Improved skeleton computation of an encoded volume

D. Ayala[1] and E. Vergés[1] and E. Vergara[1]

[1]IBEC (Bioengineering Institute of Catalonia). UPC (Polytechnical University of Catalonia).Barcelona, Spain

Abstract

In this communication, we present an improvement of an existing thinning algorithm that computes a surface skeleton from a binary volume. The method makes an intensive use of Boolean operations and works on a special encoding of the volume (the EVM encoding) in which Boolean operations are very efficient. The contribution of this work consists on a more suitable application of Boolean operations in the thinning algorithm. Computation time has been reduced more than a half.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer graphics]: Computational Geometry and Object Modeling; I.4.10 [Image processing and computer vision]: Image representations

1. Introduction

In the field of Bio-CAD technology structural parameters have to be computed for the morphology of porous bone and bio-material samples. Such parameters are classifications into rod (low density) and plate (high density) elements and porosity models, including pore size and interconnectivity. Obtaining the surface skeleton from a binary volume is a preprocess needed in these computations. The skeleton can be defined as the locus of centers of maximal inscribed balls [Lie03].

In this paper we present an improvement of an existing algorithm that allows to compute a surface skeleton from a binary volume and makes an intensive use of Boolean operations. The volume is encoded in a way, EVM encoding, which allows efficient Boolean operations. The contribution of this work is an improvement in the application of these Boolean operations.

2. Related work

Skeletonisation methods can extract a surface skeleton as well as a curve skeleton. Methods can also be classified in four families: thinning, geometric and methods based on distance transform and on generalized potential fields [CSM05]. We use a thinning method that computes a surface skeleton [CT97], [RTAR03]. Thinning methods produce skeletons by iteratively deleting voxels from the boundary of the volume. In [MBPL02] a thinning method is presented to obtain a surface skeleton that is used in [SM06] to classify bone samples into rods and plates. Another thinning method, [BNB99] computes the medial surface and, from it, the curve skeleton. In [BB04] a curve skeleton is computed by a thinning method in order to segment a triangular mesh. It uses a run-length encoding to obtain the skeleton.

An object and its skeleton are topologically equivalent if they have the same number of connected components, tunnels and cavities [KR89]. Topology preservation can be guaranteed by thinning methods by only allowing deletion of simple points. A simple point is defined in terms of its neighborhood [KR89]. Topology preservation can also be achieved by detecting the two patterns showing non-manifold vertices or non-manifold edges of a volume (see Figure 2) [BNB99], [MBPL02], [RTAR03].

3. Background

3.1. The thinning algorithm

The thinning algorithm used in this work [CT97], [RTAR03] satisfies the following properties in this order: homotopy, thinness and reconstructability [RTAR03]. It applies unitary directional erosions to the region in order to compute the corresponding *residuals* and to detect those voxels whose deletion might not preserve the topology with the two non-manifold patterns (*gaps*) already mentioned. Residuals and gaps are retained in the volume since they are part of the

skeleton and this process is repeated until no more progress is made.

Let V be a volume. The computation of residuals is based on the expression $V \perp B = V \backslash (VoB)$. B is the structuring element, a centered 3x3x3 cubic element which can be broken into a chain of 6 two-voxel elements that will allow to perform directional erosions and o is the opening operation, $VoB = ((V \ominus B) \oplus B)$.

For each directional erosion, gaps are computed with a Boolean expression. For instance, for direction y+:

$$(V \backslash V_0) \cap ((V_7 \backslash V_6) \cup (V_1 \backslash V_2) \cup (V_{12} \backslash V_{10}) \cup (V_9 \backslash V_8)) \cup (V_{21} \backslash V_{14}) \cup (V_{23} \backslash V_{15}) \cup (V_{20} \backslash V_{16}) \cup (V_{22} \backslash V_{17}) \quad (1)$$

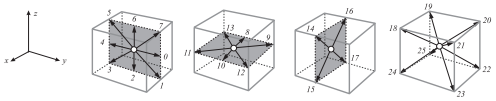

Figure 1: *Numbering corresponding to the possible 26 translations applied.*

V_i stands for the volume V translated in the i direction (see Figure 1). Here, we outline the used algorithm:

algorithm Thinning
input: V; /* binary volume*/
output: S; /* skeleton of V*/
$S \leftarrow \emptyset$; $L \leftarrow \emptyset$;
do
  /* erosion along y+ */
  $I \leftarrow \emptyset$; /* increment of skeleton */
  $E \leftarrow V \cap V_0$; /* eroded volume */
  $R \leftarrow V \backslash (E \cup E_4)$; /* residual */
  $G \leftarrow (V \backslash V_0) \cap [(V_7 \backslash V_6) \cup (V_1 \backslash V_2) \cup \ldots \cup (V_{22} \backslash V_{17})]$; /* gap */
  $I \leftarrow I \cup R \cup G$; /* increment of skeleton */
  $V \leftarrow E \cup I$; /* updated volume */
  /* repeat for directions z+, x+, y-, z-, and x- */
  ...
  $V \leftarrow V \backslash L$; /*removal of previous skeleton increment */
  $S \leftarrow S \cup L$; /* updated skeleton */
  $L \leftarrow I \backslash L$; /* last skeleton increment */
until ($V = \emptyset$);
$S \leftarrow S \cup L$;/* last updated skeleton */
end.

