Imaging Transforms for Visualizing Surfaces and Volumes

Jayaram K. Udupa and Roberto J. Gonçalves

Three-dimensional (3D) visualization in biomedical and other imaging areas is a rapidly emerging discipline. The major developments in this field are described in a unified and concise way. To this end, we introduce an operator notation to describe the basic imaging transforms commonly used in 3D visualization and to identify a comprehensive set of basic transforms. We also introduce several new basic transforms for filtering and interpolating scenes and structures and for rendering surfaces and volumes. We demonstrate not only how the existing visualization methodologies can be described concisely, but we also show how a great variety of new methodologies can be generated using both the existing imaging transforms and the new transforms introduced in this paper. A comprehensive evaluation method to compare objectively rendering methods used in visualization based on task-specific mathematical phantoms is described. We examine in detail separate transform sequences that are best suited for rendering robust and frail structures (ie, structures with well- and poorly defined boundaries).
Copyright © 1993 by W.B. Saunders Company

KEY WORDS: 3D imaging, visualization, surface rendering, volume rendering, evaluation of visualization.

THREE-DIMENSIONAL (3D) imaging in medicine encompasses visualization, manipulation, and analysis of structure information captured in 3D (and higher dimensional) digital images. In recent years, this activity has become an established discipline in medical imaging. The fast pace of development in this field is evidenced by the frequency of the conferences held and the papers published on this subject. As further testimony to the brisk activity in this field, three books[1-3] have been published over a period of 1 year on this subject.

The subject of this paper is mainly *visualization,* ie, processes relating to how the 3D structural information may be presented on a computer display screen to a human observer. Although *manipulation,* relating to how structures may be altered, and *analysis* concerning how structures may be quantified are equally important operations, there is significantly less published work on these topics than on visualization.

Visualization methods may be grouped into two classes: *surface* and *volume rendering* methods. Although this nomenclature is widely accepted in the literature, the meaning that these phrases convey is not unique. We may consider all methods in which a surface representation of the structure is first created and, subsequently, a rendition of the structure is generated by rendering the surface elements as forming the *surface rendering* class. Similarly, all methods that render the entire volume (region) occupied by the structure by rendering the volume elements forming the region can be considered to belong to the *volume rendering* class. In both, the basic elements are assigned an opacity value and renditions are created by determining the intensity of light reaching the points (pixels) in a viewing plane under certain assumed illumination conditions. In volume rendering, it is not necessary for each volume element to have a fixed opacity value (in fact, the power of volume rendering comes from such a consideration). However, every surface element in a given surface in surface rendering should necessarily have a fixed opacity value (usually 100% unless the structure is viewed with other structures and it is desirable to make it semitransparent). Surface rendering thus appears to be a special case of volume rendering. In a strict mathematical sense, however, a surface occupies zero volume (although the volume enclosed by the surface is usually nonzero), and the particularity becomes questionable. In the digital situation, surface elements each with zero volume (eg, voxel faces) or with nonzero volume (eg,

---

From the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA.

Supported by NIH grants no. CA56071 and CA50851 and an NSF grant no. IRI90-13341. An early version of this paper was presented at the SPIE Medical Imaging Conference at Newport Beach in 1992 whose proceedings contained a 3-page summary of this paper. The principles described here form the design framework of a software system, called 3DVIEWNIX, for multimodality 3D imaging, developed in the authors' department.

Address reprint requests to Jayaram K. Udupa, PhD, Medical Image Processing Group, Department of Radiology, Blockley Hall, Fourth Floor, 418 Service Dr, Philadelphia, PA 19104-6021.

voxels themselves) can be used to represent a surface.[4,5] Therefore, whether or not this particularity becomes meaningful depends on the situation. Conversely, we may use a set of surfaces, eg, using voxel faces as surface elements, to represent a volume, by making each surface an isodensity surface for a distinct image density value. To render the volume, we may render the collection of surfaces assigning a fixed opacity value to each surface.[6] Of course, between them the surfaces may have different opacity values. When surface rendering is viewed in this more general manner, the distinction between the two methodologies starts disappearing. Without digressing further, we assume a somewhat general definition of the two methodologies, a rendering method that requires the explicit creation of a surface description will be called surface rendering, and those that do not, will be classified under volume rendering.

This argument brings out an important point relating to one of the main purposes of this article. It hints at borrowing an important idea from volume rendering into surface rendering. In biomedical visualization, often the functionally independent operations in a rendering method are integrated among themselves or with the rendering method for computational efficiency. This is especially true in computer implementations. Yet, freeing the individual basic operations from the idiosyncrasies of the rendering method and viewing them in a more general setting can lead to powerful functionalities that result from combining these operations in ways that have not been envisaged when they are first designed. The first objective of this article is to demonstrate the richness resulting from a bewildering array of rendering methods that can be generated when the individual operations are identified appropriately and combined properly. Because the effectiveness of a rendering method often depends on the data, the usefulness of the richness of the environment needs no overstatement. We introduce an operator notation to describe concisely the individual basic operations and the resulting rendering methods. We do not attempt an exhaustive examination of the possible sequences of operations (ie, rendering methods) simply for want of space. We introduce several new operations and demonstrate the improved renditions resulting from sequences using such operations and describe many new rendering methods with potential for improved rendition.

This discussion naturally leads us to the question as to how rendering methods are to be compared. This forms the second topic of discussion in this article. One method of objective comparison of renditions is to use observer studies.[7] One main difficulty with observer studies is that they are expensive. Most rendering methods have a number of independent parameters. The selection of optimal values (optimal from the point of view of the underlying medical question whose answer is being sought through imaging) for these parameters for a given rendering method is itself an evaluation problem. If the studies to compare rendering methods have to additionally take into account these variabilities, the number of variables involved will be so large that conducting observer studies becomes impractical. Much sifting of the parameters and methods should be performed first using observer-independent techniques, and observer studies should be used only as the final arbiter of competing methods. The approach we suggest consists of using carefully prepared mathematical phantoms, putting them through image reconstruction processes emulating the image acquisition situation as realistically as possible,[8] applying the 3D operator sequence to the resulting data, and then quantitatively comparing those aspects that determine the appearance of specific object features in renditions with the truth. We will see that such quantification of visual entities and subsequent comparison of rendering methods using derived numbers allow a close scrutiny of individual operations used in rendering.

BASIC TRANSFORMS

This section describes the basic transforms that have been used commonly in 3D imaging and introduces a few new transforms. Each transform will be identified with an operator. The operators are generic in the sense that they do not distinguish between the different variants of the same class of transforms. For example, 3D image interpolation is a transform that converts one 3D image into another with volume elements of possibly a different size. It will be represented by a single operator, although many interpolating functions, linear, bilinear, trilinear, and various cubic forms, can be used with the transform. We will begin by defining some basic terms.

In visualization, the 3D image data available to us constitute a digitized and quantized representation of a vector-valued function of three independent variables. In medical imaging, typically the function is scalar valued and represents the distribution of some anatomic or physiological property value over a 3D region of the body. This function is called a "body function," and the 3D region is assumed to be a cuboid of finite size in Euclidean 3-space. The digitization operation can be considered to partition this cuboid into smaller cuboids by three sets of mutually orthogonal planes. The quantization operation discretizes the aggregate property value within each smaller cuboid into one of a finite set of numbers. We call the small cuboids "voxels," the quantized value associated with them their "density," the set of all smaller cuboids together with their densities a "scene," the (bigger) cuboid region the "scene region," and the set of all smaller cuboids the "scene domain."

We associate a fixed "scene coordinate system" (x^s, y^s, z^s) with the scene to describe the location of the voxels in the scene, and determine a fixed "imaging device coordinate system" (x^d, y^d, z^d) with respect to which the position and orientation of the scene coordinate system is described. The scene coordinate system is chosen so that its axes are parallel to the edges of the cuboid representing the scene region.

We consider a scene D to be a pair (V, f), where V is the scene domain, and f, called the "density function," is a mapping that assigns to every voxel in V a number in the closed interval $[L, H]$ called the "density range," where L and H are real numbers. When $L = 0$ and $H = 1$ are the only numbers in the range, we call D a "binary scene." Otherwise, D will be referred to as a "grey-level scene." V essentially represents a "digitization" of the scene region.

The voxels in V are, sometimes, not of identical size. But usually all voxels have identical, square cross sections in planes parallel to the $x^s y^s$-plane, and voxels with the same z^s coordinate for their centers have, in addition, the same size in the z^s direction. Therefore, voxels in V can be completely specified by the coordinates of their centers together with the size of the square cross section (usually referred to as "pixel size") and the voxel size in the z^s direction for each layer of voxels with the same z^s coordinate of their centers. We call the subset of voxels of V with a fixed z^s coordinate of their centers, together with their densities, a "slice" of D. If the subset constitutes the *i-th* layer, the corresponding slice will be referred to as the *i-th* slice.

The set of all scenes will be called the "scene space." Note that the scenes in a scene space need not have the same size, orientation with respect to the imaging device coordinate system, or digitization for their scene domains. The purpose of acquiring scene data is usually to study the form and function of certain structures, called "objects," about which information is captured in the scene. Objects are described geometrically via their boundary or region representations as sets of points, line segments, area elements, or volume elements. Associated with every object is an "object coordinate system" (x^o, y^o, z^o) whose location and orientation relative to the scene coordinate system, and, hence, to the imaging device coordinate system is known. We call the set of all objects the "object space." One of the main goals of visualization is to create two-dimensional (2D) images, called "renditions," depicting some 3D information of interest captured in a given scene. We consider a rendition to be a pair (P, g) where P, a rectangular array of pixels, is the "rendition domain," and g, an "intensity function," associates with every pixel in P a scalar-valued intensity from a grey scale given by $\{0, 1, \ldots, U\}$ or a vector-valued intensity from a color scale $\{0, 1, \ldots, U_r\} \times \{0, 1, \ldots, U_g\} \times \{0, 1, \ldots, U_b\}$. We call the set of all renditions the "view space" and associate with every rendition a "view coordinate system" (x^v, y^v) whose location and orientation relative to the scene or object coordinate system is known. Imaging transforms discussed in this section transform information from one space to the other among scene, object, and view spaces.

In the following discussion, we group basic imaging transforms into two classes (1) scene-related, which operate mainly on scenes; (2) structure-related, which operate mainly on objects extracted from scenes.

Scene-Related Transforms

The transforms described in this section are applicable to both grey-level and binary scenes although their applicability to binary scenes is sometimes trivial. Throughout we denote the input and output scenes by $D = (V, f)$ and $D' = (V', f')$, their density range by $[L, H]$ and $[L', H']$, and the scene space by $\mathbf{D}$.

Volume of interest (A). It is a scene space to scene space transform

$$A : \mathbf{D} \rightarrow \mathbf{D} \tag{1}$$

such that $V' \subset V$, $[L', H] = [L, H]$, and f' is a restriction of f to V'. This is often the first in a sequence of operations, and its purpose is to reduce the size of data that need to be processed subsequently to create a rendition.

Filtering (F_s). It may be described most generally by the transform

$$F_s : \mathbf{D} \rightarrow \mathbf{D} \tag{2}$$

such that $V' = V$. The form of f' is determined by the nature of the filter. Many forms of smoothing and enhancing filters have been used in picture processing,[9] which readily generalize to 3D scenes. We consider one example of a (Gaussian) smoothing filter that will be used in the next section:

$$f'(v) = \sum_{v' \epsilon N(v)} h(v' - v) f(v') \tag{3}$$

where $N(v)$ is the subset of voxels of V in an appropriate (say, the $3 \times 3 \times 3$) neighborhood of v (including v), and for all $t = (t_1, t_2, t_3) \in R^3$

$$h(t) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{t_1^2 + t_2^2 + t_3^2}{2\sigma^2}\right). \tag{4}$$

The role of scene filtering in visualization has not been studied much. As will be demonstrated in the next section, it is a powerful operation that forms a bridge between surface and volume rendering methodologies.

