COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 29, 361–376 (1985)

NOTE

Image Space Shading of 3-Dimensional Objects

DAN GORDON*

Department of Computer Studies, University of Haifa, Haifa 31999, Israel

AND

R. ANTHONY REYNOLDS†

Medical Image Processing Group, Department of Radiology, University of Pennsylvania, 3400 Spruce Street, Philadelphia, Pennsylvania 19104

Received December 13, 1983; revised July 2, 1984; accepted October 11, 1984

Two-dimensional images of 3D objects require realistic shading to create the illusion of depth. Traditional (object space) shading methods require extra data (normal vectors) to be stored with the object description. When object representations are obtained directly from measured data, these normal vectors may be expensive to compute; if the object is modified interactively, they must be recomputed frequently. To avoid these problems a simple shading method is devised which uses only information available in image space, after coordinates have been transformed, hidden surfaces removed, and a complete pre-image of all objects has been assembled. The method uses both the distance from the light source and the surface orientation as the basis for shading. The theory and its implementation are discussed and shaded images of a number of objects are presented. © 1985 Academic Press, Inc.

1. INTRODUCTION

Computer-generated displays of 3-dimensional (3D) objects fall into two classes [1]: those which produce a true 3D image in space, and those which project the object onto a 2-dimensional (2D) surface such as a TV monitor driven by a frame-buffer display system. Additionally, the object data may come from one of several sources, or be represented in different ways [2]: from a computer model synthesized with polygons or other 2D or 3D primitives, from a parametric representation of the object or its surface, or from a real medical object (human organ) examined experimentally with a device such as a computed tomography (CT) scanner. In the last case, the object data are frequently stored in the form of *voxels* (cubic or rectangular volume elements of equal size). We will refer to the coordinate system in which objects are defined as *object space*, and that within which they are displayed as *image space* (Fig. 1). Object space and image space are related by a coordinate transformation.

The realistic display of 3D objects on a 2D surface requires several depth-cues to create the illusion of the third dimension: among these are hidden surface removal, motion parallax, and shading. Shading techniques are used to simulate both the object surface characteristics, and the position and orientation of object surfaces

*Research performed while visiting the Medical Image Processing Group, University of Pennsylvania, July–August 1983. Currently visiting the Department of Computer Science, University of Cincinnati.

†Present address: Department of Computer and Information Science, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pa. 19104.

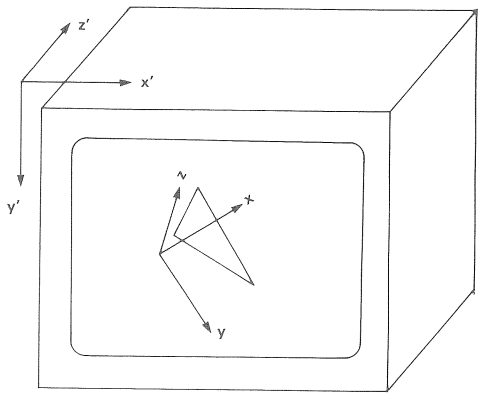


FIG. 1. Object space (x, y, z) and image space (x', y', z').

with respect to light sources. The simplest form of shading is by distance only, so that objects far from the light source appear darker. More sophisticated models distinguish between *diffuse reflection*, where the surface scatters light equally in all directions, and *specular reflection*, where a glossy surface exhibits a highlight which varies with the viewing direction.

Diffuse reflecting surfaces appear to have the same brightness from all viewing directions; the brightness depends upon the *illumination* which is given by

$$I = I_{\max} \cos\theta, \tag{1}$$

where θ is the angle between the incident light and the normal to the surface (Fig. 2a). $I_{\max}$ is the maximum illumination which results when $\theta = 0$. We can rewrite Eq. (1) as

$$I = I_{\max} \mathbf{N} \cdot \mathbf{L}, \tag{2}$$

where $\mathbf{N}$ is the unit vector normal to the surface and $\mathbf{L}$ is the unit vector along the line from the surface to the light source.

In theory, illumination falls off with the square of the distance from a point light source to the surface. In practice, however, a factor of $1/d^2$ does not produce natural looking images, and $1/d$ is used instead. This can be explained by considering natural lighting to have properties intermediate between those of a single point source and an extended light source which produces parallel rays. This point may be emphasized by adding an ambient light term to account for (diffuse) reflection of light from other surfaces in the environment. Additionally, the $\cos\theta$ term may create too harsh an angular dependence and may be replaced by $(\cos\theta)^p$ or $(\cos(\theta/2))^p$, where p is an empirically determined parameter. We have found the following expression to give good results in practice:

$$I = \frac{(I_{\max} - I_a)(D - d)(\cos\theta)^p}{D} + I_a, \tag{3}$$

where I_a is the ambient light, d is the distance to the surface from the light source,

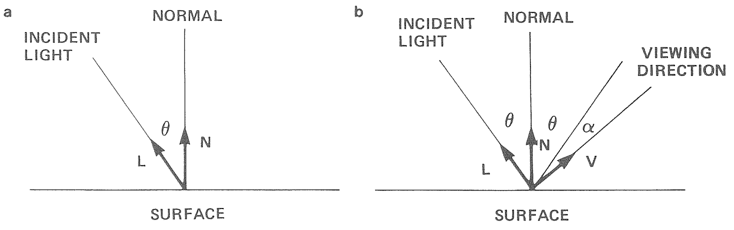


FIG. 2. (a) Diffuse Reflection. Brightness depends on direction of incident light **L**. (b) Specular Reflection. Brightness depends on the direction of incident light **L** and the direction of viewing **V**.

and D is the distance at which the illumination falls to zero (D may be considered to be the diameter of a sphere which encloses the object).

Specular reflection can be accounted for by modifying Eq. (3) to include a term $S(\theta, \alpha)$ which depends both on the direction of incident light and on the viewing direction (Fig. 2b). We have not used specular reflection in the images presented here; however, our shading technique could be modified to include this facility. More sophisticated shading models are discussed in [3 and 4]. Such state-of-the-art techniques result in greater realism than Eq. (3), but at the expense of considerably increased computation.

