

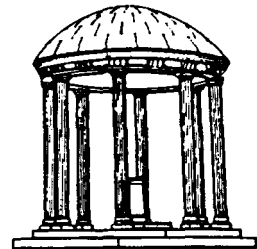
Rendering Mixtures of
Geometric and Volumetric Data

TR88-052

December 1988

Marc Levoy

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175



UNC is an Equal Opportunity/Affirmative Action Institution.

Rendering Mixtures of Geometric and Volumetric Data

Marc Levoy

December, 1988

Computer Science Department
University of North Carolina
Chapel Hill, NC 27599

Abstract

Many scientific problems require sampled functions and analytically defined objects to appear together in a single visualization. *Volume rendering* is a relatively new technique for visualizing sampled scalar functions of three spatial dimensions by computing 2D projections of a colored semi-transparent gel. This paper addresses the problem of extending volume rendering to handle polygonal objects. Two solutions are presented. The first employs a hybrid ray tracer capable of handling both geometric and volumetric data. The second consists of 3D scan-converting the geometric primitives into the volumetric dataset and rendering the resulting ensemble. Both solutions are designed to be efficient and to avoid aliasing artifacts. Their relative costs, image quality, and versatility are compared using examples from two application areas: medical imaging and molecular graphics. Techniques for adding shadows and texture are also presented.

1. Introduction

As visualization techniques move from computer graphics laboratories into scientific disciplines, they are judged not by standards of photo-realism, but by their ability to produce scientific insight. For some problems, presentation of a single type of data suffices. For others, additional insight is gained by combining more than one type of data in a single visualization. This paper addresses the problem of rendering mixtures of analytically defined objects and sampled scalar functions of three spatial dimensions.

Most previous efforts in this area have employed polygonal meshes or binary voxel representations. Pizer et al. [1986] track edges on each slice of a computed tomography (CT) dataset to yield a set of contours, tile between contours on adjacent slices, supplement the resulting polygonal mesh with analytically defined objects, and render the ensemble using conventional hidden-surface algorithms. Lørsensen and Cline [1987] apply surface detectors at each sample location to produce a large collection of voxel-sized polygons which can be supplemented with analytically defined objects and rendered using conventional algorithms. Kaufman (with others) threshold volumetric data to produce a binary voxel representation, use 3D scan-conversion to add polygons [1987b], polyhedra [1986], and cubic parametric curves, surfaces, and volumes [1987a] to the array, and render the resulting ensemble using custom-designed hardware [1988a].

All of these approaches require a binary classification of the volumetric data. As a result, they suffer from false positives (spurious objects) and false negatives (missing objects), particularly in the presence of small or poorly defined features. Kaufman's use of a binary voxel representation for analytically defined objects gives rise to additional artifacts, although a solution to this problem has apparently been worked out [1988b].

The approach taken in this paper is based on a visualization technique called *volume rendering*† in which a color and an opacity is assigned to each voxel, and a 2D projection of the resulting colored semi-transparent gel is computed [Levoy, 1988a, Drebin et al., 1988, Sabella, 1988, Upson and Keeler, 1988]. The principal advantages of volume rendering over other visualization techniques are its superior image quality and its ability to generate images without explicitly defining surfaces.

Although many researchers have suggested methods for extending volume rendering to handle analytically defined objects, no implementations have yet appeared in the literature. Sun Microsystems has reportedly extended a conventional ray tracer to handle volumetric data [Mosher, 1988], but their algorithm has not been published. In a paper currently in review, Goodsell et al. [1988] describe a two-pass approach that combines a Z-buffer algorithm for rendering polygonally defined atomic structure and a ray tracer for rendering volumetric molecular data. Although their method produces satisfactory visualizations in many cases, it cannot efficiently handle semi-transparent polygons. Furthermore, the division of labor into two passes necessitates rendering all polygons even though they might be obscured by volumetric data.

This paper presents two methods for extending the volume rendering algorithm described by Levoy [1988a] to handle polygonally defined objects. The first method employs a hybrid ray tracer. Rays are simultaneously cast through the volumetric sample array and the polygonal database, samples of each are drawn at equally spaced intervals along the rays, and the resulting colors and opacities are composited together in depth-sorted order. To avoid errors in visibility, volumetric samples lying immediately in front of and behind polygons are given special treatment. To avoid aliasing of polygonal edges, adaptive supersampling is used. The second method employs 3D scan-conversion with analytic anti-aliasing. Polygons are shaded, filtered, sampled, and combined with the volumetric data. The resulting composite dataset can then be rendered using published methods. If the polygonal data is sufficiently bandlimited prior to sampling, this method also produces images free from aliasing artifacts.

To compare the relative versatility of these two methods, techniques for adding shadows and textures will also be considered. Nelson Max has written a brief but excellent survey of algorithms for casting shadows [1986]. The present study employs a two-pass approach [Williams, 1978], but stores shadow information in a 3D light strength buffer instead of a 2D shadow depth buffer. The amount of memory required for a 3D buffer is obviously much greater, but the representation has several advantages. By computing a fractional light strength at every point in space, penumbras and shadows cast by semi-transparent objects are correctly rendered. Moreover, the shadow aliasing problem encountered by Williams does not occur. Finally, the present algorithm correctly handles shadows cast by volumetrically defined objects on themselves, as well as shadows cast by polygons on volumetric objects and vice versa.

There are several kinds of texture mapping that might be useful when rendering mixtures of geometric and volumetric data. Wrapping textures around volumetrically defined objects requires knowing where their defining surfaces lie - a hard problem. Projecting textures through space and onto these surfaces is much easier and can be handled by a straightforward extension of the shadow casting algorithm. Mapping textures onto polygons embedded in volumetric datasets is also relatively simple. The two latter techniques will be demonstrated in this paper. A good survey of texture mapping algorithms has been written by Heckbert [1986]. The present study employs an approach similar to that described by Feibush, Levoy, and Cook [1980], but since geometric, volumetric, and texture data each exhibit different spectra, care must be taken when mixing them.

† In the field of image processing, this term has been used to denote techniques based on overpainting of opaque voxels. In computer graphics, it has come to denote techniques based on blending of semi-transparent voxels. To distinguish between these approaches, Reynolds [1988] has suggested that volume rendering be used to encompass both of them and the more specific *volumetric compositing* be used to denote the partial transparency variant. To avoid confusion, I will continue to use current computer graphics terminology until a new one gains acceptance in the research community.

2. Hybrid ray tracer

The first rendering method we will consider is the hybrid ray tracer shown in figure 1. Let us assume an $N \times N \times N$ voxel array of scalar values and a set of embedded polygons. We begin by shading and classifying the volumetric data to yield a color c_V and opacity α_V for each voxel [Levoy, 1988a]. Parallel viewing rays are then traced into the data from an observer position as shown in figure 2. For each ray, a vector of colors c_V and opacities α_V is computed by resampling the volumetric data at equally spaced positions along the ray and tri-linearly interpolating from the colors and opacities in the eight voxels surrounding each sample location. Independently, all intersections between the ray and polygons in the environment are computed and shaded, yielding a color c_P and opacity α_P for each point of intersection. Finally, the resampled volumetric colors and opacities are composited with each other and with the polygonal colors and opacities in depth-sorted order to yield a color for the ray.

Volumetric compositing correctly renders the appearance of a gel composed of many small slabs each of identical size and homogeneous color and opacity as shown in figure 3a. The thickness of each slab is equal to the spacing between samples along a viewing ray, and the width of the slab is equal to the spacing between adjacent rays. If both spacings are set roughly equal to the spacing between voxels in the volumetric data, and tri-linear interpolation is used to compute the color and opacity of each slab as described in the previous paragraph, the resulting image will generally be free of aliasing artifacts.