Interpolation (I_s, I_b). It is a scene space to scene space transform in which the scene regions of input and output scenes remain the same but their domains differ, ie,

$$I_s : \mathbf{D} \rightarrow \mathbf{D} \tag{5}$$

such that $V' \neq V$ (except when I_s is an identity operator) and f' is a sampling of an interpolant of f. Various forms of interpolating functions have been used to determine the density values of voxels in V': unilinear,[10] bilinear, trilinear,[11-14] various forms of unicubic,[13,14] bicubic and combination of linear and cubic forms.[15] An example of the latter form, which we call "bilin-

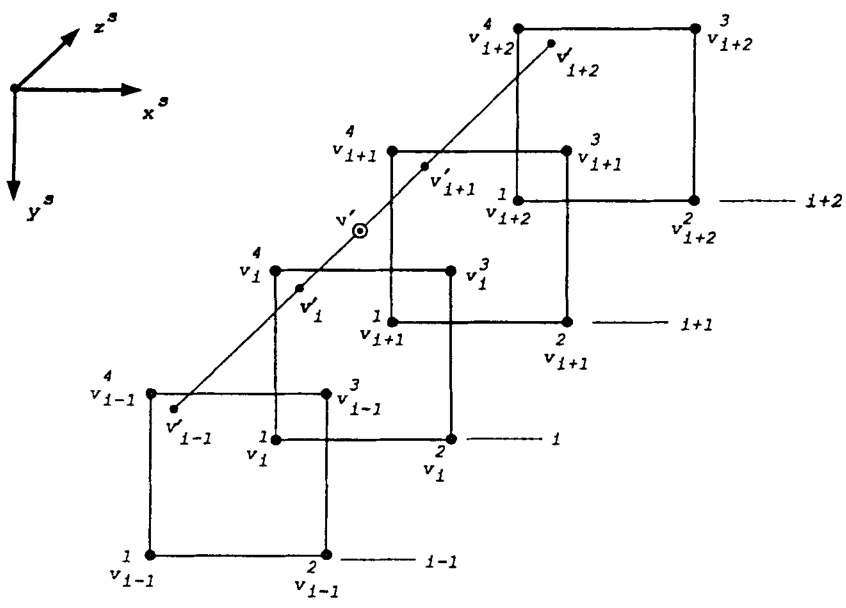


Fig 1. Illustration of bilinear-cubic density interpolation. The density of voxel v' (only the centers of voxels are shown to simplify drawing) of the output scene is determined by unicubic interpolation of the estimated densities at points v'_i, v'_{i-1}, v'_{i+1}, and v'_{i+2}. Each of these densities is determined in turn via bilinear interpolation of the densities of four voxels whose centers are closest to these points in the input scene.

ear-cubic," is illustrated in Fig 1. For simplicity we have shown only the centers of relevant voxels: $v' \epsilon V'$ represents (the center of) the voxel whose density is to be determined; v'_{i-1}, v'_i, v'_{i+1}, v'_{i+2} are the projections of the center of v' onto the two closest slice planes (a slice plane is the plane that contains the centers of all voxels in the slice) of D in the $+z^s$ and $-z^s$ directions relative to v'; v^k_t, $1 \leq k \leq 4$, are the four voxels in V closest to the point v'_t, for $i - 1 \leq t \leq i + 2$. The density value at v'_t is determined via bilinear interpolation of the four voxel densities of v_t. The density of v' is determined by a cubic spline type of interpolation of the values at v'_t: The interpolating function is such that its values at v'_i and v'_{i+1} equal the densities at v'_i and v'_{i+1}, respectively, and its derivative at v'_t, for $t = i, i + 1$, is (density at v'_{t+1} − density at $v'_{t-1})/(z^s_{t+1} - z^s_{t-1})$, z^s_{t+1} and z^s_{t-1} being the z^s coordinate of v'_t and v'_{t+1}, respectively. Imaging transforms using this form of interpolation will be illustrated in later sections.

Recently, binary scene interpolation methods have been investigated, and a method called "shape-based interpolation"[13] has been introduced and its variants[16,17] have been developed. These methods (and other shape-interpolation methods[18]) have been shown to produce more accurate results than grey-level scene interpolation in many situations. The basic idea is to convert the given binary scene into a grey-level scene and then to interpolate the later. The conversion is done by assigning to every voxel in the scene domain the shortest distance to the voxel from the boundary between 0- and 1-voxel regions in the binary scene, the distance being taken to be positive for 1-voxels (voxels with density 1) and negative for 0-voxels (voxels with density 0). The distance is usually computed within the same slice that contains the voxel in question and from the 2D boundary in that slice (although there is no reason why this cannot be generalized to a 3D distance). A practical motivation for shape-based interpolation comes from situations in which interactive outlining on slices is the only possible method of identifying objects. Clearly, object identification followed by interpolation would save a great deal of user time, especially when dealing with four-dimensional (4D) scenes, compared with interpolation followed by object identification. We denote shape-based interpolation by the operator I_b

$$I_b : \mathbf{D_b} \rightarrow \mathbf{D_b}, \tag{6}$$

where $\mathbf{D_b}$ represents the set of all binary scenes. We will indicate later how other operators can be combined with the scene interpolation operator I_s to give effects similar to that of I_b.

Segmentation (S_b, S_g). We identify two types of segmentation operators:

$$S_b : \mathbf{D} \rightarrow \mathbf{D_b} \tag{7}$$

called "binary segmentation," is such that $V' = V$ and $[L', H']$ is the set $\{0, 1\}$.

$$S_g : \mathbf{D} \rightarrow \mathbf{D_p} \tag{8}$$

called "grey segmentation," is such that $V' = V$ and $[L', H']$ is $\{0, 1\}$. Here $\mathbf{D_p}$, a subset of the scene space, is the set of all scenes with voxel densities in the interval $\{0, 1\}$.

Thresholding is the most commonly used binary segmentation operator, which can be defined by

$$f'(v) = \begin{cases} 1 \text{ if } f(v) \epsilon [T_l, T_h] \\ 0 \text{ otherwise,} \end{cases} \tag{9}$$

where T_l and T_h are fixed numbers. Clustering based on multiparametric scene data (ie, scenes in which voxel density is vector valued as in magnetic resonance [MR] imaging of the brain using multiple MR properties) is another popular method of binary segmentation. This approach seems to be capable of reproducible results with proper user training.[19-23] Binary segmentation being as difficult as it is, and yet a very crucial operation in visualization, manipulation, and analysis, a variety of general methods, as well as application-specific approaches, are being investigated (see the articles on segmentation in Ezquerra et al[24]).

Grey segmentation is a generalization of binary segmentation; density is assigned to voxels in the output scene that reflects the likelihood of the voxel being in the structure that is being segmented. In the terminology of pattern classification,[25] treating segmentation as a voxel classification problem, this likelihood can be chosen to be a function of the distance from the decision boundary. For example, in thresholding, the decision boundary is given by $f(v) = T_l$ and $f(v) = T_h$, and the distance function may be chosen to be as shown in Fig 2.[26] This idea can be generalized to higher-dimensional feature spaces, in particular to the clustering approaches previously mentioned. It should be emphasized that the introduction of fuzziness in the result of segmentation by itself does not make the segmentation problem easier to solve. It only allows a flexibility to retain the uncertainties inherent in the scene data in the segmentation result, but the problem itself remains as difficult as binary segmentation.

Masking (M_s). The purpose of masking is to clip the input scene so that objects of no interest are excluded in the clipped scene, or by applying a segmentation technique to the clipped scene, the object(s) of interest may be identified automatically. Masking is a scene space to scene space transform

$$M_s : \mathbf{D} \rightarrow \mathbf{D} \tag{10}$$

such that $V' = V, H' = H$, and L' is any number less than L, and

$$f'(v) = \begin{cases} f(v) & \text{if } v \in V_1 \subset V \\ L' & \text{otherwise.} \end{cases} \tag{11}$$

V_1 may be specified in a variety of ways. Usually this is done interactively,[27,28] by having the user either draw or paint the region occupied by V_1 slice-by-slice or cut the scene domain via some form of 3D display. Automatic mask generation to aid in segmentation seems simpler than solving the segmentation problem itself in many

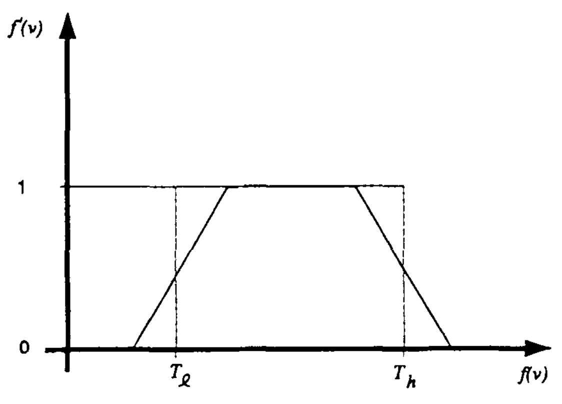


Fig 2. Grey segmentation via thresholding. T_l and T_h are fixed lower and upper thresholds which determine a decision boundary in a 1D feature space. $f'(v)$ is a function of the distance of the density of v from the decision boundary.

situations. This does not seem to have been attempted.

Structure definition (O_s, B_s). The purpose of these transforms is to generate a 3D object or surface description from the given scene. These are scene space to object space transforms

$$\begin{aligned} O_s &: \mathbf{D} \rightarrow \mathbf{Q} \\ B_s &: \mathbf{D} \rightarrow \mathbf{S} \end{aligned} \tag{12}$$

where $\mathbf{Q}$, a subset of the object space, is the set of all object descriptions (in the form of region representation), and $\mathbf{S}$, also a subset of the object space, is the set of all surface descriptions. A variety of methods (representing O_s) is available for obtaining object descriptions from scenes, which differ mainly in how the objects are represented, simply as an array of voxels constituting the object,[29-33] or other specialized schemes that encode sets of voxels via segment end,[34,35] skewed array,[36] semiboundary,[37] or various forms of octree representations.[38-40] A greater variety of techniques (representing B_s) is available for obtaining boundary surface descriptions from scenes, including methods of surface detection in grey-level scenes,[4,15,41-48] and surface tracking/formation in binary scenes.[5,49-54] Some methods express the surface as a connected set of faces of cubic voxels. Others approximate the surface via polygonal elements, in particular, triangles. Some of these methods are guaranteed to produce closed surfaces, and some produce nonclosed surfaces. The closure of surfaces is important because otherwise they do not correspond to physically realizable objects. Closure is essential in some of the new structure-related transforms introduced in the next section.

Classification (C). This transform creates a (opacity) scene from a given scene by assigning opacity values to voxels:

$$C : \mathbf{D} \rightarrow \mathbf{D}_p. \tag{13}$$

Here $V' = V$, $[L', H'] = \{0, 1\}$ (note that this represents an interval of real numbers) and $\mathbf{D}_p$ is the set of all scenes with density range $[0, 1]$. The purpose of this transform is to create a semitransparent volume that may be subsequently volume rendered.

To classify adequately to subsequently create accurate renditions is not an easy task. In fact, this is as difficult a problem as segmentation. As we outlined under Segmentation, classification is indeed analogous to grey segmentation. The existing classification techniques[12,26] have drawbacks similar to those of the simple segmentation techniques.

Volume rendering (R_{vs}, R_{vbs}, R_{vbo}, R_{vo}, R_{vbv}). Given an object description either explicitly, eg, an array of binary-valued voxels, or implicitly as an opacity scene, volume rendering to create a rendition of the object consists of two distinct operations: projection and pixel intensity calculation. Projection is performed via either ray casting[12,55-60] or direct projection of object elements.[26,29,30,32-40,61] Pixel intensity calculation is performed by a variety of methods that differ mainly in how the unit normal vector to the alleged object surface in the volume is estimated. The methods of projection differ mainly in their computational requirements. Differences in the quality of renditions attributable to the methods of projection are subtle. Although ray-casting allows more complex illumination models than voxel projection, it has not been demonstrated that the added complexity improves diagnostic information. The quality of rendition is determined mainly by the method of estimating the unit normal vectors. Therefore, in the following definition of volume rendering transforms we consider only this latter aspect and ignore the differences because of the method of projection.

Depending on the space in which the data used for estimating the normal vector are defined, we classify normal estimation methods as scene space, object space, and view space methods. Consider the scene density function

$$w = f(v) = f(x^s, y^s, z^s) \tag{14}$$

where we have represented voxel v by the coordinates of its center (x^s, y^s, z^s) in the scene coordinate system. If f is known at every point within the scene region and if its partial derivatives exist at (x^s, y^s, z^s), then its gradient at that point is given by

$$\nabla w = i\frac{\partial f}{\partial x^s} + j\frac{\partial f}{\partial y^s} + k\frac{\partial f}{\partial z^s}. \tag{15}$$

Scene space methods operate on the principle that if the object boundaries are characterized by sharp changes in voxel density, then the direction of most rapid density change at points in the vicinity of the boundary should be approximately normal to the boundary. (For isodensity surfaces, it is a known mathematical fact that the gradient vector is normal to the boundary at points on the boundary.) For high-contrast boundaries (eg, bone surfaces in scenes obtained from computerized tomography [CT]) these conditions are mostly well satisfied, and, therefore, unit vector in the direction of ∇w forms a good estimate of surface normal.[57] Because f is not known at every point in the scene region, an appropriate interpolating function may be used instead of f in Equation (15), or as usually done, the partial derivatives are approximated by central differences of voxel densities in the three directions.[12,15,26,31,43-45,57] What forms a good approximation to ∇w at points on (near) isodensity surfaces is an interesting problem that is yet to be fully explored. We now give an example of an approximation that we will use later.