Once a shading model has been chosen, it may be implemented in one of the following ways, leading to two classes of algorithms:

(1) object space algorithms, where extra data are stored along with the voxel (or polygon) to provide information about neighboring voxels (or polygons);

(2) image space algorithms, which attempt to assign the shading using information available after a pre-image has been assembled.

In this paper we present an image space shading algorithm which can be applied to a pre-image such as a distance-only shaded image. The original representation of the 3D object, as well as the algorithm by which hidden surfaces are removed and the pre-image is produced, are immaterial; the only requirement is that the distance to the closest part of the object corresponding to each pixel of the image be available. On the other hand, much of the motivation for our work has been the desire to generate images of medical objects rapidly from data collected by an imaging device such as a CT scanner. In such an application, it is frequently desirable to modify the object interactively to obtain an unobstructed view or to simulate surgical procedures [5]; such a modification can be expensive when object space shading methods are employed. To demonstrate why this is so we give an account of previous shading methods before proceeding to a discussion of the new technique.

2. OBJECT SPACE SHADING METHODS

When surfaces are approximated by planar polygons, several shading methods are available. Most of these have to do with the manner in which the surface normal is estimated. *Distance-only shading* ignores the surface orientation and uses only the distance from the light source. *Constant shading* uses the true normal to the polygon to obtain an intensity which is then applied to all points on the polygon. *Gouraud* *shading* [6] attempts to create the impression of a curved surface by finding the resultant normal at each vertex where polygons meet, using this normal to assign an intensity to the vertex, and then interpolating this intensity across the polygon surface. Gouraud shading produces an intensity which varies continuously from polygon to polygon; however, the first derivative of the intensity is discontinuous and a Mach band effect [3] may be produced. *Phong shading* [7] avoids this by interpolating the normal vector itself to obtain an effective normal at each point on the polygon surface. The Mach band is still present but is less pronounced.

Voxels may be considered to be small cubes or rectangular parallelepipeds, with at most three faces visible, whose normals lie in one of 6 possible directions (along the positive or negative coordinate axes). Each face may be considered as a polygon to be shaded by one of the above techniques. We have experimented [8] with each of the above object space shading methods (except Gouraud shading), for objects composed of voxels. Distance-only shading gives a smooth appearance that is lacking in fine detail and curvature information. Constant shading gives more structural information and is easily implemented since no knowledge of neighboring faces is required; however, since the number of face directions is limited, we find that constant shading gives a displeasing, checkerboard appearance. Gouraud and Phong shading both require examining neighboring faces to establish vertex normals. While this one-time procedure may be absorbed into the cost of synthesizing a computer model with polygons, it is expensive in both computer time and storage for real objects composed of voxels where the procedure must be repeated each time a new object is examined. Moreover, a very large number of voxels is usually required to represent an object to an acceptable precision (e.g., the skull shown in Fig. 6 consists of 500,000 voxels; its surface is represented by 270,000 voxel faces. Further examples are given in Table 1). Finally, the normals must be recalculated frequently if the object is to be modified interactively.

Our experiments indicate that although Phong shading gives very satisfactory results, it is computationally expensive with objects composed of voxels. A less expensive method which gives similar image quality, *Contextual shading*, was introduced by Herman and Udupa [9] and later revised by Chen *et al.* [8]. In this method, the orientations of four neighboring faces are taken into account when the intensity of a given face is calculated. These orientations are stored with the voxel description in the form of "neighbor codes." Although contextual shading is an overall improvement over the other object space shading methods we have tried, it shares with them the disadvantage that the normals or neighbor codes must be recalculated if the

TABLE 1

Object	Number of voxels in object	Number of voxel faces in object surface	Minimum viewport enclosing the image
Digital sphere	1092727	76902	127 × 127
Skull	501957	272182	184 × 87
Ankle-bone	607663	142684	186 × 122

object is modified. This is a serious problem for the interactive real-time display systems which have recently been proposed [10].

3. AN IMAGE SPACE APPROACH TO THE SHADING PROBLEM

Object space methods (where normals are stored as part of the object description) have the disadvantage that the normals must be recalculated if the object is modified. For systems allowing rapid interaction with the object, this problem is of great importance.

Except for the simplest models, shading is by nature a nonlocal operation, requiring knowledge of neighboring voxels. One time at which this information is accessible is at image display time, after coordinates have been transformed, hidden surfaces removed, and visible surfaces have been rendered. An "image space" shading algorithm can be formulated as follows:

> Assume that at the end of the hidden surface removal process we have a Z-buffer containing, for each pixel, the distance from the light source to the closest point on the object which projects onto that pixel. The Z-buffer then contains a complete representation of the visible surface of the object of the form $z = z(x, y)$. We can compute the normal $\mathbf{N}$ at any point by finding the gradient vector ∇z at that point. Assuming the direction of light $\mathbf{L}$ is known and the other parameters can be estimated, the appropriate intensity can be computed by applying Eqs. (2) and (3).

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 (Fig. 3). These problems and their solutions are addressed in the remaining sections.

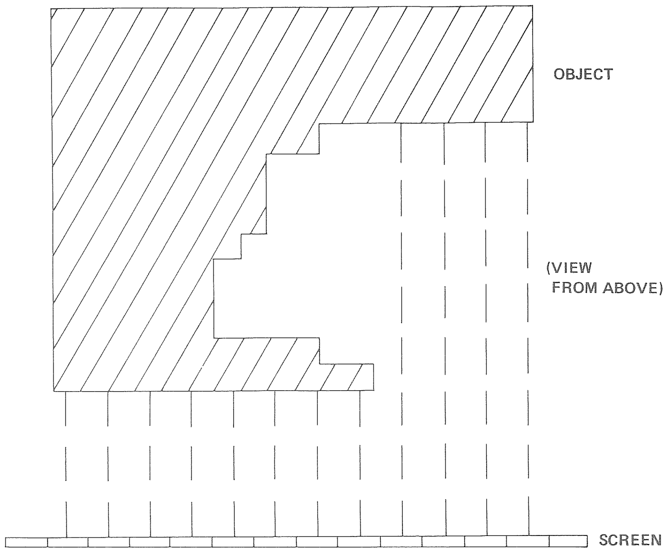


FIG. 3. Discontinuities in distance due to hidden surfaces.