When a polygon is embedded in the volumetric data as shown in figure 3b, it passes through some of these slabs, obscuring some portion of the gel in each slab and being obscured by the remainder. An exact solution of the hidden-volume problem inside every slab would be extraordinarily expensive. Supersampling is an alternative, but also expensive. A solution of less accuracy and expense that has proved satisfactory in practice is to treat a polygon locally as a plane perpendicular to the ray and placed at the point of intersection as shown in figure 3c. An exact solution of the hidden-volume problem for this restricted case is simple. The law of exponential attenuation [Johns, 1983] states that the opacity α_V of a slab of homogeneous material is exponentially related to its density d_V and thickness t_V by

$$\alpha_V = 1 - e^{-d_V t_V} \quad (1)$$

where $0 < \alpha_V < 1$, and $\alpha_V = 1$ signifies complete attenuation. Let the thicknesses of those portions of the slab lying immediately in front of and behind an embedded plane perpendicular to the viewing ray be denoted t_F and t_B respectively. Solving equation (1) for opacities α_F and α_B in terms of opacity α_V and thicknesses t_F and t_B gives

$$\alpha_F = 1 - (1 - \alpha_V)^{t_F/t_V} \quad (2a)$$

and

$$\alpha_B = 1 - (1 - \alpha_V)^{t_B/t_V}. \quad (2b)$$

Working from front to back, we first composite c_V and α_F into the ray, followed by c_P and α_P , and finally by c_V and α_B . If the ray intersects more than one polygon within a slab, the contribution made by each polygon and each sliver of volumetric gel must be computed and composited separately.

At polygonal edges, polygon-polygon intersections, and polygon shading highlights, such an approximation does not suffice to prevent aliasing. Therefore, the number of rays cast per pixel is increased in these regions using adaptive supersampling [Whitted, 1980]. One difficulty with this approach is distinguishing whether the color difference observed in a sample region is due to volumetric data or geometric data, since supersampling is only appropriate in the latter instance. The problem is solved by computing two additional colors during ray tracing as shown in figure 1. The first, denoted C_G , contains contributions only from polygons (obtained by setting $\alpha_V = 0$ for all

volumetric samples). The second, denoted C_A , contains contributions only from polygons, but attenuated by passage through the volumetric data (obtained by setting $c_V = 0$ for all volumetric samples). If the range of C_A within a sample region exceeds some ϵ but the range of C_G does not, the observed color difference is due to spatially varying volumetric data rather than a geometric event (such as a polygonal edge). Supersampling is not required in this case. If the range of C_G exceeds ϵ but the range of C_A does not, the region contains a geometric event, but that event is hidden from view by overlying opaque volumetric data. No supersampling is needed in this case either. Only if the ranges of both colors exceed ϵ , signifying that the region contains a visible geometric event, should more rays be cast.

3. 3D scan-conversion

The second rendering method we will consider is 3D scan-conversion of the geometric data as shown in figure 4. We begin by shading and classifying the volumetric data to yield a color c_V and opacity α_V for each voxel. Independently, each polygon is shaded, filtered, and sampled at the resolution of the volumetric data to yield a color c_G and opacity α_G for each voxel. The polygons used in the present studies are of homogeneous color and are filtered using a 3D version of the edge filtering algorithm described by Feibush et al. [1980]. The convolution kernel employed is a constant weight ball 2 voxels in diameter. Methods that are faster and allow continuous shading are currently under investigation.

These two sets of colors and opacities are combined using volume matting operators [Drebin et al., 1988] to yield a composite color c_C and opacity α_C for each voxel. Since matting is not commutative, the order in which polygons are scan-converted affects the appearance of any polygon-polygon intersections. Polygons may also be matted over or under volumetric data, which also affects their appearance. The composite dataset is then ray traced, resampled, and composited without giving further consideration to the geometric data it contains. Since the polygons have already been filtered, an image sampling rate of one ray per pixel usually suffices to prevent aliasing artifacts, even along polygonal edges.

4. Shadow calculations

Although both of the rendering methods described above could be extended to yield shadow information, the pre-filtered representation of polygons employed by the scan-conversion method makes it the more convenient starting point. Figure 5 summarizes the modified algorithm. The first steps are shading and classification of the volumetric data, shading and scan-conversion of the polygons, and matting to yield for each voxel a set of composite colors c_1, \dots, c_L for light sources $i \in 1, \dots, L$ and a composite opacity α_C . Parallel illumination rays are then cast into the data from each light source as shown in figure 2. For each ray, we assume some initial light strength, draw samples of the volumetric data at equally spaced positions along the ray, compute an opacity at each location by tri-linearly interpolating from the nearest eight voxels, and attenuate the strength of the ray in proportion to the computed opacities. The set of strengths $\beta_1', \dots, \beta_L'$ computed at each sample position along each ray are stored in a 3D light strength buffer.

Using volumetric compositing, attenuation of light strength along an illumination ray is similar to accumulation of opacity along a viewing ray. The strength β_{out} of a ray as it leaves a sample location can be computed from the strength β_{in} as it approaches the sample and the opacity α of the sample by the formula

$$\beta_{out} = \beta_{in} \frac{(1 - \alpha)}{k_1 + k_2 d} \quad (3)$$

where k_1 and k_2 are constants used in a linear approximation of square-law light attenuation and d

is the distance from the light source to the sample location.

When all color and light strength volumes have been computed, a total color c_T for each voxel is computed as the weighted sum

$$c_T = \sum_{i=1}^L c_i \beta_i \quad (4)$$

where the β_i 's are obtained from the β_r 's by mapping from voxel indices into the coordinate system of light source i and tri-linearly interpolating from the nearest eight light strength values. The arrays of total color c_T and composite opacity α_C are then ray traced, resampled, and composited as usual.

One problem that arises when casting shadows through volumetric data is the tendency of surfaces in the data, which generally have a thickness of several voxels, to partially shadow themselves. This holds true for naturally occurring surfaces (such as tissue boundaries) as well as those introduced using 3D scan-conversion. The solution adopted in this paper is to translate the 3-D light strength volume a few voxels away from the light source before computing the c_T 's. While this expedient necessarily reduces the accuracy of shadowing within small objects, it avoids introducing distracting aliasing artifacts.

5. Texture mapping

The pre-filtered representation of polygons that made the scan-conversion method well suited to shadow casting would severely blur any texture applied to them. For this reason, texture mapping is better handled by the hybrid ray tracer. As part of the shading calculations performed at each point of intersection between a ray and a polygon, a mapping is performed from object space to texture definition space as shown in figure 2. The texture array is then filtered and resampled at that location. In the present studies, the slow but accurate filtering algorithm described by Feibush et al. [1980] is employed.

To prevent the addition of textures from triggering the casting of excessive number of rays, the adaptive supersampling procedure described in the last paragraph of section 2 must be modified. Specifically, those aspects of the polygon shading model that are analytic and might give rise to sudden changes in polygon color c_p or opacity α_p should be included in the supersampling decision, whereas contributions by sampled functions, which are assumed bandlimited, should not. In the current implementation, this rule is satisfied by including directional shading and depth cueing but not texture mapping in the computation of colors C_G and C_A .

6. Optimizations

The principal drawback of volume rendering techniques is their cost. Since all voxels participate in the generation of each image, rendering time grows linearly with the size of the dataset. Fortunately, these costs can be reduced by taking advantage of various forms of coherence. Three such optimizations are summarized here and described in detail elsewhere [Levoy, 1988b, Levoy, 1988c].

