Technical Report CAIP-TR-216

PARAMETER CONTROLLED SKELETONIZATION OF THREE DIMENSIONAL OBJECTS

Nikhil Gagvani and Deborah Silver
Department of Electrical and Computer Engineering and CAIP center,
Rutgers University, Piscataway, NJ 08855

3 June 1997

COMPUTER AIDS FOR INDUSTRIAL PRODUCTIVITY
CoRE Building - Frelinghuysen Road
Rutgers - The State University of New Jersey
Piscataway, New Jersey 08855-1390
(908) 445-4208

Abstract

Skeletons are useful shape abstractions and have varied applications in visualization. The complexity of the desired skeletal structure depends on the application. Current techniques for extracting skeletons do not allow control over the complexity. In this paper, we describe an algorithm which uses a thinness parameter to control the density of the skeleton. We present applications from CFD and medical visualization and show how the skeletal structure can be used in these domains. We also illustrate a technique which uses the skeleton to extract the centerline for surgical navigation.

Keywords: Scientific Visualization, Medical Visualization, Skeleton, Volume Thinning, Centerline, Surgical Navigation

Acknowledgement

The research reported here was made possible through the support of the New Jersey Commission on Science and Technology and the CAIP Center's Industrial Members.

The work for this paper was done at the Laboratory for Visiometrics and Modeling at Rutgers University. The authors are grateful to Dr. Marsha Jessup of the Robert Wood Johnson Medical School for valuable input on medical visualization and to Dr. Bernhard Geiger of Siemens Corporate Research for providing the trachea dataset. Special thanks also go to Dr. N. Zabusky, X. Wang and J. Ray for useful discussions. The lab also acknowledges the support of ARPA HPCD, DOE and the CAIP center.

Table of Contents

List of Figures

1. Introduction

Skeletonization is a process for reducing an image/object [1] to a thinner remnant that largely preserves the extent and connectivity of the original region. In a more general sense, a skeleton is a reduced representation that captures the essential features of the image under investigation. Under this general description, the skeleton is not just a simple curve; rather it is a collection of points. We can define different types of skeletons depending on the domain. The geometric skeleton of an object can be defined as the locus of points that are centered with respect to the object boundary. Such a representation can serve as a powerful shape descriptor for object identification and matching. Other types of skeletons use information about the physical properties of the object, e.g. vortex cores [1]. The process of skeletonization is sometimes referred to as volume thinning, medial-axis generation or centerline extraction. In this paper, our focus is on the geometric skeleton.

Since a skeleton is an efficient and compact shape descriptor, it can give useful cues for a number of applications in visualization. These include

- automatic navigation
- compression
- shape description and abstraction
- tracking

However, these applications are so varied and the requirements so diverse, that no single method can be used to generate skeletons for all of them. Some applications require the skeleton to be as thin as possible while others impose the condition that the object be reconstructible from the skeleton. These are two conflicting requirements and existing algorithms favor one over the other. Extremely thin skeletons, which are essentially centerlines are useful in automatic navigation and surgical path planning [2, 3]. Automatic path planning aids a surgeon in precisely exploring a virtual model of the inner anatomy of a patient by a non-invasive technique. The centerline is used as a guiding path for a virtual camera. Centerlines also find use in animation control, where the motion of a human subject is mapped to an animated figure. The motion can be broken down into a set of basis motions of a "stick-like" representation of the human, and these same basis motions can then be applied to the animation. When used for shape abstraction in cognitive vision, it is desirable to have a skeleton which captures all the branches in the object. As a compression tool, we would like to be able to reconstruct the original object from the skeleton. In this case, it is necessary to preserve as much information as possible, leading to a very thick skeletal structure. The shape information in a skeleton can also be used for tracking objects [4]. While it is essential to preserve as much shape information as possible, reconstruction is not a requirement for tracking, so we could use a moderately thin skeleton.

This paper discusses a skeletonization algorithm that allows control over the density of the skeleton and thus can be used for a range of applications. In Chapter 2, we look at some of the existing work

[1] Image is a collection of pixels in 2D, object refers to a collection of voxels in 3D.

and methods for 3D skeletonization and their applications in visualization. Chapter 3 describes the algorithm for extracting the skeletal points and a method to generate a centerline. Chapter 4 deals with the data structures and issues for an efficient implementation. In Chapter 5 we present examples and applications from current research in medical and CFD visualization.

2. Previous Work

The concept of using skeletons to describe shape has been a topic of interest to researchers in Computer Vision for over 30 years. The field has been well explored in 2D and many algorithms have been extended to three dimensional thinning. Most of the three dimensional thinning algorithms, which identify points to be removed, belong to one of the three categories described below. In the literature and the descriptions that follow, 'Point' refers to a pixel/voxel and 'Ball' refers to the digitized version of a sphere.

Topological Thinning

Topological thinning is influenced by the work done in characterizing the topological properties of 2D images. Methods like [5, 6] identify *simple points*, the removal of which does not change the topology of the image. The simple point test essentially reduces to checking the local neighborhood of a point to determine whether removal of the point will disconnect the neighboring points. Since the test is purely based on local connectivity, certain primitive shapes like cuboids might be excessively thinned (to just a point). Therefore, certain simple points which are end-points are left unchanged to preserve useful information about the shape of the object. One characterization of end-points in a two-dimensional digital picture is a point which is adjacent to just one other point. Several authors [7, 8, 9] have tried to extend the idea of topological thinning to three dimensions. However, characterizing 3D end-points does not easily extend from the 2D approach. Morgenthaler [10] has attempted a study of such a characterization. This class of algorithms can also be easily parallelized as described in [11]. Topological thinning can guarantee connected skeletons at the cost of reconstructibility. Since these methods use a topological property like the Euler characteristic to test points for removal, two objects with the same Euler characteristic but different geometries would be thinned to similar skeletons. e.g. a cuboidal box with sharp corners and one with rounded corners would yield the same center line unless a robust end-point test is used.

