Surface and Volume Rendering in Three-Dimensional Imaging: A Comparison

Jayaram K. Udupa, Hsiu-Mei Hung, and Keh-Shih Chuang

Many surface rendering techniques are currently available for the three-dimensional display of structure data captured by imaging devices. Comparatively fewer volume rendering techniques are also available for the same purpose. The relative performance of these two methodologies in visualization tasks has been a subject of much discussion recently. Although it is very desirable to establish, based on observer studies, objective guidelines stating the relative merits of the two methodologies even for specific situations, it is impossible to conduct meaningful observer studies that take into account the numerousness of the techniques in each methodology, and within each technique, the numerousness of the parameters and their values that control the outcome of the technique. Our aim in this article is to compare the two methodologies purely on a technical basis in an attempt to understand their common weaknesses and disparate strengths. The purpose of this article is twofold—to report a new surface rendering technique and to compare it with two volume rendering techniques reported recently in the literature. The bases of comparison are: ability to portray thin bones; clarity of portrayal of sutures, fractures, fine textures, and gyrations; smoothness of natural ridges and silhouettes; and computational time and storage requirements. We analyze the underlying algorithms to study how they behave under each of these comparative criteria. Our conclusion is that, at the current state of development, the surface method has a slight edge over the volume methods for portrayal of information of the type described above and a significant advantage considering time and storage requirements, for implementations in identical environments.

CURRENTLY THERE are two classes of approaches available for visualizing and analyzing medical three-dimensional images, commonly known as surface rendering[1-3] and volume rendering.[4-6] In surface rendering, the premise is that the given image contains data pertaining to certain tangible structures in the human body, and hence, that each structure can be visualized by its surface estimated from the image. In volume rendering, there is no assumption of an underlying structure or its surface, rather the tissue material of each structure to be visualized is estimated in every voxel and the structures are visualized through calculations based on the opacities and colors assigned to different tissues and on some assumed optical behavior of tissue mixtures. In both methodologies, the important issue is how to do rendering so that medically relevant information is portrayed in the rendered image. In surface rendering, this is achieved through a preprocessing step that consists of determining the surface. In volume rendering, relevant information is identified through a process known as classification wherein opacities and colors are assigned to voxels based on the original voxel values or on information derived from them. In principle, for any preprocessing step used in surface rendering, it is possible to devise a preprocessor for volume rendering such that the rendered images generated by the two methods are practically identical (for example, the preprocessor may be a surface detector for both). We use binary volume rendering to refer to such a class of volume techniques and refer to the more general remainder by grey volume rendering or simply volume rendering.

The comparative assessment of surface and volume methods is a very complex subject. It is not possible to derive scientific facts on this subject independent of the organ system under study, the imaging modality and its parameters, the actual application, the algorithms used in each methodology and their parameters, and the observer. The scope of our study is, by necessity, limited, but hopefully it will initiate further studies of this nature on this important subject.

Though observer studies will be the final arbiters of this issue (such studies have already begun [see reference 7]), it is impossible to conduct them taking into account all of the above factors because of the magnitude of the

From the Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA, and the Department of Radiology, University of California at Los Angeles, Los Angeles, CA.

Supported by National Institute of Health Grant CA50851.

Address reprint requests to Jayaram K. Udupa, Medical Image Processing Group, Department of Radiology, University of Pennsylvania, Philadelphia, PA 19104-6021.

time and resources needed. It is therefore necessary to identify and eliminate parameters that may have insignificant influence on the outcome of such studies. The purpose of this article, therefore, is to compare the two methodologies purely on technical grounds in an attempt to identify common weaknesses and disparate strengths.

Clearly, it is enough to confine ourselves to the comparative study of grey volume and surface rendering since binary volume techniques are identical in effect to that of surface rendering. We concentrate on the skeletal structures of the human head derived from computed tomography (CT) data and on the external aspects of the human brain derived from magnetic resonance imaging (MRI) data. In the skeletal structure, we further confine to fine features such as sutures, fractures, thin bones, and ridges. For brain, the features considered in the study are the gyri and the sulci.

For volume rendering, we use the techniques reported in references 5 and 6 and for surface rendering, the methodology is our own. Since the main objective of this paper is comparing the methods, we will not delve into their detailed description. The surface method is described in part in our technical reports but there are some new aspects to it which we have not reported so far. We describe here only its salient aspects.

METHODS

We assume that the three-dimensional Euclidean space R^3 is divided by three sets of mutually orthogonal planes, the planes in each set being parallel to the principal planes of a coordinate system denoted $(x_1x_2x_3)$. We call the closed set of points representing every cuboid formed by such a division of R^3 a *voxel.* Note that, in addition to the interior points, the voxel also contains points defining the faces, edges, and vertices of the parallelepiped. We denote the set of all voxels by V_R, the set of the centers of all voxels by G_R, and call the discretized system consisting of V_R and G_R the g_r grid system. In a similar way, we define another grid system denoted g_I with its associated sets G_I and V_I by partitioning R^3 by another set of three sets of mutually orthogonal planes. We often refer to the points of G_L, for $L\epsilon\{R, I\}$, as grid points in the g_L system and use a grid point and the voxel whose center it represents interchangeably. Similarly, we use a subset of G_L and the corresponding subset of V_L interchangeably, since there is a 1:1 correspondence between G_L and V_L. In the following description we use L as a variable that stands for any of the elements of the set $\{R, I\}$.