The first optimization is based on the observation that many datasets contain coherent regions of uninteresting voxels. A voxel is defined as uninteresting if its opacity is zero. Such coherence is readily encoded as a hierarchical spatial enumeration represented by a pyramid of binary volumes [Levoy, 1988b]. This pyramid is used to efficiently compute intersections between viewing rays and regions of interest in the data. In the hybrid ray tracer described in section 2, the pyramid can be used to speed the tracing of rays through segments of volumetric data lying between successive polygon intersections. In the scan-conversion method described in section 3,

the composite dataset including the sampled polygons can be used during construction of the pyramid. This improves the efficiency of rendering both volumetric and geometric data.

The second optimization is based on the observation that once a ray has struck an opaque object or has progressed a sufficient distance through a semi-transparent object, opacity accumulates to a level where the color of the ray stabilizes and ray tracing can be stopped [Levoy, 1988b]. In the hybrid ray tracer, adaptive termination of ray tracing reduces both the number voxels that must be resampled and the number of ray-polygon intersections that must be shaded. Note that this optimization must be applied independently in the computation of each of C_G and C_A so as not to adversely affect the adaptive supersampling process.

If there is coherence present in a dataset, there may also be coherence present in its projections. This is particularly true for data acquired from sensing devices, where the acquisition process often introduces considerable blurring. The third optimization takes advantage of this coherence by casting a sparse grid of rays, less than one per pixel, and adaptively increasing the number of rays in regions of high image complexity [Levoy, 1988c]. Images may be formed from the resulting non-uniform array of sample colors by interpolation and resampling at the display resolution. This optimization is well suited to the scan-conversion method in which polygons are filtered prior to ray tracing. It may also be integrated with the adaptive supersampling employed in the hybrid ray tracer, but one runs the risk of missing small polygons if the initial sampling rate is too low.

The cost of ray tracing volumetric data using a hierarchical enumeration grows linearly with the size of the image, logarithmically with the length of each ray, and linearly with the depth complexity of the data [Levoy, 1988b]. Adaptive termination of ray tracing reduces the dependence on depth complexity, and for the special case of an environment consisting only of opaque objects, eliminates this dependence entirely. In this case, the cost of generating an image grows nearly linearly with the size of the image. Adaptive sampling further reduces this cost, and if used carefully, does so with a negligible loss of image quality [Levoy, 1988c].

Combining these three optimizations, savings of more than two orders of magnitude over brute-force rendering methods have been obtained for many datasets. Alternatively, the adaptive sampling method allows a sequence of successively more refined images to be generated at equally spaced intervals of time by casting more rays, adding the resulting colors to the sample array, and repeating the interpolation and resampling steps. Crude images can often be obtained in few seconds, followed by gradually better images at intervals of a few seconds each, culminating in a high quality image in less than a minute.

7. Implementation and results

Both of the rendering methods described in this paper have been implemented in the C language on a Sun 4/280 with 32MB of main memory. In order to compare their costs and image quality, let us consider a simple test environment consisting of three mutually perpendicular polygons embedded in a $256 \times 256 \times 113$ voxel computed tomography (CT) study of a human head. Figures 6a and 6b are 380×380 pixel views of this data rendered with the hybrid ray tracer described in section 2 and the scan-conversion method described in section 3 respectively.

The worst-case memory requirements for the optimized hybrid ray tracer is $2N^3$ bytes to hold the shaded volumetric data, $(8^{(\log_2 N)+1} - 1)/7$ bits to hold the pyramid of binary volumes, and a relatively small amount of storage to hold the polygonal data and the various image sample arrays. For our test environment, this works out to about 18 megabytes. In the scan-conversion method, polygon colors and opacities c_G and α_G can be matted with voxel colors and opacities c_V and α_V on-the-fly, precluding the need for additional storage during scan-conversion. The worst-case memory requirements thus occur during ray tracing and are similar to those for the hybrid ray tracer.

The computational expense of shading and classifying volumetric data is proportional to the number of interesting voxels it contains (voxels whose opacity is not zero). For this example, the two steps took about 1 minute apiece. The cost of scan-converting a polygon is proportional to its surface area. The large polygons used here were scan-converted in about 1 second each. The cost of computing a pyramid of binary volumes is proportional to the number of bits it contains. For this example, pyramid construction took about 1 minute. In the case of the hybrid ray tracer, shading, classification, and pyramid construction are all independent of both the polygon geometry and the observer position, and may thus be thought of as pre-processing expenses. In the scan-conversion method, the pyramid depends on the composite of the volumetric and geometric data and must re-computed if either changes. The remaining costs of image generation are those of adaptive ray tracing, interpolation, and resampling, of which the first dominates. The cost of adaptive ray tracing is proportional to the number of rays traced. Figures 6a and 6b were generated using slightly fewer than one ray per pixel on average, and took 80 seconds and 70 seconds respectively to generate. For comparison, the volumetric data without the polygons can be rendered with the same level of refinement in about 50 seconds.

Figures 7a and 7b show details from figures 6a and 6b. In general, polygon edges produced by the hybrid ray tracer are sharper than those produced by 3D scan-conversion, although a higher image generation cost is paid for the better rendition. The slight aliasing noticeable in the bony tissue in these and other figures is due to insufficient bandlimiting during CT scanning and is not a result of the rendering process.

A visualization of where rays were cast during generation of figures 6a and 6b is given in figures 8a and 8b. Each pixel in these visualizations corresponds to one position in the array of sample colors described in paragraph 3 of section 6, and each 4×4 block of pixels corresponds to a single image pixel in figure 6. White pixels in the visualizations correspond to cast rays, and black pixels to samples filled in by interpolation. As expected, the number of rays per unit area is lowest in the interiors of homogeneous regions, higher along the silhouettes of volumetric objects, and in the case of the hybrid ray tracer, higher still along polygonal edges.

Figure 9 shows the effect of casting shadows using the algorithm described in section 4. The scene contains two light sources: a low-intensity light shining over the observer's right shoulder and a high-intensity light shining up from below and to the left. To insure that the shapes of shadowed objects are not completely obscured, shadows were only computed for the high-intensity light. While this is not strictly correct, the goal is enhanced insight, not photo-realism. Initial light strengths for the high-intensity source were assigned from a texture containing a filtered rectangular grid. The effect is to project this texture through the dataset and onto all illuminated surfaces, including the five scan-converted polygons. The addition of shadows roughly doubles the time required to compute the array of total voxel colors c_T , but does not affect the time required to generate each frame in a rotation sequence (assuming fixed lighting and data, a moving observer, and no specular highlights).

Figure 10 shows the effect of mapping a texture onto embedded polygons using the algorithm described in section 5. The effect on image generation time of adding textures depends on the number and size of textured polygons. For this example, image generation time increased by roughly 50%.

Figure 11 suggests one possible way in which these techniques might be applied to the problem of radiation treatment planning. The same CT dataset used in the previous figures has now been rendered to show both bone and soft tissue. A polygonally defined tumor (in purple) and radiation treatment beam (in blue) have been added using the hybrid ray tracer. A portion of the CT data has been clipped away to show the 3D relationship between the various objects.

Figure 12 illustrates how scan-converted backdrop planes and cast shadows can be used to enhance comprehension of an isovalue contour surface from an electron density map of Staphylococcus Aureus ribonuclease. The polymer backbone crosses the image from bottom to top, and two Tyrosine residues with their characteristic six-atom benzene rings can be seen extending to the left

and right sides of the backbone.

These studies have led to several welcome but unexpected results. Polygons rendered using 3D scan-conversion appear to have a finite thickness when rotated. Their opacity also varies with their angle relative to the view direction, as would a real slab of semi-transparent gel. Far from being distracting, these effects enhance the viewer's understanding of their shape and orientation. The embedding of polygons in volumetric data also seems to improve comprehension of the latter. For example, when the CT study of the head is bisected by a gridded, coronally-oriented (in the plane of the face) polygon and the ensemble is rotated, the presence of a backdrop of known shape and pattern improves appreciation of the fine structure of the sinuses and eye orbits. This suggests that, in addition to their primary role representing man-made or abstract entities, geometric primitives may be useful as diagnostic tools in the study of volumetric datasets.