Distance Transform Methods

The Distance Transform (DT) at a point within an object is defined as the minimum distance to a boundary point. Since the skeleton is required to be centered with respect to the object boundary, the distance transform gives useful cues for point removal. Points closest to the center of the object would have the maximum distance transform value and would not be removed while thinning. Different metrics can be used to compute the distance transform. The computation of a correct Euclidean DT is neither efficient nor algorithmically trivial. Several algorithms have been proposed for the Euclidean DT [12, 13, 14]. The Euclidean metric can be approximated by Manhattan or *chessboard* metrics for faster computation. The medial surface/axis can then be defined as the locus of maximal circles (2D) or balls (3D). The circles are constructed such as to have a radius equal to the distance transform at a point and a maximal circle is one which is not contained in the circle of any other point. The set of points thus extracted does not guarantee a connected skeletal representation. Niblack et al. [15] identify saddle points to get a connected skeleton of 2D images. It is difficult to identify saddle points in 3D because of the absence of a unique cyclic ordering of voxels (around a given voxel). This makes it hard to enforce connectiv-

ity of the skeletal points. Helman and Hesselink [16] find saddle points in a 2D vector field by computing the eigenvalues of the Jacobian matrix of the field. A vector field can be derived as the gradient of the distance transform, but the accuracy of saddle points depends on the method used to compute the gradient. Besides, the characterization of saddle points in 3D is computationally expensive. Moreover, noisy complex datasets would have too many saddle points which might not be necessary for applications requiring a thin skeleton. Distance transform methods are well suited for reconstruction of the object from the skeletal points and their distance transform values. If a Euclidean or quasi-Euclidean metric is used for the distance transform, the skeleton is robust under rotation of the object.

Voronoi Methods

The Voronoi Diagram is a well-known tool in Computational Geometry [17]. Given a set S of n points in a plane, the Voronoi polygon of a point P_i is the polygon enclosing all points in the plane that are closer to P_i than to any other point in S. The Voronoi Diagram (VD) is the collection of the Voronoi polygons of all the points in S. This concept can be extended to 3D as well, where the VD is the collection of Voronoi polyhedra. The medial-axis is a subset of the Voronoi Diagram. Since a maximal ball is tangent to the object boundary, its center is equidistant from at least two different points on the object boundary. Therefore, the VD of points on the object boundary will yield Voronoi edges/faces near the center of the object which are equidistant from two or more boundary points, giving part of the medial-axis. A 2D skeletonization algorithm based on the Voronoi Diagram of a shape's boundary points is described in [18]. [19, 20, 21] compute the 3D medial-axis/skeleton by using some form of the Voronoi Diagram or its dual, the Delaunay triangulation. Voronoi methods are best suited for polygonally defined objects for which volumetric models are not available.

Various forms of volume-thinning have been used for visualization applications. Itoh et al. [22] use a volume thinning technique to search for cells intersected by an isosurface. They use a topological thinning strategy based on a classification of cells. Their method is a very specific application since it needs to preserve extrema for propagating the isosurface. Hong et al. [3] use the onion-peel technique [23] to produce a flythrough path for virtual colonoscopy. The onion-peel technique is a method for 2D skeletonization based on topological thinning. In the absence of a "complete" characterization of 3D end-points, such an approach could excessively thin the object, missing important features. Moreover, two end-points for the flythrough path need to be specified . Specifying two such end-points needs considerable knowledge of the shape geometry since it is desirable to have the points centered in the object interior. It is also desirable to be able to deal with bifurcated objects.

Most of the existing skeletonization algorithms do not allow much control over the complexity of the skeleton. Topological thinning works well for smooth, regular objects. However, objects in the visualization domain tend to have noisy boundaries which would cause a lot of the points to be identified as end-points by a topological thinning method. Algorithms based on the Voronoi Diagram need a description of the object boundary, and it is hard to determine the density of the boundary point distribution a priori. Boundary noise would cause the Voronoi Diagram to be very dense which needs to be appropriately pruned to generate the medial axis. In this paper, we look at a skeletonization method for 3D objects based on the distance transform, which allows the user to specify the complexity of the skeleton using a *thinness parameter*. Since it is based on the distance transform, it is centered with respect to the object. It has the additional properties of invariance under rotation and it captures sharp corners which a topological method may miss if the end-point test is not strict. The effect of boundary noise can be reduced by choosing a

thinness parameter that removes most of the "hairs" and "spikes" in the skeleton.

3. Algorithm

In this chapter, we describe an algorithm for parameter controlled volume thinning based on a quasi-Euclidean distance transform. The volume is considered to be uniformly sampled in all three dimensions. A voxel is the smallest unique element of this sampled volume. Voxels can be partitioned into *object-voxels* and *background-voxels*. The object-voxels are taken to be 26-connected and the background-voxels are taken to be 6-connected (for a discussion on connectedness see [24]). *Boundary-voxels* are object-voxels that are 6-neighbors of background-voxels, i.e. they lie on the boundary. For a voxel p, we define *F-neighbors* (face), *E-neighbors* (edge) and *V-neighbors* (vertex). F-neighbors of a voxel are the 6-neighbors and share a face with the voxel in a cubic grid. E-neighbors are the 18-neighbors that are not 6-neighbors, i.e. they share an edge of the voxel cube. V-neighbors are the 26 neighbors that are not 6-neighbors or 18-neighbors. The aim of the algorithm is to identify a set M of object voxels that satisfies most of the following properties:

- the voxels in M are geometrically centered with respect to the boundary voxels;
- the object voxels are reproducible from the set M. This is the criterion for object reconstructibility;
- M forms a minimal set for a given thinness value; and,
- the voxels in M are 26-connected.

