

Fully Parallel 3D Thinning Algorithms Based on Sufficient Conditions for Topology Preservation

Kálmán Palágyi and Gábor Németh

Department of Image Processing and Computer Graphics,
University of Szeged, Hungary
{palagyi,gnemeth}@inf.u-szeged.hu

Abstract. This paper presents a family of parallel thinning algorithms for extracting medial surfaces from 3D binary pictures. The proposed algorithms are based on sufficient conditions for 3D parallel reduction operators to preserve topology for $(26, 6)$ pictures. Hence it is self-evident that our algorithms are topology preserving. Their efficient implementation on conventional sequential computers is also presented.

1 Introduction

Thinning algorithms extract “skeletons” from binary pictures by using topology preserving reduction operations [8]. A 3D reduction operation does *not* preserve topology [7] if any object is split or is completely deleted, any cavity is merged with the background or another cavity, a new cavity is created, a hole (which doughnuts have) is eliminated or created.

A *3D binary picture* [8] is a mapping that assigns a value of 0 or 1 to each point with integer coordinates in the 3D digital space denoted by \mathbb{Z}^3 . Points having the value of 1 are called *black* points, and those with a zero value are called *white* ones. A *simple* point is a black point whose deletion does not alter the topology of the picture [8].

Parallel thinning algorithms delete a set of black points. Topological correctness of parallel 3D thinning algorithms can be verified by the help of sufficient conditions for 3D parallel reduction operators to preserve topology [10,7,14].

Thinning algorithms use operators that delete some points which are not end-points, since preserving end-points provides important geometrical information relative to the shape of the objects. *Curve-thinning* algorithms are used to extract medial lines or centerlines, while *surface-thinning* ones produce medial surfaces. Curve-thinning preserves line-end points while surface-thinning does not delete surface-end points [2,3,14].

This paper presents a family of 3D surface-thinning algorithms that are based on sufficient conditions for topology preservation proposed by Palágyi and Kuba [14]. The strategy which is used is called fully parallel: the same parallel reduction operator is applied at each iteration. Our algorithms use different types of surface-end points.

This paper is organized as follows. Section 2 gives the basic notions of 3D digital topology and the applied sufficient conditions for topology preservation.

Then in Section 3 we present three new fully parallel surface-thinning algorithms and their results on some test pictures. Finally, Section 4 proposes an efficient implementation of our algorithms, and we conclude in Section 5.

2 Basic Notions and Results

Let p be a point in the 3D digital space \mathbb{Z}^3 . Let us denote $N_j(p)$ (for $j = 6, 18, 26$) the set of points that are j -adjacent to point p (see Fig. 1a).

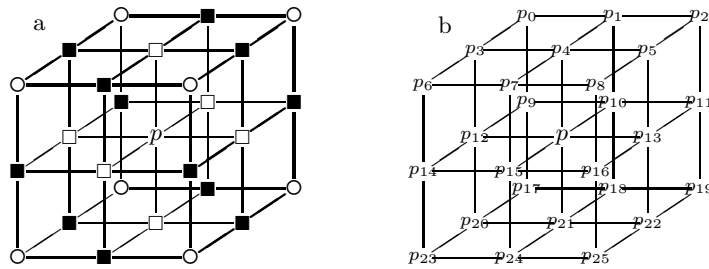


Fig. 1. Frequently used adjacencies in \mathbb{Z}^3 (a) and an indexing scheme to encode all possible $3 \times 3 \times 3$ configurations (b). The set $N_6(p)$ of the central point $p \in \mathbb{Z}^3$ contains point p and the 6 points marked “□”. The set $N_{18}(p)$ contains $N_6(p)$ and the 12 points marked “■”. The set $N_{26}(p)$ contains $N_{18}(p)$ and the 8 points marked “○”.

The sequence of distinct points $\langle x_0, x_1, \dots, x_n \rangle$ is called a j -path (for $j = 6, 18, 26$) of length n from point x_0 to point x_n in a non-empty set of points X if each point of the sequence is in X and x_i is j -adjacent to x_{i-1} for each $1 \leq i \leq n$ (see Fig. 1a). Note that a single point is a j -path of length 0. Two points are said to be j -connected in the set X if there is a j -path in X between them.

