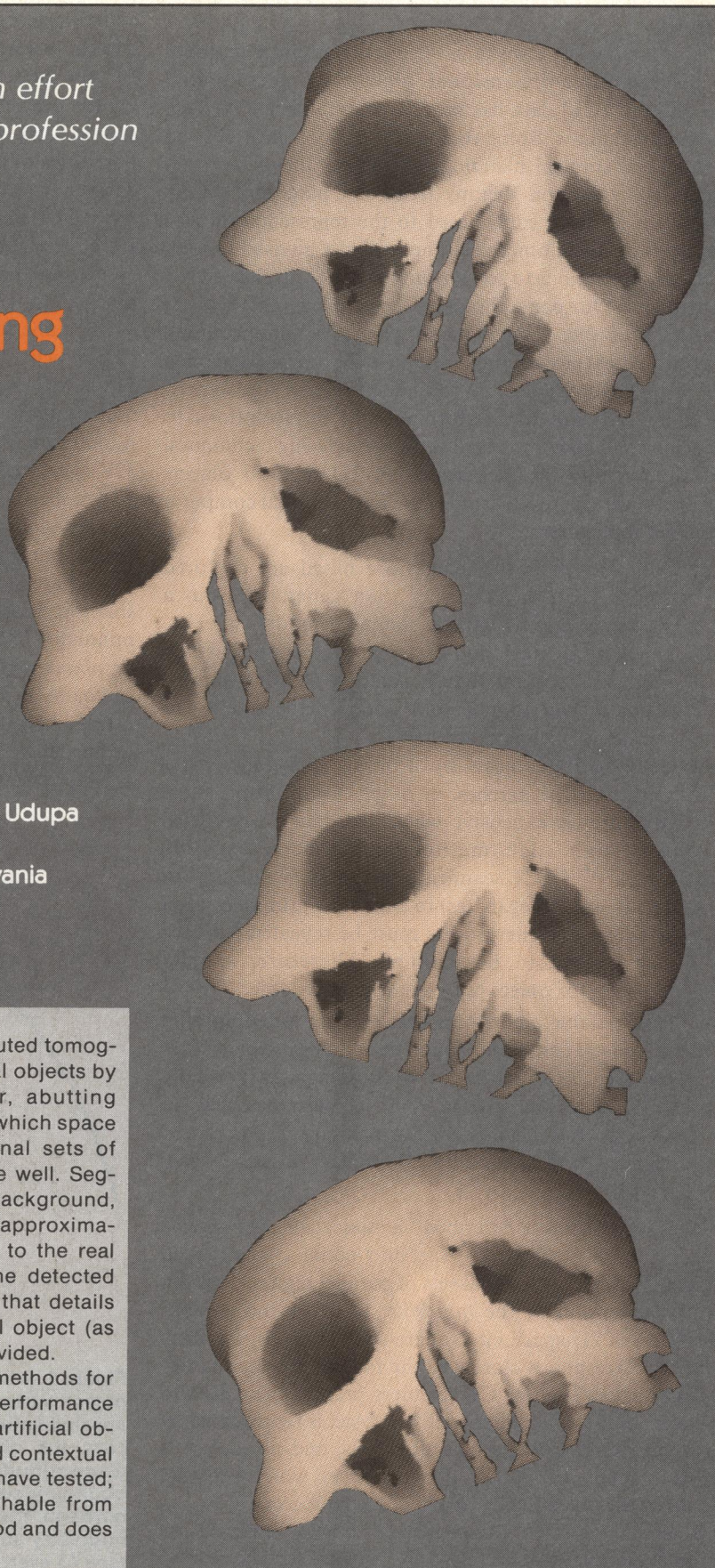
Computed tomography and the cuberille model—an effort to better serve the medical profession and its patients.

Surface Shading in the Cuberille Environment



Lih-Shyang Chen, Gabor T. Herman,
R. Anthony Reynolds, and Jayaram K. Udupa

Hospital of the University of Pennsylvania

Medical imaging devices (such as computed tomography scanners) often produce data on real objects by assigning values to small, rectangular, abutting volumes of space. The cuberille model, in which space is dissected by three mutually orthogonal sets of parallel planes, represents such data quite well. Segmentation of the cuberille into object and background, followed by boundary detection, gives an approximation (consisting of faces in the cuberille) to the real object's surface. Our aim is to render the detected boundary surface on a display screen so that details are retained, but the appearance of a real object (as opposed to one consisting of cubes) is provided.

We propose here a number of shading methods for achieving this aim, and compare their performance with existing methods using medical and artificial objects. Overall, we find our new normal-based contextual shading method superior to the others we have tested; it provides images practically indistinguishable from those provided by the Phong shading method and does so at significantly reduced cost.

0272-1716/85/1200-0033$01.00 © 1985 IEEE

Over the last few years, the use of computer graphics to display three-dimensional objects from computed tomography (CT) data has become an established medical imaging technique. Some recent applications have been to craniofacial surgical planning,[1,2] visualization of brain lesions,[3] and radiation therapy treatment planning.[4] Many recent papers have contributed to the representation and display of medical objects.[5–7] The reader is advised to look at these publications to see why and how 3D display is used in medicine. Our paper discusses the problems that arise in rendering objects using a model particularly appropriate for the medical environment, the so-called *cuberille* model. We propose some solutions to these problems, and experimentally compare the performance of our solutions to those of previously proposed surface-rendering techniques.

Requirements for the display of 3D medical objects differ in several respects from those in traditional computer graphics applications:

1. Data on which the images are based are captured from a real object, rather than generated from a mathematical or synthetic computer model.
2. The accurate depiction of actual objects under study is more important than generating pleasing images, although we do not find good image quality to conflict with clinical accuracy.
3. Representing objects by arbitrary polygons, parametric surface patches, or other 2D or 3D primitive structures is expensive when real data are involved (although such methods have been used by others[3–5,8–11]). CT scanners generate large amounts of data (several megabytes) to which the primitives must be fitted, and the process must be repeated for each new patient to be scanned. Therefore, simpler methods for object representation are needed.
4. Fast, reliable algorithms that execute on computing equipment commonly found in radiology departments are essential, since the cost of the imaging will usually be borne by the patient. All of the work described in this paper was done on the computing equipment supplied with a standard CT scanner (General Electric 8800).

We have previously described[12] a model called a *cuberille* which is a dissection of space into equal cubes (called *voxels*) by three orthogonal sets of parallel planes. This is a natural extension of the *quadrille,* which is the familiar dissection of 2D space into square-shaped pixels. Furthermore, it corresponds closely to the format in which radiological devices such as CT scanners present their data to the user. In the cuberille model, an object is represented by a (finite) subset of the set of all voxels, and its surface is represented by the set of directed *faces* which separate voxels in the object from voxels in the background.

