Acta Cybernetica **25** (2021) 351–367.

Distance-Based Skeletonization on the BCC Grid*

Gábor Karai[ab] and Péter Kardos[ac]

Abstract

Strand proposed a distance-based thinning algorithm for computing surface skeletons on the body-centered cubic (BCC) grid. In this paper, we present two modified versions of this algorithm that are faster than the original one, and less sensitive to the visiting order of points in the sequential thinning phase. In addition, a novel algorithm capable of producing curve skeletons is also reported.

Keywords: BCC grid, distance transform, topology preservation, thinning

1 Introduction

Skeletons are region-based shape descriptors which summarize the general form of (segmented) digital binary objects [17]. In 3D, *surface skeletons* (that may contain 2D thin surface patches) are to represent general objects, and *curve skeletons* (that are stick-like 1D descriptors) are concise representations of tubular and tree-like 3D objects. Distance-based skeletonization techniques focus on the detection of ridges in the *distance map* by using a proper distance and a fast algorithm for distance transform [2, 3]. Another strategy for skeletonization, called as *thinning*, is an iterative object reduction in a topology preserving way [11]. Distance-based methods are often combined with thinning. These approaches can be further classified as follows [5]:

- *Anchor-based thinning*: First, the set of *centres of maximal balls (CMB's)* is detected as safe skeletal (anchor) points. Then thinning is applied to connect CMB's and determining the set of remaining skeletal points.
- *Distance-ordered thinning*: Distance mapping is followed by the thinning process. In the k-th iteration, only object points with distance value k are taken into consideration for possible deletion.

*This research was supported by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-00002. The project has been supported by the European Union and co-funded by the European Social Fund.

[a]Institute of Informatics, University of Szeged, Hungary
[b]E-mail: karai@inf.u-szeged.hu, ORCID: 0000-0001-9609-8628
[c]E-mail: pkardos@inf.u-szeged.hu, ORCID: 0000-0001-8857-4102

DOI: 10.14232/actacyb.290076

- *Hybrid approach*: It combines anchored and distance-ordered thinning.

Most 3D skeletonization algorithms work on digital pictures sampled on the cubic grid. An alternative structure, the *body-centered cubic* (BCC) grid tessellates the space into truncated octahedra, which results in a less ambigous connectivity structure compared to the cubic grid [21]. The importance of BCC grid shows an upward tendency in the analysis of 3D digital images [1, 13, 14, 15, 24].

The first skeletonization algorithm for binary objects sampled on the BCC grid was proposed by Strand [19]. His algorithm considers a fixed distance, and is only capable of producing surface skeletons. According to our best knowledge, there is not any other similar result in the later literature. In this paper, we present two modified versions of Strand's method that work for arbitrary distances, and a further one capable of producing curve skeletons.

The rest of this work is organized as follows. Section 2 reviews the basic notions and results of 3D digital topology. In Section 3, we discuss the original algorithm of Strand working on the BCC grid. In Section 4, we propose the modified versions of that algorithm for computing surface skeletons, while in Section 5, we also present a variant to produce curve skeletons. Some experimental results are shown in Section 6, and we round off this work with some concluding remarks.

2 Definitions and Notions

In this section, we review the basic concepts of digital topology and distance transform.

Let us denote by $\mathbb{B}$ the BCC grid, whose elements are called *points*. The BCC grid is defined as the following subset of $\mathbb{Z}^3$:

$$\mathbb{B} = \{(x, y, z) \in \mathbb{Z}^3 \mid x \equiv y \equiv z \pmod 2\}. \tag{1}$$

We make a distinction among the following three types of neighborhood of a point $p = (p_x, p_y, p_z) \in \mathbb{B}$ (see Figure 1):

$$N_6(p) = \{(q_x, q_y, q_z) \in \mathbb{B} \mid |p_x - q_x| + |p_y - q_y| + |p_z - q_z| = 2\},$$

$$N_8(p) = \{(q_x, q_y, q_z) \in \mathbb{B} \mid |p_x - q_x| + |p_y - q_y| + |p_z - q_z| = 3\},$$

$$N_{14}(p) = N_6(p) \cup N_8(p).$$

Two points $p, q \in \mathbb{B}$ are *i-adjacent* if $q \in N_i(p)$ $(i = 6, 8, 14)$. Furthermore, let $N_i^*(p) = N_i(p) \setminus \{p\}$.

The *lexicographical order* relation "$\prec$" between two distinct points $p = (p_x, p_y, p_z)$ and $q = (q_x, q_y, q_z)$ in $\mathbb{B}$ is defined as follows:

$$p \prec q \quad \Leftrightarrow \quad (p_z < q_z) \vee (p_z = q_z \wedge p_y < q_y) \vee (p_z = q_z \wedge p_y = q_y \wedge p_x < q_x).$$

The sequence of distinct points $\langle x_0, x_1, \ldots, x_s \rangle$ in a non-empty set of points $X \subset \mathbb{B}$ is called an *i-path* of length s from x_0 to x_s in X if x_k is i-adjacent to x_{k-1}

