The Visual Computer

Three-dimensional skeleton and centerline generation based on an approximate minimum distance field

Yong Zhou[1, 2], Arie Kaufman[2], Arthur W. Toga[1]

[1] Laboratory of Neuro Imaging, UCLA School of Medicine, 710 Westwood Plz, RM 4–238 Reed, Los Angeles, CA 90024–1769, USA
[2] Center for Visual Computing and Department of Computer Science, SUNY at Stony Brook, Stony Brook, NY 11794–4400, USA
E-mail: yzhou, toga@loni.ucla.edu
ari@cs.sunysb.edu

We propose an algorithm for generating 18-connected skeletons and centerlines of 3D binary volume data sets. With of an approximate minimum distance field, we express skeletons as a set of clusters with a set of local maximum paths (LMpaths). Each cluster consists of geometrically adjacent voxels with the same local maximum value. Distinct clusters are connected by all possible LMpaths formed by local maximum voxels snaking along, at most, three fixed directions until they meet other clusters. As a 3D extension, we discuss an LMpath traveling on a straight line before and after reaching a saddle point. We generate the shortest centerline connecting two given points with another similar minimum field over skeletal point sets. The results generated by the algorithms on an experimental data set and colon CT and brain MRI data sets demonstrate their efficiency.

Key words: 3D skeleton and centerline – Volume visualization – Navigation – Distance transformation

Correspondence to: Y. Zhou

1 Introduction

Due to their compact shape representations, skeletons have been studied for over 30 years in computer vision and pattern recognition (Blum 1964, 1973; Rosenfeld and Pfaltz 1996). Skeletonization is a powerful tool for intermediate representations for a number of geometric operations on solid models (Sheehy et al. 1996; Sherbrooke et al. 1996; Sudhalkar et al. 1996; Turkiyyah et al. 1997; Zhu and Yuille 1996). As a popular topic in the visualization area, skeleton research has been a recent highlight (Gagvani 1997; Helman and Hesselink 1991; Hong et al. 1995; Itoh et al. 1996) providing helpful cues for visualization applications, such as feature tracking (Silver and Wang 1996), curve and surface generation (Itoh et al. 1996; Kubler 1992; Mangin 1994), or automatic navigation (Gagvani 1997; Paik et al. 1998). Skeletons aid physicians and scientists in precisely exploring virtual human body organs with noninvasive techniques (Hong et al. 1995; Paik et al. 1998). We have successfully explored a walkthrough of the human colon along a calculated path, i.e., a centerline, providing assistance in the detection of possible signs of disease (Hong et al. 1997). In this paper, we propose an efficient algorithm for extracting the skeleton and centerline from 3D binary volume data sets. We apply it to image data describing the human colon and brain. The skeleton of an object is, conceptually, defined as the locus of centers of maximal 2D disks or 3D balls contained in the objects (Blum 1967, 1973). However, a more specific and strict definition of the skeleton is surprisingly difficult. According to our application requirements, we define a skeleton of an object as a subset, SK, of the original data volume where (1) SK is geometrically centered within the boundaries, (2) SK is connected, and (3) SK is as thin as possible. On the basis of skeleton definition, a centerline is defined as an ordered sequence of voxels, $\{p_1, p_2, \ldots, p_n\}$ so that each p_i is a skeletal point and any two voxels p_i and p_{i+1} are connected.

Although some applications require skeletons to provide the possibility of reconstructing objects with an approximation error, here we are concerned with the eventual generation of a centerline. Briefly, the definition concerns three issues: *centered*, *connected*, and *thin* – basic requirements for a skeleton. However, the explanation for each differs in the published literature. First, different distance transforms are used to evaluate the degree

The Visual Computer (1998) 14:303–314

to which a voxel is centered. Since our application deals with large volume data sets, there are no strict requirements for centerlines. Thus, we choose a simple transform mechanism to solve the *centered* problem. Secondly, connectivity assumes the skeleton has the same number of components as the object. To be precise, the connectivity refers to the traditional 18-connectedness. Finally, for a skeleton that is as thin as possible, we provide criteria for a skeleton search under the precondition of connectivity.

A variety of methods for skeleton extraction have been proposed in the literature. Most methods can be allotted to one of three classes: *boundary peeling* (also called thinning, erosion, etc.), *distance coding* (distance transform), and *polygon-based Voronoi methods*. The essence of the first class (Mukerjee et al. 1989; Pavlidis 1980; Sudhalkar 1996) is to iteratively peel off the boundary layer by layer, i.e., to identify simple points, whose removal does not affect the topology of the object. This is a time-intensive process, since to detect a simple point is not trivial. Its advantage is that it can easily be implemented in a parallel way (Tsao and Fu 1983; Ma and Sonka 1996). The result guarantees connectivity. In contrast, distance coding methods (Borgefors 1986; Niblack et al. 1992) try to directly extract skeletal points by testing their local neighborhood via a distance transform that is an approximation to the euclidean distance. Our work belongs to this class. The Voronoi methods (Brandt and Algazi 1992) are mainly designed for the extraction and representation of medial axis of geometrically continuous objects.

The ideal distance-coding-based method requires these three steps: (1) approximate the minimum distance field, (2) detect all local maxima in terms of distance value, and (3) reconnect the local maxima to generate skeletons. The first priority for a skeleton is the issue of centeredness. Theoretically, the euclidean distance of the points provides enough information indicating whether a point is centered; the calculation of the real euclidean distance, however, is expensive. Fortunately, there are many discrete simulation methods, i.e., distance transform metric, represented in computer vision. Those widely used are "2-3" (Dorst 1986) or "3-4" (Borgefors 1986) in 2D cases, "2-3-4" or "3-4-5" in 3D cases. Consider metric "3-4-5" as an example – if a voxel has a distance value of 1, voxels sharing a face with it are assigned a value of 3; voxels sharing an edge have a value of 4; voxels sharing a vertex have a value of 5. The main problem involved in transform-based methods is the connectivity. Niblack et al. (1992) provide a solution to this problem. In order to connect local maxima by uphill climbing rules, they add saddle points, which are local minima along a skeleton, into a skeletal point set. As Gagvani (1997) notes, a direct application of their algorithms to three dimensions is difficult since there is no a unique cyclic sequence of voxels around a given voxel. Helman and Hesselink (1991) detect saddle points in a 2D vector field by computing the eigenvalues of the Jacobian matrix of the field. In this instance, the accuracy of the saddle points depends on the accuracy of the vector field generated from a distance transform. Essentially, both methods are sensitive since the detection of saddle points only considers the local distribution of the distance transform. In three dimensions, noisy, complex data might have too many voxels taken as saddle points, resulting in an incorrect path as a part of the skeletons, or very thick skeletons. We require thin skeletons for centerline generation.

