Shape-Based Interpolation of Multidimensional Objects

SAI PRASAD RAYA, MEMBER, IEEE, AND JAYARAM K. UDUPA, SENIOR MEMBER, IEEE

Abstract—Three-dimensional image data produced by medical imaging scanners usually have unequal scanning resolution in different dimensions, the slice separation usually being much greater than the pixel size within an individual slice. The general practice in the three-dimensional display of organs based on such data is to first interpolate between slices to obtain isotropic resolution and then perform object identification and display operations. This implies that, if user interaction is required on a slice-by-slice basis to identify the object of interest (which often is the case), the effort needed is much (two–ten times) more than that required if the uninterpolated slice data were used for identifying the object. The situation becomes much (ten–50 times) worse if a fourth dimension of time is also involved, as in the case of a dynamic organ such as a beating heart. We present a new shape-based interpolation scheme for multidimensional images, which consists of first segmenting the given image data into a binary image, converting the binary image back into a gray image wherein the gray value of a point represents its shortest distance (positive value for points of the object and negative for those outside) from the cross-sectional boundary, and then interpolating the gray image. The set of all points with nonnegative values associated with them in the interpolated image constitutes the interpolated object. The method not only greatly minimizes the user involvement in interactive segmentation, but also leads to more accurate representation and depiction of dynamic as well as static objects. We present the general methodology and the implementation details of the new method and compare it on a qualitative and quantitative basis to the existing methods. The generality of the proposed scheme is illustrated with a number of medical imaging examples.

I. Introduction

The increasingly sophisticated medical imaging modalities [1] such as computerized tomography, magnetic resonance imaging, ultrasound imaging, and positron emission tomography continue to make a significant impact on the diagnosis and treatment of diseases of internal organs. High-speed data acquisition techniques available in some scanners make possible imaging of even dynamic organs such as a beating heart. The recent developments in three-dimensional (3D) imaging [2] have further expanded the benefits from such modalities by providing more sophisticated tools to analyze multidimensional image data.

In medical 3D imaging, the main objectives are to visualize, manipulate, and analyze human internal structures based on multidimensional image data for the purpose of their diagnosis and therapy. In visualization, the main concern is how to create accurate depictions of structures on the screen of a computer display device. There are mainly two classes of approaches for visualizing objects and their surfaces. In the first category, commonly known as *surface rendering*, object surfaces are explicitly formed prior to creating their depiction on a screen via techniques such as hidden-part removal, shading, transparency, dynamic rotation, stereo projection, and coloring [3]–[9]. In the second category, which has come to be known as *volume rendering*, object surfaces are not explicitly computed; rather, depictions of surfaces, interfaces, and pseudosurfaces are generated through a process of projection of voxels [10]–[17]. In manipulation, the objective is to alter structures interactively, mainly for the purpose of simulating certain surgical procedures [18], [19]. In analysis, the concern is how to get quantitative measures of structures in an absolute, comparative, and composite fashion. The subject of this paper is mainly visualization, but it also encompasses analysis, as we explain later on.

Most of the imaging modalities referred to above produce a two-dimensional discrete map of some physical property at a particular cross section of the human body by interrogating it with different sources of energy. A 3D distribution of the physical property (which we refer to as a *scene*) is obtained in the form of a sequence of cross-sectional slices by contiguously scanning a 3D region of the body. Each element of this discrete 3D distribution is an estimate of the overall interaction of energy with the body tissue within a small 3D region. The dimensions of such a small volume element (usually abbreviated as "voxel") depend on a number of scanning parameters such as the scanning beam and detector geometry. Usually, the spacing between the planes defined by the slices is much greater than the size of a pixel within the slice. Since most of the visualization techniques operate on equally spaced 3D samples—in other words, an array of cubic voxels—a preprocessing step becomes inevitable wherein the original data are interpolated to obtain a scene with equal resolution in all the three dimensions. When a fourth dimension is involved (as in the case of a beating heart or a moving joint), usually the number of time instances sampled is not adequate to give a smooth representation of the motion of the object or the sampling in time itself may not be uniform. Consequently, some form

Manuscript received December 30, 1988; revised July 20, 1989. This work was supported by the National Institutes of Health under Grant HL28438.

S. P. Raya was with the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA 19104. He is now with the PURA Labs Inc. 603 S. Valencia Ave, Brea, CA 92621.

J. K. Udupa is with the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA 19104.

IEEE Log Number 8931255.

of interpolation becomes necessary either to account for the unequal resolution or to resample the scene uniformly at any desired resolution, for example, to get finer sampling along all the three dimensions and/or to approximate the object at more time instances than that captured by the scanner.

Whatever method is used, an essential intermediate step called segmentation becomes necessary in visualization. Segmentation is essentially a process of classifying voxels as to the type of tissue they represent [20], [21]. Often, due to numerous reasons, automatic segmentation methods fail (especially for soft tissues) and slice-by-slice interactive outlining of regions is the only alternative left for successfully separating objects of interest in scenes. Clearly, in such situations, the time spent by the user in segmentation is a function of the number of slices covered by the object. If interpolation is done to generate cubic voxels prior to segmentation, which is a common practice, the time requirement rises rapidly to a level where often interactive segmentation becomes impractical. Consider, as an example, the problem of interactively isolating cardiac structures from a gated cine sequence of magnetic resonance images consisting of 12 slices for each of a set of ten time instances over one cardiac cycle. If the pixel size were 0.7 mm and the slice spacing were a uniform 5 mm, the interpolated scene would already have about 860 slices! This problem indeed is the major motivation for the proposed interpolation scheme, wherein interpolation is done after segmentation. Even when automatic segmentation is possible, we expect, at least intuitively, that the information in the original sample points should be more appropriate for segmentation than that in the interpolated points since interpolation may introduce artifacts.

Our second reason for the proposed scheme is that, in the case of dynamic objects, we want to interpolate in the time dimension as well since, very often, the sampled time instances are either not sufficient in number to depict a smooth motion of the object or the sampling itself may be nonuniform. In the cardiac example given above, if we create 20 time instances (which seems to be the minimum needed to represent continuous motion), we will have to deal with 1720 slices! Time is an important aspect in the visualization and analysis of dynamic objects and we consider interpolation in the time dimension to be just as important as in the spatial dimensions. The user involvement problem (mentioned above) apart, though, the commonly used gray-level interpolation schemes can be easily extended to time sequence of scenes; we are not aware of any publication reporting such work.

Our final justification for the new method is that it produces more accurate results than gray-level scene interpolation methods. The published gray-scene interpolation methods [22], [23], [24], [17] all estimate the value to be associated with a new voxel v by linearly interpolating the values associated with the voxels in the given scene in a small neighborhood of v. Even higher order interpolants that we have implemented do not seem to match the accuracy of the proposed method. The subject of this paper—object interpolation—is different from, although closely related to, surface interpolation. Most of the surface interpolation techniques [5-7], [32-37] are applicable to the algorithms reported in this paper. In fact, we claim that our algorithms improve the outcome of these techniques.

The rest of the paper is organized as follows. We present the basic methodology in Section II and algorithms and implementational details in Section III. Experimental results illustrating the superiority of the proposed method are presented in the fourth section. Finally, we state our conclusions in Section V.

