A Fully Parallel 3D Thinning Algorithm and Its Applications

C. MIN MA

Department of Industrial Engineering and Management, MingChi Institute of Technology, Taipei, Taiwan 243

AND

MILAN SONKA*

Department of Electrical and Computer Engineering, University of Iowa, Iowa City, Iowa 52242

Received March 24, 1995; accepted October 12, 1995

A thinning algorithm is a connectivity preserving process which is applied to erode an object layer by layer until only a “skeleton” is left. Generally, it is difficult to prove that a 3D parallel thinning algorithm preserves connectivity. Sufficient conditions which can simplify such proofs were proposed recently in CVGIP: Image Understanding (59, No. 3 (1994), 328–339). One of the purposes of this paper is to propose a connectivity preserving fully parallel 3D thinning algorithm. The other purpose is to show how to use the sufficient conditions to prove a 3D parallel thinning algorithm to be connectivity preserving. By this demonstration, a new generation of 3D parallel thinning algorithms can be designed and proved to preserve connectivity relatively easily.

1. PRELIMINARY

1.1. Introduction

Thinning on binary images is an iterative process which erodes an object layer by layer until only a “skeleton” of the object remains. A thinning algorithm contains a set of deletion conditions which are applied iteratively to delete 1’s (object elements), that is, to change 1’s to 0’s (background elements).

Geometry preservation is a major concern of thinning algorithms. For example, an object like “b” should not be converted into an object like “o.” To preserve the geometry of the original image, a thinning algorithm may contain some preserving conditions. A 1 is preserved; i.e., it cannot be deleted, if it satisfies any of these preserving conditions. For example, a well-known preserving condition is to preserve the endpoints of a unit-width arc. Different approaches of the corresponding concept for the 3D case were proposed (see [15, 23]). We are not going to review these approaches. The concept of geometry preservation is rather vague, especially compared to the exact concept of connectivity preservation given below.

Connectivity preservation is the second major concern of thinning algorithms. Unlike geometry preservation, connectivity preservation can be defined mathematically, see Definition 1.3. For example, an object like “o” should not be thinned into an object like “c.” To establish a connectivity preservation proof, one must show, for example, that no connected object in an image is split or completely deleted and that no hole in an object is created or eliminated.

Generally, it is straightforward to show that a sequential thinning algorithm preserves connectivity since such an algorithm deletes a single 1 at a time. A parallel thinning algorithm could delete a set of 1’s in each iteration. It can be tricky to show that a parallel thinning algorithm preserves connectivity since two or more adjacent 1’s can be deleted at the same time. It should be noticed that even if a parallel thinning algorithm is only allowed to delete simple points (see Definition 1.1 below), the algorithm may not preserve connectivity. For example, in Fig. 1 the deletion of either $p$ or $q$ alone does not change the connectivity of the object. But the deletion of both $p$ and $q$ does. Sufficient conditions for proving that a 2D thinning algorithm is connectivity preserving were proposed by Rosenfeld [20], Hall [6], and Ronse [19]. The same problem was studied by Hall [5] for the 3D case. General results for the 3D case were recently proposed in [12].

The first objective of this paper is to propose a connectivity preserving fully parallel thinning algorithm for 3D images. Our design was motivated by Holt et al.’s 2D thinning algorithm [8]. The proof that this algorithm preserves con-
connectivity uses the results originally published in [12]. The second objective of this paper is to demonstrate how to establish connectivity preservation proofs using the results presented in [12]. We believe that our approach shows a possible way for constructing a new generation of 3D thinning algorithms.

1.2. Basic Definitions

Let \( r \) and \( s \) be two distinct points with coordinates \((r_x, r_y)\) and \((s_x, s_y)\), respectively, in the discrete space \( Z^2 \). Then \( r \) and \( s \) are:

- **4-adjacent** if \( |r_x - s_x| + |r_y - s_y| = 1 \); and
- **8-adjacent** if \( 1 \leq |r_x - s_x| + |r_y - s_y| \leq 2 \) and
  \[ \max(|r_x - s_x|, |r_y - s_y|) = 1. \]

Let \( p \) and \( q \) be two distinct points with coordinates \((p_x, p_y, p_z)\) and \((q_x, q_y, q_z)\), respectively, in the discrete space \( Z^3 \). Then \( p \) and \( q \) are:

- **6-adjacent** if \( |p_x - q_x| + |p_y - q_y| + |p_z - q_z| = 1 \);
- **18-adjacent** if \( 1 \leq |p_x - q_x| + |p_y - q_y| + |p_z - q_z| \leq 2 \) and
  \[ \max(|p_x - q_x|, |p_y - q_y|, |p_z - q_z|) = 1; \]
- **26-adjacent** if \( \max(|p_x - q_x|, |p_y - q_y|, |p_z - q_z|) = 1. \)

A 3D binary digital image (or briefly, a 3D image) \( P \) is embedded in \( Z^3 \); each element of \( Z^3 \) is called a point of \( P \). Each point of \( P \) is assigned either a value of 1 (an object point) or a value of 0 (a background point).

