A FAST PARALLEL THINNING ALGORITHM FOR THE BINARY IMAGE SKELETONIZATION

Weian Deng

EMBARCADERO SYSTEMS, ALAMEDA, CALIFORNIA, U.S.A.

S. Sitharama Iyengar
Nathan E. Brener

DEPARTMENT OF COMPUTER SCIENCE, LOUISIANA STATE UNIVERSITY, BATON ROUGE, LOUISIANA, U.S.A.

Summary

This paper investigates the skeletonization problem using parallel thinning techniques and proposes a new one-pass parallel asymmetric thinning algorithm (OPATA$_8$). Wu and Tsai presented a one-pass parallel asymmetric thinning algorithm (OPATA$_4$) that implemented 4-distance, or city block distance, skeletonization. However, city block distance is not a good approximation of Euclidean distance. By applying 8-distance, or chessboard distance, this new algorithm improves not only the quality of the resulting skeletons but also the efficiency of the computation. This algorithm uses 18 patterns. The algorithm has been implemented, and has been compared to both algorithm OPATA$_4$ and Zhang and Suen's two-pass parallel thinning algorithm. The results show that the proposed OPATA$_8$ has good noise resistance, perfectly 8-connected skeleton output, and a faster speed without serious erosion.

Address reprint requests to S. Sitharama Iyengar, Department of Computer Science, 298 Coates Hall, Louisiana State University, Baton Rouge, LA 70803, e-mail iyengar@bit.csc.lsu.edu.

The International Journal of High Performance Computing Applications, Volume 14, No. 1, Spring 2000, pp. 65-81
© 2000 Sage Publications, Inc.

1 Introduction

This paper studies the problem of binary image skeletonization. Skeletonization is widely used in many image-processing applications, such as character recognition, chromosome analysis, and military route finding (Benton and Brink, 1990). It provides a convenient and condensed representation of image object information. Skeletons of objects can preserve topological properties of the objects (Benton and Brink, 1990) and reduce storage requirements (Levine 1984). The skeletonization problem has been studied extensively in the past 20 years. Many methods have been proposed in the literature. These methods can be classified into two categories: distance transform methods (Arcelli, Cordella, and Levialdi, 1981; Arcelli and Di BaJa, 1985; Suzuki and Abe, 1986; Xia, 1989) and parallel thinning methods (Chen and Tsai, 1990; Chin et al., 1987; Holt et al., 1987; Mendel, 1993; Stefanelli and Rosenfeld, 1971; Wu and Tsai, 1992; Zhang and Suen, 1984).

Distance transformation was proposed for binary pictures in the Euclidean plane by Blum (1978). The idea is that a fire line, which propagates with constant speed from the contour of an object to its inside, will meet at quench points that form the skeleton. Many discrete approximations of this fire propagation technique have been reported. Basically, these approximations were realized by sequentially tracing the object's boundary pixels to avoid going back to the area that had already burned (Arcelli and Di BaJa, 1985; Suzuki and Abe, 1986). Xia (1989) further refined the technique to construct the next fire front during the tracing of the current fire contour. These algorithms perform efficiently in a sequential machine, and the skeleton obtained can be used to recover the original objects. However, these algorithms have some serious drawbacks. They are sequential in nature and not easy to parallelize. Due to the properties of discrete space, they cannot guarantee that the topology of the objects will be preserved. In addition, the skeletons obtained are sensitive to local variations and noise (Xia, 1989).

The second type of algorithm is called a parallel thinning algorithm. These algorithms are parallel in nature. They use local neighborhood patterns as sufficient conditions to determine whether a pixel is a contour pixel that can be removed. This type of algorithm iteratively deletes contour pixels that are removable. The patterns selected should guarantee the connectivity of the resulting skeleton. There are many algorithms of this type that are based on the same principle. The differences are usually the sets of patterns they use. The advantage of this type of algorithm is that one can custom design the patterns to delete certain end points and, thereby, omit certain details to make the skeleton easier to interpret. This type of algorithm can capture the structural information of objects. Some people have argued that the majority of parallel thinning algorithms are time consuming compared with distance transformation methods. Although it is true that parallel thinning algorithms have a complexity proportional to the size of images and the maximal thickness of objects in the images (Xia, 1989), this is misleading because these algorithms are constructed to take advantage of parallelism. It is primarily the sequential versions of them that can be time consuming. These versions can be made more efficient by adopting the same contour-tracing technique employed by distance transformation approaches. However, the drawback of this type of skeletonization is that it may not preserve as much detail as distance transformation methods. Therefore, the resulting skeletons may not be recoverable.

Because our goal is to use skeletonization in military route planning, we are more concerned about the preservation of structural information of objects. Therefore, we focus on parallel thinning algorithms. For more detailed discussions on distance transformation methods, one can refer to Xia (1989). In the remainder of this paper, we will present a new one-pass parallel thinning algorithm. This new algorithm inherits the asymmetrical feature of Wu and Tsai's (1992) algorithm. However, based on the fact that, mathematically, 8-distance is a better approximation of Euclidean distance than 4-distance, we designed an 8-distance one-pass parallel asymmetric thinning algorithm (OPATA$_8$). The new algorithm not only improves the quality of the resulting skeletons but also speeds up the thinning process. Our new algorithm has been implemented and compared to both Wu and Tsai's OPATA$_4$ and Zhang and Suen's (1984) two-pass parallel thinning algorithm (TPTA). In Section 3, we will first review the idea of parallel thinning algorithms, especially Wu and Tsai's OPATA$_4$. Based on the observations on OPATA$_4$, Section 4 will present in detail our new algorithm OPATA$_8$. The experiments and the comparisons of OPATA$_8$ with other thinning algorithms will be discussed in Section 5.

2 Preliminaries

A binary image is defined as a matrix P where each element (pixel) is either 1 (black) or 0 (white). Objects in the image consist of black pixels.

Neighbors: For a pixel p in image P, the 8-neighbors of p are defined to be the eight pixels adjacent to p (p_0, $p_1, \ldots, p_7$ in Figure 1a) and denoted by $\mathcal{N}_8(p)$. Also, p_0, p_2, p_4, and p_6 are referred to as the set of 4-neighbors of p, $\mathcal{N}_4(p)$. p_8 and p_9 are two pixels that will be used to introduce asymmetry.

Distance: Between two pixels $p(x_p, y_p)$ and $q(x_q, y_q)$, where x_p, y_p, x_q, and y_q are coordinates, the 8-distance, or the chessboard distance, is defined as

$$d_8(p, q) = \max(|x_p - x_q|, |y_p - y_q|),$$

and the 4-distance, or the city block distance, is

$$d_4(p, q) = |x_p - x_q| + |y_p - y_q|.$$

Connectedness: A skeleton is considered to be 4(8)-connected if between any two black pixels p_0 and p_n, there exists a path $p_0 p_1 \ldots p_{i-1} p_i p_{i+1} \ldots p_n$ such that p_{i-1} is a 4(8)-neighbor of p_i for $1 \le i \le n$.

Neighbor sequence: A sequence of 8-neighbor pixels of p ($p_i, p_{i+1}, \ldots, p_{i+n}$) is called a neighbor sequence of p when $p_i, p_{i+1}, \ldots, p_{i+n}$ are 4-connected in a clockwise order around pixel p and they are all black pixels. For example, Figures 1b and 1c contain neighbor sequences $p_6 p_7 p_0$ and $p_4 p_5 p_6 p_7$, respectively, whereas in Figure 1d, $p_2 p_3 p_5$ is not a neighbor sequence because p_3 and p_5 are separated by a white pixel p_4.