The 3D binary $(26, 6)$ digital picture \mathcal{P} is a quadruple $\mathcal{P} = (\mathbb{Z}^3, 26, 6, B)$ [8]. Each element of \mathbb{Z}^3 is called a point of \mathcal{P} . Each point in $B \subseteq \mathbb{Z}^3$ is called a black point and has a value of 1 assigned to it. Each point in $\mathbb{Z}^3 \setminus B$ is called a white point and has a value of 0 assigned to it. Adjacency 26 is associated with the black points and adjacency 6 is assigned to the white points. A black component is a maximal 26-connected set of points in B , while a white component is a maximal 6-connected set of points in $\mathbb{Z}^3 \setminus B$. Here it is assumed that a picture contains finitely many black points.

A black point is called a border point in $(26, 6)$ pictures if it is 6-adjacent to at least one white point. A black point is called an interior point if it is not a border point. A simple point is a black point whose deletion is a topology preserving reduction [8].

Parallel reduction operators delete a set of black points and not just a single simple point. Hence we need to consider what is meant by topology preservation

when a number of black points are deleted simultaneously. The following theorem provides *sufficient conditions* for 3D parallel reduction operators to preserve topology.

Theorem 1. [14] *Let \mathcal{O} be a parallel reduction operator. Let p be any black point in any picture $\mathcal{P} = (\mathbb{Z}^3, 26, 6, B)$ such that p is deleted by \mathcal{O} . Let \mathcal{Q} be the family of all the sets of $Q \subseteq (N_{18}(p) \setminus \{p\}) \cap B$ such that $q_1 \in N_{18}(q_2)$, for any $q_1 \in Q$ and $q_2 \in Q$. The operator \mathcal{O} is topology preserving if all of the following conditions hold:*

1. p is simple in the picture $(\mathbb{Z}^3, 26, 6, B \setminus Q)$ for any Q in \mathcal{Q} .
2. No black component contained in a $2 \times 2 \times 2$ cube can be deleted completely by \mathcal{O} .

Note that there is an alternative method for verifying the topological correctness of 3D parallel thinning algorithms. It is based on Ma's sufficient conditions for 3D parallel reduction operators to preserve topology for $(26, 6)$ pictures [10]. It is proved in [14] that if a parallel reduction operator satisfies the conditions of Theorem 1, then Ma's conditions are satisfied as well.

3 The New Thinning Algorithms

In this section, a family of thinning algorithms are presented for extracting medial surfaces from 3D $(26, 6)$ binary pictures. These algorithms use fully parallel strategy; the same parallel reduction operator is applied at each phase of the thinning process [5].

Deletion conditions of the proposed surface-thinning algorithms are based on the conditions of Theorem 1. Let us define their deletable points:

Definition 1. *Let us consider an arbitrary characterization of surface-end points that is called as type \mathcal{T} . A black point p in picture $(\mathbb{Z}^3, 26, 6, B)$ is \mathcal{T} -deletable if all of the following condition hold:*

1. p is not a surface-end point of type \mathcal{T} .
2. p is a simple point.
3. p is simple in the picture $(\mathbb{Z}^3, 26, 6, B \setminus Q)$ for any $Q \subseteq (N_{18}(p) \setminus \{p\}) \cap B$, where set $Q \neq \emptyset$ and it contains mutually 18-adjacent points and all points in Q satisfy the first two conditions.
4. p does not come first in the lexicographic ordering of a black component $C \subseteq B$ which is contained in a $2 \times 2 \times 2$ cube and all points in C satisfy the first three conditions.

Note that various characterizations of surface-end points yield different types of deletable points.

It can readily be seen that the simultaneous deletion of all \mathcal{T} -deletable points from any $(26, 6)$ picture is a topology preserving reduction (i.e., it satisfies both conditions of Theorem 1). Hence topology preservation by the following algorithm is guaranteed:

Fully parallel surface-thinning algorithm based on \mathcal{T} -deletable points*Input:* picture $(\mathbb{Z}^3, 26, 6, B)$ *Output:* picture $(\mathbb{Z}^3, 26, 6, B')$ $B' = B$ **repeat** $D = \{ p \mid p \text{ is } \mathcal{T}\text{-deletable in } (\mathbb{Z}^3, 26, 6, B') \}$ $B' = B' \setminus D$ **until** $D = \emptyset$

A surface-thinning algorithm does not delete surface-end points that are defined by the algorithm-specific characterization of end-points. There are numerous types of surface-end points that are used by the existing surface-thinning algorithms [1,2,3,4,6,9,11,13,14,18,20]. Hence we can get a family of fully parallel surface-thinning algorithms by applying various characterizations of surface-end points.

