Fast Skeletonization of Spatially Encoded Objects

Francisco Romero, Lluis Ros and Federico Thomas
Institut de Robòtica i Informàtica Industrial (UPC-CSIC)
Gran Capità 2-4, 08034 Barcelona, Catalonia, Spain
E-mail: {fromero, llros, fthomas}@iri.upc.es

Abstract

Some thinning algorithms for 3D objects, or generalizations of existing ones for 2D, have been proposed in recent years. The one given herein is surprisingly simple and very fast compared to most of them, and still it has theoretically favorable properties. Actually, it provides a connected surface skeleton that allows shapes to be reconstructed with bounded error. In addition, it is also very attractive because it allows discrete skeletons to be obtained directly from volumes in many representations without converting them to a voxel-based representation.

Our algorithm is a generalization of the one presented in [2] for 2D objects. It is based on the application of directional erosions, while retaining those voxels that introduce disconnections.

1. Introduction

Three-dimensional skeletons are a promising tool for an increasing number of applications in biomedical imagery [4, 8], and in general to many other applications related to shape matching and tracking, navigation, shape abstraction and animation control [3].

The word *skeleton* is usually understood in 2D to mean the medial axis of a given shape. The *medial surface* of a 3D object is defined similarly to its 2D counterpart: it is the set of the centers of all inscribed spheres of maximal radius. The computation of the medial surface for arbitrary objects is a complex problem. So far, it has only been efficiently solved for polyhedra [7]. Nevertheless, good approximations can be obtained by using the so-called *semicontinuous methods*. They proceed by distributing a set of points over the faces of the object for which we want its skeleton. Then, the Delaunay triangulation is applied to these points to obtain a set of tetrahedra whose centers are used to approximate the medial surface [6].

When working in a discrete space, spheres are always approximations of their continuous counterparts and the concept of skeleton should be redefined. Then, 3D discrete skeletons are approximations of this medial surface. In any case, the common required properties for any of these approximations are:

1. Reconstructability. A skeleton must contain sufficient information which can be used to reconstruct the original shape.
2. Rotation-invariance. Due to robustness reasons, a skeleton for a given volume must be independent from its orientation.
3. Connectedness. A skeleton must be homotopic to the shape it corresponds to. In other words, it must preserve 26-neighbor connectedness for the foreground and 6-neighbor connectedness for the background.
4. Thinness. A 3D object is thin if it can be described as a set of possibly intersecting patches of discrete surfaces.

Unfortunately, in the discrete space these requirements become mutually incompatible [2], and, hence, practical skeletonization methods are invariably a compromise between them.

Although other alternatives are possible [8], two main methods to obtain skeletons in discrete spaces have been proposed: thinning methods and methods based on distance transforms.

Thinning methods peel off the boundary of a volume. They produce skeletons by iteratively deleting voxels from the boundary of the object. The deletion of a voxel can be in a sequential algorithm or parallel one. Each iteration of sequential algorithms consists, in general, of three steps: (1) identify all border voxels, and label them with the iteration number; (2) inspect all voxels labeled with the current iteration number, and mark those that cannot be removed in order to preserve the shape of the original object (connectedness); and (3) remove all unmarked border voxels. It is obvious that the obtained skeleton using sequential versions of this algorithm depends partly on the order the voxels are processed. Up to our knowledge, the latest algorithm for thinning volumes appeared in [5].

The methods based on distance transforms first convert the volume, which consists of object (foreground) and non-object (background) voxels, into an object where every object voxel has the value corresponding to the minimum distance to the background. Different types of metrics for discrete objects are used, aiming to approximate the euclidean distance so that rotation-invariance is attained to some extend. Then,

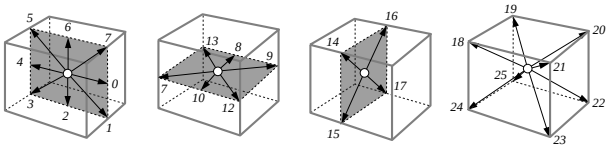


Figure 1. Defined directions.

the ridges of the induced scalar field constitute the skeleton. In general, these algorithms are not iterative so that the skeleton is produced in a fixed number of passes through the object. Up to our knowledge, the most recent skeletonization algorithm based on a distance transform appeared in [4].