Simple path: A path $\mathcal{P}$ is called a simple path if the removal of any pixel from the path except end pixels will violate the 4(8)-connectedness of the path when 4-distance (8-distance) is used.

Edge pixel: An edge pixel is a black pixel such that one of its 4-neighbors is a white pixel.

Convex corner pixel: A convex corner pixel is a black pixel such that two of its 4-neighbors p_i and p_{i+2} are white pixels. Figure 1b is an example. A convex corner pixel is an edge pixel.

Concave corner pixel: A concave corner pixel is a black pixel such that only one of its diagonal 8-neighbors is white (all other 8-neighbors are black). Figure 1e is an example.

End pixel: An end pixel is a black pixel such that only one of its 4(8)-neighbors is a black pixel when 4-distance (8-distance) is used.

Contour: A contour is the set of edge pixels.

Interior pixel: An interior pixel is a black pixel that does not belong to any contours.

Interior: The interior is the set of interior pixels of an object.

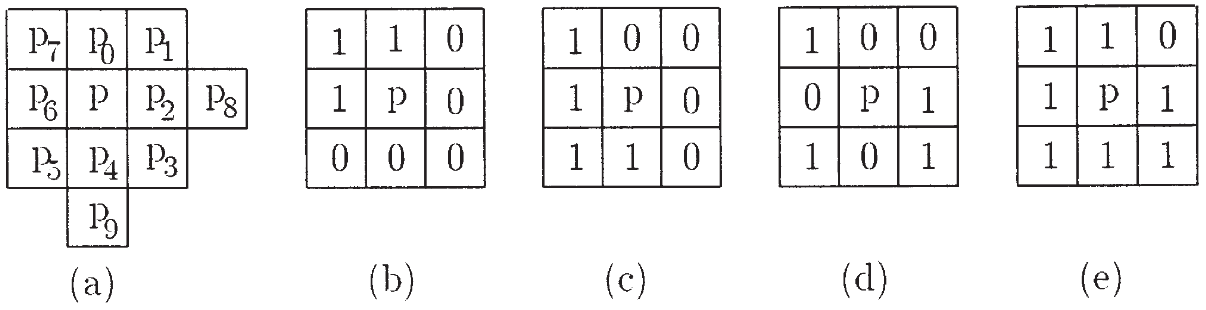


Fig. 1 4(8)-neighbors of a pixel

Thin object: A thin object is an object that has no interior pixels. A thin object is not necessarily a simple path. For example, lines that are two pixels wide are not simple paths.

3 Parallel Thinning Algorithms

Skeletonization is usually performed by iteratively removing edge pixels along the contour of image objects. The pixels erased must satisfy the following three criteria:

1. No end pixel is deleted.
2. No connectedness is violated.
3. No excessive erosion occurs.

Most of the proposed parallel thinning algorithms differ only in the way that they conduct the test to meet these criteria (Naccache and Shinghal, 1984). For the thinning algorithms before 1984, Naccache and Shinghal have given a detailed review and comparison. In the past 10 years, the most widely cited algorithm is that of Zhang and Suen (1984). This algorithm is called a two-pass algorithm because in each iteration, there are two passes, or subiterations. In the first pass, the conditions to remove pixel *p* are as follows:

1. It has a neighbor sequence of length between 2 and 6.
2. It is a northwest edge pixel or a southeast convex corner pixel.

The second pass deletes the southeast edge pixels and the northwest convex corner pixels that satisfy condition 1 above. The use of a two-pass iteration imposes a bias in

"Most of the proposed parallel thinning algorithms differ only in the way that they conduct the test to meet these criteria."

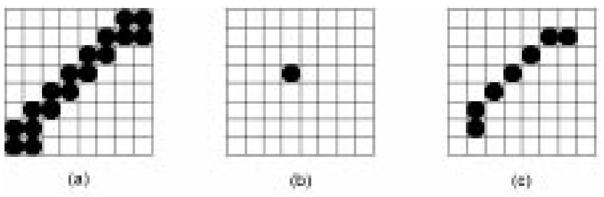


Fig. 2 **Erosion problem: (a) the input image, (b) the degraded result from Zhang and Suen's (1984) two-pass parallel thinning algorithm, and (c) the result from a one-pass parallel asymmetric thinning algorithm**

which northwest edge pixels and southeast convex corner pixels are preferred over other southeast edge pixels. This type of bias avoids excessive erosion.

Although Zhang and Suen's (1984) algorithm is good at connectivity and contour noise immunity, there are some disadvantages that need to be addressed. First, the algorithm still suffers from excessive erosion in some extreme conditions, such as two-pixel-wide diagonal lines (see Figure 2a). As shown in Figure 2b, when Zhang and Suen's algorithm is used to thin the image in Figure 2a, the original structure is totally destroyed. Second, although two-pass algorithms run faster than four-pass algorithms by reducing the time for scanning the image, the amount of time required to scan white pixels (which is a waste) is still significant. When we applied Zhang and Suen's algorithm to thin mobility maps in our route planner application, we noticed that, in some cases, half of the running time was spent on scanning white pixels. Furthermore, the resulting skeletons from Zhang and Suen's algorithm are not of unitary thickness.

Several attempts have been made to overcome these drawbacks. Lü and Wang (1986) restricted neighbor sequences to a length greater than three instead of greater than two. The improved algorithm preserved structures better but degraded the noise suppression feature. Holt et al. (1987) modified Zhang and Suen's (1984) algorithm from two-pass to one-pass. In general, the number of overall passes required to thin an image was reduced. However, they used a neighbor region of 5×5 that contains 25 pixels, which is much bigger than a 3×3 neighbor region. As a result, the algorithm turned out to be consistently slower than Zhang and Suen's algorithm. Another problem is that Holt et al.'s algorithm did not guarantee the perseverance of original patterns (Mendel, 1993). Mendel modified both Zhang and Suen's and Holt et al.'s algorithms to better preserve the original pattern in approximately the same amount of running time.

"Recently, much effort has been devoted to developing one-pass parallel thinning algorithms."

3.1 ONE-PASS PARALLEL THINNING ALGORITHM

Recently, much effort has been devoted to developing one-pass parallel thinning algorithms. The paper by Holt et al. (1987) was one of the attempts. Chin et al. (1987) came up with a more efficient one-pass algorithm that used mainly 3×3 operators. The algorithm used eight 3×3 thinning patterns to remove edge pixels and two restoring patterns (a 1×4 and a 4×1) to preserve continuity. Eight more patterns were employed to trim noise effects.