3.1 The Distance Transform

The distance transform at a voxel p = {x,y,z} is defined as

$$DT_p = \min_{(i,j,k)} \{d_t((x,y,z),(i,j,k)) : (i,j,k) \in BV\}$$

where d_t is the distance from voxel (x,y,z) to voxel (i,j,k) and BV is the set of boundary voxels.

The distance transform can be computed by using neighborhood masks which are based on the idea that global distances in the image are approximated by propagating local distances [25]. A 3x3x3 neighborhood mask is used with a 3-4-5 metric which approximates the Euclidean metric fairly well.

3.1.1 Algorithm for the Distance Transform

Let **S** be the set of object-voxels, $\bar{\mathbf{S}}$ be the set of background-voxels and **BV** denote the set of boundary-voxels. We use a peeling technique which propagates the boundary inwards, assigning distance transform values to object-voxels which are in the neighborhood of the boundary-voxels. The distance transform value for a voxel is updated only if the new value is smaller than the current value.

For all voxels $p \in \mathbf{S}$, assign a distance transform $DT_p \leftarrow \infty$
Calculate the Distance transform of boundary-voxels
For all voxels $p \in \mathbf{S}$ that have a (face/edge/vertex) neighbor $q \in \bar{\mathbf{S}}$
    $DT_p \leftarrow$ (3 for face / 4 for edge / 5 for vertex)
    Add p to **BV**
Propagate the boundary inward
Repeat for all $p \in \mathbf{BV}$
        Find all voxels $r \in \mathbf{S}$ which are (face/edge/vertex) neighbors of p
        Assign $DT_r \leftarrow \min\{ DT_r, DT_p +$ (3 for face / 4 for edge / 5 for vertex)$\}$
        Remove p from **BV**
        Add r to **BV**
until no DT_r is modified.

3.2 Skeleton Extraction

One desirable property of a skeleton is the ability to reconstruct the object from the skeletal voxels and the distance transform values associated with these voxels. The condition of reconstructibility also makes the skeleton accurate in the sense that there are longer spines in regions with sharp corners or curvature changes. We state some definitions and observations below :

Definition 1 *If a voxel p has a distance transform DT_p, the ball $B(p)$ associated with p is the set of object voxels q such that the transform distance $d_t(p, q)$ from p to q is strictly less than DT_p.*

Definition 2 *The ball for an object voxel is* maximal *if it is not contained in the ball of any other voxel.*

Observation 1 *The set of voxels whose balls are maximal is sufficient to reconstruct the object.*

This observation is true because every voxel in the object is contained at least in its own ball, and all non-maximal balls are contained in maximal ones. We now introduce the concept of a witness voxel in a manner similar to that in [15].

Definition 3 *The* witness *for a voxel p is any 26-neighbor q such that the distance transform of q, $DT_q = DT_p$ - (3 for F-neighbor / 4 for E-neighbor / 5 for V-neighbor). Thus, if a voxel q is a witness for a voxel p, $B(q) \subset B(p)$.*

Since the ball associated with a witness voxel is contained in the ball of another voxel, the set of non-witness voxels should suffice for reconstruction of the object. However, this set of non-witness voxels is not a minimal set for reconstruction. This holds because the ball of a voxel may not be completely contained in that of another, but may be contained in the union of the balls of several other voxels. This is illustrated by the example in Figure 3.1. The figure shows a 5x5x5 cube with the distance transform value at every voxel indicated by the number inside. Figure 3.1a shows the ball of the black voxel, the voxels in the ball being marked gray. Figure 3.1b,c show the ball of each of the other voxels marked black. All the black voxels are non-witness voxels, but the ball of the black voxel in Figure 3.1a is contained in the union of the other two balls. Hence, the non-witness voxel marked black in Figure 3.1a is not essential to the skeleton.

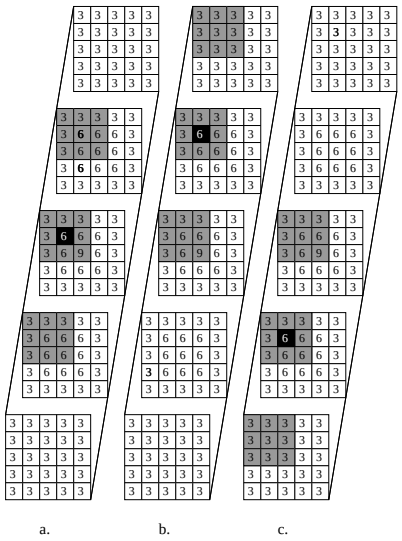


Figure 3.1: Minimal set for reconstruction

The identification of the essential non-witness voxels is not a simple problem. A brute force approach which grows the ball of every non-witness voxel and checks it for inclusion in the balls of all other voxels would be computationally intensive. An equivalent approach is to check only the 26-neighborhood of every voxel.

Claim 1 *The ball of a voxel p must be contained in the ball of one of its 26 neighbors if it is to be contained in the ball of any other voxel in the object.*

If the ball of a voxel p is contained in the ball of another voxel q, $DT_q > DT_p$. Therefore, there exists an uphill path of increasing distance transform value from p to q, which passes through voxel p' which is a neighbor of p. Therefore, the ball of p' contains the ball of p. It also follows by a similar argument that if the ball of p is contained in the union of the balls of voxels q_1, q_2 and q_3, it must be contained in the union of the balls of the neighbors of p, namely p', p'' and p'''. The complete proof is given in [26].

By definition, the ball of every non-witness voxel is maximal, so no single neighbor's ball completely contains the ball of a non-witness voxel p. As mentioned above, it can be contained in the union of the balls of neighboring voxels. If such neighbors exist, then their balls have to be at least as big as the ball of p, which implies that their distance transform should be greater than or equal to DT_p. Rather than grow the ball for every neighbor and scan for containment in the union of balls, a simple approach is to average the distance transform values of the neighbors q_i of p. The motivation behind summing the distance transform value of the neighbors is that if there are several neighbors with a higher distance transform value, the ball of p could be contained in the union of their balls. Therefore, if the mean of the neighbors' distance transform, MNT_p is close to or greater than DT_p, we do not want to keep p in the skeleton.