Since the surface of a medical object tends to be complex, and minute details in the original surface have to be retained in its presentation, the number of faces in a cuberille representation is usually very large. In the two clinical examples used in this paper there are 210,874 and 133,424 faces (more than 500,000 faces is not uncommon). On the other hand, all the faces are of equal size and there are only six possible face orientations, with the normal pointing along a positive or negative coordinate axis. From a given viewpoint, only three of these orientations can result in visible faces.[13] Furthermore, we can simplify the Z-buffer algorithm we use for hidden-surface removal by extracting the distance computation from the loop over pixels onto which a face projects.[12]

The cuberille model lends itself to organizations of the data which allow the Z-buffer to be partitioned.[14] Thus, the 512×512 images of the rather complex surfaces used for illustration in this article were produced in from two to five minutes on a minicomputer (the DG Eclipse S/200) using 32K words of main memory and Fortran programming. The cuberille environment also lends itself to parallel display algorithms, making possible real-time display by special-purpose hardware.[15–17]

A major problem in the cuberille model is surface shading. In order to explain this, we introduce some terminology: We refer to a physical object's actual surface as an *object surface;* we refer to the surface made up of faces of voxels as a *boundary surface.* The boundary surface is usually obtained through a boundary detection process.[18] In the cuberille environment, a boundary surface is used as an approximation to an object surface (see Figure 1). In

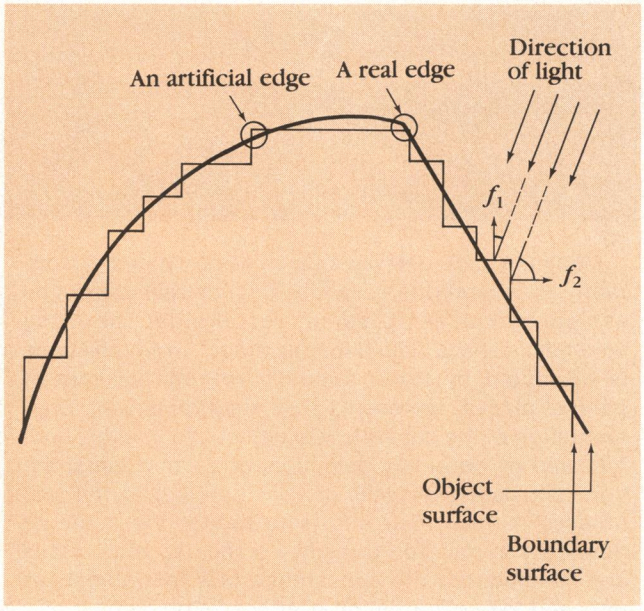


Figure 1. A two-dimensional depiction of the notions of an object surface and the boundary surface which is the approximation to it in the cuberille environment: In this case the object surface has only one (real) edge (a discontinuity in the normal to the surface), but the boundary surface has many (artificial) edges. The angle between the direction of light and the normal to the boundary surface can change from small (see f_1) to large (see f_2) even when the corresponding angle for the object surface is constant.

rendering the boundary surface, we desire to make it appear as similar as possible to the unknown object's surface appearance under certain lighting conditions. The appearance of a surface at one of its points depends in particular on the angle between the normal to the surface at that point and the direction of light.

Clearly, even if a boundary surface closely approximates an object surface (in the sense that any point on either surface is near some point of the other), it is unlikely that the normal at a point on the boundary surface will make a direction even approximately equal to the direction of the normal to the object surface at a nearby point. Thus, an accurate rendering of the boundary surface is unlikely to be a good representation of the object surface. By its very nature, a boundary surface will have sharp edges (those of the individual voxels) at points where the corresponding object surface is quite smooth. We refer to such edges as *artificial;* by contrast, we call those edges *real* which correspond to edges in the object surface (see Figure 1).

We have investigated six shading procedures, three of which we recently developed. The method we call normal-based contextual shading (presented here for the first time) seems to be superior to other methods because it produces images which are:

- accurate representations of the underlying object surfaces both in stationary displays and in dynamic displays created by rotation of the object,
- visually pleasing, and
- computationally inexpensive.

We demonstrate these results with two artificial objects (one spherical and one made up from planes) and two clinical objects (a spine and a skull).

Shading methods

In all the shading techniques discussed in this paper, the input is a boundary surface. The image of this surface is calculated by orthographic projection; that is, a point P on the surface of the object is mapped into a point Q on the display in such a way that QP is always orthogonal to the display screen. If P is a visible point on the surface, we use the phrase "the *shading* assigned to P" to describe the intensity on the display screen at the point Q onto which P projects. The methods we discuss differ in the way shading is assigned to points on the boundary surface.

The idea is that even though we obtain the image by projecting points of the boundary surface onto the display screen, we should assign shadings to these points so that the image adequately represents the underlying object surface. One way of doing this is to estimate, at a point P of the boundary surface, the direction of the normal to the object surface at the point nearest to P (we refer to this normal vector as the *object normal* at P). The different shading methods discussed in this paper estimate and use the object normal in different ways. However, they all make use of the following basic formula:

$$S(P) = \left[\frac{M-L}{2R}\ (R-d)\ N(\theta) + L\right]_{L}^{M} \qquad (1)$$

where

$$[v]_{L}^{M} = \begin{cases} L, & \text{if } v < L \\ v, & \text{if } L \le v \le M \\ M, & \text{if } M < v \end{cases} \qquad (2)$$

In this formula, d is the distance of P from the source of light and θ is the angle between the direction of light and the estimated object normal at P. $N(\theta)$ is a function of θ which simulates the properties of a diffuse reflecting surface. In some cases, it is the manner in which the object normal is estimated (and thus the value of θ) which differs from method to method; in other cases, the function N is also varied. For simplicity of implementation we have assumed in all cases that the light rays are orthogonal to the display screen; thus, we do not generate shadows. The other variables (M, L and R) are user-defined constants. For the illustrations in this paper we have chosen the following values:

$M = 255$,
$L = 30$,
R = the radius of the smallest sphere that encloses the smallest rectangular box (with faces parallel to the voxel faces) enclosing the boundary surface.

The values of these parameters are to some extent dictated by the available hardware (e.g., $M = 255$ since our display unit is capable of displaying 256 different gray levels), and have been chosen after extensive experimentation to optimize image quality (e.g., we have chosen $L = 30$ to represent ambient or background light).

In the implementation of the methods discussed in this paper, the transformation from object space to image space, hidden-surface removal, and scan-conversion (area-fill of the projected faces) are all accomplished in the same manner.[19] In all cases we use the following simple anti-aliasing method: The image is calculated at 512 × 512 resolution, but displayed at 256 × 256 resolution, employing a nine-point averaging scheme.[20]

The six shading methods we discuss in the following subsections are: distance-only shading; a version of constant shading; two versions of contextual shading (one image-based, the other based on the estimated object normal); Phong shading; and gradient shading. We recently developed the two versions of contextual shading, and gradient shading.