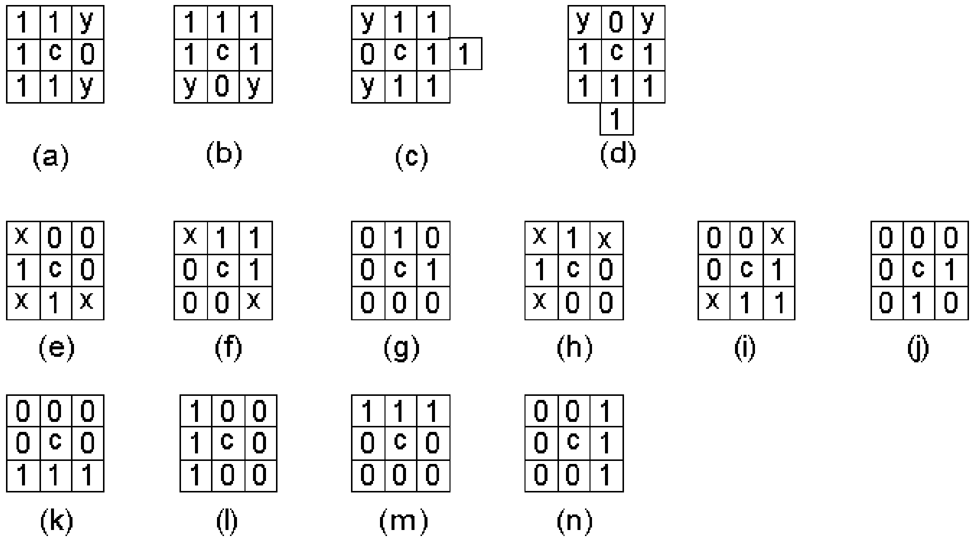


Fig. 3 **Wu and Tsai's (1992) 14 thinning patterns**

However, a major problem with this algorithm is that it generates biased skeletons. Convex corners are removed faster than concave corners, as shown in Figure 7b. Linear objects with a sharp turn generate skeletons that do not run along the Euclidean medial axis at the turn.

Wu and Tsai (1992) designed a new set of matching patterns that eliminated the need to distinguish between thinning patterns, restoring patterns, and trimming patterns. The set of 14 patterns are shown in Figure 3. In these patterns, *x* indicates that the pixel can be either 0 or 1, *y* indicates that in this pattern at least one of the *y*s is 0, and *c* indicates the current contour pixel that may be removed. Each black pixel whose neighbor area matches one of these patterns will be removed in the current pass.

This set of patterns was derived from the idea of asymmetry. When an object is thinned to a thin object (two pixels wide, generally), the pixels on one side of the object are removed according to preset preferences, whereas the pixels on the other side are retained. For example, patterns (a) and (c) are used to thin vertical lines. When a vertical line is not a thin object, both patterns will be used to thin the contour on both sides of the object, with pattern (a) thinning the right side and pattern (c) the left side. When the object is thinned to be a thin object, pattern (a) will continue to thin the contour pixels on the right side, whereas pattern (c) will leave the contour pixels on the left side intact. In the same way, patterns (b) and (d) will preferentially remove contour pixels on the top over those on the bottom for horizontal lines. Patterns (e) and (f) deal with diagonal contours that go from the top left corner to the bottom right corner, where pixels on top right side are thinned whereas those on the bottom left side remain. Pattern (g) is a complement pattern for pattern (f), since the condition in pattern (f), that the top right pixel p_1 must be black, excludes pattern (g). Patterns (h), (i), and (j), in a similar way, thin the diagonal contour that goes from the top right corner to the bottom left corner. As an example, in Figure 3, patterns (e), (f), (h), and (i) are used to get the resulting skeleton in Figure 2c from the input image in Figure 2a. Patterns (k), (j), (l), and (n) were designed to remove noise.

The algorithm presented by Wu and Suen is a pattern match algorithm. In parallel, all black pixels are checked against the 14 patterns. When any pattern matches a pixel's neighbor setting, the pixel is whited out (changed from value 1 to 0). This procedure continues until no more pixels can be thinned.

The advantages of Wu and Tsai's (1992) algorithm are its quickness and uniqueness. To our knowledge, it is one of the fastest parallel algorithms currently in use. The algorithm is unique. It treats all 14 patterns in the same way. It is also noise insensitive. It produces perfect 8-connected skeletons, and the resulting skeletons are quite isotropic in terms of city block distance. However, this algorithm inherits the major problem from which Chin et al.'s (1987) algorithm suffers, namely, that the resulting skeletons are biased, cutting corners. As a result, these skeletons do not preserve the structures of original objects as well as those from Zhang and Suen's (1984) algorithm.

4 OPATA$_8$

Both Chin et al.'s (1987) algorithm and Wu and Tsai's (1992) algorithm suffer from the different thinning speed at convex corners and concave corners. This results in deterioration of the skeleton's structural features. The problem comes from their discrete approximation of the Euclidean distance. Both algorithms implemented the city block distance (4-distance) approximation d_4. The city block distance from a convex corner pixel to a background area is always 1. Thus, the pixel is always considered to be an edge pixel and may be thinned in the current pass. A concave corner pixel, on the other hand, has a city block distance of 2 from the background area and can become an edge pixel only in the next pass. Figure 4b is an example where a convex corner is removed in the first pass and a concave corner is removed in the second pass. Figure 4a is the input image with five convex corners and one concave corner. In Figures 4b and 4c, a number in a pixel square indicates the pass in which the pixel is thinned.

This problem can be solved if we adopt the chessboard distance (8-distance). Both convex corner pixels and concave corner pixels have a chessboard distance of 1 and are thinned at the same speed. For example, in Figure 4b, all edge pixels and the concave pixel have d_8 equal to 1 and are removed in one pass. The result is a perfect skeleton (Figure 4c).

The patterns used by Wu and Tsai's (1992) OPATA$_4$ remove edge pixels including convex corner pixels. However, concave corner pixels are not edge pixels and, thus, cannot be removed by those 14 patterns. To develop an algorithm that has the chessboard distance approximation, we have to find a way to recognize concave corner pixels and to find the condition necessary to preserve connectedness when these pixels are removed.

4.1 LOCATING A CONCAVE CORNER

To recognize a concave corner pixel, 3×3 patterns are not the correct choice. The 3×3 patterns that could be used to locate a concave corner are those four patterns obtained from the rotation of Figure 5a. However, these four new patterns enhance noise if they are combined with the 14 existing patterns. Figure 5 is an example. In Figure 5c, pixels with black dots are part of a rectangular object with a white noise pixel at the center of its bottom. Applying the 14 patterns in Figure 3 along with the four 3×3 patterns from the rotation of Figure 5a once, the original white noise pixel becomes two new white noise pixels (see Figure 5d). The problem is that the pattern shown in Figure 5b is not a member of the thinning pattern set. However, this pattern cannot be added to the thinning pattern set because it always preserves the kind of white noise shown in Figure 5c. Thus, 3×3 patterns fail to provide enough information to locate a concave corner.

5×5 patterns could be used in this case because they provide more information. However, they require one to check a much larger neighborhood.

