Determination of a Vessel Tree Topology by Different Skeletonizing Algorithms

Andre Siegfried Prochiner[1], Heinrich Martin Overhoff[2]

[1]Carinthia University of Applied Sciences, Klagenfurt, Austria
[2]University of Applied Sciences Gelsenkirchen, Gelsenkirchen, Germany
heinrich-martin.overhoff@fh-gelsenkirchen.de

Abstract. For navigated interventions of parenchymatous organs, plans must be adjusted according to organ motion and deformation. Vessel trees and especially their furcations are assumed to serve as landmarks for continuous registration. Three heuristic skeletonization algorithms were investigated to determine their ability to detect furcations correctly. A body-centered cubig grid-based algorithm was rated to fulfill these specifications, and an efficient implementation was developed.

1 Introduction

In navigated surgery and radiotherapy, interventions are often planned with respect to anatomical landmarks. Whereas distances between bony landmarks stay constant, landmarks in parenchymatous organs change their relative location e.g. due to altered patient positions or during resections. A vessel tree can serve as landmark system for such organs, because at least its topology remains – aside from resected parts – unaltered even during organ deformation, and vessel furcations can securely be detected.

Skeletonization is a common preprocessing step during the raster to vector transformation of a volume data set and consists of the reduction of objects in the 3D image volumes to a data reduced but topology conserving representation. For simplicity, a skeleton can be regarded as a medial axis of an object [1, 2]. More formally, it is the set of the centers of those spheres, which fully cover the object's volume and do not fully cover each other [3, 4].

Several algorithms, defined on a regular cartesian or on a body centered cubic grid respectively, were implemented in Matlab and investigated for their ability to generate correct skeletons of synthetic 3D objects. The most reliable one was chosen to skeletonize a vessel tree, that was constructed from a segmented 3D ultrasound image volume. Furthermore, the execution times of the programs were measured to detect hints for an efficient implementation.

2 Materials and Methods

In the following, a skeletonization is performed by removing voxels $\mathbf{p} \in \mathcal{O}$ from the object voxel set $\mathcal{O}$ while preserving the object's topology.

A 3D object $\mathcal{O}^c \subset \mathbb{Z}^3$ with known skeleton $\mathcal{S}^c \subset \mathcal{O}^c$ was defined on a regular cartesian grid. This object was transformed into object $\mathcal{O}$ and skeleton $\mathcal{S} \subset \mathcal{O}$ defined on a body-centered cubic (BCC) grid). Skeletonization algorithms were implemented for both grids:

A the algorithm reported in [1] with modifications [5] for objects on a regular Cartesian grid,
B the algorithm reported in [4] for objects on a BCC grid and
C Algorithm B applied after distance transform.

Each algorithm generated a skeleton $\mathcal{T}_i$, $i = 1, 2, 3$. A visual evaluation of the coincidence generated vs. correct skeleton ($\mathcal{T}_1$ vs. $\mathcal{S}^c$, $\mathcal{T}_2$ and $\mathcal{T}_3$ vs. $\mathcal{S}$) was performed. In a second step, the algorithm with the best skeleton reconstruction was applied to generate the skeleton of the hepatic vein vessel tree.

2.1 Distance Transform

A distance transform determines the shortest distance d_i between a voxel $\mathbf{p}_i \in \mathcal{O}$ and the surface of this object. Voxels $\mathbf{p}_i$ with identical distances $d_i = D_\kappa$ construct a voxel set $\mathcal{S}_\kappa$ for $0 \leq D_\kappa \leq D_{\max}, 0 \leq \kappa \leq \kappa_{\max}$. $\mathcal{O}_\kappa = \bigcup_{1 \leq j \leq \kappa} \mathcal{S}_j$ denotes the object's surface with thickness D_κ. Without the distance transform, all voxels $\mathbf{p}$ of the surface of object $\mathcal{O}$ are analyzed during skeletonization and can potentially be removed from the objects. When applying the transform, only the surface voxels $\mathbf{p}$ having the current maximum surface distance D_κ are analyzed. The voxel decimation begins with $\tilde{\mathcal{O}}_0 = \mathcal{S}_0$. Due to the voxel decimation, surface voxels are eliminated from $\tilde{\mathcal{O}}$ which is thus transformed into $\hat{\mathcal{O}}$. Iteratively, for $1 \leq \kappa \leq \kappa_{\max}$ the decimation is performed on voxel sets $\tilde{\mathcal{O}}_\kappa = \hat{\mathcal{O}}_{\kappa-1} \cup \mathcal{S}_\kappa$.