II. The Interpolation Method

Our algorithms are based on a generalization of a theory proposed in [25] wherein the notion of univariate data interpolation is generalized to achieve high-order interpolation between cross sections of a 3D body of general topology. We first outline the main idea behind this theory and then describe how we have generalized it.

Suppose we have an object O defined as a subset of the 3D Euclidean space R^3. We are given a set of parallel cross sections $C_1, C_2, \cdots, C_k$ of O at $z_1, z_2, \cdots, z_k$ (see Fig. 1). Our interpolation problem is to estimate new cross sections $C'_1, C'_2, \cdots, C'_m$ at $z'_1, z'_2, \cdots, z'_m$.

To solve this problem, it is first converted into a univariate interpolation problem: given the cross sections $C_1, C_2, \cdots C_k$; determine if $(x, y, z'_j) \in C'_j$ for some fixed real numbers x, y and some integer $j \in \{1, 2, \cdots, m\}$. To solve the latter problem, to each (x, y, z_i), for $1 \le i \le k$, a real number d_i is assigned which represents the shortest distance from (x, y, z_i) to the boundary[1] of C_i, with the convention that d_i is positive if $(x, y, z_i) \in C_i$, negative if $(x, y, z_i) \in \overline{C}_i$, the complement of C_i, and 0 if (x, y, z_i) is a boundary point of C_i. A univariate function f is then determined that interpolates $d_1, d_2, \cdots, d_k$. Finally, (x, y, z'_j) is considered to belong to C'_j if $f(z'_j) \ge 0$. Of course, f can be any appropriate interpolating function.

We generalize the above method not only to interpolate static and time-varying 3D objects, but also to generate finer representations of coarsely sampled multidimensional objects. The generalization, thus, would allow us to interpolate time-varying 3D objects from its two-dimensional cross sections both in the z dimension and the time axis, as well as enable us to interpolate within the cross section to create finer representations.

Our n-dimensional (nD) formulation of the problem is as follows. We define two nD rectangular grid systems denoted g_R and g_I in the nD Euclidean space R^n. g_R is defined by n sets of mutually orthogonal hyperplanes; within each set, the hyperplanes are parallel, but not necessarily equidistant. Each point where n hyperplanes meet

[1]The boundary of C_i is the set of closure points common to both C_i and $\overline{C}_i$. A point P is a closure point of C_i if for every positive real number a, there exists a point P' in C_i such that the Euclidean distance from P to P' is less than a.

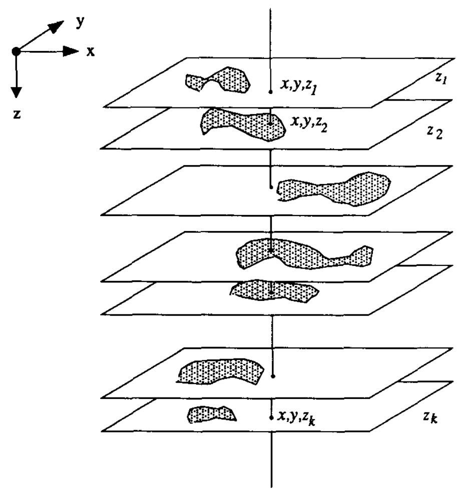

Fig. 1. Illustration of the method of interpolating objects from their slices. The rectangular borders of slices containing the object are shown just to indicate the level of the slice plane. $z_1, z_2, \cdots, z_k$ are the location of the slices. The blob within each rectangle represents the cross section of the object.

defines a grid point of g_R. We define g_I similarly, with the only difference that the hyperplanes within each set now are equidistant in addition to being parallel. We denote by G_R and G_I the set of all grid points in the g_R and g_I systems, respectively.

We define an *object* O in R^n as a closed subset of R^n. In practice, only a discrete (sampled) representation of O, rather than O itself, is available to us. We define a *discretization* O_L of O in the g_L system, for $L \in \{R, I\}$, as the set

$$O_L = O \cap G_L. \tag{1}$$

We assume that the object O we are interested in is of finite support (we also assume that the grid spacing in G_L represents an adequate sampling rate). This means that, for $1 \leq i \leq n$, there exist finite numbers B_i^l and B_i^h such that

$$O_B \cap O = O \tag{2}$$

where

$$O_B = \{X = (x_1, x_2, \cdots, x_n) | \quad \text{for } 1 \leq i \leq n, \; B_i^l \leq x_i \leq B_i^h\}. \tag{3}$$

Clearly, O_L as defined above is a finite set. Our generalized problem of interpolating an nD object can now be made precise:

given a discretization O_R of an *nD* object O of finite support in the g_R grid system and another grid system g_I,

determine the discretization O_I of O in the g_I system.

We now describe our approach.

We use the notation $\text{INF}_{iL}(y)$ to denote the largest ith grid-point coordinate in the g_L system that is smaller than the number y, and $\text{SUP}_{iL}(y)$ to denote the smallest ith grid-point coordinate in g_L that is greater than y. We define the *domain* E_R of O_R to be the set

$$E_R = \{X = (x_1, x_2, \cdots, x_n) | X \in G_R \text{ and for } 1 \leq i \leq n, R_i^l \leq x_i \leq R_i^h \text{ where } R_i^l = \text{INF}_{iR}(B_i^l) \text{ and } R_i^h = \text{SUP}_{iR}(B_i^h)\}. \tag{4}$$

Roughly speaking, E_R is the smallest set of grid points that completely sample O and still leave at least one layer of grid points around O that do not fall inside O. We define the *domain* E_I of O_I similarly based on the domain E_R of O_R.

$$E_I = \{X = (x_1, x_2, \cdots, x_n) | X \in G_I \text{ and for } 1 \leq i \leq n, I_i^l \leq x_i \leq I_i^h \text{ where } I_i^l = \text{INF}_{iI}(R_i^l) \text{ and } I_i^h = \text{SUP}_{iI}(R_i^h)\}. \tag{5}$$

Given O_R and g_I, we can determine E_R and hence E_I, both being finite. We define the domain of $\overline{O_R} = E_R - O_R$ also to be E_R.

The idea of interpolating the distance to the cross-sectional boundary is central to shape-based interpolation. The basic premise here is that the boundary of O is available as a sampled set of cross-sectional (2D) boundaries. To estimate O_I and its boundary in the given grid system g_I, hence, it is very appropriate to interpolate the cross-sectional boundaries obtained on the g_R system. For most medical imaging devices, the 2D cross sections are naturally defined by the plane of the slice images. For shape-based interpolation to be most effective, the slicing planes should be defined in such a way that the sampling resolution in these planes is higher than in other possible sets of parallel planes. (A theoretical proof of this assertion is beyond the scope of this paper, but an intuitive justification is fairly straightforward. Since we think of the "distance to boundary" to capture shape information, the more accurate this information is, the better is the interpolated result derived from it.) In multidimensional medical images, this is usually the case. If this is not so, the slicing planes should be appropriately redefined. Without loss of generality, we assume hereafter that if $X = (x_1, x_2, \cdots, x_n) \in G_R$, then x_1, x_2 represent coordinates within a slicing plane. In other words, the slicing planes are orthogonal to the $x_3, x_4, \cdots, x_n$ axes in R^n.