Each concave corner pixel has two adjacent edge pixels. For example, in Figure 5a, concave corner pixel c has

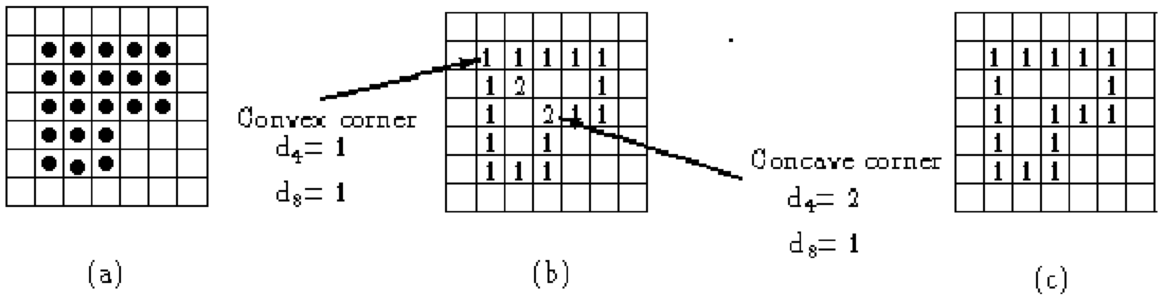


Fig. 4 **The d_4 and d_8 approximations of the Euclidean distance**

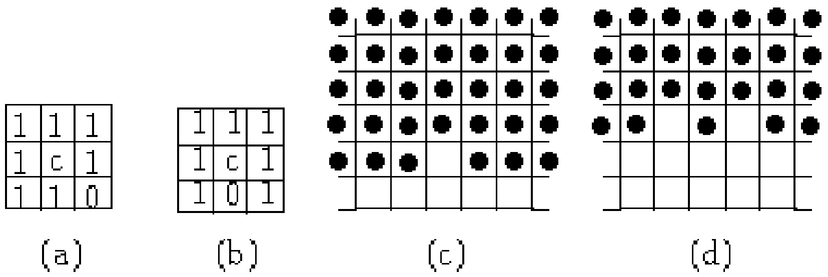


Fig. 5 **3×3 patterns are not suitable for locating a concave corner pixel**

two neighboring edge pixels: p_2 and p_4. When these two pixels are checked against the patterns and found to be edge pixels, it is possible to identify concave corner pixels such as pixel *c* in Figure 5a by a few additional checks. For example, if a pixel and its 3×3 neighbors match the pattern in Figure 3a and if pixel p_1 is a black pixel, then pixel p_0 will be a candidate for a concave corner pixel. Similarly, if pixel p_3 is a black pixel, then p_4 is another candidate. Because a concave corner pixel has exactly two adjacent edge pixels, a pixel marked twice as a concave corner candidate is a concave corner pixel. For example, in Figure 5a, the concave pixel *c* is a concave corner candidate when either pixel p_2 is checked against the pattern in Figure 3c (or 3h) and when pixel p_4 is checked against the pattern in Figure 3a (or 3h). Therefore, pixel *c* is a concave corner pixel. This method provides a means to locate concave corner pixels.

Of all the 14 patterns used in Wu and Tsai's (1992) algorithm, only patterns (a) to (f), (h), and (i) are related to the detection of concave corners. When a pixel matches one of these patterns, at most two more checks are required to decide whether one of its 4-neighbors is a concave corner candidate. The detection of concave corner pixels for pattern (a) has been discussed in the previous paragraph. A similar technique can be applied to patterns (b), (c), (d), (f), and (i). Patterns (e) and (h) require a different treatment. In pattern (e), for example, p_5 is an unspecified pixel that can be either 0 or 1. In addition to the condition $p_4 = 1$ (or $p_6 = 1$), to make pixel p_4 (or p_6), a concave corner, p_5, must also be 1 so that the object is at least two pixels wide. Thus, for pattern (e), two checks are needed to determine a concave corner. To summarize, the following is the checklist for these eight patterns:

Pattern (a): If pixel $p_1 = 1$ then p_0 is a concave corner pixel
If pixel $p_3 = 1$ then p_4 is a concave corner pixel
Pattern (b): If pixel $p_3 = 1$ then p_2 is a concave corner pixel
If pixel $p_5 = 1$ then p_6 is a concave corner pixel
Pattern (c): If pixel $p_5 = 1$ then p_4 is a concave corner pixel
If pixel $p_7 = 1$ then p_2 is a concave corner pixel
Pattern (d): If pixel $p_1 = 1$ then p_2 is a concave corner pixel
If pixel $p_7 = 1$ then p_6 is a concave corner pixel
Pattern (e): If pixel $p_7 = 1$ and $p_5 = 1$ then p_6 is a concave corner pixel
If pixel $p_5 = 1$ and $p_3 = 1$ then p_4 is a concave corner pixel
Pattern (f): If pixel $p_7 = 1$ then p_0 is a concave corner pixel
If pixel $p_5 = 1$ then p_2 is a concave corner pixel
Pattern (h): If pixel $p_7 = 1$ and $p_1 = 1$ then p_0 is a concave corner pixel
If pixel $p_7 = 1$ and $p_5 = 1$ then p_6 is a concave corner pixel
Pattern (i): If pixel $p_1 = 1$ then p_2 is a concave corner pixel
If pixel $p_5 = 1$ then p_4 is a concave corner pixel

To locate a concave corner pixel, a marker is introduced for each pixel. As discussed above, each concave pixel *c* is visited by both of its neighboring edge pixels (e.g., Figure 5a). When the corner is first found to be a concave corner candidate (suppose by pixel p_2 in this example), its marker is set. Thus, when the second neighboring edge pixel (pixel p_4) also finds that pixel *c* is a concave corner candidate while the marker has already been set, the corner will be designated as a concave corner pixel.

4.2 PRESERVING CONNECTEDNESS

Not all the concave corner pixels detected are eventually thinned in a pass, since this would lead to discontinuity due to the symmetric feature of one-pass algorithms. To preserve connectedness, an asymmetric treatment for convex corners over concave corners is needed. For example, Figure 6a is the general situation after pixel *c* is recognized as a concave corner pixel. The pixels marked by 0, 1, and *x* are the neighboring pixels that have already been checked when both of pixel *c*'s 4-neighbor edge pixels are processed. *x* means that the pixel is either 0 or 1 according to a particular situation. The 3×3 area with heavy shadow is examined when the pixel to the right of pixel *c* is processed. At the time we process the pixel below pixel *c*, the area with light shadow is checked. The pixels marked *u* have not been checked and are still unknown. If all of these *u* pixels are white pixels, the removal of pixel *c* will cut the object into two halves, which is not desired. In general, we have the following observations:

Remark I: If an object is a thin object (less than two pixels wide), the removal of concave corner pixels would introduce discontinuity.

Because for three-pixel-thick (or four) diagonal lines, all the interior pixels are concave pixels and can all be removed in one pass, it is sufficient to state the following:

Remark II: Three-pixel-thick (or four) diagonal lines will be removed in a single pass if all concave corner pixels are deleted.