Note that the distances for each pixel, from the light source to the closest point of the object are the only data needed as input to this algorithm. This does not imply, however, that a "Z-buffer algorithm" must be used for hidden surface removal: any algorithm may be used, provided the distances to the object surface are obtained. We have successfully used the Z-buffer algorithm as described in [2], various "back-to-front" voxel readout schemes [10, 11], and a ray-tracing algorithm [3, 4, 12] as the input to this shading method. In fact, the technique can be used with any program which produces distance-only shaded images (provided the intensities are directly related to the distances to the object surface), and could thus be made a hardware option of an intelligent image display system. The quality of the shaded image will, of course, depend on the precision to which the distances are known. Our experience has shown 16 bits of precision to be adequate, and 8 bits to be sufficient for many practical applications.

4. THE GRADIENT SHADING METHOD

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

$$\nabla z = \left(\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, 1 \right). \tag{4}$$

This necessitates obtaining the derivatives $\partial z/\partial x$, $\partial z/\partial y$ numerically. For an $N \times N$ Z-buffer, given the distances

$$z_{i,j} = z(x = i, y = j) \qquad (1 \leq i \leq N, 1 \leq j \leq N) \tag{5}$$

from the observer to the object at each pixel (i, j), $\partial z/\partial x$ may be approximated by the *backward difference*

$$\delta_b = z_{i,j} - z_{i-1,j} \qquad (2 \leq i \leq N, 1 \leq j \leq N), \tag{6}$$

by the *forward difference*

$$\delta_f = z_{i+1,j} - z_{i,j} \qquad (1 \leq i \leq N - 1, 1 \leq j \leq N), \tag{7}$$

or by the average of the two,

$$\delta_c = \tfrac{1}{2}(z_{i+1,j} - z_{i-1,j}) \qquad (2 \leq i \leq N - 1, 1 \leq j \leq N). \tag{8}$$

The last is termed the *central difference* and is, in general, a better approximation than either the backward or the forward difference alone. The forward or backward difference, as appropriate, may be used at edge pixels where the central difference does not apply.

The central difference would be a good approximation to the derivative $\partial z/\partial x$ if z were a continuous, smooth function of x. However, this is not necessarily the case. Suppose, for example, that $z_{i-1,j}$ and $z_{i,j}$ are distances to one object surface but $z_{i+1,j}$ is the distance to another, then the backward difference is a better approximation to the derivative $\partial z/\partial x$. (A similar argument, of course, applies to $\partial z/\partial y$. For simplicity we are considering the x-derivative only.) Fig. 3 is an example.

What is required is a method of distinguishing gradual changes along a single surface from abrupt changes from one surface to another. One possible method is to threshold the differences δ_b and δ_f: if both differences are less (in absolute value) than the threshold, use the central difference δ_c; otherwise, ignore the difference that is greater than the threshold. There are two problems with this approach. One is that a flat surface viewed at a very steep slope may project onto adjacent pixels with z-values differing by more than the threshold. The second problem is that adjacent pixels may differ by less than the threshold but may still be the projections of different surfaces.

The solution we have adopted is to take a weighted average of the forward and backward differences

$$\frac{\partial z}{\partial x} \simeq \frac{W_b\delta_b + W_f\delta_f}{W_b + W_f}, \tag{9}$$

where W_b and W_f are positive weights which depend on $|\delta_b|$ and $|\delta_f|$, respectively. (A similar expression is used for $\partial z/\partial y$.) Small differences are given large weights and vice versa. If $W_b = W_f$ then we obtain the central difference; this will be the case regardless of whether W_b and W_f are both large or both small. However, if δ_b is small and δ_f is large, then more weight is given to the backward difference, which is more likely to represent the true slope at that point. It follows that any portion of the surface which has constant slope and projects onto at least 3×3 pixels will be assigned the correct normal vector by this method. It is only in places of rapidly changing slope that an incorrect value for the normal may result.

We describe now some weighting functions which we have used successfully. Let Δ be the maximum possible difference between values of z (e.g., the maximum extent of the object). For $0 \leq \delta \leq \Delta$, we define a weighting function $W(\delta)$ having two parameters a, b $(0 \leq a \leq b \leq \Delta)$ as follows:

$$W(\delta) = \begin{cases} 1 & \text{if} \, \delta \leq a \\ \epsilon & \text{if} \, \delta \geq b \\ \dfrac{1+\epsilon}{2} + \dfrac{1-\epsilon}{2}\cos\left(\dfrac{\delta - a}{b - a}\pi\right) & \text{otherwise.} \end{cases} \tag{10}$$

The weighting function is used to define the two weights $W_b = W(|\delta_b|)$ and $W_f = W(|\delta_f|)$, from which $\partial z/\partial x$ is obtained using (9); $\partial z/\partial y$ is obtained similarly. ϵ is a small number large enough so that division by 2ϵ maintains sufficient precision. In our implementation we used $\epsilon = 10^{-5}$.

The parameters a and b control the points at which distances change their weights: a is the maximal distance at which the large weight ($W = 1$) is given, and b is the minimal distance at which the small weight ($W = \epsilon$) is given. For distances between a and b the weight varies continuously and smoothly due to the cosine function. This minimizes discontinuities in the image when the object orientation is changed slightly. For the objects that we have examined, we found that $a = 2$, $b = 5$ produce good images. Since there are only a small number of possible values for δ, a table lookup can be used to compute $W(\delta)$ efficiently.

