
            if (P's neighbors match one of Extra deletion templates) **then**
              $DeleteMatrix[r][c] = 1$
              $Flag = True$
            end if
          end if
        end if
      end if
    end for
  end for
  $I = I - DeleteMatrix$
  Set DeleteMatrix to Zero Matrix
until $(Flag == False)$
$S = I$
// Post Processing Procedure
for $(r = 3; r \leq Row - 2; r++)$ **do**
  for $(c = 3; c \leq Col - 2; c++)$ **do**
    $P = S[r][c]$
    if (Any one of the formula (12) to (15) holds in P's 8-neighborhood) **then**
      $S[r][c] = 0$
    end if
  end for
end for
return S;

5 Proof of topology preservation

5.1 Ronse's conditions for topology preservation

It is more convenient to conduct the proof of topology preservation for the proposed algorithm by introducing some terms.

Terms such as *i-path*, *i-connected* and *i-component* are used in same sense as in [37] for $i = 4$ and 8. *C(P)* is defined as the number of distinct 8-components of 1's in $N_8(P)$ of pixel P. A foreground pixel is defined as a *border* if its $N_4(P)$ contains a 0. We say that a foreground pixel P is *8-simple* if $C(P) = 1$ and P is a border.

In terms of the topology (connectivity), the foreground and background are understood to have 8-connectivity and 4-connectivity, which are denoted as *(m, n) connectivity*, where m is 8 and n is 4.

Many methods [38–42] have been proposed to examine topology (connectivity) preservation. Here, we start from an early connectivity preservation proof for the parallel thinning algorithm [40].

A *reduction operation O*, which is used in the deletion condition in the parallel thinning algorithm to change a foreground pixel into a background pixel, is said to preserve (m, n) connectivity if all of the following properties hold:

1. An m-component of the foreground is not split into two or more m-components of the foreground after applying reduction operation O.
2. An m-component of the foreground is not completely deleted by the use of reduction operation O.
3. Two or more n-components of the background will not merge into one n-component of the background after the use of reduction operation O.
4. No new n-components of the background are created by the use of reduction operation O.

Based on this classical early method, Ronse [38, 39] presented a rather simple, local set of sufficient conditions for a reduction operator O to preserve (8,4) connectivity. He defined the concept of an *8-deletable set* in which a pair of 8-simple 1's, $\{p, q\}$, is 8-deletable if and only if q is 8-simple after p is deleted. This set can be deleted while

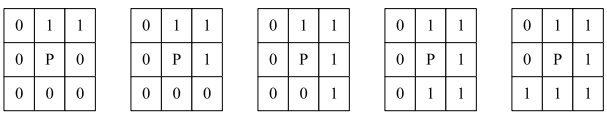


Fig. 8 Different patterns of pixel P and its 8-neighbor

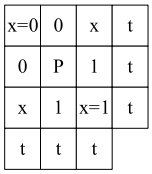


Fig. 9 Compulsory deletion template under the condition that $A(P)$ is 1

maintaining the connectivity of the image. The following sufficient conditions for topology preservation can be derived from Ronse's result [41]:

1 If a foreground pixel is deleted by the reduction operation, then it must be 8-simple.
2 If two 4-adjacent foreground pixels are both deleted by the reduction operation, then they must constitute an 8-deletable set.
3 No 8-component composed of two, three or four mutually 8-adjacent foreground pixels is completely deleted by the reduction operation.

Here, we use Ronse's method to prove the connectivity preservation of our algorithm.

5.2 Ronse's first condition holds in the proposed method

In this case, we suppose the restoring template does not work to simplify the discussion. (This is because the function of the restoring template is to retain some pixels rather than delete some pixels.) When $B(P)$ is within the range of 2 to 6 and $A(P)$ is equal to 1, all the kinds of patterns of 8-neighbors of pixels that meet the above condition can be categorized into 5 different patterns (all other patterns can be obtained by rotating 45 degrees clockwise in integer multiples), which are shown in Fig. 8. It is obvious that they are all 8-simple. It may not easily be found that pixel P in the compulsory deletion template is 8-simple because there may be doubt that *C(P)* is not equal to 1. However, the compulsory deletion template works only when $A(P)$ is equal to 1 (see line 10 of the pseudocode in Algorithm 1). As a result, the compulsory deletion template should look like Fig. 9, in which the 'x' in the upper left corner of P can only be 0 and the 'x' in the lower right corner can only be 1. This can be recognized as patterns 2, 3 or 4 in Fig. 8. Then, we find that pixel P in the compulsory deletion template is indeed an 8-simple pixel.

For the extra deletion templates, it is obvious that all pixels P are 8-simple pixels.

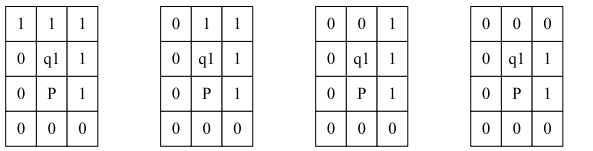


Fig. 10 Four possible sub-patterns

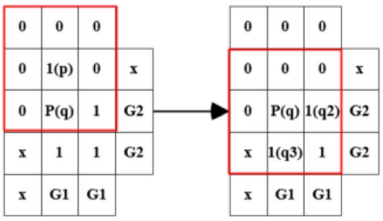


Fig. 11 The process of removing pixel P

5.3 Ronse's second condition holds in the proposed method

We temporarily assume that pixel P and all the pixels 4-adjacent to it are 8-simple pixels in the patterns in Fig. 8. In addition, they and their 8-neighbors either do not match any of the restoring templates or do match with the compulsory deletion template in the case that $B(P)$ is greater than or equal to 2 but not more than 6 and that $A(P)$ is one, which is the current reduction operation. For convenience, the symbols $q1$, $q2$, $q3$ and $q4$ are used to denote the north foreground pixel, east foreground pixel, south foreground pixel and west foreground pixel that are 4-adjacent to pixel P, respectively.

In pattern 1 (Fig. 8), $q1$ cannot be deleted by the reduction operation because the minimum value of $A(q1)$ is 2, whereas in the current assumption, $A(q1)$ should equal 1. In this case, only pixel P can be deleted.

In pattern 2, there are two pixels 4-adjacent to pixel P, which are pixels $q1$ and $q2$; here, $q1$ is taken as an example. According to our assumption, $q1$ should be the pixel that can be deleted by the current reduction operation and should be an 8-simple pixel. As a result, there are only 4 possible sub-patterns, which are shown in Fig. 10.

After P is deleted, $N_8(q1)$ in sub-pattern 1, sub-pattern 2, sub-pattern 3 and sub-pattern 4 belong to pattern 4, pattern 3, pattern 2 and pattern 1 in Fig. 8., respectively. Therefore, $q1$ is still an 8-simple pixel after the removal of pixel P. The same proof can also be conducted on pixel $q2$. As a result, we can conclude that the set of $\{P, q1\}$ and $\{P, q2\}$ are 8-deletable sets.

Note that pixel $q2$ in pattern 3 cannot be deleted by the reduction operation because $N_8(q2)$ does not belong to any of the patterns in Fig. 8 regardless of which combination of values (0 or 1) it takes in its rightmost extended column. In contrast, $q1$ is an 8-simple pixel and can be deleted by our deletion condition, and the set $\{P, q1\}$ constitutes an 8-deletable set. The proof of this is the same as the one for pattern 2.

In pattern 4, $q1$ and $q3$ can be proven by using the same proof as in pattern 2. Pixel $q2$ cannot be a removable pixel that satisfies the reduction operation because if $q2$ is a removable pixel, pixel P will match restoring template (a) but will not match the compulsory deletion template. Then, pixel P will become a restoring pixel, which contradicts our assumption.

In pattern 5, only $q1$ is an 8-simple pixel and can be deleted by reduction operations. Pixel $q2$ faces the same situation as pixel $q2$ in pattern 4, which cannot be a removable pixel. Pixel $q3$ in pattern 5 cannot be deleted by the reduction operation, which is similar to the case of pixel $q2$ in pattern 3.

Thus far, we have proven that the 8-simple pixels in patterns 1 to 5, which are 4-adjacent to pixel P and meet the deletion conditions, are still 8-simple pixels after the deletion of pixel P. They can constitute a deleted set with pixel P.

Since the compulsory deletion template can be considered as patterns 3 to 5, the proof for the compulsory deletion template can follow the proofs for patterns 3 to 5.