One method to avoid these problems is to stop thinning concave corner pixels when that portion of an object is thin. To determine whether the corner area is thin, all of the pixels marked *u* in Figure 6a, except the one at the bottom right corner, need to be checked. This requires 10 more checks, which is quite expensive. Because our goal is to preserve connectedness in an efficient manner, we further relax the condition to only 4 checks. Pattern (o) (Figure 6b) is one of the four new patterns we add to the thinning pattern set to guarantee that no connectedness is violated when a pixel is thinned. In pattern (o), *x* indicates the pixel is unspecified, that is, it can have a value of either 0 or 1. The reason we designate pixels (0, 0), (0, 2), (1, 1), and (2, 0) as unspecified is that once pixels (0, 1), (0, 3), (1, 0), and (3, 0) are all black pixels, pixels (1, 2) and (2, 1) will not be thinned in this pass no matter what values pix-

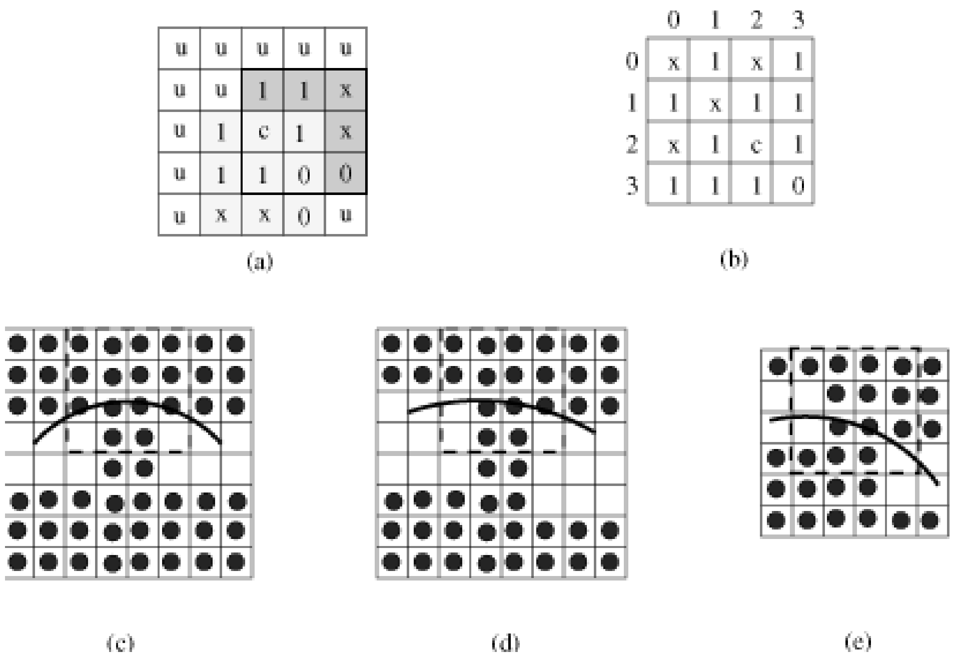


Fig. 6 **(a) An example of concave corner pixel detection, (b) a pattern for removing concave corners, (c) an example that introduces discontinuity when a check for object pixel at (0, 2) replaces the check for object pixel (0, 3), (d) an example that introduces discontinuity when the check for object pixel at (0, 3) is dropped, (e) an example that introduces discontinuity when the checks for object pixels at (0, 1) and (1, 0) are replaced by the checks at (0, 0) and (1, 1)**

els (0, 0), (0, 2), (1, 1), and (2, 0) have. Connectedness at this point is preserved by pixels (1, 2) and (2, 1) even when concave corner pixel *c* is removed. Pattern (o) is the pattern to be used in the situation like Figure 6a. Three other patterns, (p), (q), and (s), are required to make the algorithm complete. They are from the rotation of pattern (o).

In pattern (o), the four checks for black pixel at (0, 1), (1, 0), (0, 3), and (3, 0) are minimal. If the check at (0, 3) is replaced by a check at (0, 2) for a black pixel, Figure 6c is a counter example. The area surrounded by dashed lines is checked against the modified pattern (o). In a single pass, the four pixels crossed by the curve will be thinned. (The left-most pixel and the right-most pixel are edge pixels. The other two pixels are both concave corner pixels.) The object will be divided into two. If we simply dismiss the check at (0, 3), Figure 6d is another counter example. Therefore, pixel (0, 3) has to be a black pixel. Based on the same argument, pixel (3, 0) has to be checked. If the checks at (1, 0) and (0, 1) are replaced by a single check at (0, 0), Figure 6e is a counter example.

Table 1
Algorithm OPATA

```
    Input: A binary image $I_0$
    Output: The skeleton $S$ of $I_0$ after thinning
 1. $i = 0$;
 2. do {
 3.       $flag = false$; $i = i + 1$;
 4.     for all (pixel $p$ in $I_{i-1}$ {
 5.         $I_i(p) = I_{i-1}(p)$;
 6.         if ($p = 1$ && $p$'s neighbors match one of the patterns from (a) to (n)) {
 7.             $p = 0$; $flag = true$;
 8.             for ($p_c$ is a concave corner candidate next to $p$)
 9.                 if ($p_c$ is not marked) mark $p_c$;
10.                 else
11.                     if ($p_c$'s neighbors match pattern (o)) $p_c = 0$;
        }
      }
12. } while ($flag$);
13. $S = I_i$;
```

Therefore, when a pixel is found to be a concave corner pixel, four more checks are required to guarantee connectedness. If all of these four pixels are black pixels, the concave corner pixels will be removed.

4.3 THE NEW THINNING ALGORITHM

Based on the above discussion, our OPATA$_8$ is constructed in the following way. In each pass, Wu and Tsai's (1992) 14 thinning patterns are employed to thin convex corner pixels and other edge pixels. When a pixel matches pattern (a)-(f), (g), or (i) (and thus is going to be thinned), the identification procedure described in Section 4.1 is applied to locate concave corner pixels among its four 4-connected neighbors. If a concave corner pixel is found, it will be checked against patterns (o), (p), (q), or (s) to ensure connectedness. When a match is found, the concave corner pixel is removed. To locate concave corners, a marker matrix is needed as an intermediate data structure. Each pixel in an image has its corresponding marker. A pseudocode description of the algorithm is given above.

"To locate concave corners, a marker matrix is needed as an intermediate data structure."

In the algorithm (Table 1), I_i, $i = 1, 2, 3, \ldots$, is the intermediate result after the ith pass. Step 4 processes all pixels simultaneously. Step 5 copies the intermediate thinned image from the $i - 1$ pass. The search for edge pixels that can be thinned is performed in step 6. Step 8 uses the condition listed in Section 4.1 to determine whether any of p's 4-neighbors is a concave corner pixel candidate and then processes each one that is found. Step 11 checks the neighbors of p_c against pattern (o). The algorithm stops when no more pixels can be thinned.

Let $B(I)$ be the number of black pixels in image I. Because edge pixels and concave corner pixels are removed from image I_{i-1} to get image I_i, $B(I_{i-1}) \geq B(I_i)$ for $i = 1, 2, 3, \ldots$, and the $B(I_i)$s keep decreasing. Because an image has a finite number of black pixels that can be thinned, eventually, the algorithm will stop when $B(I_{i-1}) = B(I_i)$.

Theoretically, since

$$d_8(p, q) \leq d_4(p, q)$$

for any two pixels p and q in an image, the 8-distance from an object pixel to the background is at most the same as its 4-distance, and most of the time the 8-distance is smaller. Therefore, a thinning algorithm that implements 8-distance, such as OPATA$_8$, will require an equivalent, if not smaller, number of passes to converge. This fact is verified in the experiments discussed in Section 5. Also, an 8-distance thinning algorithm better preserves the structural features of objects. However, since the removal of a concave corner requires extra effort (at most two more checks to locate the corners and four more checks to ensure connectedness), each pass of OPATA$_8$ is expected to take slightly longer than a corresponding pass of OPATA$_4$.