Distance-only shading. One way to avoid the difficulty of estimating the object normal is simply to ignore it. This can be achieved by setting $N(\theta)$ equal to the constant 1 in Formula 1. In our implementation of this well-known method we use Formula 1 to calculate the shading at the center of each visible voxel face, and then assign the same

shading to all visible points on the same face. This is justified, since in the medical application voxel faces are minute, and precise calculation of distance d from the source of light would not make a noticeable difference to the images.

Distance-only shading produces smooth looking images. This is because the distance d for neighboring faces is about the same. Consequently, small structural features are not noticed, resulting in a loss of textural information. If the texture is due to digitization alone, its loss may be an advantage; however, our experience shows that small, clinically important details in the object surface are also lost. Another difficulty is that real-edge information is lost, since faces on either side of an edge are similarly distant from the light source. Since much of human perception is edge-detection oriented, this is an important drawback—especially since textures and real edges can be important cues in determining an object's shape. In spite of such drawbacks, distance-only shading has been used extensively in medical computer graphics literature because of its ease of implementation.[2]

Constant shading. A simple way of estimating the object normal at a point P is to use the normal to the boundary surface at P. We call this method "constant shading" since it is analogous to the method used in computer graphics to shade a polygon mesh that represents a faceted rather than a curved surface.[21] Constant shading is appropriate when the normals to the boundary surface approximate the normals to the object surface, a situation which is more likely to arise for artificial objects than for medical ones. Nevertheless, constant shading is worth investigating since it provides more structural information to the viewer than does distance-only shading.

With constant shading, the intensity assigned to all visible points on a face on the boundary surface is provided by Formula 1, applied to its center point P with

$$N(\theta) = [\cos(\frac{\theta}{n})]^p \qquad (3)$$

In all experiments reported in this paper the values assigned to p and n are: $p = 0.6$, and $n = 2$. (The variables p and n appear also in the methods described in the next four subsections. The same values for p and n have been used in the experimental evaluation of all five shading methods. These values have been chosen empirically after extensive experimentation.)

A major problem with constant shading is the strong appearance of artificial edges at certain angles of rotation. The reason for this is illustrated in Figure 1, in which we see that constant shading will assign considerably greater brightness to face f_1 than to f_2, even though the underlying object surface is smooth. This prediction of constant shading behavior is borne out by experiments reported in the next section. Note that our choice of $p = 0.6$ and $n=2$ in Formula 3 reduces the effect of artificial edges, as compared to the choice $p = 1.0$ and $n=1$, which corresponds to a pure cosine shading rule. Making p very small makes constant shading indistinguishable from distance-only shading. Much of our early work in 3D display used constant shading.[12]

Image-based contextual shading. We have designed contextual shading to overcome the problems caused by artificial edges in the cuberille environment.[22] The idea is to assign a shade to points on a face that depends not only on the orientation of the face itself, but also on the orientations of the four faces which share an edge with it. (We call these the *adjoint faces.*)

Image-based contextual shading[22] is based on the observation that major problems in constant shading arise when the θ in Formula 3 is small for a particular face, but not nearly so small for an adjoint face (see Figure 1). For each orientation of the surface those visible faces, whose normals make less than 45° with the direction of light, are called *critical;* the rest of the visible faces are called *noncritical.* Image-based contextual shading reduces the effects of artificial edges between critical and noncritical faces by an averaging process.

More precisely, let θ_0 be the angle between the normal to a face f and the direction of light, and let θ_1, θ_2, θ_3, and θ_4 be similarly defined angles for the four adjoint faces (according to some fixed labeling scheme).[19] Then the shading assigned to all points on the face f is determined by Formula 1 evaluated at its center point P with

$$N(\theta) = \sum_{i=0}^{4} w_i [\cos(\frac{\theta_i}{n})]^p, \quad \sum_{i=0}^{4} w_i = 1, \qquad (4)$$

c.f., Formula 3. The weights w_i depend on whether the face f is critical or noncritical, and on the number of visible adjoint faces which are critical. In particular, if f is critical then zero weight is given to adjoint critical faces; if f is noncritical then zero weight is given to adjoint noncritical faces. Thus the shading assigned to a noncritical face surrounded by noncritical faces is the same as in the constant shading method, while the shading of a critical face surrounded by noncritical faces will be smoothed. Table 1 shows the values of w_0 used for the critical and noncritical faces as a function of the number n_c of visible critical faces in the neighborhood. In our implementation the nonzero values of w_i, for $1 \leq i \leq 4$, have been chosen to be $(1\text{-}w_0)/n$,

Table 1.
The weight w_0 used in the image-based contextual shading method.

	Number of visible critical faces in the neighborhood				
	0	1	2	3	4
Critical	1/16	1/8	1/4	1/2	1
Noncritical	1	15/16	7/8	3/4	1/2

where n is the number of visible critical faces adjacent to f if f is noncritical; otherwise, n is the number of visible noncritical faces adjacent to f. If $n=0$, then for $1 \leq i \leq 4$, w_i is assigned a zero value, and the shading assigned is the same as in the constant shading method. Since the major problem we have found with image-based contextual shading (to be stated below) is not dependent on the choice of the w_i's, we shall not repeat here the rather involved formulas by which the shading is determined. We concentrate instead on the implementation, which is similar for both image-based and normal-based contextual shading methods. The implementations of both contextual shading techniques are very efficient (as compared, for example, with the Phong method to be discussed below).

In the cuberille environment, for any face f there are 81 possible arrangements of the adjoint faces relative to f (three independent possibilities at each of the four edges, as shown in Figure 2). Our boundary detection algorithm[18] has been modified so that with each face in the boundary surface we store in a one byte *neighbor-code* the information as to which of the 81 possible cases apply. Also during boundary detection the faces in the boundary are sorted into six sets, one for each of the possible directions in the cuberille model. In displaying the visible surface for a particular object orientation, we need to loop through (at most) three of these six sets. The important observation is that for each of these sets there are only 81 possible values of $N(\theta)$ in Formula 4. We precalculate these values and store them in a look-up table. Then, as we loop through the faces in the set, we find the value of $N(\theta)$ for the face by making a simple look-up based on the neighbor-code. Thus, contextual shading is no more expensive than constant shading in the cuberille environment.

The major problem with image-based contextual shading can be discussed with reference to Figure 1. With the indicated direction of light (which is also the observer position) there are both critical and noncritical visible faces to the right of the real edge, while there are only noncritical visible faces to the left of the real edge. Thus, to the left of the real edge image-based contextual shading is identical to the smooth-looking distance-only shading (except for a multiplicative factor), while to the right of the real edge it is a somewhat smoothed version of the originally rough-looking constant shading. If the direction of the light was from top left (rather than top right) then the right side of the real edge would appear smooth and the left sides textured. Thus the appearance of the object changes in an essential (and unnatural) way, depending on its orientation relative to the direction of light.