Now, we prove that the extra deletion templates also meet Ronse's second condition. In this case, we take the extra deletion template (d) as an example, as shown in Fig. 11. We denote P as q. The pixel that is the north neighbor of P is denoted by a lowercase p. Note that p can be deleted because it belongs to pattern 1, and q can also be deleted because it matches the extra deletion template. Additionally, these are all 8-simple pixels. Suppose now that pixel p is deleted; pixel q is still an 8-simple pixel because it can belong to pattern 2 or pattern 3. Then, we can use the proof of pattern 2 or pattern 3 to prove that set of $\{q, q2\}$ and $\{q, q3\}$ are 8-deletable sets if $q2$ and $q3$ meet the reduction conditions. The proofs for the extra deletion templates (e) to (k) are similar to this procedure. In Templates (l) and (m), there are not 4-adjacent foreground pixel can be parallel deleted.

5.4 Ronse's third condition holds in the proposed method

In [39], a test set for the last Ronse condition was presented, as shown in Fig. 12. The result of our algorithm is shown in Fig. 13. All the fraction patterns have been retained. The result confirmed that the last Ronse condition also holds in our algorithm.

Thus far, we have proven that all three Ronse conditions hold in our algorithm. Therefore, our algorithm has the characteristic of topology preservation.

6 Measures used for comparison

To evaluate the proposed algorithm, several measures are defined for comparison purposes.

6.1 Measures for evaluating unit-width pixels

A measurement method of convergence to unit width was proposed in [14]. A skeleton S is one-pixel wide if and only if skeleton S does not contain any of the patterns Q^n, as shown in Fig. 14. The indicator m_t is introduced to measure the width of the resulting skeleton:

$$\mathrm{m_t} = 1 - \frac{\mathrm{Area}\left[\cup_{\mathrm{n=1}}^{4}(\mathrm{S}\Theta\mathrm{Q^n})\right]}{\mathrm{Area[S]}} \tag{16}$$

where $Area[]$ is an operation that counts the number of foreground pixels. Θ is an operation of erosion and, in fact, Q^n are structuring elements.m_t is a non-negative value no larger than one. S is an ideal single-width skeleton when m_t is one.

6.2 Parameter for evaluating the sensitivity to boundary noise

Let I be a given image without boundary noise and I'' be the noisy version of it obtained by randomly removing or adding some pixels along the edge of I. The concept of the *signal-to-boundary noise ratio* (*SBNR*) was defined in [17] as follows:

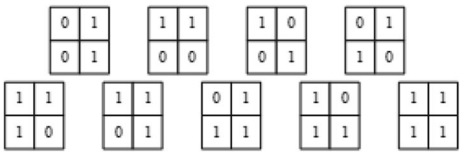


Fig. 12 Test set for the last Ronse condition

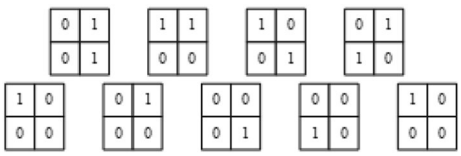


Fig. 13 Test result of the proposed algorithm

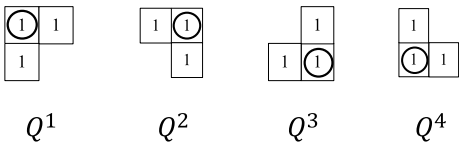


Fig. 14 Templates used to test the width of the skeleton

$$SBNR = \frac{Area[\partial I]}{Area\left[I''/I\right]+Area[I/I'']} \tag{17}$$

where ∂I denotes the boundary of I and I''/I denotes the pixels that belong to set I'' but do not belong to set I. Thus, *the error* caused by boundary noise at a particular *SBNR* can be measured by the normalized quantity

$$m_e = \min\left[1, \frac{Area\left[S/S''\right]+Area[S''/S]}{2\times Area[S]}\right] \tag{18}$$

where S is the resulting skeleton of I and S'' is the resulting skeleton of I''. Algorithms that are highly sensitive to boundary noise will yield a m_e value close to 1.

6.3 Parameter for evaluating speed

The *thinning speed* (*TS*) was proposed in [22] to measure the number of pixels thinned per time unit (seconds), and it is computed as follows:

$$\mathrm{TS} = \frac{\mathrm{DP}}{\mathrm{ET}} \tag{19}$$

$$\mathrm{DP} = \mathrm{OP} - \mathrm{SP} \tag{20}$$

where *execution time* (*ET*) is the actual time consumed by the algorithm to skeletonize the binary picture on a computer, *deleted pixels* (*DP*) is the number of foreground pixels removed during the thinning procedure, and *object pixels* (*OP*) is the number of foreground pixels in the original image. *Skeletal pixels* (*SP*) is the number of foreground pixels remaining in the skeleton.

In addition, we use the measure of the *number of iterations* (*NIT*). *NIT* counts the number of iterations. Generally, a faster algorithm has fewer iterations. One iteration can be considered a full scan of the image, in which many foreground pixels are transformed to background pixels.

7 Experiments and results

The proposed algorithm, ZS-series algorithms (including the ZS, Tarabek, RIEPTA and MZS algorithms), and the OPTA-series algorithms mentioned above were implemented in the C++programming language. Simple test, boundary noise test and complex image test were conducted to evaluate the performances with regard to thinning rate, noise sensitivity, and computational speed.

In the first test, the test picture is deliberately designed according to the known drawbacks of the algorithms.

The boundary noise test starts with a simple rectangle, from which every algorithm can extract a clean skeleton (ignoring differences in the skeleton shape between them),

and then we manually set the probability of mutation (noise level) of the boundary pixels, which gradually increases from 2.5% to 20%. We randomly add this noise to the boundary of the original image and observe the response of these algorithms to the noise.

Finally, in the last test, we use more complex test images, which mainly come from the famous datasets Kimia-99, MPEG-7, and DRIVE. We add some images of Chinese characters collected from the internet. The benchmark dataset usage is shown in Table. 1.

7.1 Simple test

As shown in Fig. 15, a simple pattern consisting of a 2×2 square (top center of the original image), a 3×3 square (on the left), a 3×4 rectangle (on the right), a 3×6 rectangle (at the bottom) and crossed, sloped lines was designed to test the performances of the different algorithms.

The test image and the results of the various algorithms are shown in Fig. 15, where the gray pixels are the original shape and the black superposed black pixels are the skeleton.

Clearly, the ZS algorithm and the RIEPTA algorithm suffer from severe excessive erosion problems on the crossed lines and preserve only the door-like shapes in the center. These two algorithms completely delete all four pixels of the 2×2 pattern located at the top of the original picture, leaving nothing. Furthermore, the 3×3 square located on the left of the original picture and the 3×4 rectangle located on the right are processed by the ZS algorithm in the same way, which can be considered another outcome of the excessive erosion problem. The Tarabek algorithm and the MZS algorithm largely address the drawbacks of the complete disappearance of the 2×2 pattern and the erosion of the crossed lines resulting from the ZS algorithm, but their results still fail to distinguish between the 3×3 square and the 3×4 rectangle. In addition, MZS suffers from a new erosion problem—the 3×6 rectangle at the bottom yields only one pixel, which does not represent the original pattern well.

The OPTA algorithm preserves the topology of the initial shape well, but it is unable to ensure a one-pixel thickness for the crossed lines. The shortcoming of the OPTA4 algorithm is that it loses the 2×2 pattern. The skeleton produced by FCTA fails to distinguish between the square and the rectangle. The proposed algorithm preserves the topology of the original pattern well without excessive erosion of the intersecting lines and ensures a one-pixel thickness

Table. 1 Dataset usage

Dataset Name	KIMIA-99	DRIVE	MPEG-7
Total image number	99	20	1402
Usage	99	20	752

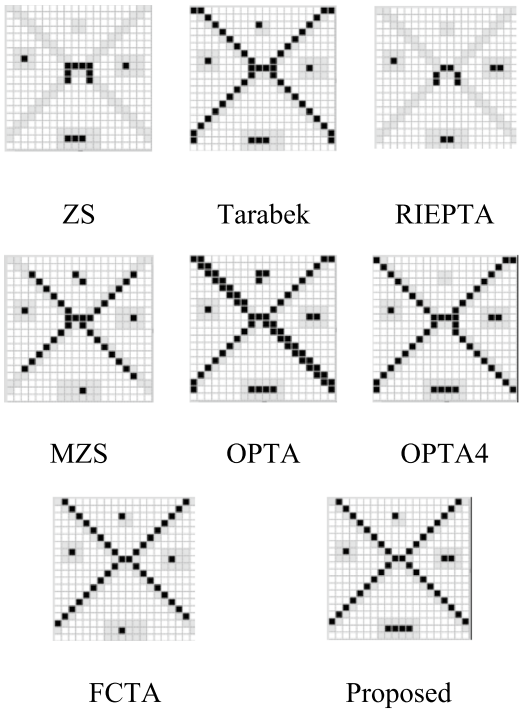


Fig. 15 The original pattern and skeletons of the simple pattern produced by the various algorithms