8. Conclusions

Two methods for rendering mixtures of geometric and volumetric data have been described. Although both are reasonably efficient and produce images free from aliasing artifacts, each have advantages and disadvantages in terms of cost, image quality, and versatility.

A number of improvements to these methods can be suggested. The current implementation makes frequent use of tri-linear interpolation for resampling 3D data. A better filter would reduce the amount of blurring required during 3D scan-conversion, yielding sharper polygonal edges. In a similar vein, the shadow casting algorithm includes three successive resampling steps. By reorganizing the order of operations, one resampling can be eliminated, yielding crisper shadows. In the area of computational expense, the addition of a separate hierarchical data structure to handle polygonal data would speed up the hybrid ray tracer. It may also be possible to use a single data structure to represent both geometric and volumetric data.

Several strategies for treating mixtures of geometric and volumetric data remain unexplored. For example, geometric objects defined by extrusion of 2D profiles can be rendered using the shadow casting algorithm. If the profile is loaded into the array of initial light strengths, and all voxels in the volumetric data are assigned a slightly non-zero opacity, one obtains the effect of sculpted shafts of light passing through a dust-filled room. This technique could be used to visualize a radiation beam for cancer treatment. If the beam is assumed to pass through the volumetric data without attenuation, it can be rendered without a separate shadow pass and without the accompanying 3D representation.

9. Acknowledgements

The author wishes to thank Profs. Henry Fuchs, Stephen M. Pizer, Frederick P. Brooks Jr., and Turner Whitted of the Computer Science Department and Drs. Julian Rosenman and Edward L. Chaney of the Radiation Oncology Department for their encouragement and support. Thanks are also due to John Gauch for many enlightening discussions. The CT scans used in this paper were provided by the Radiation Oncology Department at North Carolina Memorial Hospital. The electron density map was provided by Dr. Chris Hill of the University of York Chemistry Department, and was reformatted and brought on-line with the help of Mark Harris of the University of North Carolina. The texture map is from the feature film Heidi by Hanna-Barbera Productions of Hollywood, California. This work was supported by ONR grant N00014-86-K-0680, NIH grant R01-CA39060, and IBM.

10. References

- Drebin, R.A., Carpenter, L., and Hanrahan, P., "Volume Rendering," *Computer Graphics*, Vol. 22, No. 4, August 1988, pp. 65-74.
- Feibush, E., Levoy, M., and Cook, R., "Synthetic Texturing using Digital Filters," *Computer Graphics*, Vol. 14, No. 3, July, 1980, pp. 294-301.
- Goodsell, D.S., Mian, S., and Olson, A.J., "Rendering of Volumetric Data in Molecular Systems" (in review, *Journal of Molecular Graphics*).
- Heckbert, P., "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, Vol. 6, No. 11, November, 1986, pp. 56-67.
- Johns, H. and Cunningham, J., *The Physics of Radiology*, Charles C. Thomas, 1983.
- Kaufman, A. and Shimony, E., "3D Scan-Conversion Algorithms for Voxel-Based Graphics," *Proc. ACM Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October, 1986, pp. 45-75.
- Kaufman, A., "Efficient Algorithms for 3D Scan-Conversion of Parametric Curves, Surfaces, and Volumes," *Computer Graphics*, Vol. 21, No. 4, July, 1987, pp. 171-179.
- Kaufman, A., "An Algorithm for 3D Scan-Conversion of Polygons," *Proc. EUROGRAPHICS '87*, Amsterdam, Netherlands, August, 1987, pp. 197-208.
- Kaufman, A. and Bakalash, R., "Memory and Processing Architecture for 3D Voxel-Based Imagery," *IEEE Computer Graphics and Applications*, Vol. 8, No. 6, November, 1988, pp. 10-23.
- Kaufman, A., personal communication, December, 1988.
- Levoy, M., "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications*, Vol. 8, No. 3, May, 1988, pp. 29-37.
- Levoy, M., "Efficient Ray Tracing of Volume Data," Technical Report 88-029, Computer Science Department, University of North Carolina at Chapel Hill, June, 1988 (in review, *ACM Transactions on Graphics*).
- Levoy, M., "Volume Rendering by Adaptive Refinement," Technical Report 88-030, Computer Science Department, University of North Carolina at Chapel Hill, June, 1988 (to appear in *The Visual Computer*).
- Lorenson, W. and Cline, H., "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics*, Vol. 21, No. 4, July, 1987, pp. 163-169.
- Max, N., "Atmospheric Illumination and Shadows," *Computer Graphics*, Vol. 20, No. 4, August, 1986, pp. 117-124.
- Mosher, C., personal communication, December, 1988.

Pizer, S.M., Fuchs, H., Mosher, C., Lifshitz, L., Abram, G.D., Ramanathan, S., Whitney, B.T., Rosenman, J.G., Staab, E.V., Chaney, E.L. and Sherouse, G., "3-D Shaded Graphics in Radiotherapy and Diagnostic Imaging," *NCGA '86 conference proceedings*, Anaheim, CA, May, 1986, pp. 107-113.

Reynolds, A., Letter to the editor of *IEEE Computer Graphics and Applications* (to appear).

Sabella, P., "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graphics*, Vol. 22, No. 4, August 1988, pp. 51-58.

Upton, C. and Keeler, M., "VBUFFER: Visible Volume Rendering," *Computer Graphics*, Vol. 22, No. 4, August 1988, pp. 59-64.

Whitted, T., "An Improved Illumination Model for Shaded Display," *Communications of the ACM*, Vol. 23., No. 6, June, 1980, pp. 343-349.

Williams, L., "Casting Curved Shadows on Curved Surfaces," *Computer Graphics*, Vol. 12, No. 3, August 1978, pp. 270-274.

