

A 3-Subiteration Surface-Thinning Algorithm

Kálmán Palágyi

Department of Image Processing and Computer Graphics,
University of Szeged, Hungary
palagyik@inf.u-szeged.hu

Abstract. Thinning is an iterative layer by layer erosion for extracting skeleton. This paper presents an efficient parallel 3D thinning algorithm which produces medial surfaces. A three-subiteration strategy is proposed: the thinning operation is changed from iteration to iteration with a period of three according to the three deletion directions.

1 Introduction

Skeleton is a region-based shape feature that is extracted from binary image data. A very illustrative definition of the skeleton is given using the prairie-fire analogy: the object boundary is set on fire and the skeleton is formed by the loci where the fire fronts meet and quench each other [4]. In discrete spaces, the thinning process is a frequently used method for producing an approximation to the skeleton in a topology-preserving way [7]. It is based on digital simulation of the fire front propagation: border points of a binary object that satisfy certain topological and geometric constraints are deleted in iteration steps. The entire process is repeated until only the “skeleton” is left.

A simple point is an object point whose deletion does not alter the topology of the image [9]. Sequential thinning algorithms delete simple points which are not end points, since preserving end-points provides important information relative to the shape of the objects. Curve thinning (i.e., a thinning process for extracting medial line) preserves line-end points while surface thinning (i.e., a thinning process for extracting medial surface) does not delete surface-end points.

Parallel thinning algorithms delete a set of simple points simultaneously. A possible approach to preserve topology is to use directional approach (often referred to as subiteration-based or border sequential strategy) [6]: the thinning operation is changed from iteration to iteration with a period of n ($n \geq 2$); each iteration of a period is then called a subiteration where only border points of certain kind can be deleted. Since there are six kinds of major directions in 3D images, 6-subiteration thinning algorithms were generally proposed [3,5,8,11,12,17]. Note, that 3-, 8-, and 12-subiteration algorithms were also developed [13,14,15].

In this paper, a new non-conventional 3-subiteration surface thinning algorithm is proposed. Some experiments are made on synthetic objects and the effectiveness is demonstrated.

2 Basic Notions

Let p be a point in the 3D digital space \mathbb{Z}^3 . Let us denote $N_j(p)$ (for $j = 6, 26$) the set of points j -adjacent to point p (see Fig. 1). A binary image I is a mapping ($I : \mathbb{Z}^3 \rightarrow \{0, 1\}$), that assigns value 1 to black (or object) points and value 0 is assigned to white points.

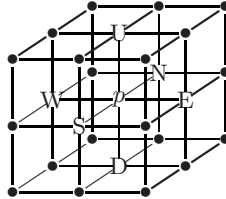


Fig. 1. The set $N_6(p)$ of the central point $p \in \mathbb{Z}^3$ contains the central point p and the 6 points marked $U = u(p)$, $N = n(p)$, $E = e(p)$, $S = s(p)$, $W = w(p)$, and $D = d(p)$. The set $N_{26}(p)$ contains $N_6(p)$ and the additional 18 points marked “•”.

We are dealing with $(26, 6)$ images [7] (i.e., the equivalence classes of the set of black points induced by the transitive closure of the 26-adjacency form the objects of the given image; white components (the background and the cavities) are the equivalence classes of the set of white points induced by the transitive closure of the 6-adjacency). It is assumed that any image contains finitely many black points.

A black point is called *border point* if it is 6-adjacent to at least one white point. A border point p is called **U-border point** in image I if $I(u(p)) = 0$ (see Fig. 1). We can define **N-, E-, S-, W-, and D-**border points in the same way. A black point p is called *interior point* if it is not border point (i.e., $I(p) = I(u(p)) = I(n(p)) = I(e(p)) = I(s(p)) = I(w(p)) = I(d(p)) = 1$). A black point is called *simple point* if its deletion does not alter the topology of the image [7]. (Note, that the simplicity of point p in a $(26, 6)$ image is a local property; it can be decided in view of $N_{26}(p)$.)