We define the *slice* that contains a point $X' = (x_1', x_2', \cdots, x_n')$ in E_R to be the set

$$C(X') = \{X = (x_1, x_2, \cdots, x_n) | X \in E_R \text{ and for } 3 \leq i \leq n, x_i = x_i'\}. \tag{6}$$

To each point X in E_R we associate four points, called *neighbors* of X, which are the points of G_R closest to X along each of the directions $x_1, -x_1, x_2, -x_2$. If $A_R \subset O_R$, $X \in A_R$, and at least one neighbor of X that is in E_R is not in A_R, then we call X a *boundary point* of A_R. We define the boundary of O_R in a slice $C(X')$ to be the set

$$\delta(C(X')) = \{X = (x_1, x_2, \cdots, x_n) | X \in C(X') \text{ and } X \text{ is a boundary point of } O_R\}. \quad (7)$$

Since E_I samples the entire set O, to determine O_I, we just have to find out for every element X_I of E_I if it qualifies as a member of O_I. To do this, we first choose an interpolating function F, and then based on F, determine a neighborhood $N_F(X_I)$ around X_I consisting of t points of G_R. To each point Z_i in $N_F(X_I)$, we assign a number D_i that represents the shortest distance of Z_i from $\delta(C(Z_i))$ (we will describe later on how this distance is computed) with the convention that D_i is positive if $Z_i \in O_R$, and negative otherwise. As an example, suppose we choose F to be an nth-order linear interpolating function; then $t = 2^n$ and the elements of $N_F(X_I)$ are the vertices of the nD parallelepiped that contains X_I. (If X_I falls on the boundary of such a parallelepiped, any appropriate tie breaker can be used to assign X_I to a unique nD parallelepiped.) Knowing $D_1, D_2, \cdots, D_t$, we estimate the parameters of F such that F interpolates $D_1, D_2, \cdots, D_t$. Finally, we decide

$$X_I \in O_I \qquad \text{iff } F(X_I) \geq 0. \quad (8)$$

In medical imaging applications, we consider a sequence of 3D scenes corresponding to a sequence of contiguous time instances as representing a four-dimensional (4D) scene. The cardiac images, for example, obtained through gating or through a fast scanning technique [30], [31] represent data sampled in three spatial dimensions and one temporal dimension. Of course, the unit of measurement along the temporal axis is different from that along the spatial axes. Otherwise, the temporal aspect is similar to the spatial in spirit because both represent a sampling of a certain property, the temporal sequence representing a sampling of the dynamic behavior of the objects scanned. It may be argued, at least in the cardiac case, that the temporal aspect is quite different from the spatial aspect because of periodicity of cardiac dynamics, and that the time sequence wraps around. This may be true for perfectly periodic motion, but often, the very reason for imaging may be a suspected abnormality in the dynamics, in which case, the temporal aspect should be considered as a sampling of the dynamic behavior over an interval of time.

In our approach, the g_R grid system is specified by the sampling parameters of the given scene (pixel size and location of each slice for 3D scenes, and in addition to these parameters, the time instances for 4D scenes). The g_I grid system is specified by the user (via the specification of the desired pixel size, slice separation, and time separation). Our overall method consists of first applying a segmentation procedure to the given scene to convert it into a binary scene. The set of the centers of all scene elements (voxels in 3D, hypervoxels in 4D) that have the value 1 assigned to them in the binary scene represents the descritization O_R in the g_R system of our object of interest. Having obtained O_R, we apply the algorithms of the following section to compute the interpolated object O_I.

III. Algorithms

In this section, we describe two algorithms. Algorithm A is a direct incorporation of the idea described in the previous section and is useful in understanding the underlying process of shape-based interpolation. Algorithm B is an improvement on Algorithm A for the special cases of $n = 3$ and 4 and is to be preferred due to its computational efficiency. In these algorithms, we assume that, given O_R, we have already computed E_R and E_I.

A. The Simple Algorithm

Input: The discretization O_r (a nonempty set) of the object of interest, the grid system g_R, the domain E_R of O_R and E_I of OI, the desired grid system g_I, the type of the interpolating function F, and the object dimensionality n.

Output: A list OI containing the points of O_I.

Data Structures: A list WR that initially contains the points of O_R or $\overline{O_R}$, a list M containing boundary points, an array D containing the distance of every point X_R of E_R from $\delta(C(X_R))$, and an array N_F which contains points in a neighborhood of the point $X_I \in E_I$.

The distance from boundary is determined by repeated elimination of boundary points. In every iteration, one layer of boundary points is removed from WR, and at the same time, the distance of these points from the true boundary is upgraded in a cumulative fashion. Eventually, when WR is empty, the distance from boundary will have been calculated for all points. The process is repeated for O_R and $\overline{O_R}$ separately.

Algorithm A

```
begin
 1) set all elements of D to 0;
    for i = 1 and -1 do {loop on $O_R$ and $\overline{O_R}$}
 2)   if i = 1 then
        store $O_R$ in WR;
        boundary ← TRUE; {flag to ensure distance is 0 for true boundary points}
 3)   else store $E_R - O_R$ in WR;
 4)   while (WR is not empty) do {calculate shortest distance from boundary by repeated elimination of boundary points}
 5)       for each point $X_R$ in WR do
 6)           if $X_R$ is a boundary point of the set of points in WR then
 7)               put $X_R$ in M;
 8)               if (boundary = FALSE) then
 9)                   find $Y_R$, the neighbor of $X_R$ not in WR that is closest to $X_R$, and its distance $DY_R$ from $X_R$;
10)                   $D[X_R] \leftarrow D[Y_R] + i * DY_R$;
                  end_if;
              end_if;
          end_for;
```

```
            11)    boundary ← FALSE; {boundary is true only for first iteration}
            12)    for each point X in M remove X from WR and M;
                end_while;
            end_for;
    13) for each point $X_I$ in $E_I$ do {interpolate distances and estimate OI}
        14)    determine the points of $E_R$ in the neighborhood $N_F(X_I)$;
        15)    compute $F(X_I)$ knowing $D[X_R]$ for each $X_R$ in $N_F(X_I)$;
        16)    if $F(X_I) \geq 0$ then put $X_I$ in $O_I$;
    end_for;
end
```

In this algorithm, $D[X_R]$ in Step 10) means the number contained in the array D at a location corresponding to the point X_R. D can be, of course, either one-dimensional or n-dimensional.

Steps 2) and 3) represent the beginning of a loop first on the set O_R and then on $\overline{O_R} = E_R - O_R$. The calculation of the shortest distance of each point from the boundary is done in Step 4). In every execution of the loop in Step 4), a layer of boundary points in *WR* is detected and removed. This loop terminates when no more boundary points (or points) are left in *WR*. Note that the variable *boundary* is TRUE only when X_R belongs to the boundary of O_R. Hence, the Step 8), the shortest distance entry in D for such points is retained as 0. Step 13) is where interpolation is done to determine the points of O_I. In Step 9), "distance DY_R" means the nD Euclidean distance between the points X_R and Y_R.

B. The Fast Algorithm

Note that, in Step 13) of Algorithm A, if every point X_R in $N_F(X_I)$ is in O_R, then $D[X_R]$ is positive for each X_R, and clearly, $F(X_I) \geq 0$ and so $X_I \in O_I$. Similarly, if every point X_R in $N_F(X_I)$ is in $\overline{O_R}$, then $D[X_R]$ is negative for each X_R, $F(X_I) < 0$, and hence X_I does not belong to O_I. Clearly, in both of these situations, it is not necessary to compute $F(X_I)$, which implies that it is not necessary to compute $D[X_R]$ for $X_R \in N_F(X_I)$. In other words, for points that are far away from $\delta(C(X))$ (whether inside or outside O_R), we need not compute the shortest distance. This indeed is a great simplification since the number of points in the vicinity of the boundary is usually a small fraction (1-3%) of the number of points in the domain of the object.