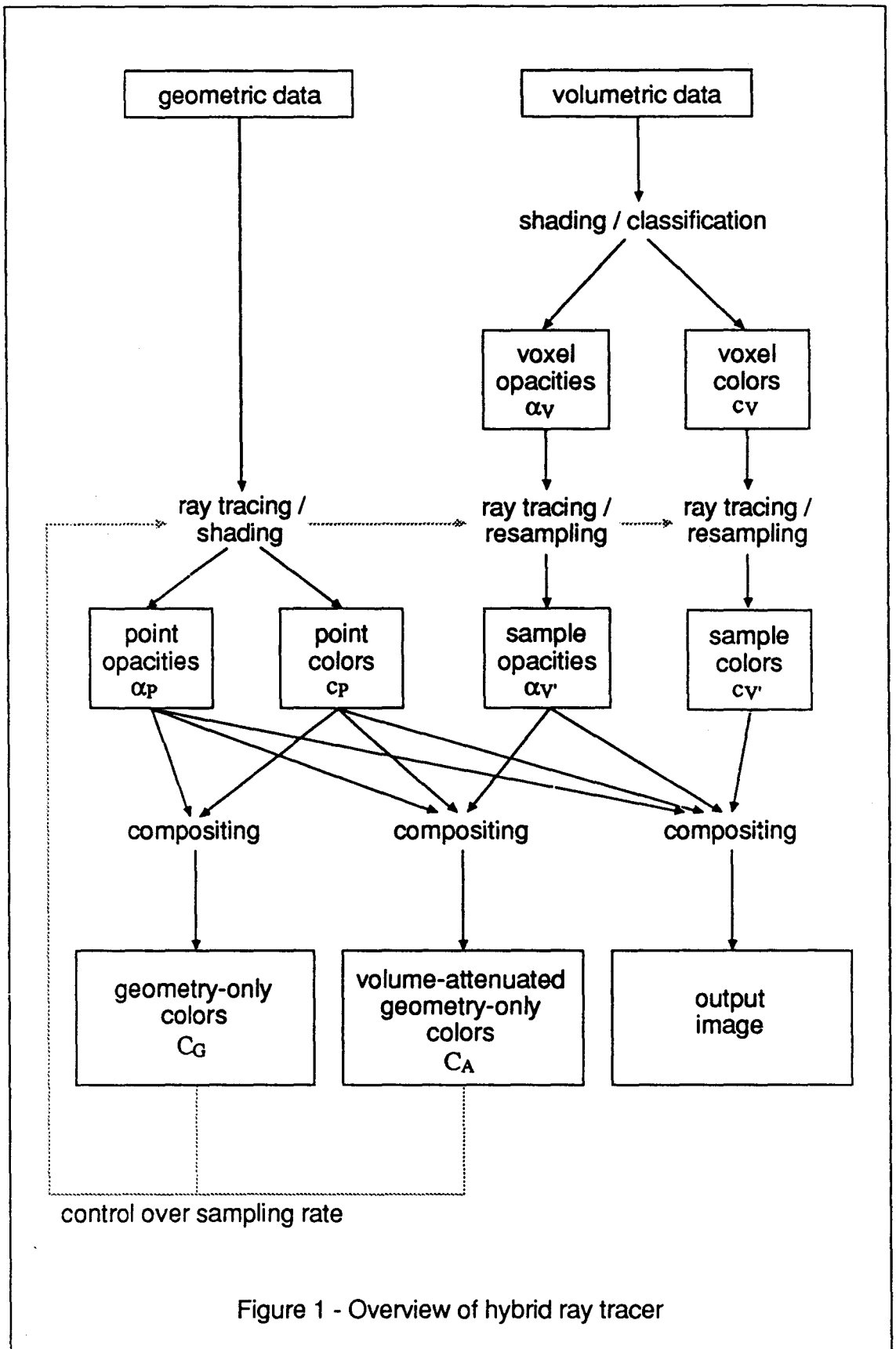


Figure 1 - Overview of hybrid ray tracer

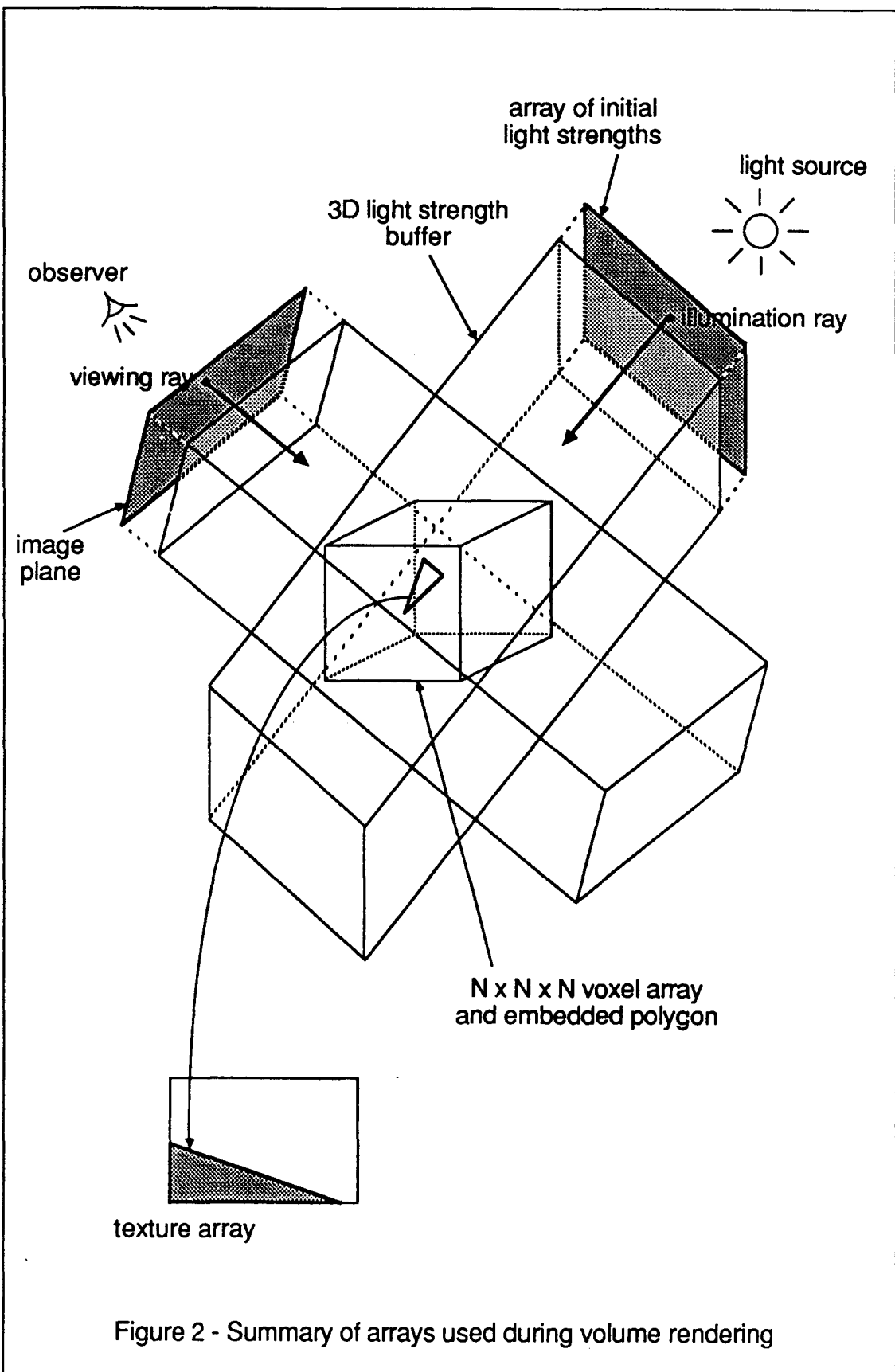
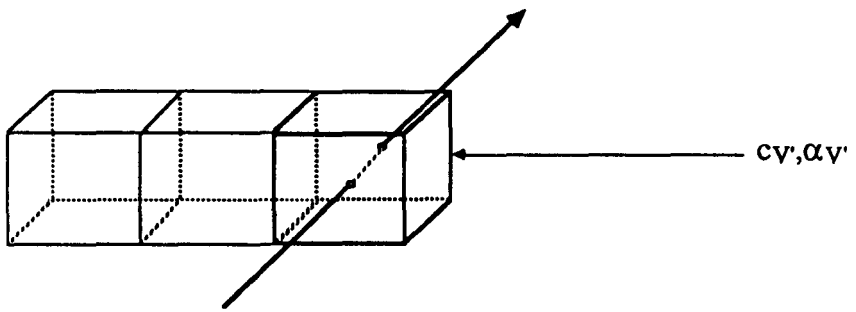
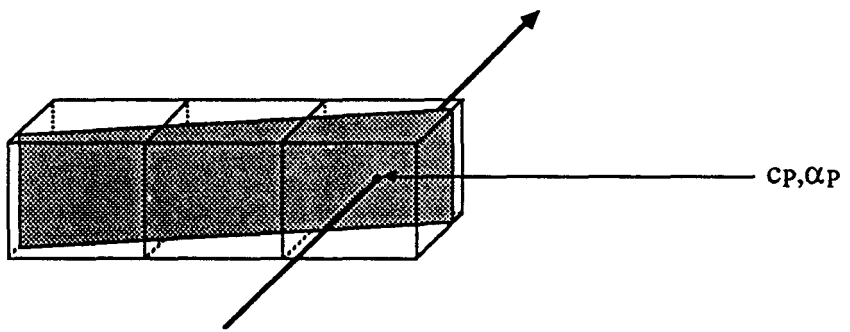