Normal-based contextual shading. This new method, reported here for the first time, has been designed to overcome the above problem, but without any loss of implementational efficiency. The idea is to estimate the object normal at the center P of a face based on its orientation and neighbor-code, and then simply use Formulas 1 and 3 with θ as the angle between the object normal at P and the direction of light.

The particular way in which we define the object normal ω at P is illustrated in Figure 2. There is a right-handed (u, v, w)-coordinate system (with its origin at P) in which w points away from the object and u and v are parallel to the edges of the face (Figure 2a). The normal vector ω is defined as the cross-product of vectors μ (in the u-w plane) and ν (in the v-w plane) (Figure 2b). The vectors μ and ν each can have (independently) one of five possible directions, depending on the adjoint faces in the positive and negative u-directions (for μ) and v-directions (for ν). The

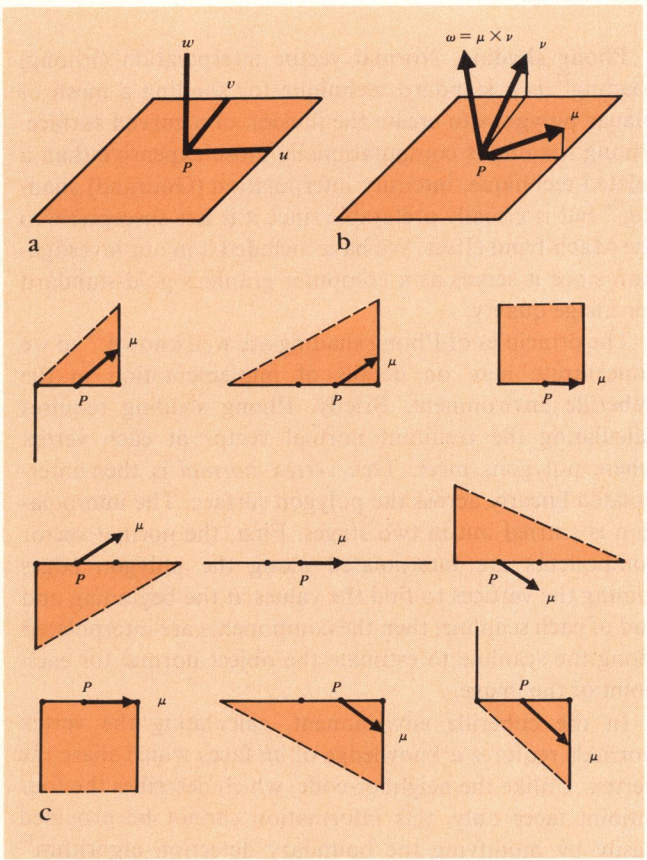


Figure 2. (a) Local coordinate system for the face of a voxel. The positive *w*-axis points from the center of the face away from the object. The *u*- and *v*- axes are parallel to the edges of the face, so that (*u*, *v*, *w*) is a right-handed coordinate system. (b) Determination of the object normal ω for normal-based contextual shading. The vector ω is the cross product of vectors μ (in the *u*-*w* plane) and ν (in the *v*-*w* plane). (c) Determination of the vector μ (determination of ν is analogous). There are five possible directions of μ (making angles $\pm 45°$, $\pm \tan^{-1}(0.5)$, and 0° with the positive *u*-axis), and these are determined by the ns of the adjoint faces in the positive and negative *u* directions. There are nine possible arrangements of these adjoint faces. These, together with the nine possible arrangements for the *v*-axis, result in a total of 81 possible arrangements of adjoint faces, and give rise to 25 different orientations of ω.

exact method by which μ (and similarly ν) is determined is illustrated in Figure 2c for each of the nine arrangements of the adjoint faces in the positive and negative u-directions.

Clearly, the implementational properties of normal-based contextual shading are the same as those of image-based contextual shading—its advantage over image-based contextual shading is that the results are rotationally invariant; that is, the texture pattern does not change appreciably as the object is rotated. One disadvantage of the method is that for a given face f, of the 81 possible contexts only 25 produce distinct values of ω (Figure 2). However, when the face orientation and distance are taken into account we find that a sufficient range of intensity values is generated, and the quality of the images is quite outstanding.

Phong shading. Normal vector interpolation (Phong) shading[23] is a standard technique for shading a mesh of planar polygons to create the illusion of a curved surface. Phong shading is computationally more expensive than a related technique, intensity interpolation (Gouraud) shading,[24] but is visually preferable since it is less susceptible to the Mach-band effect. We have included it in our investigation since it serves as a computer graphics gold-standard for image quality.

The principles of Phong shading are well known,[21] so we concentrate here on details of implementation in the cuberille environment. Briefly, Phong shading requires calculating the resultant normal vector at each vertex where polygons meet. This *vertex normal* is then interpolated linearly across the polygon surface. The interpolation is carried out in two stages. First, the normal vector components are interpolated along the polygon edges joining the vertices to find the values at the beginning and end of each scanline; then the components are interpolated along the scanline to estimate the object normal for each point of the image.

In the cuberille environment, calculating the vertex normals requires a knowledge of *all* faces which share the vertex. Unlike the neighbor-code, which describes the four adjoint faces only, this information cannot be provided easily by modifying the boundary detection algorithm[18] that we use to obtain the boundary surface. Therefore, a second pass through the data is needed, increasing the pre-processing time.

We use a stand-alone program to obtain and append to the face descriptor the vertex normal information for each face f in the boundary surface. We found that Phong shading requires appending five bytes to the three-byte face descriptor (the neighbor-code is not needed) for a total of eight bytes per face. To render a boundary surface, therefore, the amount of data Phong shading has to store and process is twice what is needed by contextual shading.

When a face is processed, the vertex normals are interpolated and normalized to produce a value for θ in Formulas 1 and 3. Note that, unlike the methods previously described in which we assigned the same shading to all points on a face f, Phong shading is separately calculated for each point P projecting onto a display pixel. This is unimportant in practice, however, since we are working with objects whose voxels are similar in size to the pixels of the image.

The Phong shading method produces excellent results, perhaps superior to the other methods we have investigated; nevertheless, we find Phong and normal-based contextual shading to be visually very similar. However, Phong shading is expensive for our application because of the extra pre-processing step, the extra storage, and the extra computation to interpolate the vectors.

To be specific, the following timings apply on the computer of our CT scanner (a DG Eclipse S/200). Creation of a surface with 200,000 faces to be displayed from the original CT data takes typically 15 minutes. The surface detection process[18] includes finding and storing the neighbor-code for each face. The production of such a surface's image, using our normal-based contextual shading, is under three minutes. In order to do Phong shading we need the extra pre-processing step, which in our implementation took over five hours. After the pre-processing, the creation of each image with Phong shading takes over five minutes. Thus, for a typical medical object and a typical 36 views, the total time needed for the contextual normal-based shading is well under two hours—using Phong shading this same process takes over eight hours.