Voronoi methods (Brandt and Algazi 1992; Saito and Toriwaki 1994), theoretically, guarantee connected skeletons, which are best suited for polygonally defined objects (Sheehy et al. 1996; Sherbrooke et al. 1996; Turkiyyah et al. 1997) in solid modeling. However, objects in scientific visualization often have noisy boundaries that cause the Voronoi Diagram to be very dense, requiring appropriate pruning to generate the medial axes.

In this paper, we introduce a 3D skeleton algorithm that follows the same procedures as that of Niblack et al. (1992). Yet, we adopt a simpler distance coding to approximate the euclidean distance and use a global method to detect saddle points. A saddle point is associated with a *local maximum path* (LMpath). Both are computed in the same process. Other contributions of the paper include a strict definition and description of LMpath properties, a proof of computation of saddle points on a LMpath, and a method for fast generation of the shortest centerline.

The remainder of the paper is organized as follows. In Sect. 2, we introduce the concept and application of an approximate minimum distance field, analyze the requirements of skeletal points, then give an explicit definition of a local maximum path

and describe its properties. Finally, we provide a basic algorithm for skeleton generation. The centerline generation method is discussed in Sect. 3. Section 4 gives implementation details and results, and we conclude in Sect. 5.

2 Skeleton extraction

We work with 3D binary volume data sets, considered to be uniformly sampled in all three dimensions. A voxel is the smallest unit cube in the volume, with eight nodes taking values of zero or one. A voxel is regarded as an *inside voxel* if all its nodes take a value of one, as a *outside voxel* if all its nodes take a value of zero, and otherwise, as a *boundary voxel*. Outside voxels have sometimes been called background voxels (Niblack et al. 1992). Both boundary voxels and inside voxels are called object voxels. Following the same notations as Gagvani (1997), for a voxel p, an adjacent voxel q is called a *F-neighbor*, *E-neighbor*, or *V-neighbor* of p if it shares a face, an edge, or a node, respectively, with voxel p. Voxels p and q are also called *F-connected*, *E-connected*, or *V-connected*, corresponding respectively to traditional 6-connected, 18-connected, or 26-connected voxels. In the present algorithm, a sequence of voxels is E-connected or connected if any adjacent two of its voxels are E-connected. A set of voxels (with at least two elements) are regarded as connected if, for any two voxels, there is an E-connected sequence of voxels to connect them. For conformity with traditional expression, the words *voxel* and *point* are regarded as having an interchangeable meaning (thus a node is different from a point).

The goal of this section is to extract skeletons – a set SK of inside voxels – that satisfies the centeredness, connectedness, and thinness features mentioned in Sect. 1. Our algorithm strategy consists of the following three stages:

1. Minimum distance approximation
2. Cluster generation
3. Cluster connection.

Definition 1 A *cluster* is defined as a set of E-connected local maximum voxels with the same distance value. A *local maximum voxel* has a distance value not less than those of all its F-neighbors.

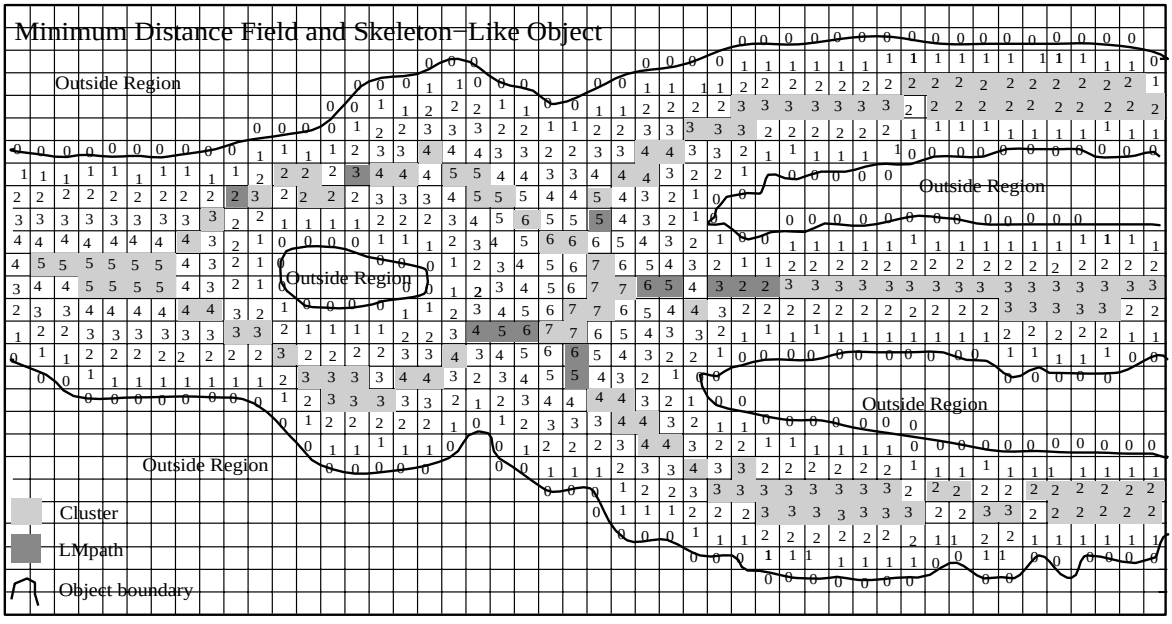
2.1 Minimum distance field

A minimum distance field has many applications, such as in offset surface construction (Payne and Toga 1992) or navigation (Hong et al. 1997). In a minimum distance field, for each voxel, the minimum value of distances to all boundary voxels is given. Theoretically, the distance should be the euclidean distance, but computation is expensive. Although many discrete methods have been proposed to solve this problem, our strategy is to find a simple and efficient method, which is not necessarily initially correct. Thus, the method is as follows. First, for boundary voxels, the distance values are assigned zero; then for all the F-neighbors of the boundary voxels, the values are assigned one. Suppose that voxels with a value of n are processed iteratively. All their F-neighbors are given a value of $n + 1$. This coding process continues until all the voxels are visited. The accuracy of this approach is less than "3-4-5" transform metric (Gagvani 1997) and other methods. However, there are several reasons why this processing is useful. First, the data from medical applications generated by computed tomography (CT) and magnetic resonance imaging (MRI) equipment are usually large, so that the precise calculation of distances for skeletons and centerlines is expensive. Second, if a precise simulation is required, a revoxelized volume with a higher resolution can be used for the coding. Finally, this coding strategy has several advantages over other methods in skeleton searches, which will be discussed later. Figure 1 shows a portion of the minimum distance field in two dimensions.