The algorithm we propose here is an extension of the one presented in [2]. It can be classified as a thinning algorithm. The approach adopted satisfies the requirements given above in the following order of priority: connectedness, thinness and reconstructability. The preservation of the connectivity is the essential condition for the skeleton in order to extract the shape of the original object. Then, shapes can be nearly reconstructed with an error, in our case, bounded to one voxel.

The proposed procedure can be outlined as follows. Those voxels whose deletion by a directional erosion might destroy the connectedness are retained and classified as gaps, then the region is eroded and the corresponding residuals computed. Gaps and residuals are retained in the volume, and this process is repeated until no progress is made.

As explained above, the obtained skeleton is an approximation to the medial surface. In our case, the skeleton is the locus of maximal cubes. The algorithm is sequential so that the result depends partly on the order in which the directional erosions are performed. It is first developed for spatial enumeration (voxel-based) representations and then generalized to spatially encoded squemes and arbitrary representations, provided they allow boolean and displacement operations to be performed efficiently.

This paper is structured as follows. In order to make it as self-contained as possible, the required morphological operations are reviewed in Section 2. Next, the concepts of residuals and gaps associated with directional erosions are introduced. These are two key points for our skeletonization algorithm, which is presented in Section 3. It is also shown how a simple spatial encoding technique speeds up the performance of the algorithm. We conclude in Section 4.

2. Background

Let Z^3 be the discrete space. Let $X \subset Z^3$ a 3D volume. Let $\bar{X} = Z^3 \backslash X$ denote the background of X. The connectivity used herein is (26,6)-connectivity, which means 26-connectivity for the volume and 6-connectivity for the background. Each of the 26 neighbors of a voxel in the volume defines a direction which will be numbered as shown in figure 1.

The *erosion* of X using the structuring element B is defined as $X \ominus B = \{y | \forall b \in B, y + b \in X\}$, and its *dilation* using the same structuring element as $X \oplus B = \{y | y = x + b, x \in X, b \in B\}$, and its *opening* as $XoB = ((X \ominus B) \oplus B)$.

The *residual*, $X \perp B$, is the set made of those points in X which do not belong to its opening using the structuring element B, that is, $X \perp B = X \backslash (XoB)$.

If X_b denotes the translation of X in the direction associated with $b \in B$, then it can be shown that

$$X \ominus B = \bigcap_{b \in B} X_{-b}.$$

In other words, erosion can be accomplished by taking the intersection of all the translates of X, where the shifts in the translates are the negated members of B seen as vectors.

An especially interesting case for B is that in which B consists of two voxels, where one is the origin. Then, the erosion of X using B can be computed simply by $X \ominus B = X \cap X_{-b}$, and its opening by

$$XoB = (X \cap X_{-b}) \cup (X \cap X_{-b})_b.$$

Since $(B1 \ominus B2) \ominus B3 = B1 \ominus (B2 \oplus B3)$, then, if $B = B1 \oplus B2 \oplus \ldots \oplus Bk$, one concludes that

$$X \ominus B = (\ldots[(X \ominus B1) \ominus B2] \ominus \ldots \ominus Bk).$$

Thus, if a structuring element can be broken down to a chain of dilations of smaller substructuring elements, the desired operation may be performed as a sequence of suboperations.

As a first approximation, a skeleton can be defined as the set of all the residuals of the successive erosions of X, using the following simple algorithm:

algorithm T1;
input: X;
output: S;
$S \leftarrow \emptyset$;
while $X \neq \emptyset$
  $E \leftarrow X \ominus B$;
  $S \leftarrow S \cup (X \backslash (E \oplus B))$;
  $X \leftarrow E$;
endwhile;
end.