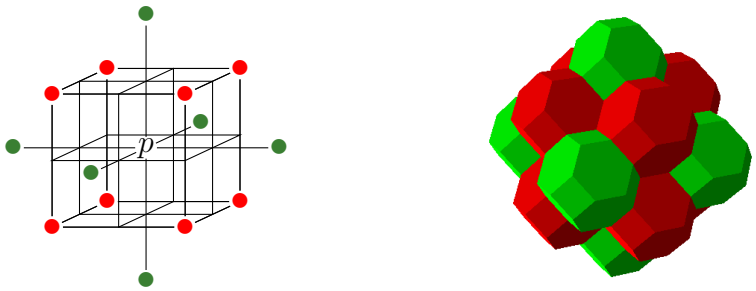


Figure 1: Elements of $N_{14}(p)$ in $\mathbb{B}$ (left), voxel representation of these points (right). The set $N_6(p)$ contains the central point p and the six points (voxels) marked green. The set $N_8(p)$ contains p and the eight points (voxels) marked red.

($i \in \{6, 8, 14\}, k = 1, \ldots, s$). Note that a single point is an *i-path* of length 0. Two points $p, q \in \mathbb{B}$ are said to be *i-connected* in $X \subseteq \mathbb{B}$ if there is an i-path from p to q in X. The set X is *i-connected* in the set of points $Y \supseteq X$ if any two points in X are i-connected in Y.

Following the concept of digital pictures in [9], we define the $(14, 14)$ *binary digital picture* as a quadruple $\mathcal{P} = (\mathbb{B}, 14, 14, B)$. Each point in $B \subseteq \mathbb{B}$ is called a *black point* and has a value of 1 assigned to it. Picture $\mathcal{P}$ is finite if it contains finitely many black points. Each point in $\mathbb{B} \setminus B$ is called a *white point* and has a value of 0 assigned to it. 14-adjacency is associated with both black and white points. A *black component* or an *object* is a maximal 14-connected set of points in B, while a *white component* is a maximal 14-connected set of points in $\mathbb{B} \setminus B$. A black point is called a *border point* in a picture if it is 14-adjacent to at least one white point. Furthermore, a border point p is called a *strong border point*, if $N_8(p)$ contains at least one white point, or else p is called a *weak border point*. In a finite picture, there is a unique white component that is called the *background*. A finite white component is called a *cavity*.

A *reduction* transforms a binary picture only by changing some black points to white ones (which is referred to as the deletion of 1's). A 3D reduction does *not* preserve topology [8] if

- any object in the input picture is split (into several objects) or is completely deleted,
- any cavity in the input picture is merged with the background or another cavity,
- a cavity is created where there was none in the input picture, or
- a hole (that e.g. doughnuts have) is eliminated or created.

A *simple point* is a point whose deletion is a topology preserving reduction [9]. Sequential thinning algorithms traverse the border points of a picture, and focus on

the actually visited single point for possible deletion, hence for such algorithms, the deletion of only simple points ensures topology preservation. The following result states that simplicity is a local property in $\mathbb{B}$ which can be verified by investigating the 14-neighborhood of points:

Theorem 1. [23] *Let p be a black point in a $(\mathbb{B}, 14, 14, B)$ picture. Then p is a simple point if and only if the following conditions hold:*

1. *Point p is 14-adjacent to just one 14-component of $N_{14}^*(p) \cap B$.*
2. *Point p is 14-adjacent to just one 14-component of $N_{14}(p) \setminus B$.*

Figure 2 shows examples for simple and non-simple points.

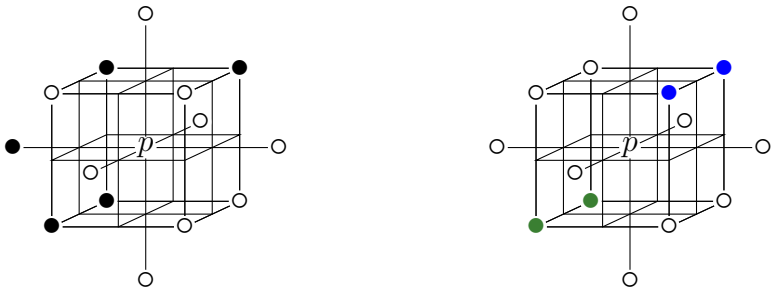


Figure 2: Example for a simple (left) and a non-simple point p (right) on the BCC grid. The distinct black 14-components are labeled with different colors.

Let Γ be the set of white points in a picture. The *distance transform* is a function that assigns to each point p the distance between p and the closest border point $q \in \Gamma$ to it. The most frequently used distance functions of two points p and q are the Euclidean distance, $d_e(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2}$ and the neighborhood distances, where $d_i(p,q)$ denotes the length of the shortest i-path between p and q [12]. In $\mathbb{B}$, neighborhood distances $d_8(p,q)$ and $d_{14}(p,q)$ are taken into consideration. These distances, however, do not give a good approximations to the Euclidean distance. As a better solution for this purpose, the lengths of the moves from a point to its neighbors can be weighted according to some criteria, and the path of the minimal sum of weights called as *Chamfer distance*s can be used [4, 12]. Let $\langle a, b \rangle$ denote the general *Chamfer mask* for weighted distance transform on the BCC grid, where a and b are the weights assigned to all points in $N_8^*(p)$ and $N_6^*(p)$, respectively. Some Chamfer masks are presented in [7, 22].