Other weighting functions we have tried include a threshold function (Eq. (10) with $a = b$), a linear function

$$W(\delta) = 1 - \delta/b \tag{11}$$

and the reciprocal function

$$W(\delta) = \frac{1}{W_0 + \delta}, \tag{12}$$

where W_0 is an additional parameter. However, none of these has been found to be superior to the weighting function given in (10).

An explanation for the choice of the cosine function and the parameters $a = 2$, $b = 5$ can be given as follows: Suppose our unit of distance in image space is the distance between two adjacent pixels. Then the smallest differences encountered are in the range $0 \leq \delta \leq 2$, corresponding to an angle θ in the range $0 \leq \theta \leq 63°$. In order to obtain an accurate estimate of the gradient of such surfaces, it is important to give these small differences equal weight. A difference of 5 or more on the other hand means an angle $\theta \geq 78°$, so no harm results in giving such differences equal weight—they all represent very steep slopes. The cosine function ensures a continuous and smooth transition from $W = 1$ to $W = \epsilon$.

A difficult problem associated with the display of objects represented by voxels—such as medical objects produced by CT scans—is that there are two conflicting objectives: one is the requirement of displaying the object with as much detail as possible, preferably by some form of shading that will accentuate the curvature of the surface. The second objective is to eliminate or deemphasize artifacts due to the discretization of the object. When slopes are emphasized strongly, artifact patterns stand out too sharply. By smoothing the image one could reduce the artifact patterns, but at the danger of smoothing away important details. Our approach has been to use the parameter p in Eq. (3) to balance the curvature against the artifact patterns: for most objects we have examined, $p = 0.2$ has been found to be appropriate. In a hardware implementation, parameter p could be left as a "knob" similar to brightness and contrast, to be fine-tuned to suit individual requirements. The effect of varying p is shown in Fig. 5.

One other problem results from a straightforward implementation of the above ideas. Consider three distant relatively flat surfaces such that the middle surface projects onto exactly one pixel (or line of pixels) between one pixel from the near surface and one pixel from the far surface (Fig. 4). The method will assume that such a combination represents a very steep slope and will shade the middle pixel very dark. This results in very dark patches or stripes one pixel wide. Some of the medical objects that were tested were of sufficient complexity for this to occur.

A simple solution is to check for such dark pixels using a threshold function, and to replace such dark pixels by the average of their neighbors. Note however, that the object may contain true "holes" which may also appear after shading as very dark pixels. It is therefore necessary to check the pixel values for true holes before shading, and ensure that true holes are left unchanged.

Theoretically, the above situation can occur with more than three surfaces, and our method really has no solution for that problem. In practice, however, this would

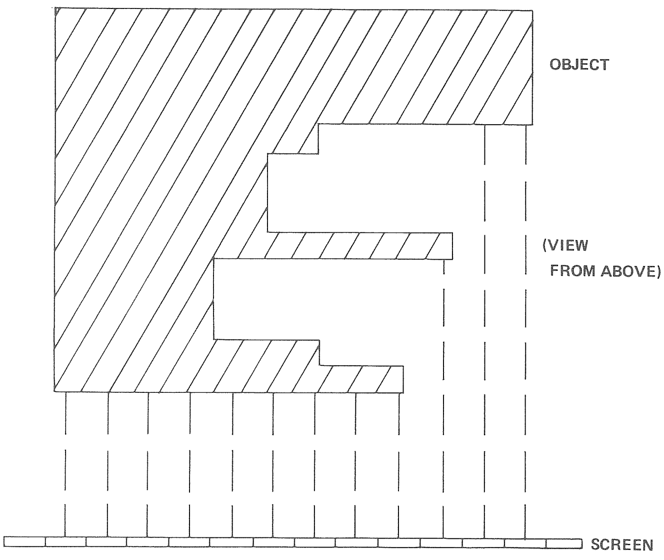


FIG. 4. Slope artifact caused by hidden surfaces.

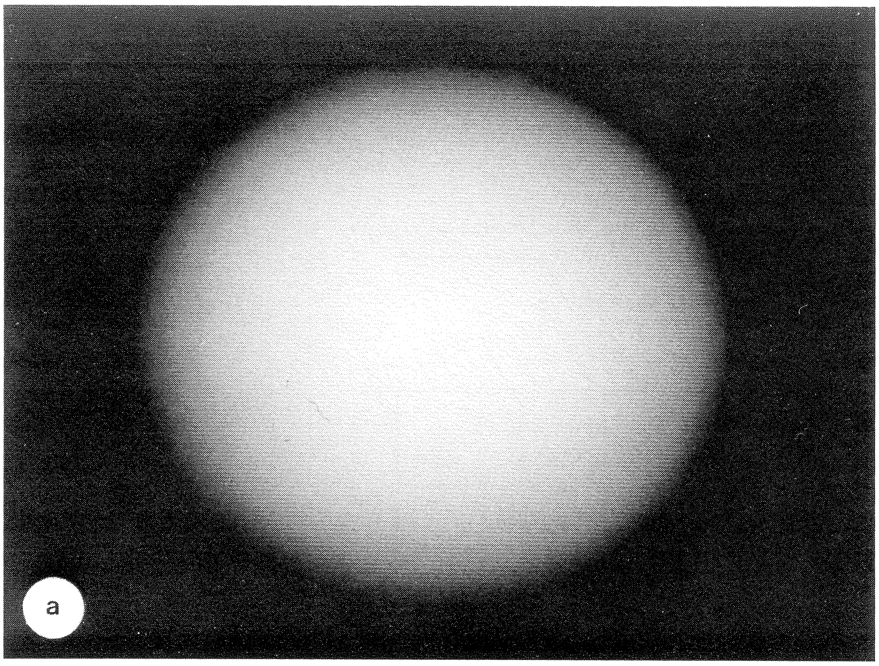
probably occur only very rarely, and would simply result in what appears to be a very steep slope rather than a discontinuous edge.