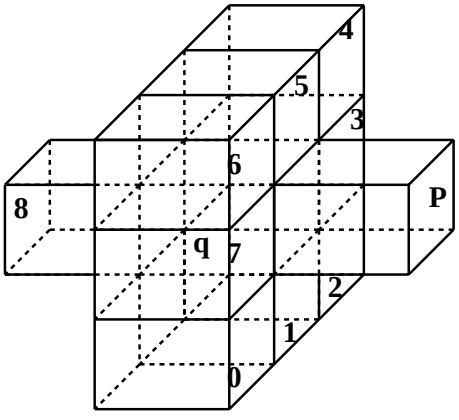
In addition to the skeleton search, the minimum distance field also can be used to search for the shortest path in a voxelized environment. After the skeleton is obtained, the same coding techniques are applied on a skeletal point set for centerline extraction.

2.2 Skeletal points

On the basis of the minimum distance field, we further analyze the properties that skeletal points should meet. *Skeletal points* or *skeletal voxels* are voxels of the skeletons. Before giving more constraints on skeletal points, we need to introduce several concepts. A voxel taking a value of n is



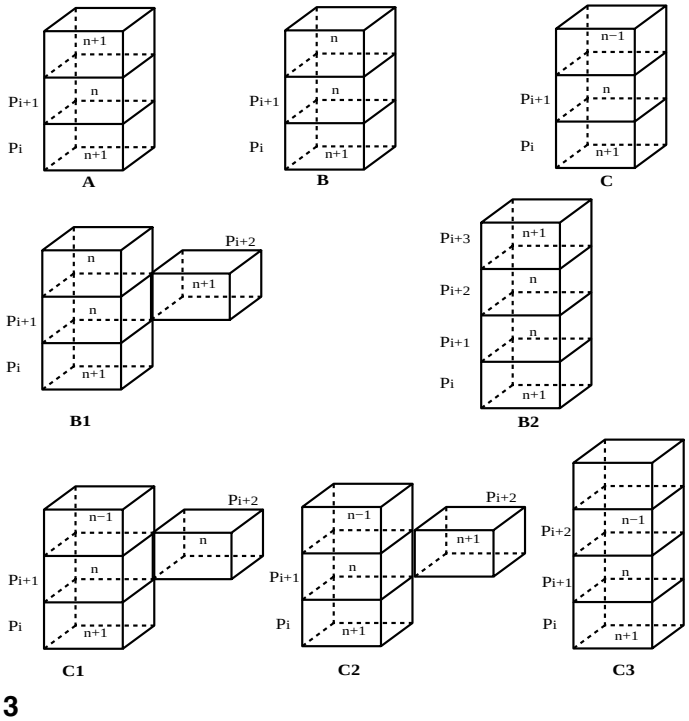
1



2

Fig. 1. Minimum distance field and skeleton-like object

Fig. 2. Relationship between indirect F-neighbor and F-neighbor: voxel 8 is an indirect F-neighbor of voxel *q* relative to voxel *p*; voxels 1, 3, 5 and 7 are direct F-neighbors of *q* relative to *p*; voxels 0, 2, 4 and 6 are indirect E-neighbors of *q* relative to *p*



3

Fig. 3. The cases for path generation

also called a *n-voxel*. Furthermore, if all F-neighbors of a n-voxel p take values not greater than n, the n-voxel p is referred to as a *noncontributor* (n is a nonzero integer). In our algorithm, noncontributors and local maxima have the same meaning, differing from the definition of a local maximum used in other literature (Gagvani 1997) where the comparison with adjacent V-neighbors is also considered. A voxel p is referred to as a *partialcontributor* if it only has skeletal voxels as its F-neighbor with a value no less than p. The remaining voxels, which do not belong to these two classes, are called *contributors*. For simplification, noncontributor, partialcontributor and contributor are denoted by *N-voxel*, *P-voxel* and *C-voxel*, respectively. In addition to the skeletal point set, SK (discussed in the Introduction), abbreviations the NV (the set of N-voxels) and PV (the set of P-voxels) are also used.

At this stage of the algorithm, we have the N-voxel set, NV belonging to the skeleton set SK. The set NV consists of clusters. By definition, each cluster is connected, but usually two clusters are disconnected, although they usually form skeletonlike objects. Figure 1 illustrates this situation. Now the challenge is to reconnect the clusters to produce skeletons, which requires finding all the possible paths, each of which connects two clusters. A definition for such a path, or LMpath, is:

Definition 2. A *LMpath* $Q = \{p_i | \, i = 1,\ldots, m\}$ is a sequence of voxels such that:

1. Each p_i has at least one F-neighbor with a distance greater than that of p_i.
2. Voxels p_1 and p_m are at least E-neighbors of N-voxels in clusters.
3. p_i and $p_{i+1} (i = 1,\ldots, m-1)$ are at least E-connected, and p_i and p_{i+2} are disconnected $(i = 1,\ldots, m-2)$.
4. Any F-neighbor of p_i that does not belong to $Q \biguplus \mathrm{NV}$ must have distance values not greater than that of p_i.
5. p_{i+2}'s value cannot be less than the values of p_{i+1}'s E/F-neighbors that are disconnected with p_i.

The requirements for a LMpath are not complex. Since LMpaths are designed to be a part of the skeletons, adjacent voxels in a LMpath must at least be E-connected for connectivity; furthermore, each voxel p_i in the LMpath must have an F-neighbor with a value greater than that of p_i. The third requirement guarantees that the LMpath is as thin as possible. The last requirement guarantees that the next voxel in a LMpath is the maximum among all possible next voxels. Note that the last two requirements are for skeleton centeredness. The definition is sequential in the sense that one element belongs to skeletal points depending on whether the next element falls in skeletal sets, leading to an iterative search. Fortunately, according to our distance coding, LMpaths possess properties that simplify the search process. For comparison with methods previously mentioned (Niblack et al. 1992), we now introduce the definition of 3D saddle points.