We assume every image contains finitely many object points. Two object points are **adjacent** if they are 26-adjacent; two background points, or one background point and one object point are **adjacent** if they are 6-adjacent. Two points are \( k \)-**neighbors** to each other if they are \( k \)-adjacent. Let \( p \) be a point in \( P \). The \( k \)-neighborhood of \( p \) is the union of \( p \) and all points that are \( k \)-adjacent to \( p \). Let \( N(p) \) be defined as the 26-neighborhood of \( p \). By “\( p = 0 \)” we mean that \( p \) is a background point, and by “\( p = 1 \)” we mean that \( p \) is an object point. For all figures in this paper, we use a \( \bullet \) to denote an object point, and a \( \bigcirc \) to denote a background point.

An object point of \( P \) is called a **border point** if it is 6-adjacent to a background point of \( P \). A **unit edge** is a set of two points of \( P \) of distance 1; a **unit square** is a set of four corners of a 1 \( \times \) 1 square in \( P \); a **unit cube** is a set of eight corners of a 1 \( \times \) 1 \( \times \) 1 cube in \( P \). The concepts of unit squares and unit cubes are very important in this paper, since whether a thinning algorithm preserves connectivity can be verified by checking all configurations of a unit cube and a unit square (see Proposition 1.9 below).

Let \( X \) be a set of points of \( P \). A **k-path** in \( X \) is a sequence of distinct points of \( X \) in which every two subsequent points are \( k \)-adjacent. Two object points are **26-connected** if there exists a 26-path of object points joining them. An **object** on \( X \) is a maximal 26-connected subset of the set of object points of \( X \). Similarly, a **background component** on \( X \) is a maximal 6-connected subset of the set of background points of \( X \). Two points, \( p \) and \( q \), of \( P \) are **diagonally adjacent** if they are 18-, but not 6-adjacent and **diametrically adjacent** if they are 26-, but not 18-adjacent. An object point is said to be **simple** if it satisfies the following definition.

**Definition 1.1** [21, 13]. Let \( p \) be an object point of a 3D image. Then \( p \) is called a **simple** point if

1. \( p \) is 26-adjacent to only one object in \( N(p) \) \( \setminus \{p\} \); and
2. \( p \) is 6-adjacent to only one background component in \( p \)'s 18-neighborhood.

An object point which is not simple is called a **non-simple point**.

Since a parallel thinning algorithm may delete more than one point in one iteration, we need a definition for the deletion of a set of object points to be connectivity preserving. Kong in [9] suggested a general definition for 3D images as follows.

**Definition 1.2.** Let \( D \) be a set of object points of a 3D image. Then

\[ \text{FIG. 2. The } x-, y-, \text{and } z\text{-axes, and the arrangement of each orientation.} \]
1. *D* is called a **simple sequence** if *D* can be ordered as a sequence in which every point is simple after all its predecessors in the sequence are deleted;

2. *D* is called a **simple set** if *D* can be ordered as a simple sequence; and

3. the deletion of *D* from the image is said to **preserve connectivity** if *D* is a simple set.

Note that Ronse [18] showed that for the 2D case, the deletion of a set of object points *D* preserves connectivity if and only if *D* can be ordered as a simple sequence. We still need a precise definition of what it means for a parallel thinning algorithm to preserve connectivity. The following definition was proposed by Kong in [9].

**Definition 1.3.** A 3D parallel thinning algorithm is said to **preserve connectivity** if it is only allowed to delete simple sets from a 3D image.

### 1.3. Existing Algorithms

Generally, thinning algorithms are applied iteratively to remove the outmost layer of every object. It is straightforward to see that deletion of one simple point from an image is connectivity preserving. There are three different classes of parallel thinning algorithms according to the operations applied in each iteration:

- **n-subiteration parallel thinning algorithms.** There are *n* subiterations in each iteration. This term specifically refers to the algorithms where only border points of certain kind are considered for deletion in each subiteration. For 3D images, since there are six kinds of border points, normally we consider the case of six subiterations. A number of 6-subiteration parallel thinning algorithms were proposed. We will discuss some of them later.

- **n-subfield parallel thinning algorithms.** Each image is subdivided into *n* different subfields and the thinning algorithm is alternatively applied on all object points in each subfield. Hence, there are actually *n* subiterations in each iteration. Two possible cases are:

  - 2-subfield parallel thinning algorithms, where two distinct points of a 3D image are in the same subfield if and only if they are connected by a 26-path in which every two consecutive points are diametrically adjacent; and
  - 4-subfield parallel thinning algorithms where two distinct points of a 3D image are in the same subfield if and only if they are connected by a 26-path in which every two consecutive points are diagonally adjacent.

- **Fully parallel thinning algorithms.** Algorithms from this group are applied in parallel to all object points of the given image. Hence, there is only one application of the thinning algorithm in each iteration. Although it seems intuitively obvious that fully parallel thinning algorithms would be “faster” than subfield or subiteration algorithms, it may be appropriate to verify this assumption on an individual basis. (Practically, “faster” means that the number of applications of the algorithm for generating an output image is smaller).

The orientations of *x*- , *y*- , and *z*-axes are shown in Fig. 2. Let *p* be a point in a 3D image. Then, let *e*(*p*), *w*(*p*), *n*(*p*), *s*(*p*), *u*(*p*), and *d*(*p*) be 6-neighbors of *p* in the following order: east, west, north, south, up, and down, respectively (see Fig. 2). Let *nu*(*p*), *nd*(*p*), *ne*(*p*), *nw*(*p*), *se*(*p*), *sw*(*p*), *wu*(*p*), *wd*(*p*), *eu*(*p*), and *ed*(*p*) be the north-up, north-down, north-east, north-west, south-up, south-down, south-east, south-west, west-up, west-down, east-up, and east-down neighbors of *p*, respectively.