To estimate ∇w at any point $P = (x_o^s, y_o^s, z_o^s)$ in the scene region (not necessarily voxel centers), we determine 26 neighboring points $P_1, \ldots, P_{26}$ forming the vertices, centers of edges, and centers of faces of an $\alpha \times \alpha \times \alpha$ cube centered at P with its edges parallel to the scene coordinate axes. The density at each neighboring point is determined using the bilinear-cubic scene interpolation method described earlier (Fig 1). Let L_{x^s}, be the set of straight line segments l such that l passes through P, l does not lie in the plane $x^s = x_o^s$, and l's end points are in $\{P_1, \ldots, P_{26}\}$. Define L_{y^s}, and L_{z^s}, similarly. Then

$$\nabla w \approx iPD_{x^s} + jPD_{y^s} + kPD_{z^s} \tag{16}$$

where

$$PD_{x^s} = \sum_{l \in L_{x^s}} \frac{1}{|l|} CD(l) \tag{17}$$

$$PD_{y^s} = \sum_{l \in L_{y^s}} \frac{1}{|l|} CD(l) \tag{18}$$

$$PD_{z^s} = \sum_{l \in L_{z^s}} \frac{1}{|l|} CD(l) \tag{19}$$

and $|l|$ and $CD(l)$ is the central difference (from the positive side of the plane to its negative side) of the density of the two end points of l.

Volume rendering methods employing scene space normal estimation may be characterized by the following transforms:

$$R_{vs} : \mathbf{D}p \times \mathbf{D} \rightarrow \mathbf{V} \tag{20}$$

$$R_{vbs} : \mathbf{D_b} \times \mathbf{D} \rightarrow \mathbf{V} \tag{21}$$

where **V** denotes the view space. R_{vs} represents rendering methods that use nonbinary volumes (opacity scenes)[12,26,55,58-61] and R_{vbs} represents rendering methods that use binary volumes.[29-40] R_{vs} and R_{vbs} both require the given scene for estimating the surface normal (Equation 15) in addition to an appropriate object description (an element of $\mathbf{D_p}$ for R_{vs} and of $\mathbf{D_b}$ for R_{vbs}).

Object space methods estimate normals based on the local geometry of the surface in the object/surface description.[10,38,62-66] Rendering methods using such normal estimation techniques have been used to render mainly binary volumes. They may be characterized by the following transform:

$$R_{vbo} : \mathbf{D_b} \rightarrow \mathbf{V}. \tag{22}$$

Object space methods can also be used to render nonbinary volumes (we are not aware of any such method reported in the literature). One possible approach quantizes the density range of a given opacity scene into a finite number of discrete levels. (Because the density range of scenes usually consists of only integers, there is an obvious quantization.) We compute an isodensity surface for each of these discrete opacity values, estimate surface normals using object space methods for each surface, and then volume render the set of surfaces. Obviously a variety of methods can be used for both determining the surface and estimating the normals. Accordingly, such volume-rendering methods can be described by the following transform:

$$R_{vo} : \mathbf{D_p} \rightarrow \mathbf{V}. \tag{23}$$

View space methods operate on the assumption that the distance to the object surface from the viewpoint is known for all visible parts of the surface to be captured in the rendition. (This distance is available as a by-product, or otherwise easily calculated during the projection operation in both classes of projection methods.) If this distance map is expressed as the following function

$$z^v = d(x^v, y^v) \text{ or } w = d(x^v, y^v) - z^v, \tag{24}$$

then from Equation 15

$$\nabla w = i\frac{\partial d}{\partial x^v} + j\frac{\partial d}{\partial y^v} - k. \tag{25}$$

Because d is known only at pixel centers in the rendition domain, view space methods[67] use various approximations[65,68,69] to estimate the partial derivatives in Equation 25 to determine a unit vector in the direction of ∇w, which is taken to be the surface normal at the point (x^v, y^v, z^v).

View space normal estimation methods have been used to render only binary volumes. We characterize such rendering methods by the following transform:

$$R_{vbv} : \mathbf{D_b} \rightarrow \mathbf{V}. \tag{26}$$

It is not clear how the view space methods can be used to render nonbinary volumes especially because the normal estimated at a point on the surface depends on the viewpoint.

This concludes our discussion of scene-related transforms. Clearly, the filtering (F_s) and grey-level scene interpolation (I_s) operators can be modified to operate in view space on renditions. We denote these modified operators by F_v and I_v, respectively.

Structure-Related Transforms

The transforms described in this section are applicable to object descriptions.

Structure conversion (O_o, O_{so}, B_o, B_{os}). The purpose of these transforms is to convert one form of object/boundary description to another. We identify four types of transforms:

$$\begin{aligned} O_o &: \mathbf{Q} \rightarrow \mathbf{Q} \\ O_{so} &: \mathbf{S} \rightarrow \mathbf{Q} \\ B_o &: \mathbf{S} \rightarrow \mathbf{S} \\ B_{os} &: \mathbf{Q} \rightarrow \mathbf{S} \end{aligned} \tag{27}$$

O_o converts one form of object description into another, eg, a segment-end[34] or run-length code description into an octree.[38] O_{so} converts a surface description into an object description, eg, a discrete surface[50] into an octree.[39] B_o converts one form of surface description into another, eg, from a polygonal form into a triangular form.[52] This conversion is often useful because modern workstations have graphics engines that allow rapid rendering of triangles. Another class of methods exemplifying B_o is the so-called tiling approach. These methods convert a surface description in the form of a stack of contours defined on slices[70] into an enveloping surface by tiling triangular,[71-77] square[27] or other polygonal elements[78] between contours in successive slices. B_{os} converts an object description into a surface description, eg, an octree object representation into a surface from.[79]

The need for structure conversion operators comes from the fact that a form of object representation useful for one operation may not be appropriate for another. For example, a particular polygonal representation of a surface may be appropriate to create high quality renditions of a structure, yet if we wish to create a physical model of the surface, (ie, for manufacturing prosthesis), it has to be converted into a stack of contours for driving a numerically controlled milling machine.[80]

Gray transform (G_s, G_o). This new operation allows converting an object or surface description into a scene. Accordingly we have

$$\begin{aligned} G_o &: \mathbf{Q} \to \mathbf{D} \\ G_s &: \mathbf{S} \to \mathbf{D}. \end{aligned} \tag{28}$$

The scene domain V is such that the object/surface is properly contained in it. The density function f is such that $f(v) = H$ if $v \epsilon V$ is in the interior of the object/surface, and $f(v) = L$, if $v \epsilon V$ is not in the interior of the object/surface. As an example of G_o, suppose the object is simply a set Q of voxels. Then V is chosen such that $Q \subset V$ and f is given by

$$f(v) = \begin{cases} H & \text{if } v \epsilon Q \\ L & \text{if } v \epsilon V - Q \end{cases}. \tag{29}$$

As an example of G_s, consider a closed surface S. $G_s(S)$ is a scene (V, f) such that

$$f(v) = \begin{cases} d_1(v, S) & \text{if } v \text{ is in the interior of } S \\ -d_1(v, S) & \text{if } v \text{ is in } V \text{ but in the exterior of } S \\ 0 & \text{if } v \text{ is in } V \text{ and intersects } S, \end{cases} \tag{30}$$

where $d_1(v, S)$ is an appropriate measure of distance of v from S. This may be the 2D distance from S in the slice containing v or a 3D distance from S.

Clearly, the "interior" or a surface S to be well defined, it should be closed (ie, S should partition V into two subsets, an interior set and an exterior set, such that it is not possible to get to the exterior from the interior without crossing S. A precise definition of such notions in the digital space can be found elsewhere.[49,50,54] G_s is not defined for surfaces that do not satisfy this condition.

As we will demonstrate in the next section, these operations lead us to a variety of powerful operations on objects and surfaces including interpolation, filtering, and volume rendering.

Digitization (T_o, T_b, T_s). Often, parametrically defined continuous geometric objects have to be dealt in conjunction with digital entities, such as scenes, objects, and surfaces in applications, such as prosthesis design and radiation therapy planning.[80,81] Recently, algorithms have been developed[82,83] to digitize such continuous descriptions into digital object representations. Once the digital representations are obtained, the complete battery of structure-related operations are available to visualize, manipulate, and analyze these objects in conjunction with scene-derived objects and surfaces. This digitization operation may be characterized by the transforms

$$\begin{aligned} T_o &: \mathbf{Q}_c \to \mathbf{Q} \\ T_b &: \mathbf{S}_c \to \mathbf{S} \end{aligned} \tag{31}$$

where Q_c and S_c are respectively the set of all parametric object and surface descriptions.

We suggest that the digitization operation should attempt to retain the inaccuracies accompanying this operation in the digitized representation. If the digitized object is represented as a set of voxels, this retention can be performed naturally by assigning to voxels a membership value that reflects the fraction of the total voxel volume occupied by the continuous object in the voxel. Instead of assigning an all-or-none membership value,[82,83] the voxels now have values in the range [0, 1]. That is, we are creating a scene (much like in grey segmentation) from a continuous object or surface description:

$$T_s : \mathbf{Q}_c \cup \mathbf{S}_c \to \mathbf{D}_p. \tag{32}$$

If the continuous object/surface is a sphere, eg, the region of the resulting scene is a cube that encloses the sphere. The sphere is digitized so that the voxels that lie completely inside or outside the sphere are assigned the density 1 or 0, respectively, and those intersecting the boundary are assigned the fraction of the voxel volume interested by the sphere.

Surface rendering (R_{ss}, R_{so}, R_{sv}). Given an object/surface description explicitly as a (hard) set of volume or surface elements, these operations create renditions of the object/surface. (Note that rendering methods that operate on (hard) surfaces, as well as on the more redundant (hard) volumetric descriptions, are considered here. Recall that methods for rendering hard volumetric descriptions were considered under volume rendering also. This dichotomy reflects the unsettled nature of the otherwise well accepted terminology.) We identify three types of operations, scene space, object space and view space, depending on the method used for estimating the unit surface normal vector.

$$R_{ss} : (\mathbf{Q} \cup \mathbf{S}) \times \mathbf{D} \rightarrow \mathbf{V} \tag{33}$$

$$R_{so} : \mathbf{Q} \cup \mathbf{S} \rightarrow \mathbf{V} \tag{34}$$

$$R_{sv} : \mathbf{Q} \cup \mathbf{S} \rightarrow \mathbf{V}. \tag{35}$$

The three normal estimation methods are exactly as described under volume rendering. Again, as before, a projection or a ray-casting method may be used for creating projections of volume and surface elements.

This concludes our discussion of basic imaging transforms. In the next section, we shall examine how these basic operations may be combined to derive other more sophisticated composite operations, and, in particular, a host of 3D imaging methodologies.

DERIVED TRANSFORMS AND 3D IMAGING METHODOLOGIES

The operators defined in the previous section represent some of the basic imaging transforms used in visualization, manipulation, and analysis of 3D structures. They are generic, in the sense that each operator stands for a whole class of similar operators. In this section, we study (1) how these basic operators may be combined to generate useful composite operators that perform more sophisticated scene- and structure-related transforms, and (2) how the basic operators may be chained to produce a wide variety of complete 3D imaging methodologies, which when given a scene, enable us to create its renditions. We will not attempt to cover all possible composite transforms or complete imaging methodologies. Only examples illustrating the richness of the resulting environment will be given. We will identify specific complete imaging sequences that we have found to produce better renditions than commonly used approaches. We substantiate these improvements in the next section using our evaluation methods. In addition, we point out a host of new methodologies that seem to have the potential for improved rendition. We do not evaluate these methods because of the finiteness of the scope of the article, but invite the reader to dive into this expanse.

Composite Transforms

Scene preprocessing: I_sF_s, F_sI_s. We assume throughout that the order of operation is right to left. The output scene created by the above transforms for an input scene D is thus given by

$$D_1 = I_s(F_s(D))$$

$$D_2 = F_s(I_s(D))$$

Of course, generally, $D_1 \neq D_2$

Object/surface filtering: $O_sF_sG_o$, $B_sF_sG_s$. The first transform converts a given object $Q \epsilon \mathbf{Q}$ into a scene $G_o(Q)$, say, using equation 29 and then filters this scene. The filtered object Q' is determined from the filtered scene $F_s(G_o(Q))$. (If F_s is a smoothing filter, O_s may be a threshold operator, the actual threshold determined from a knowledge of L and H in equation 29.) The second transform converts a given surface $S \epsilon \mathbf{S}$ into a scene $G_s(S)$ (eg, using equation 30) and subsequently filters this scene. The filtered surface S' is determined from the filtered scene using an appropriate surface detector B_s. Note that although S' and a surface detected from Q' may represent the same underlying continuous surface, they may differ because of different filtering effects.

Object/surface interpolation: $O_sI_sG_o$, $B_sI_sG_s$. The basic idea is to convert the object/surface into a scene, then to interpolate the scene, and, subsequently, to extract the interpolated object/surface using object/surface definition operators. Conversion to scene may be performed using equation 29 or preferably 30. The latter actually allows capturing some aspects of the shape of the object/surface and therefore to interpolate the shape.

Both the previously mentioned class of filtering and interpolation operators are powerful when appropriately used and lead to significant improvements in visualization methods as we will later demonstrate.