Definition 3. A *voxel* is a saddle point if there is a LMpath such that it belongs to the LMpath and is a local minimum in the LMpath.

If a saddle point is a local minimum, then the preceding and succeeding voxels in the LMpath have distance values no less than that of the saddle point. In order to simplify the LMpath search, we discuss LMpath properties in Sect. 2.3.

According to this analysis, the generated skeleton can be expressed in the following equation.

$$\mathrm{SK} = \sum_{i=0}^{m} \mathrm{Clus}[i] + \sum_{i=0}^{n} \mathrm{LM\,path}[i]. \qquad (1)$$

Here Clus[i] and LMpath[i], respectively, indicate i-th cluster and i-th LMpath.

2.3 LMpath properties

Definition 4. *A connected sequence of voxels* $\{q_i\}$ *has a corner* if there are two voxels p_i and p_j $(i \neq j)$ in the sequence so that they are only E-neighbors of each other.

Definition 5. Suppose voxel q is an F-neighbor of voxel p. *q's unique F-neighbor* that *is not an E-neighbor of p* is called q's indirect F-neighbor relative to p, denoted by $F_i(q \mid p)$; *q's F-neighbors* that *are also E-neighbors of p* are called q's direct F-neighbor relative to p, denoted by $F_d(q \mid p)$; *q's E-neighbors that are not E-neighbors of p* are called q's indirect E-neighbors relative to p; their

set is denoted by $E(q \mid p)$. Figure 2 illustrates the relationship among voxels p, q and q's F-neighbors and E-neighbors.

If we let $Q = \{p_i \mid i = 1,\ldots, m\}$ as a LMpath, and let $d(p_i)$ be the distance value of voxel p_i, we have the following results.

Lemma 1. *If p_{i+1} is an F-neighbor of p_i, then p_{i+2} must be an indirect F-neighbor or indirect E-neighbor of p_{i+1} relative to p_i.*

Proof. According to Definition 2, p_{i+2} is connected with p_{i+1}, and p_{i+2} is not an E-neighbor of p_i, thus the result holds.

Lemma 2. *If $d(p_i)<d(p_{i+1})$, then $d(p_{i+1}) = d(p_i) + 1$.*

Proof. Since p_i and p_{i+1} are adjacent in Q, p_{i+1} is either an E-neighbor or F-neighbor of p_i. If p_{i+1} is an F-neighbor, according to the minimum distance transform criteria, the result holds. If p_{i+1} is an E-neighbor, there must be a common F-neighbor p of p_i and p_{i+1}, according to Definition 1(4),

$$d(p_i) - 1 \le d(p) \le d(p_i) \tag{2}$$

$$d(p_{i+1}) - 1 \le d(p) \le d(p_{i+1}); \tag{3}$$

thus, $d(p_{i+1}) - 1 \le d(p_i) < d(p_{i+1})$. Further, $d(p_i)$ and $d(p_{i+1})$ are integers; these lead to the result.

Lemma 3. If $d(p_i) \le d(p_{i+1})$, and p_{i+1} is an F-neighbor of p_i, then p_{i+2} is the indirect F-neighbor of p_{i+1} relative to p_i with $d(p_{i+2}) = d(p_{i+1}) + 1$.

Proof. According to Lemma 1, p_{i+2} is either an indirect E-neighbor or indirect F-neighbor of p_{i+1} relative to p_i. First we prove p_{i+2} cannot be an indirect E-neighbor. If p_{i+2} is an indirect E-neighbor, note that p_{i+1} is not a N-voxel. There is an F-neighbor q with $d(q) > d(p_{i+1})$; this conflicts with Definition 2 (4). Thus p_{i+2} must be an indirect F-neighbor, and according to Lemma 2, we have $d(p_{i+2}) = d(p_{i+1})+1$. Thus the results hold.

Note that the results of Lemma 3 also hold if p_i is an N-voxel, $d(p_i) = d(p_{i+1})$, and p_{i+1} is an F-neighbor of p_i. Geometrically, Lemma 3 means that the subsequence $\{p_i, p_{i+1}, p_{i+2}\}$ has no corner.

Theorem 1. Given a LMpath $Q = \{p_i \mid i = 1,\ldots, m\}$, its first element p_1 or last element p_m must be an F-neighbor of a N-voxel.

Proof. First, suppose p_1 is not an F-neighbor of any N-voxel. By Definition 2 (2), p_1 is a E-neighbor of a N-voxel, and p_2 must be its F-neighbor with $d(p_2) > d(p_1)$; otherwise, p_1 will be a N-voxel. According to Lemma 3, p_3 must be an F-neighbor of p_2 with $d(p_3) > d(p_2)$. Following this derivation, p_m must be an F-neighbor of p_{m-1} with $d(p_m) > d(p_{m-1})$. Since p_m is not a N-voxel, p_m must have an F-neighbor as N-voxel. However, if p_m is not an F-neighbor of any N-voxel, for the same reason, p_1 will be.

This theorem guarantees that the search for LMpaths starting with F-neighbors of N-voxels suffices; we do not need to start the search from already generated LMpaths.

Lemma 4. *If e and f are, respectively, E-neighbor and F-neighbor of p_i with $d(e) = d(f) = d(p_i) + 1$, and e and f are disconnected from p_{i-1}, then p_{i+1} must be f, not e.*

Proof. According to Definition 2(5), p_{i+1} must be a voxel with maximum from E/F-neighbors of p_i. According to Lemma 2, the maximum cannot exceed $d(p_i) + 1$, thus p_{i+1} comes from e or f. If p_{i+1} is e, p_i will violate Definition 2(4), thus p_{i+1} must be f.

Lemma 5. *If $d(p_i) > d(p_{i+1})$, and p_{i+1} is p_i's F-neighbor, then p_{i+3} is an F-neighbor of p_{i+2} with $d(p_{i+3}) = d(p_{i+2}) + 1$, and p_{i+2} meets one of the following results:*

Case 1. $p_{i+2} = F_i(p_{i+1} \mid p_i)$, if $\forall r,\ r \in E(p_{i+1} \mid p_i)$, $d(F_i(p_{i+1} \mid p_i) \ge d(r)$.

Case 2. $p_{i+2} = r$, if $r\ r \in E(p_{i+1} \mid p_i)$, $d(F_i(p_{i+1} \mid p_i) < d(r)$.