For more details, see [CT97], [RTAR03]. This algorithm was first performed in a voxel-based representation [CT97] and then using the EVM encoding [RTAR03].

3.2. Extreme Vertices Encoding

The boundary of any volume can be represented by an orthogonal pseudo-polyhedron (OPP), i.e., a polyhedron with all its faces oriented according to the three orthogonal coordinate axes and with a non-manifold boundary (see Figure 2). Let P be an OPP, a *brink* is a maximal uninterrupted segment built out of a sequence of collinear edges of P. The

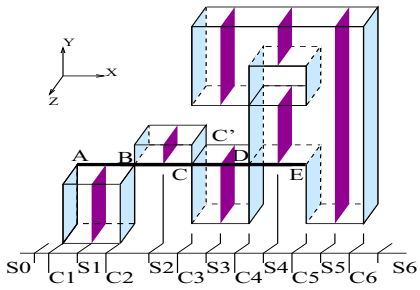

Figure 2: *Example of an OPP.* B *is a non-manifold vertex and* CC' *is a non-manifold edge. A brink from vertex* A *to vertex* E *is shown (vertices* A *and* E *are extreme and vertices* B, C *and* D *are non-extreme). Cuts and sections perpendicular to the X axis are shown in light and dark color, respectively.*

ending vertices of a brink are called *extreme vertices*. P can be encoded by its extreme vertices which are a subset of its vertices (*EVM encoding*). A *cut* of P is the set of extreme vertices of P lying on a plane perpendicular to a main axis of P, say X. A slice is the region between two consecutive planes of vertices and a *section* is the polygon resulting from the intersection between a slice and a plane perpendicular to X. Cuts and sections are pseudo-polygons and their 1D cuts and sections can be computed analogously. Sections can be computed from cuts and vice versa:

$$S_i(P) = S_{i-1}(P) \otimes C_i(P);\ C_i(P) = S_{i-1}(P) \otimes S_i(P) \quad (2)$$

for $i = 1 \ldots n$, where $\otimes$ denotes the XOR Boolean operation.

EVM encoding is efficient for operations performed on volumes [RAA04]. For more details, see [Agu98].

4. Improved Boolean operations

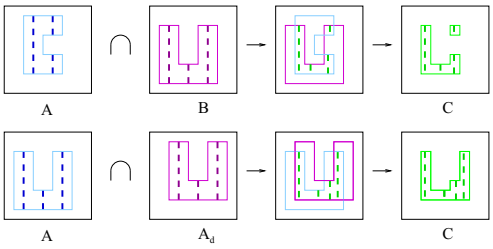

Figure 3: *Above: 2D example of the Boolean operations algorithm between two different object. Below: 2D example of the Boolean operations algorithm between an object and a translation of itself. NOTE: vertical dashed lines represent the sections of the objects*

Given two volumes EVM encoded using the same sequence of coordinate axes, say XYZ, any Boolean operation

(B.O.) between them can be reduced by applying the same operation to the sections perpendicular to the X axis of both operands, which are 2D objects. In turn, this boils down to apply the same operation to the 1D sections perpendicular to the Y axis of both 2D operands. Then, with the EVM encoding, 3D B.O. are reduced to 1D B.O. and consist of a merging process involving sections. Computing sections is the bottleneck of this process, but this computation only involves XOR operations which hold the following property. Let $EVM(V)$ and $EVM(W)$ be the two encoded volumes, then [Agu98]:$EVM(V \otimes W) = EVM(V) \otimes EVM(W)$. Figure 3, above, illustrates the B.O. algorithm with a simple 2D example, in which sections of A and B are computed and used to compute sections of $C = A \cap B$.