Global/local segmentation: S_bM_s, S_gM_s. One of the most difficult tasks in scene segmentation is to design global criteria that can ward off errors caused by local decision making. One approach to specifying global guidelines (perhaps the most commonly used) is to interactively indicate a subset of V to which automatic segmentation must be confined.[27] When such a masking operation is implemented properly, user interation time required can be kept to a minimum (a shell around the boundary is often all that is needed). Recently, we have had considerable success in automatically finding globally optimal boundaries using 2D dynamic programming[84] when the search is confined to a mask. Because this method seems to work in a variety of different applications, automatic mask generation seems to be a useful problem to pursue.

Fuzzy boundary definition: $G_sB_sM_s$. Suppose M_s is such that V_1 in equation 19 is a shell around the boundary of interest and that B_s finds an optimal boundary[84] within the shell. Then G_s creates a scene $D_p \in \mathbf{D_p}$ such that voxels in V_1 are assigned the boundary likelihood value used in finding the optimal boundary and voxels not in V_1 are assigned a 0 value.

It is clear by now how other composite operators may be designed along similar lines. The study of the grammar[85] defining the permissible ways the basic operators may be combined is an interesting exercise.

Imaging Methodologies

We now describe transform sequences that define complete visualization methods. We first examine important classes of visualization methods that have already been reported and subsequently describe a variety of new methods.

Existing methodologies. Early methods developed for rendering scenes, objects, and surfaces may be described by the following operator sequences:

(1) $R_{so}B_sI_s$

(2) $R_{so}B_oB_s$
$R_{so}B_oB_sS_bI_s$

$R_{so}B_sS_bI_sA$
(3) $R_{so}B_sS_bF_sI_sA$
$R_{so}B_sS_bM_sI_s$

The method in expression 1[10] is based on scene interpolation, surface detection, and surface rendering using object space normals. Expressions in expression 2 represent methods based on contour tiling and object space normals for surface rendering.[27,71,73,86] Early software packages[87,88] for medical 3D imaging incorporated the approaches expressed in expression (3).[27,49,50,62,63]

In the quest for speed, a variety of rendering methods for binary volumes were subsequently developed. These can be described by the following expressions:

(4) $R_{so}O_sS_bI_s$

(5) $R_{so}B_sI_s$

$R_{vbv}S_bI_sA$
(6) $R_{vbv}O_sS_bI_s$
$F_vR_{sv}B_sS_bI_sA$

Expression 4 represents an early method that was incorporated into hardware[38] based on an octree representation of the binary segmented objects defined in interpolated scenes. Expression 5 represents a class of techniques based on contour-defined objects[70,89-91] that use object space normals. The idea of using the entire binary volume or some encoded version of it is expressed in the sequences shown under expression 6.[29-37] A view-space normal estimation method was introduced in this connection,[67] which has been subsequently modified by other investigators.[65,69]

Scene-space normal estimation was introduced by Höhne and Bernstein[57] for surface rendering. This rendering method, characterized by expression 7 used ray-casting and binary segmentation to determine object boundary locations. This normal estimation method has been used in later surface rendering techniques[43-46] (expression 8) and has also been improved further.[14,15,92]

(7) $R_{vbs}(S_bI_s, I_s)$

(8) $R_{ss}(B_sI_s, I_s)$.

Given a scene D, the previously mentioned (and similar) expressions should be interpreted as follows:

$$R_{vbs}(S_b I_s, I_s)(D) = R_{vbs}(S_b I_s(D), I_s(D))$$

$$R_{ss}(B_S I_s, I_s)(D) = R_{ss}(B_s I_s(D), I_s(D)).$$

Volume-rendering methods were introduced by Levoy[12] and Drebin et al.[26] They use scene space normals and can be described by the following expressions:

$$\begin{array}{ll} & R_{vs}(CI_s, I_s) \\ (9) & R_{vs}(I_s C, I_s) \\ & I_v R_{vs}(CI_s, I_s).^{93} \end{array}$$

Shape-based interpolation is a recent addition[13,16,17] to 3D imaging. It has been used mainly for surface rendering and the visualization methods investigated using this operator[13,69,94] are:

$$\begin{array}{ll} & F_v R_{sv} B_s I_b S_b M_s \\ (10) & R_{ss}(B_s I_b S_b, I_s) \\ & R_{ss}(O_s I_b S_b, I_s). \end{array}$$

The digitization operators may be used,[82,83] if mathematically defined objects/phantoms are to be rendered alone or in combination with the objects scanned by an imaging device. In the latter case, the two object/surface descriptions should be combined before rendering.

$$\begin{array}{ll} & R_{so} B_{os} T_o \\ (11) & R_{sv} T_b \\ & R_{so}(B_{os} T_o(Q_c) \cup B_{os} O_s I_s(D)). \end{array}$$

In expression 11, D and Q_c are a scene and a continuous object description, respectively.

New methodologies. It is clear how even confining to the operators used by the existing methodologies, it is possible to generate a variety of new methodologies. Some examples are given in 12.

$$\begin{array}{ll} & R_{vbv} I_b S_b \\ (12) & R_{vbo} I_b S_b \\ & R_{so} B_s I_b S_b. \end{array}$$

The use of the digitization operator T_s is illustrated in 13.

$$\begin{array}{ll} (13) & R_{ss}(B_s T_s, T_s) \\ & R_{vs}(T_s, T_s) \end{array}$$

The main thrust of this section is the use of operators F_s, G_o, and G_s in conjunction with other scene- and object-related operators. These three operators lead us to a host of new methodologies, each having its own interesting features, as illustrated later. We encourage the reader to examine the expressions representing these methodologies carefully because they embody a variety of ideas expressed compactly.

To systematize our discussion, we consider structures to be visualized as belonging to one of two classes—*robust* and *frail.* A structure is robust in a scene if (1) it has consistently well-defined boundaries in the scene, and (2) in the vicinity of the structure boundary, the scene density changes most rapidly normal to the boundary. A structure is frail in a scene if it is not robust in the scene. Bone in CT scans is a good example of a robust structure, although there are some aspects of bony structures, such as thin parts, that are closer to being frail than robust; we shall come back to this point later. Bone in MR images is an example of a frail structure. Frail structures usually do not satisfy both the conditions.

The main purpose of introducing the classification is to suggest that the transforms that prepare a structure for (volume or surface) rendering should be treated independently of the methods of estimating normals. This separation allows us to select the operations best suited for structure preparation and normal estimation separately, so that the resulting renditions are optimal for the given situation. For illustration, suppose we have a frail structure that can be segmented (either automatically or in the worst case, interactively) that does not satisfy condition number 2. Clearly, scene space normal estimation methods that use the given scene would generate unreliable normals. Nonetheless, we can create scenes using a combination of operators F_s, G_o, G_s on the segmented object, which can be used for scene space normal estimation that is more reliable. On the other hand, the segmented object may be filtered in a variety of ways (eg, to smooth it), and still the normals may be estimated from the given scene if condition 2 is satisfied. These points will become clearer in the following expressions. We treat robust and frail structures separately.

Robust structure rendering. Because robust structures satisfy condition 2, we retain the original scene for normal estimation but suggest that the structure itself may be prepared in numerous ways via combinations of interpolation, grey transform, and filtering operations to minimize digitization effects.

Use of F_s only

(14)
a. $R_{ss}(B_sF_sS_bI_s, F_sI_s)$
b. $R_{vbs}(S_bF_sS_bF_sI_s, I_s)$
c. $R_{vs}(F_sCI_s, I_s)$.

(15)
a. $R_{ss}(B_sI_bS_bF_s, I_s)$
b. $R_{ss}(B_sF_sI_bS_b, I_s)$
c. $R_{ss}(B_{os}O_sF_sI_bS_bF_s\,I_s)$
d. $R_{vs}(S_gF_sI_bS_b, I_s)$.

(16)
a. $R_{vo}F_sT_s$
b. $R_{vs}(T_s, F_sT_s)$.

Expressions in 14 illustrate how scene filtering may be used to filter a structure. Expressions 14*a* and *b* show how the result of binary segmentation may be filtered to create a filtered (hard) structure, whereas expression 14*c* suggests that the result of classification may be filtered to create a filtered version of fuzzily defined structures.

Expressions in 15 show how filtering may be combined with structure (shape-based) interpolation to create possibly even smoother representations of structures both for hard (a, b, c) and fuzzily defined (d) structures. The method in 15*d* is particularly interesting. It suggests that the shape interpolated binary scene on filtering (which results in a grey scene) be grey segmented, which essentially creates a shell around the object boundary, and be (grey) volume rendered.

Expressions in 16 show how filtering may be used in conjunction with fuzzily digitized continuous parametrically defined objects.

Use of G_o, G_s with F_s

(17)
a. $R_{ss}(B_sF_sG_oO_sI_s, I_s)$
b. $R_{ss}(B_sF_sI_sG_sB_s, I_s)$
c. $R_{vs}(S_gI_sF_sG_oO_s, I_s)$

(18)
a. $R_{ss}(B_sF_sG_oO_sI_bS_bF_s, I_s)$
b. $R_{vbs}(S_bF_sG_sB_sI_bS_b, F_sI_s)$
c. $R_{vs}(S_gF_sG_sB_sI_bS_bF_s, I_sF_s)$

(19)
a. $R_{so}B_sF_sG_sT_b$
b. $R_{vs}(S_gF_sG_oT_o, T_s)$.

Examples of G_s and G_o given in equations 29 and 30 allow us to create an effect similar to that of shape-based interpolation[13] when combined with F_s. Expressions 17*a* and *b* use G_o and G_s with F_s to smooth the surface, whereas 17*c* smooths the structure for volume rendering in a manner similar to 15*d*. Expressions in 18 combine the effects of G_o, G_s, and I_b (shape interpolation) together with scene filtering (F_s) to prepare the structure that is being rendered. 18*c* which is similar to 15*d* and 14*c* is an interesting volume rendering method. Expressions 19*a* and *b* show how G_s, G_o, and F_s may be combined for rendering digitized continuous objects.

Frail structure rendering. In methods for rendering robust structures, we concentrated mainly on how the structure may be appropriately prepared while relying mostly on scene space methods applied to the given scene for normal estimation. Frail structure rendering is more challenging because normals estimated in that manner are not reliable. The following expressions, therefore, emphasize the need to prepare the scene carefully if scene-space normal estimation were to be used, in addition to preparing the structure appropriately, as in robust structure rendering.

Use of F_s only

(20)
a. $R_{ss}(B_sF_sS_bI_s, F_sS_bI_s)$
b. $R_{vbv}(S_bI_sF_sS_b)$
c. $R_{vs}(S_gF_sS_bI_s, I_sF_sS_b)$

(21)
a. $R_{ss}(B_sI_bS_b, F_sI_bS_b)$
b. $R_{ss}(B_sS_bF_sI_bS_bF_s, F_sI_bS_b)$
c. $R_{so}(B_sF_sI_bS_b)$
d. $R_{vs}(S_gF_sI_bS_bU, F_sI_bS_b)$.

Expressions in 20 show how F_s alone may be used to prepare the structure, as well as create a scene artifically for computing scene space normals. Expression 20*c* is an interesting method that shows how a binary scene may be brought into the grey volume rendering paradigm by filtering the binary scene and subsequently doing grey segmentation and using the filtered binary scene for normal estimation. Expressions in 21 combine shape-based interpolation with F_s both for preparing the structure and for creating the scene used for normal estimation. View space and object space methods (20*b* and 21*c*) may also be appropriate for normal estimation Expressions 20*c* and 21*d* embody the shell idea referred to earlier for (grey) volume rendering.

Use of G_o, G_s with F_s

(22)
a. $R_{ss}(B_sF_sI_sG_oO_s, I_sF_sG_sB_s)$
b. $R_{ss}(B_sI_sF_sG_sB_sS_bU, F_sI_bS_b)$
c. $R_{vbv}(S_bF_sG_sB_sI_s)$
d. $R_{vs}(S_gF_sI_sG_sB_s, F_sI_sG_oO_sI_s)$

(23)
a. $R_{ss}(B_sF_sG_sB_sI_bS_bF_s, F_sF_s\, I_bS_b)$
b. $R_{sv}(B_sF_sG_oO_sF_sI_bS_b)$
c. $R_{vbs}(S_bF_sG_sB_sF_sI_bS_bF_s, F_sG_sB_sF_sI_bS_b)$
d. $R_{vs}(S_gF_sS_gF_sI_bS_bF_s, F_sG_oO_sI_bS_bF_s)$.

Expressions in 22 show how operators G_s and G_o may be combined with F_s for preparing the structure for rendering as well as for estimating normals from specially created scenes (22*a*, *b*, and *d*). Expressions in 23 show how G_o, G_s, and F_s may be combined with I_b for structure preparation and normal estimation.

