Parameter Controlled Volume Thinning

Nikhil Gagvani[1]
Sarnoff Corporation, CN 5300, Princeton, NJ 08543-5300.
Email : ngagvani@sarnoff.com

Deborah Silver
Department of Electrical and Computer Engineering and CAIP center,
CoRE Building, P.O. Box 1390,
Rutgers University, Piscataway, NJ 08855-1390.
Email : silver@caip.rutgers.edu

Manuscript for Graphical Models and Image Processing
May 1999

[1]Author for correspondence

Abstract

The availability of large 3D datasets has made volume thinning essential for compact representation of shapes. The density of the skeletal structure resulting from the thinning process depends on the application. Current thinning techniques do not allow control over the density and can therefore address only specific applications. In this paper, we describe an algorithm which uses a thinness parameter to control the thinning process and thus the density of the skeletal structure. 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 to construct a centerline for surgical navigation.

1 Introduction

With the widespread availability of MR and CT scanning equipment three dimensional volume datasets are becoming commonplace. Such images find application in medicine, luggage scanners and automated inspection of moulded and cast manufactured products. Large fluid dynamics and scientific simulations also give rise to three dimensional datasets. Regions of interest in these datasets, referred to as voxelized *objects* or *features* can be segmented out and presented to the user or scientist. However, these regions are sometimes too large and unwieldy for further inspection or analysis. A *thinning* procedure may be used to extract essential properties of the object. Thinning refers to the process of reducing an object to a simpler representation with fewer pixels/voxels that still retains the essential features of the original object. The process of thinning results in the *skeleton* of the object. Skeletal representations are used in a variety of applications.

The skeleton is related to the *medial-axis* which is the locus of points centered with respect to the boundaries of an object. For three dimensional objects, the medial-axis is not just a curve, but a surface, often called the *medial-surface*. A *centerline* is a curve like representation of the medial-surface for 3D shapes. It is very useful for path planning in *Virtual Endoscopy* [22, 10]. After segmenting an organ from an MRI scan, the centerline provides the camera path for automatic navigation and inspection of the organ. Such a virtual fly-through simulates the video from a real camera used in endoscopic procedures to detect polyps and tumours. The centerline can also be used to generate an accurate "stick" model of a volume object to achieve realistic animation [6]. In computer graphics animation, the motion of animated characters is controlled by manipulating the motion of a "stick" representation. In addition, if the original object can be reconstructed from the skeleton, the thinning process can be used for compression and speedup for fast matching and recognition.

It is desirable for the skeleton to be thin, centered, topologically accurate, and in many cases, allow reconstruction of the original object. When skeletons are used for shape description, the topological characteristics of the object need to be preserved. *Topological Thinning* algorithms deal with this property. The primary concern of these algorithms is the identification of simple points and end points. (Points refer to 2D pixels or 3D voxels which are part of the segmented region of interest). The simple point test checks the local neighborhood of a test point to determine whether removal of the test point will disconnect its neighbors. Since the test is purely based on local connectivity, certain primitive shapes like cuboids might be excessively thinned (to just one point). Therefore, certain simple points which are end-points are left unchanged to preserve useful information about the shape of the object. While these methods work well for two dimensional shapes [1, 19], efficient and complete characterization of end points in 3D is not easy [3, 14]. Connectivity of the resulting skeleton is implicit in topological methods. These methods work well for smooth objects but noisy (real-world) objects have to be smoothed prior to topological thinning.

Reconstructibility is a necessary property for compression applications. *Distance Transform* methods can satisfy this requirement by storing the minimum boundary distance or the Distance Transform value at every skeletal point. However, thinness and reconstructibility are two conflicting characteristics. Forcing connectivity of the skeleton may introduce extraneous points which are not essential for reconstruction, which conflicts with the thinness requirement. Distance Transform methods for 2D shapes [16] achieve connectivity by identifying "saddle-points" and "local maxima". Due to the absence of a unique cyclic ordering of points around a given point in 3D, identification of saddle points is not easy. Vector field characterizations of saddle points like those described in [9] require computation of the gradient and eigenvalues which is expensive. When a Euclidean or quasi-Euclidean metric is used for the Distance Transform, the skeleton is invariant under rotation of the object.