Proof. Let $q = F_i(p_{i+1} \mid p_i)$. q has three cases: (A) $d(q) > d(p_{i+1})$, (B) $d(q) = d(p_{i+1})$, (C) $d(q) < d(p_{i+1})$. First, for case A, according to Lemma 4 and Definition 2(5), $p_{i+2} = q$. In cases B and C, according to Definition 2(5), the results for p_{i+2} hold. Further, we prove the results for p_{i+3}. p_{i+2} is not an N-voxel; it has at least one F-neighbor with a value greater than that of p_{i+2}. According to Lemma 4, the results for p_{i+3} hold.

Figure 3 shows three possible cases according to the value of $d(q)$. Additionally, case B is further divided into two subcases: cases B1 and B2, while case C is divided into three subcases C1–C3. In all these subcases, remember that voxels p_i and

p_{i+1} are assumed to be elements of a LMpath. Thus, all other p_i's F-neighbors that are not illustrated in Fig. 3 have values not greater than $d(p_i)$. The result also holds for p_{i+1}; each subcase shows the choice of p_{i+2}. Cases B1, C1, and C2 meet case 2 in Lemma 5; all other cases meet case 1 in Lemma 5.

Note that Lemma 5 holds, too, if p_i is an N-voxel.

Theorem 2. *A LMpath* $Q = \{p_i \in \mathrm{PV} \mid i = 1,\ldots, m\}$ *has at most one corner, and the corner must be the only saddle point.*

Proof. According to Theorem 1, let p_1 be an F-neighbor of a N-voxel p_0. There are two cases: (A) $d(p_0) = d(p_1)$; (B) $d(p_0) > d(p_1)$. For case A, using Lemma 3 iteratively, neither corner nor saddle point is generated. For case B, according to Lemma 5 and Fig. 3, for the sequence $\{d(p1), d(p2),\ldots, d(p_m)\}$, there is an integer i $(1 \le i \le m-2)$ such that the subsequence $\{d(p_k) \mid k=1,\ldots, i+1\}$ is monotonously decreasing without corners. The subsequence $\{d(p_k) \mid k=i+2,\ldots, m\}$ is monotonously increasing without corners. Furthermore, $d(p_{i+1})$ and $d(p_{i+2})$ take values as follows: (1) n, $n+1$; (2) n, $n+1^*$; (3) n, n; (4) n, n^*; (5) n, $n-1$. Suppose $d(p_{i+1})=n$, and n^* indicates that p_{i+2} is an E-neighbor of p_{i+1} with a value of n, in this case. Thus, we have results that p_{i+1} will be a unique possible saddle point, and the corner will occur between p_{i+1} and p_{i+2}.

According to Theorem 2, corners only occur at voxels with a minimum distance value along a LMpath; i.e., geometrically, a LMpath with no corner means that all the voxels in the sequence are queued on a straight line. In addition, the distance values of the endpoints in a LMpath determine the values and positions of other voxels in the sequence. If the first two elements do not decrease monotonously, the whole sequence increases monotonously. Otherwise, if the first two are monotonously decreasing, the sequence decreases monotonously to the end, or to the saddle point, then increases monotonously. During the increasing or decreasing process, the sequence follows a straight line.

2.4 Basic algorithm

In Eq. 1, the skeleton consists of two sets of voxels. One is the cluster set; the other is the LMpath set. Theorem 2 provides a useful principle for a LMpath search. From Theorem 1, the path search starts with F-neighbors of N-voxels. Each such F-neighbor p_1 of N-voxel p_0 is a candidate starting voxel of a possible LMpath. First, p_1 must meet Definition 2(4), i.e., all $F_d(p_1\} \mid p_0)$ must have values no greater than that of p_1, as a desirable LMpath should be as centered and thin as possible. Second, we divide two cases according to the values of p_0 and p_1: (A) $d(p_0) \le d(p_1)$ and (B) $d(p_0) > d(p_1)$. For case A, according to Lemma 3 and Theorem 2, a possible LMpath must follow a straight line with no saddle point until it meets another cluster. In this case, if there is an indirect E-neighbor or F-neighbor that is an N-voxel, the search succeeds, and we output the path. If there is an indirect E-neighbor with a value greater than the indirect F-neighbor's value, the search fails, and we discard the path.

For case B, our strategy is to convert it to case A if possible. First we use the same steps to obtain the next voxel as in case A. The difference is that, in this process, the generated sequence does not increase monotonously until a voxel p_{i+1} is found that possesses one of three cases, as follows. One occurs when p_{i+1} is a boundary voxel, and the search fails; another occurs when an indirect F-neighbor or an E-neighbor of p_{i+1} is found as an N-voxel, and the search succeeds; and the final case occurs when an indirect E-neighbor q of p_{i+1} is found with a value greater than that of the indirect F-neighbor of p_{i+1}, but q is not an N-voxel. According to Theorem 2, if the already generated sequence $\{p_0, p_1,\ldots, p_{i+1}\}$ can eventually become a part of a LMpath, q must have an F-neighbor p_{i+3} with a p_{i+3} value greater than q. If such a q is found (i.e., a corner occurs between p_{i+1} and q), we take q as p_{i+2}; the case becomes case A. If it is not found, this does not mean the search fails. We need to test all of p_{i+1}'s other F-neighbors. If all the cases fail, the already generated LMpath is discarded.

This procedure, **LMPathSearch**(), is summarized in the pseudo-code of Fig. 4, where NB is the set of nonboundary voxels, $F(p)$ is the set of F-neighbors of voxel p, $d(E(p_1 \mid p_0))$ means all the values of indirect E-neighbors of p_1, likewise, $d(F_d(p_1 \mid p_0))$ means all the values of direct F-neighbors of p_1; *path*[] is an array for storing the path generated, *No* is a counter for the path. Function **CaseASearch**() is for a case A search.

```
LMPathSearch (path) {
for (∀p1∈F (p0), p0∈NV,d (p1)≥d (Fd(p1|p0)) {
   path [No+ +] = p0: path [No + +] = p1;
   if (d (p1)≥d (p0))
      CaseASearch (path.No);
   else
      while ( p1∈N B, d (Fi (p1|p0))≥d (E (p1|p0)){
         path [N o + +] = Fi (p1|p0):
         p0 = p1; p1 = Fi (p1|p0;
      }
      if (∃ (Fd (p1|p0)||E (p1|p0))∈SK)
         OutPut (path);
      else if (p1∈N B) { // corner occurs
         for (∀r∈E (p1|p0), d (r)>d (p1)){
            p0 = p1, p1 = r;
            path [N o + +] = r:
            for (∀q∈F (r), d (q)>d (r)) {
               path [N o + +] = q;
               CaseASearch (path,No – –);
            }
            No – –;
         }
      }
}
}

CaseASearch (path, n){
p0 = path [n–2]; p1 = path [n–1];
do {
   if (∃ (Fd (p1|p0))||E (p1|p0))∈SK)
      Output (path);
   else if (∃r∈E (p1|p0)), d (r)>d (Fi (p1|p0))
      return;
   else {
      p0 = p1; p1 = Fi (p1|p0:
      path [n + +] = p1;
   }
}
}
```