Examples

We can realize in our current implementation many of the methodologies previously outined for specific instances of the individual generic operators. However, we will not attempt to illustrate every methodology with an example because of space limitations. Nevertheless, we intend to demonstrate the improvements that are possible both for robust and for frail structures compared with commonly used techniques. We have selected the new methods based on our own intitution and somewhat limited experience of their performance. We have not yet studied them systematically to grade them. To keep the number of images manageable, we have selected two scenes. One represents a robust structure (a dry skull in a 256 × 256 × 68 CT scene with identical voxels of size 0.8 mm × 0.8 × 3.0 mm), and the other represents a frail structure (one of the four bones (taulus) at the midtarsal joint of a normal human volunteer in an MR scene with 256 × 256 × 62 scene domain and identical voxels of size 0.7 mm × 0.7 mm × 1.5 mm). The methods selected are as follows:

Robust structure rendering:

R1. $R_{sv}B_sI_bS_b$
R2. $R_{ss}(B_sI_bS_b, I_s)$
R3. $R_{vs}(I_sC, I_s)$
R4. $R_{ss}(B'_sI_s, I'_s)$
R5. $R'_{ss}(B_sF_sI'_bS_b, I'_s)$

Frail structure rendering:

F1. $R_{ss}(B'_sI_s, I'_s)$
F2. $R_{sv}B_sI_bS_b$
F3. $R_{sv}B_sF_sS_bI_s$
F4. $R_{ss}(B_sS_bF_sI'_bS_b, F_sI'_bS_b)$
F5. $R_{vs}(CF_sI'_bS_b, F_sI'_bS_b)$
F6. $R_{ss}(B_sS_bF_sG_sB_sI'_bS_b, F_sG_sB_sI'_bS_b)$

The operators used in these methods are as follows:

I_s	trilinear scene interpolation
I'_s	bilinear-cubic scene interpolation (Fig 1)
I_b	shape-based binary scene interpolation using city block distance[13]
I'_b	shape-based binary scene interpolation using a more accurate approximation to Euclidean distance[16]
S_b	segmentation using thresholding
B_s	surface detection in binary scenes[50] with integer coordinate for boundary faces
B'_s	surface detection using a threshold in grey scenes with real coordinate for boundary faces,[15] the location of the face determined by computing exactly where the threshold is satisfied in a direction perpendicular to the face
C	classification based on voxel density and gradient magnitude[12]
R_{sv}	surface rendering using view space normals[69]
R_{ss}	surface rendering using scene space normals estimated from central differences of density at 6 points equally spaced from the center of the boundary face[94]
R'_{ss}	surface rendering using the scene space normal estimation method described in Equations 16 to 19
R_{vs}	volume rendering using scene space normals estimated as in R_{ss} (but from center of voxels)
F_s	scene filtering as in Equations 3 and 4
G_s	grey transform as in Equation 30 with 2D distance from boundary in the slice

Figures 3A to 3E show renditions created by methods R1 through R5 of the robust structure. R4 has been our best method for rendering robust structures, and we previously have shown[15] their rendition quality to be as good as that of high-quality grey-volume rendering techniques. R5 seems to be a further improvement of this method because it allows better approxi-

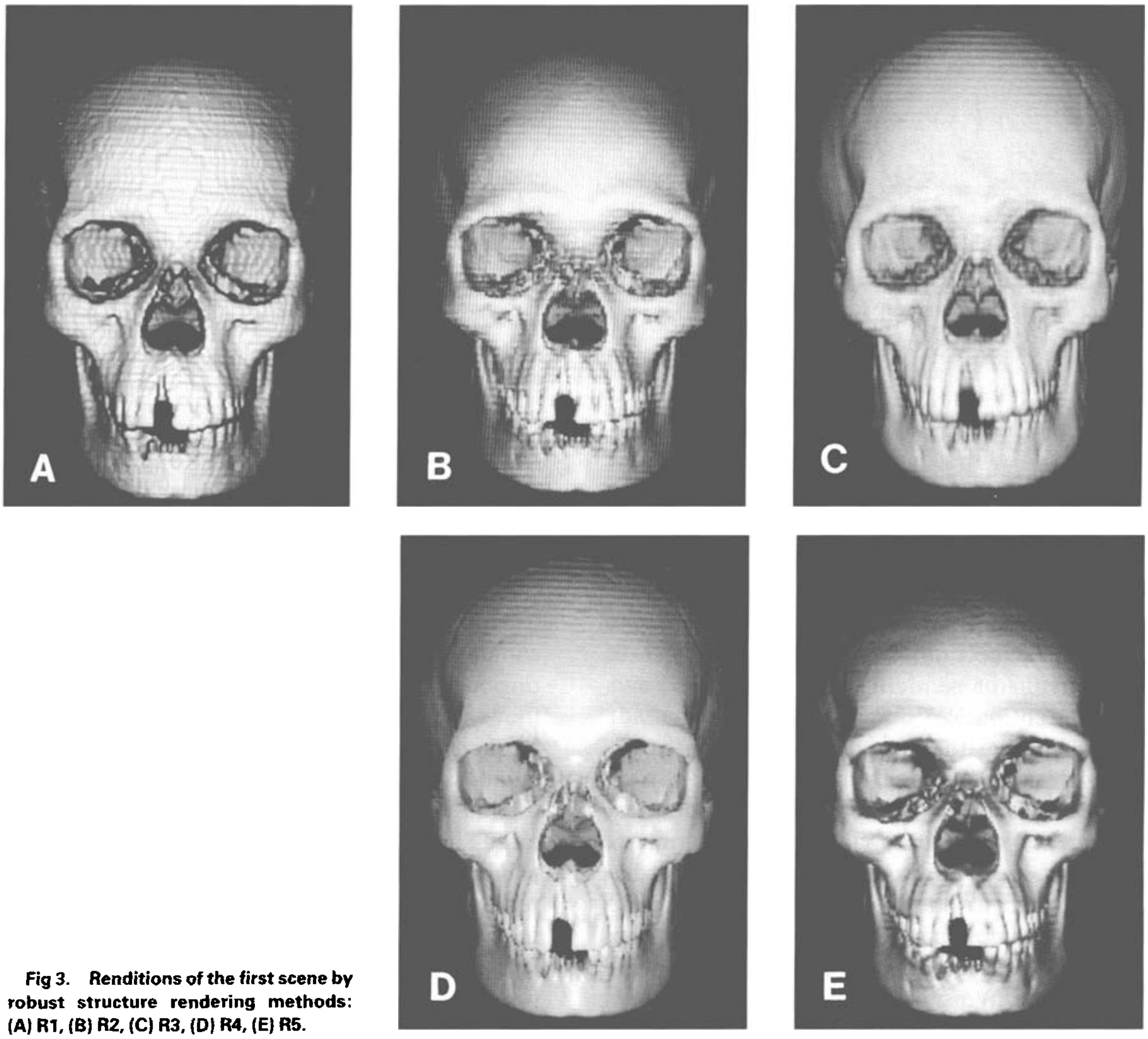


Fig 3. **Renditions of the first scene by robust structure rendering methods: (A) R1, (B) R2, (C) R3, (D) R4, (E) R5.**

mation of surface shape via a combination of shape-based interpolation and surface filtering. The "slicing artifact" seems to be reduced and some of the sutures (especially in the temporal region) seem to be better defined compared with the rendition created by R4 (objective criteria for comparing renditions are described in the next section). We expect a further improvement of R5 when B_s is replaced by B'_s in R5. Unfortunately, because our implementation in the existing programs are not designed around the basic operators, such modifications are not trivial. We will come back to this point later.

Figures 4A through 4F show renditions of the frail structure created by methods F1 through F6. Note that, although F1 is identical to R4, which undoubtedly produces excellent results for robust structures, it clearly produces the poorest rendition of the frail structure, mainly because the two conditions mentioned earlier that characterize robust structures are not fulfilled by the MR scene of the midtarsal joint. Of the remaining methods, F3 to F6 all have a clear advantage over F2, which shows a good deal of digital artifacts, which often camouflage real features. We will come back to these points in the next section in a more objective manner, but conclude this section with the statement that F5 seems to produce better renditions than F3, F4, and F6.

EVALUATION OF RENDERING METHODS

This is one area in medical 3D visualization that calls for considerable further work. Compar-

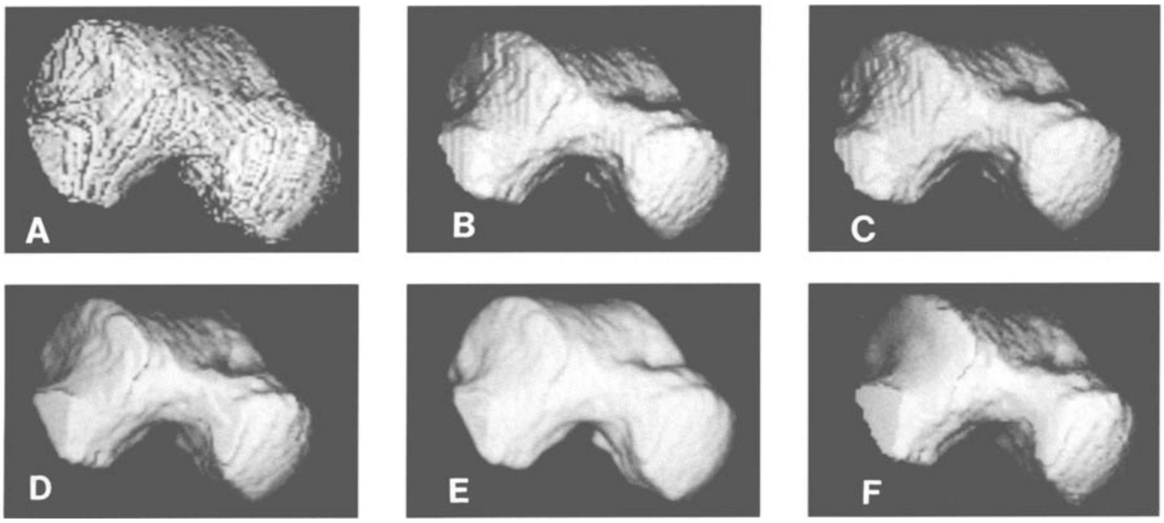


Fig 4. Renditions of the second scene by frail structure rendition methods: (A) F1, (B) F2, (C) F3, (D) F4, (E) F5, (F) F6.

isons are usually made visually as we have done in the last section. Recently, the subject of comparison of rendering methods has drawn some attention.[7,15,92,95-99]

Considering the genericity of the operators and the fact that each method given by an expression has itself a number of independent parameters whose optimal selection is often not trivial, the enormousness of the problem becomes clear. Conduction of observer studies[100] would be simply impractical at the global level of grading methods even for a specified small set of tasks. We suggest a two-tiered approach. Given the set of tasks that form the basis for comparison (eg, how well ridges, small holes, and fine separations [ie, sutures and fine fractures] are portrayed, how well smoothness of surfaces is retained), the first tier consists of creating mathematical phantoms that embody the task-related features, subjecting the phantoms to image reconstruction processes[101] to realistically simulate tomographic scanning, and then applying the visualization methods to be compared to the resulting scenes. Because reality is known precisely, how well it is portrayed in renditions can be quantified (as described later) and, therefore, rendering methods can be quantitatively compared objectively and graded, of course, for the chosen tasks. The second tier consists of resolution among closely competing methods via observer studies. The size of the problem will now have been reduced drastically because not only is the number of methods to be compared reduced, but optimal parameter settings will have been determined and, therefore, they will not enter into observer experiments.

There is a definite advantage to using mathematical phantoms instead of physical phantoms in the first tier. It may be difficult (even impossible) to quantify certain aspects of reality using physical phantoms. For example, as already observed, surface normal plays a crucial role in determining the quality of renditions, and, therefore, it is vital to be able to know true normals at points on the physical object surface. Still, it is difficult to establish a correspondence between points on the modelled surface and points on the physical surface, and worse even, to determine the true normal at points on the physical surface. Although this difficulty does not arise in mathematical phantoms, they have sometimes a different disadvantage in that it may not be possible to capture the reality adequately in the phantoms, either geometrically or in simulating the scanning process. Despite this drawback, we think that there are many situations in which comparison based on mathematical phantoms can shed light on the effectiveness of imaging transforms.

We concentrate on surface (and binary volume) rendering for quantifying the quality of portrayal and later indicate how the ideas may be generalized to nonbinary volume rendering methods. Our ideas are best described using the ray-casting paradigm. For a given viewing direction ω, imagine a ray emanates (orthogonal to the viewing plane) from the center of each pixel p in the rendition domain P into the modelled object/surface. We compute the value of a number of features, $\phi_1, \ldots, \phi_n$, of the object/surface at the point of entry of the ray into the object/surface. (When the ray does not intersect the object/surface, we assign a 0 value for each of the features.) For identical viewing conditions for the same ray, we determine the feature values for the mathematical object/surface. We define a disparity function $\delta_i(\phi_i)$ for each feature which expresses the disparity between the modelled and the true object/surface for each ϕ_i. (When the ray is nearly tangential to the surface, it may intersect only one among the modelled and mathematical surface and not both. Such rays clearly do not provide reliable disparity information.) Thus, for a given ω, we arrive at a *disparity image*

$$E_\omega(p) = \begin{bmatrix} \delta_1(\phi_1) \\ \vdots \\ \delta_n(\phi_n) \end{bmatrix}_{\omega,p}, \quad p \epsilon P. \tag{36}$$

We define *pose disparity* $\rho(\omega)$ to be the vector resulting from taking componentwise root mean squared value of the disparity image over all pixels:

$$\rho(\omega) = \begin{bmatrix} \rho_1(\omega) \\ \vdots \\ \rho_n(\omega) \end{bmatrix} = \begin{bmatrix} \sqrt{\frac{1}{n_p}\sum_p (\delta_1(\phi_1))^2} \\ \vdots \\ \sqrt{\frac{1}{n_p}\sum_p (\delta_n(\phi_n))^2} \end{bmatrix} \tag{37}$$

where n_p is the number of pixels in P. Finally, we define the *portrayal disparity* Δ of the given visualization method over a given solid angle Ω of viewing directions to be simply the componentwise integral of the pose disparity vector over all $w \epsilon \Omega$:

$$\Delta = \begin{bmatrix} \int_\Omega \rho_1(\omega)d\omega \\ \vdots \\ \int_\Omega \rho_n(\omega)d\omega \end{bmatrix}. \tag{38}$$

For better rendering methods, we expect E_ω, ρ and Δ to have componentwise smaller values.