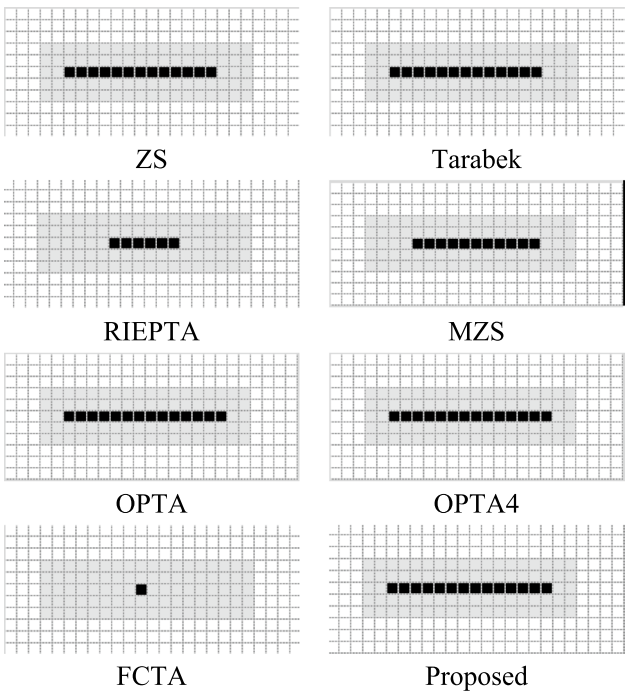


Fig. 16 Skeleton resulting from the clean pattern

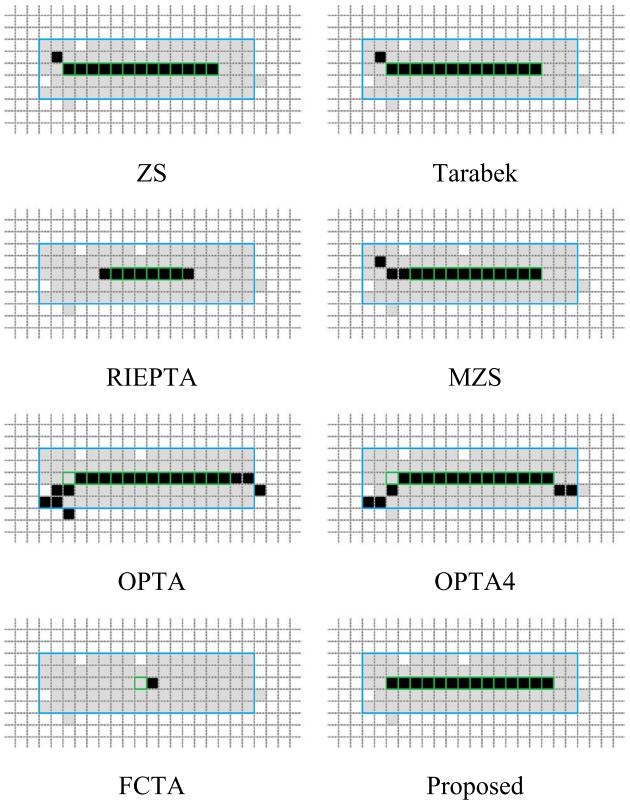


Fig. 17 Under 5% boundary noise, SBNR = 8.4; ZS: $m_e = 0.038$; Tarabek:$m_e = 0.038$; RIEPTA: $m_e = 0.166$; MZS: $m_e = 0.136$; OPTA: $m_e = 0.321$, OPTA4: $m_e = 0.214$; FCTA: $m_e = 1$; proposed: $m_e = 0$. (Original clean boundary and clean skeleton are colored with blue and green)

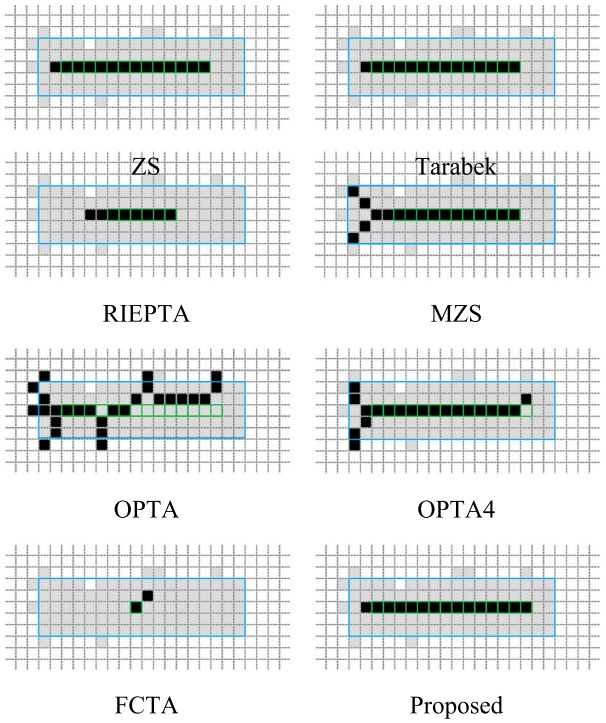


Fig. 18 Under 10% boundary noise, SBNR = 4.66; ZS: $m_e = 0.038$; Tarabek:$m_e = 0.038$; RIEPTA: $m_e = 0.166$; MZS: $m_e = 0.272$; OPTA: $m_e = 1$, OPTA4: $m_e = 0.285$; FCTA: $m_e = 0.5$; proposed: $m_e = 0.035$. (Original clean boundary and clean skeleton are colored with blue and green)

throughout. The proposed algorithm properly identifies the rectangle and square, representing the square and the rectangle by an individual pixel and by several successive pixels, respectively.

7.2 Boundary noise test

In this section, we first build an ideal rectangular pattern, whose size is 5 by 18. Then, eight thinning algorithms are used to extract the skeleton. The results are shown in Fig. 16, where the gray pixels are the original pattern and the black pixels are the skeleton.

To study the robustness, which mainly concerns the results of the algorithm under boundary noise, we temporarily ignore the difference in the shape of the skeleton between different algorithms and consider the skeletons extracted from the clean pattern as the clean skeletons. These clean skeletons will be compared with those generated from the noisy pattern by the same algorithm. The parameters of *SBNR* and m_e, which were introduced in Sect. 6.2, will be used in this section to evaluate the results.

The noise pattern is generated by randomly adding some pixels (changing a background pixel to a foreground pixel) or removing some pixels (changing a foreground pixel to a background pixel) along the boundary of the clean pattern with a given probability, which is also said to be the noise level.

The whole test is divided into eight subtests according to the different noise levels, which start at 2.5% and gradually increase to 20%, and the increment is set to 2.5%. In each subtest, noise is randomly added to the original boundary, and all 8 thinning algorithms are used to extract the skeletons. To objectively evaluate the robustness of the thinning algorithms, each subtest is conducted independently 100 times. Figures 17, 18, 19 and 20 present noisy images (gray pixels) under noise levels of 5%, 10%, 15% and 20%, respectively, and the results (black pixels) generated by the thinning algorithms superposed on these images. In additional, for convenience in comparing, we, respectively, mark the original pattern and initial clean skeleton with blue and green colors. Table 2 presents the confidence interval of the whole test.