For the 2D case, since there are only four kinds of border points, Rosenfeld in [20] proposed a famous 4-subiteration thinning algorithm that is to alternatively delete all north, east, south, and west border simple nonend points, where an *end point* of a 2D image is an object point with exactly one neighbor being an object point. Morgenthaler in [15] proposed a definition of “end-points” for the 3D case and studied the problem for constructing 3D 6-subiteration and fully parallel thinning algorithms. Tsao and Fu in [23] studied the same problem for 3D 6-subiteration thinning algorithms by their own definition of 3D “end points.”

Tsao and Fu in [22] proposed a 2-step 3D 6-subiteration parallel thinning algorithm on 3D images. After the first step, each object in the original image is converted into a **medial face**. After the second step, each medial face is converted into a **medial axis**. Suppose *p* is the origin of **Z**3. Then *N*1(*p*) is the set of all points in *N*(*p*) with *x*-coordinates being 0. Define *N*2(*p*) and *N*3(*p*) similarly.

The first step of this algorithm can be proved to preserve connectivity by the following proposition. We omit the second step of this algorithm.

**Proposition 1.4.** The deletion of all north border simple points *p* of 3D images preserves connectivity if all following conditions hold:

1. *p* is adjacent to only one object in *N*1(*p*) and *N*2(*p*); and
2. *p* is adjacent to at least two object points in *N*1(*p*) and *N*3(*p*); and
3. *p* is adjacent to at least two object points in *N*(*p*).

Gong and Bertrand in [4] proposed another 3D 6-subiteration parallel thinning algorithm. The output of this algorithm is an image containing a set of medial faces. Let a **two-cube** be the union of two unit cubes sharing a common unit square. Their algorithm can be proved to preserve connectivity by the following proposition.
PROPOSITION 1.5. The deletion of all north border points $p$ on 3D images preserves connectivity if all following conditions hold:

1. $s(p) = 1$;
2. $p$ is adjacent to only one object in $N_1(p)$ and $N_2(p)$; and
3. $p$ is adjacent to only one object in every two-cube containing $\{p, s(p), n(p)\}$.

In [16], a 3D 6-subiteration parallel thinning algorithm called MESPTA was proposed that is extended from a 2D thinning algorithm called SPTA (see [17]). This algorithm also generates an image containing a set of medial faces. In this algorithm, two “slant” 2D $3 \times 3$ neighborhoods of $p$ were used. Generally, these “slant” 2D neighborhoods can be obtained by $45^\circ$ rotations of $N_1(p)$ and $N_2(p)$ according to the $y$-axis. Let $N_1(p)$ be such a “slant” 2D neighborhood of $p$, where $N_1(p)$ contains $p, n(p), s(p)$, and their east-up and west-down neighbors. Let $N_2(p)$ be another “slant” 2D neighborhood of $p$ where $N_2(p)$ contains $p, n(p), s(p)$, and their east-down and west-up neighbors. Their algorithm can be proved to preserve connectivity by the following proposition.

PROPOSITION 1.6. The deletion of all north border points $p$ on 3D images preserves connectivity if all following conditions hold:

1. $s(p) = 1$;
2. $p$ is adjacent to only one object in $N_1(p)$ and $N_2(p)$; and
3. $p$ is adjacent to only one object in $N_1(p)$ and $N_2(p)$.

1.4. Proof Techniques

It is not trivial to prove that a 3D parallel thinning algorithm preserves connectivity. In [5], Hall proposed efficient tests for 3D 2-subfield and 4-subfield parallel thinning algorithms to preserve connectivity.

PROPOSITION 1.7. Any 2-subfield parallel thinning algorithm for 3D images preserves connectivity if both following conditions hold:

1. Let $p$ and $q$ be two diagonally adjacent object points. Then $p, q$ cannot be deleted at the same time if $(N(p) \cap N(q)) - \{p, q\}$ contains two objects.
2. No objects of two, three, or four points in which every pair of points are diagonally adjacent can be completely deleted at the same time.

PROPOSITION 1.8. Any 4-subfield parallel thinning algorithm for 3D images preserves connectivity if no objects of two diametrically adjacent points can be deleted at the same time.

More general 3D sufficient conditions were proposed in [12] which were extended from Ronse’s efficient 2D results (see [19]). It can be shown that both of the above propositions are special cases of the following general 3D propositions [12].

PROPOSITION 1.9. A thinning algorithm for 3D images preserves connectivity if all following conditions hold:

1. only simple points can be deleted;
2. if two object points, $p$ and $q$, belong to a distinct unit square and are deleted at the same time, then $(p, q)$ is simple;
3. if three object points, $p, q, r$, belong to a distinct

![FIG. 3. The support of Algorithm 2.3 and Algorithm 2.5, where each lattice point is an object point.](image)

![FIG. 4. Four basic template cores of our parallel thinning algorithm where an unmarked point is a “don’t-care” point which can be either an object point or a background point. For (d), $p$ must be simple.](image)
2. THE NEW 3D THINNING ALGORITHM AND ITS APPLICATIONS

2.1. The Deleting Conditions

Our new thinning algorithm contains a set of deleting templates. If the neighborhood configuration of an object point in a 3D image matches any deleting template, then the point is a candidate to be deleted and is said to satisfy the deleting template. Note that each point of a deleting template can be either a black point, a white point, or a “don’t-care” point, where a “don’t-care” point matches either a black point or a white point. The support of a thinning algorithm defined with a point \( p \) is a neighborhood of \( p \) with a minimal number of points that are necessary for determining whether \( p \) can be deleted by the algorithm. The support of our Algorithm 2.3 and Algorithm 2.5 defined with \( p \) is shown in Fig. 3.