We define a *scene* (Σ_L) in the g_L system to be a tuple (E_L, γ_L), where

$$\mathrm{E}_L = \{\mathrm{X} = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3) \mid \text{for } 1 \le \mathrm{i} \le 3, B^l_{iL} \le x_i \le B^h_{iL}, B^l_{iL} \text{ and } B^h_{iL} \text{ are finite positive integers}\} \quad \text{(Equation [1])}$$

is called the domain of the scene, and γ_L is a mapping from E_L to the set of numbers $D = \{D_l, D_{l+1}, \ldots, D_h\}$. If $D_l = 0$ and $D_h = 1$ are the only two numbers in D we call Σ_L a *binary scene.* We refer to the number $\gamma_L(X)$ associated with the voxel X as the *density* of X.

Surface Rendering

An object of interest in the scene is usually specified by a segmentation procedure that converts the given scene into a binary scene. It can be specified by a predicate P and a vector function f, such that, if a voxel X belongs to the object, then $P(f(X))$ is true. If $\Sigma_L = (E_L, \gamma_L)$ is a scene and $\Sigma'_L = (E_L, \gamma'_L)$ is a binary scene resulting from a segmentation specified by P and f, then for any $X \in E_L$,

$$\gamma'_L(X) = \begin{cases} 1, \text{ if } P(f(X)) \text{ is true,} \\ 0, \text{ otherwise} \end{cases} \quad \text{(Equation [2]).}$$

We define the structure U_L identified by the segmentation procedure P to be the set

$$U_L = \{X \mid X \in E_L \& \gamma'_L(X) = 1\} \quad \text{(Equation [3]).}$$

Segmentation is rarely ever perfect, as a result U_L often contains voxels that do not really belong to the object we want to visualize. One way of getting rid of unwanted clutters is to extract a "connected" boundary surface of U_L and then to render the surface. Several mathematical definitions of the notion of a connected surface have been reported.[8-10] We choose the one given in reference 10 since that leads to a more efficient algorithm than others. We will omit the rather complex details but describe only those aspects that are relevant to understanding our modifications. An *object* W_L of U_L (also of Σ_L and of Σ'_L) is a maximal subset of U_L of voxels that are connected by their faces. Maximality implies that no voxel in U_L that is not already in W_L is connected to a voxel in W_L. A *co-object* O_L of U_L (also of Σ_L and Σ'_L) is a maximal subset of $Z_L = E_L - U_L$ of voxels that are connected by any of the edges parallel to the x_1 and x_2 directions. A *boundary face* F is the intersection of two voxels X_1 and X_2 such that $X_1 \in U_L$, $X_2 \in Z_L$ and X_1 and X_2 share a face. We denote F by the ordered pair (X_1, X_2). A *boundary surface* S_L of U_L is the set of boundary faces

$$S_L = \{F = (X_1, X_2) \mid X_1 \in W_L \,\&\, X_2 \in O_L\} \quad \text{(Equation [4]).}$$

The boundary surface is, thus, defined between a connected component of U_L and a connected component of its background. If the two connected components are not adjacent—meaning that no voxel of W_L shares a face with some voxel in O_L—then S_L is empty. Clearly, if U_L is nonempty, it has at least one nonempty boundary surface. We are interested in visualizing nonempty boundary surfaces such as S_L.

The theory described in reference 10 allows us to assign to boundary faces directions in which neighbors are defined (Fig 1A). In this assignment, there are faces with only one direction, called type-1 faces, and faces with two directions called type-2 faces. Consider a boundary face $F = (X, X')$ as in Fig 1B. For each of its directions, we define a unique

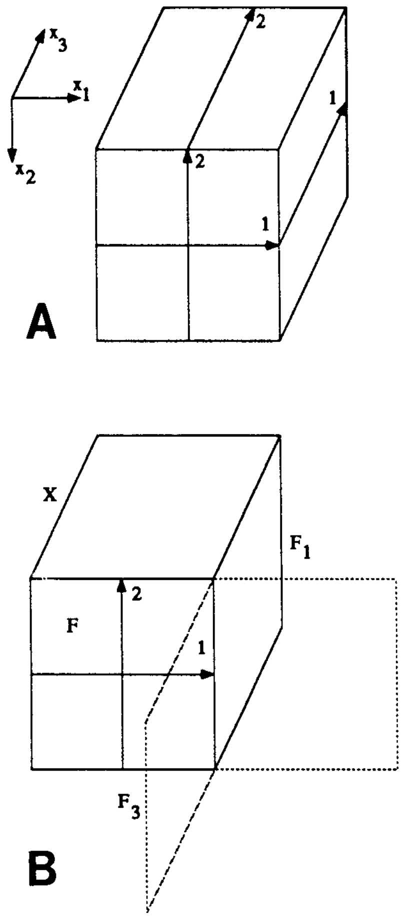

Fig 1. (A) Assignment of directions to boundary faces to define neighbors. (B) In direction 1, the neighbor of *F* is one of F_1, F_2, and F_3. Note that being a neighbor is not a symmetric relation.

neighbor. For example, for direction labelled 1, the neighbor of F is one of $F_1 = (X, X_1)$, $F_2 = (X_1, X_2)$, and $F_3 = (X_2, X')$. Depending on the membership of X_1 and X_2 in U_L (note that $X' \in Z_L$ and $X \in U_L$), the theory allows us to pick exactly one of F_1, F_2 and F_3 as the neighbor of F (see reference for details).