Figures 17, 18, 19 and 20 give an intuitive sense of the robustness of the different thinning algorithms. In Fig. 17, from the perspective of visual effect, FCTA and RIEPTA and

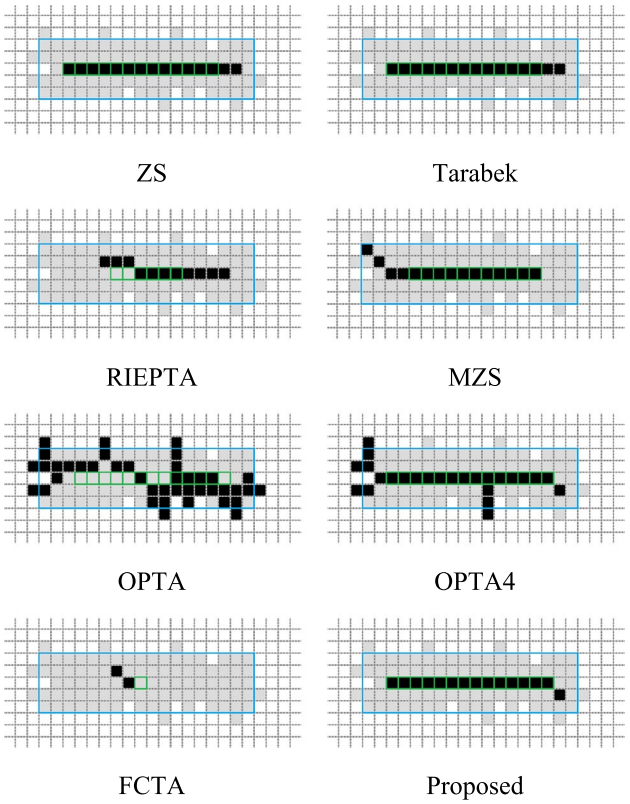


Fig. 19 Under 15% boundary noise, SBNR = 3; ZS: $m_e = 0.076$; Tarabek:$m_e = 0.076$; RIEPTA: $m_e = 0.75$; MZS: $m_e = 0.181$; OPTA: $m_e = 1$, OPTA4: $m_e = 0.357$; FCTA: $m_e = 1$; proposed: $m_e = 0.0357$. (Original clean boundary and clean skeleton are colored with blue and green)

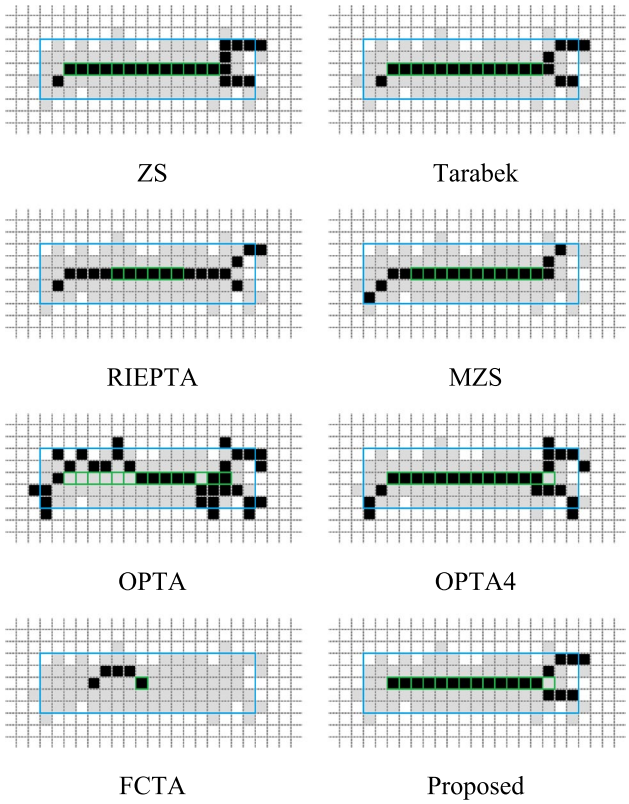


Fig. 20 Under 20% boundary noise, SBNR = 2.21; ZS: $m_e = 0.384$; Tarabek:$m_e = 0.307$; RIEPTA:$m_e = 1$; MZS: $m_e = 0.318$; OPTA: $m_e = 1$; OPTA4: $m_e = 0.535$; FCTA:$m_e = 1$; proposed: $m_e = 0.285$. (Original clean boundary and clean skeleton are colored with blue and green)

the proposed algorithm yield good skeletons, whose shape seem to be the same as those extracted from the clean pattern. In contrast, slight changes occur in the ZS, Tarabek and MZS results. However, from the perspective of the error parameter m_e, the performance of FCTA and RIEPTA is much worse than that of ZS, Tarabek and MZS. The reason for this contradiction is that the slight movement of the location of the skeleton and the slight changes in the length of the skeleton are usually ignored by people; however, the parameter m_e is very sensitive to these changes and tends to give a high value to them. Regardless of the visual effect or the view of parameter comparison, the proposed algorithm suppresses this faint boundary noise, whereas the skeletons resulting from OPTA and OPTA4 are relatively unsatisfactory.

In Figs. 18, 19 and 20, as the noise level increases, an increasing number of unwanted branches are noticed in the skeletons, and the error caused by the boundary noise continues to ascend, which indicates that all the algorithms are influenced by boundary noise to different extents. Overall, our algorithm has a similar capacity in terms of anti-boundary noise as the ZS, Tarabek and RIEPTA algorithms. These are followed by MZS and OPTA4. The OPTA algorithm is the most sensitive algorithm to boundary noise.

In Table 2, we present the confidence intervals of the signal boundary noise rate and that of the error caused by boundary noise m_e under different levels of noise. It can be seen that the lower and upper bound of the confidence interval of *SBNR* are both continuously decreasing because an increasing number of noisy pixels appear near the primitive edge. By observing the parameter m_e, the proposed algorithm has the best property tolerating the boundary noise.

7.3 Complex image test

This experiment was conducted to evaluate the algorithm performances in terms of single-pixel thickness and thinning speed.

To determine the performances of these thinning algorithms, the skeletons extracted from 6 different real-life binary patterns, which are a person, bonefish, airplane, Chinese character, retina and fingerprint, along with their

Table 2 Confidence interval of the error caused by the boundary noise (m_e) of the eight algorithms under different noise levels

Noise Level	Confidence Interval (Confidence level is 95%, sample size at each noise level is 100)							
	SBNR	Error caused by boundary noise m_e						
		ZS	Tarabek	RIEPTA	MZS	OPTA	OPTA4	FCTA	Proposed
2.5%	[19.833, 25.287]	[0.0184, 0.0401]	[0.0182, 0.0394]	[0.2004, 0.3479]	[0.0232, 0.0551]	[0.1816, 0.2670]	[0.0502, 0.7971]	[0.2321, 0.4179]	[0.0063, 0.0173]
5.0%	[10.705, 14.241]	[0.0329, 0.0609]	[0.0331, 0.0607]	[0.3327, 0.4738]	[0.0436, 0.0763]	[0.3588, 0.4569]	[0.0942, 0.1336]	[0.4113, 0.6085]	[0.0184, 0.0386]
7.5%	[6.401, 7.690]	[0.0399, 0.0701]	[0.0405, 0.0702]	[0.2946, 0.4204]	[0.0648, 0.1024]	[0.5574, 0.6612]	[0.1384, 0.1816]	[0.6086, 0.7913]	[0.0315, 0.0543]
10.0%	[5.114, 5.932]	[0.0846, 0.1340]	[0.0834, 0.1319]	[0.4688, 0.5895]	[0.1340, 0.1868]	[0.7377, 0.8265]	[0.2254, 0.2817]	[0.6376, 0.8123]	[0.0708, 0.1070]
12.5%	[3.932, 4.417]	[0.1044, 0.1579]	[0.0991, 0.1508]	[0.4687, 0.5712]	[0.1385, 0.1996]	[0.7996, 0.8817]	[0.2792, 0.3415]	[0.6414,0.8185]	[0.0871, 0.1272]
15.0%	[3.391, 3.849]	[0.1327, 0.2019]	[0.1308, 0.1991]	[0.4801, 0.5782]	[0.1709, 0.2435]	[0.8494, 0.9205]	[0.3392, 0.4171]	[0.6425, 0.8174]	[0.1131,0.1690]
17.5%	[2.702, 2.969]	[0.1789, 0.2480]	[0.1755, 0.2422]	[0.5613, 0.6520]	[0.2460, 0.3258]	[0.8991, 0.9551]	[0.3924, 0.4612]	[0.6973, 0.8626]	[0.1571,0.2121]
20.0%	[2.535,2.877]	[0.2019, 0.2857]	[0.1957, 0.2765]	[0.5861, 0.6822]	[0.2580, 0.3464]	[0.9693, 0.9913]	[0.4512, 0.5373]	[0.7317, 0.8882]	[0.1974, 0.2668]

original patterns, are shown in Figs. 21, 22, 23, 24, 25 and 26. The visual effect is enhanced by using different colors to indicate mistakes. We use a red circle to denote a disconnected skeleton, a yellow circle to denote a skeleton with excessive erosion, and a blue circle to denote a skeleton without a single-pixel thickness.