The templates of our algorithms can be classified into four subsets. Each subset can be obtained by a template core and by the rules of the following definition. We define the four template cores in Figs. 4a–d. Let \( T \) be the set of all reflections and all 90° rotations around any one of the three major axes of the four template cores, i.e., \( T \) is a set of \( 3 \times 3 \times 3 \) configurations. By the following definition, each element \( T_i \) of \( T \) is expanded to a (deleting) template \( T'_i \) by including zero or more points out of \( T_i \). The set of deleting templates of our thinning algorithm is the set of all \( T'_i \) expanded from \( T_i \) of \( T \) by the following definition.

**Definition 2.1.** For each \( T_i \in T \) where \( p \) is the central point of \( T_i \), \( T'_i \) can be obtained as follows:

1. if \( T_i \) is isomorphic to any template core in Figs. 4a–c, then \( T'_i = T_i \), and all following conditions are satisfied:
   (a) if \( s(p) = 1 \) in \( T_i \), then \( s(s(p)) = 1 \) in \( T'_i \);
   (b) if \( w(p) = 1 \) in \( T_i \), then \( w(w(p)) = 1 \) in \( T'_i \);
   (c) if \( d(p) = 1 \) in \( T_i \), then \( d(d(p)) = 1 \) in \( T'_i \);

2. if \( T_i \) is isomorphic to the template core in Fig. 4d, then \( T'_i = T_i \), and all following conditions are satisfied:
   (a) \( p \) is a simple point in \( N(p) \);
   (b) if \( q = su(p) \) is a 1 in \( T_i \), then at least one of the points, \( s(s(p)), s(q), su(q), u(q), \) and \( u(u(p)) \) is a 1 in \( T'_i \);
   (c) if \( q = sd(p) \) is a 1 in \( T_i \), then at least one of the points, \( s(s(p)), s(q), sd(q), d(q), d(d(p)) \) is a 1 in \( T'_i \);
   (d) if \( q = se(p) \) is a 1 in \( T_i \), then at least one of the points, \( s(s(p)), s(q), se(q), e(q), \) and \( e(e(p)) \) is a 1 in \( T'_i \);
   (e) if \( q = sw(p) \) is a 1 in \( T_i \), then at least one of the points, \( s(s(p)), s(q), sw(q), w(q), \) and \( w(w(p)) \) is a 1 in \( T'_i \);
   (f) if \( q = uw(p) \) is a 1 in \( T_i \), then at least one of the

**FIG. 5.** In (a)–(f), all six Class A templates for deleting west, east, south, north, down, and up border points, respectively.

**FIG. 6.** In (a), there is a Class B template for deleting north-west border points; in (b) is also a Class B template for deleting north-east border points.
points, \( w(w(p)), w(q), w(u(q)), u(q), \) and \( u(u(p)) \) is a 1 in \( T_i' \);

\[ (g) \text{ if } q = wd(p) \text{ is a 1 in } T_i, \text{ then at least one of the points, } w(w(p)), w(q), wd(q), d(q), \text{ and } d(d(p)) \text{ is a 1 in } T_i'. \]

Definition 2.1(1) is stimulated by Holt et al.'s 2D parallel thinning algorithm in [8]. Let \( \text{Class A} \) be the union of all deleting templates of our algorithm that are generated by the above definition from the template core shown in Fig. 4a. Define \( \text{Classes B, C, and D} \) similarly (see Figs. 4b–d, respectively). With all reflections and all 90° rotations around any one of the three major axes of the four template cores in Fig. 4a–d, there are 6 templates in Class A, 12 templates in Class B, 8 templates in Class C, and 12 templates in Class D. All six templates in Class A are shown in Fig. 5, where for three of them we require an additional black point that is not in the \( 3 \times 3 \times 3 \) neighborhood of \( p \). Note that the two templates shown in Figs. 5a–b are for deleting east and west border points. They together prevent the puncture of a hole in a flat object of thickness 2. This is true when Figs. 5c–d and (e–f) together for deleting the north–south border point and up–down border points, respectively. For Classes B, C, and D, we just show two templates as examples for each of them (see Figs. 6 to 8).

As mentioned in previous sections, geometry preservation is a major concern of thinning algorithms. To preserve geometry, an object point with only one 26-neighbor being an object point should not be deleted. For our algorithm, we need to preserve some object points which are 26-adjacent to two distinct object points. The following definition is the preserving conditions of our algorithm.