Potential features are properties of the surface that are locally determinable at specified points, such as orientation of surface normal (consideration of normals for comparing renditions was suggested earlier in[92]), distance of the surface point from a fixed view point, and principal curvatures at the surface point or other properties derived from them such as mean and Gaussian curvatures. We illustrate the use of the first two features in this article. In our actual implementation, we have replaced the integral in Equation 38 by summation. We have used the following disparity functions for the two features:

$$\delta_1(\phi_1) = \sin\frac{\phi 1}{2}$$

$$\delta_2(\phi_2) = \frac{\phi_2}{\phi_{2\max}} \tag{39}$$

where ϕ_1 is the angle between the normal to the modelled and to the mathematical surface, and ϕ_2 is the distance between the two surfaces along the ray. $\phi_{2\max}$ is a scaling constant that we have chosen to be the diameter of the sphere that encloses the phantom.

It is not always necessary to compute pose disparity and portrayal disparity over the whole rendition domain as pointed out earlier. In fact, it may be useful to get a close scrutiny of specified regions of interest in P as to how good portrayal is in those regions. For example, we may be interested in determining how good a method is in portraying ridges. Once the disparity images for desired ω are computed, the user may outline a region of interest containing ridges in appropriate renditions of the modeled object/surface. The computation of pose and portrayal disparity confined to this region will indicate the effectiveness of the method in displaying ridges. This approach also alleviates the difficulty associated with considering those pixels in the disparity image whose rays are almost tangential to the object/surface.

It is much more difficult to devise evaluation methods for nonbinary volume rendering approaches because there are no obvious physical models corresponding to the commonly used volume rendering techniques. We suggest a generalization to the previously mentioned approach (we have not yet implemented this generalization). Continuing with the ray-casting paradigm, we determine the values of the feature along each ray at the most likely surface point (of the modelled object/surface) that is closest to the viewpoint and proceed as before. Alternatively, we may determine the disparity function value $\delta_i(\phi_i)$ at each sampled point along the ray and take a weighted sum of the disparity values, the weight chosen to be an appropriate function of the opacity value of the point. The first approach seems more reasonable, although we cannot give sound justification to either of these methods.

The mathematical phantom we created (Fig 5) consists of a cylindrical base with a hemispherical top. The base consists of a number of curved and straight slits (mimicking suture and fracture lines) of varying width (1, ½, ⅓, and ¼ pixel) and small structures of circular cross section (mimicking gyrations). The hemisphere has four small spherical holes (2, 1, ½, and ¼ pixel diameter) at its base (mimicking foramina).

Simulation of x-ray projection and image reconstruction is done using the SNARK89 package[8] developed in our group. The scanner geometry, noise characteristics, resolution and other properties are chosen to mimic closely the data collection and reconstruction processes of the General Electric CT/T 8800 scanner.[101] Figure 5A and B shows a rendition of the true geometrical phantom obtained via ray casting and using the Phong shading model[102] and Fig 5C shows a reconstructed cross section through the cylindrical base of the phantom.

Renditions of the phantom created by methods R1 through R5 are shown in Fig 6A through E. Disparity images for the first view in Fig 6A, B and E are shown in Fig 7. Figure 8 shows pose disparity as a graph for the methods illustrated in Fig 7 for the different views in a complete 360° rotation. Viewing angles shown in Fig 8 (and all similar graphs) are illustrated in Fig 5C. Figure 9 shows pose disparity in ϕ_1 (normal) separately for the upper spherical part and the lower cylindrical part for the different views in a complete 360° rotation for the three methods. Table 1 lists portrayal disparity for the three methods over the views illustrated in Fig 8 and 9. We also performed several "region-of-interest" portrayal disparity analysis by selecting rectangular regions in P in the region of one of the foramina, straight and curved slits, and of the gyrations. The results are summarized in Table 2 for the three methods for these regions.

Clearly R2 and R5 produce more accurate estimation of surface normal and location than R1. R5 produces more accurate normals than R2 for smooth regions of the surface with little detail, as shown in Fig 8A (roughly from 210° to 90°, see Fig 5C) and 9A. Its accuracy in estimation of location is comparable to that of R2 for smooth aspects of the surface (Fig 8B). However, in corrugated regions and regions with subtle details (especially between 90° to 180°; Fig 5C), R5 is inferior to R2 in the estimation of surface normal and location (Fig 8B and 9B). The reason for this behavior is that, in such

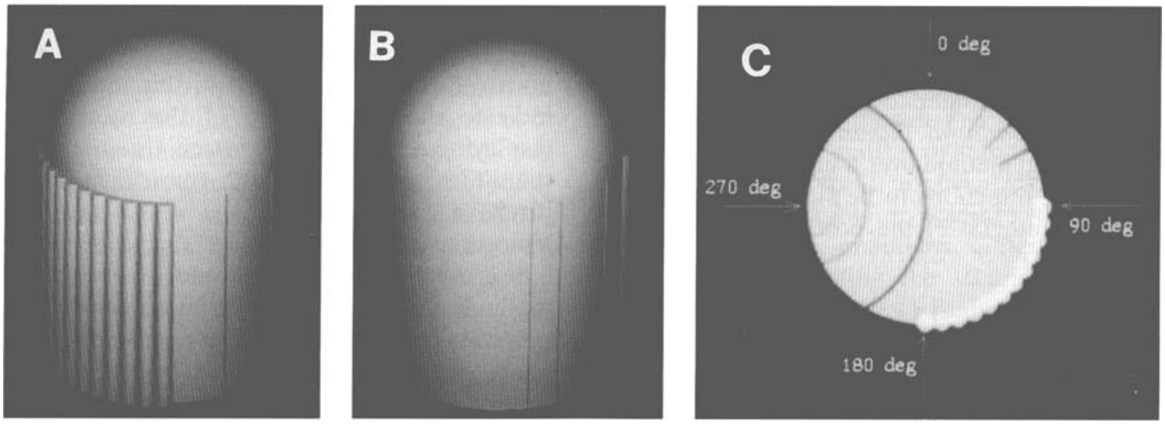


Fig 5. The mathematical phantom used in our evaluation. (A, B) renditions of two views (roughly at 90° and 0°, see C) of the true geometrical phantom by ray tracing. (C) A reconstructed cross section through the lower cylindrical base part of the phantom. The cross sections in this cylindrical part are more-or-less identical. The upper part of the phantom consists of a hemisphere with small holes near its base.

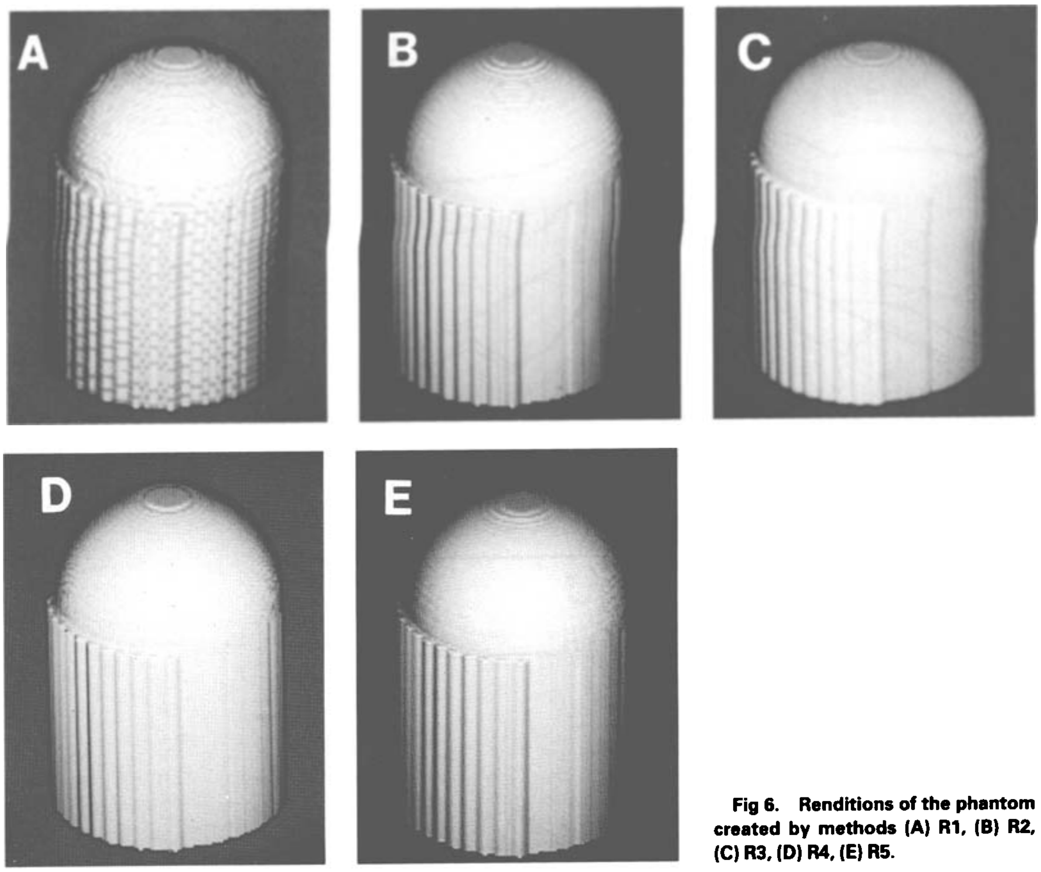


Fig 6. Renditions of the phantom created by methods (A) R1, (B) R2, (C) R3, (D) R4, (E) R5.

regions, the composite operator $B_s F_s I'_b S_b$ tends to shift the surface from its true location (because of the filtering effect), and, therefore, in addition to location, the estimated normal can be expected to be wrong. This phenomenon is further illustrated in Table 2, which lists portrayal disparity for regions with small details.

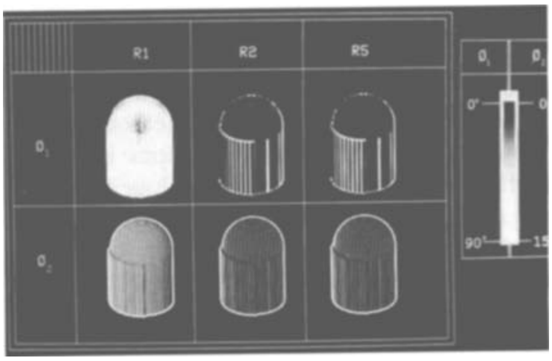
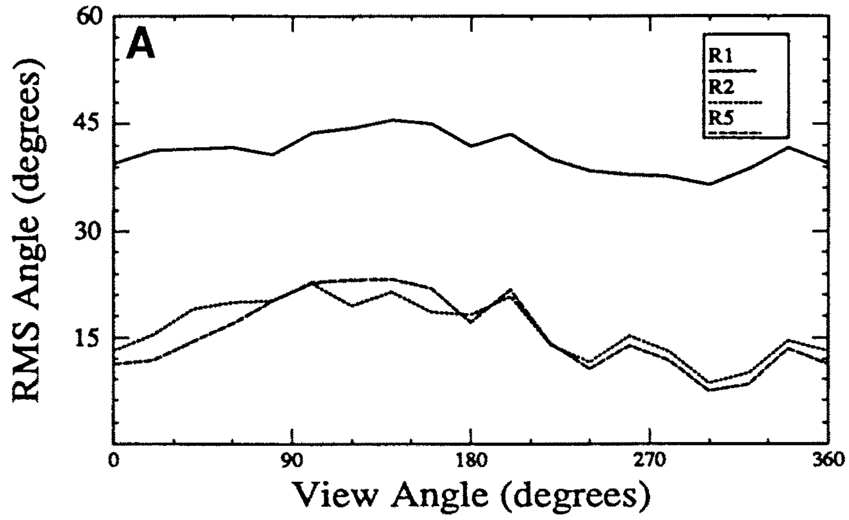


Fig 7. Disparity images corresponding to the first view in Figure 6 for methods (A) R1, (B) R2, (C) R5, with disparity in ϕ_1 shown at the top and in ϕ_2 at the bottom.