So far, our description has been entirely independent of the grid system. In practice, the set of slice images captured by a scanner represents a scene in one grid system, say, g_R. Usually the spacing between grid points in G_R in the x_1 and x_2 directions is identical but smaller than the spacing in the x_3 direction. The g_I system represents the grid system in which we would like the surface to be visualized to be computed. We allow the grid spacing in the g_I system to be smaller or greater than the spacing in any direction in the g_R system. This implies that the surfaces can be computed on a grid coarser or finer than that in which the given scene is defined. We track surfaces S_I defined in g_I directly in the given scene Σ_R. We assume that a segmentation procedure P is available that allows us to decide during surface tracking if a voxel of E_I belongs to U_I or Z_I.

Input. A scene $\Sigma_R = (E_R, \gamma_R)$, a segmentation procedure P and a vector function f, a boundary face F_0 of the binary scene Σ_I' that would be generated if P were applied to Σ_R, the grid system g_I.

Output. A list of boundary faces in the boundary surface of an object in Σ_I' that contains F_0.

Data structure. A queue Q that contains boundary faces of Σ_I' that have already been found but whose neighbors are still to be discovered, a list M of type-2 boundary faces of Σ_I' that have been visited exactly once.

Functions called: N_1, N_2, P, and f.

```
Algorithm SD
begin
(1) queue $F_0$ and put two copies of it in $M$;
(2) while $Q$ is not empty do
    (a) remove a face $F = (X, X_0)$ from $Q$ and find its type $t$;
    (b) output $F$;
    (c) for $i \leftarrow 1$ to $t$ do {loop on neighbors of $F$};
    (d) $X_1 \leftarrow N_1(X, F, i)$;
    (e) if $(P(f(X_1)) = FALSE)$ then
            $F_i \leftarrow (X, X_1)$;
        else
            $X_2 \leftarrow N_2(X, F, i)$;
            if $(P(f(X_2)) = TRUE)$ then
                $F_i \leftarrow (X_2, X_0)$;
            else
                $F_i \leftarrow (X_1, X_2)$;
    (f) if $F_i$ is of type 1 then queue $F_i$;
        else
            (f1) if $F_i \in M$ then delete $F_i$ from M;
            (f2) else queue $F_i$ and put $F_i$ in $M$;
      end for;
  end while;
end
```

In this algorithm, N_1 and N_2 are functions that, given a voxel X, a face F, and a direction i return voxels X_1 and X_2 respectively, where F, X, X_1 and X_2, and the directions are as defined in Fig 1. Clearly N_1 and N_2 can be implemented as table lookup operations.

We have assumed P and f to be given. The vector function f allows us to evaluate a set of features at any given point $X \in G_I$ that best characterize the object of interest and P is essentially a classifier that classifies the point based on the feature vector. Commonly, we use density thresholding in which case $f(=\gamma_I)$ is scalar and P is a comparator. In this simple case, we may implement f as any suitable interpolating function. If we assume a trilinear interpolant, then given a point $X \in G_I$, the implementation of f should first identify those eight points of G_R that form the vertices of the smallest parallelepiped that contains X. $f(X)$ is then computed by trilinearly interpolating the known densities associated with the eight points. If we consider a higher-degree interpolant, such as a cubic-spline function, a larger subset of points of G_R has to be determined to evaluate the function. One word of caution is in order regarding the type of classifier P and the function f used. For the algorithm to work correctly, it is necessary that the faces discovered in the algorithm truly form a boundary surface as we have defined earlier. If P and/or f are such that a given point is classified differently at different times it is encountered in Step 2e during the execution of the algorithm (nonstationarity), then generally correct performance of the algorithm cannot be guaranteed. There are a variety of forms of P and f that satisfy stationarity and that provide a segmentation more sophisticated than thresholding. When P and f do not satisfy the stationarity requirements, it is possible to modify the above algorithm to incorporate a "surface correction" step so that it eventually terminates and generates a surface having the properties defined in Equation (4). A detailed discussion of these topics is beyond the scope of this article.

Finally, note that, in computing a boundary surface, Algorithm *SD* does not compute any of Σ_I, $\Sigma_I'U_I$, and Z_I. Besides, based on the nature of the function f, it estimates feature values only at points in the vicinity of the boundary surface. The beauty of the algorithm is that it works on both scenes and binary scenes irrespective of whether the input and output grid systems are different or identical. It is less obvious in the case of binary scenes how the algorithm works when the g_R and g_I systems are not identical. If we choose f to be a scalar function that estimates the shortest distance of X from the boundary with a convention that distances to points outside the boundary are negative and to those inside are positive, then to classify X, P has to check for the sign of $f(X)$. We have recently shown[11] that this form of interpolating binary scenes based on distance to boundary is generally more accurate than interpolating densities.

The output of Algorithm *SD* is a list of square faces, each face F described by the location of its center C and a normal vector N assigned to C that as closely as possible approximates the normal at a point on the underlying continuous surface that is closest to C. In addition to the effectiveness of P and f in detecting the surface, the location of C and the normal N of each face determine the quality of the depiction created by the rendering method.