Now, let us assume that B is a centered 3x3x3 cubic structuring element which can be broken down into a chain of 6 dilations of two-voxel elements in the directions 0, 2, 4, 6, 8, and 10. Then, the above algorithm can be rewritten as follows:

algorithm T2;
input: X;
output: S;
$S \leftarrow \emptyset$;
while $X \neq \emptyset$
  $E \leftarrow X \cap X_0 \cap X_2 \cap X_4 \cap X_6 \cap X_8 \cap X_{10}$;
  $S \leftarrow S \cup (X \backslash (E \cup E_4 \cup E_2 \cup E_8 \cup E_0 \cup E_6 \cup E_{10}))$;
  $X \leftarrow E$;
endwhile;
end,

The main advantage of this algorithm over the previous one is that it only involves directional erosions and dilations along the coordinate axes. Although the skeleton thus obtained allows us to entirely reconstruct the initial set by simply dilating each voxel according to its distance to the boundary of the object, it is neither thin nor connectivity preserving. The first drawback can be easily overcame as follows:

algorithm T3;
input: X;
output: S;
$S \leftarrow \emptyset$;
$A \leftarrow \emptyset$;
while $X \neq \emptyset$
  /* directional erosion along y+ */
  $E \leftarrow X \cap X_0$;
  $A \leftarrow A \cup (X \backslash (E \cup E_4))$;
  $X \leftarrow E$;
  /* repeat for directions z+, x+, y-, z-, and x- */
  ⋮
  $S \leftarrow S + A$;
endwhile;
end.

Now, since residuals are independently thin (they are obtained from single directional erosions), the obtained skeleton is thin. As a counterpart, the original shape can only be nearly reconstructed but, as it has already been pointed out, thinness and reconstructability are mutually incompatible goals. Then, shapes can be nearly reconstructed with an error, in our case, bounded to one voxel.

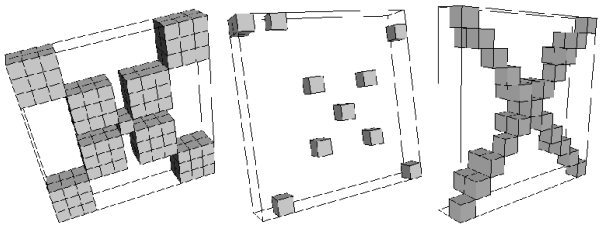
In order to overcome the remaining drawback – connectivity– we first introduce the concept of directional gaps. Those voxels required to ensure connectivity in the final skeleton and not included in the medial surface computed by algorithm T3, will be part of a set of disjoint regions that we call *gaps*. Contrary to what one might expect, when considering only directional erosions, gaps can be easily computed. For example, the directional gap of a binary region X in direction 0 (the coordinate axis y in the positive direction) can be obtained by computing:

$$X \cap \bar{X}_0 \cap ((X_7 \cap \bar{X}_6) \cup (X_1 \cap \bar{X}_2) \cup (X_{20} \cap \bar{X}_{16}) \cup (X_{22} \cap \bar{X}_{17}) \cup$$
$$(X_{21} \cap \bar{X}_{14}) \cup (X_{23} \cap \bar{X}_{15}) \cup (X_{12} \cap \bar{X}_{10}) \cup (X_9 \cap \bar{X}_8)).$$

Gaps along the other coordinate axes, either in positive or negative directions, can analogously be obtained. The above expression is obtained as a generalization of the two-dimensional case. It is worth noting that the concept of gaps, first introduced in [2], is closely related to the set of β templates recently presented in [5].