2.2 Cartesian Grid Algorithm

In [5], an algorithm is presented where object surface points are classified by 62 neighborhood templates. If a voxel $\mathbf{p}^c$ of the surface of object $\mathcal{O}^c$ coincides with one of the templates, it can be removed.

2.3 Body-centered Cubic Grid Algorithms

A BCC grid is a subset of a regular cartesian grid and is defined by $\mathcal{B} = \left\{\mathbf{p} \in \mathbb{Z}^3 \,\middle|\, p_x \equiv p_y \equiv p_z \,(\mod 2)\right\}$. Given an object $\mathcal{O} \subset \mathcal{B}$, the object's background is defined by $\bar{\mathcal{O}} = \{\mathbf{p} \,|\, \mathbf{p} \notin \mathcal{O} \wedge \mathbf{p} \in \mathcal{B}\} = \{\mathbf{p} \,|\, \mathbf{p} \in \mathcal{B} \setminus \mathcal{O}\}$.

On a BCC grid, the distance vector sets $\mathcal{D}_\alpha$ are composed of the vectors $\mathbf{v}_\alpha$ to the nearest neighbors of a point $\mathbf{p} \in \mathcal{B}$ and the neighborhoods $\mathcal{N}_\alpha(\mathbf{p})$ are given by

$$\begin{aligned} \mathcal{D}_6 &= \left\{\mathbf{v}_6 \in \mathbb{Z}^3 \,\middle|\, |v_x| + |v_y| + |v_z| = 2\right\} \\ \mathcal{D}_8 &= \left\{\mathbf{v}_8 \in \mathbb{Z}^3 \,\middle|\, |v_x| + |v_y| + |v_z| = 3\right\} \\ \mathcal{D}_{14} &= \mathcal{D}_6 \cup \mathcal{D}_8 \end{aligned} \tag{1}$$

$$\mathcal{N}_\alpha(\mathbf{p}) = \{\mathbf{q} \,|\, \mathbf{q} \in \mathcal{B} \wedge \mathbf{v}_\alpha = \mathbf{p} - \mathbf{q} \in \mathcal{D}_\alpha\}, \quad \alpha = 6, 8, 14. \tag{2}$$

A point set $\mathcal{P} \subset \mathcal{Q} \subset \mathcal{B}$ is a connected component of $\mathcal{Q}$, if for each pair of points $\mathbf{p}, \mathbf{q} \in \mathcal{P}$ a sequence of points $\mathbf{p}_1, ..., \mathbf{p}_m \in \mathcal{P}$ exists, such that $\mathbf{p}_1 = \mathbf{p}$, $\mathbf{p}_m = \mathbf{q}$ and $\mathbf{p}_i \in \mathcal{N}_{14}(p)$ holds for $1 < i < m$. $C(\mathcal{Q})$ is the number of connected components of $\mathcal{Q}$. An object point $\mathbf{p} \in \mathcal{O}$ is a simple point, if

$$C(\mathcal{N}_{14}(\mathbf{p}) \setminus \mathcal{O}) = 1 \wedge C\left(\mathcal{N}_{14}(\mathbf{p}) \setminus \bar{\mathcal{O}}\right) = 1. \tag{3}$$

Simple points can be removed from object $\mathcal{O}$ with conservation of the object's topology. In addition to pure skeletonization, in each step, a removed voxel can be assigned to its nearest remaining voxel. This assignment stops at skeleton voxels. Thus, the voxels each skeleton arm represents are known explicitly.