Definition 4 $MNT_p = \frac{\sum_{i=1}^{26} DT_{q_i}}{26}, p, q \in \mathbf{S}, q$ *is a 26-neighbor of* p.

We introduce the *thinness parameter* TP, that is used to control how close MNT_p should be to DT_p for p to be added to the skeleton.

Condition 1 *If* $MNT_p < DT_p - TP$, *add* p *to the skeleton.*

A low value of TP indicates that p is retained in the skeleton if its distance transform is slightly greater than that of its neighbors. This results in a thick skeleton. A high value of TP means that for inclusion in the skeleton, p must have a distance transform that is much greater than that of its neighbors, resulting in a thinner skeleton.

3.3 Extracting the Centerline

The parameter controlled skeletonization outlined above thins the volume, keeping only the voxels that satisfy **Condition 1**. These skeletal voxels are not generally connected. However, since they were based on the criterion for reconstructibility, the skeletal voxels capture the essential shape properties of the object. They can now be processed for a variety of applications.

A centerline is a curve that is centered with respect to the object boundaries. It can serve as the path for a virtual camera in surgical path planning. We describe a simple midpoint subdivision algorithm to generate the centerline from the skeletal voxels. It is a semi-automatic algorithm in which the user specifies end-points for centerline generation. These end-points are specified from among the skeletal voxels. This strategy enables the user to accurately pick end-points since the skeletal voxels are already centered with respect to the object. Moreover, it allows multiple centerlines to be rapidly generated for interactive exploration of the dataset. This is true because the skeletal voxels are pre-computed and for every new path, the object does not have to be thinned again. The operator can see the skeletal voxels and thus all bifurcations and bumps are available as potential navigation paths.

3.3.1 Algorithm for the Centerline

Let **SK** be the set of skeleton voxels. Let p_1 and p_2 be the end-points of the centerline such that $p_1, p_2 \in$ **SK**. We have a subdivision parameter ("fineness") F, which determines the number of points along the centerline and gives a stopping condition for the recursion.

Centerline (p_1, p_2, F)
{
    If (distance (p_1, p_2) $< F$) return
    Find the midpoint p_m, of p_1 and p_2 such that $p_m = \frac{p_1+p_2}{2}$
    Locate point $q \in$ **SK** such that
        distance (q, p_m) is minimum for all $q \in$ **SK**
    Call Centerline (p_1, q, F)
    Call Centerline (q, p_2, F)
}

The method outlined above is a simple, fast approach and works well for tube-like objects frequently occuring in medical applications. Since it uses the closest points, it could get perturbed by small "hairs" in the skeleton. A better method would use the distance transform value stored in the skeletal voxels and use the closest point with the greatest distance transform value for the recursive subdivision. This is discussed in Chapter 6.

4. Implementation

The object is represented as an octree structure with object voxels in the leaf nodes and background voxels stored as NULL leaves. The octree facilitates fast searching and optimizes memory utilization.

4.1 The Distance Transform

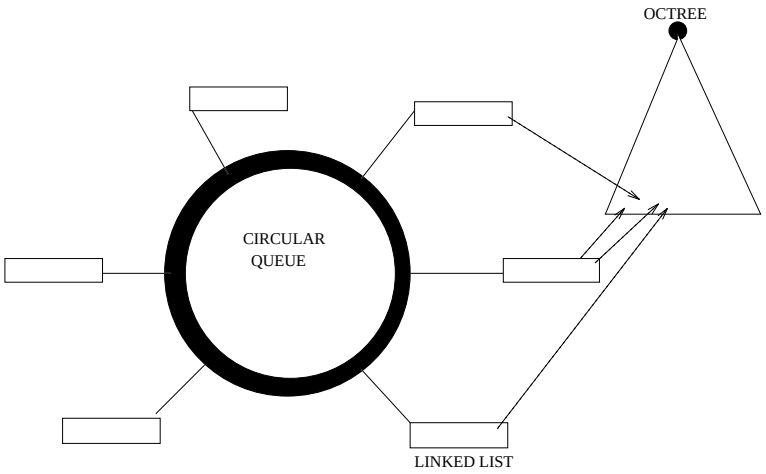


Figure 4.1: Data Structure for the Distance Transform

To compute the distance transform, a circular queue (Figure 4.1) is used to keep track of the distance values for successive peeling. We use the fact that at any given instant, if we are processing points with a distance value k, their neighbors can get DT values of $k+3$, $k+4$ or $k+5$ only. We create a linked list for every DT value; the list stores pointers to the point in the octree. The circular queue is initialized with lists for DT values 3, 4 and 5 with the head of the queue at 3. As we process the list for DT=3, lists for DT values of 6, 7 and 8 are created and are added in order to the tail of the queue. Once the list for DT=3 is processed, we move the head of the queue. A point is added to the list only if its new DT value comes out to be smaller than the existing one. Such an implementation makes the computation very fast, since the time taken is of the order of the number of object-voxels. The computation stops when the queue is empty.

4.2 Extracting the Skeleton

Once the distance transform value at every object point is known, a linked list of all the object points is created with pointers to the octree nodes. One pass is done over this list to test if the point is a local maximum or if it satisfies **Condition 1** (described in the previous chapter). If the point is either a local maximum or it satisfies the thinness test, the point is marked as being in the skeleton.

The above implementation is easily parallelizable. Each point can be independently examined for inclusion in the skeleton, and the order of scanning the points does not affect the results. With N processors, one can theoretically achieve a speedup of N.

Since the algorithm is not based on connectivity, the set of points marked as skeletal points are not necessarily a single connected component. A simple connection scheme is used, which consists of marching in the direction of maximum distance transform gradient (also called steepest uphill path) from points which have less than two neighbors in the skeleton. The uphill march proceeds by successively stepping to equal or greater valued neighbors, and terminates at another skeletal voxel or a local maximum.