For simplicity, let us consider the following three types of surface-end points.

Definition 2. A border point p in picture $(\mathbb{Z}^3, 26, 6, B)$ is a surface-end point of type i if the i -th condition holds:

1. There is no interior point in the set $N_6(p) \cap B$.
2. There is no interior point in the set $N_{18}(p) \cap B$.
3. There is no interior point in the set $N_{26}(p) \cap B$.

Note that surface-end points of type 1 are considered by some existing 3D surface-thinning algorithms [1,15,16,17] and all the three types are used by the algorithms proposed by Manzanera et al. [13].

The proposed three algorithms are called as **Alg- i** that are determined by surface-end points of type i ($i = 1, 2, 3$). In experiments these fully parallel surface-thinning algorithms were tested on objects of different shapes. Here we present some illustrative examples below (Figs. 2-5).

Note that due to the considered types of surface end-points, our medial surfaces may contain 2-point thick surface patches [1,15,16,17]. Fortunately, it is not hard to overcome this problem here (e.g. by applying a final thinning step [1]). Note that the proposed three algorithms use symmetric deletion conditions, hence they are invariant by reflections and $k \cdot \pi/2$ rotations. We should mention that the skeleton (as a shape feature) is sensitive to coarse object boundaries. As a result, the “skeleton” produced generally includes unwanted segments that must be removed by a pruning step [19].

4 Implementation

One may think that the proposed algorithms are time consuming and it is rather difficult to implement them. That is why this section will present an efficient way for implementing algorithms **Alg-1**, **Alg-2**, and **Alg-3** on a conventional sequential computer.

Our method uses a pre-calculated look-up-table to encode the simple points in (26, 6) pictures and an array to encode all possible sets that are considered in condition 3 of Definition 1. In addition, two lists are used to speed up the process: one for storing the border-points in the current picture; the other list is to store all deletable points in the current phase. At each iteration, the deletable points are deleted, and the list of border points is updated accordingly. The pseudocode of the proposed algorithms is given by the following:

```

procedure FULLY_PARALLEL_SURFACE_THINNING ( A, i )
  /* collect border points */
  border_list = < empty list >
  for each  $p = (x, y, z)$  in A do
    if  $p$  is border point then
      if  $p$  is surface-end point of type i then
         $A[x, y, z] = 2$ 
      else
        border_list = border_list + <  $p$  >
         $A[x, y, z] = 3$ 
  /* thinning process */
  repeat
    deleted = ITERATION_STEP ( border_list, A, i )
  until deleted = 0

```

The two parameters of the procedure are the array *A* (which stores the input picture to be thinned) and index *i* (that is to give the type of the considered surface-end points ($i = 1, 2, 3$)). In input array *A*, the value “1” corresponds to black points and the value “0” is assigned to white ones.

First, the input picture is scanned and all the border points that are not surface-end points of type *i* are inserted into the list *border_list*. Note that it is the only time consuming scanning of the entire array *A*. In order to avoid storing more than one copy of a border point in *border_list* and verifying the deletability of surface-end points again and again, the array *A* represents a five-color picture during the thinning process: the value of “0” corresponds to the white points, the value of “1” corresponds to (black) interior points, the value of “2” is assigned to all (black) surface-end points of type *i*, the value of “3” is assigned to all (black) border points in the actual picture that are not end-points (added to *border_list*), and the value of “4” corresponds to (black) points that satisfy the first two conditions of Definition 1.