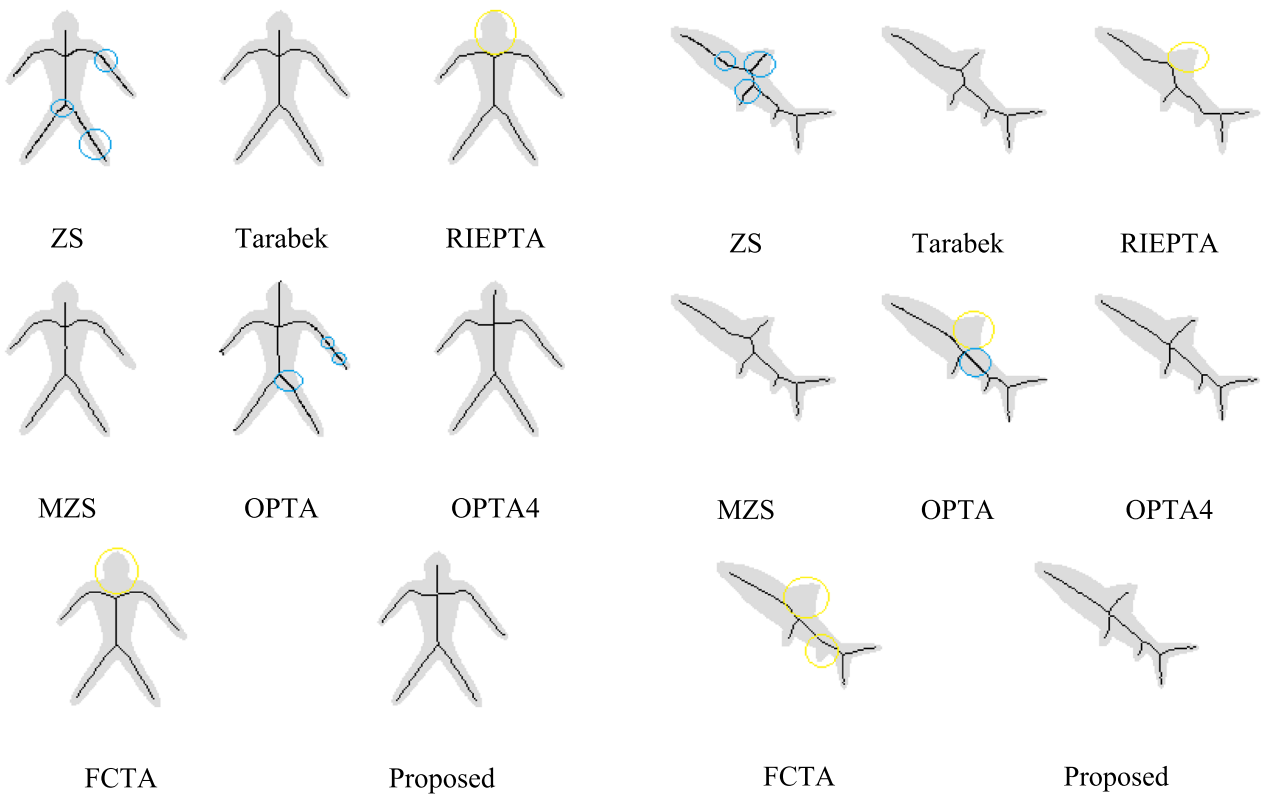


Fig. 21 Skeleton results on the person image

Fig. 22 Skeleton results on the bonefish image

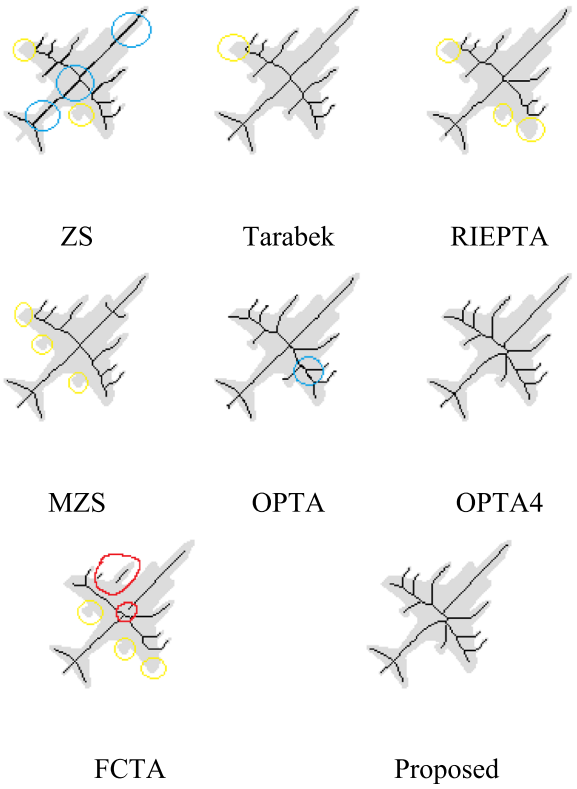


Fig. 23 Skeleton results on the airplane image

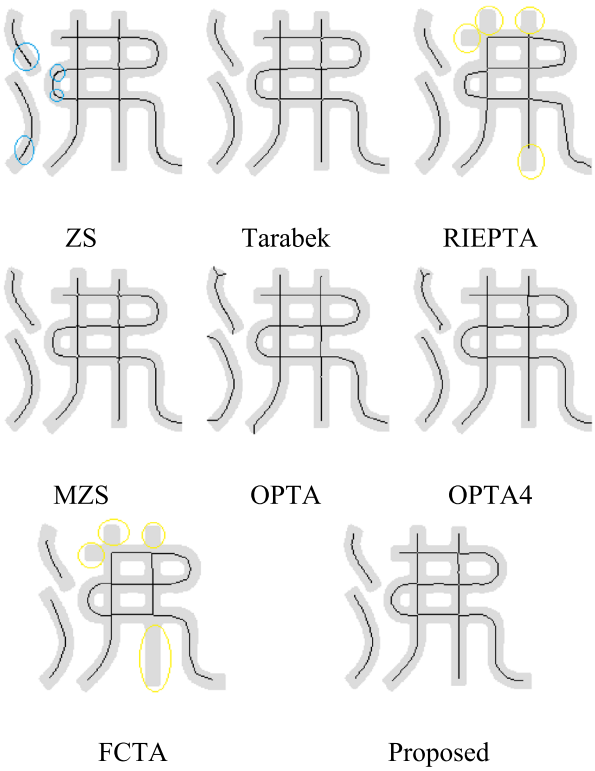


Fig. 24 Skeleton results on the character image

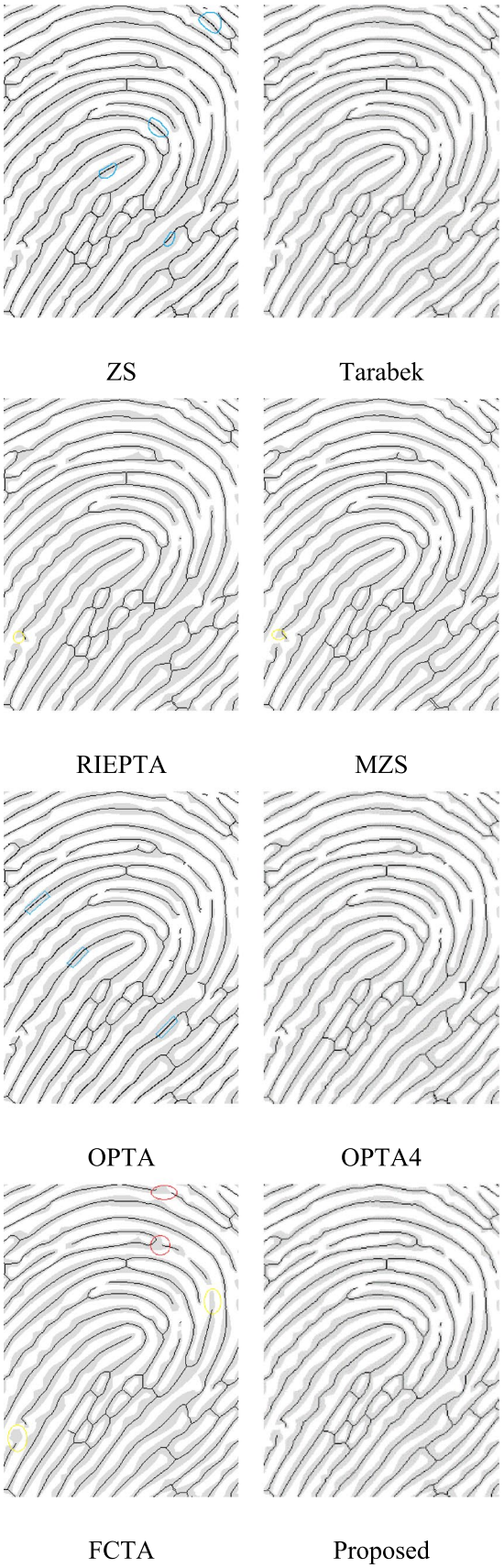


Fig. 25 Skeleton results on the fingerprint image

In the person image (Fig. 21.), the FCTA algorithm and the RIEPTA algorithm lost the head part (marked with a yellow circle), while the OPTA algorithm and the ZS algorithm could not ensure a skeleton with single-pixel thickness (see the blue circle). The proposed algorithm, MZS algorithm, OPTA4 algorithm and Tarabek algorithm preserved the original structure well, and the skeletons all have ideal single-pixel thicknesses.