The used thinning algorithm involves the computation of B.O. between the input volume, V, and translates of itself, V_d, $d = 0, \ldots, 25$. The direct implementation of these operations in the thinning algorithm and particularly in equation 1 would require computing the EVM encoding of all the translated operands (i.e., their cuts and sections) and store or recompute sections of the partial results. These memory and computational requirements can be reduced by taking into account the following two properties:

1. Performing B.O. between two volumes rely on their sections and the sections of a translated volume are the same as those of the input volume, conveniently translated, that is, $S_i(V_d) = (S_i(V))_d$, for any i.
2. Computing $V \diamondsuit V_d$, where $\diamondsuit$ stands for an arbitrary B.O., involves operating any section of V with the same, the previous or the next section of V_d, because all translations are unitary, according to the following rules:
   - $S_i(V \diamondsuit V_d) = S_i(V) \diamondsuit S_i(V_d), d \in \{0,1,2,3,4,5,6,7\}$
   - $S_i(V \diamondsuit V_d) = S_i(V) \diamondsuit S_{i+1}(V_d)$, $d \in \{8,9,13,16,17,19,20,22,25\}$
   - $S_i(V \diamondsuit V_d) = S_i(V) \diamondsuit S_{i-1}(V_d)$, $d \in \{10,11,12,14,15,18,21,23,24\}$

These properties mean that computing B.O. between a volume V and a translate of itself V_d does not imply to first translate V and then compute sections of V and V_d. Instead, only sections of V have to be computed and then translated. Moreover, computing sections of a resulting volume only involves two consecutive sections of the input volume and, therefore, they can be computed sequentially without relying on additional data structures. Figure 3, below, shows a 2D example involving an object A and a translate, A_d. Sections of A_d are the same as sections of A conveniently translated and to compute $C = A \cap A_d$, only one or two consecutive sections of A are involved.

These properties are applied in the computation of the eroded volume, $V \cap V_0$, and in the computation of the second part (union) of the residual, $E \cup E_4$. For directions y+, y-, z+ and z- (d = 0, 4, 6, 2, respectively) these properties can be recursively applied to the 2D sections (as in these cases we always operate a 2D section and a translated version of itself). But for directions x+ and x- (d = 8, 10, respectively) it is not possible, in general, as in several cases we are operating different sections ($S_i(V)$ with $S_{i-1}(V)$). We could solve this problem by reordering the extreme vertices but this would be more expensive than to perform the general B.O.

The application of these properties is even more efficient in the computation of gaps (equation 1). We apply them to compute $G = V \backslash V_0$ and to compute $D = V_0 \backslash V$, as explained before. D is computed because the remaining differences of equation 1 are unitary translates of D. For instance, $V_7 \backslash V_6 = (V_0 \backslash V)_6 = D_6$. So, equation 1 can be rewritten as: $G \cap U, U = D_6 \cup D_2 \cup D_{10} \cup D_8 \cup D_{14} \cup D_{15} \cup D_{16} \cup D_{17}$. Furthermore, we can compute all the unions of U simultaneously and at the same time as the computation of D as well as we can compute $V \cap V_0$, $G = V \backslash V_0$ and $D = V_0 \backslash V$ also at the same time, as they involve the same object and translation.

Another way to compute equation 1 would be in the union of intersections form (UOI), i.e., $(G \cap D_6) \cup (G \cap D_2) \cup (G \cap D_{10}) \cup (G \cap D_8) \cup (G \cap D_{14}) \cup (G \cap D_{15}) \cup (G \cap D_{16}) \cup (G \cap D_{17})$ and then getting smaller (eventually null) operands. But this way would involve an intensive computation of sections.

Now we are going to analyze the time complexity of this improvement with the direct application of equation 1, taking as basic operations the translation and the computation of sections of a volume. They are basic operations because they are performed on the same initial volume. The direct application of equation 1 involves 17 translations and 17 B.O., each involving computing sections of 2 operands: 34 section computations. On the other hand, the improved version involves only 1 section computation for the initial volume V, and 9 translations, one for V_0 and 8 to compute U. What follows is the justification of why we only need to compute the sections of one volume. Each computed section of V is translated to obtain the corresponding section of V_0 and from them the corresponding sections of volumes $V \backslash V_0$ and $D = V_0 \backslash V$ can be computed; then the actual computed section of D is translated 8 times and the resulting sections are operated; finally the resulting section is operated with the corresponding section of $V \backslash V_0$. This explanation applies directly for cases $d \in \{0,1,2,3,4,5,6,7\}$. For the other cases more than one consecutive section is involved, but this fact only affects memory requirements. So, from this analysis, we can conclude that the new version is a real improvement.

5. Experimental results

The presented algorithm has been implemented in C on a PC Core2, 2.4 GHz with 3 Gbytes of RAM. Table 1 shows the running times in seconds for several datasets before and after the improvement presented in this work. Figure 4 shows two datasets with their skeleton. The EVM of all the models in Table 1 can be found in http://truja.lsi.upc.edu/movibio/ResearchLines/VEVM/.