4.3 Reconstruction and Sphere Growing

In order to reconstruct the object from the skeletal points, spheres of radius equal to the distance transform value have to be constructed, with their center at each of the points. A recursive strategy is adopted to implement this. A sphere growing path is initiated from every skeletal point and starts with a value equal to the distance transform at that point. Every move to a neighboring point incurs a cost of either 3, 4 or 5 for face, edge and vertex neighbors respectively, and the sphere value is decremented by the cost. The neighbor points then serve as starting points for a new sphere growing path with the decremented value. A path terminates when its value is less than or equal to 3. All points along every sphere growing path are inserted into the reconstructed object.

5. Examples and Applications

In this chapter, we apply our algorithm to three example datasets. The first example uses a set of simple digitized shapes. We then look at an example from medical visualization and another from computational fluid dynamics.

Figure 5.1a shows a 9x9x9 cube. The skeleton is shown in Figure 5.1b for a thinness value of 2.0. Using the skeletal points, the object is reconstructed as shown in Figure 5.1c.

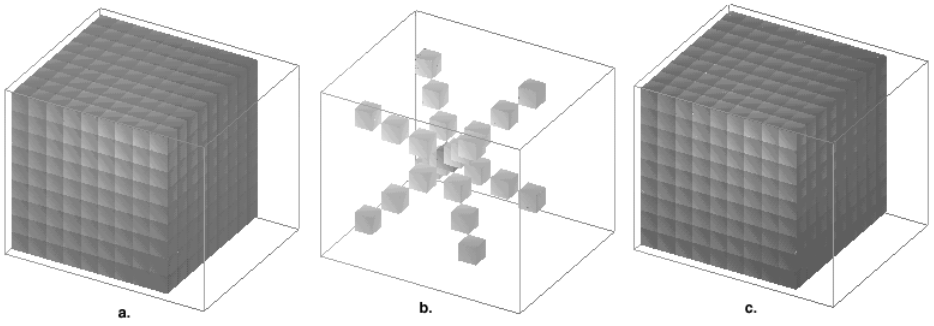


Figure 5.1: Exact reconstruction

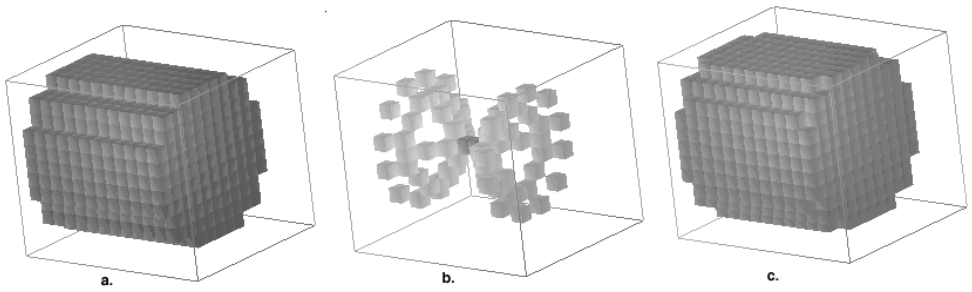


Figure 5.2: Inexact reconstruction

Figure 5.2a is a digitized cylinder with height 12 and radius 6. The skeleton in Figure 5.2b has been extracted using a thinness value of 1.6. Figure 5.2 (c) shows the reconstructed object. Note how the ball growing process *squares out* the rounded corners because of the pseudo-Euclidean metric, hence extra voxels are observed in the reconstructed object.

Figure 5.3 shows the skeleton for a cylindrical object with varying thinness parameter. The sharp curvature discontinuity at the faces of the cylinder is responsible for the multiple spikes, which increase in density as the thinness parameter is reduced.

The result of applying our algorithm to an S-shaped object constructed from cuboidal elements is shown in Figure 5.4. A thinness parameter of 2.0 was used for this example.