For the bonefish image (Fig. 22), all 8 algorithms were able to maintain skeleton connectivity; however, not every algorithm was able to retain the shape embodied by the original image. Methods such as FCTA, OPTA and RIEPTA are good examples. In addition, a double-pixel problem occurred in the results of the OPTA and ZS algorithms.

As shown in Fig. 23, all four of the ZS-series algorithms eliminated the wing of the airplane to some degree. The ZS, Tarabek and MZS algorithms removed the left wing but preserved the right wing, whereas the RIEPTA algorithm removed both wings. Among the OPTA-series algorithms (excluding the FCTA algorithm), all of them preserved the basic shape of the original input image. The FCTA algorithm not only erased the left wing but also broke the connectivity of the plane. The broken sites are highlighted in red. Redundant pixels can be observed inside the blue circle in the ZS algorithm and OPTA algorithm results. The proposed algorithm and the OPTA4 algorithm were able to produce good results in terms of shape preservation and maintaining one-pixel thickness on the airplane image, but their skeletons seem slightly asymmetric in the axis of the airplane.

Figure 24 shows that on the Chinese character image, almost all the algorithms generated good results with regard to shape fidelity and connectivity except for the FCTA and RIEPTA algorithms. Both methods failed to preserve the basic shape, and erosion occurred in the lines enclosed by the yellow circles, which could cause recognition difficulties in some fields, such as optical character recognition (OCR).

The fingerprint image experiments (see Fig. 25) indicate that almost all the algorithms roughly preserved the primary structure of the original image, but some algorithms did not output a proper skeleton. The FCTA algorithm suffered from disconnection problems in the top right plane and lost the end points of some lines; these problems also occurred in the MZS and RIEPTA algorithms. The ZS algorithm and the OPTA algorithm are still plagued by double-width thickness problems. The proposed algorithm, along with the OPTA4 and Tarabek algorithms, produced better results than the other algorithms.

On the retinal image (in Fig. 26), the skeletons of the blood vessels produced by the Tarabek algorithm, the OPTA4 algorithm and the proposed algorithm preserved both shape and connectivity. The FCTA algorithm generated two disconnected segments, and some algorithms, including MZS, ZS and RIEPTA, removed parts of some lines. The OPTA algorithm and the ZS algorithm yielded some thicker segments (marked in yellow).

A quantitative comparison in these test images of the parameters of the performances is shown in Table 3, based on the measures described in the previous section, for the proposed algorithm and the seven other algorithms in the experiments. The best and worst values are displayed in green and red, respectively.

The initial values (on the original images) are listed in the second column; object pixels (OP) can be obtained by counting all the foreground pixels. The results generated by the different algorithms are shown in the third to last columns. From the resulting data, a comparison confirms that the proposed algorithm achieves better performance in terms of the characteristics below.

7.3.1 Thinness

Comparing the results in Table 3, RIEPTA, FCTA and the proposed algorithm are clearly the top three thinness algorithms among the 8 algorithms in these images. If one considers only the one-pixel-thick skeleton property, the RIEPTA algorithm earns first place in processing the bonefish, character and blood vessel images among these algorithms. However, for people and fingerprints, the proposed algorithm is a better choice. The FCTA algorithm outperforms the other 7 algorithms on only the airplane image. The ZS algorithm performs the worst in terms of single-pixel thickness and has the lowest m_t value on all 6 images, followed by the OPTA, OPTA4, Tarabek and MZS algorithms.

7.3.2 Efficiency

Three main parameters describe algorithm efficiency: NIT, ET and TS. In Table 3, among the 8 algorithms, from the perspective of the number of iterations, the two fastest algorithms are the proposed algorithm and the OPTA4 algorithm.

The ET results, which reflect the real execution times, largely support the above speculation but with some differences. On some images, the OPTA4 algorithm is not faster than the ZS algorithm. This is caused by the use of template matching for each pixel in the OPTA4 algorithm, which is a more complex operation than the simple logical operation that the ZS algorithm uses to find potentially removable candidate points. In the proposed algorithm, not all deleted pixels are identified by template matching. In fact, most of them are removed through logical operations, which explains why the proposed algorithm is faster than all the other algorithms.

TS, a measurement parameter mentioned in the previous section, considers not only the time spent but also the number of removed pixels; therefore, it provides a more