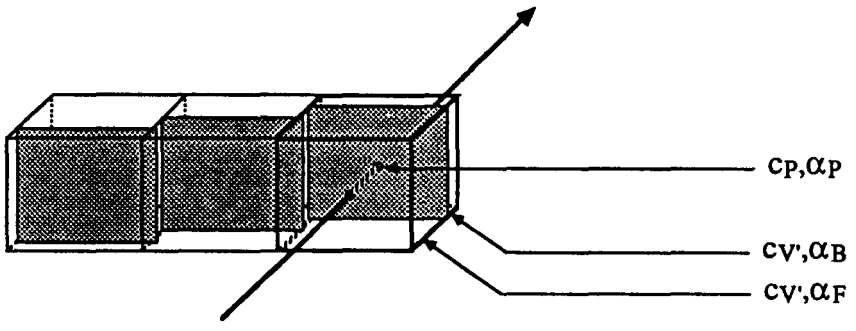
Figure 2 - Summary of arrays used during volume rendering



a - Slab of volumetric data along viewing ray



b - Polygon embedded in volumetric data



c - Approximate solution to hidden-volume problem

Figure 3 - Rendering of polygon embedded in volumetric data

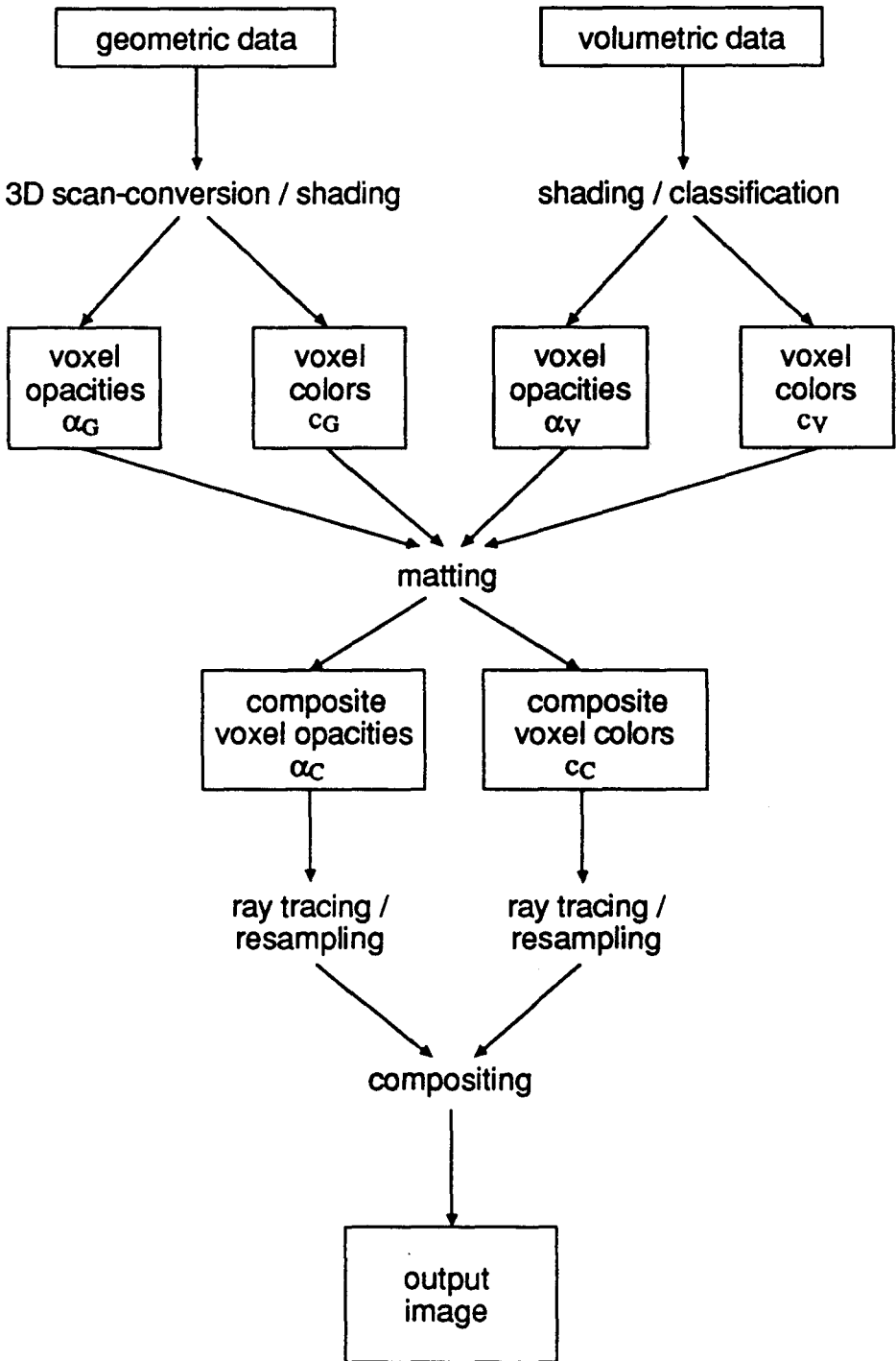


Figure 4 - Overview of 3D scan-conversion method

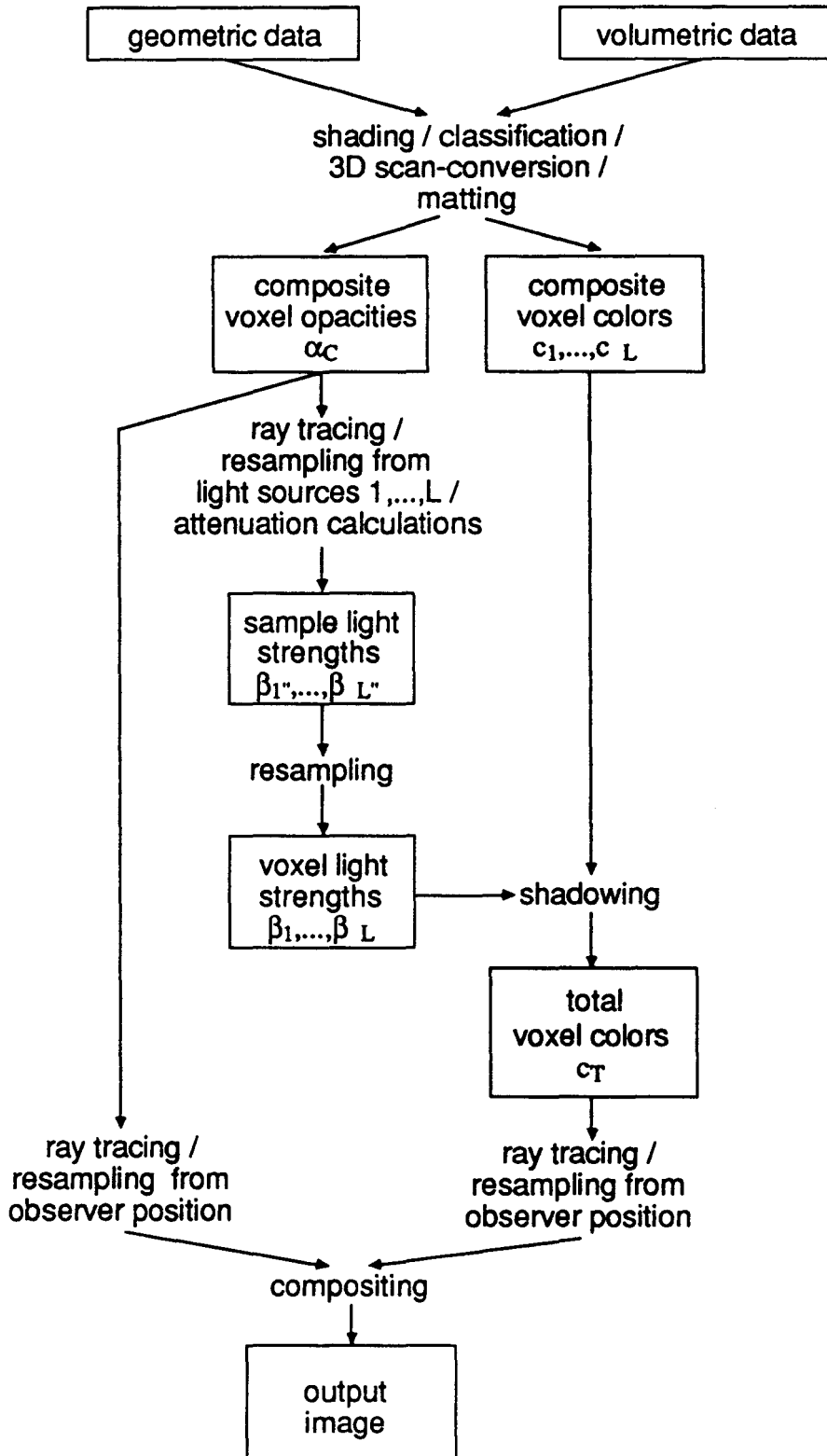


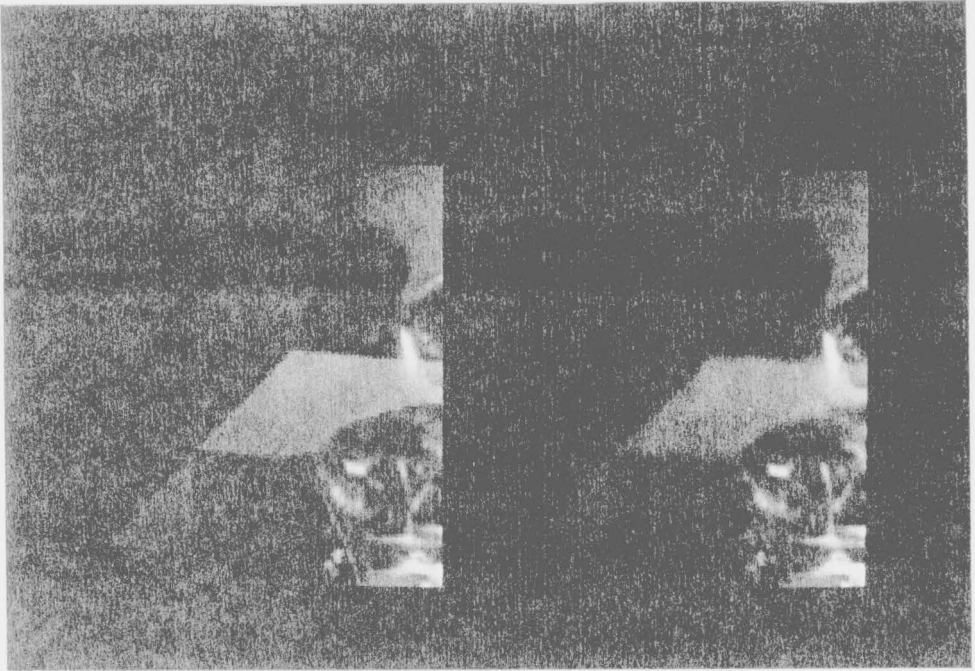
Figure 5 - Addition of shadow calculations to 3D scan-conversion method



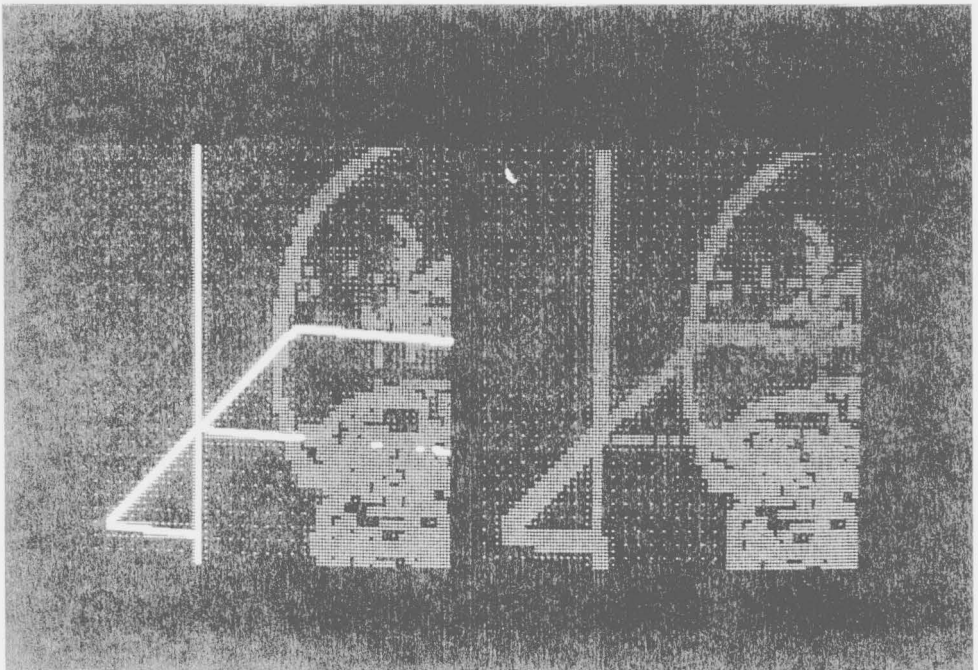