The size of the vicinity of the boundary where distance calculation is required depends on the type of the interpolating function F chosen. If, the example, F is an nth-order linear function, then it can be shown that we need to run the loop in Step 4) only two times for both O_R and $\overline{O_R}$. In general, given the type of F, we can determine the number of iterations M_F for which the loop in Step 4) should be executed so that we do a minimal computation of the shortest distances.

Our description so far has been very general, independent of the dimensionality of the space in which the scene is defined. The algorithms are applicable to sampled objects of any (finite) dimensionality. In medical imaging, we come across mostly 3D and 4D objects, and usually, the scene data come naturally in the form of a sequence of slices. In such situations, with a few assumptions that are realistic, it is possible to devise a specialized algorithm that still incorporates the ideas of the previous section, but that is more efficient than Algorithm A. In the rest of this section, we describe one such algorithm which we call Algorithm B.

We make the following assumptions: 1) n is either 3 or 4, 2) the slice planes are parallel to the x_1x_2 plane, 3) the grid spacing the x_1 and x_2 directions in the g_R system is uniform and is equal to each other, and 4) the grid positioning and spacing in the g_I system in the x_1 and x_2 directions is identical to that of the g_R system.

Our following presentation of Algorithm B is for $n = 4$, but the changes necessary for $n = 3$ are trivial. We also assume a second-order linear (i.e., bilinear) interpolating function. Modifications required to incorporate higher degree interpolants are straightforward.

Input: The discretization O_R (a nonempty set) of the object of interest, the grid systems g_R and g_I, and the domains E_R and E_I of O_R and O_I.

Output: A list *OI* containing the points of O_I.

Function Called: DIST.

Algorithm B

```
begin
    1) for i ← 3 to 4 do {find min, max location of slices in $E_I$ that intersect object}
        2)    find $XRMIN_i$ and $XRMAX_i$, the largest and smallest numbers such that $XRMIN_i < XRMN_i$ and $XRMAX_i > XRMX_i$ where $XRMN_i$ and $XRMX_i$ are the smallest and largest $x_i$-coordinate of all points in $O_R$.
        3)    find $XIMIN_i$ and $XIMAX_i$, the smallest and largest $x_i$-coordinate in $E_I$ such that $XIMIN_i > XRMIN_i$; and $XIMAX_i < XRMAX_i$;
    end_for;
    4) for $x_3^I$ ← $XIMIN_3$ to $XIMAX_3$ do {loop on all relevant slices of $E_I$}
       for $x_4^I$ ← $XIMIN_4$ to $XIMAX_4$ do
        5)    find $x_{il}^R$ and $x_{ih}^R$, for $3 \leq i \leq 4$, the largest and smallest $x_i$-coordinate of the points in $E_R$ such that $x_{il}^R < x_i^I$ and $x_{ih}^R > x_i^I$;
        6)    for each point $X_I = (x_1^I, x_2^I, x_3^I, x_4^I)$ in $E_I$ do
            7)    $X_1 \leftarrow (x_1^I, x_2^I, x_{3l}^R, x_{4l}^R)$;
            8)    $X_2 \leftarrow (x_1^I, x_2^I, x_{3h}^R, x_{4l}^R)$;
            9)    $X_3 \leftarrow (x_1^I, x_2^I, x_{3l}^R, x_{4h}^R)$;
            10)   $X_4 \leftarrow (x_1^I, x_2^I, x_{3h}^R, x_{4h}^R)$;
            11)   if all of $X_1, X_2, X_3, X_4 \in O_R$ then put $X_I$ in OI;
                  else if at least one of $X_1, X_2, X_3, X_4 \in O_R$ then
                      if DIST$(X_I, X_1, X_2, X_3, X_4, O_R) \geq 0$ then put $X_I$ in OI;
```

```
      end_for;
    end_for;
  end_for;
end
```

The function DIST returns the shortest distance of the point X_I from $\delta(C(X_I))$ by interpolating the distance of X_1, X_2, X_3, X_4 from $\delta(C(X_1))$, $\delta(C(X_2))$, $\delta(C(X_3))$, $\delta(C(X_4))$,respectively.

```
Function DIST($X_I$, $X_1$, $X_2$, $X_3$, $X_4$, $O_R$)
  1) for $i \leftarrow 1$ to 4 do
  2)   if $X_i \in O_R$ then
          $d \leftarrow 1$; {increment for $d_i$}
          $d_i \leftarrow -1$; {initialization for distance of $X_i$ from $\delta(C(X_i))$}
  3)   else
          $d \leftarrow -1$;
          $d_i \leftarrow 0$;
  4)   $d_n = 0$; {represents city-block distance from $X_i$ of the points of $E_R$ tested for membership in $E_R$}
  5)   done ← FALSE;
  6)   while (done = FALSE) do
  7)       $d_n \leftarrow d_n + 1$;
  8)       $d_i \leftarrow d_i + d_i$;
  9)       find the four points $NX_1$, $NX_2$, $NX_3$, $NX_4$ of $E_R$ that represent the $d_n$th grid point from $X_i$ in $-x_1$, $-x_2$, $x_1$, $x_2$ directions, respectively;
  10)      for each point X in $E_R$ on the boundary of the polygon $NX_2NX_2NX_3NX_4$ do
               if ($X \in O_R$ and $X_i \in \overline{O_R}$) or ($X \in \overline{O_R}$ and $X_i \in O_R$) then
                   done ← TRUE;
                   loopexit; {exit from the loop beginning at Step 1.5.4}
               end_if;
           end_for;
       end_while
  end_for;
  11) compute the distance DIST of $X_I$ from $\delta(CX_I)$) by bilinearly interpolating $d_1$, $d_2$, $d_3$, $d_4$;
  12) return DIST;
```

Note that the computationally expensive step of evaluating the function DIST in Step 11) in Algorithm B is done only for the points in the Exclusive OR region between slices. For $n = 3$, this implies that when a new slice is to be estimated in E_1, the distance computation and interpolation are done only for points of O_R that are not common in the two slices that are closest to the new slice in the $-x_3$ and x_3 directions. In fact, in our implementation, we first compute the Exclusive OR between the relevant slices before entering the loop in Step 6), and then execute this loop only for points in the Exclusive OR region. Clearly, the number of points in the Exclusive OR region depends on the shape of the object and on how close the slices are spaced.

IV. Experimental Results and Discussion

In this section, we present our results using specimen as well as patient data sets pertaining to static as well as dynamic objects. We also discuss possible extensions to the presented methodology.

Our aim here is to compare the performance of the new algorithms to that of gray-level interpolation techniques currently used in 3D imaging, both on quantitative and qualitative grounds. We compare among four methods—both linear and natural four-point cubic spline interpolation [29] for both gray-level and shape-based interpolation. (Shape-based interpolation was done using Algorithm B and a modified version of it that incorporates cubic-spline interpolation.) In the description to follow, we use the following abbreviations for these four methods: GL—gray-level linear, GC—gray-level cubic spline, BL—binary (shape-based) linear, BC—binary (shape-based) cubic spline. In all examples, segmentation was done by thresholding and the threshold selection was done interactively by displaying simultaneously the slices of the gray scene and that of the resulting binary scene. The same threshold was used in the various methods of interpolation for a given object. We have used our earlier reported algorithms for surface tracking [26] and rendering [27] for creating the shaded-surface images presented here, but used a modified version of the image-space gradient shading technique reported in [28].