We did not take extreme care to optimize the Phong shading procedure, and the exact figures might change by such optimization; still, normal-based contextual shading will be significantly faster than Phong shading. It must be emphasized, however, that Phong shading is applicable to a much larger class of surfaces than those found in the cuberille environment.

Gradient shading. Except for distance-only shading, the shading methods we have discussed so far may be termed *object-space shading* methods since they require extra data (face orientation, neighbor-codes, or vertex normals) to be stored with the object representation. In some instances (e.g., Phong shading), the extra data are expensive to compute; in all instances, they must be recomputed frequently if the user wants to modify the object representation interactively as is desirable for surgical planning.[25]

To avoid these problems we have recently devised an *image-space shading* method (gradient shading)[26] that uses no input data other than a complete pre-image obtained by coordinate transformation, hidden-surface removal, and scan-conversion. The pre-image must contain, for each pixel, the distance to the visible point of the object that projects onto that pixel. One suitable pre-image (used for the gradient-shaded images presented in this paper) is the Z-buffer used for hidden-surface removal; another is a distance-only shaded image, since with $N(\theta)=1$ the distances can be recovered from the intensities by solving Formula 1 for d.

The pre-image is, in effect, a representation of the visible

object surface of the form $z = z(x, y)$. It follows that we can estimate the object normal ω at any point (x, y) by finding the gradient vector ∇z at that point. Intensity can then be computed using Formulas 1 and 3 with θ being the angle between the estimated object normal and the direction of light. This is the basis of gradient shading. There are, however, some problems with the algorithm as stated above. For example, it is necessary to ensure that the gradient operator performs correctly at discontinuities. We require a method of distinguishing gradual changes along a single surface as contrasted with abrupt changes from one surface to another. Since these problems and their solutions are addressed elsewhere,[26] only a brief description is given here.

Given the surface $z = z(x, y)$, the normal at any point may be obtained from the vector

$$\left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, -1\right). \tag{5}$$

This necessitates obtaining the derivatives $\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}$, numerically. They can be estimated from the backward difference δ_b, the forward difference δ_f, or the central difference δ_c.[27] The central difference would provide the best approximation if z were a continuous function of x and y; however, this is not usually the case. The solution we have adopted is to take a weighted average of the forward and backward differences

$$\frac{\partial z}{\partial x} \approx \frac{W(|\delta_b|) \cdot \delta_b + W(|\delta_f|) \cdot \delta_f}{W(|\delta_b|) + W(|\delta_f|)} \tag{6}$$

where, for $\delta \geq 0$,

$$W(\delta) = \begin{cases} 1, \text{ if } \delta \leq a, \\ \epsilon, \text{ if } \delta \geq b, \\ \frac{1+\epsilon}{2} + \frac{1-\epsilon}{2} \cos \pi \left(\frac{\delta - a}{b - a}\right), \text{ otherwise.} \end{cases} \tag{7}$$

In our implementation we have used $a = 2$, $b = 5$, and $\epsilon = 10^{-5}$.

The main advantage of gradient shading is that no extra data need be stored with the face descriptor. A second advantage is that it is asymptotically very efficient, since the complexity is proportional to the number of pixels and independent of the number of faces. The weighted average of forward and backward differences[27] ensures that gradient shading averages out the artificial edges (found with constant shading) with the result that it is nearly as good visually as normal-based contextual or Phong shading. One disadvantage is that the method is not rotationally or dilationally invariant: The shading patterns may change as the object is rotated or scaled. However, this effect has not been noticeable with the objects we have tested.

Results

We use four objects to illustrate the images produced by the shading methods described in this paper. The first object's surface consists of polygonal facets. Its cuberille representation is obtained by the directed contour method of representing and creating digital objects.[28] The second data set is a digital approximation of a sphere. The third data set is obtained from a post-operative CT scan of the head of a patient who underwent reconstructive surgery of his left orbit. The fourth data set is based on the CT scan of part of a patient's cervical spine.

Based on a combined consideration of image quality and computational efficiency, the newly proposed normal-based contextual shading method seems to fare best in the cuberille environment.

In Figures 3, 4, 5, and 6, we show (for each of the data sets) the images produced by the six shading methods. The orientation of the boundary surfaces used in these images is produced by an initial tilt of 45° about the x-axis followed by a rotation of 6° about the y-axis.

The loss of real edge information in the distance-only shading method is apparent in Figures 3a, 5a, and 6a. The loss of texture information can be appreciated by comparing the image of the skull in Figure 5a with the corresponding images in Figures 5d, e, and f.

Constant shading tends to emphasize artificial edges. This is apparent in Figures 3b, 4b, 5b, and 6b.

The image-based contextual method (Figures 3c, 4c, 5c, and 6c) replaces the strong artificial edges with a smooth texture. However, there is a significant difference between the appearances of the two sides of the images.

The normal-based contextual method (Figures 3d, 4d, 5d, and 6d) eliminates this problem. It also appears to produce better edge definition than the image-based method (compare Figures 3c and 3d).

The Phong method (Figures 3e, 4e, 5e, and 6e) produces images nearly identical to those produced by normal-based contextual shading, but it is computationally expensive.

The quality of the images in the gradient shading method (Figures 3f, 4f, 5f, and 6f) is generally not quite as good as those produced by normal-based contextual shading or Phong shading (see especially Figure 3), but is better than those produced by the other techniques we have investigated.

Figures 3, 4, 5, and 6 show images produced by six shading methods: (a) Distant-only shadings; (b) constant shading; (c) image-based contextual shading; (d) normal-based contextual shading; (e) Phong shading; and (f) gradient shading.