The kernel of the **repeat** cycle corresponds to one iteration step of the process. Function ITERATION_STEP returns the number of deleted points by the actual iteration step. This number is then stored in the variable called *deleted*. The entire process terminates when no more points can be deleted. After the thinning, all points having a nonzero value belong to the medial surface. Note that array *A* contains the resultant “skeleton”, hence the input and output pictures can be stored in the same array. The key part of the proposed method is the organization of function called ITERATION_STEP. Let us see its pseudocode:

```

function ITERATION_STEP ( border_list, A, i )
  /* collect i-deletable points */
  deleted = COLLECT_DELETABLE ( A, i, border_list, deletable_list )
  /* deletion */
  for each point  $p = (x, y, z)$  in deletable_list do
     $A[x, y, z] = 0$ 
    border_list = border_list -  $\langle p \rangle$ 
    for each point  $q = (x', y', z')$  being 6-adjacent to  $p$  do
      if  $A[x', y', z'] = 1$  then
        border_list = border_list +  $\langle q \rangle$ 
    for each point  $p = (x, y, z)$  in border_list do
      if  $p$  is surface-end point of type  $i$  then
         $A[x, y, z] = 2$ 
        border_list = border_list -  $\langle p \rangle$ 
      else
         $A[x, y, z] = 3$ 
  return deleted

```

Function ITERATION_STEP uses an additional auxiliary data structure called *deletable_list*. Its processing task is composed of two basic parts. First, *i*-deletable points (see Definition 1) in the actual picture are transferred from *border_list* to *deletable_list*. Second, all the points in *deletable_list* are deleted from picture *A* and *border_list* is updated. If a border point is deleted, all interior points that are 6-adjacent to it become border points. These brand new border points of the actual picture are added to the *border_list*.

The pseudocode of the function COLLECT_DELETABLE is given by the following:

```

function COLLECT_DELETABLE ( A, i, border_list, deletable_list )
  deleted = 0
  deletable_list =  $\langle$  empty list  $\rangle$ 
  /* condition 2 */
  for each point  $p$  in border_list do
    index_26_nonzero = COLLECT_26_NONZERO (  $p$ , A )
    if LUT_simple [index_26_nonzero] = 1 then
       $A[x, y, z] = 4$ 
      deletable_list = deletable_list +  $\langle p \rangle$ 
      deleted = deleted + 1
  /* condition 3 */
  for each point  $p$  in deletable_list do
    index_26_nonzero = COLLECT_26_NONZERO (  $p$ , A )
    index_18_4 = COLLECT_18_4 (  $p$ , A )
    for  $j = 1$  to 66 do
      if ( index_18_4 AND  $Q[j]$  ) =  $Q[j]$  then
        if LUT_simple [index_26_nonzero AND NOT  $Q[j]$ ] = 0 then
          deletable_list = deletable_list -  $\langle p \rangle$ 
          deleted = deleted - 1
  return deleted

```

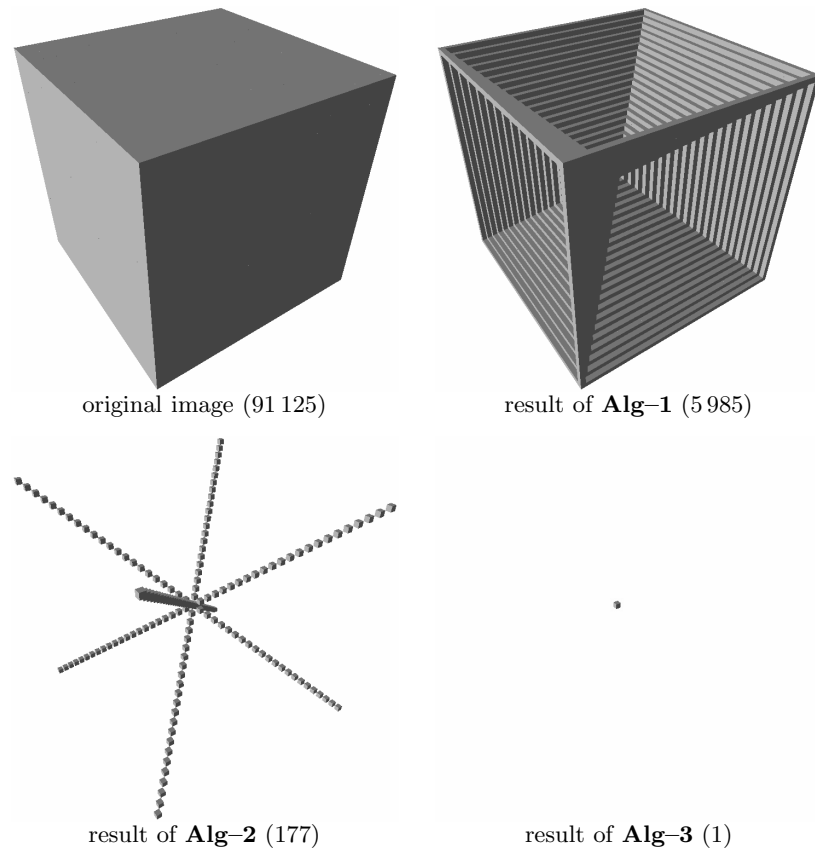


Fig. 2. The 3D image of a $45 \times 45 \times 45$ cube and its medial surfaces produced by our algorithms. Numbers in parentheses mean the count of black points.