We propose a new surface thinning algorithm for extracting medial surfaces from 3D $(26, 6)$ images. The deletable points of the algorithm are border points of certain types and not *surface end-points* (i.e., which are not extremities of surfaces). The proposed algorithm uses the following characterization of the surface end-points: A black point is *surface end-point* in a image if it is border point and it is not 6-adjacent to any interior point. Note, that the same characterization has been used by other authors [1,10].

3 The New Thinning Algorithm

Each conventional 6-subiteration 3D thinning algorithm uses the six deletion directions that can delete certain **U-, D-, N-, E-, S-, and W-**border points,

respectively [3,5,8,11,12,17]. In our 3-subiteration approach, two kinds of border points can be deleted in each subiteration. The three deletion directions correspond to the three kinds of opposite pairs of points, and are denoted by **UD**, **NS**, and **EW**. The first subiteration assigned to the deletion direction **UD** can delete certain **U**- or **D**-border points; the second subiteration associated with the deletion direction **NS** attempt to delete **N**- or **S**-border points, and some **E**- or **W**-border points can be deleted by the third subiteration corresponding to the deletion direction **EW**.

The proposed algorithm is given as follows:

```

Input: binary image A
Output: binary image B
3-subiteration_thinning(A,B)
begin
    B = A;
    repeat
        B = deletion_from_UD(B);      /* 1st subiteration */
        B = deletion_from_NS(B);      /* 2nd subiteration */
        B = deletion_from_EW(B);      /* 3rd subiteration */
    until no points are deleted;
end.
    
```

The new value of a black point depends on the values of 28 additional points. The considered special neighbourhoods are presented in Fig. 2.

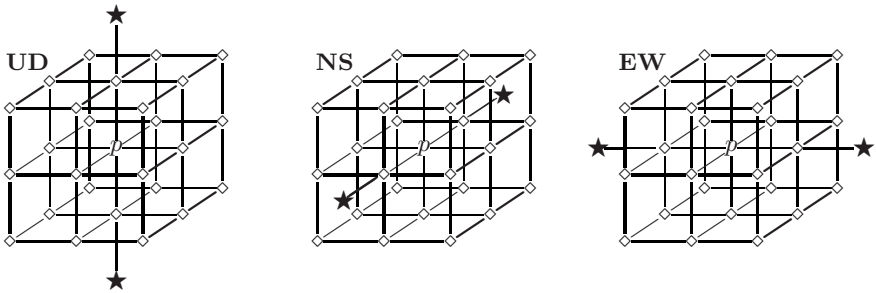


Fig. 2. The special local neighbourhoods assigned to the deletion directions **UD**, **NS**, and **EW**, respectively. The new value of a black point p depends on $N_{26}(p)$ (marked “ \diamond ”) and two additional points (marked “ \star ”).

Deletable points in a subiteration are given by a set of matching templates. A black point is deletable if at least one template in the set of templates matches it.

The set of templates T_{UD} is given by Fig. 3. Note that Fig. 3 shows only the eight base templates **TU1–TU4,TD1–TD4**. Additionally, all their rotations around the vertical axis belong to T_{UD} , where the rotation angles are 90° , 180° , and 270° . It is easy to see that the complete T_{UD} contains $2 \cdot (1 + 4 + 4 + 4) = 26$ templates. This set of templates was constructed for deleting some simple points