3. The thinning algorithm

The motivation behind our thinning algorithm is seen as follows. First those voxels whose deletion by a directional erosion might destroy the connectedness are retained and classified as gaps, then the region is effectively eroded and the corresponding residual computed. Gaps and residuals are removed from the object in order to concentrate the thinning effort on the thick region. Iterations continue until an object becomes empty. The following algorithm in pseudocode implements this procedure.



(a) **(b)** **(c)**

Figure 2. Result of the application of T3 (b), and T4 (c) on the object in (a)

The result always depends on the starting direction but, in any case, the error of the reconstruction process is bounded to one voxel.

algorithm T4
input: X;
output: S; /* skeleton of X*/
$S \leftarrow \emptyset$;
$L \leftarrow \emptyset$;
do
  /* erosion along y+ */
  $I \leftarrow \emptyset$; /* increment of skeleton */
  $G \leftarrow X \cap \bar{X}_0 \cap \left[\left(X_7 \cap \bar{X}_6\right) \cup \left(X_1 \cap \bar{X}_2\right) \cup \ldots \left(X_9 \cap \bar{X}_8\right)\right]$;
  /* gap */
  $E \leftarrow X \cap X_0$; /* eroded image */
  $R \leftarrow X \backslash (E \cup E_4)$; /* residual */
  $I \leftarrow I \cup R \cup G$;
  $X \leftarrow E \cup I$;
  /*repeat for directions z+, x+, y-, z-, and x- */
  ⋮
  $X \leftarrow X \cap \bar{L}$;
  $S \leftarrow S \cup L$;
  $L \leftarrow I \cap \bar{L}$;
until $(X == \emptyset)$;
end.

Figure 2 shows an object and the obtained skeletons using algorithms T3 and T4. The thinning action of T3 is greedy: clear topological changes occur. The introduction of gaps by T4 fixes the problem.

The above algorithms have been implemented in C on a Sun Ultra2-2200 workstation. It has been recently criticized that in many papers on skeletonization of volume objects the only given examples are tiny test images, which makes it difficult to understand what would be the results for reasonable sized real images [1]. In our case, T4 has been tested on 3-D images obtained by a TAC device. Figure 3 shows the results for a human vertebra and a partial skull. The vertebra contains 22,263 voxels, its skeleton is computed in 2.644 seconds and contains 5,606 voxels. The skull contains 20,001 voxels, it is computed in 3.095 seconds and has 8,165 voxels. The skeleton of the skull is computed in six iterations. Table I shows the evolution of the processing time for each iteration. Note the exponential reduction in the time required for each iteration thanks to the adopted incremental strategy.

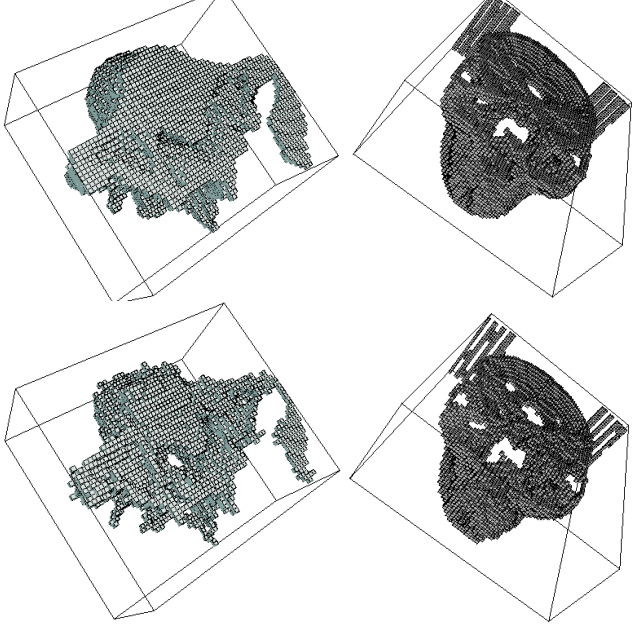


Figure 3. Skeletons (lower row) obtained using the proposed algorithm on real data (upper row).