Though, in principle, $F = (X_1, X_2)$ is considered to be the intersection of X_1 and X_2, we allow it (its center C) to have any location along the line segment connecting the centers of X_1 and X_2. The actual location is computed by determining where the decision boundary of P lies along this line segment. When P is a comparator with threshold T, and f a linear interpolant of density, C is at a distance d from X_1 given by

$$d = \frac{T - \gamma_I(X_2)}{\gamma_I(X_1) - \gamma_I(X_2)} \qquad \text{(Equation [5])}.$$

Thus, whereas two coordinates of C are integers, the third (along the axis to which F is perpendicular) is a real number.

The normal N associated with F is determined considering six points situated equidistant from C along the principal directions: one each at the center of the four edges of F, and the remaining two along the line segment connecting X_1 and X_2 (see Fig 2). The components of N are computed from pairwise differences in density of these points. The density at each of these six points X is determined based on the densities of 16 voxels in Σ_R, eight with X_3 coordinates greater than that of X and eight with less than that of X (see reference 12). Thus, potentially, 96 voxels in the vicinity of C contribute to the computation of N.

Given a surface of the form described so far, we use standard techniques for visibility[1] and shading calculations.[13] The shading associated with a face F is given by

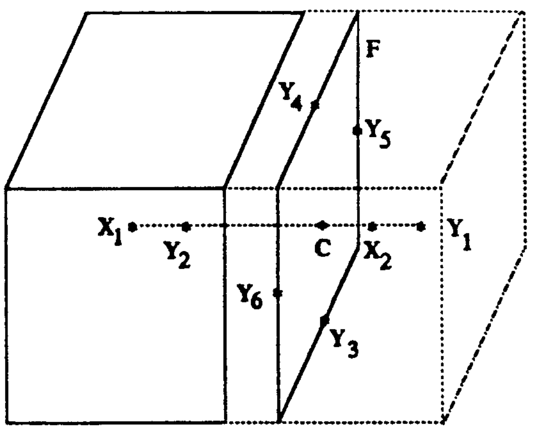

Fig 2. A face F (its center C) may have any real location within the voxel which indicates the location of the decision boundary. The normal N is computed from the density at each of $Y_1, \ldots, Y_6$ each of which in turn is computed based on the densities of 16 voxels in Σ_R.

$$s(F) = k[k_d f_d(N, L) + k_s f_s(N, L)]e(d_F) + A, \qquad \text{(Equation [6])}$$

where k, k_d, k_s, and A are constants f_d and f_s are functions that determine the diffuse and specular components of reflection from F, L indicates the direction of light source and e is a function of the distance of C from the viewpoint.

Volume Rendering

Since we are not introducing any new ideas in this section (in fact, our aim is to implement the chosen techniques[5,6] as close as possible to how they are originally reported), we give only a brief description of one of the methods[6] (the other[5] is similar to but more sophisticated than the former). There are two aspects to this rendering technique: preprocessing or classification, and rendering.

Preprocessing. The purpose of preprocessing a given scene Σ_R is to assign to every voxel in E_R an opacity value and a brightness value. Opacity determines the light transmission property and brightness determines the reflective property of the voxel. The main idea here is that if we somehow assign high opacity values to voxels in the vicinity of an alleged boundary and low values to those away from the boundary, then essentially voxels of the former type determine light intensity reaching the viewpoint. Described in our terminology, the process of assigning opacity essentially creates an opacity scene $\Sigma_R^o = (E_R, \gamma_R^o)$, where γ_R^o is a function that assigns opacity to voxels (the range of γ_R^o is assumed to be the closed interval [0, 1]). In reference 6, γ_R^o is considered to be a piecewise linear function of density and the magnitude of the gradient of γ_R evaluated using a digital gradient operator. (In our implementation γ_R^o is considered to be a product of two ramp functions—one a function of density and the other a function of the magnitude of the density gradient). The process of assigning brightness creates a brightness scene $\Sigma_R^b = (E_R, \gamma_R^b)$, where γ_R^b is a function that assigns brightness to each voxel. The form of γ_R^b we use is identical to the Phong shading function of Equation [6], with the only difference that N now represents a normal vector assigned to the center of the voxel under consideration, which is computed using a gradient of γ_R estimated at this point taking central differences.

Rendering. The purpose of rendering is to create a depiction of the alleged boundary by calculating the intensity of the light reaching the viewpoint from a combined process of reflection from and transmission through voxels in E_R. Using the ray-casting paradigm, to determine the brightness to be assigned to a pixel in view space for a given viewing transformation, we work from yon to hither along the ray assigned to the pixel. A sequence $X_1, \ldots, X_n$ of identical and abutting voxels is determined along the ray between the point of entry of the ray into the domain of the scene and its point of departure. The brightness of light gets modified from B_{in} to B_{out} as it passes through each voxel X_i traveling toward the viewpoint, in the following fashion:

$$B_{out} = \gamma_I^o(X_i)\gamma_I^b(X_i) + (1 - \gamma_I^o(X_i))B_{in}, \quad \text{(Equation [7])}$$

where $\gamma_I^o(X_i)$ and $\gamma_I^b(X_i)$ are the opacity and brightness values associated with X_i, which are calculated via trilinear interpolation of Σ_R^o and Σ_R^b, respectively. Starting with the voxel X_n farthest from the viewpoint and working toward X_1, B_{out} is computed using Equation (7) assuming B_{in} to be the value of B_{out} computed for X_{i+1} (B_{in} for X_n is assumed to be equal to the ambient light intensity). The value assigned to the pixel under consideration is the value of B_{out} computed for X_1.