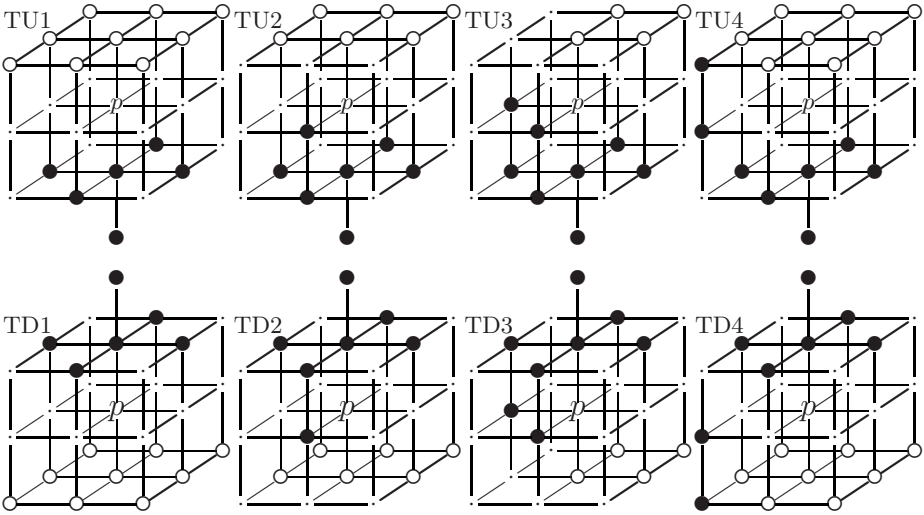


Fig. 3. Base templates **TU1–TU4,TD1–TD4** and their rotations around the vertical axis form the set of templates T_{UD} assigned to the deletion direction **UD**. This set of templates belongs to the first subiteration. Notations: each position marked “ p ” and “ \bullet ”, matches a black point; each position marked “ \circ ” matches a white point; each “.” (“don’t care”) matches either a black or a white point.

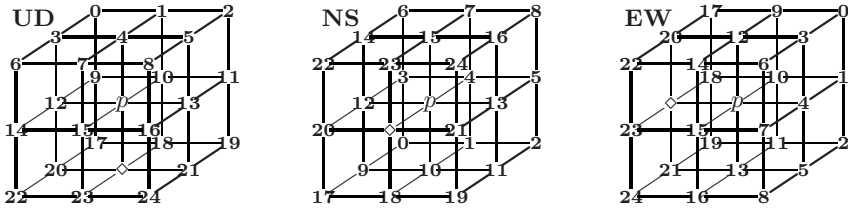


Fig. 4. Indices of the 25 Boolean variables (i.e., the considered points in $N_{26}(p)$). Note, that investigating the point marked “ \diamond ” is not needed. Since the deletion rule of a subiteration can be derived from the deletion rule of the reference subiteration **UD** by the proper rotation, the indexing scheme of a subiteration corresponds to the proper permutation of positions assigned to the reference subiteration.

which are neither surface end-points nor extremities of surfaces. The deletable points of the other two subiterations (corresponding to deletion directions **NS** and **EW**) can be obtained by proper rotations of the templates in T_{UD} .

Note that choosing another order of the deletion directions yields another algorithm. The proposed algorithm terminates when there are no more black points to be deleted. Since all considered input images are finite, it will terminate.

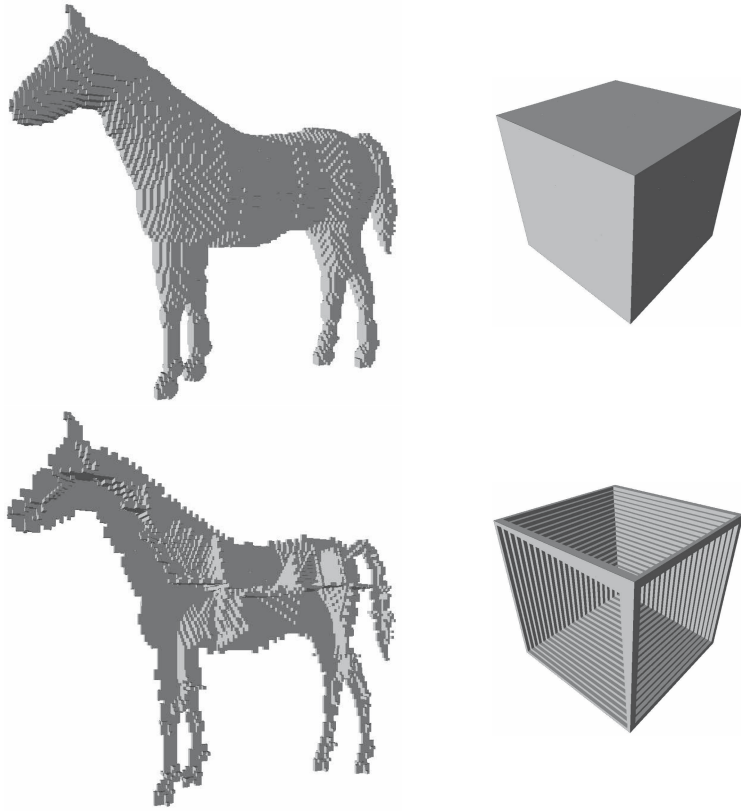


Fig. 5. Two synthetic images containing a $140 \times 140 \times 50$ horse and a $45 \times 45 \times 45$ cube (top); and their skeletons produced by the proposed surface–thinning algorithm (bottom)