5. EXPERIMENTAL RESULTS

A number of objects were displayed using (a) distance-only shading and (b) gradient shading, with different values of a, b, and p. These included both artificial objects (Fig. 5) and real objects obtained from CT scans of live patients (Figs. 6–8). From our experiments we concluded that $a = 2$, $b = 5$, and $p = 0.2$ give good results, and these are the values used with the gradient shaded images presented here. We include one image (Fig. 5c) with $p = 1.0$ for comparison. All the images were produced using the back-to-front voxel readout method of hidden surface removal [10, 11] and were rendered on a frame-buffer display with a resolution of 256×256 pixels and a gray-scale of 256 levels. In Eq. (3) we used $I_{\max} = 255$ and $I_a = 30$. No anti-aliasing or smoothing was used. For each object the distance-only image (a) was the pre-image from which the gradient shaded images (b) and (c) were derived.

Figure 5 shows three images of a computer generated digital sphere with a diameter of 127 voxels. Figure 5a is a distance-only shaded image ($p = 0$), Fig. 5b is gradient shaded with $p = 0.2$, and Fig. 5c is gradient shaded with $p = 1.0$. On the one hand, there is a definite improvement in conveying depth and curvature in Fig. 5c as compared to Fig. 5a; on the other hand, the artifact patterns due to the discrete voxel representation are more pronounced. We feel that Fig. 5b, with $p = 0.2$, strikes the appropriate balance between the conflicting objective of representing the underlying object (the sphere) and the details of its structure (the voxel representation).

Figure 6 shows a skull of a live patient, obtained from CT data. These images show the improvement in detail and perception of depth that can be obtained by



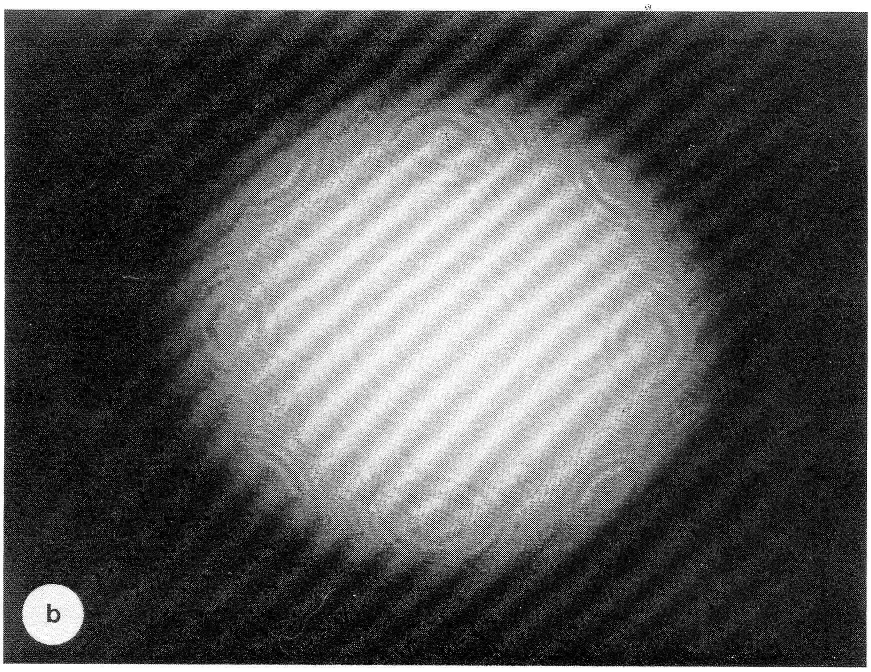


FIG. 5. (a) Digital sphere with distance-only shading. (b) Digital sphere with gradient shading, $p = 0.2$. (c) Digital sphere with gradient shading, $p = 1.0$.

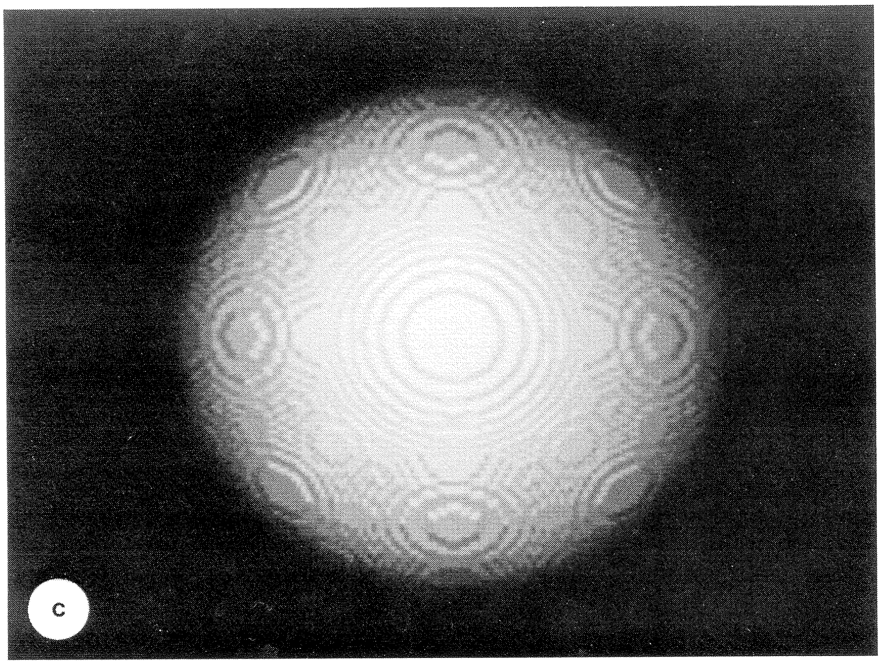


FIG. 5—*Continued.*

using gradient shading as a post processing technique on distance-only shaded images.