Combining the best features of R2 and R5 we can create the following method: $R_{ss}(B_s I_b S_b, I'_s)$. This method should produce overall better surface normal and location than R2 and R5. Unfortunately, our current implementation does not allow the flexibility needed in producing such composite methods. (It is also caused by an implementation issue that we did not include R4 in our comparison of pose disparity, although we believe that its performance (as measured by pose disparity) will be comparable to that of the method expressed by the previous sequence. We will come back to this issue in the next section). In answer to this flexibility issue, we have been developing a data-, machine-, and application-independent software system, called 3DVIEWNIX,[103,104] for the visualization, manipulation, and analysis of multidimensional images, whose design is based on the principles described in this article.

The phantom used in this section represents a robust structure; therefore, the analysis and the



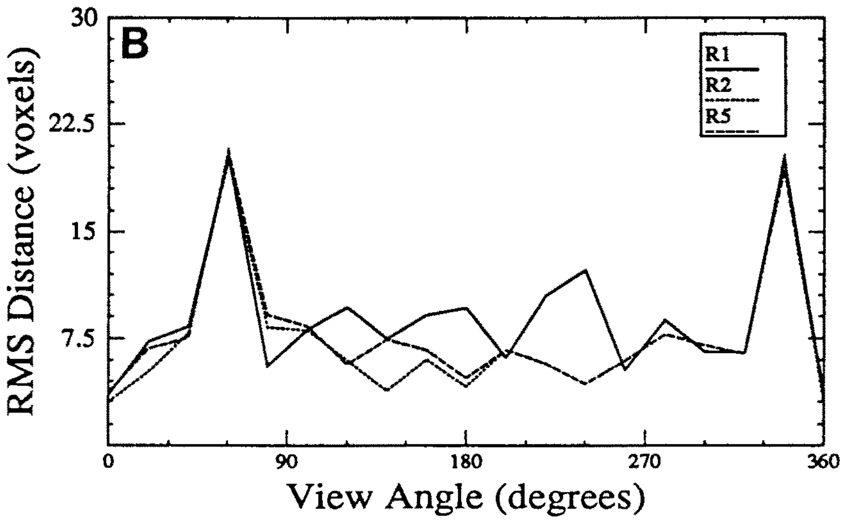


Fig 8. Pose disparity for a complete 360° rotation for the methods R1, R2, and R5. (A) RMS angle versus view angle. (B) RMS distance versus view angle.

associated remarks previously made apply mostly to robust structures. However, a similar methodology can be used to construct phantoms depicting frail structures and to evaluate the effectiveness of rendering methods for such structures.

CONCLUSION

An operator notation to describe concisely the basic 3D imaging transforms commonly used in biomedical 3D visualization was introduced, and a comprehensive set of basic transforms was identified. We have described several new basic transforms for filtering and interpolating structures and scenes and for rendering surfaces and volumes. We have also demonstrated the power of the principle of treating 3D imaging methodologies comprised of an appropriate combination of the basic operators. We have shown how such a treatment leads to a great variety of new rendering methods, and have demonstrated how many such methods can lead to improved portrayal.

A comprehensive evaluation method to compare objectively rendering methods based on mathematical phantoms was developed, making the grading of rendering methods for specified medical tasks possible. We have developed separate transform sequences to optimally render robust and frail structures (ie, structures represented in scenes with well- and ill-defined boundaries, respectively). Of the methods we have evaluated, those labeled R5 and F5 seem to produce the best portrayal of robust and frail structures, respectively.

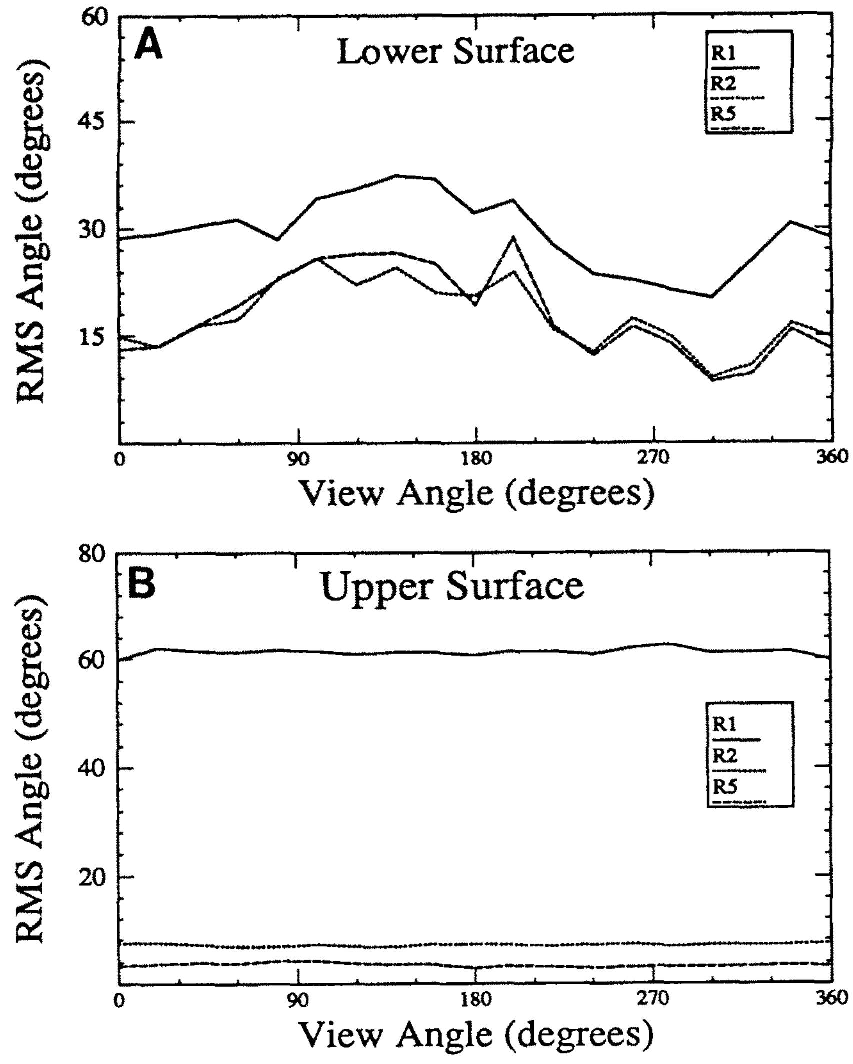


Fig 9. Regional pose disparity for a complete 360° rotation for methods R1, R2 and R5. (A) Lower surface. (B) Upper surface.

When a rendering method is expressed as a sequence of operators, there is no implication that that is how the method should be implemented. Often it is computationally more efficient to combine basic operators in the implementation. Consider method F3 as an example. It can be implemented in two modules, one doing $B_sF_sS_bI_s$ and the other R_{sv}. Clearly, because the determination of surface location using B_s is based on local information in the scene, and the computation of the result of F_s, S_b and I_s also depends only on local information, all four operators can be combined in the implementation as follows. The surface detection operator B_s starts from a specified initial surface element and looks for neighboring elements. In determining the actual location of these new elements we need to determine the

Table 1. Portrayal Disparity Over the Views Illustrated In Fig 8 for Methods R1, R2, and R5

	Entire Surface		Upper Surface	Lower Surface
Method	ϕ1	ϕ2	ϕ1	ϕ2
R1	39.6	9.9	62.12	29.80
R2	15.6	8.6	7.19	18.22
R5	15.9	8.9	3.45	19.00

Table 2. Portrayal Disparity in "Regions-of-Interest" for Methods R1, R2, and R5

	Foramen		Straight Slits		Gyrations		Curved Slits	
Method	ϕ1	ϕ2	ϕ1	ϕ2	ϕ1	ϕ2	ϕ1	ϕ2
R1	19.2	3.87	29.48	8.57	32.22	7.8	25.9	6.75
R2	12.46	2.92	17.42	7.72	25.27	3.85	13.74	6.47
R5	16.9	3.85	19.2	7.47	31.26	7.67	9.73	6.82

density of a few voxels in the neighborhood. Each of these densities can be determined by interpolation, segmentation and filtering of a few voxel densities in a small neighborhood. Thus, the operations I_s, S_b, and F_s can be confined to a thin shell around the surface that is being detected. Clearly, this is vastly more efficient than applying each of the operators on the whole scene because most of the voxels interior or exterior to the surface do not enter into any computation. However, the flexibility of combining the individual operators to realize other rendering methods is lost in the efficient implementation. One solution to this dilema is to identify the basic and often-used composite operators (ie, $B_sF_sS_oI_s$) and have each of them efficiently implemented as a single operator. These composite operators now become available for chaining with other basic or composite operators, providing a rich and powerful environment retaining efficiency.

The principles outlined in this article form the design framework underlying a software system called 3DVIEWNIX[103,104] that is being developed in our group.

ACKNOWLEDGMENT

We thank Ms. Hsiu-Mei Hung for assistance with some of the volume rendering programs and to Mary A. Blue for typing the manuscript.

REFERENCES

1. Udupa JK, Herman GT: 3D Imaging in Medicine. Boca Raton, FL, CRC Press, 1991

2. Kaufman A: A Tutorial on Volume Visualization. Los Alamitos, CA, IEEE Computer Society Press, 1990

3. Höhne KH, Fuchs H, Pizer SM: 3D Imaging in Medicine, Algorithms, Systems, Applications. Berlin, Germany, Springer-Verlag, 1990

4. Liu HK: Two- and three-dimensional boundary detection. Comput Graph Image Process 6:123-134, 1977

5. Udupa JK, Srihari SN, Herman GT: Boundary detection in multidimensions. PAMI 4:41-50, 1982

6. Herman GT, Udupa JK: Display of multiple surfaces, *in Proceedings, World Federation of Nuclear Medicine and* Biology. Third World Congress, Paris, 1982. New York, NY, Pergamon Press, pp 2165-2168

7. Vannier MW, Hildebolt CF, Marsh JL, et al: Craniosynostosis: Diagnostic value of three-dimensional CT reconstruction. Radiology 173:669-673, 1989

8. Herman GT, Lewitt RM, Odhner D, et al: SNARK89—A programming system for image reconstruction from projections. Technical Report MIPG160, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA, November 1989

9. Hall EL: Computer Image Processing and Recognition. New York, NY, Academic Press, 1979, pp xxx-xxx

10. Herman GT, Liu HK: Three-dimensional display of human organs from computed tomograms. Comput Graph Image Proc 9:1-29, 1979

11. Goldwasser SM, Reynolds RA, Talton D, et al: Real-time display and manipulation of 3-DCT, PET, and NMR data, in Nalcioglu O, Cho ZH, Budinger TF (eds): SPIE Proceedings, vol 671. Newport Beach, CA, 1986, pp 139-149

12. Levoy M: Display of surfaces from volume data. IEEE Compu Graph Appl 8(3):29-37, 1988

13. Raya SP, Udupa JK: Shape-based interpolation of multidimensional objects. IEEE Trans Med Imaging 9:32-42, 1990

14. Chuang KS, Udupa JK, Raya SP: High-quality rendering of discrete three-dimensional surfaces. Technical Report MIPG130, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA, July 1988

15. Udupa JK, Hung HM, Chuang KS: Surface and volume rendering in 3D imaging: A comparison. J Dig Imaging 4:159-168, 1991

16. Herman GT, Zheng J, Bucholtz CA: Shape-based interpolation. IEEE Comput Graph Appl 12(3):69-79, 1992

17. Higgins WE, Morice C, Ritman EL: Shape-based interpolation technique for three-dimensional images, in Proceedings of IEEE 1990 International Conference on Acoustics, Speech and Signal Processing, Albuquerque, NM, April 3-6, 1990, pp 1841-1844

18. *Lin WC, Liang CC, Chen CT: Dynamic elastic* interpolation for 3-D medical image reconstruction from serial cross sections. IEEE Trans Med Imaging MI-7:225-232, 1988

19. Vannier MW, Butterfield RL, Rickman DL, et al: Multispectral magnetic resonance image analysis. CRC Crit Rev Biomed Engin 15:117-144, 1987

20. Kohn MI, Tanna NK, Herman GT, et al: Analysis of brain and cerebrospinal fluid volumes with MR imaging. Part I. Methods, reliability and validation. Radiology 178: 115-122, 1991

21. Rusinek H, de Leon MJ, George AE, et al: Alzheimer's disease: Measuring loss of cerebral gray matter with MR imaging. Radiology 178:109-114, 1991

22. Tanna NK, Kohn MI, Horwich DN, et al: Analysis of brain and cerebrospinal fluid volumes with MR imaging: Impact on PET data correction for atrophy. Part II. Aging and Alzheimer dementia. Radiology 178:123-130, 1991

23. Snell JW, Merickell MB: Tissue labelling of MRI using recurrent artificial neural networks, in 13th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, November 1991, pages 66-67

24. Ezquerra N, Garcia E, Arkin ER: Proceedings of the First Conference on Visualization in Biomedical Computing, IEEE Computer Society, Los Alamitos, CA, May 1990