COMPARISON

We use two data sets for comparing the renditions created by the two methods. The first is a $512 \times 512 \times 62$ scene (in the g_R system) pertaining to a trauma victim obtained via CT with a uniform grid spacing of 0.4 mm in the x_1 and x_2 directions and 1.5 mm in the x_3 direction. The structure of interest here is the patient's skull. The second data set is a $256 \times 256 \times 32$ scene of a human brain specimen obtained via MRI. The grid spacing for this data is a uniform 0.78 mm in the x_1 and x_2 directions and 3 mm in the x_3 direction. In creating renditions, we have used those values of parameters that we determined to be optimal for the surface method, and the values specified by Levoy[6] for the volume method. All images are created without any antialiasing (in the image space). We have used three different "resolutions" for the rendered images. For medium resolution, the grid spacing a_I in the g_I system is assumed to be identical in all dimensions and equal to the grid spacing in the x_1 direction (pixel size) a_R of the g_R system. (For volume rendering, this implies that the size of the cubic voxels sampled along any ray is equal to the pixel size.) For high resolution, we assume that $a_I = \frac{1}{2} a_R$ and for low resolution $a_I = 2a_R$. For all resolutions and for both methods, the size of the two-dimensional array defining the image is determined such that the projection of a single boundary face for any orientation of the object of interest falls just inside a single pixel of the image.

In the surface rendering method, we have used thresholding (sometimes on multiple features) to determine the location of boundary elements. In our implementation of volume rendering, only the opacity scene is computed during preprocessing. The brightness value to be assigned to each voxel along each ray is computed during rendering by first computing N at the center of the voxel using tri-linear interpolation and taking central differences and then evaluating Equation [6]. This, in our opinion, produces smoother depictions, though computationally more expensive, than the method of first determining the brightness scene during preprocessing and then estimating the voxel brightness value through trilinear interpolation. In an attempt to determining what should be the optimal values for the parameters of γ_R^o (before obtaining the values from Levoy[6]), we have created literally hundreds of images corresponding to some chosen view of the structure by varying the values of the parameters continuously over an interval whose end points in our view represented extreme situations. We do not claim that the values we have used for these parameters are the best possible (we have observed considerable variation of what we considered as optimal values from one data set to another in the same modality), yet this arduousness signals a difficulty that indeed is a counterpart of the difficulty of segmentation in surface rendering. How to determine γ_R^o with least effort and most effectiveness is an interesting research topic.

Figures 3 through 6 show some renditions derived from the two scene data sets described earlier. Figures 3 and 4 show identical views generated from the first data set by the surface and volume methods at medium (Fig 3) and low (Fig 4) resolution. Figure 5 shows identical views generated from the second data set by the two methods at high resolution. We have also implemented another volume rendering method.[5] Figure 7 is an image created by this

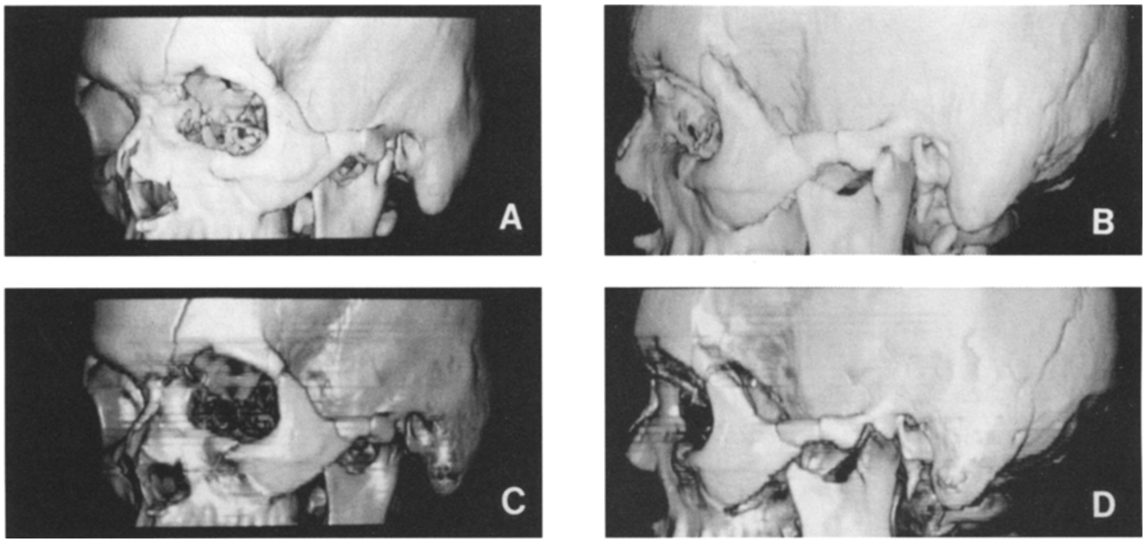

Fig 3. Renditions derived from the first scene at medium resolution: (We decided not to create renditions at high resolution for this data set because of the considerable difficulties coming from time and storage requirements of volume rendering). (A and B) surface rendering, (C and D) volume rendering.

method for the first data set. Our following comparative statements are based on these and many other images we have generated and on our experience with the specific implementations of these methods. We should emphasize that there was no interactive editing of any sort involved in the creation of any of these images.