When implemented in parallel, the markers may be accessed concurrently by a pixel's four 4-connected neighbors. This causes the problem of read/write conflict. However, this problem can be solved. If the algorithm is implemented on single instruction multiple data (SIMD) architectures, then, assuming that each processing unit processes a pixel, the conflict problem can be avoided if all the processing units check their 8-neighbors in the same order, that is, $p_0, p_1, p_2, p_3, p_4, p_5, p_6$, then p_7. Hence, no concurrent access will occur. On multiple instruction multiple data (MIMD) platforms, each marker has to be treated as a shared piece of data and accessed exclusively. The order in which the pixel's four 4-connected neighbors check and change the marker has no effect on the result.

OPATA$_8$ and OPATA$_4$ are both one-pass algorithms. Their implementations differ in the sets of templates they use. The SIMD implementation and MIMD implementation described above can naturally be applied to OPATA$_4$. However, the parallel implementation of TPTA is different because it is a two-pass algorithm. The two passes have to be implemented in sequence. The discussion of the parallel implementation of TPTA is beyond the scope of this paper.

5 Experiments

We implemented the proposed algorithm (OPATA$_8$) along with Zhang and Suen's (1984) TPTA and Wu and Tsai's (1992) OPATA$_4$. All three algorithms were programmed using the decision tree method. Based on the sets of templates for OPATA$_8$ and OPATA$_4$, we developed the decision trees for both algorithms. Using the decision trees, the algorithms determine whether a pixel is an edge pixel that should be removed. This paper concentrates on the advantages of OPATA$_8$ versus OPATA$_4$. The detailed implementation will be presented in a future paper. Here, we focus on experimental results that compare these thinning algorithms.

The experimental results confirm the improvement of our new algorithm over the two existing algorithms in both the quality of the results and the speed. In this section, we will compare the qualities and the speeds of these three methods.

A comparison of the results from the three algorithms over several binary images are shown below. All the input images are displayed as shadow areas. The corresponding skeletons are shown in black color. In all of the following figures, (a), (b), and (c) are the results of our OPATA$_8$, Wu and Tsai's (1992) OPATA$_4$, and Zhang and Suen's (1984) TPTA, respectively. The comparisons focus on five aspects of the algorithms:

1. The preservation of structures of original objects
2. Erosion of the resulting skeletons
3. Noise resistance
4. 8-connectedness of the resulting skeletons
5. Localization

Figure 7 shows the significant difference between 4-distance and 8-distance based thinning algorithms with regard to the preservation of the structure of a corner. The input image is 24×50 in dimension. OPATA$_8$, which is an 8-distance–based algorithm, produces a skeleton that perfectly captures the corner feature. The output from OPATA$_4$ has a significant deterioration at the corner because it implements 4-distance. The small deterioration of TPTA in this case comes from its special treatment for corners.

The skeletons of the letter H (Figure 8) provide another excellent example where OPATA$_8$ and TPTA get the same

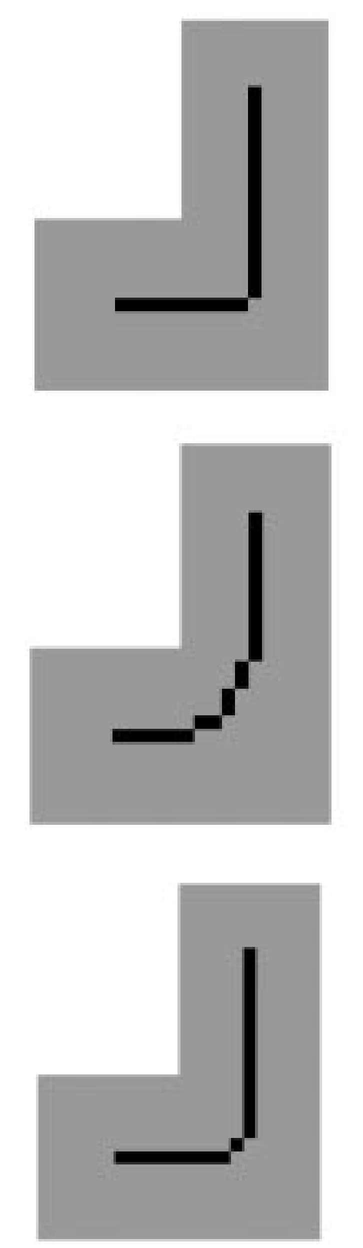


Fig. 7 The difference between 8-distance thinning and 4-distance thinning

results that well capture the structure of the letter H, whereas the skeleton from OPATA$_4$ misses at the corners.

TPTA has been found to have serious erosion problems. OPATA$_4$ overcomes this problem using asymmetric patterns. OPATA$_8$ successfully inherits this useful feature. Figure 9 is a typical case. In Figure 9c, which is the result from TPTA, the skeleton of the letter X deteriorates to a three-pixel-long bar. In contrast, the results from OPATA$_4$ (Figure 9b) and OPATA$_8$ (Figure 9a), which are the same in this case, preserve the original structure. Figure 2 is another example of both OPATA$_4$ and OPATA$_8$'s ability to resist erosion.

Figure 10 is a bar of 15 pixels wide and 46 pixels long. Noises are added to it along its edges and at its corners. The results indicate that all three algorithms are good at resisting noise on the edges. For the noise at the corners, TPTA has the best performance, which suppresses all but one kind of corner noise (at the top left corner). OPATA$_4$ and OPATA$_8$ fail to suppress corner noise. However, there is a trade-off between good erosion resistance and good noise suppression. One way to reduce corner noise for OPATAs is to have a preprocessing stage to smooth corner noise.

Figure 11 is the result of a comparison on localization. Theoretically, 8-distance is a better approximation of Euclidean distance than 4-distance. As a result, the output of an 8-distance medial axis transformation has a better localization than that of a 4-distance transformation. The results from the experiment support this argument. In all the figures, the synthetic object is shown as the background. Figure 11a presents the 4-distance medial axis (in light gray) along with the Euclidean medial axis (in dark gray). Figure 11b shows the comparison of the output from an 8-distance medial axis transformation (in light gray) with the Euclidean medial axis. The black pixels are the common pixels for both axes in a figure. The 8-distance medial axis is closer to the Euclidean medial axis than the 4-distance medial axis.

Figures 11c, 11d, and 11e are the output from TPTA, OPATA$_4$, and OPATA$_8$ (in dark gray) in comparison with the 8-distance axis, the 4-distance axis, and the 8-distance axis (in light gray), respectively. All three algorithms are able to extract the major segment of the corresponding medial axis, 8-distance for TPTA and OPATA$_8$ and 4-distance for OPATA$_4$. Because the 8-distance medial axis is closer to the Euclidean medial axis, TPTA and OPATA$_8$ have better localization.