Fig. 4. The algorithm for skeleton extraction

3 Centerline generation

One of the applications of skeletons is for the centerline generation. Given two points a source point p_s and a target point p_t, the problem consists of finding a sequence of voxels CL to connect them so that CL meets the *centered* and *thin* requirements, and its first and last voxels are p_s and p_t, respectively. We solve this in three steps.

Step 1. Find a connected sequence CL1 from p_s to the skeletal voxel closest to p_s, denoted by p_{s1}.

Step 2. Find a connected sequence CL3 from p_t to the skeletal voxel closest to p_t, denoted by p_{t1}.

Step 3. Find a connected sequence CL2 from p_{s1} to p_{t1}.

On the basis of the generated skeleton, Gagvani (1997) proposes using the midpoint search method to adaptively approximate skeletons for centerline generation. Although simple and fast, it is inefficient for control of the approximation accuracy, since in some twisted cases, the midpoints do not approximate to expected points. Also, the generated skeletons are not necessarily connected.

For various segments of the centerline, we use similar methods. Take the search of CL1 as an example. Following the same strategy as in the minimum distance transform, use the voxel p_s as the only "boundary" instead of the real boundary voxels. Then calculate the minimum distances layer by layer until a skeletal voxel, i.e., p_{s1}, is reached. Then, starting from voxel p_{s1}, the algorithm searches for the p_{s1} E/F-neighbor p_1 with the smallest value, followed by searching for a p_1 E/F-neighbor with the smallest value. This process continues until p_s is finally reached. Segment CL3 can be obtained in the same manner.

With the same strategy, we also can find CL2. Now, taking voxel p_{s1} as the only "boundary", we calculate the minimum distance of the voxels layer by layer in SK rather than in the whole data set, until voxel p_{t1} is covered. Then starting with p_{t1}, we find the next connected voxel step by step so that CL1 can also be found.

Figure 5 illustrates the procedure for centerline extraction from the skeleton area generated from the configuration of Fig. 1. The coding scheme starting with the closest point to the source point and the retrieving process from the target point guarantee the shortest centered and connected line. At branching points, voxel values indicate the best choice of next point. For flight path planning for virtual endoscopy, Paik et al. (1998) use a similar coding scheme on the segmented volume rather than on the skeletal region.

4 Implementation and results

Three data sets are used to test our algorithm. One set consists of two cylinders and a torus. The torus is penetrated vertically by these two cylinders with

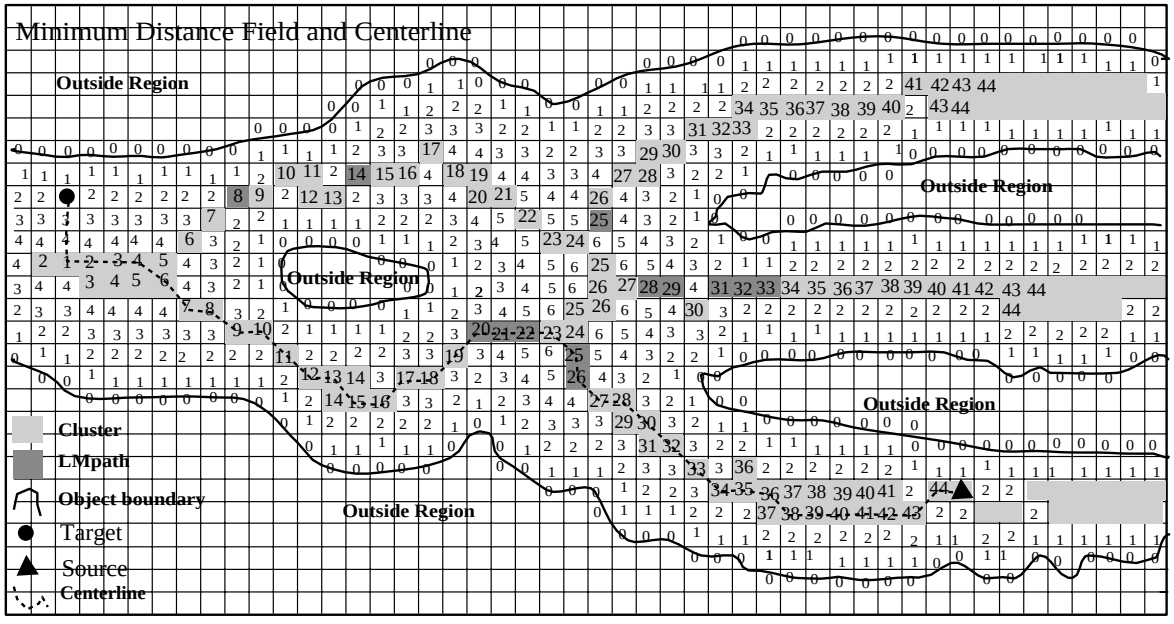


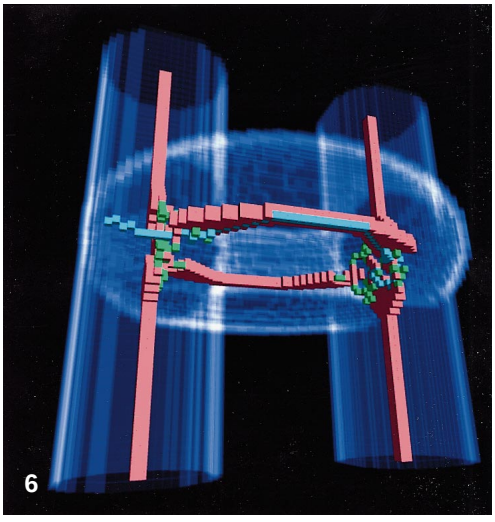
Fig. 5. Voxel coding for generation of the centerline within skeleton areas based on the configuration of Fig. 1