A. Quantitative Aspects

A description of the three static object data sets we have used is given in Table I. A typical slice of the binary scene (uninterpolated) for these objects is shown in Figs. 2, 3, and 4. The quantitative results (explained below) are summarized in Tables II, III, and IV.

Often, one of the main objectives in 3D and 4D scene analysis is to estimate the volume of a specified object. While the accuracy of volume computation is very much dependent on the segmentation process, we believe that the interpolation scheme also determines the accuracy. How accurately an interpolated slice represents the object cross section is therefore important in computing volume (as well as other measures based on the object or its boundary). To determine this, we intentionally subsampled the three test objects in the x_3 direction by removing slices at regular intervals from the given scene and then estimated interpolated slices exactly at the locations corresponding to the removed slices from the subsampled scene using the four methods. To see how well the removed slices are estimated, we define a percentage error ϵ as

$$\epsilon = \frac{1}{N} \sum_{i=1}^{N} \frac{|OA_i - IA_i|}{OA_i} \times 100 \tag{9}$$

where

N	is the number of original slices removed,
OA_i	is the area of the object cross section in the ith removed slice,

TABLE I
A Summary of the Three Static Object Data Sets Used in the Experiment

Object	Description	Imager	Pixel size	Slice spacing (uniform)	Scene dimensions (in pixels and number of slices)
1	Skull Specimen	GE8800 - CT	0.8 mm	3.0 mm	256x256x68
2	Patient Skull	GE9800 - CT	0.4 mm	1.5 mm	256x256x64
3	Brain Specimen	GESigna-MR	0.78 mm	3.0 mm	256x256x31

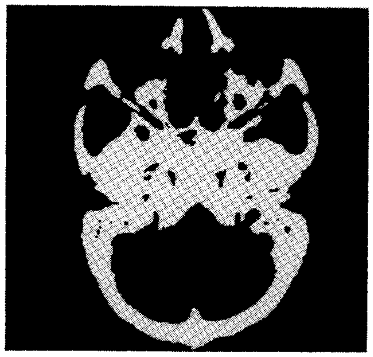
Fig. 2. A typical cross section of Object 1.

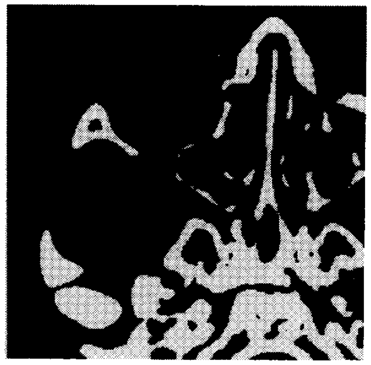
Fig. 3. A typical cross section of Object 2.

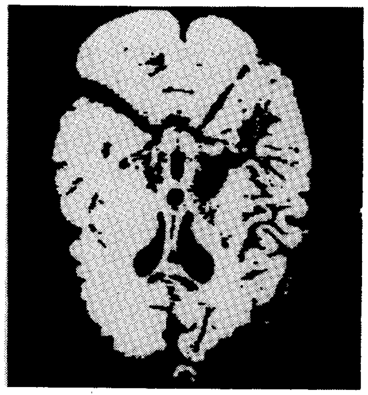
Fig. 4. A typical cross section of Object 3.

TABLE II
Quantitative Comparison of Different Interpolation Schemes for Object 1

Interpolation factor	Percentage error			
	GL	GC	BL	BC
2	25.214	20.891	5.206	3.952
3	36.135	32.184	11.568	7.239
4	44.145	40.53	18.145	11.752
5	59.418	55.863	28.163	16.267

TABLE III
Quantitative Comparison of Different Interpolation Schemes for Object 2

Interpolation factor	Percentage error			
	GL	GC	BL	BC
2	13.575	12.779	5.134	3.864
3	21.876	19.775	10.972	7.017
4	27.652	24.981	13.491	10.562
5	41.367	41.657	14.629	13.228

TABLE IV
Quantitative Comparison of Different Interpolation Schemes for Object 3

Interpolation factor	Percentage error			
	GL	GC	BL	BC
2	8.881	6.706	4.879	3.608
3	10.2	7.953	6.953	4.718
4	15.902	11.968	9.12	6.775
5	18.823	16.507	12.318	11.406

IA_i is the area of the object cross section in the ith estimated slice.

We express the rate of subsampling by a number called *interpolation factor* which we define as the ratio between the slice spacing after and before removal of slices. We computed ϵ for the four methods for different interpolation factors. The results are summarized in Tables II, III, and IV for objects 1, 2, and 3, respectively.

It is clear from these tables that BC outperforms the rest and BL is superior to both GL and GC. It is also clear that ϵ increases when the interpolation factor is increased (meaning that the slice spacing for the scene input to the interpolation algorithm is increased). The improvement in performance of BL and BC seems to be greater for more complex objects, as seen by the values of ϵ in Tables II and IV. To further illustrate the superior performance of shape-based interpolation schemes, we present an analytical explanation with the help of a simple object of known geometry.

Consider a cone of height H and base radius R in R^3 as shown in Fig. 5. For the sake of simplicity of analysis, we assume the apex of the cone to be the origin of R^3. Assume that the cone is homogeneous, the density of its material is p_i, and that the density outside the cone is p_o such that $p_i > p_o$. Given the cross sections of the cone at

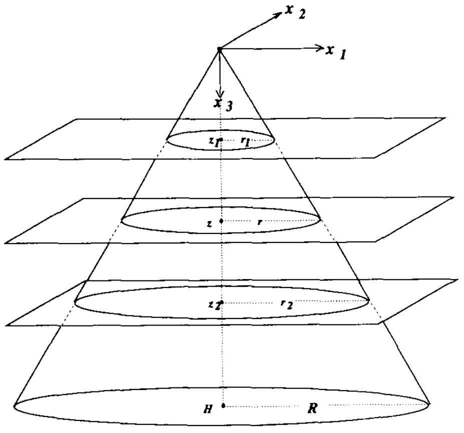

Fig. 5. A simple object used to analytically explain the superior performance of shape-based interpolation schemes.

$x_3 = z_1$ and $x_3 = z_2$, $0 < z_1 < z_2 < H$, our idea is to estimate the cross section at $x_3 = z$, $z_1 < z < z_2$, using both gray-scale and shape-based schemes and then compare them to the actual cross section of the cone at $x_3 = z$.