The classification of an object point $\mathbf{p}$ as simple is the most frequently processed step and should be done efficiently. Two alternative simple point qualifications were implemented for Algorithm B as well as for algorithm 3:

- like in [4], the connectivity of 24 so-called checking rings $\mathcal{R}_{ij}(\mathbf{p}) \subset \mathcal{N}_{14}(\mathbf{p})$, $1 \le i \le 3, 1 \le j \le 8, |\mathcal{R}_{ij}| = 6$ was determined during skeletonization.
- The simplicity of a point $\mathbf{q}$ was determined once for all of its $N = 2^{14}$ possible $\mathcal{N}_{14}(\mathbf{q})$-neighborhoods and the binary results were stored in a database before skeletonization. During skeletonization, the simplicity of the individual $\mathcal{N}_{14}(\mathbf{p})$-neighborhood was determined by data base access.

For large objects with irregular surfaces – like a highly resolved vessel tree volume – overskeletonization is a common problem. To avoid it, object surface's voxels $\mathbf{p}$ are classified as stubble points, if $\mathbf{p} \in \hat{\mathcal{O}}_{\kappa-1} \wedge q = \mathcal{N}_{14}(p) \cap \tilde{\mathcal{O}}_\kappa \wedge |\mathcal{N}_{14}(q) \cap \tilde{\mathcal{O}}_\kappa| > 2$. Stubble points typically generate only short skeleton arms and are removed from $\tilde{\mathcal{O}}_\kappa$ before the next simple point classification is processed.

3 Results

The skeletonization results are demonstrated in the following figures. Figure 1 shows a synthetic F-shaped object which consists of 313 voxels and its skeleton (opaque voxels) found by Algorithm C. Comparison with the ideal skeleton

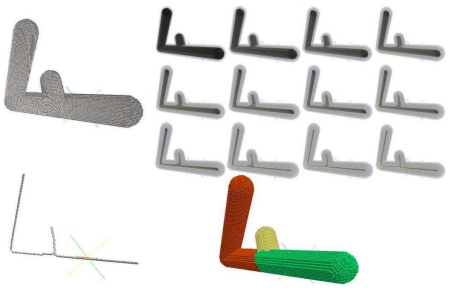


Fig. 1. Process of skeletonization and reconstruction of a synthetic object. (top left) Object, (top right) sequence of thinning steps with removed simple points (semi transparent) and remaining points (opaque), (bottom left) skeleton, (bottom right) voxels assigned to skeleton branches.

Table 1. Execution times for different realizations of simplicity testing (BCC algorithms).

	object		
simplicity test	single voxel	vessel tree 1	vessel tree 2
procedural	5.5 ms	1407 s	45178 s
data base	3.8 ms	1300 s	44874 s

(semi-transparent voxels) is performed by the euclidean distances between of the pairwise nearest voxels of ideal vs. real skeleton: 228 (80/0/3/2) voxels had distance 0 (1/1.41/1.73/2).

Algorithms A and B (Fig. 2) show results with additional furcations and reduced precision at the skeleton's branches. Obviously, the distance transform supports the detection of the correct skeleton.

Algorithm C was chosen to skeletonize two hepatic vein vessel trees. The trees were acquired in ultrasound image volumes and contained the inferior vena cava, the three branches of the hepatic vein and some of the segmental veins. Due to the limited image volume, the vessels are partially cut off. Anatomically, vessel trees 1 and 2 are identical, but tree 2 has the 8-fold resolution of tree 1. Visually, the skeletons reflect the topological properties well, especially for the segmental veins. In detail, for vessel tree 1, the left and right hepatic vein are marginally overskeletonized (Fig. 3). Each of the skeleton's arms belongs to one vein, i.e. none was erroneously assigned to two veins and no furcation was missing.

The Matlab code was processed on a PC equipped with an Intel Core2Duo E8400 3 GHz processor, 3 GB RAM, 1333 MHz main board FSB frequency. The execution times for the simplicity analysis are shown in Tab. 1 demonstrating that data base access is superior to procedural determination.