Let DT be a distance map of a $(14, 14)$ picture. Point $p \in B$ is called a *local maximum* if for any point $q \in N_{14}^*(p)$, $DT(p) \geq DT(q)$. An *inscribed ball* is a sphere that is contained within the object and tangent to at least one of the object's faces, and it's called *maximal* if it is not covered by any other inscribed ball. A *center of maximal ball (CMB)* is labeled with the radius of that ball [5]. Point $p \in B$ is a CMB if for any point $q \in N_{14}^*(p)$, $DT(q) < DT(p) + w$, where w is the weight that

corresponds to q on the Chamfer mask when placed on p. In case of d_8 distance, only N_8^* is considered for detection of local maxima and CMB's.

3 Strand's algorithm

The method proposed by Strand uses d_{14} distance and sequential thinning, and it can produce only surface skeletons by preserving non-simple points and surface edge points. A point $p \in B$ is called a *surface edge point*, if the following two conditions hold:

- there are two points $q, r \in N_{14}^*(p) \cap B$ such that $N_{14}(q) \cap N_{14}(r) = \{p\}$ and
- there is no point $s \in N_{14}^*(p) \cap B$ such that $N_{14}(p) \cap N_{14}(s) \subseteq B$.

The thinning part consists of two phases. Forward thinning reduces the input object to a 3–4 voxel thick object, which is peeled further during backward thinning. Note that surface edge point condition is applied only in the last phase. The unusual in this process is that during the backward thinning phase, object points are visited in descending order of their distance value. The sequential thinning suffers from the disadvantage of being sensitive to the visiting order of border points with the same distance value. As a result the final skeleton usually has some "false" skeleton points, which causes insignificant segments. This situation is illustrated in Figure 3 and an example for its occurrence can be found in Figure 4.

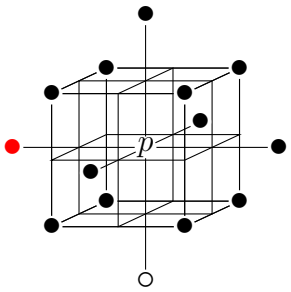


Figure 3: Example for unfavorable visiting order during sequential thinning. Weak border point p becomes non-simple after deletion of its neighbor marked red.

Theorem 2. *The runtime complexity of Strand's algorithm (see Algorithm 1) is* $O(|I|^{4/3})$*, where* $|I|$ *denotes the number of points in the image.*

Proof. The first phase of the algorithm is linear since computing the d_{14} distance transform requires two raster scans [22] and the sequential detection of CMB's takes one extra scan. Forward thinning is linear too, because the object point's local neighborhood is examined exactly once. In worst case, weak border points on the original object may appear as surface or line endpoints, which means the

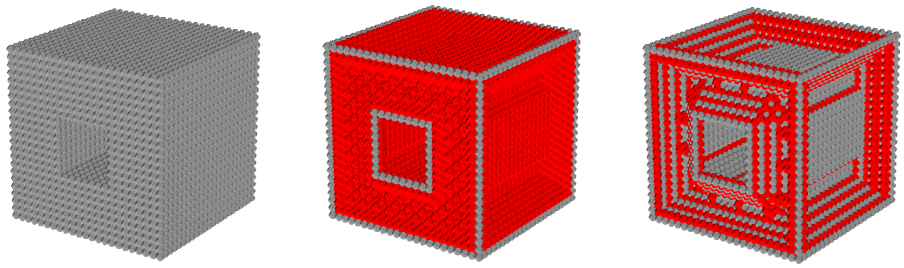
Algorithm 1 Strand's skeletonization algorithm.

Input: picture $(\mathbb{B}, 14, 14, X)$ with the initial objects in it
Output: picture $(\mathbb{B}, 14, 14, X)$ containing the surface skeleton

```
   // Distance mapping and identifying CMB's
 1: DT ← computeDT(X)
 2: H ← {p ∈ X | p is a CMB}
   // Forward thinning
 3: for k ← 1 to max(DT) do
 4:   for each p ∈ X do
 5:     if DT(p) = k and p is simple and N14(p) ∩ H = ∅ then
 6:       X ← X \ {p}
 7:     end if
 8:   end for
 9: end for
   // Backward thinning
10: repeat
11:   changed1 ← false
12:   for k ← max(DT) downto 1 do
13:     repeat
14:       changed2 ← false
15:       D ← {p ∈ X \ H | DT(p) = k and p is simple}
16:       for each p ∈ D do
17:         if p is simple and p is not a surface edge point then
18:           X ← X \ {p}
19:           changed1 ← true
20:           changed2 ← true
21:         end if
22:       end for
23:     until changed2 = false
24:   end for
25: until changed1 = false
```

algorithm may classify black points incorrectly during the first iteration. As a consequence, the length of these segments is equal to half of the object's thickness, which we refer to as T_{obj}.