Implementing the proposed algorithm seems to be rather difficult and time consuming, but it is wide of the mark. We can state that a border point in image B is to be deleted from deletion direction \mathbf{UD} if:

$$\left(\left(B(d(p))=1 \text{ and } B(u(p))=0 \text{ and } d(p) \text{ is interior point} \right) \text{ or} \right. \\ \left. \left(B(u(p))=1 \text{ and } B(d(p))=0 \text{ and } u(p) \text{ is interior point} \right) \right) \text{ and} \\ f(x_0, x_1, \dots, x_{24}) = 1,$$

where f is a Boolean–function of 25 variables derived from the set of templates. It is easy to see, that function f can be given by a pre-calculated 4 Mbyte (unit time access) look-up-table. The considered 25 variables correspond to 25 points in $N_{26}(p)$ (see Fig. 4). More details concerning the efficient implementation of 3D thinning algorithms are presented in [16].

4 Discussion and Results

Thinning algorithms have to take care of the following four aspects:

1. forcing the “skeleton” to retain the topology of the original object (i.e., topology is to be preserved);
2. providing “shape preservation” (i.e., significant features of the original object are to be produced);
3. forcing the “skeleton” to be in its geometrically correct position (i.e., in the “middle” of the object);
4. producing “maximal” thinning (i.e., the desired “width” of the “skeleton” is one point).

It is easy to see the topological correctness (the 1st requirement) by using a characterization of simple points [9] and a sufficient condition for parallel reduction operations of 3D (26, 6) images [13].

Shape preservation (the 2nd requirement) is a fairly important requirement, too. For example, an object like “b” cannot be thinned into an object like “o”. The aim of the thinning is not to produce the topological kernel [2] of an object: the thinning differs from shrinking. That is the reason why end-point criteria are used in thinning. It is easy to see that surface-end points are removed by none of our templates.

Geometrical correctness (the 3rd requirement) of the extracted skeleton is mostly achieved by the subiteration (multi-directional) thinning approach. An object is to be shrunk uniformly from each directions.

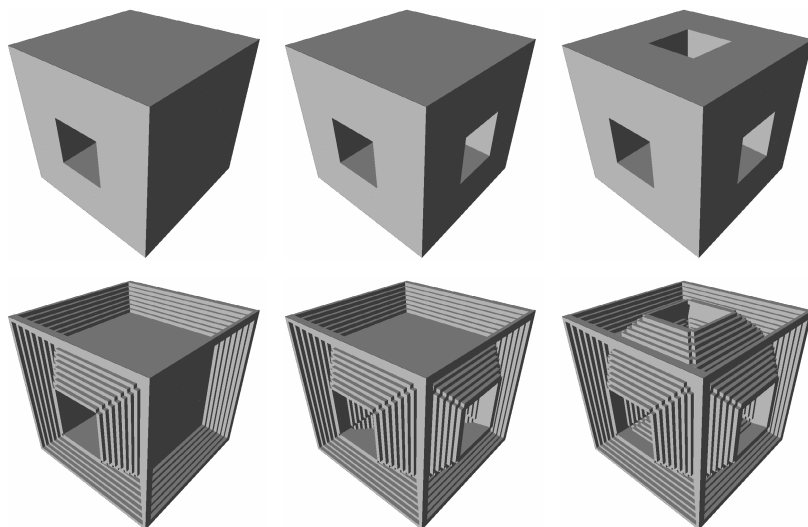







Fig. 6. Three synthetic images containing a $45 \times 45 \times 45$ cube with one, two, and three hole(s), respectively (top); and their skeletons produced by the proposed surface-thinning algorithm (bottom)

Table 1. Computation times for the considered five kinds of test objects. The implemented surface-thinning algorithm was run under Linux on an Intel Pentium 4 CPU 2.80 GHz PC. Due to the efficient implementation [16], 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 image.