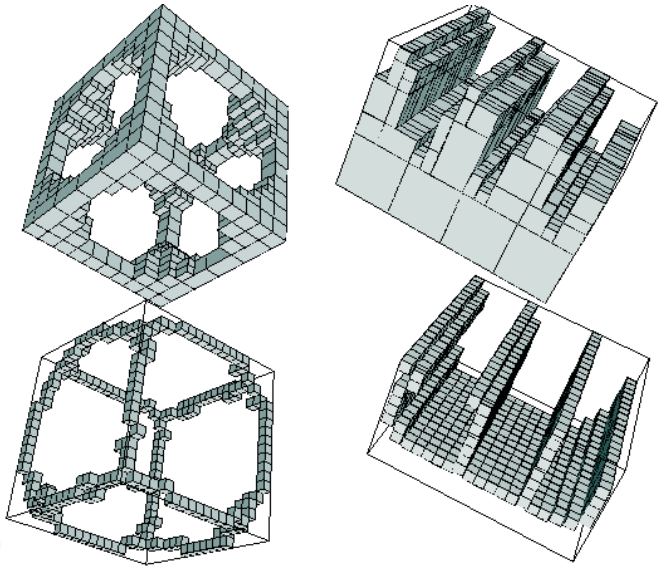


Figure 4. Skeletons (lower row) obtained using the proposed algorithm on encoded synthetic volumes (upper row).

Table I

Iter.	CPU time
1	1874
2	954
3	180
4	63
5	22
6	2
Total	3095

Table II

Iter.	CPU time (encoded)	CPU time (unencoded)
1	30	134
2	24	84
3	14	35
4	2	5
Total	70	258

A relevant feature of algorithm T4 is that, contrary to [5], it can be applied to other object representations different from explicit spatial enumeration. In particular, we have applied it to volumes represented by binary subdivision trees. Figure 4 shows two such encoded objects and the obtained skeletons. In order to show the effect of this codification in the processing time, Table II compares its evolution for the second encoded volume against its unencoded counterpart. While the encoded object contains 435 boxes, the unencoded version has 3.920 voxels. Thus, the benefits of avoiding to work at voxel level are clear both in memory requirements and computational time.

4. Conclusions

In this paper we have presented an algorithm based on concise boolean expressions able to compute the skeleton of 3-D objects in different representation schemes. In particular, it has been shown how the efficiency of the algorithm can be greatly improved and its memory requirements dramatically reduced when dealing with objects represented with binary subdivision trees. The enormous algorithmic difficulties caused when working at voxel level due to the identification of all removable voxels using large look-up tables –as standand thinning algorithms usualy do– is thus avoided.

References

[1] G. Borgefors, I. Nystrom, and G. Sanniti Di Baja. Computing skeletons in three dimensions. *Pattern Recognition*, 32:1225–1236, 1999.

[2] R. Cardoner and F. Thomas. Residuals + directional gaps = skeletons. *Pattern Recognition Letters*, 18:343–353, 1997.

[3] N. Gagvani. 3d skeletonization and volumen thinning. *See http://www.caip.rutgers.edu/˜gagvani/skel.html*.

[4] G. Malandain and S. Fernández-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16:317–327, 1998.

[5] A. Manzanera, T. Bernard, F. Prêteux, and B. Longuet. Medial faces from a concise 3d thinning algorithm. In *7th IEEE Conf. on Computer Vision*, Vol. 1, 1999.

[6] D. Sheehy, C. Armstrong, and D. Robinson. Computing the medial surface of a solid from a domain Delaunay triangulation. In *ACM/IEEE Symposium on Solid Modeling and Applications*, May 1995.

[7] E. Sherbrooke, N. Patrikalakis, and E. Brinson. An algorithm for the medial axis transform of 3-d objects. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):44–61, 1996.

[8] Y. Zhou and A. Toga. Efficient skeletonization of volumetric objects. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):196–209, 1999.