The actual cross section at $x_3 = z$ is given by

$$C_{za} = \left\{ (x_1, x_2, z) \,\middle|\, \sqrt{x_1^2 + x_2^2} \le \frac{R}{H} z \right\}. \quad (10)$$

Given the cross-sectional images at $x_3 = z_1$ and $x_3 = z_2$ and a threshold T (a point is considered to belong to the object if its density is greater than or equal to T), the cross section at $x_3 = z$ estimated by the GL method can be expressed as

$$C_{zg} = \left\{ (x_1, x_2, z) \,\middle|\, \sqrt{x_1^2 + x_2^2} \le r' \right\} \quad (11)$$

where

$$r' = \begin{cases} r_1 & \text{if } p'(z) < T \le p_i \\ r_2 & \text{if } p_o < T \le p'(z) \end{cases} \quad (12)$$

$T > p_i$ and $T \le p_o$ cases are of no interest

$$\text{and } p'(z) = p_o + \frac{p_i - p_o}{z_2 - z_1}(z - z_1) \quad (13)$$

is the density of the cross section at z in the annular region $\{(x_1, x_2, z) \mid r_1 < \sqrt{x_1^2 + x_2^2} \le r_2\}$ (note that the density within this region is constant). Clearly, there is no value of T for which the estimated cross section at z would equal the true cross section. Further, as seen from (12), the estimated cross section at z is of radius either r_1 or r_2, depending on the value of T chosen. The effect of the GL method is, hence, essentially to replace an original slice by a block of slices of similar shape, the thickness of the block being equal to the slice interval. [In practice, this blockiness appears to be somewhat smoothed because of the inherent blur in the cross sectional boundaries which is due to the finite width of the point-spread function of the tomographic device. The blurring itself is dependent on the "thickness" of the slice. The effect of blockiness is maximum when this thickness is small compared to the distance between slices, minimum when the thickness equals the distance (i.e., the slices exactly abut), and is quite peculiar when the thickness is greater than the distance (i.e., the slices overlap).] In principle, then, the GL method is equivalent to nearest neighbor interpolation of binary cross sections.

Suppose now that we use any threshold such that $p_o < T \le p_i$ to obtain binary cross sections at $x_3 = z_1$ and $x_3 = z_2$. The cross section at $x_3 = z$ estimated by the BL method can then be expressed as

$$C_{zb} = \left\{ X = (x_1, x_2, z) \,\middle|\, D(X) = \left((r_1 - r) + \frac{r_2 - r_1}{z_2 - z_1}(z - z_1) \right) \ge 0 \right\} \quad (14)$$

where $r = \sqrt{x_1^2 + x_2^2}$ and $D(X)$ is the distance of X from $\delta(C(X))$. Clearly,

$$D(X) > 0 \qquad \text{if } r \le r_1$$

and

$$D(X) < 0 \qquad \text{if } r \ge r_2. \quad (15)$$

Further, there are some points X such that $r_1 < r < r_2$ for which $D(X) \ge 0$. It is readily seen that C_{zb} is a disk of radius r such that $D(X) = 0$ or, from (14),

$$(r_1 - r)(z_2 - z_1) + (r_2 - r_1)(z - z_1) = 0. \quad (16)$$

Substituting $r_1 = Rz_1/H$ and $r_2 = Rz_2/H$ in (16), we get

$$r = \frac{R}{H} z, \quad (17)$$

which indeed equals the radius of the actual (circular) cross section C_{za} of the cone at $x_3 = z$.

Of course, as $(z_2 - z_1)$ tends to zero, the difference between C_{zg} and C_{zb} also becomes negligible. However, the grid spacing in the x_3 direction is, in practice, large enough for this difference to be significant. It should be clear by now that the BL and BC methods actually try to interpolate the shape of the object; hence, we have the name "shape-based interpolation."

B. Qualitative Aspects

Objective criteria to compare the quality of shaded-surface images generated by different techniques have not yet evolved in 3D imaging. Our comments regarding qualitative comparisons are, hence, necessarily subjective. Clearly, in the limit as the slice spacing goes to zero, there should be little difference due to the method of interpolation in the appearance of the shaded-surface images for a given method of visualization. In our limited experience with these four methods so far, we find little difference between the performance of GL and GC. The difference between BL and BC is insignificant for small slice spacing, but becomes more pronounced as the slice spacing increases. BC seems to produce smoother boundaries than BL. Similarly, BL and BC seem to generate smoother boundaries than GL and GC, the difference becoming quite apparent in the images when the slice separation is large (more than about six times the pixel size). We demonstrate this aspect with the help of a CT data set pertaining to the broken pelvis of a patient. The slice spacing for the data set is 5 mm. Fig. 6 shows shaded surface images derived from the four interpolation techniques. Note that the BL and BC methods produce a smoother surface representation than the GL and GC methods. The characteristic of turning the slices into thick wedges and the consequent "slicing artifact" are quite apparent in the images corresponding to the latter methods. To further illustrate this aspect, we precisely subsampled the original data set by leaving out alternate slices and then reconstructed the resultant scene (with an effective slice spacing of 10 mm) with the four interpolation schemes. Fig. 7 shows shaded surface images of this subsampled scene. BL and BC, in our view, seem to approximate the original shape of the object more closely than GL and GC.

The shape-based interpolation schemes are most effective in analyzing dynamic objects, especially cardiac structures. In images generated by most commercially available cardiac imaging devices, often the only method of segmentation that is effective in identifying cardiac structures is slice-by-slice interactive outlining of object regions. Obviously, the proposed interpolation method greatly reduces the tedium of interactive segmentation. Equally important is the ability of the method to interpolate the object at more time instances than that available in the original data. In our opinion, shaded-surface image display based on temporal interpolation greatly enhances the portrayal of dynamics of the object. In all the images from different sources that we have analyzed so far (Cine CT, Dynamic Spatial Reconstructor, and Magnetic Resonance Imagers), improvement in the depiction of dynamics of the object has consistently been striking. Unfortunately, it is not possible to substantiate this point using the static medium of publishing images in printed form. Nonetheless, we shall try to illustrate this point by showing all shaded surface images for one cardiac cycle. Figs. 8 and 9 show with and without temporal interpolation a time sequence of shaded surface images of the blood pool in the left ventricle, left atrium, and aorta of a patient. The data were obtained from an Imatron Cine CT scanner; the original ten volumes were interpolated to twenty volumes using the BL method.

Finally, we present the results of an experiment set up to examine the qualitative accuracy of temporal interpolation. We picked volumes 4 and 6 of the original ten-volume data set and interpolated the fifth volume. We then compared the shaded surface image of this interpolated volume to the shaded surface image of the original volume which was deliberately left out. Fig. 10 shows the two shaded surface images. It is clear that the interpolation scheme is able to effectively preserve the shape of the object even in the temporal domain.

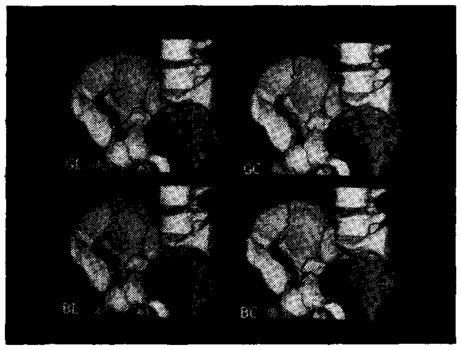

Fig. 6. Shaded surface images of a CT data set pertaining to the broken pelvis of a patient derived using the four interpolation techniques: GL—Gray-scale linear, GC—gray-scale cubic spline, BL—binary (shape-based) linear, BC—binary (shape-based) cubic spline.