**Rule 2.2.** Let \( p \) be an object point of a 3D image. Then:

1. \( p \) is called a **line-end point** if \( p \) is 26-adjacent to exactly one object point,
2. \( p \) is called a **near-line-end point** if \( p \) is 26-adjacent to exactly two object points which are:
   
   - either \( s(p) \) and \( e(p) \), or \( s(p) \) and \( u(p) \) but not both;
   - either \( n(p) \) and \( w(p) \), or \( u(p) \) and \( w(p) \) but not both;
   - either \( n(p) \) and \( d(p) \), or \( e(p) \) and \( d(p) \) but not both;
3. \( p \) is called a **tail point** if it is either a line-end point or near-line-end point; otherwise it is called a nontail point.

**2.2. The New Fully Parallel Algorithm**

An object point is deleted by a thinning algorithm if it satisfies at least one deleting condition of the algorithm. Our thinning Algorithm 2.3 has the traditional structure of parallel thinning algorithms (as shown below). In each iteration of this algorithm, all nontail object points satisfying at least one of the deletion templates are deleted.
For each elongated object, the algorithm will generate a medial line as its skeleton. This algorithm terminates when no object points can be deleted. Since we assume that all input images contain finitely many object points, this algorithm will eventually terminate.

Algorithm 2.3.

repeat

in parallel, delete every nontail object point which satisfies at least one deleting template in Class A, B, C, or D;

until no point can be deleted;

The proof of this algorithm to preserve connectivity will be given later in Theorem 3.7. For “smooth” 3D objects, the skeletons generated by this algorithm are reasonably good. Different shapes of objects have been tested by this parallel algorithm. Here, only four cases are presented (see Figs. 9 to 12). Note that in these figures, each small cube represents an object point in the image.

2.3. The Applications

Skeletonization followed by graph construction is frequently used to describe structural information in image data. While two-dimensional skeletonization was shown to be practical in many applications, including character recognition and remote sensing, three-dimensional skeletonization is often useful in medical image analyses [24–26]. Three-dimensional image data acquired from computed tomography (CT), magnetic resonance (MR), or positron emission tomography (PET) frequently visualize three-dimensional objects consisting from elongated mutually interconnected pieces like vascular trees in brain, heart, lungs, or pulmonary airway trees.

In these applications, the skeleton may be used to identify the three-dimensional midline of vessels or airways, to determine branch points, and to identify the tree structure. The resulting tree structure may identify the tree topology across time and can be used in progression/regression studies for quantitative comparisons. More specifically, our
It can be shown that there are no simple points (in the 2D sense; see [10]) in this object. Hence, no points can be deleted and the resulting skeleton would not be of unit width. Generally, we do not want this to happen in applications. To overcome such problems in real images, the concept of well-composed images was introduced in [2, 3, 11] and we use an analogous definition for the 3D case as follows.

**Definition 2.4.** A 3D image $P$ is called well composed if both of the following conditions hold:

1. each unit square of $P$ contains at most one black 6-component; and
2. each unit cube of $P$ contains at most one black 6-component.

By the above definition, every object in a 3D well-composed image is also 6-connected. Since every object and every background component are both 6-connected, the number of special white/black configurations is reduced. In [2, 11], efficient rules were introduced for adding object points in unit squares of special configurations on 2D images. Analogous rules for 3D images can be obtained similarly. For example, the left object in the following diagram is not well composed, but the right one is:

```
1 1 1
1 1 1 1
1 1 1 1
1 1 1 1
1 1
```

Note that all computer-generated 3D objects shown in Figs. 9–12 are well-composed. The real 3D medical objects shown in Fig. 14a and Fig. 15a are also well composed. However, their skeletons, shown in Fig. 14b and Fig. 15b are not well composed.

**Dilation.** The concept of dilation operation was first proposed by Arcelli in [1]:

We now introduce these steps.

**Well-Composed Images.** The purpose of our thinning algorithm is to generate an image in which every object is of unit width. However, the unit width of the thinned result cannot be guaranteed in real images. For example, consider the following object on a 2D image (see [10] for the definitions of 2D images). This example was first proposed by Arcelli in [1]:

```
1 1
1 1
1 1
1 1
```

1. Convert an input image $P$ into a well-composed image $P_1$, (the definition of well-composedness is given below).

2. Apply dilation of mathematical morphology on $P_1$ and generate a “smoother” image $P_2$.

3. Apply a revised version of our fully parallel thinning algorithm, Algorithm 2.5, on $P_2$ to obtain the final image $P_3$. 

We now introduce these steps.
dilation. In Fig. 13b, there is a “U”-like object (marked by thicker lines). The middle concave portion of the “U”-like object is called a “valley.” We can use the dilation operation of mathematical morphology to fill small valleys and, hence, to obtain a “smoother” image. In Fig. 13b, the “U”-like object is contained in a block object (marked by thinner lines). The block object can be obtained from the “U”-like object by a single dilation using a $3 \times 3 \times 3$ structure element. In this case, the origin of the structure element is its central point, all 26-neighbors of the central point are object points. In our application, the input images are dilated in two or three iterations using a $3 \times 3 \times 3$ structure element, where the origin of the structure element is also its central point and only the set of all its 6-neighbors are object points. After an input image was dilated by the structure element, the objects in the output—input were “smoother” and it was easier to generate better skeletons.

A Revised Parallel Thinning Algorithm. For the purpose of generating better skeletons in 3D images, we revise the structure of Algorithm 2.3 and generate the following Algorithm 2.5. In each iteration of the following algorithm, we first mark every object point which is 26-adjacent to a background point, then we apply Algorithm 2.3 on the set of all marked points. This algorithm terminates when no object points can be deleted. Since we assume that all input images contain finitely many object points, this algorithm will eventually terminate. In our experiments, the revised algorithm preserves better geometry properties than Algorithm 2.3; i.e., a skeleton generated by the revised algorithm has a structure that looks closer to the original object.