Thin Bones

Depiction of thin bone in the vicinity of the orbits is equally poor in the two methods. There are two main reasons for this difficulty. First, the characteristics of the scene in the vicinity of thin bones are very similar to those of the skin. Therefore, except when dry skull specimens are used (portrayal is now equally good in the two methods), it is very difficult to segment (using P and f) or to classify (using γ_R^o) voxels constituting thin bone differently from those constituting skin using global criteria such as those used in this paper. More sophisticated neighborhood operators have been studied[14] for segmenting

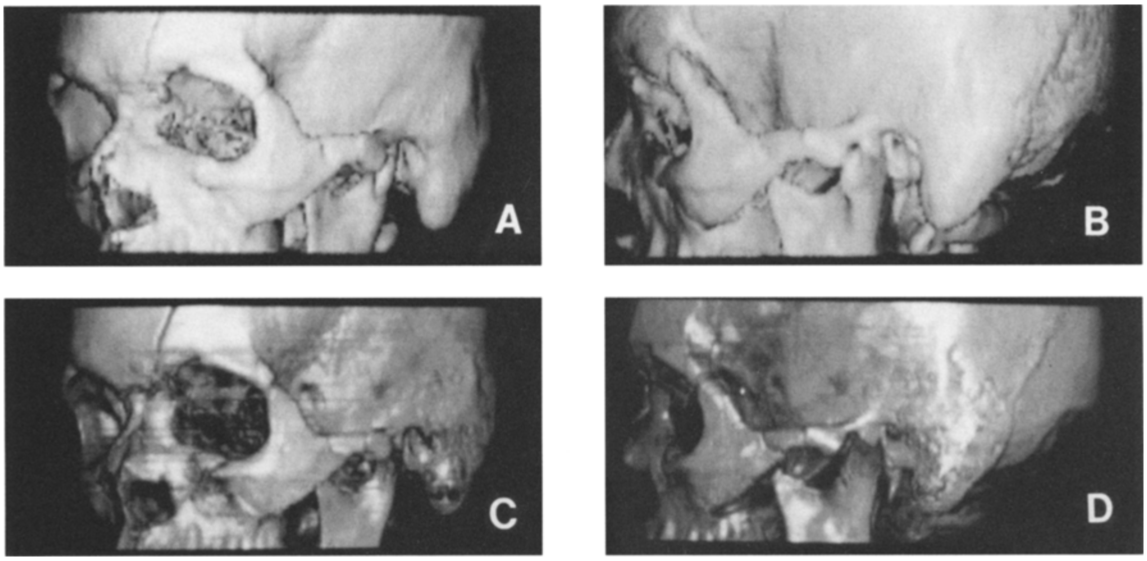

Fig 4. (A-D) Renditions as in Fig 3 but computed at low resolution.

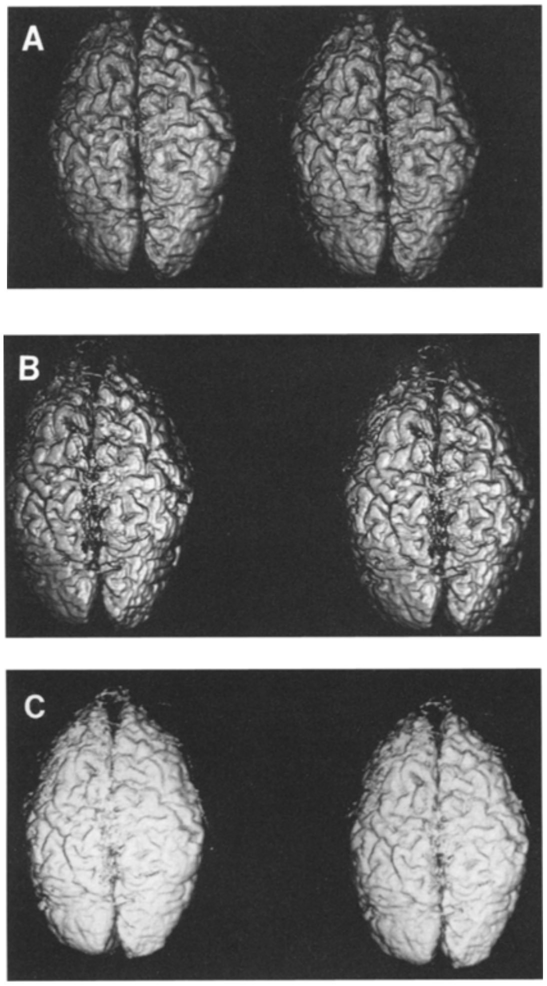

Fig 5. Renditions derived from the second scene at high resolution: (A) surface rendering, (B) volume rendering at high specularity, (C) volume rendering at low specularity (still somewhat higher than that in [A], all in stereo.

such voxels with some success in scenes of lower resolution than that used here. In our implementation of this technique, the improvement achieved was negligible, perhaps because the

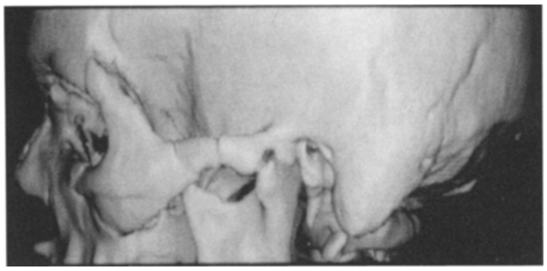

Fig 6. Surface rendition as in Fig 4A except for 60 HU higher threshold.