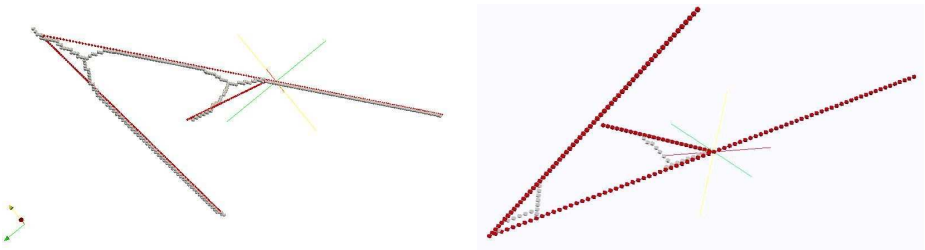
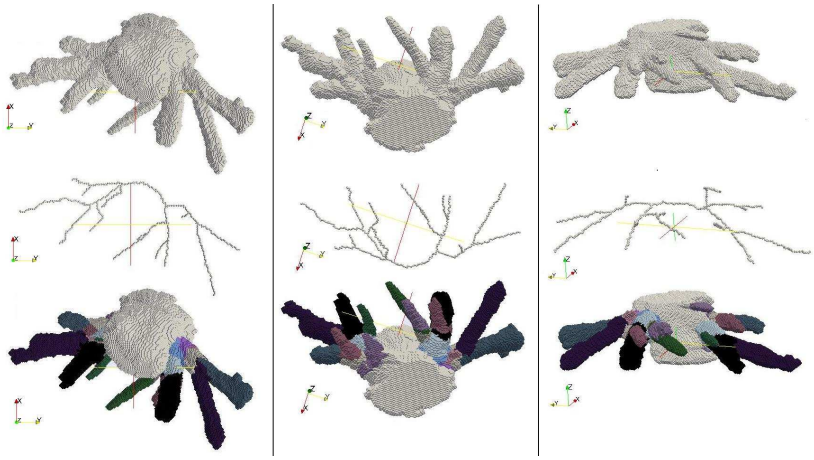


Fig. 2. Skeleton of the synthetic object (gray voxels) generated by (left) BCC grid based Algorithm B without distance transform and (right) cartesian grid based Algorithm A. Ideal skeleton is shown for comparison (red cubes).

Fig. 3. Hepatic vein vessel tree (top), its skeleton from Algorithm C (center), and its reconstruction (bottom). Distal (left column), proximal (center column) and right oblique (right column) view.



4 Discussion

The heuristics reported in [4] for objects defined on a BCC grid is rated as promising for the skeletonization of vessel trees. Simplicity analysis is faster using data base access than using explicit procedural analysis. Obviously, the skeletonization's execution times depend severely on the cardinality of $\mathcal{O}$.

Ongoing work is addressed to analyze the observed overskeletonization of vessels and to investigate the coincidence of vessel tree skeletons for data sets representing the same object, but but with reduced cardinality.

References

1. Ma M, Sonka M. A fully parallel 3d thinning algorithm and its applications. Comput Vis Image Underst. 1996;64:420–33.
2. Palagyi K, Kuba A. A 3D 6-subiteration thinning algorithm for extracting medial lines. Pattern Recognit Lett. 1998;19:613–27.
3. Blum H. A transformation for extracting new descriptors of shape. In: Wathen-Dunn W, editor. Models for the Perception of Speech and Visual Form. Cambridge: MIT Press; 1967. p. 362–80.
4. Brunner D, Brunnett G, Strand R. A high-performance parallel thinning approach using a non-cubic grid structure. Chemnitzer Informatik-Berichte. 2006; p. 1–13. Available from: http://www.tu-chemnitz.de/informatik/service/if-berichte/pdf/CSR-06-08.pdf.
5. Wang T, Basu A. An improved fully parallel 3D thinning algorithm. Univ. of Alberta, Dept. of Computer Science; 2005. http://www.cs.ualberta.ca/TechReports/2005/TR05-31/TR05-31.pdf, rev. 2009.09.05.