Figures 7 and 8 show two views of the same object, an ankle-joint of a live patient. In Fig. 7, as in Figs. 5 and 6, there is no object rotation; that is, the voxels by which the object is represented are arranged in planes orthogonal to the direction of viewing. In Fig. 8, the object has been rotated by 80° about a horizontal axis. We include this image to demonstrate that good results can be obtained with gradient shading irrespective of the object orientation.

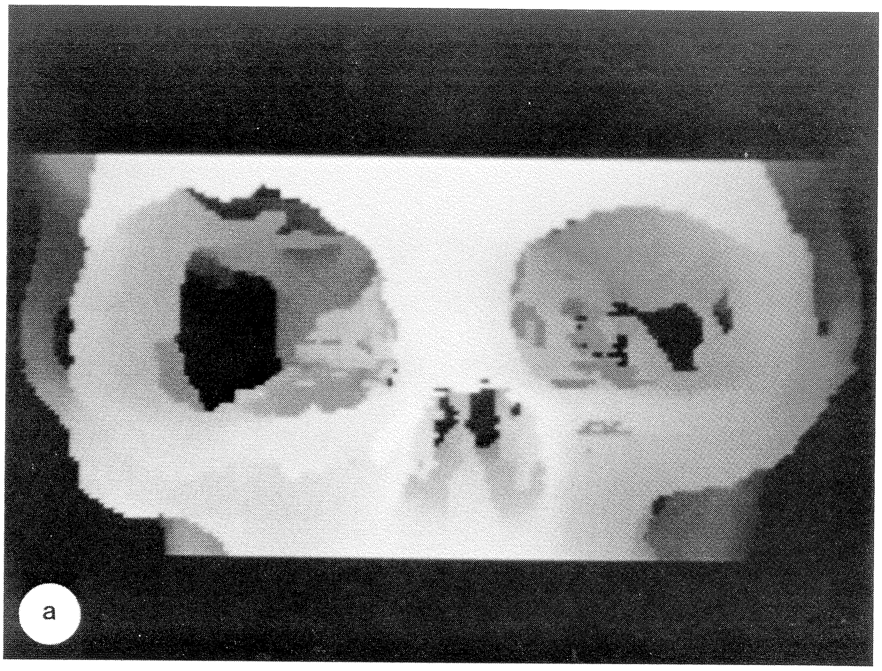
6. DISCUSSION

We have presented a simple shading method which requires only information available in image space. The method uses both the distance from the light source and the orientation of the object to calculate the intensity assigned to each pixel.

There are three grounds on which our technique can be compared to existing methods: applicability, image quality, and computational effort.

We turn to applicability first, because gradient shading can be used in some situations where other shading methods simply do not apply. For example, gradient shading can be used with existing 3D display algorithms which produce distance-only shaded images [11–13]. Such algorithms are attractive because of their simplicity and speed, but the images are lacking in detailed structural information. The gradient shading algorithm could be microcoded into a sophisticated frame-buffer display system which could process any distance-only shaded image, without any knowledge of vertex normals or how the image was produced.

As regards image quality, our images are certainly not as good as state-of-the-art object space methods applied to, for example, ray tracing [4]. However, in a direct



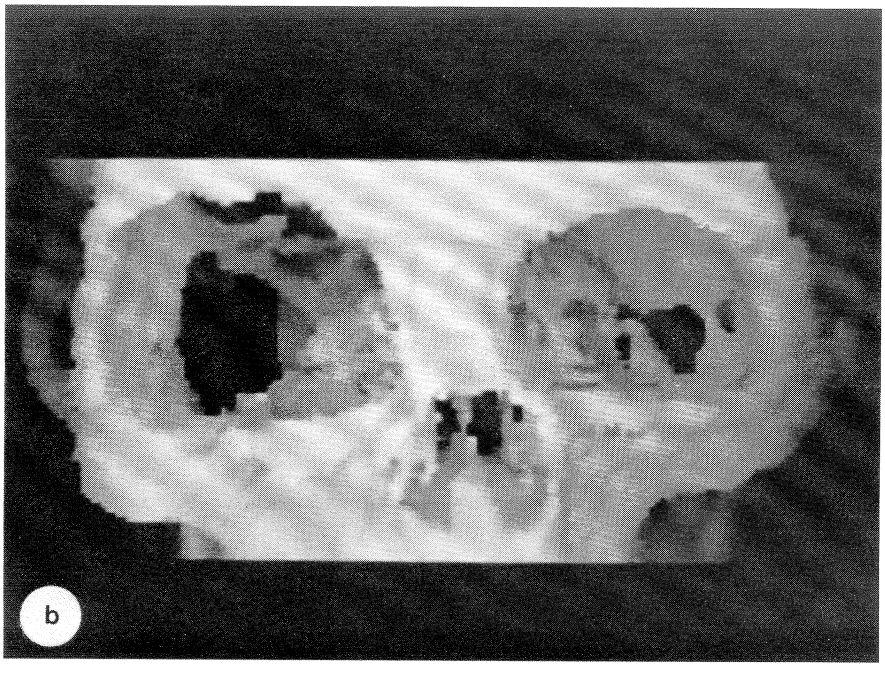


FIG. 6. (a) Skull with distance-only shading. (b) Skull with gradient shading.