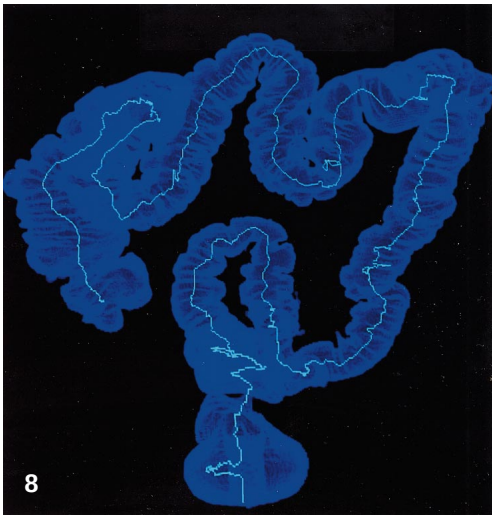
a size of 64^3. The second set is a CT data set showing a whole human colon with a size of $512^2 \times 361$. The third set is a 384×256×383 MRI data set representing the human brain.

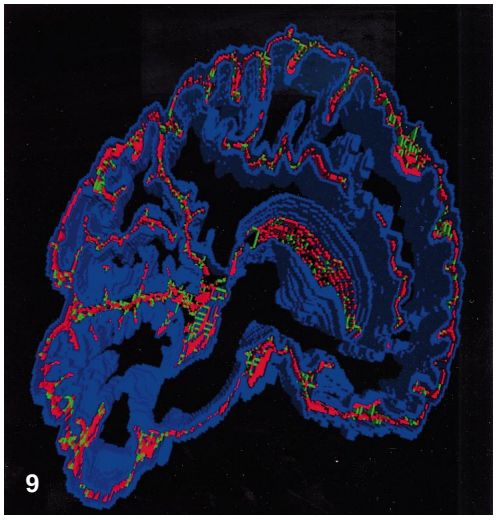
Before the data sets are input, they are preprocessed to generate the inside voxels. The entire implementation includes five sequential steps. Our program starts with reading the inside voxels' data files, followed by detecting the boundary voxels. The second step is a voxel layer-by-layer coding beginning with boundary voxels. Then, in the third step, the clusters are generated by first picking up all local maximum voxels and then collecting all geometrically adjacent voxels as a single cluster. The fourth step generates paths that connect all possible isolated clusters. In this step, before a generated voxel sequence is taken as a path, a check is made to see whether it is a valid path. A candidate path is discarded if (a) the same two clusters are already connected by previously generated paths or (b) the first and the last voxels in the sequence are E-adjacent. Once all possible paths are found, the implementation of skeleton generation ends. If required, a final step is invoked for the extraction of a centerline with a thickness of one voxel, based on previously generated skeletons.

In the following images (Figs. 6–10), the blue represents the boundary voxels, the red represents the clusters, and the green represents the generated paths, while the turquoise indicates the centerlines. All the images are rendered with Open Gl graphics routines. Figure 6 is generated from the first data set (two cylinders and a torus) showing the combination of boundaries, clusters, paths, and the centerline. The image clearly shows the connection of the isolated clusters through green paths. This example also shows the strength of our algorithm for the solution of the branch problem, although the colon data set does not have branches.

The second data set (human colon) is challenging, due to its size, namely 3181745 inside voxels and 7417 clusters found. The calculation from voxel coding to cluster and path generation takes 180 s, while the centerline search takes another 19 s, measured when running on a SGI Power Onyx – R10000 CPU with Infinite Reality graphics. Figures 7 and 8 show corresponding results representing the boundaries, clusters, paths, and a centerline. Figures 9 and 10 illustrate brain results with 2444675 inside voxels and 2799 clusters; the original data sets represent cerebral spinal fluid (CSF) and the gray matter of a brain (see Fig. 9), where







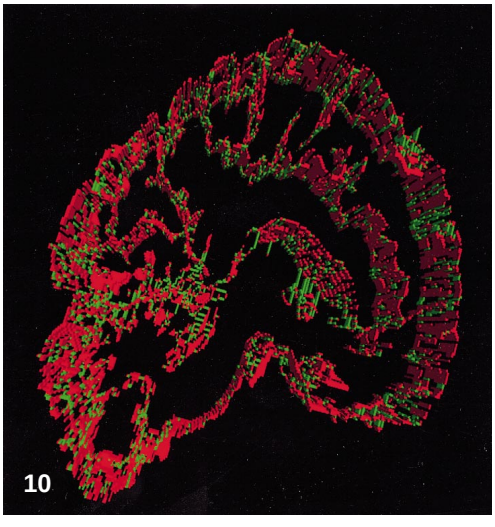


Fig. 6. First data sets: the boundaries, clusters, paths and a centerline

Fig. 7. Colon data set: the boundary, clusters, and paths

Fig. 8. Colon data set: the boundary and centerline

Fig. 9. Brain data set: the boundary, clusters and paths

Fig. 10. Brain data set: clusters and paths

both boundaries and skeletons are displayed. The skeletons indicate the the median surfaces of convoluted ribbons, called sulci, embedded in the brain, separatively displayed in Fig. 10. For clarity, Figs. 9 and 10 only display part of the data set.

5 Conclusions

We have presented distance transform-based skeleton extraction and centerline generation algorithms. Our skeletons consist of clusters and LMpaths. The advantage for introducing the concept of clusters is that the spatial coherence of the same value point is employed for testing whether the cluster is isolated or connected with other clusters. In order to meet the connectivity of skeletons, the definition of a LMpath is given, and its properties are discussed. Also, basic algorithm and implementation details are given. Theoretical proof and experimental results show the efficiency and usefulness of our algorithm. The algorithm complexity is linear. A simple and less accurate distance approximation mechanism is used in the present algorithm, but fortunately, resampling techniques can be applied at a small voxel size to reduce such errors. We are currently applying these techniques to MRI data sets for the comprehensive generation of medial surfaces of cerebral sulci.

Acknowledgements. We thank Dr. L. Hong for sharing his insight of related issues with the first author. This work has been supported by the National Science Foundation under grants CCR-9205047, MIP-9527694 to A.K. and BRI 9322434 to A.W.T., as well as NCRR RR05956 to A.W.T.