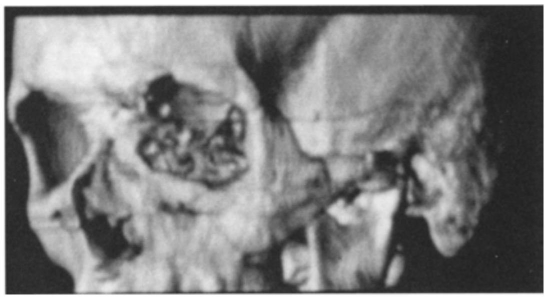

Fig 7. Volume rendition created by the method of reference 5 for the first data set. (Courtesy of Elliot Fishman and Derek Ney, The Johns Hopkins Medical Institutions).

neighborhood characteristics responsible for the earlier success are no longer valid in the higher resolution scene. The second reason is that even when thin bones can be segmented/classified effectively (say using some specialized knowledge), their effective rendering is difficult because the normal N computed from an estimate of the gradient of γ_R in the vicinity of thin bones is not reliable. In fact, when we modified algorithm SD to incorporate some specialized knowledge such as specifying the approximate region in which thin bones may lie and to use features in addition to density, we were constantly misled by the rendering because of unreliable N. Only when we resorted to distance-only shading, we saw the improvement in the detection of thin bones resulting from the modified SD. These problems are common to both methodologies and constitute important research topics.

Sutures, Fractures, Fine Textures

There is no clear advantage in either of the methods for the depiction of these features. The surface method seems to produce sharper depiction of the features than the volume method.

There is a certain amount of tolerance and robustness in the surface method described here, which are responsible for the effectiveness of the depiction of fine features irrespective of the digital nature of the surface. For example, for a suture of a skull to be portrayed effectively, it is not necessary that the feature be modelled precisely in the digital geometry of the surface. All that is required is that the feature be in the close vicinity of the modelled surface. Appropriate interpolation and normal estimation procedures guided by the geometry of the surface can bring out the feature in the rendition. In fact, the rendition using even a thresholding method is surprisingly insensitive to the actual threshold used. Figure 6 shows a surface detected at a threshold value 60 Hounsfield Units above that used for the surface shown in Fig 3B. In our observation based on implementing other voxel-based surface rendering techniques[9,15] where boundary elements are voxels rather than voxel faces, the latter (of the type described in this article) seem to capture fine details better than the former. It will be interesting to see if a similar parallel exists for the volume rendering method.

One other method-independent aspect that influences the portrayal of fine features is the relative specularity of the surface (determined by k_d, f_d, k_s, and f_s in Equation [6]). In both methods, higher specularity tends to emphasize discontinuities, which has the undesirable property of emphasizing unwanted discontinuities along with those that we want enhanced. This enhances some artifacts such as the "slicing artifact"; compare Fig 3A and C where specularity used for the former is smaller than the recommended value we have used for the latter. The boldness with which fine features appear at a given specularity is about the same in both methods. The slicing artifact depends on other factors as described later on.

Gyrations, Ridges, Silhouettes

The silhouettes and natural edges appear to be portrayed smoother by the surface method than by the volume technique in higher resolutions. Though the underlying surface geometry is digital and hence jagged, because of the method of estimating N and of determining the real (rather than digital) location of boundary faces, the display of ridges, edges, and gyrations in the surface method does not show jaggedness even without the antialiasing operation. The volume method has an advantage in that the apparent smoothness of natural edges can be controlled by appropriately choosing the slopes of the ramps defining γ_R^o. As the slope decreases the smoothness of the edges increases and the "slicing artifact" is minimized. However, at the same time sharpness of some of the desirable features is also lost. Because of such complex interplay of many parameters, even if we assume identical optical models and values of parameters, the renditions may not display identical optical characteristics. For example, to achieve a given degree of specularity, the values of the parameters in Equation [6] controlling specular reflection should be chosen higher for volume rendering than for surface rendering, perhaps due to the integrating effect of the former. Thus, it becomes difficult controlling even those parameters that are common to the two methods. At lower resolutions, surprisingly the volume method seems to produce smoother ridges and silhouettes than the surface method and than the volume method at higher resolutions for a fixed γ_R^o.

The gyrations and sulci of the brain are portrayed equally well by the two methods provided they can be segmented/classified effectively, which is the case with the data set used in this comparison. In fact, we intentionally selected such a data set to separate the segmentation/classification issues from the rendering issues. If segmentation/classification were not effective, the methods in their automatic mode as reported in this article are both equally poor in portrayal, again because of the unreliability of N and inclusion of nonobject points in rendition. Improvements can be made using more sophisticated segmentation/classification functions and combining in an adaptive fashion the object-space and scene-space normal estimation methods. The discussion of such topics to improve the display of frail objects forms the subject of a separate paper. Others have observed[16] that integration of image intensity values in the vicinity of presegmented boundaries in the direction of viewing can overcome the problem of unreliable N for rendering the brain surface. Of course, this method totally ignores N, but is an example of combined surface and volume strategies for rendering.

Time and Storage

It is not possible to give a precise comparison of speed of rendering for the two methods based on our implementation of the volume method since it is much less optimized compared to the surface method. Three- to 10-fold speed up over a straightforward implementation has been reported for the volume method[17] using a variety of techniques whose objective is to minimize the number of rays cast and the number of voxels sampled along each ray without sacrificing image quality. Based on the reported timings, our estimate is that, for the current state of optimization for the two methods and implementation in an identical environment, the surface technique is 25 to 30 times faster in the rendering step than the volume method. The number of surface elements (1.2 million for the example of Fig 3A) is usually much smaller than the total number of voxels that enter into computation in volume rendering. Further, the actual number of computations done per element during rendering are fewer and simpler in the former method than in the latter. In our implementation, the preprocessing time required by the former method is roughly ⅓ that required by the latter.