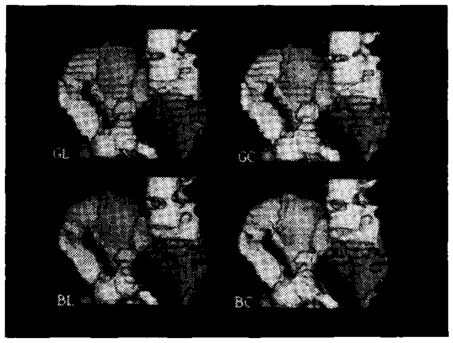

Fig. 7. Shaded surface images of a subsampled scene (with alternate slices removed from the data set described in Fig. 6) derived using the four interpolation techniques: GL—gray-scale linear, GC—gray-scale cubic spline, BL—binary (shape-based) linear, BC—binary (shape-based) cubic spline.

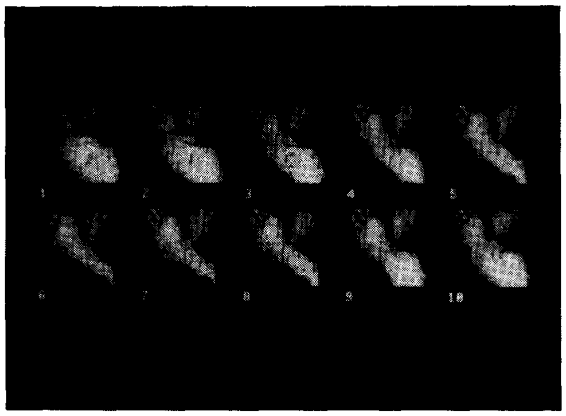

Fig. 8. The original ten-volume time sequence of shaded surface images of the blood pool in the left ventricle, left atrium, and aorta of a patient. The data were obtained from an Imatron Cine CT scanner.

It may appear at the outset the shape-based interpolation is applicable only to surface rendering methods of visualization because of the notion of "distance from boundary" involved in this scheme. With a slight modi-

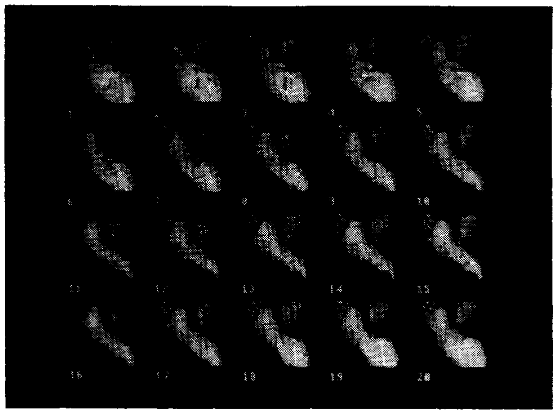
Fig. 9. The 20-volume time sequence (obtained by applying shape-based interpolation in temporal domain) of shaded surface images of the data described in Fig. 8.

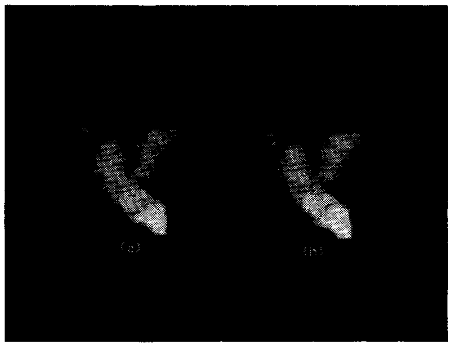
Fig. 10. Comparison of the original and interpolated shaded surface images of one time instance of the data described in Fig. 8. (a) Fifth time instance of the original data (volume = 89.5699 cc). (b) Interpolated based on fourth and sixth time instances (volume = 89.36894 cc).

fication of this notion, the interpolation scheme can be extended to other methods of visualization, in particular, to volume rendering techniques where neither the object nor its boundary may be explicitly available. (Note that in the algorithms presented in this paper, we assumed that the object of interest is available and not its boundary. For some forms of boundary representation (especially discrete), the object and boundary representations are interchangeable. In medical imaging, it is almost always the case that the object is obtained first and then its boundary; hence, the object is always available whenever its boundary is available.) However, it is always the case that some form of classification of the voxels in the scene as to which object each voxel is most likely to belong is available. In other words, we have a probability map associated with each object that tells how likely is each voxel to belong to the object. We thus have probabilistic boundaries instead of the definitive boundaries considered in this paper. The notion of shortest distance from boundary now becomes probabilistic too. We may define the probabilistic distance as the integral of the shortest distances to probable boundaries where each distance is weighted by the probability associated with the boundary.

In this paper, we have not addressed the sampling rate requirements nor the smoothness criteria that the objects should satisfy for the stated results to hold good. It is possible to create examples by making the sampling rate sufficiently low so that none of the interpolation techniques would produce any sensible result. This would certainly form an appropriate topic for further study. A somewhat related issue is whether it is necessary to have the hyperplanes that define the grid points orthogonal between sets and parallel within a set. In principle, the algorithms can be readily extended to this situation provided the hyperplanes within a set do not mutually intersect (i.e., they are nearly parallel). We may have to weight each element of $N_F(X_l)$ (Step 15) of Algorithm A) appropriately knowing the hyperplane in which it lies. Measures other than the Euclidean distance from boundary as characterizers of shape are also worthy of further study.

The shape-based interpolation techniques may appear to be computationally more expensive than the gray-level interpolation techniques currently used in 3D imaging. Of course, the calculation of distances is an additional step (which can be very expensive in the case of probabilistic distances described above). However, this step, as well as actual interpolation, is carried out usually on a small fraction of the points in the domain E_R of the object. In gray-level interpolation, in general, the interpolation calculation has to be carried out on all points of E_R except when thresholding is used to segment the object where, of course, the calculation can be confined to a vicinity of the boundary. A very desirable feature of the algorithms presented here is that they are highly parallel and are, hence, best suited to the fast-emerging parallel computational technology.

V. Conclusions

We have presented a methodology and algorithms for interpolating multidimensional objects which, we feel, should be a method of choice, particularly in medical 3D imaging, and possibly in other areas of multidimensional imaging. It has several advantages over the currently used gray-level interpolation techniques.

1) When automatic segmentation methods fail (which is often the case in soft-tissue imaging and disarticulation of joints), the tedium associated with interactive segmentation is greatly minimized; even when automatic segmentation is possible, we believe that the proposed method speeds up object identification.

2) The new method leads to greatly enhanced portrayal of dynamics in the display of dynamic objects such as internal structures of the heart. The temporal interpolation afforded by the method is quite unique.

3) The method seems to give more accurate quantitative measures than gray-level interpolation. This is because, in our view, the new method approximates object shape more accurately than the method commonly used.

It interpolates the object rather than an image of the object.

4) The method leads to improved display of static objects, particularly when the slice spacing is large (more than six times the pixel size).

Acknowledgment

The authors are grateful to Dr. W. Barrett of Brigham Young University, Provo, UT, and Dr. L. Burk of Thomas Jefferson Medical College, Philadelphia, PA, for providing cardiac and pelvic data sets.

References

[1] G. T. Herman, Guest Editor, *Proc. IEEE*, Special Issue on Computerized Tomography, vol. 71, Mar. 1983.

[2] J. K. Udupa, "3D imaging in medicine," in *Proc. 9th Annu. Conv. Exposition, Nat. Comput. Graphics Ass.*, vol. II, Philadelphia, PA, 1987, pp. 73-104.

[3] J. C. Mazziotta and K. H. Huang, "THREAD (three-dimensional reconstruction and display) with biomedical applications in neuron ultrastructure and computerized tomography," *Amer. Fed. Inform. Processing Soc.*, vol. 45, pp. 241-250, 1976.