If the boundary representation of an object is available, the medial-axis and medial-surface can be extracted by pruning the *Voronoi Diagram* [20, 2] of the boundary points. 2D medial-axis algorithms based on the Voronoi Diagram of a shape's boundary points are described in [18, 5]. Computation of the Voronoi Diagram for arbitrary point sets needs special methods like rational arithmetic or perturbation schemes like simulation of simplicity [8]. [21, 24, 25] compute the 3D medial-axis/skeleton by using some form of the Voronoi Diagram or its dual, the Delaunay triangulation. These methods do not compute the Voronoi Diagram of general point sets and work well only for polyhedral solids used in CAD and solid modeling. A 3D Voronoi skeletonization algorithm for large, complex datasets with a topologically correct regularization method is described in [15]. It tries to address the issue of keeping the skeleton topologically accurate while pruning the Voronoi Diagram which often results in a very dense skeleton for complex objects. Voronoi methods need surface descriptions rather than volume models, therefore objects with holes and cavities would need special treatment. Moreover, extraction of the boundary for complex volume models results in a very dense set of

points [13], and the memory costs for the Voronoi diagram of such large points sets can be prohibitively large.

It is clear that existing methods for 3D thinning emphasize certain properties of the resulting skeleton depending on the application. The varied applications of skeletons have several conflicting requirements which makes it difficult for a single method to address all of them. While some applications require the skeleton to be as thin as possible, others impose the condition that the object be reconstructible from the skeleton. Automatic navigation of medical datasets needs an extremely thin skeleton or a centerline. This is also the case for generating "stick-like" representations for animation control. For compression, the original object must be reconstructible from the skeleton. Furthermore, if every minor feature in the original object needs to be captured, the resulting skeleton is very thick and dense. A tracking application [26] requires a balance between thinness and accuracy in order to be correct and fast.

Existing methods for 3D thinning do not allow control over the density of the skeleton. Topological thinning works well for smooth, regular objects. However, real world objects tend to have noisy boundaries which would cause a lot of the points to be identified as end-points by a topological thinning method resulting in fairly thick skeletons. Boundary noise can cause the Voronoi Diagram to be very dense, and topologically correct pruning techniques for 3D Voronoi skeletons are difficult to derive.

In Section 2, we discuss a 3D thinning algorithm based on the Distance Transform that allows *control* over the density of the skeleton and can therefore be used for a range of applications. The algorithm is applied to problems from surgical navigation and computational fluid dynamics (CFD) in Section 3.

2 Algorithm

This section discusses an algorithm for thinning volumetric objects. It is based on a thinness parameter which allows control over the density of the skeletal structure. A simple method for constructing a continuous centerline from the skeletal points is also described.

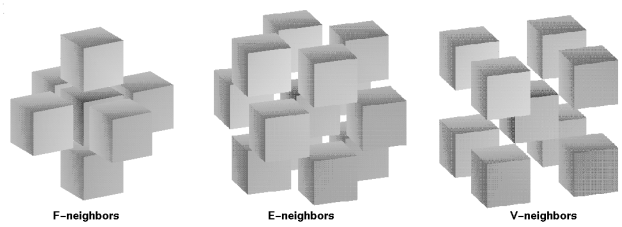

Figure 1: The Neighbors of a Voxel

The volume is considered to be uniformly sampled in all three dimensions. A voxel is the smallest unique element of this sampled volume. Voxels are 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 [11]). *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, i.e they share a vertex of the voxel cube. This concept is illustrated in Figure 1.

2.1 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. We compute the distance d_t using a $< 3, 4, 5 >$ weighted distance metric, which approximates the Euclidean metric fairly well [4].

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. In [4], Borgefors describes a two-pass method to compute the weighted distance transform in three dimensions. Her algorithm works on the complete cuboidal array consisting of object and background voxels. It consists of two passes over the entire array, each pass starting at diagonally opposite corners of the array cuboid. In each pass, a new value is computed for each object voxel by summing the values of already visited neighbors and the corresponding local distances, a, b or c and using the minimum of these sums. Leymarie and Levine [12] have shown that a mask propagation method can have the same complexity for Euclidean and weighted metrics. As described below, we do not compute the distance transform in scan-line order, therefore a Euclidean metric would introduce additional computational complexity. However, our parameter controlled thinning method applies equally for the Euclidean metric as shown in the next subsection.

In order to deal with extremely large data sets, the object is represented by an octree [23] structure and background voxels are not stored. Since it is not efficient to traverse the octree in array order, we use a peeling technique that successively propagates the distance transform inwards starting from the boundary voxels. In the first pass, all boundary voxels are identified. This is done by checking the 26-neighborhood of every object voxel for background voxels. If the object voxel has a background voxel as a F-neighbor, it is assigned a DT value of 3, otherwise, if it has any background voxel as its E-neighbor, but not as a F-neighbor, it is assigned a DT value of 4. If there are no background voxels which are F-neighbors or E-neighbors, but there is some background voxel which is a V-neighbor, the boundary voxel is assigned a DT value of 5. In the second pass, the algorithm then recursively checks neighbors of each of the marked voxels, adding 3, 4 or 5 to the distance transform depending on whether it is a F, E or V neighbor.