**Algorithm 2.5.**

repeat
- mark every object point which is 26-adjacent to a background point;
- apply Algorithm 2.3 on the set of marked points;
- release all marked but not deleted points;
until no marked point can be deleted;

In each iteration of Algorithm 2.3, only those object points 6-adjacent to at least one background point can be considered for deletion; an object point that is 26- but not 6-adjacent to any background point cannot be deleted. Let $X$ be an object. The first step of Algorithm 2.5 is to mark every object point $x$ of $X$, where $x$ is 26-adjacent to a background point. We may consider the set of marked points as the outmost layer of $X$. The deletion operation in each iteration of Algorithm 2.5 is applied only to the marked points. Since an object point 26-adjacent to a background point may not be 6-adjacent to any background point, the set of marked points may not be deleted at the same time. Each iteration of Algorithm 2.5 stops when no marked point can be deleted. After a number of marked points have been deleted, some object points originally in the “second outmost layer” may become 6-adjacent to a background point. However, before we know that no currently marked points can be deleted, the algorithm cannot be applied to unmarked points.

Note that the second step of the above algorithm is also an iterative process which applies Algorithm 2.3 to delete all possible marked points until no marked point can be deleted. The correctness of this algorithm to preserve connectivity can be obtained immediately after Algorithm 2.3 is proved to preserve connectivity.

Shown in Fig. 14a is a 3D rendering of image data from a CT scan of a glass phantom used in validation of our pulmonary airway tree segmentation method [25]. The original three-dimensional CT scan offered in-slice resolution of 0.35 mm (image slices $256 \times 256$; 93 slices). Shape-based interpolation was used to obtain cubic voxels and the interpolated image consisted of 529 slices. The interpolated image ($256 \times 256 \times 529$) was skeletonized. Figure 14b shows the 3D skeleton determined using our parallel skeletonization algorithm and Fig. 14c demonstrates the overlay of 3D-rendered skeleton over the 3D-rendered original. Figure 15a gives example of a canine pulmonary airway tree determined from a 3D in vivo CT scan. The original image was acquired at a resolution of $256 \times 256$ with pixel size of 0.70 mm and 3 mm slice thickness. After shape-based interpolation, the image consisted of 171 slices ($256 \times 256 \times 171$). Figure 15b gives the corresponding 3D skeleton, determined using our parallel skeletonization algorithm. The skeleton accurately represents the topology of the pulmonary airway tree and appears to be approximately medial in the examples shown in the overlay in Fig. 15c.

### 3. VERIFICATION

To simplify the proof of our algorithm to preserve connectivity, we use $\Omega$ to denote the set of deleting templates of our thinning Algorithm 2.3. An object point in a 3D image is said to satisfy $\Omega$ if it satisfies any deleting template in $\Omega$.

Clearly, every point deleted by our algorithm satisfies $\Omega$. It is not difficult to see that it is enough to prove that our algorithm is connectivity preserving by showing that the set of object points deleted by $\Omega$ satisfies all five conditions of Proposition 1.9. For verifying Proposition 1.9(1), we must show that $\Omega$ can only delete simple points from any 3D image.

The following lemma is valid by showing that each template of $\Omega$ can only delete simple points. This can be easily seen by checking that each of the four template cores in Fig. 4 can only delete simple points.
LEMMA 3.1. \( \Omega \) deletes only simple points of 3D images.

Let \( p \) and \( q \) be object points where \( p \) satisfies \( \Omega \). By “\( q \in \Omega(p) \)” we mean that \( q \) must be an object point for \( p \) to satisfy \( \Omega \); that is, \( p \) does not satisfy \( \Omega \) after \( q \) is deleted. Then “\( q \not\in \Omega(p) \)” means \( p \) still satisfies \( \Omega \) after \( q \) is deleted. By investigating all template cores in Fig. 4, it can be seen that if \( q \in \Omega(p) \), then \( q \) must be 18-adjacent to \( p \). Hence, we have the following lemma.

LEMMA 3.2. Let \( p, q \) be two diametrically adjacent object points in a 3D image where both \( p \) and \( q \) satisfy \( \Omega \). Then \( q \not\in \Omega(p) \) and \( p \not\in \Omega(q) \).

LEMMA 3.3. Let \( p \) and \( q \) be two diagonally adjacent object points of a 3D image. Suppose both \( p \) and \( q \) satisfy \( \Omega \). Then either \( q \not\in \Omega(p) \) or \( p \not\in \Omega(q) \).