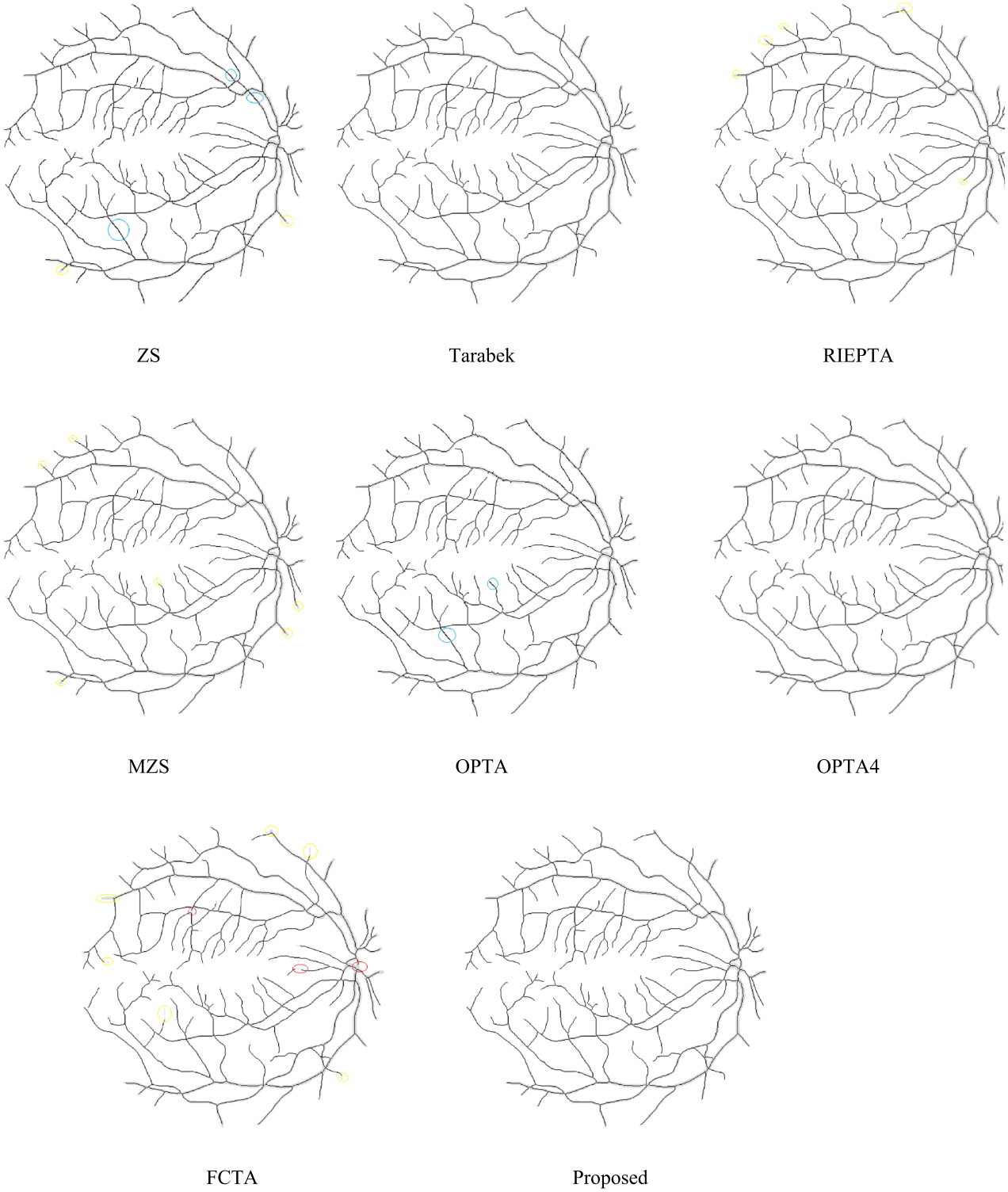


Fig. 26 Skeleton results on the retina image

Table 3 Primary comparison of the performance in 6 complex images

	Initial value		PM	ZS	Tarabek	RIEPTA	MZS	OPTA	OPTA4	FCTA	Proposed
People	OP	2754	m_t	0.6720	0.9902	0.9892	0.9846	0.7920	0.9764	0.9887	0.9903
			SP	247	205	186	195	250	212	178	208
			NIT	24	24	36	25	33	19	31	19
			ET (ms)	3.7042	4.0270	5.9139	3.5648	9.9584	3.8192	10.5771	2.8020
			TS (p/s)	676,786	632,974	434,230	717,846	251,446	665,578	243,546	908,630
Bonefish	OP	7206	m_t	0.69518	0.98734	0.99231	0.98101	0.74251	0.97041	0.98437	0.98717
			SP	187	158	130	158	167	169	128	156
			NIT	24	24	39	25	26	17	26	17
			ET (ms)	8.7122	10.1619	14.2621	8.96732	10.0736	8.2616	13.8031	7.2927
			TS (p/s)	805,647	693,571	496,138	785,965	698,753	851,772	512,785	966,726
Airplane	OP	9942	m_t	0.4971	0.9829	0.9843	0.9781	0.9238	0.9497	0.9914	0.9879
			SP	348	293	255	274	328	338	233	318
			NIT	34	34	68	35	39	20	38	20
			ET (ms)	8.8081	9.2063	15.4594	8.2761	11.1594	7.2667	10.1546	6.6397
			TS (p/s)	1,089,215	1,048,086	626,610	1,168,187	861,517	1,321,647	956,116	1,449,472
Character	OP	9946	m_t	0.8990	0.9884	0.9983	0.9841	0.9677	0.9651	0.9788	0.9956
			SP	723	691	613	694	743	717	566	691
			NIT	18	16	31	17	24	16	61	16
			ET (ms)	8.5028	8.9272	11.8306	9.1671	7.8948	7.9192	14.3858	6.6874
			TS (p/s)	1,084,696	1,036,711	788,887	1,171,902	1,003,919	1,165,382	652,033	1,383,939
Fingerprint	OP	63,563	m_t	0.7267	0.9922	0.9957	0.9936	0.7841	0.9902	0.9950	0.9961
			SP	6799	5826	5658	5822	6521	5853	5447	5702
			NIT	18	18	27	19	23	15	25	15
			ET (ms)	57.7111	59.5721	69.5280	56.0488	60.1683	52.7672	63.4155	51.9558
			TS (p/s)	983,589	969,196	832,831	1,030,192	948,041	1,093,672	916,431	1,113,658
Blood Vessels	OP	25,001	m_t	0.7551	0.9855	0.9948	0.9867	0.8688	0.9809	0.9883	0.9935
			SP	8748	7699	7559	7548	8187	7697	7350	7616
			NIT	11	10	20	11	13	7	23	7
			ET (ms)	17.6103	19.2947	27.2041	18.3435	19.2422	12.2018	30.7406	10.7219
			TS (p/s)	922,927	896,724	641,152	951,452	873,806	1,418,156	574,193	1,621,441

comprehensive indication of the efficiency of these algorithms. From this viewpoint, the proposed algorithm retains its superiority in terms of speed, while the FCTA and RIEPTA algorithms are the two slowest.

The performance of the thinness and efficiency of these thinning algorithms in 871 test images are summarized in Tables 4, 5 and Tables 6, 7, respectively.

In both Tables 4 and 5, the value of m_t is directly used as sample data. Table 4 presents the descriptive statistics of m_t for different algorithms. Table 4 shows that the proposed algorithm has relatively good performance in terms of single-pixel thickness (thinness), and its thinness abilities rank well among all 8 algorithms, second only to RIEPTA. Two-factor ANOVA without a replication test was conducted, and the results are presented in Table 5.

From Table 3, it is reasonable to speculate that the proposed algorithm has the highest efficiency among these algorithms. To study the time consumption of the proposed algorithm compared with that of the other 7 algorithms, the ratio of the execution time of our algorithm and the execution time of the other algorithms are used as new measures when the algorithms are performed on a large dataset. This new parameter is denoted as $R_{proposed}$. To understand this new parameter, an example is given. Suppose there exists another algorithm A, the $R_{proposed}$ of A can be obtained by dividing ET of the proposed algorithm by that of the A algorithm. If proposed algorithm is faster than algorithm A, then the $R_{proposed}$ value should be less than 1. Otherwise, it should be larger than 1.

Table 6 shows that the mean values of ratio $R_{proposed}$ are all less than 1, which means that in most of the test images, whose total number is 871, the proposed algorithm is faster than the others.

Table 4 Statistical description of the m_t on the dataset

	m_t							
	ZS	Tarabek	RIEPTA	MZS	OPTA	OPTA4	FCTA	Proposed
Mean	0.77600	0.98825	0.99504	0.98903	0.93951	0.98117	0.99308	0.99480
Standard Error	0.00396	0.00048	0.00023	0.00043	0.00224	0.00077	0.00045	0.00022
Standard Deviation	0.11714	0.01415	0.00703	0.01264	0.06611	0.02263	0.01327	0.00660
Sample Variance	0.01372	0.00020	0.00005	0.00016	0.00436	0.00051	0.00018	0.00004
Confidence Interval (95%)	0.00779	0.00094	0.00046	0.00084	0.00439	0.00150	0.00088	0.00044
Sample Size	871	871	871	871	871	871	871	871

Table 5 Two-factor ANOVA test without replication for *mt* ($\alpha = 0.05$)

Source of Variation	SS	df	MS	F	*P*-value	F crit
Between Algorithms	34.6935	7	4.9562	2487.654	0	2.0109
Between Images	4.5554	870	0.0052	2.6186	3.6E-99	1.0863
Error	12.1773	6090	0.002			
Total	51.4263	6967				

Table 6 Statistical description of ratio R on the dataset

	The ratio $R_{proposed}$ of the time of the proposed algorithm and of each other algorithm						
	ZS	Tarabek	RIEPTA	MZS	OPTA	OPTA4	FCTA
Mean	0.74414	0.71289	0.53330	0.78660	0.62873	0.83660	0.58572
Standard Error	0.00245	0.00172	0.00333	0.00249	0.00178	0.00249	0.00125
Standard Deviation	0.07249	0.05104	0.09837	0.07369	0.05267	0.07369	0.03717
Sample Variance	0.00525	0.00260	0.00967	0.00543	0.00277	0.00543	0.00138
Confidence Interval (95%)	0.00482	0.00339	0.00654	0.00490	0.00350	0.00490	0.00247
Sample Size	871	871	871	871	871	871	871

Table 7 Two-factor ANOVA test without replication of ratio $R_{proposed}$ ($\alpha = 0.05$)

Source of Variation	SS	df	MS	F	*P*-value	F crit
Between Algorithms	63.9840	6	10.6640	3688.3831	0	2.1003
Between Images	4.5554	870	0.01520	5.2603	0	1.0873
Error	15.0922	5220	0.0028			
Total	92.3082	6096				

In Table 7, the results of two-factor ANOVA without a replication test based on the value of $R_{proposed}$ are presented. The test confirms that the value of $R_{proposed}$ is related to both the algorithms and test images.

8 Conclusion

This study proposes a novel fully thinning algorithm that combines the merits of ZS-series algorithms and OPTA-series algorithms. The property of topology preservation is proven in Sect. 5. Then, the proposed algorithm is implemented and compared against four ZS-series algorithms and three OPTA-series algorithms on numerous images with different patterns. Three tests are conducted to evaluate the performance of the proposed algorithm in terms of insensitivity to boundary noise, thinning rate and execution speed. The noise test confirms that the proposed algorithm is as insensitive to boundary noise as are the ZS-series algorithms. In addition, the results of the simple and complex pattern tests confirm that the proposed algorithm achieves good results with high speed and produces a clean skeleton with single-thickness pixels. Consequently, the

proposed algorithm is indeed an effective and robust parallel algorithm that can be applied in different fields.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10044-021-01039-y.