Algorithm for the Distance Transform

Let $\mathbf{S}$ be the set of object-voxels, $\bar{\mathbf{S}}$ be the set of background-voxels and $\mathbf{BV}$ 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.

A circular queue is used to keep track of the distance values for successive peeling. We use the fact that at any given instant, if points with a distance transform value k are being processed, their neighbors can get DT values of $k+3$, $k+4$ or $k+5$ only. A linked list is created for every distance transform value; the list stores pointers to all nodes in the octree which have a distance transform value corresponding to the list value k. In the first pass, the circular queue is initialized with lists for DT values 3, 4 and 5 with the head of the queue at 3. In the second pass, the boundary is propagated inwards. The list at the head of the queue is traversed and all neighboring object voxels are evaluated for a DT value which is the sum of the list value and the local distance increment, i.e. $3, 4$ or 5. If the new DT value is lower than the existing value, the voxel is added to a list corresponding to the new value. Once the list at the head of the queue has been processed, the head is moved to the next higher list for processing. The computation stops when the queue is empty, i.e there are no more distance transform values to process.

2.2 Skeleton Extraction

Once the distance transform has been computed, voxels which are essential to the skeleton have to be identified. We exploit the reconstructibility property of skeletons to identify these voxels. In the strict sense, this property implies that if the object were to be correctly reconstructed from the skeletal voxels and their distance transform values, the skeletal structure should capture all the shape characteristics. Therefore, the property of reconstructibility makes the skeleton accurate in the sense that there are longer spines in regions with sharp corners or curvature changes. By relaxing the reconstruction criterion, skeletons of varying density can be obtained. The essential theoretical considerations are developed by stating 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.*

The ball is a digitized version of a sphere. Its radius is determined by the distance transform value at a voxel and the ball is constructed using the distance metric, in this case, the $< 3, 4, 5 >$ metric.

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. The concept of a witness voxel is now introduced in a manner similar to that for witness pixels in [16].

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, $B(q) \subset B(p)$.*

Let the cost incurred in moving from p to a neighbor q be d_F, d_E or d_V for the cases when q is an F-neighbor, E-neighbor or V-neighbor. Then, the witness to a pixel p is any 26-neighbor q such that the cost incurred in moving from q to p is equal to the difference in their distance transforms. Thus, if a voxel q is a witness of a voxel p, then $B(q) \subset B(p)$.

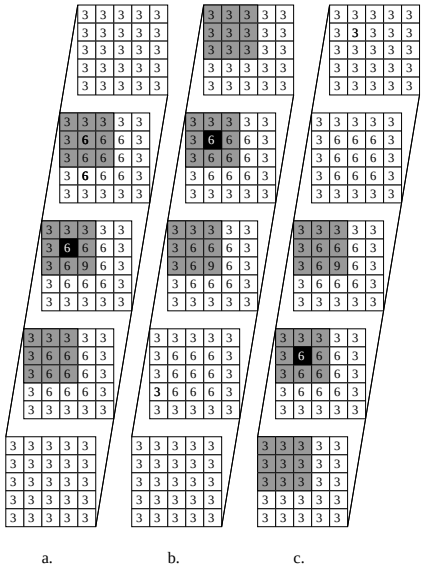

Figure 2: Minimal set for reconstruction

Since the set of balls associated with witness voxels is contained in the balls of other voxels, 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. Nilsson and Danielsson [17] have described a 2D method to iteratively identify the minimal set based on boundary coverage of locally maximum pixels. Since our method allows control over the density, the resulting set of points is allowed to be thinner than the minimal set required for complete reconstruction. 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. Figure 2 shows a 5x5x5 cube with the distance transform value at every voxel indicated by the number inside. Figure 2a shows the ball of the black voxel, the voxels in the ball being marked gray. Figures 2b and 2c 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 (a) is contained in the union of the other two. Hence, the non-witness voxel marked black in (a) is not essential to the skeleton.

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 expensive. It is however possible to check the 26-neighborhood of every voxel and this is equivalent to checking for inclusion in all other voxels.

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.*

Proof : Consider voxel p, the ball of which is contained in the ball of some voxel q. Then $B(q) > B(p), p \neq q$. Therefore, there exists an uphill path from p to q consisting of voxels of increasing distance transform value. Let this path go through voxel p', where p' is a 26-neighbor of p. Since p' is on the uphill path from p, $DT_{p'} \geq DT_p$.

Consider growing a ball around a voxel (Figure 3). We step through successive voxels reducing the distance transform by either d_F, d_E or d_V. Since the ball of q contains the ball of p, consider a series of ball propagating moves from q leading to p. The reduced distance transform value DT_{q-p} when the ball propagation reaches p must be greater than or equal to DT_p since $B(q) > B(p)$. Therefore, $DT_{q-p} \geq DT_p$.
The ball propagation path passes through p',
$\Rightarrow DT_{p'} = DT_{q-p} + (d_F, d_E, d_V)$ since $DT_{p'-p} = (d_F, d_E, d_V)$.