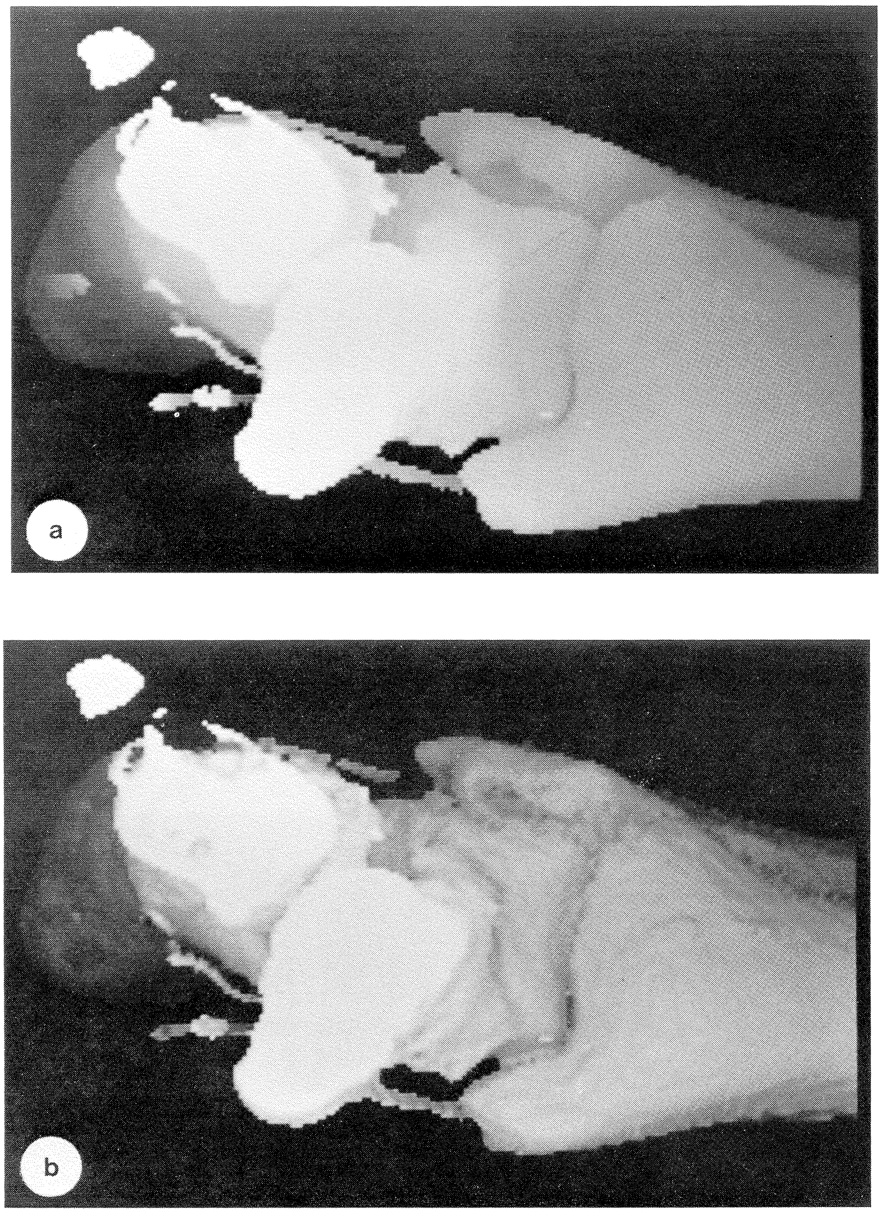
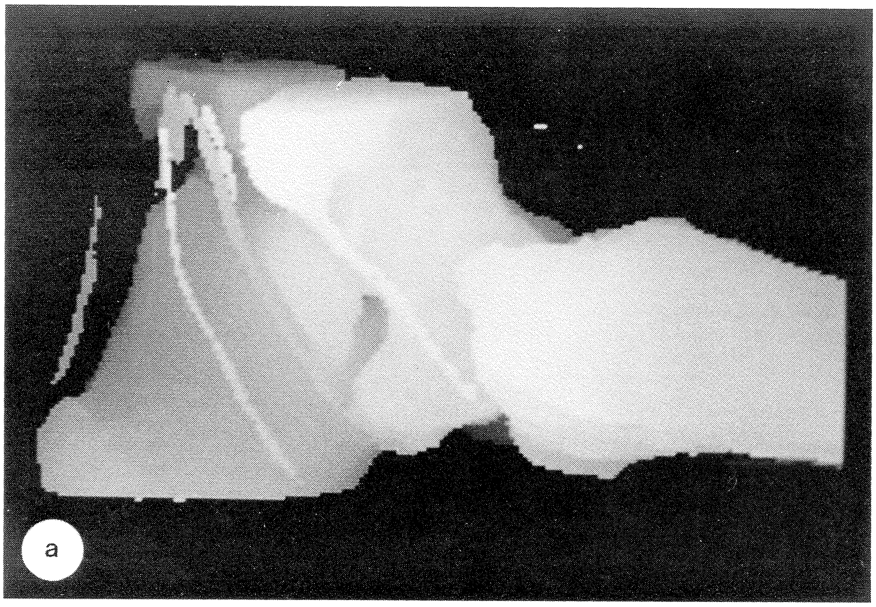


FIG. 7. (a) Ankle-bone with distance-only shading, no rotation. (b) Ankle-bone with gradient shading, no rotation.

comparison [8] we have judged gradient shaded images to be only slightly inferior to Phong shaded images of the same objects using the same 3D display algorithm.

As to the computational effort, one must distinguish between different viewing requirements.

In many computer graphics applications, an unchanging object is viewed many times from different directions. In such a case, the one-time cost of computing object