The basic task of the function `COLLECT_DELETABLE` is comprised of two testing parts according to the conditions 2 and 3 of Definition 1. Since *border_list* contains all border points that are not surface-end point of type i in the actual picture, condition 1 of Definition 1 is satisfied by any points in *border_list*. Let us notice that the proposed algorithms **Alg-1**, **Alg-2**, and **Alg-3** can ignore condition 4 of Definition 1 (that is corresponds to condition 2 of Theorem 1), since a point that is 6-, 18-, or 26-adjacent to an interior point may not be an element of any black component contained in a $2 \times 2 \times 2$ cube.

This function uses two pre-calculated arrays as global variables. They are *LUT_simple* to encode the simple points and *Q* to encode all possible sets that are considered in condition 3 of Definition 1. We denote by “NOT” and “AND” the bitwise negation and the bitwise conjunction, respectively.

Simple points in $(26, 6)$ pictures can be locally characterized; the simplicity of a point p can be decided by examining the set $N_{26}(p)$ [12]. There are 2^{26}

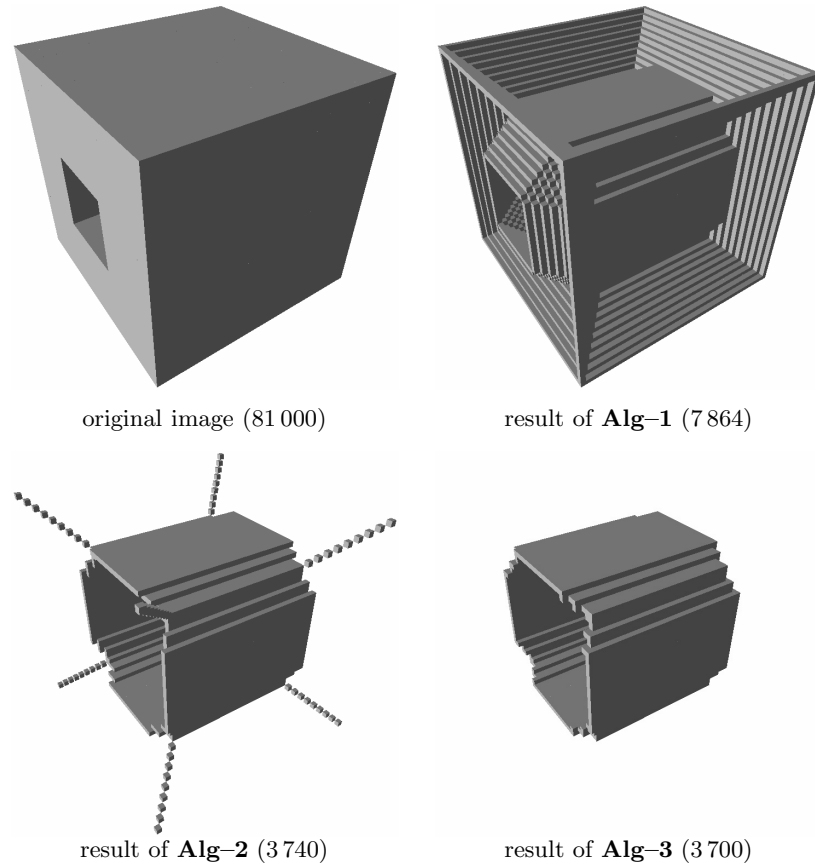


Fig. 3. The 3D image of a $45 \times 45 \times 45$ cube with a hole and its medial surfaces produced by our algorithms. Numbers in parentheses mean the count of black points.

possible configurations in the $3 \times 3 \times 3$ neighborhood if the central point is not considered. Hence we can assign an index (i.e., a non-negative integer code) for each possible configuration and address a pre-calculated (unit time access) look-up-table *LUT_simple* having 2^{26} entries of 1 bit in size, therefore, it requires only 8 megabytes storage space in memory.