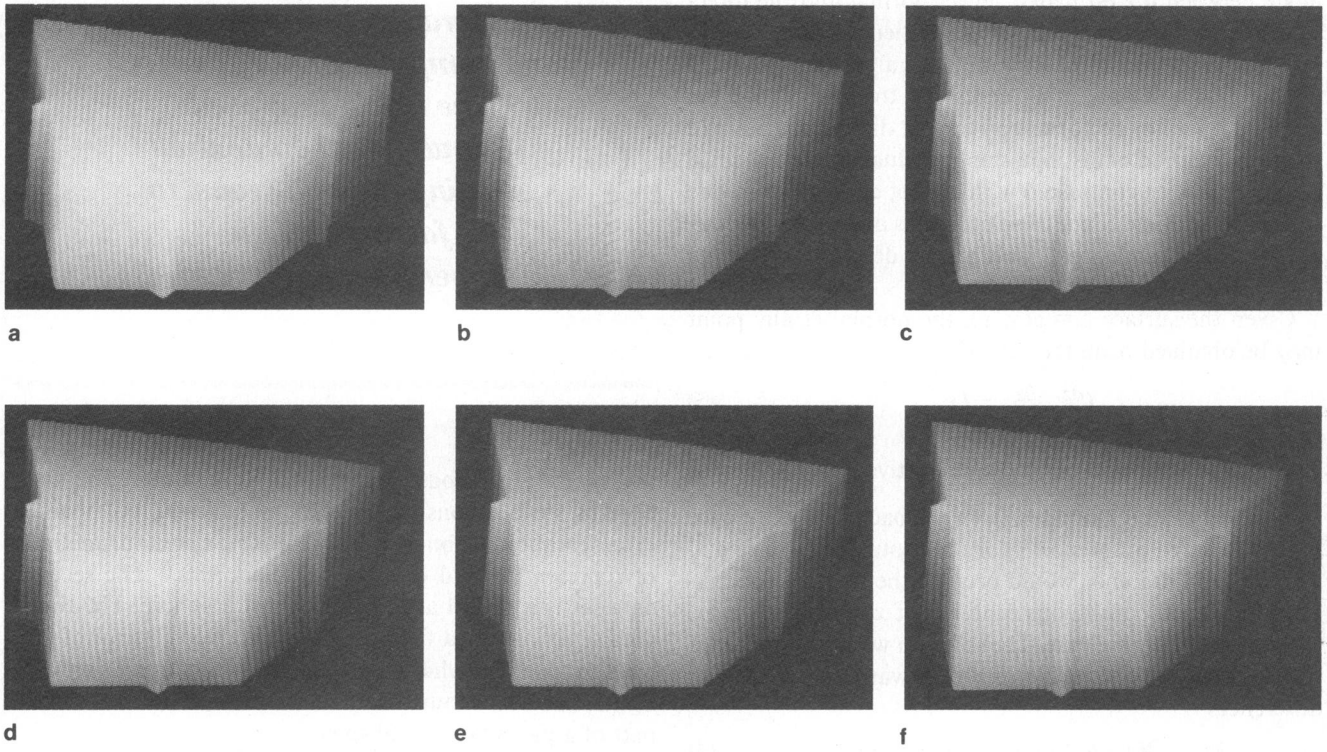


Figure 3. 3D displays of polygonal facets.

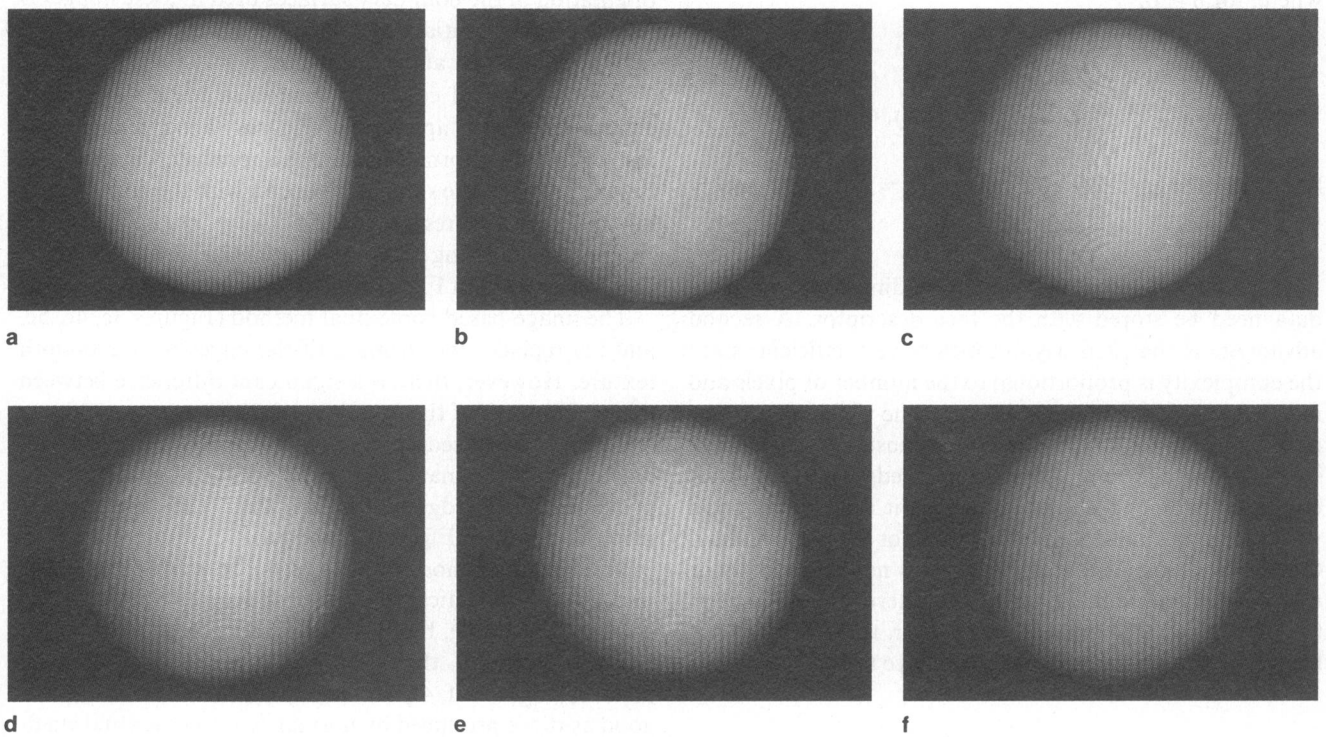


Figure 4. 3D displays of a sphere.