References

Blum H (1964) A transformation for extracting new descriptors of shape. Symposium on Models for the Perception of Speech and Visual Form, MIT Press, Cambridge, Mass

Blum H (1973) Biological shape and visual science, Part I. J Theo Biol 38:205–287

Borgefors G (1986) Distance transformations on digital images. Comput Vision Graph Image Processing 34:344–371

Brandt JW, Algazi VR (1992) Continuous skeleton computation by Voronoi diagram. CVGIP: Image Understanding 55:329–338

Dorst L (1986) Pseudo-euclidean skeletons. The 8th International Conference on Pattern Recognition (ICPR), Paris France, Washington, D.C. IEEE Computer Society Press, Los Angeles, CA., pp 286–289

Gagvani N (1997) Skeletons and volume thinning in visualization. Master of Science Thesis, Deptartment of Electrical and Computer Engineering, Rutgers University, New Brunswick, NJ

Helman JL, Hesselink L (1991) Visualization of vector field topology in fluid flows. IEEE Comput Graph Appl 11:36–46

Hong L, Kaufman A, Wei Y, Viswambharn A, Wax M, Liang Z (1995) 3D Virtual colonoscopy. Proceedings of the 1995 Symposium on Biomedical Visualization Atlanta Ga., IEEE Computer Society Press, Los Angeles, CA., pp 26–32

Hong L, Muraki S, Kaufman A, Bartz D, He T (1997) Virtual voyage: interactive navigation in the human colon. SIGGRAPH'97, Los Angeles, Calif., Proceeding of the Computer Graphics 1997 Conference, 27–34

Itoh T, Yamaguchi Y, Koyamada K (1996) Volume thinning for automatic isosurface propagation. IEEE Proceeding of Visualization'96, San Francisco, Calif., Assoc. for Computing Machinery, New York, NY, 303–310

Kubler O, Szekely, Brechbuhler, Ogniewicz, Budinger T, Sander P (1992) Charting the human cerebral cortex. SPIE Math Methods Med Imaging 1768:193–205

Mao CM, Sonka M (1996) A fully parallel 3D thinning algorithm and its applications. Comput Vision Image Understanding 64:420–433

Mangin JF, Frouin V, Bloch I, Regis J, Lopez-Krahe J (1994) Automatic construction of an attributed relational graph representing the cortex topography using homotopic transformations. SPIE Math Methods Med Imaging 2299:110–121

Mukerjee J, Das PP, Chatterji BN (1989) Thinning of 3D images using the safe point thing algorithm (PTA). Patt Recogn Lett 10:167–173

Niblack CW, Gibbons PB, Capson DW (1992) Generating skeletons and centerlines from the distance transform. CVGIP: Graph Models Image Processing 54:420–437

Paik DS, Beaulieu CF, Jeffrey RB, Rubin GD, Napel S (1998) Automated flight path planning for virtual endoscopy. Med Phys 25:629–637

Pavlidis T (1980) A thinning algorithm for discrete binary images. Comput Graph Image Processing 13:142–157

Payne BA, Toga AW (1992) Distance field manipulation of surface models. IEEE Comput Graph Appl 12:65–71

Rosenfeld A, Pfaltz JL (1966) Sequential operations in digital picture processing. J ACM 13:471–494

Saito T, Toriwaki JI (1994) New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications. Patt Recogn 27:1551–1565

Sherbrooke EC, Patrikalakis NM, Brisson E (1996) An algorithm for the medial axis transform of 3D polyhedral solids. IEEE Trans Visualization Comput Graph 2:44–61

Sheehy DJ, Armstrong CG, Robinson DJ (1996) Shape description by medial surface construction. IEEE Trans Visualization Comput Graph 2:62–72

Silver D, Wang X (1996) Volume tracking. IEEE Proceeding of Visualization'96, San Francisco, Calif., Assoc. for Computing Machinery, New York, NY, 157–164

Sudhalkar A, Gursoz, Prinz F (1996) Box-skeletons of discrete solids. Comput Aided Design 28:507–517

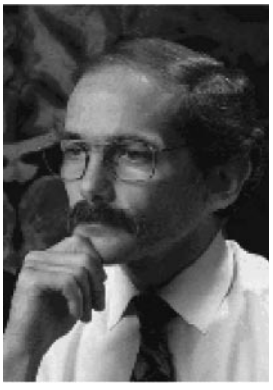
Tsao YF, Fu KS (1982) A 3D parallel skeletonwise thinning algorithm. IEEE Pattern Recognition and Image Processing Conference, Las Vegas, NV, IEEE Computer Society Press Silver Spring, MD, pp 678–683

Turkiyyah GM, Storti DW, Ganter M, Chen H, Vimawala M (1997) An accelerated triangulation method for computing the skeletons of free-from solid models. Comput Aided Design 29:5–19

Zhu SC, Yuille A (1996) FORMS: a flexible object recognition and modeling system. Int J Comput Vision 20:187–212

YONG ZHOU received his MS in Mathematics from Zhejiang University in 1991, and his PhD in Computer Science from Tsinghua University, China, in 1995. Zhou is currently a research Assistant Professor at the Laboratory of Neuro Imaging, University of California at Los Angeles (UCLA). From August 1996 to July 1997, he was a Postdoctoral Associate with the Center of Visual Computing, State University of New York at Stony Brook. His research interests include computer graphics, volume visualization, geometrical modeling, computer vision, image processing, and applications on medical images. He is a member of the IEEE computer society.

ARIE E. KAUFMAN is the Director of the Center of Visual Computing (CVC), a Leading Professor of Computer Science and Radiology at the State University of New York (SUNY) at Stony Brook. He has conducted research and consulted for 27 years specializing in volume visualization; graphics architectures, algorithms, and languages; vitual reality; user interfaces; and multimedia. He received a BS in Mathematics and Physics from the Hebrew University of Jerusalem in 1969, an MS in Computer Science from the Weizmann Institute of Science, Rehovot, in 1973 and a PhD in Computer Science from the Ben-Gurion University, Israel, in 1977.



ARTHUR W. TOGA is a Professor in the Neurology Department at the UCLA School of Medicine. He is the Editor-in-Chief of the Journal of Neuro Imaging. He is the Director of the Laboratory of Neuro Imaging, Associate Director of the Brain Mapping Division of the Neuropsychiatric Institute, and Assistant Chairman of the Department for Research Affairs. His research interests include metabolic brain mapping, cognitive neuroscience, and brain structure visualization. He received his MS and PhD degrees from St. Louis University. He is a member of the ACM and the IEEE Computer Society.