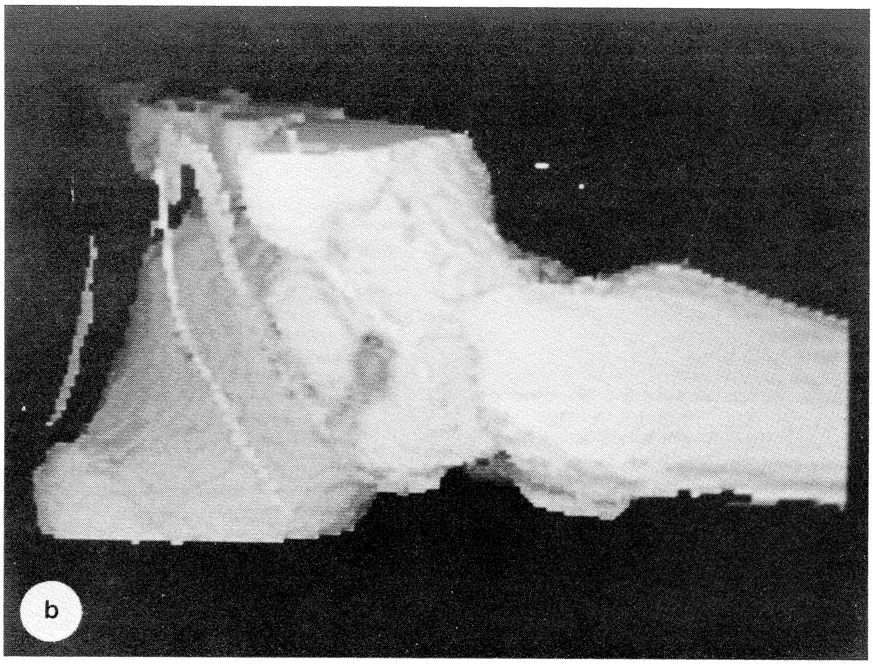


FIG. 8. (a) Ankle-bone with distance-only shading, rotated 80° about horizontal axis. (b) Ankle-bone with gradient shading, rotated 80° about horizontal axis.

normals may be neglected, and object space shading methods are certainly appropriate. In other applications (e.g., medical applications), the object may change frequently, perhaps because a new patient is imaged, or because the user has the facility to modify an existing patient database interactively. In these cases, the cost of computing new normals must be taken into account. In object space, this effort is typically proportional to the number of polygons (or voxel faces) required to represent the surface of the object. The work done by gradient shading, on the other hand, is independent of the object and depends only on the number of nonzero pixels in the image. As objects become more complex, the number of polygons will eventually become greater than the number of pixels (figures for the objects used to illustrate this article are presented in Table 1). Thus image space shading is bounded by the number of pixels in the display frame-buffer and, in principle, a gradient shading processor can be constructed which will shade any image presented on such a device in constant time, regardless of the object complexity.

Going a step further, by noting the minimum and maximum x and y pixel coordinates of the image, one can avoid processing the entire screen. The work done in shading would then be proportional to the area of the smallest viewport containing the image. The following theoretical observation about the time complexity of the different shading methods can then be stated: due to their dependence on object surface area, object space algorithms—however efficiently they are implemented—will take longer than image space methods for objects whose surface is represented by a sufficiently large number of polygons.

In conclusion, we find the advantages of our method to be the following: First, it does not require any normal vector data to be stored with or computed from the object representation, and therefore is appropriate for data obtained experimentally or for interactive systems where the object is modified frequently. Second, it can be used with any object representation or appended to any visible surface rendering algorithm which produces the distance to the object associated with each pixel of the image. Third, it can be integrated into the hardware of sophisticated display units to perform gradient shading directly on distance-only shaded images in a time independent of the object complexity. The disadvantage is slightly inferior image quality, when compared with sophisticated object space shading methods.

ACKNOWLEDGMENTS

Thanks are due to Ms. Mary A. Blue for typing the manuscript.

Part of this work was supported by NIH grant HL28438. We wish to thank L. S. Chen, G. T. Herman, and J. K. Udupa for many stimulating discussions on shading models, and Gideon Frieder for his help in implementing the back-to-front voxel readout algorithm. We also wish to thank Sushma Trivedi and Gerald Waxler for their help in constructing the digital sphere used in Fig. 5, G. T. Herman for the patient data used in Fig. 6, and David Roberts and J. K. Udupa for the patient data used in Figs. 7 and 8.

REFERENCES

1. R. A. Reynolds, *Some Architectures for Real-time Display of Three-Dimensional Objects: A Comparative Survey*, Technical Report MIPG84, Department of Radiology, University of Pennsylvania, October 1983.
2. 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. Soc. Photo-Opt. Instrum. Eng.* **367**, 1982, 3–14.

3. J. D. Foley, and A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison–Wesley, Reading, Mass. 1982.
4. R. A. Hall, and D. P. Greenberg, A testbed for realistic image synthesis, *IEEE Comp. Graphics Appl.* **CGA-3**, No. 8, 1983, 10–20.
5. L. J. Brewster, S. S. Trivedi, H. K. Tuy, and J. K. Udupa, Interactive surgical planning, *IEEE Comput. Graphics Appl.* **CGA-4**, No. 3, 1984, 31–40.
6. H. Gouraud, Continuous shading of curved surfaces, *IEEE Trans. Comput.* **C-20**, 1971, 623–628.
7. B-T. Phong, Illumination for computer generated pictures, *Comm. ACM* **18**, No. 16, 1975, 311–317.
8. L. S. Chen, G. T. Herman, R. A. Reynolds, and J. K. Udupa, *Surface Rendering in the Cuberille Environment*, Technical report MIPG87, Department of Radiology, University of Pennsylvania, January 1984.
9. G. T. Herman, and J. K. Udupa, Display of three-dimensional discrete surfaces, *Proc. Soc. Photo-Opt. Instrum. Eng.* **283**, 1981, 90–97.
10. S. M. Goldwasser, and R. A. Reynolds, An architecture for the real-time display and manipulation of three-dimensional objects, in *Proc. 1983 International Conference on Parallel Processing*, Bellaire, Michigan, August 1983, pp. 269–274.
11. G. Frieder, D. Gordon, and R. A. Reynolds, Back-to-Front display of voxel-based objects, *IEEE Comput. Graphics Appl.* **CGA-5**, No. 1, 1985, 52–60.
12. H. K. Tuy, and L. T. Tuy, Direct 2-D display of 3-D objects, *IEEE Comput. Graphics Appl.* **CGA-4**, No. 10, 1984, 29–33.
13. M. W. Vannier, J. L. Marsh, and J. O. Warren, Three-dimensional computer graphics for craniofacial surgical planning, *Comput. Graphics* **17**, No. 3, 1983, 263–273.