The storage requirement during the rendering process is determined mainly by how the algorithms are implemented. In general, as the implementation is made more optimal for speed, the memory requirement increases and vice versa. The volume method requires enough space to store two scenes (Σ_R^o and Σ_R or Σ_R^o and Σ_R^b) in addition to the space required to store the image being generated. The surface method requires random access only to an image and a z-buffer and not to the surface data, and therefore the memory required is determined by the size of these buffers. With these considerations, the memory required during rendering by the surface method for the example in Fig 3 is about 20 times less than that required by the volume method, though this factor for disk space requirement is about 3 to 4.

Finally, in our own experience, the volume method is easier for a straightforward first implementation than the surface method reported here.

CONCLUSIONS

In conclusion, from purely a technical viewpoint, for the type of display considered in this article, we feel that the surface rendering method of the type described here should be preferable to the volume method considered here on the basis of the nature of the information portrayed in the renditions and time and storage requirements. As we tried to improve our implementation of the two methods it became clear to us that the differences between them start disappearing (for example, we recently devised a new data structure for the volume method that has now brought its rendering speed close to that of the surface method). Therefore it is just as relevant to compare between these methods as it is to compare techniques in the same methodology. In our view, the sources of the shortcomings in the two methodologies are common. There is a great deal to be benefited by each methodology from developments in the other since a new development in one has often an appropriate parallel connotation in the other. For example, it is possible to combine some of the ideas related to normal estimation and interpolation described under surface rendering with the volume rendering method. We could have done even a connectivity analysis to discard the clutter appearing in the volume rendition examples. Then the differences between the two methodologies start disappearing and only their common weaknesses remain. The bottom line, in our view, that decides which methodology is superior in a particular visualization task is which is more effective segmentation or classification in successfully identifying tissue regions and interfaces. If situations exist where there is such a difference, the design of appropriate techniques that yield the difference is an interesting research topic.

An issue we have not addressed in this article that is usually considered while comparing surface and volume methods[18] is the structure oriented mensuration and manipulation found useful in interactive surgical planning. These are not possible with the volume methods at present; but with an appropriate data structure (of the type mentioned above) to represent the fuzzy structure it is possible to incorporate such capabilities into volume rendering methods. The notion of a boundary is therefore essential even to volume methods. We do not know what it means to visualize, manipulate, and analyze a totally diffuse entity (which has no boundary by any definition), either on its own or in combination with other diffuse or tangible entities.

ACKNOWLEDGMENT

The authors are grateful to Mary A. Blue for preparing the manuscript, to Steve Strommer for photography, to Marc Levoy of Stanford University, and to Derek Ney of The Johns Hopkins Medical Institutions for help with some implementational details of volume techniques.

REFERENCES

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

2. Cook LT, Dwyer III SJ, Batnitzky S et al: A three-dimensional display system for diagnostic imaging applications. Institute of Electrical and Electronic Engineers Comput Graph and Appl 3:13-19, 1983

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

4. Frieder G, Gordon D, Reynolds RA: Back-to-front display of voxel-based objects. Institute of Electrical and Electronic Engineers Comput Graph Appl 5:52-60, 1985

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

6. Levoy M: Display of surfaces from volume data. Institute of Electrical and Electronic Engineers Comput Graph Appl 8(3):29-37, 1988

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

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

9. Udupa JK, Srihari SN, Herman GT: Boundary detection in multidimensions. Institute of Electrical and Electronic Engineers Trans Patt Anal Mach Intel PAMI-4:41-50, 1982

10. Gordon D, Udupa JK: Fast surface tracking in three-dimensional binary images. Comput V Graph and Imag Proc 45:196-214, 1989

11. Raya SP, Udupa JK: Shape-based interpolation of multidimensional objects. Institute of Electrical and Electronic Engineers Trans Med Imag MI-9:32-42, 1990

12. 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, 1988

13. Phong BT: Illumination for computer generated images. Comm Associate for Computer Machinery 18:311-317, 1975

14. Herman GT, Roberts D, Rabe B: The reduction of pseudo-foramina (false holes) in computer graphic presentations for craniofacial surgery. Proceedings of the 8th Ann Conf and Expo of the National Computer Graphics Association 3:81-85, 1987

15. Udupa JK, Odhner D: Interactive surgical planning: High-speed object rendition and manipulation without specialized hardware. Proceedings, First International Conference on Visualization in Biomedical Computing, Atlanta, GA. Institute of Electrical and Electronic Engineers Computer Society, Los Alamitos, CA, 1990, pp 330-336

16. Bomans M, Höhne K-H, Tiede U, et al: 3-D segmentation of MR images of the head for 3-D display. Institute of Electrical and Electronic Engineers Trans Medical Imaging MI-9:177-183, 1990

17. Levoy M: Efficient ray tracing of volume data. Associate for Computer Machinery Transactions on Graphics 9:245-261, 1990

18. Becker SC, Barrett WA: Interactive morphometrics from three-dimensional surface images. Proceedings, First International Conference on Visualization in Biomedical Computing, Atlanta, GA. Institute of Electrical and Electronic Engineers Computer Society, Los Alamitos, CA 1990, pp 418-425