Therefore, it follows that $DT_{p'} \geq DT_p + (d_F, d_E, d_V)$ which implies that the ball of voxel p is contained within the ball of its neighbor p'. If p is a non-witness voxel, this relation is a strict inequality. If we have the case where the ball of p is contained in the union of the balls of voxels q', q'' and q''', we consider ball growing paths from each of these voxels which pass through neighbors p', p'' and p'''. By a similar argument as above, the ball of p must therefore be contained in the union of the balls of p', p'' and p'''.

To find non-witness voxels, the 26-neighbors of all object voxels need to be scanned. Since all non-witness voxels have maximal balls, no single neighbor's ball completely contains the ball of a non-witness voxel p. As illustrated above, it can however 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 value 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 averaging the distance transform values 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 $MNT_p = \frac{\sum_{i=1}^{26} DT_{q_i}}{26}, p, q_i \in \mathbf{S}, q_i$ *is a 26-neighbor of* p.

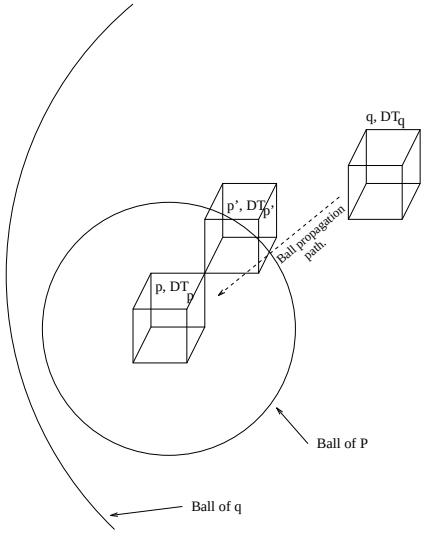

Figure 3: Growing the ball around a voxel

We introduce the *thinness parameter* TP, that allows control over the removal of non-witness voxels yielding skeletons of varying density. TP determines 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.*

The above condition requires that the distance transform of a voxel be greater than the mean of its neighbors' distance transform by at least TP for it to be included in 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. In a single pass over all the object voxels, skeletal voxels are marked if they satisfy Condition 1.

Figure 4 illustrates the concept of the thinness parameter using a $< 2, 3 >$ weighted metric. The mean of the neighbors distance transforms are 4.0, 3.875 and 3.75 for the dark pixel in Figs. 4a, 4b and 4c respectively. Therefore, the dark pixel will be included in the skeleton only if TP is less than 2, 2.125 and 2.25 respectively. This shows that the center pixel increases in importance as the shape boundary becomes irregular. Skeletons at thinness values of 0, 2, 4, and 6 for a 2D shape are shown in Figure 5. The squared Euclidean metric was used to compute the Distance Transform.

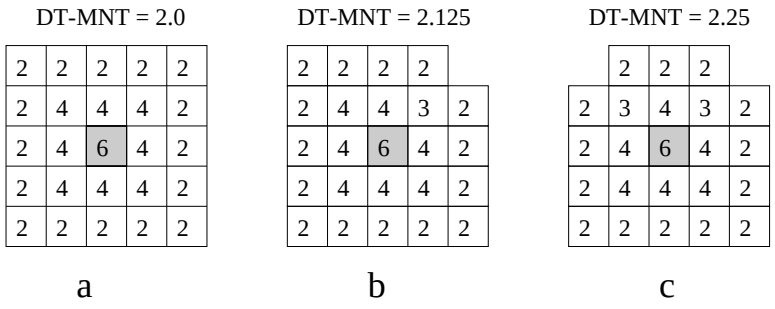

Figure 4: Mean Neighbor Distance Transform (MNT) and its relation to the maximum thinness (TP). The dark voxel will be included in the skeleton if TP is less than DT-MNT.

Figure 5: A maple leaf and its skeleton at various thinness values. Thinness is increasing from left to right.

The range of TP for a given metric can be computed by considering the maximum and minimum possible distance transforms for the neighbors of a voxel. As an example of the $< 3, 4, 5 >$ metric, for a voxel (p, DT_p) the minimum sum of the the neighbors distance transforms would be $6 * (DT_p - 3) + 12 * (DT_p - 4) + 8 * (DT_p - 5)$, giving a minimum mean $minMNT_p = DT_p - 4.077$. Therefore, the maximum value of TP for the $< 3, 4, 5 >$ metric is 4.077. Similarly, the minimum TP can be computed by considering the maximum MNT_p. Such a condition occurs when p is part of a cluster of equal valued local maxima. For practical purposes, the minimum value of TP can be assumed to be zero, which occurs when all neighbors of a voxel have a distance transform equal to the voxel's distance transform.