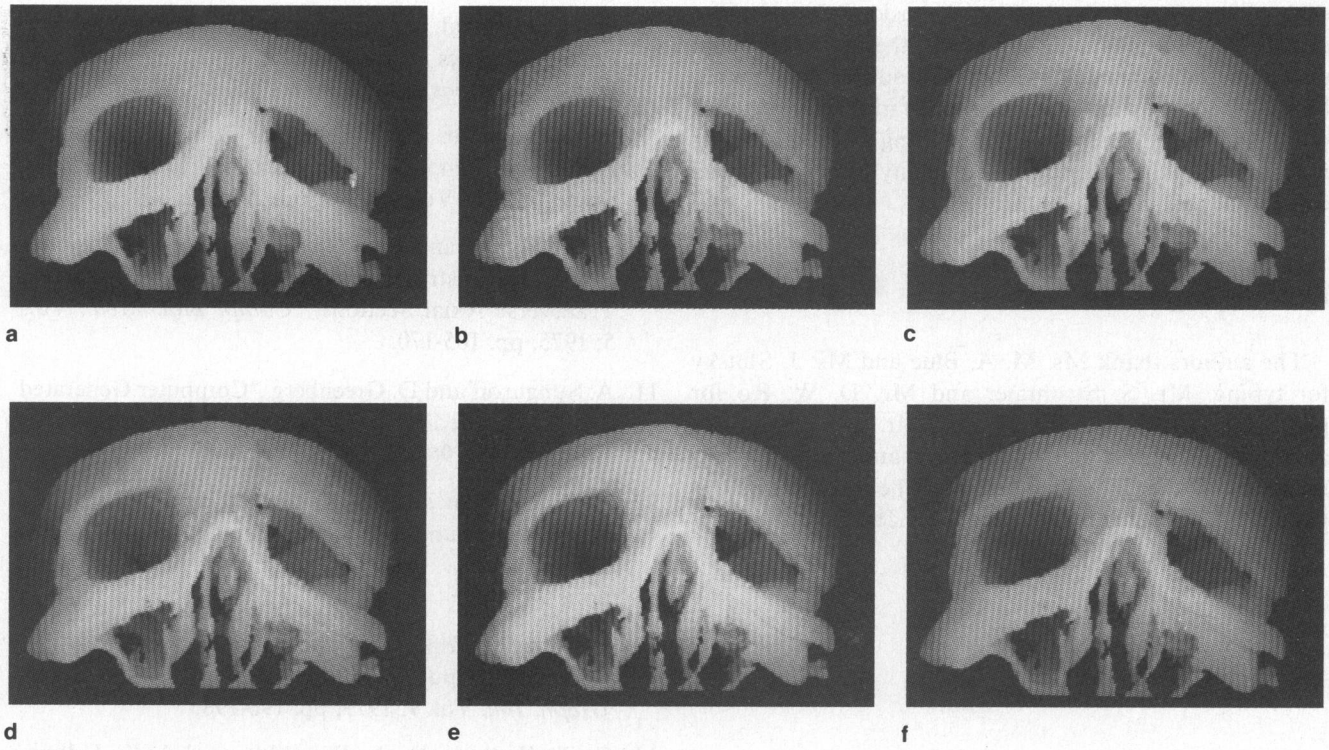


Figure 5. 3D displays of a skull.

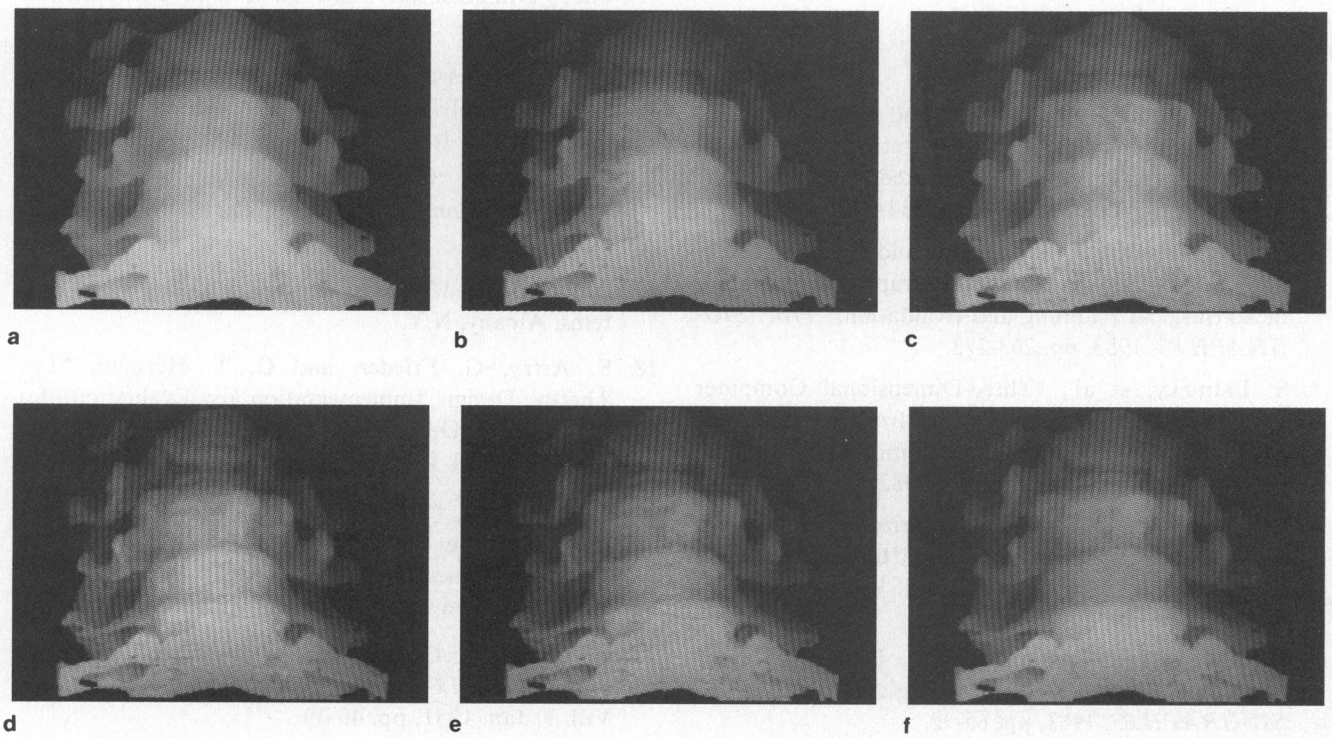


Figure 6. 3D displays of a spine.

Conclusions

We have described, discussed, and experimentally investigated six shading methods appropriate to the cuberille environment. Based on a combined consideration of image quality and computational efficiency, the newly proposed normal-based contextual shading method seems to fare the best. A possible exception is the area of interactive surgical planning, where the convenience of using gradient shading may outweigh the superior image quality of normal-based contextual shading.

Acknowledgments

The authors thank Ms. M. A. Blue and Ms. J. Slutsky for typing, Mr. S. Strommer and Mr. D. W. Ro for photography, Dr. S. Trivedi and Mr. G. Waxler for assistance in creating the digital sphere, and Prof. H. Fuchs for comments on an earlier version. The research for this paper was supported by NIH grant HL28438.