During the *for* loop from Step 12 to 24, all remaining object points are visited, but it can peel only a one-unit layer from each segment because of the descending visiting order of the object point's distance value, since only simple points can be deletable. Hence, each object point will be visited at most T_{obj} times until the algorithm terminates, so the runtime complexity is $O(T_{obj} \cdot |I|)$. T_{obj} is maximal, if the input image has a cubic shape, which means the image size is the same for all dimensional axes. In this case, the original object is $\sqrt[3]{|I|}\,/\,2$ thick, if all points in the image are black. By inserting it into the previous formula, we get



Original object (20 280) | Result of forward thinning | Final skeleton (4 926)

Figure 4: Result of Strand's algorithm on a holey cube. In the middle and right figure anchor voxels are gray and further skeleton voxels are red. Numbers in parentheses show the number of black points.

$O(\sqrt[3]{|I|} / 2 \cdot |I|) = O(|I|^{4/3})$, which is not just the third phase's, but the whole algorithm's runtime complexity. □

4 Our two modified algorithms to produce surface skeletons

To construct linear time algorithms, we merge the thinning phases in Algorithm 1 and simplify the organization of the thinning iterations. Our deletion rule also preserves non-simple points and surface edge points. We introduce a new parameter N which gives an upper limit to the visiting number of all black points during the thinning phase (i.e. each iteration will be repeated at most N times). Note that setting N to ∞ means that the visiting number is not limited. This parameter is applied in both of our improved algorithms. The motivation is to reduce the sensitivity to the visiting order of border points. The sequential thinning part consists of two substeps in our modified algorithms. First, we collect the deletable points from the actual image into set D. Then we individually delete each point in this set, if they are still simple when visited. In order to lessen the occurrence of false skeleton points, elements of D are visited in lexicographical order.

Our first improved variant, called *AB-N-visits*, is an anchor-based thinning method that considers local maxima instead of CMB's to be anchor points because many CMB's are proved to be "false" skeleton points on weighted distance maps [5]. In this approach, we consider only strong border points to make sure the strong and weak border points will be visited in different iterations. At the end of each iteration, all visited but non-deleted points are insterted in the set S of skeleton points (see Step 17 of Algorithm 2). This operation guarantees that no border points will be examined during the further iterations which already have been visited.

Our second improved variant, called *DO-N-visits*, is a distance-ordered thinning method that omits the detection of anchor points, i.e. the set H is not used in this version. The border points are visited in ascending order of their distance value during the thinning phase.

Algorithm 2 Anchor-based variant *AB-N-visits* to extract surface skeleton.

Input: picture $(\mathbb{B}, 14, 14, X)$ with the initial objects in it, visiting limit N
Output: picture $(\mathbb{B}, 14, 14, S)$ containing the surface skeleton

$DT \leftarrow computeDT(X)$
$H \leftarrow \{p \in X \mid p \text{ is a local maximum}\}$
$S \leftarrow \emptyset$
`// Thinning`
repeat
  $L \leftarrow \{p \in X \setminus (S \cup H) \mid p \text{ is a } \textit{strong border point}\}$
  $D \leftarrow \{p \in L \mid p \text{ is } \textit{simple} \text{ for } X$ **and** p is not a *surface edge point*$\}$
  $t \leftarrow 0$
  repeat
    $t \leftarrow t + 1$
    for each $p \in D$ in lexicographical order **do**
      if p is simple **then**
        $X \leftarrow X \setminus \{p\}$
        $L \leftarrow L \setminus \{p\}$
      end if
    end for
  until $t = N$ **or** no points are deleted
  $S \leftarrow S \cup L$
until $D = \emptyset$

Theorem 3. *The runtime complexity of Algorithm 2 and Algorithm 3 is linear, if* $N \in \mathbb{N}$.

Proof. As we mentioned in Theorem 2, d_{14} distance mapping has linear computational cost. Here we note that d_8 and any $\langle a, b \rangle$ Chamfer distance have the same property [7]. Moreover, there are also linear time adaptations of the Euclidean distance transform on the BCC grid [20].

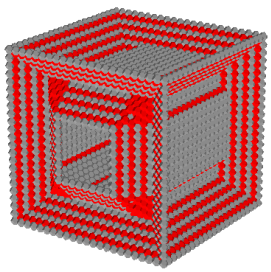
Following the indication in Theorem 2, $|I|$ denotes the number of points in the image. During the thinning phase, each object point is visited in exactly one iteration. In Algorithm 2, this is ensured by collecting all previously investigated but not deleted border points. In Algorithm 3, the distance-ordered strategy guarantees this property. It is also obvious that each border point is visited up to N times during one thinning iteration. As a consequence, all object points are visited maximum N times during the thinning phase. Hence, the computational cost is $O(N \cdot |I|)$, which means linear time complexity if N is a positive integer since it is a fixed parameter. □

Algorithm 3 Distance-ordered variant *DO-N-visits* to extract surface skeleton.