It is readily confirmed that there are 66 possible sets to be verified in condition 3 of Definition 1, since a set Q may contain only one, two, or three mutually 18-adjacent points, where any point in Q is 18-adjacent to the central point p . Note that we have to investigate at most 53 of these sets since p is a border point (i.e., it is 6-adjacent to at least one white point).

Function COLLECT_26_NONZERO finds the index of the given configuration (i.e., the $3 \times 3 \times 3$ neighborhood of the point p in question excluding p itself). Its result (i.e., *index_26_nonzero*) is calculated as

$$\sum_{k=0}^{25} 2^k \cdot p_k,$$

where $p_k = 1$ if the corresponding neighbor of point p (see Fig. 1b) is an object point in the actual image A , $p_k = 0$ otherwise ($k = 0, 1, \dots, 25$). Function COLLECT_18_4 calculates the index *index_18_4* similarly: $p_k = 1$ if the corresponding neighbor is 18-adjacent to p and it has the value of “4” in the actual image A , $p_k = 0$ otherwise ($k = 0, 1, \dots, 25$). Element $Q[j]$ of the pre-calculated array Q is encoded in a similar way: $p_k = 1$ if the corresponding neighbor of point p is in the j -th set of mutually 18-adjacent points, $p_k = 0$ otherwise ($j = 1, \dots, 66$, $k = 0, 1, \dots, 25$).

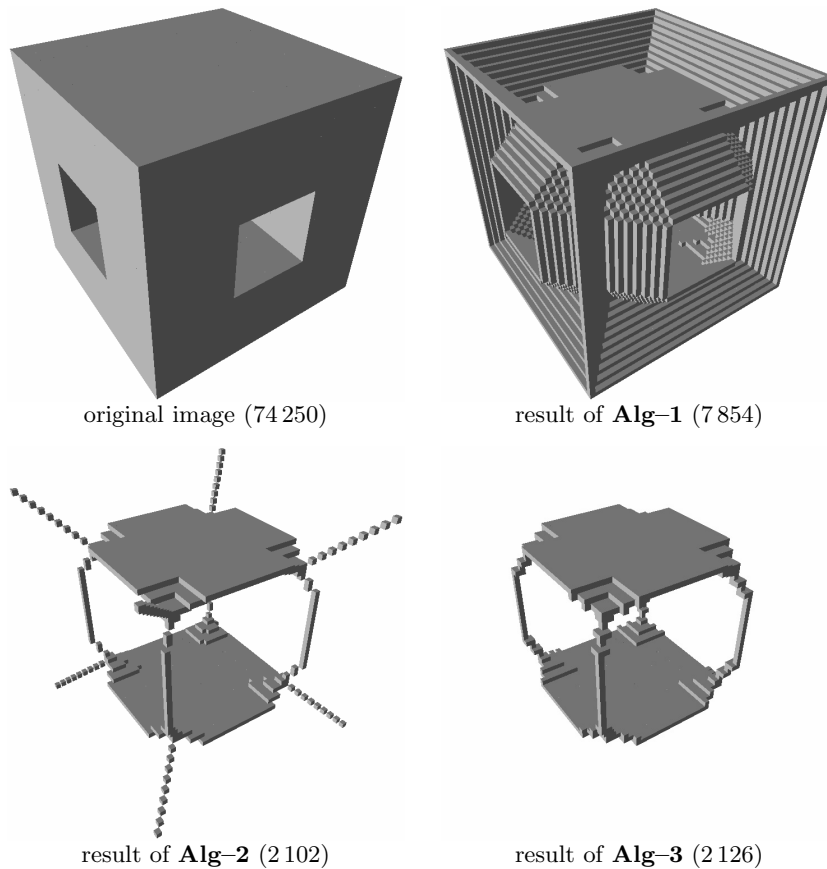


Fig. 4. The 3D image of a $45 \times 45 \times 45$ cube with two holes and its medial surfaces produced by our algorithms. Numbers in parentheses mean the count of black points.

It is important that the support of all tests that are used is $3 \times 3 \times 3$. The proposed implementation method is fairly straightforward and computationally efficient (see Table 4). Here we find that the time complexity depends only on the number of object points and the compactness of the objects (i.e., volume to area ratio); but it does not depend on the size of the volume which contains the objects to be thinned.