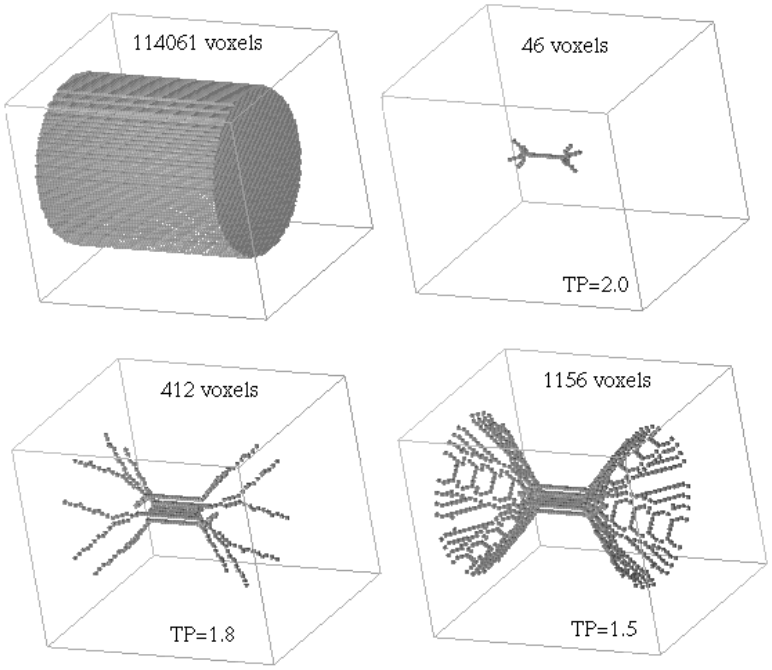


Figure 5.3: Varying Thinness Skeletons

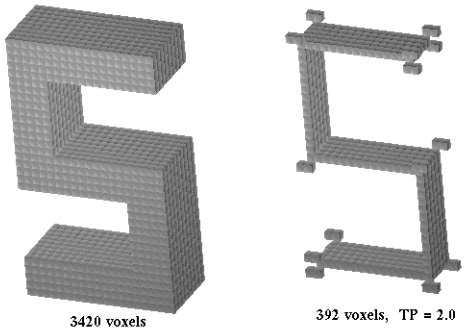
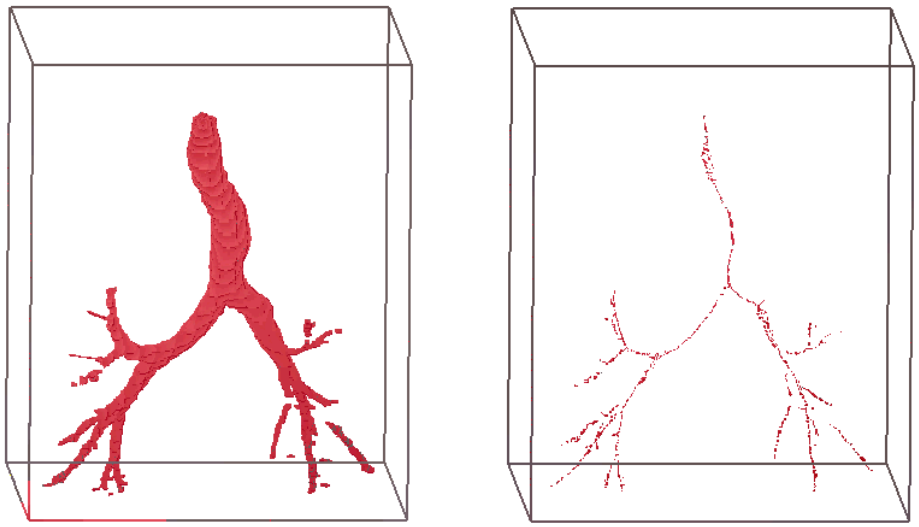


Figure 5.4: Skeleton of S shape

5.1 Centerlines for Surgical Navigation

We demonstrate our centerline algorithm on the segmented human trachea obtained from Siemens Corporate Research (courtesy of Dr. Bernhard Geiger). The dataset consists of 281 slices of 512x512 images, and the segmented trachea has 148,892 voxels. The complete trachea and its skeleton with a thinness parameter of 2.5 are shown in Figure 5.5. The skeleton captures all the bifurcations in the object and includes the disjoint fragments of the object.

We apply our algorithm to the first 180 slices of the dataset which contain the major bifurcation of the trachea. The user defines two end-points per centerline from the set of skeletal points. Two different paths are generated, with the user-defined end-points as indicated in Figure 5.6a The trachea with the centerline inside it is shown in Figure 5.6b. The centerline for both the paths was constructed using a subdivision parameter of 5.0. Figure 5.7 shows camera shots taken from 4 different points along the centerline with the camera looking downwards into the bifurcated part.



a. b.

Figure 5.5: Trachea and its Skeleton (Thinness 2.5)

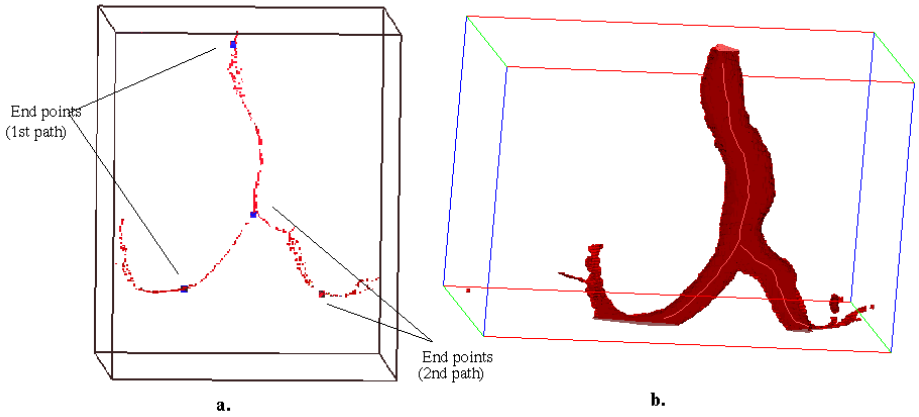
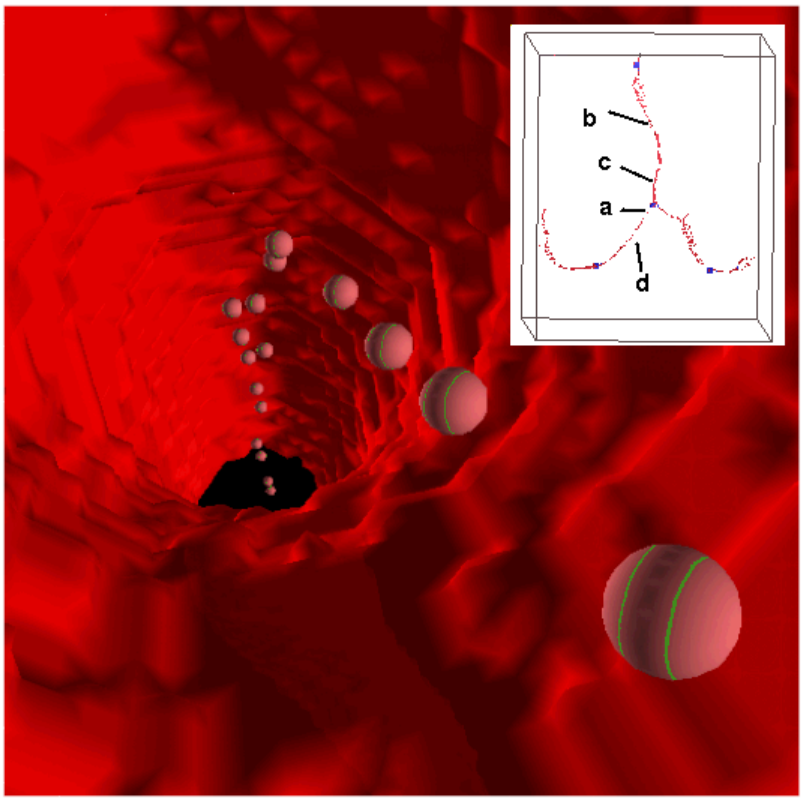
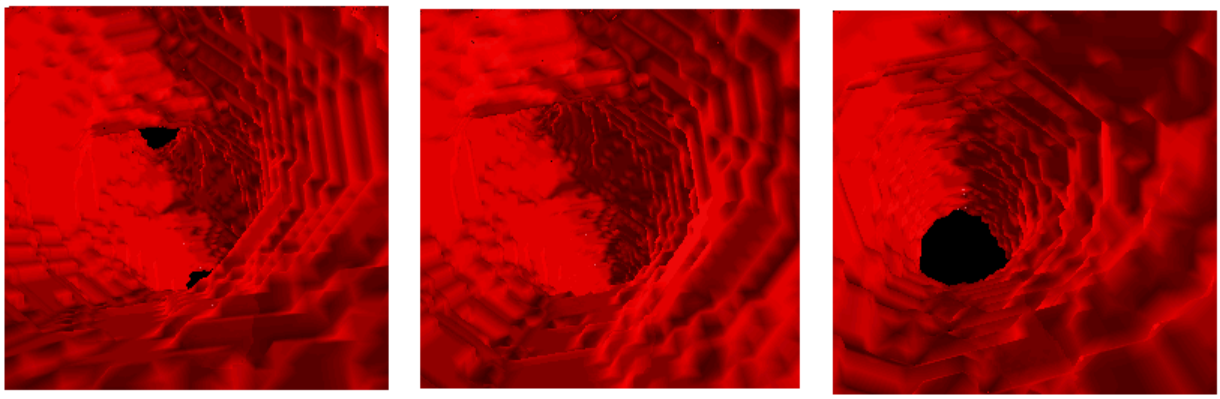