The parameter controlled skeletonization method outlined above thins the volume, keeping only the voxels that satisfy **Condition 1**. It requires three passes over the skeletal voxels, two for computing the distance transform and one pass to identify the skeletal voxels. The complexity is therefore $O(N)$, where N is the number of object-voxels. 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. Applications that require a connected representation can further process the small set of skeletal voxels in a manner desired for the application. Centerline construction for automatic navigation is one such application.

2.3 Constructing the Centerline

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. A centerline description is also useful for animation control. 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. Such a strategy enables the user to accurately pick end-points near the center since the skeletal voxels are already centered with respect to the object boundary. 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 visible as potential navigation paths.

2.4 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 $q \in \mathbf{SK}$ such that
        distance (q, p_m) is minimum for all $q \in \mathbf{SK}$
    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 occurring in medical applications. Since it uses the closest points, it could get perturbed by small "hairs" in the skeleton. A better method which refines the "fineness" value as the recursion proceeds has also been implemented which shows a better tolerance to small spikes. Various other strategies to connect the set of skeletal voxels can also be used. The skeletal voxels can be considered vertices of a strongly connected undirected weighted graph. The edge weight is a linear combination of the spatial proximity and difference in distance transform of the voxels. A minimum spanning tree (MST) [7] of this graph connects skeletal voxels which are close to each other and have similar distance transform values. An example of the MST is shown in the next section.

2.5 Reconstruction and Ball Growing

In order to reconstruct the object from the skeletal voxels, balls of radius equal to the distance transform value have to be constructed, with their center at each of the skeletal voxels. A recursive strategy is adopted to implement this. A ball growing path is initiated from every skeletal voxel and starts with a value equal to the distance transform at that voxel. Every move to a neighboring voxel incurs a cost of either 3, 4 or 5 for face, edge and vertex neighbors respectively, and the path value is decremented by the cost. The neighboring voxels then serve as starting points for new ball growing paths with the decremented values. The ball growing function is invoked on each of the neighboring voxels using the decremented path value. A path terminates when its value is less than or equal to 3 but greater than zero. All voxels along every ball growing path are inserted into the reconstructed object. The quality of reconstruction depends upon the thinness of the skeleton. There can be a loss of boundary voxels when reconstruction is done from a very thin skeleton.

3 Results and Applications

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

Figure 6a shows a 9x9x9 cube. The skeleton is shown in Figure 6b for a thinness value of 2.0. Using the skeletal points, the object is reconstructed as shown in Figure 6c.

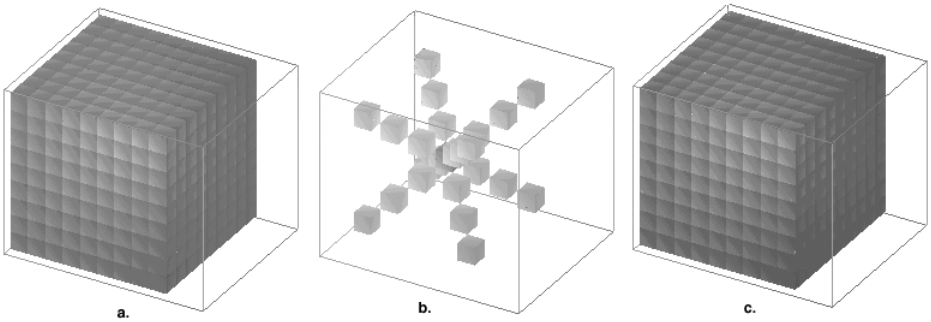

Figure 6: Lossless reconstruction

Figure 7a is a digitized cylinder with height 12 and radius 6. The skeleton in Figure 7b has been extracted using a thinness value of 1.6. Figure 7c shows the lossy 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 8 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. From this figure, it is clear that high thinness values like 2.0 can be used to generate very thin skeletons, as required in centerline generation. Applications which need more shape information would need a thicker skeleton which can be obtained with thinness values of 1.5 or lower. The various skeletons shown could also be used to get varying degrees of reconstruction.