[4] G. T. Herman and H. K. Liu, "Three-dimensional display of human organs from computed tomograms," *Comput. Graphics Image Processing*, vol. 9, pp. 1-29, 1979.

[5] H. Fuchs, Z. M. Kedem, and S. P. Uselton, "Optimal surface reconstruction for planar contours," *Commun. ACM*, vol. 20, pp. 693-702, 1977.

[6] J. K. Udupa, "Interactive segmentation and boundary surface formation for 3D digital images," *Comput. Graphics Image Processing*, vol. 18, pp. 213-235, 1982.

[7] L. T. Cook, S. J. Dwyer, III, S. Batnitzky, and K. R. Lee, "A three-dimensional display system for diagnostic imaging applications," *IEEE Comput. Graphics Appl.*, vol. 3, pp. 13-19, Aug. 1983.

[8] P. Dev, S. L. Wood, J. P. Duncan, and D. N. White, "An interactive graphics system for planning reconstructive surgery," in *Proc. 5th Annu. Conv. Exposition, Nat. Comput. Graphics Ass.*, Chicago, IL, June 1983, pp. 130-135.

[9] P. B. Heffernan and R. A. Robb, "A new method for shaded surface display of biological and medical images," *IEEE Trans. Med. Imaging*, vol. MI-4, pp. 26-38, 1985.

[10] L. D. Harris, R. A. Robb, T. S. Yuen, E. L. Ritman, "Non-invasive numerical dissection and display of anatomic structures using computerized X-ray tomography," *Proc SPIE*, vol. 152, pp. 10-18, 1978.

[11] E. J. Farrel, "Color display and interactive interpretation of three-dimensional data," *IBM J. Res. Develop.*, vol. 27, pp. 356-366, 1983.

[12] G. Frieder, D. Gordon, and R. A. Reynolds, "Back-to-front display of voxel-based objects," *IEEE Comput. Graphics Appl.*, vol. 5, pp. 52-60, 1985.

[13] D. Meagher, "Geometric modeling using octree encoding," *Comput. Graphics Image Processing*, vol. 19, pp. 129-147, 1982.

[14] K. H. Hohne and R. Bernstein, "Shading 3D images from CT using gray-level gradients," *IEEE Trans. Med. Imaging*, vol. MI-12, pp. 252-255, 1985.

[15] R. A. Reynolds, D. Gordon, and L. S. Chen, "A dynamic screen technique for shaded graphics display of slice-represented objects," *Comput. Vision, Graphics, Image Processing*, vol. 38, pp. 275-298, 1987.

[16] R. A. Drebin, L. Carpenter, P. Hanrahan, "Volume rendering," *Comput. Graphics*, vol. 22, no. 4, pp. 65-74, 1988.

[17] M. Levoy, "Display of surfaces from volume data," *IEEE Comput. Graphics Appl.*, pp. 29-37, May 1988.

[18] G. T. Herman, S. S. Trivedi, and J. K. Udupa, "Manipulation of 3D imagery," in *Progress in Medical Imaging*, V. L. Newhouse, Ed. New York: Springer-Verlag, 1988.

[19] C. Cutting, D. D. S. Grayson, F. Bookstein, L. Fellingham, and J. G. McCarthy, "Computer-aided planning and evaluation of facial and orthognathic surgery," *Comput. Plastic Surg.* vol. 13, pp. 449-462, 1986.

[20] S. S. Trivedi, G. T. Herman, and J. K. Udupa, "Segmentation into three classes using gradients," *IEEE Trans. Med. Imaging*, vol. MI-5, pp. 116-119, 1986.

[21] C. N. de Graaf, S. M. Pizer, A. Toet, J. J. Koenderink, P. P. Zuidema, and P. P. van Rijk, "Pyramid segmentation of medical 3D images," in *Proc. Int. Joint Alpine Symp. Med. Comput. Graphics Image Commun. Clin. Adv. in Neuro. CT/NMR*, IEEE Comput. Soc. (IEEE Catalog No. 84CH20006-5), Feb. 1984, pp. 71-77.

[22] J. K. Udupa, G. T. Herman, P. S. Margasahayam, L. S. Chen, and C. R. Meyer, "3D98: A turnkey system for the display and analysis of 3D medical objects," *SPIE Proc.*, vol. 671, pp. 154-168, 1986.

[23] S. M. Goldwasser, R. A. Reynolds, D. Talton, and E. Walsh, "Real-time display and manipulation of 3D CT, PET, and NMR data," *SPIE Proc.*, vol. 671, pp. 139-149, 1986.

[24] S. L. Wood, "Surface definition of 3D objects from CT images," in *Proc. 8th Ann. Conf. IEEE Eng. Med. Biol. Soc.*, vol. 2, Dallas-Fort Worth, TX, 1986, pp. 1118-1121.

[25] D. Levin, "Multidimensional reconstruction by set-valued approximations," *IMA J. Numer. Anal.*, vol. 6, pp. 173-184, 1986.

[26] D. Gordon, and J. K. Udupa, "Fast surface tracking in three-dimensional binary images," *Comput. Vision, Graphics, Image Processing*, to be published.

[27] G. T. Herman, R. A. Reynolds, and J. K. Udupa, "Computer techniques for the representation of three-dimensional data on a two-dimensional display," *SPIE Proc.*, vol. 367, pp. 3-14, 1982.

[28] D. Gordon, and R. A. Reynolds, "Image-space shading of three-dimensional objects," *Comput. Vision, Graphics, Image Processing*, vol. 29, pp. 361-376, 1985.

[29] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*. Boston, MA: Prindle, Weber & Schmidt, 1978.

[30] D. P. Boyd and M. J. Lipton, "Cardiac computed tomography," *Proc. IEEE*, vol. 71, pp. 298-307, Mar. 1983.

[31] R. A. Robb, E. A. Hoffman, L. J. Sinak, L. D. Harris, and E. L. Ritman, "High-speed three-dimensional X-ray computed tomography: The dynamic spatial reconstructor," *Proc. IEEE*, vol. 71, pp. 308-319, Mar. 1983.

[32] S. B. Wu and W. X. Lu, "Surface reconstruction of 3D objects in computerized tomography," *Comput. Vision, Graphics, Image Processing*, vol. 44, pp. 270-278, 1988.

[33] R. J. P. Kehtarnavaz and de Figueiredo, "A framework for surface reconstruction from 3D contours," *Comput. Vision, Graphics, Image Processing*, vol. 42, pp. 32-47, 1988.

[34] J. D. Boissonnat, "Shape reconstruction from planar cross sections," *Comput. Vision, Graphics, Image Processing*, vol. 44, pp. 1-29, 1988.

[35] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Comput. Graphics*, vol. 21, no. 4, pp. 163-168, 1987.

[36] D. Vandermeulen, P. Suetens, J. Gybels, and A. Oosterlinck, "Surface reconstruction from planar cross sections for 3-D integrated representation of medical images," in "Advances in image processing," *Proc. SPIE*, vol. 804, pp. 183-190, 1987.

[37] A. Shaw and E. L. Schwartz, "Construction of polyhedral surfaces from serial sections: Exact and heuristic solutions," *Proc. SPIE*, vol. 1091, pp. 221-233, 1989.