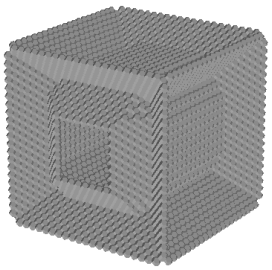
Input: picture $(\mathbb{B}, 14, 14, X)$ with the initial objects in it, visiting limit N
Output: picture $(\mathbb{B}, 14, 14, X)$ containing the surface skeleton

```
1: DT ← computeDT(X)
   // Thinning
2: for k ← 1 to max(DT) do
3:    D ← {p ∈ X | DT(p) = k and p is simple and p is not a surface edge point}
4:    t ← 0
5:    repeat
6:       t ← t + 1
7:       for each p ∈ D in lexicographical order do
8:          if p is simple then
9:             X ← X \ {p}
10:         end if
11:      end for
12:   until t = N or no points are deleted
13: end for
```

The extracted surface skeletons of the holey cube by our improved algorithms are shown in Figure 5. It is easy to see that these surface skeletons contain less insignificant skeleton points compared to the result in Figure 4.



AB-N-visits
(4 432)



DO-N-visits
(3 881)

Figure 5: Result of our improved algorithms on a holey cube shown in Figure 4 on d_{14} distance map with $N = 2$. In the left figure anchor voxels are gray and further skeleton voxels are red. Numbers in parentheses indicate the number of skeleton points.

5 Modified variant to produce curve skeletons

By modifying our distance-ordered variant, we can also extract the curve skeleton directly from the original object. For this purpose, instead of surface edge points we retain either of two types of curve endpoints. Point $p \in B$ is a *C1 endpoint*, if $|N_{14}^*(p) \cap B| = 1$, i.e., p has only one black neighbor in $N_{14}^*(p)$. Let u be this black neighbor. Point p is a *C2 endpoint*, if p is a *C1* endpoint and u is a line point. Point u is a *line point* if there are exactly 2 black points in its 14-neighborhood and they are not 14-adjacent to each other. We consider a border point deletable if it is simple but not an endpoint. The corresponding algorithm is called *DO-CS*. We reorganised the thinning iterations: the non-deleted black points must be visited again during the further iterations in order to shrink the surface patches until line branches are only left. Parameter N is not used since the sufficient number of iterations repetition depends on the structure of the objects in the input picture. This algorithm terminates only if there are no more deletable points. Therefore, it is impossible to set a constant upper limit k to the visiting number of object points. As a result, the extraction of the curve skeleton has nonlinear time complexity.

It is important to note that anchor point condition should not be used because the detected ridge points belong to the surface skeleton, so surface segments will probably be also generated.

According to our experiments, visiting elements of D in lexicographical order has a relevant effect only on d_8 and d_{14} distance maps. The reason behind this phenomenon is the fact that on Chamfer or Euclidean distance maps the number of considered border points in a thinning iteration are less than in d_{14} or d_8 distance map, so there is a lower chance of unfavorable visiting order. Furthermore, there are much more configurations for surface edge points than for line endpoints. Hence, the curve endpoint criteria are less sensitive to the visiting order, especially the $C2$ condition.

Extracted surface skeletons of the holey cube are shown in Figure 6. Notice that d_8 skeleton is much more jagged due to the unlucky visiting order of border points.

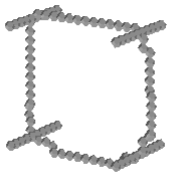
6 Results

Our algorithms were tested on numerous objects of different shapes. Figures 7–10 show the surface skeletons and 11–13 show the curve skeletons. To make the difference between the applied parameters more visible, some of the following figures consist of fusioned images, where red, green and gray voxels belong to the first, second and both skeletons extracted with the indicated parameters, respectively. In the case of Strand's method the gray and red points mean the anchor points and further skeletal points, respectively. In case of any $\langle a, b \rangle$ weighted distances, local maxima are considered as anchor points instead of CMB's even for Strand's method. Numbers in parentheses show the number of object or skeleton points.

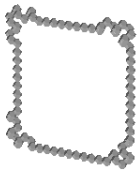
Algorithm 4 Variant *DO-CS* to extract the curve skeleton.

Input: picture $(\mathbb{B}, 14, 14, X)$ with the initial objects in it, line endpoint criterion Ci $(i \in \{1, 2\})$
Output: picture $(\mathbb{B}, 14, 14, X)$ containing the curve skeleton

```
1: DT ← computeDT(X)
   // Thinning
2: for k ← 1 to max(DT) do
3:   repeat
4:     D ← {p ∈ X | DT(p) ≤ k and p is simple and p is not a Ci endpoint}
5:     repeat
6:       for each p ∈ D do
7:         if p is simple then
8:           X ← X \ {p}
9:         end if
10:      end for
11:    until no points are deleted
12:  until D = ∅
13: end for
```



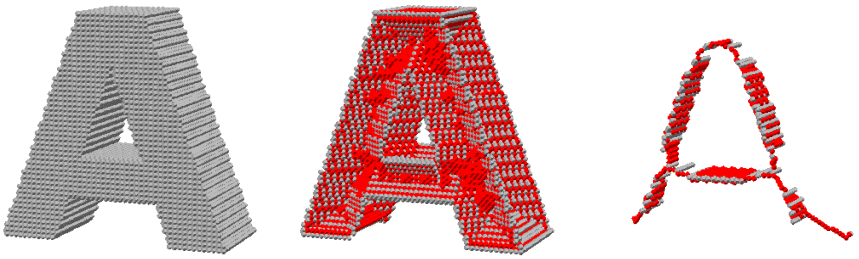
d_{14}
(97)



d_8
(84)

Figure 6: Produced curve skeletons of a holey cube shown in Figure 4 with different distances. The resulting curve skeletons with C1- and C2-endpoint condition coincide with each other because no C2-endpoint was detected. Numbers in parentheses indicate the number of skeleton points.