Figure 6a - Volume rendering of human head and embedded polygons, generated using hybrid ray tracer



Figure 6b - Volume rendering of human head and embedded polygons, generated using 3D scan-conversion method



Figures 7a and 7b - Details from figures 6a and 6b, showing comparative image quality of hybrid ray tracer and 3D scan-conversion methods



Figures 8a and 8b - Visualization of where rays were cast during generation of figures 6a and 6b

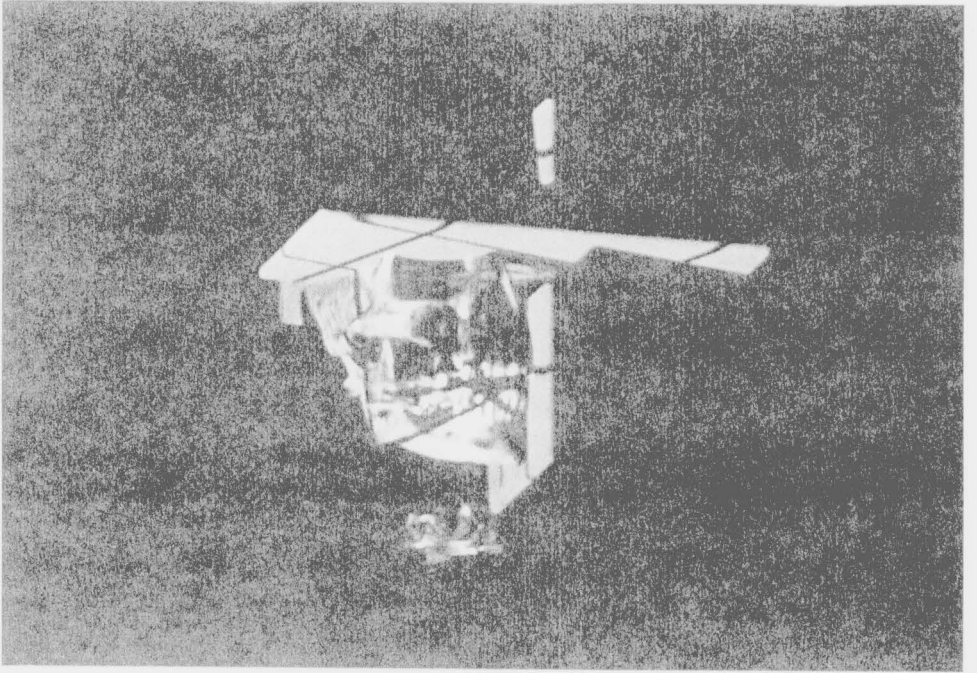


Figure 9 - Volume rendering with shadows,
generated using modified 3D scan-conversion method