References

1. Blum H (1962) An associative machine for dealing with the visual field and some of its biological implications. Biolo Prototyp Synthetic Syst. https://doi.org/10.1007/978-1-4684-1716-6_34
2. Kimia BB, Tannenbaum AR, Zucker SW (1995) Shapes, shocks, and deformations I: The components of two-dimensional shape and the reaction-diffusion space. Int J Comput Vision 15:189–224. https://doi.org/10.1007/BF01451741
3. Leymarie F, Levine MD (1992) Simulating the grassfire transform using an active contour model. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/34.107013
4. Siddiqi K, Pizer SM (2008) Medial Representations: Mathematics. Springer, Algorithms and Applications
5. Saha PK, Borgefors G, Sanniti di Baja G (2016) A survey on skeletonization algorithms and their applications. Pattern Recognit Letters 76:3–12. https://doi.org/10.1016/j.patrec.2015.04.006
6. Saha PK, Borgefors G, Sanniti di Baja G (2017) Skeletonization: Theory, Methods, and Applications. Academic Press, London. https://doi.org/10.1016/B978-0-08-101291-8.00017-1
7. Brandt JW, Algazi VR (1992) Continuous skeleton computation by Voronoi diagram. CVGIP: Image Understanding. 55:329–338. https://doi.org/10.1016/1049-9660(92)90030-7
8. Ogniewicz RL, Ilg M (1992) Voronoi skeletons: theory and applications. Proceed IEEE Comput Soc Conf Comput Vis Pattern Recognit 62:63–69. https://doi.org/10.1109/CVPR.1992.223226
9. Ogniewicz RL, Kübler O (1995) Hierarchic voronoi skeletons. Pattern Recogn 28:343–359. https://doi.org/10.1016/0031-3203(94)00105-U
10. Aslan C, Erdem A, Erdem E, Tari S (2008) Disconnected skeleton: Shape at its absolute scale. IEEE Trans Pattern Anal Mach Intell 30:2188–2203. https://doi.org/10.1109/TPAMI.2007.70842
11. Arcell C, Sanniti Di Baja G, Serino L (2011) Distance-driven skeletonization in voxel images. IEEE Trans Pattern Anal Mach Intell 33:709–720. https://doi.org/10.1109/TPAMI.2010.140
12. Lam L, Lee SW (1992) Thinning methodologies—a comprehensive survey. IEEE Trans Pattern Anal Mach Intell 9:869–885. https://doi.org/10.1109/34.161346
13. Chin RT, Wan HK, Stover DL, Iverson RD (1987) A one-pass thinning algorithm and its parallel implementation. Comput Vis Gr Image Process 40:30–40. https://doi.org/10.1016/0734-189X(87)90054-5
14. Jang BK, Chin RT (1992) One-pass parallel thinning: analysis, properties, and quantitative evaluation. IEEE Trans Pattern Anal Mach Intell 11:1129–1140. https://doi.org/10.1109/34.166630
15. Wu RY, Tsai WH (1992) A new one-pass parallel thinning algorithm for binary images. Pattern Recogn Lett 13:715–723. https://doi.org/10.1016/0167-8655(92)90101-5
16. Deng W, Iyengar SS, Brener NE (2000) Fast parallel thinning algorithm for the binary image skeletonization. Int J High Perform Comput Appl 14:65–81. https://doi.org/10.1177/109434200001400105
17. Zhou RW, Quek C, Ng GS (1995) A novel single-pass thinning algorithm and an effective set of performance criteria. Pattern Recogn Lett 16:1267–1275. https://doi.org/10.1016/0167-8655(95)00078-X
18. Wang RZ, Zhao YR, Ji TH, Liu LP (2020) Fingerprint Refinement Model Based on Improved OPTA. J Comput 31:274–283. https://doi.org/10.3966/199115992020123106021
19. Lei G, LI Z, Zhi W, (2017). A Fast and Complete Thinning Algorithm for Character Image. DEStech Transactions on Engineering and Technology Research
20. Gang C, Ning C, Yong Z (2012) An improved OPTA fingerprint thinning algorithm based on neighborhood searching. Proceedings - 2012 International Conference on Computer Science and Information Processing CSIP 2012. https://doi.org/10.1109/CSIP.2012.6308934
21. Zhang TY, Suen CY (1984) A fast parallel algorithm for thinning digital patterns. Commun ACM 27:236–239. https://doi.org/10.1145/357994.358023
22. Ben Boudaoud L, Solaiman B, Tari A (2018) A modified ZS thinning algorithm by a hybrid approach. Vis Comput 34:689–706. https://doi.org/10.1007/s00371-017-1407-4
23. Shen Y, Ai T, Yang M (2019) Extracting centerlines from dual-line roads using superpixel segmentation. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2895016
24. Tarabek P (2012) A robust parallel thinning algorithm for pattern recognition. 2012 7th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI) 75–79. IEEE.
25. Chen W, Sui L, Xu Z, Lang Y (2012) Improved Zhang-Suen thinning algorithm in binary line drawing applications. In 2012 International Conference on Systems and Informatics (ICSAI2012) 1947–1950. IEEE.
26. Lü HE, Wang PSP (1986) A Comment on "a fast parallel algorithm for thinning digital patterns." Commun ACM 29:239–242. https://doi.org/10.1145/5666.5670
27. Jagna A, Kamakshiprasad V (2010) New parallel binary image thinning algorithm. ARPN J Eng Appl Sci 5:64–67
28. Abdulla WH, Saleh AO, Morad AH (1988) A preprocessing algorithm for hand-written character recognition. Pattern Recogn Lett 7:13–18
29. Sossa JH (1989) An improved parallel algorithm for thinning digital patterns. Pattern Recogn Lett 10:77–80. https://doi.org/10.1016/0167-8655(89)90070-6
30. Kwon JS, Gi JW, Kang EK (2001) An enhanced thinning algorithm using parallel processing. IEEE Int Conf Image Process. https://doi.org/10.1109/icip.2001.958228
31. Boudaoud LB, Sider A, Tari A (2015) A new thinning algorithm for binary images. 2015 3rd International Conference on Control, Engineering & Information Technology (CEIT) 1–6. IEEE.
32. Li R, Zhang X (2018) Research on the Improvement of EPTA Parallel Thinning Algorithm. https://doi.org/10.2991/ncce-18.2018.167
33. Dong J, Chen Y, Yang Z, Ling BWK (2017) A parallel thinning algorithm based on stroke continuity detection. SIViP. https://doi.org/10.1007/s11760-016-1034-y
34. Gökmen M, Hall RW (1990) Parallel shrinking algorithms using 2-subfields approaches. Comput Vis, Gr Image Process 52:191–209. https://doi.org/10.1016/0734-189X(90)90054-Y
35. Ma CM, Wan SY, Der LJ (2002) Three-dimensional topology preserving reduction on the 4-subfields. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2002.1114851
36. Neusius C, Olszewski J, Scheerer D (1992) An efficient distributed thinning algorithm. Parallel Comput 18(1):47–55. https://doi.org/10.1016/0167-8191(92)90110-S
37. Kong TY (1995) On topology preservation in 2-D and 3-D thinning. Int J Pattern Recognit Artif Intell 9:813–844. https://doi.org/10.1142/S0218001495000341
38. Ronse C (1986) A topological characterization of thinning. Theoret Comput Sci 43:31–41. https://doi.org/10.1016/0304-3975(86)90164-7

39. Ronse C (1988) Minimal test patterns for connectivity preservation in parallel thinning algorithms for binary digital images. Discrete Appl Math 21:67–79. https://doi.org/10.1016/0166-218X(88)90034-0
40. Rosenfeld A (1975) A characterization of parallel thinning algorithms. Inf Control 29:286–291. https://doi.org/10.1016/S0019-9958(75)90448-9
41. Hall RW (1992) Tests for connectivity preservation for parallel reduction operators. Topol Appl 46:199–217. https://doi.org/10.1016/0166-8641(92)90015-R
42. Hall RW (1996) Parallel connectivity-preserving thinning algorithms. In Machine Intelligence and Pattern Recognition. North-Holland, Elsevier 145–179. https://doi.org/10.1016/S0923-0459(96)80014-0.