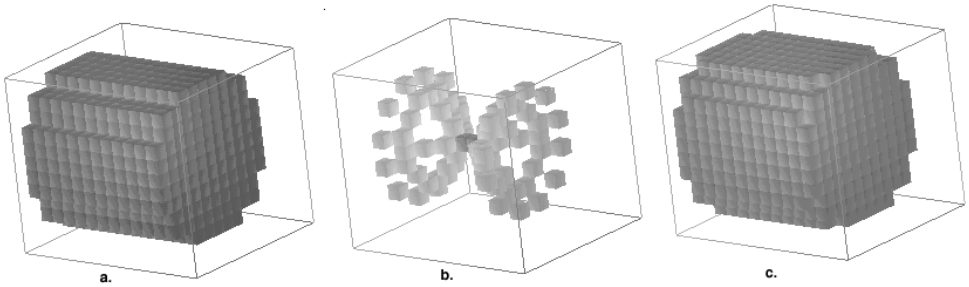

Figure 7: Lossy reconstruction

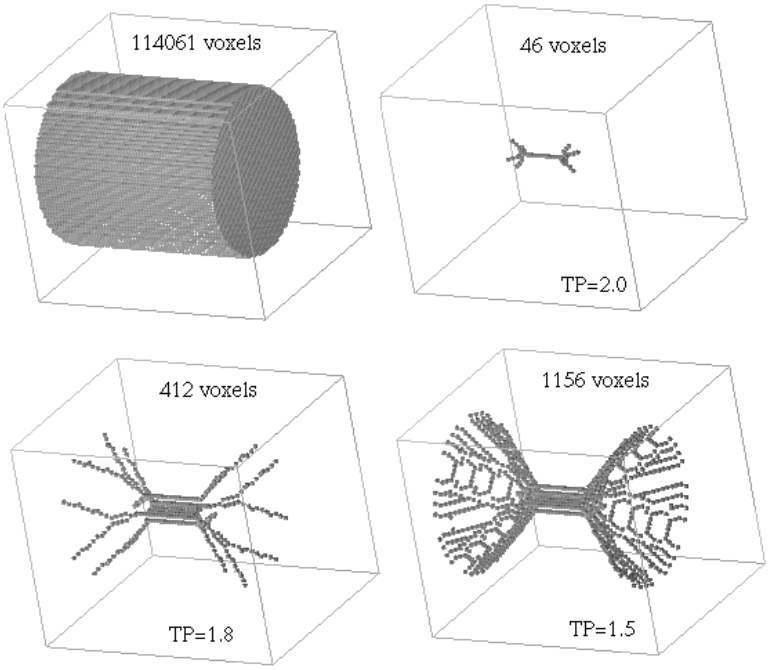

Figure 8: Effect of the Thinness Parameter

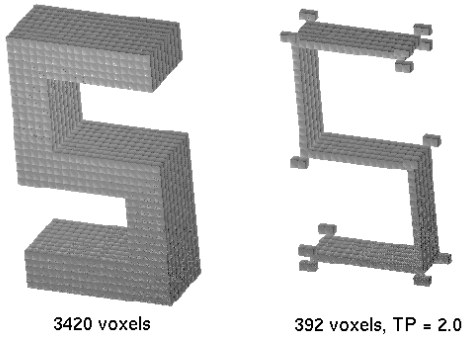

Figure 9: Skeleton of S shape

The result of applying our algorithm to an S-shaped object constructed from cuboidal elements is shown in Figure 9. A thinness parameter of 2.0 was used for this example.

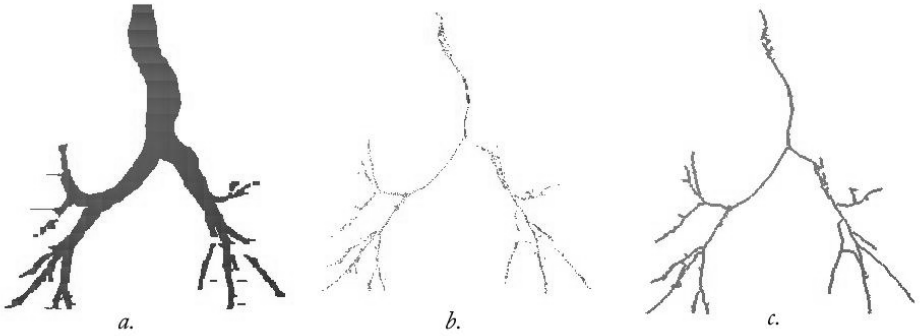

Figure 10: MRI dataset, 512x512x281 of the trachea. (a) The segmented trachea ($\tilde{1}$49k voxels). (b) Skeleton with a thinness = 2.5. (c) The minimum spanning tree of the skeletal voxels.