25. Duda RO, Hart PE: Pattern Classification and Scene Analysis. New York, NY, Wiley & Sons, 1973, pp 10-36

26. Drebin RA, Carpenter L, Hanrahan P: Volume rendering. Comput Graph 22:65-74, 1988

27. Udupa JK: Interactive segmentation and boundary surface formation for 3D digital images. Comput Graph Image Proc 18:213-235, 1982

28. Robb RA, Barillot C: Interactive display and analysis of 3-D medical images. IEEE Transact Med Imag 8:217-226, 1989

29. Frieder G, Gordon D, Reynolds RA: Back-to-front display of voxel-based objects. IEEE Compu Graph Appl 5:52-60, 1985

30. Farrell EJ, Yang WC, Zapulla RA: Animated 3D CT imaging. IEEE Compu Graph Appl 5:26-32, 1985

31. Schlusselburg DS, Smith W, Woodward DJ, et al: Use of computed tomography for a three-dimensional treatment planning system. Compu Med Imag Graph 12:25-32, 1988

32. Yasuda T, Hashimoto.Y, Yokoi S, et al: Computer system for craniofacial surgical planning based on CT images. IEEE Trans Med Imaging 9:270-280, 1990

33. Ylä-Jääski J, Klein F, Kübler O: Fast direct display of volume data for medical diagnosis. CVGIP: Graph Mod Image Proc 53:7-18, 1991

34. Trivedi SS: Interactive manipulation of three-dimensional binary scanners. Vis Comp 2:209-218, 1986

35. Reynolds RA, Gordon D, Chen LS: A dynamic screen technique for shaded graphics display of slice-represented objects. Computer, Vision, Graphics, Image Proc 38:275-298, 1987

36. Kaufman A, Bakalash R: Memory and processing architecture for 3-D voxel-based imaging. IEEE Comp Graph Appl 8:10-23, 1988

37. Udupa JK, Odhner D: Fast visualization, manipulation, and analysis of binary volumetric objects. IEEE Comp Graph Appl 11(6):53-62, 1991

38. Meagher D: Geometric modeling using octree encoding. Comp Graph Image Proc 19:129-147, 1982

39. Gargentini I, Atkinson HH, Schrack GF: Multiple-seed 3D connectivity filling for inaccurate borders. CVGIP: Graph Mod Image Proc 53(6):563-573, 1991

40. Wilhelms J, Gelder AV: Octrees for fast isosurface generation. Comp Graph 24(5):57-62, 1990

41. Zucker SW, Hummel RA: An optimal three-dimensional edge operator, in Proceedings of the IEEE Computer Science Conference on Pattern Recognition and Image Processing. 1978, pp 162-168

42. Morgenthaler D, Rosenfeld A: Multidimensional edge detection by hypersurface fitting. PAMI 3:482-486, 1981

43. Cline HE, Lorenson WE, Ludke S, et al: Two algorithms for the three-dimensional reconstruction of tomograms. Med Phys 15:320-327, 1987

44. Baker HH: Building, visualizing and computing on surfaces of evaluation. IEEE Compu Graph Applica 8(4):31-41, 1988

45. Wyvill G, McPheeters C, Wyvill B: Data structures for soft objects. Vis Compu 2:227-234, 1986

46. Payne BA, Toga AW: Surface mapping brain function on 3D models. IEEE Comp Graph Appl 10(2):41-53, 1990.

47. J.D. Cappelletti and A. Rosenfeld. Three-dimensional boundary following. Computer, Vision, Graphics, Image Proc 48(1):80-92, 1989

48. Wallin A: Constructing surfaces from CT data. IEEE Com Graph Appl 11(6):28-33, 1991

49. Artzy E, Frieder G, Herman GT: The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm. Comp Graph Image Proc 15:1-24, 1981

50. Gordon D, Udupa JK: Fast surface tracking in three-dimensional binary images. Computer, Vision, Graphics, Image Proc 45:196-214, 1989

51. Udupa JK, Ajjanagadde VG: Boundary and object labelling in three-dimensional images. Computer, Vision, Graphics, Image Proc 51:355-369, 1990

52. Kalvin AD: Segmentation and surface-based modeling of objects in three-dimensional biomedical images. New York, NY, Department of Computer Science, New York University, March 1991 (thesis)

53. Shu R, Kruger A: A 3D surface construction algorithm for volume data, in Proceedings of the International Computer Graphics Conference, Massachusetts Institute of Technology, Cambridge, MA, October 1991, pp 221-226

54. Rosenfeld A, Kong TY, Wu AY: Digital surfaces. CVGIP: Graph Mod Image Proc 53(4):305-312, 1991

55. Harris LD, Robb RA, Yuen TS, et al: Non-invasive numerical dissection and display of anatomic structure using computerized x-ray tomography, SPIE Proc Med. Img VI 152:10-18, 1978

56. Tuy HK, Tuy L: Direct 2D display of 3D objects. IEEE Comp Graph Appl 4:29-33, 1984

57. Höhne KH, Bernstein R: Shading 3D images from CT using gray-level gradients. IEEE Trans Med Imaging 5:45-47, 1986

58. Kajiya JT, Von Herzen BP: Ray tracing volume densities. Comp Graph 18(3):165-174, 1984

59. Sabella P: A rendering algorithm for visualizing 3D scalar fields. Comp Graph 22:51-58, 1988

60. Meinzer HP, Meetz K, Scheppelmann D, et al: The Heidelberg ray tracing model. IEEE Comp Graph Appl 11(6):34-43, 1991

61. Upson C, Keeler M: V-buffer: Visible volume rendering. Comp Graph 22:59-64, 1988

62. Herman GT, Udupa JK: Display of 3D discrete surfaces. SPIE Proc Med Img VI 283:90-97, 1981

63. Chen LS, Herman GT, Reynolds RA, et al: Surface rendering in the cuberille environment. IEEE Comp Graph Appl 5:33-43, 1985

64. Lenz R, Gudmundson B, Lindskog B, et al: Display of density volumes. IEEE Comp Graph Appl 6:20-29, 1986

65. Cohen D, Kaufman A, Bakalash B, et al: Real-time discrete shading. Vis Comp 6(1):16-27, 1990

66. Webber RE: Ray tracing voxel data via biquadratic local surface interpolation. Vis Comp 6:8-15, 1990

67. Gordon D, Reynolds RA: Image-space shading of three-dimensional objects. Computer, Vision, Graphics, Image Proc 29:361-376, 1985

68. Lenz R, Danielsson PE, Constrom S, et al: Presentation and perception of 3D images, in Höhne KH (ed): Pictorial Information Systems in Medicine, NATO ASI. Berlin, Germany, Springer-Verlag, 1986, pp 459-465

69. Raya SP, Udupa JK, Barrett WA: A PC-based 3D imaging system: Algorithms, software, and hardware considerations. Comp Med Imaging Graph 14:353-370, 1990

70. Udupa JK, Hung HM, Chen LS: Interactive display of 3D medical objects, Höhne KH (ed): Pictorial Information Systems in Medicine, NATO ASI. Berlin, Germany, Springer-Verlag, 1986, pp 445-457

71. Fuchs H, Kedem ZM, Uselton SP: Optimal surface reconstruction for planar contours. Commun ACM 20:693-702, 1977

72. Cook LT, Dwyer SJ III, Batnitzky S, et al: A three-dimensional display system for diagnostic imaging applications. IEEE Comp Graph Appl 3:13-19, 1983

73. Christiansen HN, Sederberg TW: Conversion of complex contour line definitions into polygonal element mosaics. Comp Graph 12:187-192, 1978

74. Shantz M: Surface definition for branching contour-defined objects. Comp Graph 15:242-259, 1981

75. Ganapathy S, Dennahy TG: A new general triangulation method for planar contours. Comp Graph 16(3):69-74, 1982

76. Shaw A, Schwartz EL: Automatic construction of polyhedral surfaces from voxel representation. Technical Report Robotics Report RR-150, Courant Institute of Mathematical Sciences, New York, NY, New York University, June 1988

77. Shinagawa Y, Kunii TL: Constructing a Reeb Graph automatically from cross sections. IEEE Comp Graph Appl 11(6):44-50, 1991

78. Mazziotta JC, Huang KH, THREAD (three-dimensional reconstruction and display) with biomedical applications in neuron ultrastructure and computerized tomography. Am Fed Inform Proc Soc 45:241-250, 1976

79. Atkinson HH, Gargantini I, Ramanath MVS: Determination of the 3D border by repeated elimination of internal surfaces. Computing 32:279-295, 1984

80. Robertson DD: Three-dimensional modelling and the design of hip and knee prostheses, in Udupa JK, Herman GT (eds): 3D Imaging in Medicine. Boca Raton, FL, CRC Press, 1991, pp 255-270

81. Rosenman J: 3D imaging in radiotherapy treatment planning, in Udupa JK, Herman GT (eds): 3D Imaging in Medicine. Boca Raton, FL, CRC Press, 1991, pp 313-330

82. Kaufman A: Efficient algorithms for 3-D scan conversion of parametric curves, surfaces, and volume. Comp Graph 21:171-179, 1987

83. Cohen D, Kaufman A: Scan conversion algorithms for linear and quadratic objects, in Kaufman A (ed): Volume Visualization. Los Alamitos, CA, IEEE Computer Society Press, 1991, pp 280-301

84. Udupa JK, Samarasekera S, Barrett WA: Boundary detection via dynamic programming. Proc Visualization in Biomedical Computing, Chapel Hill, NC, October, SPIE Proc 1808:33-39, 1992

85. Aho AV, Ullman JD: The Theory of Parsing, Translation and Compiling, Vol I: Parsing. Englewood Cliffs, NJ, Prentice Hall, 1972

86. Ledley RS, Park CM: Molded picture representation of whole body organs generated from CT scan sequences, in Proceedings of the First Annual Symposium of Computer Applications in Medical Care (IEEE), Washington, DC, 1977, pp 363-367

87. Udupa JK: DISPLAY82—A system of programs for the display of three-dimensional information in CT data. Technical Report MIPG67, Medical Image Processing Group, Philadelphia, PA, Department of Radiology, University of Pennsylvania, 1983

88. Udupa JK, Herman GT, Margasahayam PS, et al: 3D98: A turnkey system for the display and analysis of 3D medical objects. SPIE Proc 671:154-168, 1986

89. Dev P, Wood SL, Duncan JP, et al: An interactive graphics system for planning reconstructive surgery. Proceedings of the Fourth Annual Conference and Exposition of the National Computer Graphics Association, Philadelphia, PA, June 1983, pp 130-135

90. Heffernan PB, Robb RA: A new method for shaded surface display of biological and medical images. IEEE Trans Med Imaging 4:26-38, 1985

91. Vannier MW, Marsh JL, Warren JO: Three-dimensional computer graphics for craniofacial surgical planning and evaluation. Comp Graph 17:263-274, 1983

92. Tiede U, Höhne KH, Bomans M, et al: Investigation of medical 3D rendering algorithms. IEEE Comp Graph Appl 10:41-53, 1990

93. Levoy M: Efficient ray tracing of volume data. ACM Trans Graph 9:245-261, 1990

94. Raya SP: SOFTVU—A software package for multidimensional medical image analysis. Technical Report MIPG148, Medical Image Processing Group, Philadelphia, PA, Department of Radiology, University of Pennsylvania, June 1989

95. Gillespie JE: Three-dimensional computed tomographic reformations: Assessment of clinical efficacy, in Udupa JK, Herman GT (eds): 3D Imaging in Medicine. Boca Raton, FL, CRC Press, 1991

96. Rusinek H, Karp NS, Cutting CB: A comparison of two approaches to three-dimensional imaging of craniofacial anomalies. J Digit Imaging 3:81-88, 1990

97. Udupa JK, Herman GT: Volume rendering versus surface rendering. Commun ACM 32:1364-1366, 1989

98. Lang P, Stieger P, Lindquist T: Three-dimensional reconstruction of magnetic resonance images: comparison of surface and volume rendering techniques, in Arenson RL, Friedenberg RM (eds): Proceedings of SCAR90, Computer Applications to Assist Radiology, Symposia Foundation, Carlsbad, CA, 1990, pp 520-526

99. Wood SL: Visualization and modeling of 3-D structures. IEEE Engin Med Biol Mag 11(2):72-79, 1992

100. Swets JA, Pickett RM: Evaluation of Diagnostic Systems, New York, NY, Academic Press, NY, 1982

101. Herman GT: Computer simulation of data collection in CT, in: Image Reconstructions from Projections: The Fundamentals of Computerized Tomography. New York, NY, Academic Press, 1980, pp 55-64

102. Phong BT: Illumination for computer generated images. Commun ACM 18:311-317, 1975

103. Udupa JK, Hung HM, Odhner D, et al: Multidimensional data format specification: A generalization of the American College of Radiology—National electric manufacturers association standards. J Digit Imaging 5(1):26-46, 1992

104. Udupa JK, Hung HM, Odhner D, et al: 3DVIEWNIX: A data-, machine-, and application-independent software system for multidimensional data visualization and analysis. SPIE Proc 1653:185-191, 1992