However, all three algorithms fail to obtain the branches that lead from the main axis to the rectangular corners. These results are reasonable because all these al-

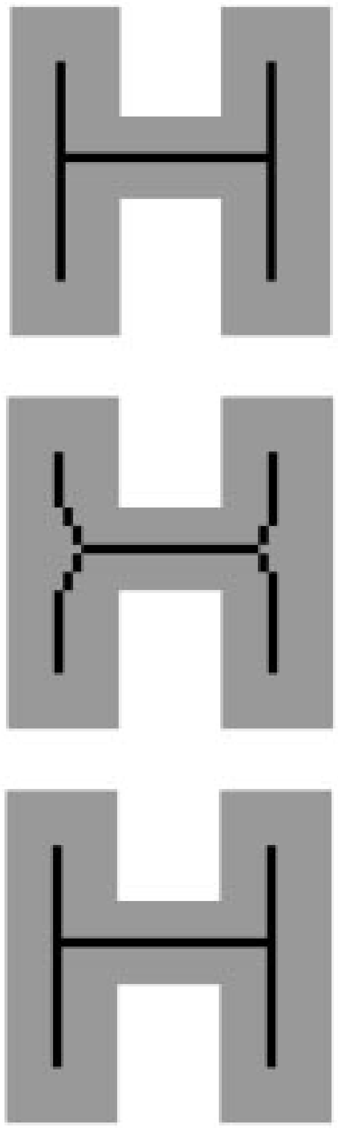


Fig. 8 The letter H

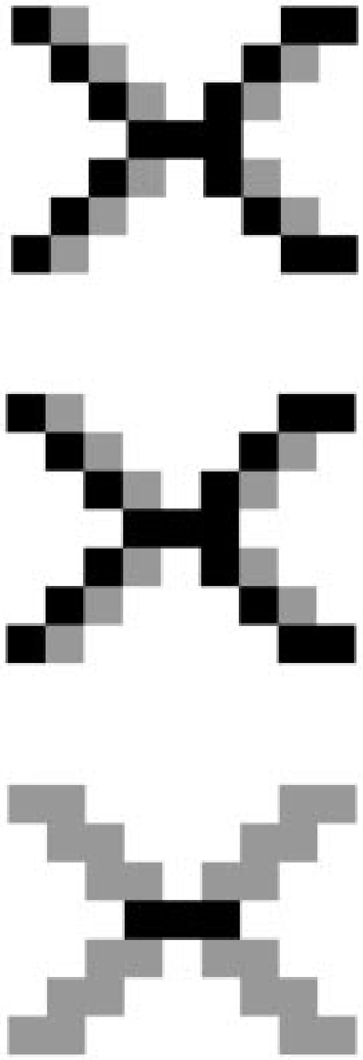


Fig. 9 Another example of erosion

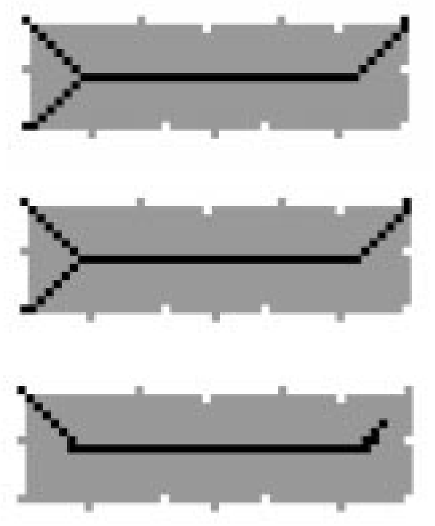


Fig. 10 Noise suppression

gorithms are customized to trim these branches and to extract clear and simple skeletons. If necessary, we can custom design these algorithms to obtain the real medial axis. Replacing patterns (e), (f), (h), and (i) with patterns (e′), (f′), (h′), and (i′) in Figure 12, the output of OPATA$_8$ matches the 8-distance medial axis perfectly (Figure 11f). In any of these patterns, at least one "z" pixel has to be black.

Figures 13 and 14 are two more examples that show the high quality of the skeleton produced by OPATA$_8$. In both examples, OPATA$_8$ and OPATA$_4$ get perfect 8-connected skeletons that are unitarily connected. However, the results of TPTA are not unitary; rather, the pixels are 4-connected in some place. In Figure 14, one can see that the noise suppression of TPTA is better than that of OPATA$_8$, whereas the result of OPATA$_4$ is noisier than that of the OPATA$_8$.

We implemented all three algorithms in C on a DECstation 5000/240. Table 2 gives the times for the examples discussed earlier. For a given method and a given image, the running time shown in the table, which is given in mil-

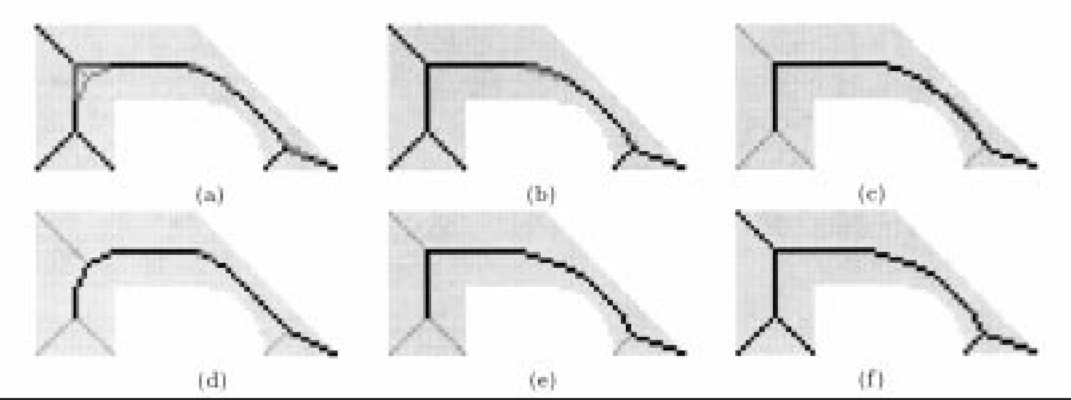
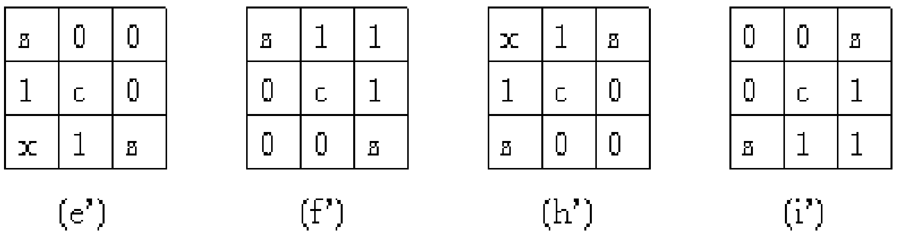


Fig. 11 A comparison of localization for parallel thinning algorithms: (a) The 4-distance medial axis and the Euclidean medial axis, (b) the 8-distance medial axis and the Euclidean medial axis, (c) the output of TPTA and the 8-distance medial axis, (d) the output of OPATA$_4$ and the 4-distance medial axis, (e) the output of OPATA$_8$ and the 8-distance medial axis, (f) the output of the modified OPATA$_8$ and the 8-distance medial axis



z	0	0
1	c	0
x	1	z

(e')

z	1	1
0	c	1
0	0	z

(f')

x	1	z
1	c	0
z	0	0

(h')

0	0	z
0	c	1
z	1	1

(i')