3.1 Centerlines for Surgical Navigation

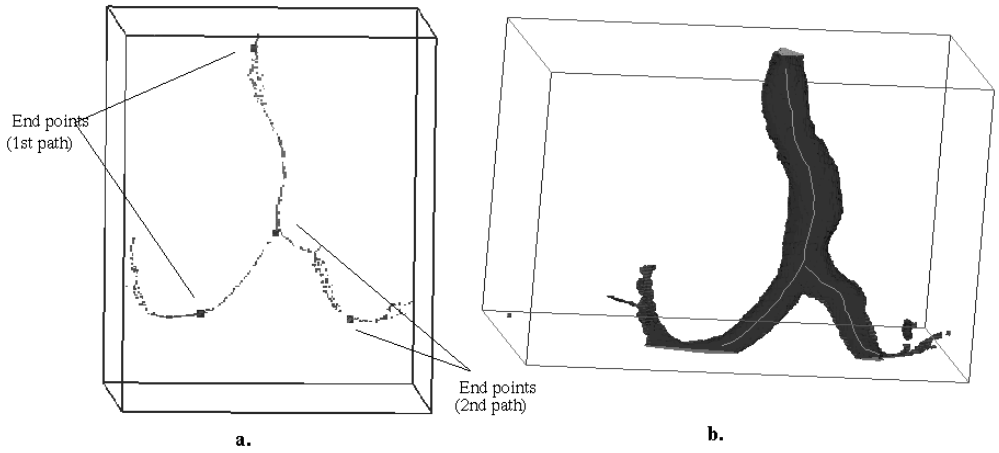

Figure 11: Interactive generation of the trachea centerline. (a) Two endpoints are defined for a navigation path. (b) A centerline is generated for each path.

We demonstrate our centerline algorithm on the segmented human trachea 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 Figures 10a,b. The skeleton captures all the bifurcations in the object and includes the disjoint fragments of the object. The disjoint parts of the skeleton can be connected by using a minimum spanning tree as shown in Figure 10c.

Two different centerlines are generated, with the user-defined end-points as indicated in Figure 11a. The trachea with the centerline inside it can be seen in Figure 11b. In Figure 12 camera shots taken from four different points along the centerline can be seen. The camera is looking downwards into the bifurcation of the trachea. The inset legend indicates the camera position for each of the views.

3.2 Applications in Feature Tracking

In [26], 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 time-steps 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 the matching. The number of voxels

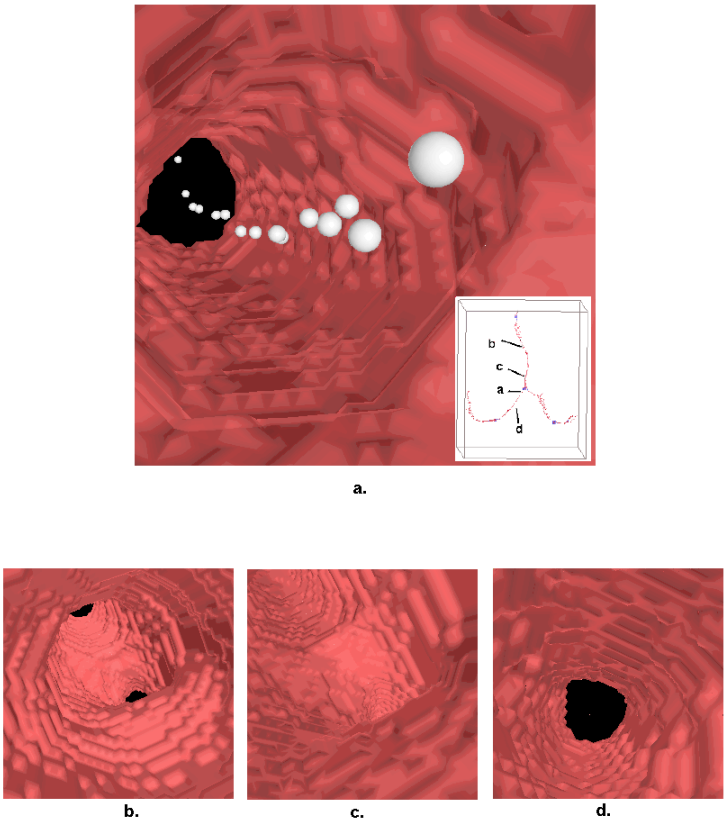

Figure 12: 3D trachea dataset, Camera views from different points along the Trachea centerline, (shown inset in (a)). Points on the camera path are shown as spheres in (a). Note the visible bifurcation in (b).