References

1. D. C. Hemmy, D. J. David, and G. T. Herman, "Three-Dimensional Reconstruction of Craniofacial Deformity Using Computed Tomography," *Neurosurgery,* Vol. 13, 1983, pp. 534-541.
2. M. W. Vannier, J. L. Marsh, and J. O. Warren, "Three-Dimensional Computer Graphics for Craniofacial Surgical Planning and Evaluation," *Proc. SIGGRAPH 83,* 1983, pp. 263-273.
3. S. Batnizky, et al., "Three-Dimensional Computer Reconstructions of Brain Lesions from Surface Contours Provided by Computed Tomography: A Prospectus," *Neurosurgery,* Vol. 11, 1982, pp. 73-84.
4. D. L. McShan, et al., "A Computerized Three-Dimensional Treatment Planning System Utilizing Interactive Color Graphics," *Brit J. Radiology,* Vol. 52, 1979, pp. 478-481.
5. H. Fuchs, G. D. Abram, and E. D. Grant, "Near Real-Time Shaded Display of Rigid Objects," *Proc. SIGGRAPH 83,* 1983, pp. 65-72.
6. M. L. Rhodes, "Toward Fast Edge Detection for Clinical 3-D Applications of Computer Tomography," *Proc. SIGGRAPH 79,* 1979, pp. 134-142.
7. J. K. Udupa, "Display of 3D Information in Discrete 3D Scenes Produced by Computerized Tomography," *Proc. of IEEE,* Vol. 71, 1983, pp. 420-431.
8. H. Fuchs, et al., "Adding a True 3D Display to a Raster Graphics System," *IEEE Computer Graphics and Applications,* Vol. 2, Feb. 1982, pp. 73-78.
9. W. V. Glenn, et al., "Multiplanar Display Computerized Body Tomography Applications in the Lumbar Spine," *Spine,* Vol. 4, 1979, pp. 282-352.
10. H. K. Huang and R. S. Ledley, "Three-Dimensional Image Reconstruction from *In Vivo* Consecutive Transverse Axial Sections," *Comp. Biol. Med.,* Vol. 5, 1975, pp. 165-170.
11. A. Sunguroff and D. Greenberg, "Computer Generated Images for Medical Application," *Proc. SIGGRAPH 78,* 1978, pp. 196-202.
12. G. T. Herman and H. K. Liu, "Three-Dimensional Display of Human Organs from Computed Tomograms," *Proc. Comp. Graph. Im.,* Vol. 9, 1979, pp. 1-21.
13. E. Artzy, "Display of Three-Dimensional Information in Computered Tomography," *Proc. Comp. Graph. Im.,* Vol. 9, 1979, pp. 196-198.
14. G. T. Herman, R. A. Reynolds, and J. K. Udupa, "Computer Techniques for the Representation of Three-Dimensional Data on a Two-Dimensional Display," *Proc. SPIE,* Vol. 367, 1982, pp. 3-14.
15. S. M. Goldwasser, et al., "A 3D Physicians Workstation with Real-Time Performance," *IEEE Computer Graphics and Applications,* Dec. 1985.
16. D. Meagher, "Geometric Modelling Using Octree Encoding," *Comp. Graph. Im. Proc.,* Vol. 19, 1982, pp. 129-147.
17. *Insight Solid Modelling System,* Phoenix Data Systems, Albany, N.Y.
18. E. Artzy, G. Frieder, and G. T. Herman, "The Theory, Design, Implementation and Evaluation of a 3D Surface Detection Algorithm," *Comp. Graph. Im. Proc.,* Vol. 15, 1981, pp. 1-24.
19. J. K. Udupa, "DISPLAY82 — A System of Programs for the Display of 3D Information in CT Data," Technical Report MIPG67, Department of Radiology, University of Pennsylvania, 1983.
20. F. Crow, "A Comparison of Anti-Aliasing Techniques," *IEEE Computer Graphics and Applications,* Vol. 1, Jan. 1981, pp. 40-49.
21. J. D. Foley and A. Van Dam, *Fundamentals of Interactive Computer Graphics,* Addison-Wesley, Reading, Mass., 1982.

22. G. T. Herman and J. K. Udupa, "Display of Three-Dimensional Discrete Surfaces," *Proc. SPIE,* Vol. 283, 1981, pp. 90-97.

23. B. T. Phong, "Illumination for Computer Generated Pictures," *Comm. ACM,* Vol. 18, 1975, pp. 311-317.

24. H. Gouraud, "Continuous Shading of Curved Surfaces," *IEEE Trans. Comp.*, Vol. C-20, 1971, pp. 623-628.

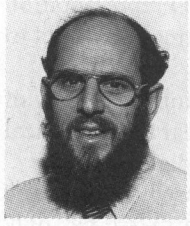
25. L. J. Brewster, et al., "Interactive Surgical Planning," *IEEE Computer Graphics and Applications,* Vol. 4, Mar. 1984, pp. 31-40.

26. D. Gordon and R. A. Reynolds, "Image Space Shading of Three-Dimensional Objects," *Comp. Vision Graph. Im. Proc.,* Vol. 29, 1985, pp. 361-376.

27. G. Dahlquist, A. Bjorck, and N. Anderson, *"Numerical Methods,"* Prentice-Hall, Englewood Cliffs, N.J., 1974.

28. J. K. Udupa, "Interactive Segmentation and Boundary Surface Formation for 3D Digital Images," *Comp. Graph. Im. Proc.,* Vol. 18, 1982, pp. 213-235.

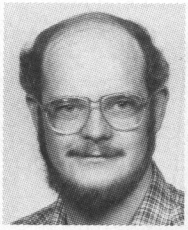
Lih-Shyang Chen received his BS and MS in electrical engineering from National Cheng-Kung University, Taiwan in 1978 and 1980, respectively. He has been a PhD student in the Department of Computer and Information Science at the University of Pennsylvania, Philadelphia, since 1983. He has several publications in medical imaging and also in the area of computer peripherals and devices. He is a member of Phi Tau Phi Honor Society. His research interests include computer graphics, image and signal processing, computer vision, and computer peripherals and devices.



Gabor T. Herman is currently a professor, and Chief of the Medical Imaging Section, Department of Radiology, at the University of Pennsylvania. His main research interest is biomedically motivated computer science.

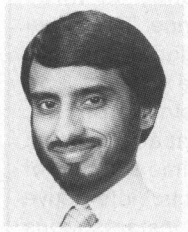
From 1969 to 1981, he was with the Department of Computer Science, SUNY at Buffalo, where he became Director of the Medical Image Processing Group in 1976.

Herman received his MSc and PhD in mathematics from the University of London in 1964 and 1968, respectively. He received an MS in engineering science from the University of California in 1966. He is a senior member of IEEE.



R. Anthony Reynolds is currently with Dynamic Digital Displays (3D) Inc., where he is Director of Clinical Applications. He is a founding member of 3D Inc., which was formed to develop hardware and software approaches to 3D display of medical objects. He has been active in medical imaging since 1971, and from 1976 to 1980 was a physicist with the Cancer Control Agency of British Columbia, Canada, working with computers and imaging systems for radiation therapy planning.

Reynolds obtained a BA in physics from Trinity College, Dublin, in 1971, and an MSc in physics from the University of Alberta, Edmonton, in 1973. In May 1985 he obtained a PhD in computer and information science from the University of Pennsylvania. He is a member of the Canadian Medical and Biological Engineering Society, the American Association of Physicists in Medicine, the ACM, and IEEE.



Jayaram K. Udupa received his PhD in computer science from the Indian Institute of Science, Bangalore, in 1976, with a medal for best research. After working at the institute for two years as a scientific officer, he joined the Medical Image Processing Group (MIPG), first at SUNY, Buffalo in 1978, and then at the Department of Radiology at the University of Pennsylvania in 1981, where he is currently the Director of MIPG.

Dr. Udupa has worked for the past 10 years in the areas of medical signal and image processing, computer graphics and pattern recognition, and has published over 40 articles in these areas. He is the main architect of the software packages DISPLAY82, 3D82, 3D83 and 3D98 produced by MIPG. He is a senior member of IEEE, and a member of ACM, and NCGA.

The authors' address is Hospital of the University of Pennsylvania, 3400 Spruce St./G1, Philadelphia, PA 19104.