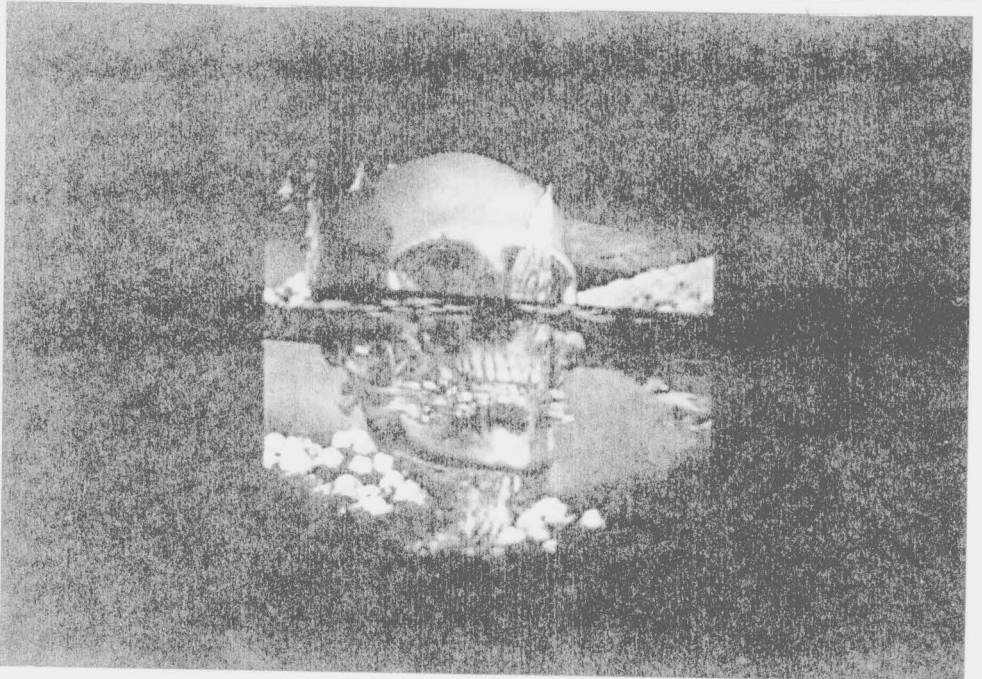


Figure 10 - Volume rendering with textured polygons,
generated using modified hybrid ray tracer

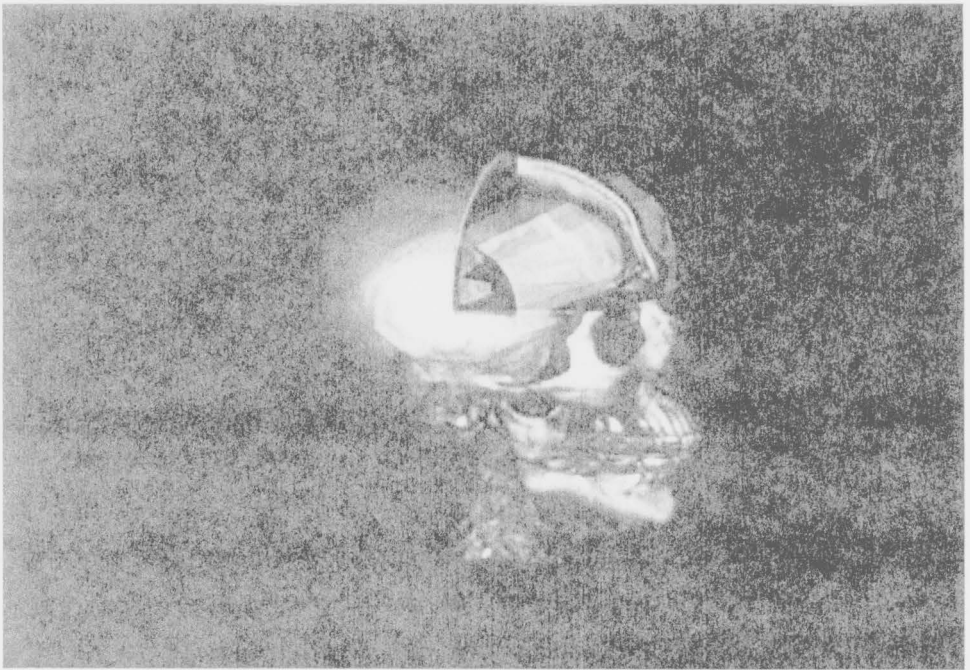


Figure 11 - Volume rendering of human head, showing bone, soft tissue, tumor (purple) and radiation treatment beam (blue)

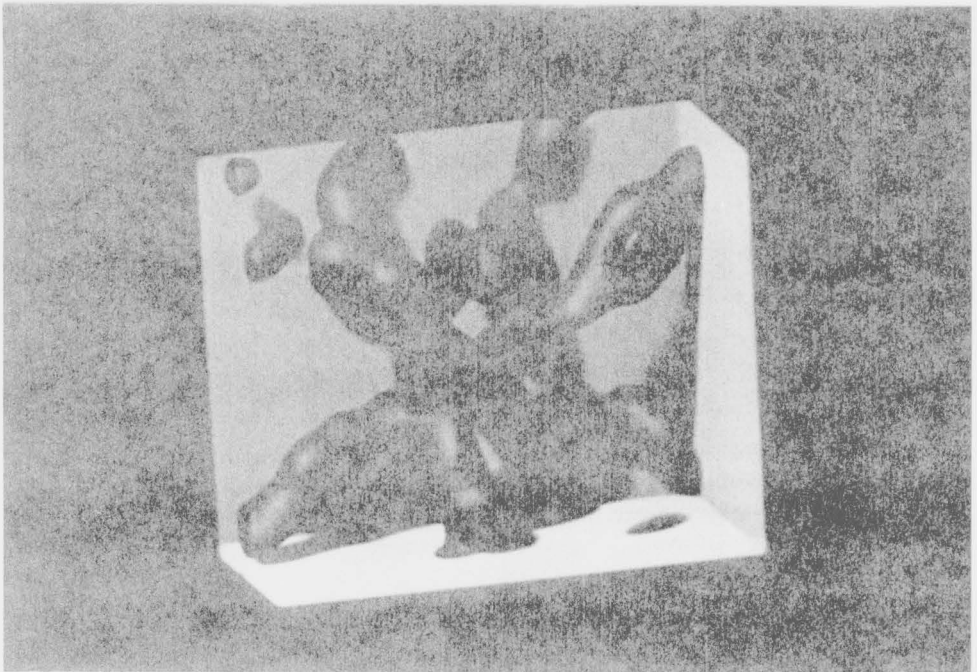


Figure 12 - Volume rendering of isovalue contour surface from electron density map of *Staphylococcus Aureus* ribonuclease