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 30 time-steps, 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

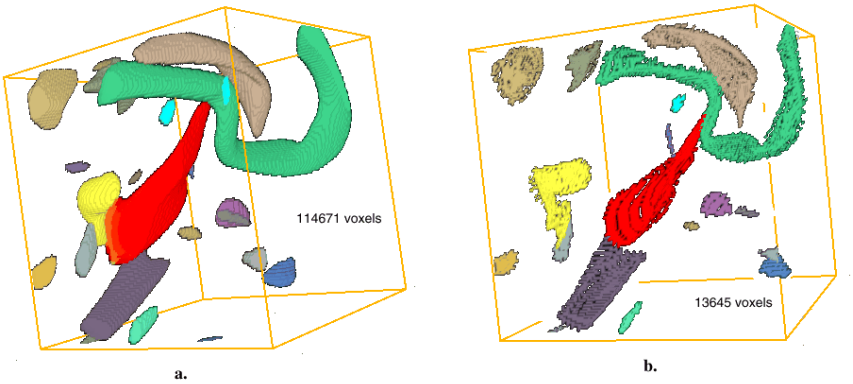

Figure 13: Skeletons for fast Vortex Tracking. (a) Segmented vortex structures.(b) Skeleton of these structures, thinness = 1.0

Thinness	% Error	Skel. Time (sec)	Tracking time (sec)	% Total Time
None	0	0	437.00	100
0.5	4.32	314.64	109.25	97.00
1.0	5.15	300.23	69.30	84.56
2.0	14.50	272.08	22.45	67.40

Table 1: Error and speedup for feature tracking vortices in a fluid dynamics simulation. Skeletons of the vortex shapes have between 1% and 15% of the voxels of the original shapes.

and results are tabulated in Table 1. The percentage total times shown in the table are for the combined skeletonization and tracking process. A significant speedup is observed when the skeletal points are tracked. The time for skeletonization can be amortized if several tracking runs need to be performed. The speedup achieved by tracking the skeletons is also observed to be greater as the number of time-steps increase. The skeleton of a typical dataset being tracked is shown in Figure 13. A thinness value of 1.0 was used for this figure.

4 Conclusion and Future Work

Three dimensional skeletons have a variety of applications, each of which requires a different density of the skeletal structure. Current methods are specific to the application that they address and most do not work well with complex shapes. In this paper, we have presented a versatile and fast algorithm to extract the set of skeletal points from a volumetric object.

Since it involves two passes to compute the distance transform and one pass to extract the skeletal points, the algorithm runs in linear time with respect to the number of object voxels. The algorithm automatically favors reconstruction over connectivity because of a formulation based on the distance transform. However, by using a low value of the thinness parameter, a connected skeleton can be obtained for applications that desire connectivity.

We have also described a simple method to construct the centerline curve from the skeletal voxels. The method is fast and captures all the shape properties of the object. Since skeletal points are identified in one pass, that step is easily parallelizable. Therefore, once the distance transform has been computed, several skeletons with different thinness values can be rapidly generated.

The relation between the thinness parameter and reconstruction coverage needs to be investigated. The range of the thinness parameter is uniquely determined based on the distance metric. A normalized thinness parameter which has a fixed range irrespective of the distance metric would be useful. We also hope to apply the parameter controlled thinning technique to various application domains.

References

[1] 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.

[2] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[3] G. Bertrand. A Boolean Characterization of Three-Dimensional Simple Points. *Pattern Recognition Letters*, 17(2):115–124, 1996.

[4] G. Borgefors. On Digital Distance Transforms in Three Dimensions. *Computer Vision and Image Understanding*, 64(3):368–376, November 1996.

[5] J. W. Brandt and V. R. Algazi. Continuous Skeleton Computation by Voronoi Diagram. *CVGIP: Image Understanding*, 55(3):329–338, May 1992.

[6] N. Burtnyk and M. Wein. Interactive Skeleton Techniques for Enhancing Motion Dynamics in Key Frame Animation. *Communications of the ACM*, 19:564–569, October 1976.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to algorithms*. MIT Press and McGraw-Hill Book Company, 6th edition, 1992.

[8] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*. Springer-Verlag, Berlin, 1987.

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

[10] 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.

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

[12] F. Leymarie and M. D. Levine. Fast Raster Scan Distance Propagation on the Discrete Rectangular Lattice. *CVGIP : Image Understanding*, 55(1):84–94, January 1992.

[13] W. E. Lorensen and H. E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. *ACM Siggraph Proceedings*, 21(4):163–166, July 1987.

[14] 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.

[15] M. Naf, G. Szekely, R. Kikinis, M.E Shenton, and O. Kubler. 3D Voronoi Skeletons and Their Usage for the Characterization and Recognition of 3D Organ Shape. *Computer Vision and Image Understanding*, 66(2):147–161, 1997.

[16] 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.

[17] F. Nilsson and P-E. Danielsson. Finding the Minimal Set of Maximum Disks for Binary Objects. *Graphical Models and Image Processing*, 59(1):55–60, January 1997.

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

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

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

[21] 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.

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

[23] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, Massachusetts, 1989.

[24] 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.

[25] 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.

[26] D. Silver and X. Wang. Tracking and Visualizing Turbulent 3D Features. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):129–141, June 1997.