As we discussed in Section 3, Strand's algorithm leaves many insignificant skeleton segments in case of d_{14} distance. It's easy to see that Strand's method remarkably overshrinks the input object, when weighted distance is used. Also note that the more accurate approximation of the Euclidean distance is used, the less anchor points are detected. In the case of $N = 1$, algorithm *DO-N-visits* leaves one side of the letter A almost unchanged due to the unfavorable visiting order of border points. This phenomenon is successfully handled with setting N to 2. However, the modified versions may overshrink the object, if N is too large, especially in d_8 or d_{14} distance maps (see Figure 8). Hence, options $N \in \{2, 3\}$ are usually sufficient.



Original object
(42 522)

d_{14}
Strand
(8 805)

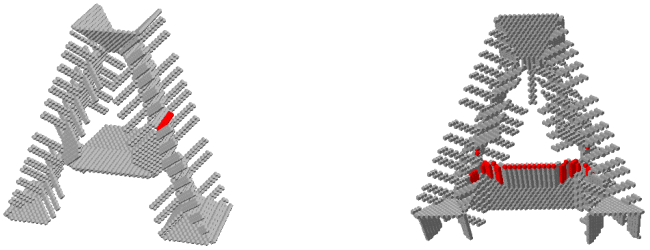
$\langle 5, 6\rangle$
Strand
(1 110)

$\langle 6, 7\rangle$, $N = 1, 2$
AB-N-visits
(4 365) + (4 329)

d_{14}, $N = 1$
DO-N-visits
(6 162)

d_{14}, $N = 2$
DO-N-visits
(3 918)

Figure 7: Produced surface skeletons of the letter A with various parameters.



d_8, $N = 2, \infty$
AB-N-visits
(4 873) + (4 852)

d_{14}, $N = 2, \infty$
DO-N-visits
(3 918) + (3 792)

Figure 8: Overshrinked surface skeletons of the letter A. For $N = \infty$, the highest repetition number, i.e. the highest value of t in Algorithm 2 and Algorithm 3 was 14 and 12, respectively.

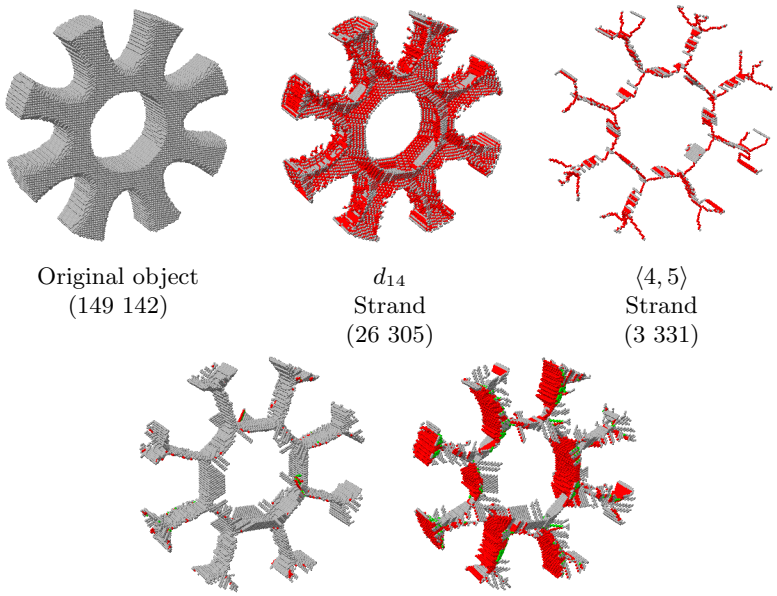


Figure 9: Produced surface skeletons of a gear with various parameters.

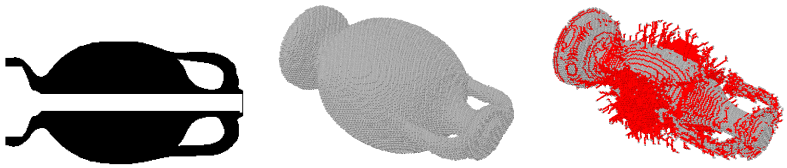
We can also observe that the medial surface is strongly jagged due to the nature of sequential thinning.

A similar phenomenon can be observed on the gear and the amphora. By setting N to 2 we get a much cleaner skeleton. Note that the number and distribution of skeletal points also depend on the thinning strategy.

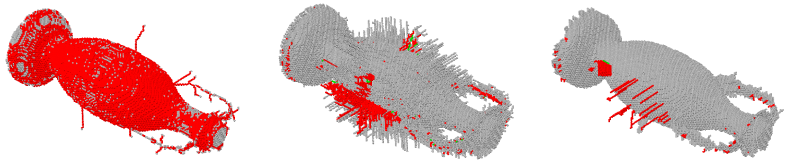
It can be well observed that significantly less false line segments were produced on the curve skeletons with condition $C2$ compared to $C1$.