Figure 5.6: Generating the Centerline



a.



b. c. d.

Figure 5.7: Camera views from the Centerline

The inset legend indicates the camera position for each of the views. Figure 5.7a highlights the points along the centerline. Note that the two ends of the trachea and the bifurcation are clearly seen in Figure 5.7b which shows the view from a point halfway into the straight part. Figure 5.7c shows the camera view from close to the bifurcation point and Figure 5.7d shows the view from

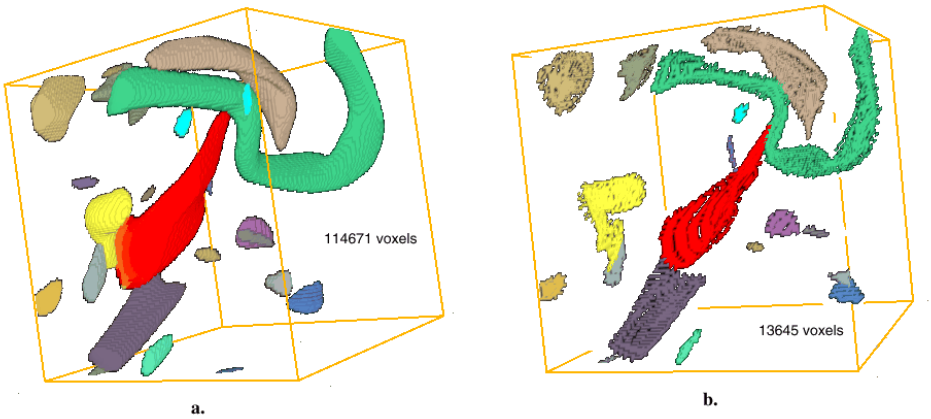


Figure 5.8: Vortex structures and their Skeletons

Thinness	% Error	Total time (sec)	% Time
None	0	344.62	100
0.5	4.32	89.38	25.9
1.0	5.15	52.48	15.2
2.0	14.50	15.92	4.6

Table 5.1: Error and speedup for feature tracking with skeletons

a point in the left branch.

5.2 Skeletons in Feature Tracking

In [4], Silver and Wang use a volume based approach for tracking features. They extract features from a time varying dataset and perform a volume difference test over timesteps to match features. Objects which match to a tolerance value are considered to be the same. Since the complete volumes are used for the difference test, the algorithm is computationally intensive. However, the skeleton of each volume can be used to perform matching across timesteps. The number of voxels in the skeletons ranges from 1% to 15% of the original object voxels. We present results for tracking a 128^3 dataset which is from a simulation of turbulent vortex structures. The vortex structures are tracked over 10 timesteps, first using the complete volume data and then using skeletons of three different thinness values. We compare the time taken for tracking the skeletal points and the the time to track the complete volume. The percentage error is also calculated and results are tabulated in Table 5.1. A significant speedup is observed when the skeletal points are tracked. Most of the errors are due to very small features disappearing when subjected to skeletonization. Very often, these small features arise due to noise or improper segmentation and are not of great importance. The skeleton of a typical dataset being tracked is shown in Figure 5.8. A thinness value of 1.0 was used for this figure.

6. Conclusions and Future Work

Skeletons are useful for a range of visualization applications including automatic navigation, shape abstraction and feature tracking. In this paper, we have presented a versatile and flexible algorithm to extract the set of skeletal points from a 3D object.