Proof. Suppose \( q \in \Omega(p) \) and \( p \in \Omega(q) \) (see the diagram below). Since \( q \in \Omega(p) \), \( p \) must satisfy a Class D template in \( \Omega \) and \( p_1 = p_2 = p_3 = 0 \). Since \( p \in \Omega(q) \), \( q \) also must satisfy a Class D template in \( \Omega \) and \( q_1 = q_2 = q_3 = 0 \):

\[
\begin{array}{ccc}
p_1 & p_2 & \text{—} \\
\text{—} & p & p_3 \\
q_1 & q & \text{—} \\
q_2 & q_3 & \text{—}
\end{array}
\]

But then either \( q \) is one of the following neighbors of \( p \): \( sd(p), su(p), se(p), sw(p), wu(p), \) and \( wd(p) \), or vice versa. WLOG let \( q = sw(p) \). Then by Definition 2.1 (2e)
at least one point in \{q_1, q_2, q_3\} must be an object point; a contradiction. ■

**Corollary 3.4.** Let \(p, q, r,\) and \(s\) belong to a distinct unit square of a 3D image where \(p\) is diagonally adjacent to \(q\). Suppose both \(p\) and \(q\) satisfy \(\Omega\) and either \(r = 1\) or \(s = 1\). Then \(q \notin \Omega(p)\) and \(p \notin \Omega(q)\).

**Proof.** If \(p \in \Omega(q)\), then \(p\) must satisfy a Class D template and \(r = s = 0\); if \(q \in \Omega(p)\), then \(q\) must satisfy a Class D template and \(r = s = 0\). ■

Suppose \(q \in \Omega(p)\) and \(q\) is 6-adjacent to \(p\). WLOG let \(q = d(p)\). Then by investigating the basic templates in Fig. 4, \(p\) must be an up border point (since if \(p\) is not an up border point, then \(q \notin \Omega(p)\)). Then we have the following lemma.

**Lemma 3.5.** Let \(p, q\) be two 6-adjacent object points in a 3D image where both \(p\) and \(q\) satisfy \(\Omega\). Then either \(q \notin \Omega(p)\) or \(p \notin \Omega(q)\).

**Proof.** Let \(a - q - p - b\) be four points on a straight line such that \(a\) is 6-adjacent to \(q\) and \(b\) is 6-adjacent to \(p\). Suppose \(p \in \Omega(q)\) and \(q \in \Omega(p)\). By checking Fig. 4, \(p \in \Omega(q)\) implies that \(a = 0; q \in \Omega(p)\) implies that \(b = 0\). But then either \(p\) or \(q\) must be a north, east, or up border point. WLOG let \(p\) be an up border point. Then by Definition 2.1 (1c) since \(q = d(p)\) and \(a = d(d(p))\), \(a\) must be an object point for \(p\) to satisfy \(\Omega\), a contradiction. ■

Now we claim the following result by using the above lemmas.

**Proposition 3.6.** The set of object points deleted by \(\Omega\) satisfies all five conditions of Proposition 1.9.

**Proof.** To prove this proposition, we claim that the parallel deletion by \(\Omega\) satisfies Proposition 1.9. By Lemma 3.1, every point satisfying \(\Omega\) is a simple point. Hence Proposition 1.9(1) holds.

Let \(p\) and \(q\) be two distinct object points of a unit square. Suppose both of them satisfy \(\Omega\). Since \(p\) is 18-adjacent to \(q\), by Lemmas 3.3 and 3.5, WLOG let \(q \notin \Omega(p)\). Then \(p\) still satisfies \(\Omega\) after \(q\) is deleted; that is, \(p\) is simple after \(q\) is deleted. Then, by Definition 1.3, \(\{p, q\}\) is simple. Thus, Proposition 1.9(2) holds.

Let \(p, q, \) and \(r\) be three distinct object points of a unit square, where \(q\) is diagonally adjacent to \(r\). Suppose all three of them satisfy \(\Omega\). Since \(p = 1\), by Corollary 3.4, \(q \notin \Omega(r)\) and \(r \notin \Omega(q)\). If \(p \notin \Omega(r)\), then \(r\) satisfies \(\Omega\) after \(\{p, q\}\) is deleted, which implies that \(r\) is simple after \(\{p, q\}\) is deleted. But since \(\Omega\) satisfies Proposition 1.9(2), \(\{p, q\}\) is simple (with \(r = 1\)), which by Definition 1.3 implies that \(\{p, q, r\}\) is simple. Suppose \(p \notin \Omega(r)\). Then, by Lemma 3.5, \(r \notin \Omega(p)\). Hence, both \(p\) and \(q\) satisfy \(\Omega\) after \(r\) is deleted. Again, by Proposition 1.9(2), since \(\{p, q\}\) is simple after \(r\) is deleted, \(\{p, q, r\}\) is simple and Proposition 1.9(3) holds.

Let \(p, q, r,\) and \(s\) be four distinct object points of a unit square shown below where every point in \(\{p, q, r, s\}\) satisfies \(\Omega\):

\[
\begin{array}{cccc}
- & x_2 & w_4 & - \\
& w_1 & p & q & x_1 \\
b_1 & r & s & w_3 \\
- & w_2 & b_4 & - \\
\end{array}
\]

Suppose there exists a point in \(\{p, q, r, s\}\), say \(r\), such that \(p \notin \Omega(r)\) and \(s \notin \Omega(r)\). Then by Lemma 3.5, \(r \notin \Omega(p)\) and \(r \notin \Omega(s)\). Since, by Corollary 3.4, \(r \notin \Omega(q)\), after \(r\) is deleted each point in \(\{p, q, s\}\) still satisfies \(\Omega\). But since \(\Omega\) satisfies Proposition 1.9(3), \(\{p, q, s\}\) is simple after \(r\) is deleted. Thus, by Definition 1.3, \(\{p, q, r, s\}\) is simple.