Fig. 12 Patterns to extract detail skeletons

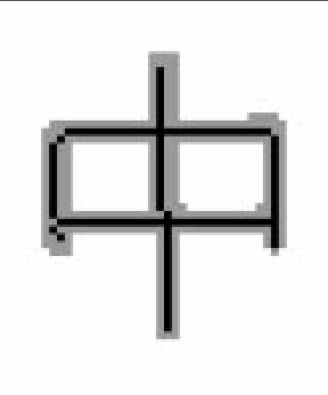
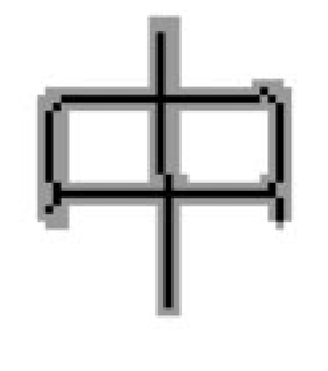
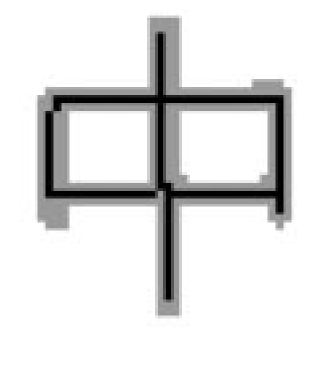




Fig. 13 A Chinese character

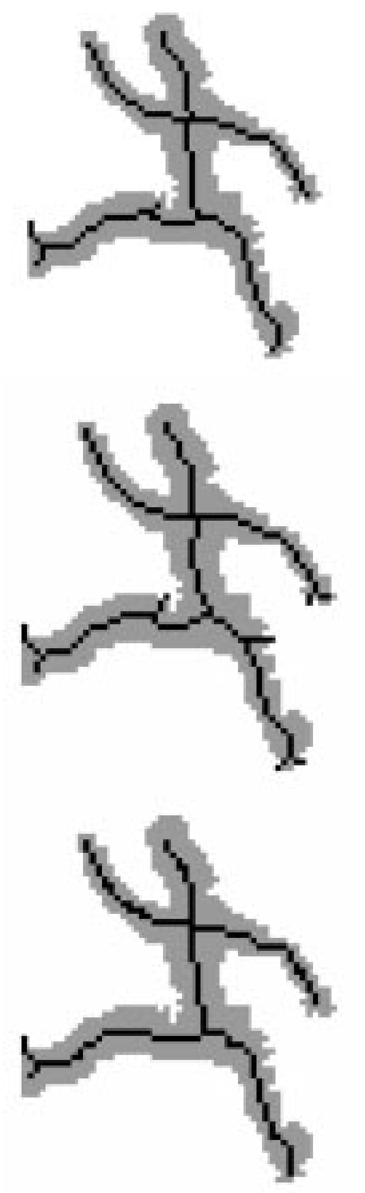


Fig. 14 A walking man

Table 2
A Comparison of Running Times (the last three lines give running time/passes)

Time/Passes	Corner	Letter H	Erosion	Noise	Chinese Character	Walking Man
Size	24×30	40×40	8×9	50×30	40×50	55×65
TPTA	15.08/14	34.41/14	0.98/8	21.01/14	13.91/6	47.97/12
$OPATA_4$	11.64/9	27.30/9	0.35/2	15.58/8	10.04/4	35.54/8
$OPATA_8$	10.59/7	25.08/7	0.35/2	16.17/8	10.09/4	33.94/6

liseconds, is the average time over 100 repeated executions. The number of passes is the number of times the program scans over the image in the thinning process. The results show that $OPATA_8$ and $OPATA_4$ are significantly faster than TPTA in all cases. When $OPATA_8$ requires fewer passes than $OPATA_4$ does, $OPATA_8$ is faster. When both require the same number of passes, $OPATA_4$ is slightly faster, since the search for concave corners takes extra time.

6 Conclusions

In this paper, we have proposed a new one-pass parallel asymmetric thinning algorithm called $OPATA_8$. Because of the implementation of the chessboard distance (8-distance), the resulting skeletons preserve the topological information and structural information of the input image better than those from 4-distance thinning algorithms. In our cross-country route-planning system, this feature is very important. It significantly improves the selection of optimal paths. In addition, $OPATA_8$ is faster than many other thinning algorithms as a result of reducing both the number of passes and the actual running time. This proposed algorithm has the feature of good noise resistance, better localization, and unitary 8-connected skeleton output. Recently, this algorithm was used successfully in an automated battlefield analysis system.

"This proposed algorithm has the feature of good noise resistance, better localization, and unitary 8-connected skeleton output."

ACKNOWLEDGMENTS

This work is supported by the U.S. Army Research Office. Special thanks to John Benton for reviewing this paper.

BIOGRAPHIES

Weian Deng is a research scientist at Embarcadero Systems in Alameda, California. He received his Ph.D. degree in Computer Science at Louisiana State University and has published papers in the area of parallel algorithms, vision, and image processing.

S. Sitharama Iyengar is the chairman of the Computer Science Department and a professor of computer science at Louisiana State University. His publications include several books and over 250 research papers in areas of high-performance parallel and distributed algorithms and data structures for image processing and pattern recognition, autonomous navigation, and distributed sensor networks. He is also a series editor for *Neuro-Computing of Complex Systems* and an editor for the *Journal of Computer Science and Information*.

Nathan E. Brener is a faculty member in the Department of Computer Science and Louisiana State University. He has published more than 30 papers in the areas of physical sciences and algorithms.

REFERENCES

Arcelli, C., and G. S. Di BaJa. 1985. A width-independent fast thinning algorithm. *IEEE Trans PAMI* 7:463-74.

Arcelli, C., L. P. Cordella, and S. Levialdi. 1981. From local maxima to connected skeletons. *IEEE Trans PAMI* 3:134-43.

Benton, J. R., and A. Brink. 1990. Hierarchical route planner. U.S. Army Engineer Topographic Laboratories, Fort Belvoir, Virginia.

Blem, H., and R. N. Naggel. 1978. Shape description using weighted symmetric axis features. *Pattern Recognition* 10:167-80.

Chen, C., and W. Tsai. 1990. A new fast one-pass thinning algorithm and its parallel hardware implementation. *Pattern Recognition Letters* 11:471-77.

Chin, R. T., H. Wan, D. L. Stover, and R. D. Iverson. 1987. A one-pass thinning algorithm and its parallel implementation. *Computer Vision, Graphics, and Image Processing* 40 (1): 30-40.

Holt, C. M., A. Stewart, M. Clint, and R. H. Perrott. 1987. An improved parallel thinning algorithm. *Comm ACM* 30:156-60.

Levine, D. 1984. *Vision in man and machine*.

Lü, H. E., and P.S.P. Wang. 1986. A comment on “A fast parallel algorithm for thinning digital patterns.” *CACM* 29:239-42.

Mendel, G. 1993. Optical character recognition using morphological attributes. Dissertation, Department of Computer Science, Louisiana State University.

Naccache, N., and R. Shinghal. 1984. SPTA: A proposed algorithm for thinning binary patterns. *IEEE Trans SMC* 14:409-18.

Stefanelli, R., and A. Rosenfeld. 1971. Some parallel thinning algorithm for digital pictures. *JACM* 18:255-64.

Suzuki, S., and K. Abe. 1986. Sequential thinning of binary pictures using distance transformation. *Proc 8th ICPR*, Paris, 289-92.

Wu, R.-Y., and W.-H. Tsai. 1992. A new one-pass parallel thinning algorithm for binary images. *Pattern Recognition Letters* 13:715-23.

Xia, Y. 1989. Skeletonization via the realization of the fire front’s propagation and extinction in digital binary shapes. *IEEE Trans PAMI* 11:1076-86.

Zhang, T., and C. Suen. 1984. A fast parallel algorithm for thinning digital patterns. *CACM* 27:236-39.