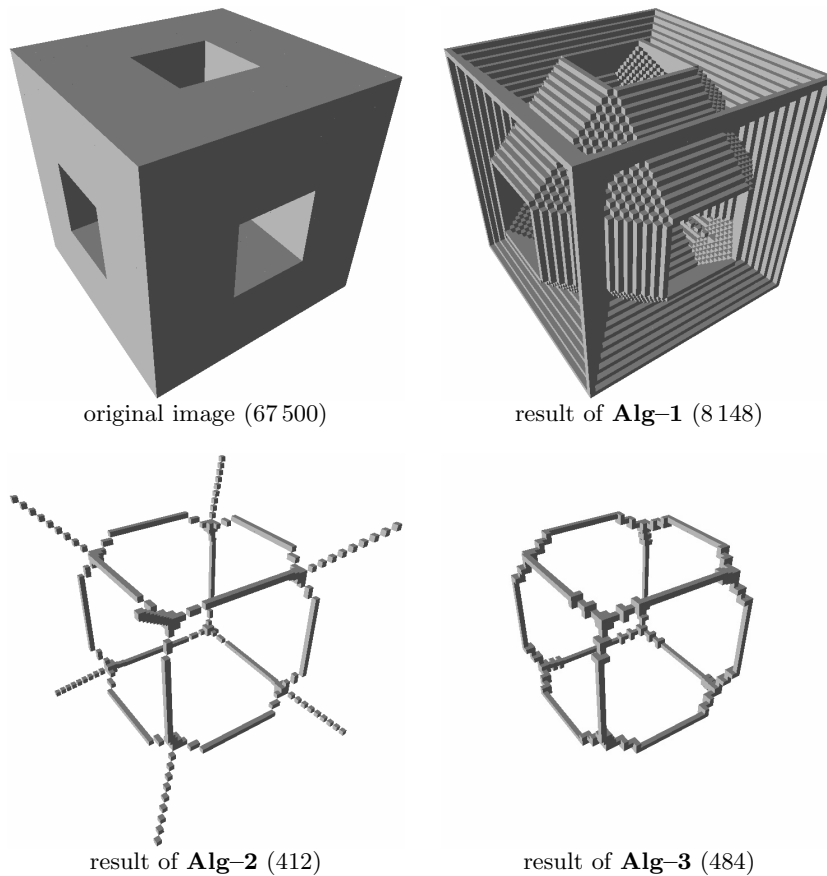






Fig. 5. The 3D image of a $45 \times 45 \times 45$ cube with three holes and its medial surfaces produced by our algorithms. Numbers in parentheses mean the count of black points.

Note that our implementation method is fairly general, as similar schemes can be used for the other parallel and some sequential thinning algorithms as well [17].

Table 1. Computation times for the considered four kinds of test objects. Our algorithms were run under Linux on an Intel Pentium 4 CPU 2.80 GHz PC. (Note, that just the thinning itself was considered here; reading the input volume, the 8 MB look-up-table, and writing the output image were not taken into account.)

Test object	Size	No. object points	No. iterations	No. skeletal points			Running time (sec.)		
				Alg-1	Alg-2	Alg-3	Alg-1	Alg-2	Alg-3
	45 ³	91 125	23	5 985	177	1	0.06	0.06	0.05
	93 ³	804 357	47	25 761	369	1	0.55	0.47	0.48
	141 ³	2 803 221	71	59 361	561	1	2.03	1.71	1.69
	45 ³	81 000	10	7 864	3 740	3 700	0.05	0.05	0.05
	93 ³	714 984	21	33 384	15 104	15 024	0.46	0.43	0.44
	141 ³	2 491 752	32	76 916	34 432	34 312	1.64	1.54	1.52
	45 ³	74 250	10	7 854	2 102	2 126	0.05	0.04	0.04
	93 ³	655 402	21	34 062	8 262	8 342	0.42	0.40	0.39
	141 ³	2 284 106	32	78 718	18 542	18 678	1.56	1.38	1.40
	45 ³	67 500	10	8 148	412	484	0.05	0.04	0.04
	93 ³	595 820	21	35 660	780	980	0.40	0.36	0.36
	141 ³	2 076 460	32	82 636	1 180	1 508	1.46	1.26	1.35

5 Conclusion and Future Work

In the present work, we proposed a new family of parallel 3D thinning algorithms. Our attention was focused on an efficient implementation of the proposed algorithms since fast skeleton extraction is extremely important for the large 3D shapes. Our future works concern subiteration and subfield based 3D thinning algorithms that are derived from sufficient conditions for parallel reduction operators to preserve the topology as well.