7 Conclusions

The proposed algorithms *AB-N-visits* and *DO-N-visits* have linear runtime complexity and are less sensitive to the visiting order of border points compared to Strand's method. According to our experiments, it is sufficient to set N to at most 3 to produce "clear" surface skeletons. All examined algorithms preserve topology due to the fact that only a single simple point is deleted at a time. All the proposed methods can be extended to arbitrary distances including any $\langle a, b \rangle$ weighted distance or even the Euclidean distance. The optimal parameters (e.g. distance, thinning strategy) depend on the shapes of the objects to be represented. We note that, though numerous authors have proposed methods to evaluate the performance

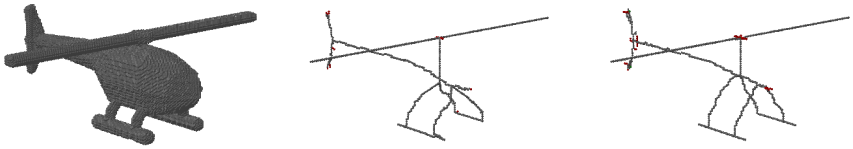


Cross-sectional image | Original object (566 765) | d_{14} Strand (51 901)



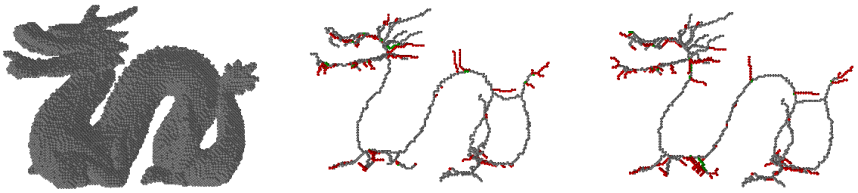
$\langle 3,4\rangle$ Strand (25 892) | d_{14}, $N=1,2$ AB-N-visits (53 522) + (51 493) | d_e, $N=1,2$ DO-N-visits (29 749) + (29 246)

Figure 10: Produced surface skeletons of an amphora with various parameters.



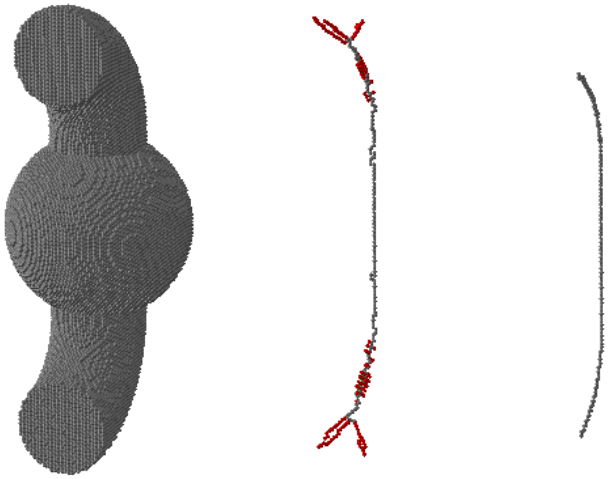
Original object (94 031) | d_8 C1 + C2 (539) + (524) | $\langle 3,4\rangle$ C1 + C2 (576) + (537)

Figure 11: Extracted curve skeletons of a helicopter with various parameters.



Original object (105 933) | $\langle 4,5\rangle$ C1 + C2 (1 131) + (861) | d_e C1 + C2 (1 153) + (818)

Figure 12: Extracted curve skeletons of a dragon with various parameters.



Original object (212 639) | d_{14} C1 + C2 (283) + (145) | d_e C1 + C2 (121) + (121)

Figure 13: Extracted curve skeletons of an object with various parameters. In the right figure, the C1- and C2-skeletons are coincide with each other since no C2-endpoint was detected.

of skeletonization algorithms (see for example [6, 10, 18, 25]), due to the lack of definition of the "true skeleton" for a discrete object, a widely accepted approach evaluating the goodness of skeletonization algorithms is yet absent [16].

Future research should be devoted to constructing similar distance-based algorithms combined with parallel thinning strategies and adapting the presented results to the face-centered cubic grid [21].

References

[1] Ankudinov, V. and Galenko, P. K. Growth of different faces in a body centered cubic lattice: A case of the phase-field-crystal modeling. *Journal of Crystal Growth*, 539:125608, 2020. DOI: 10.1016/j.jcrysgro.2020.125608.

[2] Arcelli, C. and Sanniti di Baja, G. Finding local maxima in a pseudo-Euclidian distance transform. *Computer Vision, Graphics, and Image Processing*, 43(3):361–367, 1988. DOI: 10.1016/0734-189X(88)90089-8.

[3] Arcelli, C. and Sanniti di Baja, G. Ridge points in Euclidean distance maps. *Pattern Recognition Letters*, 13(4):237–243, 1992. DOI: 10.1016/0167-8655(92)90074-A.