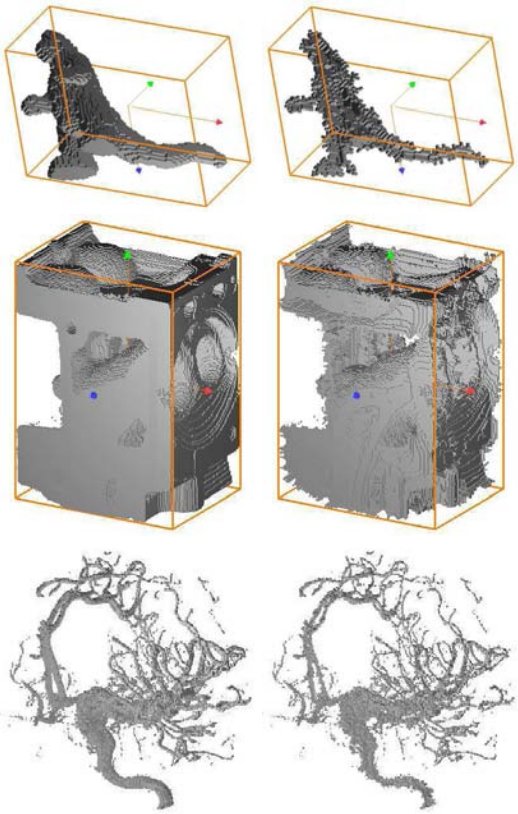

Figure 4: *Datasets* monster *(above)* engine *(middle) and* aneurysm *(below) on the left, with their corresponding skeletons on the right*

Model	Size Voxel	Size EVM	Time before	Time after
monster	$87 \times 39 \times 62$	4288	26.5	13.8
skull	256^3	23464	89	49.5
aneurysm	256^3	70260	450	165
lobster	$301 \times 324 \times 56$	102394	591	269
engine	$256^2 \times 128$	106390	932	516

Table 1: *Results for several known datasets. Values shown are model size (SizeVoxel), number of extreme vertices (SizeEVM) and running times in seconds (Time).*

From the results, we can see that the computation time has been reduced by approximately a half.

6. Conclusions and future work

We have presented an improvement of a thinning algorithm by using a suitable encoding of the volume that allows to perform efficient Boolean operations. The improvement consists in the application of suitable Boolean operations, based on the fact that these operations are performed between volumes and translates of themselves.

As a future work, we plan to apply the obtained skeleton to characterize bones and scaffolds into rods and plates and to model the porosity.

7. Acknowledgments

This work has been partially supported by the project MAT2005-07244-C03-03 from the Spanish government and by the IBEC (Bioengineering Institute of Catalonia).

References

[Agu98] AGUILERA A.: *Orthogonal Polyhedra: Study and Application*. PhD thesis, LSI-Universitat Politècnica de Catalunya, 1998.

[BB04] BRUNNER D., BRUNNETT G.: Mesh segmentation using the object skeleton graph. In *Computer Graphics and Imaging- 2004* (2004), pp. 48 – 55.

[BNB99] BORGEFORS G., NYSTROM I., BAJA G. S. D.: Computing skeletons in three dimensions. *Pattern Recognition 32* (1999), 1225–1236.

[CSM05] CORNEA N., SILVER D., MIN P.: Curve-skeleton applications. In *Proceedings of IEEE Visualization 2005* (2005), pp. 23–28.

[CT97] CARDONER R., THOMAS F.: Residuals + directional gaps = skeletons. *Pattern Recognition Letters 18* (1997), 343–353.

[KR89] KONG T., ROSENFELD A.: Digital topology: Introduction and survey. *Computer Vision, Graphics and Image Processing 48* (1989), 357 – 393.

[Lie03] LIEUTIER A.: Any open bounds subset of R^n has the same homotopy type than its medial axis. In *ACM Shape Modeling and Applications* (2003), pp. 65 – 75.

[MBPL02] MANZANERA A., BERNARD T., PRÊTEUX T., LONGUET B.: nD skeletonization: a unified mathematical framework. *Journal of Electronic Imaging 11*, 1 (2002), 25 – 37.

[RAA04] RODRÍGUEZ J., AYALA D., AGUILERA A.: *Geometric Modeling for Scientific Visualization*. Springer, 2004, ch. EVM: A Complete Solid Model for Surface Rendering, pp. 259 – 274. ISBN: 3-540-40116-4.

[RTAR03] RODRÍGUEZ J., THOMAS F., AYALA D., ROS L.: Efficient computation of 3D skeletons by extreme vertex encoding. In *Discrete Geometry for Computer Imagery. 11th Int. Conf.* (2003), Springer, pp. 338 – 347.

[SM06] STAUBER M., MÃIJLLER R.: Volumetric spatial decomposition of trabecular bone into rods and plates: A new method for local bone morphometry. *Bone 38*, 4 (2006), 475 – 484.