Acknowledgements

This research was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency.

References

1. Arcelli, C., Sanniti di Baja, G., Serino, L.: New removal operators for surface skeletonization. In: Kuba, A., Nyúl, L.G., Palágyi, K. (eds.) DGCI 2006. LNCS, vol. 4245, pp. 555–566. Springer, Heidelberg (2006)
2. Bertrand, G., Aktouf, Z.: A 3D thinning algorithm using subfields. In: Proc. SPIE Conf. on Vision Geometry III, vol. 2356, pp. 113–124 (1994)

3. Bertrand, G.: A parallel thinning algorithm for medial surfaces. *Pattern Recognition Letters* 16, 979–986 (1995)
4. Gong, W.X., Bertrand, G.: A simple parallel 3D thinning algorithm. In: *Proc. 10th IEEE Internat. Conf. on Pattern Recognition, ICPR 1990*, pp. 188–190 (1990)
5. Hall, R.W.: Parallel connectivity-preserving thinning algorithms. In: Kong, T.Y., Rosenfeld, A. (eds.) *Topological algorithms for digital image processing*, pp. 145–179. Elsevier Science, Amsterdam (1996)
6. Jonker, P.: Skeletons in N dimensions using shape primitives. *Pattern Recognition Letters* 23, 677–686 (2002)
7. Kong, T.Y.: On topology preservation in 2-d and 3-d thinning. *Int. Journal of Pattern Recognition and Artificial Intelligence* 9, 813–844 (1995)
8. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing* 48, 357–393 (1989)
9. Lee, T., Kashyap, R.L., Chu, C.: Building skeleton models via 3-D medial surface/axis thinning algorithms. *CVGIP: Graphical Models and Image Processing* 56, 462–478 (1994)
10. Ma, C.M.: On topology preservation in 3D thinning. *CVGIP: Image Understanding* 59, 328–339 (1994)
11. Ma, C.M., Wan, S.-Y.: A medial-surface oriented 3-d two-subfield thinning algorithm. *Pattern Recognition Letters* 22, 1439–1446 (2001)
12. Malandain, G., Bertrand, G.: Fast characterization of 3D simple points. In: *Proc. 11th IEEE Internat. Conf. on Pattern Recognition*, pp. 232–235 (1992)
13. Manzanera, A., Bernard, T.M., Pret  ux, F., Longuet, B.: Medial faces from a concise 3D thinning algorithm. In: *Proc. 7th IEEE Internat. Conf. Computer Vision, ICCV 1999*, pp. 337–343 (1999)
14. Pal  gyi, K., Kuba, A.: A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing* 61, 199–221 (1999)
15. Pal  gyi, K.: A 3-Subiteration Surface-Thinning Algorithm. In: Kropatsch, W.G., Kampel, M., Hanbury, A. (eds.) *CAIP 2007. LNCS*, vol. 4673, pp. 628–635. Springer, Heidelberg (2007)
16. Pal  gyi, K.: A Subiteration-Based Surface-Thinning Algorithm with a Period of Three. In: Hamprecht, F.A., Schn  rr, C., J  hne, B. (eds.) *DAGM 2007. LNCS*, vol. 4713, pp. 294–303. Springer, Heidelberg (2007)
17. Pal  gyi, K.: A 3D fully parallel surface-thinning algorithm. *Theoretical Computer Science* 406, 119–135 (2008)
18. Saha, P.K., Chaudhuri, B.B., Majumder, D.D.: A new shape-preserving parallel thinning algorithm for 3D digital images. *Pattern Recognition* 30, 1939–1955 (1997)
19. Shaked, D., Bruckstein, A.: Pruning medial axes. *Computer Vision Image Understanding* 69, 156–169 (1998)
20. Tsao, Y.F., Fu, K.S.: A parallel thinning algorithm for 3-D pictures. *Computer Graphics and Image Processing* 17, 315–331 (1981)