[4] Borgefors, G. Distance transformations in arbitrary dimensions. *Computer Vision, Graphics, and Image Processing*, 27(3):321–345, 1984. DOI: 10.1016/0734-189X(84)90035-5.

[5] Borgefors, G., Nyström, I., and Sanniti di Baja, G. *Discrete Skeletons from Distance Transforms in 2D and 3D.* In *Medial Representations: Mathematics, Algorithms and Applications*, pages 155–190. Springer Netherlands, 2008. DOI: 10.1007/978-1-4020-8658-8_5.

[6] Cornea, N. D., Silver, D., and Min, P. Curve-Skeleton Properties, Applications, and Algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548, 2007. DOI: 10.1109/TVCG.2007.1002.

[7] Fouard, C., Strand, R., and Borgefors, G. Weighted distance transforms generalized to modules and their computation on point lattices. *Pattern Recognition*, 40(9):2453–2474, 2007. DOI: 10.1016/j.patcog.2007.01.001.

[8] Kong, T.Y. On topology preservation in 2-d and 3-d thinning. *International Journal of Pattern Recognition and Artificial Intelligence*, 09(05):813–844, 1995. DOI: 10.1142/S0218001495000341.

[9] Kong, T.Y. and Rosenfeld, A. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48(3):357–393, 1989. DOI: 10.1016/0734-189X(89)90147-3.

[10] Kruszynski, K. J., Liere van, R., and Kaandorp, J. A. Quantifying differences in skeletonization algorithms: a case study. In Villanueva, J.J., editor, *Visualization, Imaging, and Image Processing (Proceedings 5th IAESTED International Conference, VIIP'05, Benidorm, Spain, September 7–9, 2005)*, Canada, 2005. ACTA Press.

[11] Lam, L., Lee, S.-W., and Suen, C. Y. Thinning methodologies - a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992. DOI: 10.1109/34.161346.

[12] Marchand-Maillet, S. and Sharaiha, Y. M. *Binary Digital Image Processing: A Discrete Approach.* Academic Press, 2000. DOI: 10.1016/B978-0-12-470505-0.X5000-X.

[13] Mujahed, H. and Nagy, B. Wiener Index on Lines of Unit Cells of the Body-Centered Cubic Grid. In Benediktsson, Jón Atli, Chanussot, Jocelyn, Najman, Laurent, and Talbot, Hugues, editors, *Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 597–606, Cham, 2015. Springer International Publishing. DOI: 10.1007/978-3-319-18720-4_50.

[14] Čomić, L. and Magillo, P. Repairing 3D binary images using the BCC grid with a 4-valued combinatorial coordinate system. *Information Sciences*, 499:47–61, 2019. DOI: 10.1016/j.ins.2018.02.049.

[15] Čomić, L. and Nagy, B. A combinatorial coordinate system for the body-centered cubic grid. *Graphical Models*, 87:11–22, 2016. DOI: 10.1016/j.gmod.2016.08.001.

[16] Saha, P. K., Borgefors, G., and Sanniti di Baja, G. A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters*, 76(1):3–12, 2016. DOI: 10.1016/j.patrec.2015.04.006, Special Issue on Skeletonization and its Application.

[17] Saha, P. K., Borgefors, G., and Sanniti di Baja, G. *Skeletonization: Theory, Methods and Applications*. Academic Press, 2017.

[18] Saha, P. K., Chaudhuri, B. B., and Majumder, D. D. A new shape preserving parallel thinning algorithm for 3d digital images. *Pattern Recognition*, 30(12):1939–1955, 1997. DOI: 10.1016/S0031-3203(97)00016-2.

[19] Strand, R. Surface skeletons in grids with non-cubic voxels. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, pages 548–551, Cambridge, 2004. DOI: 10.1109/ICPR.2004.1334195.

[20] Strand, R. The Euclidean Distance Transform Applied to the FCC and BCC Grids. In Marques, J. S., de la Blanca, N. P., and Pina, P., editors, *Pattern Recognition and Image Analysis*, pages 243–250, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. DOI: 10.1007/11492429_30.

[21] Strand, R. The face-centered cubic grid and the body-centered cubic grid: a literature survey. Technical Report 35, Centre for Image Analysis, Uppsala University, Uppsala, Sweden, 2005.

[22] Strand, R. and Borgefors, G. Distance transforms for three-dimensional grids with non-cubic voxels. *Computer Vision and Image Understanding*, 100(3):294–311, 2005. DOI: 10.1016/j.cviu.2005.04.006.

[23] Strand, R. and Brunner, D. Simple Points on the Body-Centered Cubic Grid. Technical Report 42, Centre for Image Analysis, Uppsala University, Uppsala, Sweden, 2006.

[24] Strand, R., Nagy, B., and Borgefors, G. Digital distance functions on three-dimensional grids. *Theoretical Computer Science*, 412(15):1350–1363, 2011. DOI: 10.1016/j.tcs.2010.10.027.

[25] Tagliasacchi, A., Delame, T., Spagnuolo, M., Amenta, N., and Telea, A. 3D Skeletons: A State-of-the-Art Report. *Computer Graphics Forum*, 35(2):573–597, 2016. DOI: 10.1111/cgf.12865.