We use a single pass method which gives $O(N)$ complexity both for distance transform and skeleton extraction (where N is the number of object-voxels). The algorithm automatically favors reconstruction over connectivity because of a formulation based on the distance transform. Complex shapes with noisy boundaries generate many skeletal points near the boundary to capture the sharp curvature changes. Very often, it is not possible to connect up the fragments of the skeleton by a simple strategy as described in this paper, due to the existence of loops to equal valued neighbors in an uphill climb. If all the points are finally connected, the skeleton comes out to be thicker than desired, with extraneous points. A hybrid scheme which looks at local connectivity while considering potential skeletal points could solve this problem. To address the problem of thick skeletons, the Voronoi diagram of a sparse distribution of boundary points could be used to generate a rough medial axis, which could then be used as a guide for identifying outlying skeletal points which need to be culled.

We have also described a simple method to construct the centerline from the skeletal voxels. The method is fast and captures all the shape properties of the object. No prior knowledge of the object geometry is necessary. It is also more interactive than existing methods since every new path is generated from the small set of skeletal voxels avoiding the huge computational cost associated with thinning the volume for every new path. A better method to construct the centerline would exploit the distance transform at the skeletal points to propagate the line, and avoid the problems due to "spikes" and "hairs" in the skeleton. We are exploring such a possibility which would result in a robust centerline which is insensitive to boundary noise.

References

1. David Banks and Bart Singer. Vortex Tubes in Turbulent Flows: Identification, Representation, Reconstruction. In *Proceedings Visualization'94 IEEE*, pages 132–139. Computer Society Press, 1994.

2. R. A. Robb. Virtual (Computed) Endoscopy: Development and Evaluation Using the Visible Human Datasets. In *Visible Human Project Conference*, October 1996.

3. L Hong, A. Kaufman, Y-C. Wei, A. Viswambharan, M. Wax, and Z. Liang. 3D Virtual Colonoscopy. In *IEEE Symposium on Frontiers in Biomedical Visualization*, pages 26–32, 1995.

4. D. Silver and X. Wang. Volume Tracking. In *Proceedings IEEE Visualization '96*, pages 157–164, San Francisco, CA, 1996.

5. C. Arcelli and G. Sanniti di Baja. A Width-Independent Fast Thinning Algorithm. *IEEE Trans. Pattern Recognition and Machine Intelligence*, 7(4):463–474, 1985.

6. T. Pavlidis. A Thinning Algorithm for Discrete Binary Images. *Computer Graphics and Image Processing*, 13:142–157, 1980.

7. Y. F. Tsao and K.S. Fu. A 3D Parallel Skeletonwise Thinning Algorithm. *Proc. IEEE Pattern Recognition Image Processing Conf.*, pages 678–683, 1982.

8. G Bertrand. A Parallel Thinning Algorithm For Medial Surfaces. *Pattern Recognition Letters*, 16:979–986, 1995.

9. J. Mukerjee, Das P.P, and B.N. Chatterji. Thinning of 3-D Images Using the Safe Point Thinning Algorithm (PTA). *Pattern Recognition Letters*, 10:167–173, 1989.

10. Morgenthaler D.G. Three Dimensional Simple Points: Serial Erosion, Parallel Thinning, and Skeletonization, TR-1005. Technical report, Computer Science Center, University of Maryland, College Park, 1981.

11. C.M. Ma and M. Sonka. A Fully Parallel 3D Thinning Algorithm and Its Applications. *Computer Vision and Image Understanding*, 64(3):420–433, November 1996.

12. H. Yamada. Complete Euclidean Distance Transformation by Parallel Operation. In *Proc. 7th Intl. Conf. on Pattern Recognition*, pages 69–71, Montreal, Canada, 1984.

13. I. Ragnemalm. The Euclidean Distance Transformation in Arbitrary Dimensions. *Pattern Recognition Letters*, 14:883–888, 1993.

14. T. Saito and J. Toriwaki. New algorithms for Euclidean Distance Transformation of an n-Dimensional Digitized Picture with Applications. *Pattern recognition*, 27:1551–1565, 1994.

15. W. Niblack, P.B. Gibbons, and D. Capson. Generating Skeletons and Centerlines from the Distance Transform. *CVGIP : Graphical Models and Image Processing*, 54(5):420–437, September 1992.

16. J.L. Helman and L. Hesselink. Visualization of Vector Field Topology in Fluid Flows. *IEEE Computer Graphics and Applications*, 11(3):36–46, 1991.

17. F. P. Preparata and M. I. Shamos. *Computational Geometry.* Springer-Verlag, New York, 1990.

18. R.L. Ogniewicz and O. Kubler. Hierarchic Voronoi Skeletons. *Pattern Recognition*, 28(3):343–359, 1995.

19. J. M. Reddy and G. M. Turkiyyah. Computation of 3D Skeletons Using a Generalized Delaunay Triangulation Technique. *Computer-Aided Design*, 27(9):677–694, September 1995.

20. D.J. Sheehy, C.G. Armstrong, and D.J. Robinson. Shape-Description by Medial Surface Construction. *IEEE Trans. on Visualization and Computer Graphics*, 2(1):62–72, March 1996.

21. E.C. Sherbrooke, N.M. Patrikalakis, and E. Brisson. An Algorithm for the Medial Axis Transform of 3D Polyhedral Solids. *IEEE Trans. on Visualization and Computer Graphics*, 2(1):44–61, March 1996.

22. T. Itoh, Y. Yamaguchi, and K. Koyamada. Volume Thinning for Automatic Isosurface Propagation. In *Proceedings IEEE Visualization '96*, pages 303–310, San Francisco, CA, 1996.

23. T. Pavlidis. *Algorithms for Graphics and Image Processing.* Computer Science Press, 1982.

24. T.Y. Kong and A. Rosenfeld. Digital Topology: Introduction and Survey. *Computer Vision, Graphics and Image Processing*, 48:357–393, 1989.

25. G. Borgefors. A New Distance Transformation Approximating the Euclidean Distance. In *Proc. 8th ICPR*, pages 336–339, 1986.

26. Nikhil Gagvani. Skeletons and Volume Thinning in Visualization. MS. Thesis, Dept. Of Electrical and Computer Engineering, Rutgers University, New Brunswick, New Jersey, 1997.