test object	size	number of object points	running time (sec.)
	$140 \times 140 \times 50$	92 534	0.125
	$45 \times 45 \times 45$	91 125	0.035
	$93 \times 93 \times 93$	804 357	0.383
	$141 \times 141 \times 141$	2 803 221	1.388
	$45 \times 45 \times 45$	81 000	0.032
	$93 \times 93 \times 93$	714 984	0.359
	$141 \times 141 \times 141$	2 491 752	1.320
	$45 \times 45 \times 45$	74 250	0.028
	$93 \times 93 \times 93$	655 402	0.335
	$141 \times 141 \times 141$	2 284 106	1.268
	$45 \times 45 \times 45$	67 500	0.026
	$93 \times 93 \times 93$	595 820	0.322
	$141 \times 141 \times 141$	2 076 460	1.105

It is rather difficult to prove that the 4th requirement about maximal thinning is satisfied. Due to the used surface end-point criteria, the produced skeleton may contain 2-point thick surface patches [1,10]. It is easy to overcome this problem (e.g., by applying the final thinning step proposed by Arcelli et al. [1]).

Our algorithm has been tested on objects of different shapes. Here we present five examples (see Figs. 5–6).

The computation time of a thinning process depends on the complexity of an iteration step and the required number of iteration steps. The 3-subiteration 3D thinning strategy has been compared with other subiteration-based approaches with periods of 6, 8, or 12. It has been shown that the 3-subiteration approach requires the least number of iterations [15]. If we use unit time access look-up-tables (corresponding the deletion rules of the considered algorithms) and our efficient implementation method [16] is applied, then the 3-subiteration algorithms are the fastest subiteration-based ones. The efficiency of the proposed method is illustrated in Table 1.

Acknowledgements

The author is grateful to Stina Svensson (Centre for Image Analysis, Swedish University of Agricultural Sciences, Uppsala, Sweden) for supplying the horse image data (see Fig. 5).

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 algorithms 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. Blum, H.: A transformation for extracting new descriptors of shape. *Models for the Perception of Speech and Visual Form*, pp. 362–380. MIT Press, Cambridge (1967)
5. Gong, W.X., Bertrand, G.: A simple parallel 3D thinning algorithm. In: Proc. 10th Int. Conf. on Pattern Recognition, pp. 188–190 (1990)
6. 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)
7. Kong, T.Y., Rosenfeld, A.: Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing* 48, 357–393 (1989)
8. 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)
9. Malandain, G., Bertrand, G.: Fast characterization of 3D simple points. In: Proc. 11th IEEE Internat. Conf. on Pattern Recognition, pp. 232–235 (1992)
10. Manzanera, A., Bernard, T.M., Pretêux, F., Longuet, B.: Medial faces from a concise 3D thinning algorithm. In: Proc. 7th IEEE Internat. Conf. Computer Vision, ICCV'99, pp. 337–343 (1999)
11. Mukherjee, J., Das, P.P., Chatterjee, B.N.: On connectivity issues of ESPTA. *Pattern Recognition Letters* 11, 643–648 (1990)
12. Palágyi, K., Kuba, A.: A 3D 6-subiteration thinning algorithm for extracting medial lines. *Pattern Recognition Letters* 19, 613–627 (1998)
13. Palágyi, K., Kuba, A.: Directional 3D thinning using 8 subiterations. In: Bertrand, G., Couprie, M., Perroton, L. (eds.) DGCI 1999. LNCS, vol. 1568, pp. 325–336. Springer, Heidelberg (1999)
14. Palágyi, K., Kuba, A.: A parallel 3D 12-subiteration thinning algorithm. *Graphical Models and Image Processing* 61, 199–221 (1999)
15. Palágyi, K.: A 3-subiteration 3D thinning algorithm for extracting medial surfaces. *Pattern Recognition Letters* 23, 663–675 (2002)
16. Palágyi, K.: Efficient implementation of 3D thinning algorithms. In: Proc. 6th Conf. Hungarian Association for Image Processing and Pattern Recognition, pp. 266–274 (2007)
17. Tsao, Y.F., Fu, K.S.: A parallel thinning algorithm for 3-D pictures. *Computer Graphics and Image Processing* 17, 315–331 (1981)