Now suppose there exists a point in \(\{p, q, r, s\}\), say \(r\), such that \(p \notin \Omega(r)\) and \(s \notin \Omega(r)\). Since, by Corollary 3.4, \(q \notin \Omega(s)\), after \(\{p, q, s\}\) is deleted, \(r\) still satisfies \(\Omega\) and
FIG. 15. Canine pulmonary airway tree determined from in vivo CT data. (a) 3D rendering. (b) 3D skeleton of the airway tree determined using our revised parallel thinning Algorithm 2.5. (c) Overlay of the 3D rendered skeleton over the 3D rendered original. (In reality, the skeleton is located inside the airway branches.)
hence is simple. But \( \Omega \) satisfies Proposition 1.9(3) which implies that \( \{p, q, s\} \) is simple. Thus, by Definition 1.3, \( \{p, q, r, s\} \) is simple.

Now suppose for every point \( x \) in \( \{p, q, r, s\} \) exactly one of its 6-neighbors in \( \{p, q, r, s\} \) is in \( \Omega(x) \). WLOG we consider the following case: \( q \in \Omega(p) \), \( p \in \Omega(r) \), \( r \in \Omega(s) \), \( s \in \Omega(q) \). It is easily seen that \( q \in \Omega(p) \) implies \( w_1 = 0 \), \( p \in \Omega(r) \) implies \( w_2 = 0 \), \( r \in \Omega(s) \) implies \( w_3 = 0 \), and \( s \in \Omega(q) \) implies \( w_4 = 0 \). Since \( w_1 = w_3 = 0 \), either \( p \) or \( s \) is a north, east or up border point. WLOG let \( s \) be an east border point. Then \( b_1 = 1 \). Since \( w_2 = w_4 = 0 \), WLOG let \( q \) be a north border point. Then \( b_3 = 1 \) which implies that \( r \) cannot satisfy any template in Class A as a south border point. Since \( b_1 = b_2 = 1 \), \( r \) cannot satisfy any template in Class B or Class C. Since \( p = s = b_2 = 1 \), \( r \) cannot satisfy any template in Class D. Thus, \( r \) must satisfy some template in Class A as an up or down border point. Then \( p \not\in \Omega(r) \), a contradiction. Hence, Proposition 1.9(4) holds.

We claim Proposition 1.9(5) by showing that one cannot construct an object contained in a unit cube (see Fig. 16) such that every point of this object satisfies \( \Omega \). Assume all points not in the cube are 0’s. Suppose \( p_7 = 1 \). Then \( p \) satisfies \( \Omega \). By Definition 2.1(1c) and (2e), if either \( p_1 \) or \( p_3 \) is in \( \Omega(p_7) \), then there exists an object point connected to \( p_7 \) but not in the cube, a contradiction. Hence, both \( p_1 \) and \( p_3 \) are not in \( \Omega(p_7) \). Similarly, all of \( p_3, p_4, p_6, p_8 \) are not in \( \Omega(p_7) \). But the fact that \( p_7 \) satisfies \( \Omega \) does not depend on \( p_2 \). Thus, \( p_7 \) does not satisfy \( \Omega \) which implies that \( p_7 = 0 \).

Suppose \( p_8 = 1 \). Since all points out of the cube are 0’s, similar to above, the only way for \( p_8 \) to satisfy \( \Omega \) is \( p_7 = 1 \). But since \( p_7 = 0, p_8 = 0 \). Now suppose \( p_5 = 1 \). Since all points out of the cube are 0’s, similar to above, the only way for \( p_5 \) to satisfy \( \Omega \) is either \( p_7 = 1 \) or \( p_8 = 1 \), a contradiction. Hence, \( p_5 = 0 \). Finally, suppose \( p_6 = 1 \). Since all points out of the cube are 0’s, similarly the only way for \( p_6 \) to satisfy \( \Omega \) is either \( p_5 = 1 \) or \( p_7 = 1 \) or \( p_8 = 1 \). This leads to a contradiction. Hence, \( p_6 = 0 \).

Since \( p_5, p_6, p_7, \) and \( p_8 \) are all 0’s, the remaining four points are contained in a unit square. Now since Proposition 1.9.(4) holds, if every object point in the square with corners \( \{p_1, p_2, p_3, p_4\} \) satisfies \( \Omega \), then the set is simple. Hence, Proposition 1.9(5) holds.

As an immediate consequence of the proof that Proposition 1.9 holds, we can present the final theorem.

**Theorem 3.7.** Algorithm 2.3 is connectivity preserving.

### 4. Conclusion

A new fully parallel 3D connectivity preserving thinning algorithm was presented and the connectivity preservation proof was given. Note that using the tests in [12] (introduced in Proposition 1.9) made the proof considerably simpler than similar proofs in the past (e.g., [22]).

The practical applicability of our 3D fully parallel thinning algorithm was demonstrated on computer-generated images and real three-dimensional image sets acquired from computed tomography scanners. The visual analysis of the resulting skeletons shows that the thinning algorithm can be successfully used in many three-dimensional thinning problems in the area of 3D medical image analysis.

**Acknowledgments**

The authors thank Eric A. Hoffman for providing original CT lung data, Krishnan B. Chandran for providing CT images of the glass phantom, and Gopal Sundaramoorthy and Susanne Shamsolkottabi for technical assistance.

**References**


