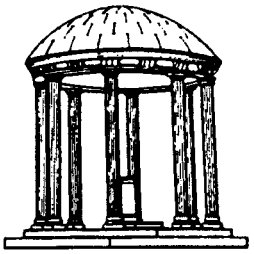
A Hybrid Ray Tracer for Rendering Polygon and Volume Data

TR89-035

October, 1989

Marc Levoy



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

A Hybrid Ray Tracer for Rendering Polygon and Volume Data

Marc Levoy

September, 1989

Department of Computer Science
University of North Carolina at Chapel Hill

Abstract

Volume rendering is a relatively new technique for visualizing sampled 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 polygonally defined objects. The solution proposed is a hybrid ray tracing algorithm. Rays are simultaneously cast through a set of polygons and a volume data array, 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, volume data lying immediately in front of and behind polygons are given special treatment. To avoid aliasing of polygon data, a form of selective supersampling is employed. The cost, image quality, and versatility of the algorithm are evaluated using data from 3D medical imaging applications.

1. Introduction

Many data visualization problems require geometrically defined objects and sampled fields to appear together in a single image. Medical applications include superimposition of radiation treatment beams over patient anatomy for the oncologist and display of medical prostheses for the orthopedist. This paper addresses the problem of rendering mixtures of polygonally defined objects and sampled scalar functions of three spatial dimensions.

The solution most commonly adopted is to convert the polygon and volume data into a common representation. Examples of this approach include fitting of polygonal meshes to the volume data (Fuchs et al., 1977; Lorensen and Cline, 1987) and 3D scan-conversion of the polygon data to yield a binary voxel representation (Kaufman, 1987). Both of these approaches require a binary classification of the volume data, leading to artifacts in the generated image. Kaufman's use of a binary voxel representation for polygonally defined objects gives rise to additional artifacts, although a solution to this problem has apparently been worked out (Kaufman, 1988).

An alternative approach is to directly display both types of data using a hybrid rendering algorithm. By preserving the original representations, conversion artifacts are eliminated. Examples of this approach include separate rendering of geometrically defined objects and volume data followed by a depth-sorted compositing post-process (Goodsell et al., 1989), and extension of a conventional ray tracer to handle volume data (Johnson and Mosher, 1989). The first method uses a Z-buffer visibility algorithm for the geometric data and consequently cannot handle multiple layers of semi-transparent geometry interspersed with volume data. The second method is closest to the present technique, but a detailed description of it has not been published.

The solution presented in this paper is an extension of the volume rendering algorithm presented in (Levoy 1988a). Its extension to handle polygon data was first described in (Levoy, 1988b). This paper represents a condensation of material appearing in (Levoy, 1989a).

2. Overview of the algorithm

Figure 1 gives an overview of the hybrid ray tracer. It begins with a 3D array of voxels and a list of polygons. In this paper, we treat voxels as point samples of a continuous function rather than as volumes of homogeneous value. For simplicity, let us assume a scalar-valued array forming a cube V voxels on a side. Voxels are indexed by a vector $\mathbf{i} = (i,j,k)$ where $i,j,k = 1, \ldots, V$. Using local operators, a scalar or vector color $C_V(\mathbf{i})$ and an opacity $\alpha_V(\mathbf{i})$ is derived for each voxel.

Parallel rays are then traced into the data from an observer position. Let us assume for the purpose of this overview that one ray is cast per pixel and that the image is a square measuring P pixels on a side. Pixels and hence rays are indexed by a vector $\mathbf{u} = (u,v)$ where $u,v = 1, \ldots, P$. For each ray, a vector of colors and opacities is computed by resampling the volume data at W evenly spaced locations along the ray and trilinearly interpolating from the colors and opacities in the eight voxels surrounding each sample location. Samples are indexed by a vector $\mathbf{U} = (u,v,w)$ where (u,v) identifies the ray, and $w = 1, \ldots, W$ corresponds to distance along the ray with $w = 1$ being closest to the eye. The color and opacity of sample $\mathbf{U}$ are denoted $C_V(\mathbf{U})$ and $\alpha_V(\mathbf{U})$ respectively.

Independently, all intersections between the ray and polygons in the environment are computed and shaded, yielding a color $C_P(\mathbf{x})$ and opacity $\alpha_P(\mathbf{x})$ for each point of intersection, which is denoted by a real vector of the form $\mathbf{x} = (x,y,z)$ where $1 \le x,y,z \le V$. For simplicity, we assume that all polygons lie strictly within the boundaries of the volume dataset.

Finally, the resampled volume colors and opacities are composited with each other and with the polygon colors and opacities in depth-sorted order to yield for each ray $\mathbf{u}$ a color $C(\mathbf{u})$.

The number of contributions N affecting a ray is equal to the number of volume samples W drawn along the ray plus the number of polygons intersected by the ray. Contributions are indexed by an integer $n = 1, \ldots, N$, and each contribution consists of a color $C(n)$ and an opacity $\alpha(n)$. If contribution n is a volume sample, then $C(n) = C_V(\mathbf{U})$ and $\alpha(n) = \alpha_V(\mathbf{U})$ for some $\mathbf{U}$. If the contribution is a ray-polygon intersection point, then $C(n) = C_P(\mathbf{x})$ and $\alpha(n) = \alpha_P(\mathbf{x})$ for some $\mathbf{x}$. Working from front to back, the color $C_{out}(n)$ and opacity $\alpha_{out}(n)$ of a ray after processing contribution n is related to the color $C_{in}(n)$ and opacity $\alpha_{in}(n)$ of the ray before processing the contribution and the color $C(n)$ and opacity $\alpha(n)$ of the contribution by the transparency formula (Porter and Duff, 1984)

$$\hat{C}_{out}(n) = \hat{C}_{in}(n) + \hat{C}(n)(1 - \alpha_{in}(n)) \tag{1a}$$

and

$$\alpha_{out}(n) = \alpha_{in}(n) + \alpha(n)(1 - \alpha_{in}(n)). \tag{1b}$$

where $\hat{C}_{in}(n) = C_{in}(n)\alpha_{in}(n)$, $\hat{C}_{out}(n) = C_{out}(n)\alpha_{out}(n)$, and $\hat{C}(n) = C(n)\alpha(n)$. After all N contributions to a ray have been processed, the color C of the ray is obtained from the expression $C = \hat{C}_{out}(N) / \alpha_{out}(N)$.

3. Computing visibility at polygon-volume intersections

Compositing at evenly spaced sample locations along a viewing ray correctly solves the visibility of a gel composed of rectangular slabs each of identical size and homogeneous color and opacity. 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 as shown in figure 2a.

When a polygon is embedded in the volume data, it intersects some subset of these slabs, obscuring some portion of the gel in each slab and being obscured by the remainder. Two types of intersections can be identified:

(1) *The polygon intersects only the side faces of the slab.* An example is shown in figure 2b. In this case, an exact solution of the hidden-volume problem inside the slab can be encorporated into the ray tracer. The contributions made to the color of the ray by the trapezoidal volumes lying in front of and behind the polygon are equivalent to those made by rectangular volumes separated by a plane lying perpendicular to the ray and placed at the point of intersection between the polygon and the ray as shown in figure 2c. Let the thicknesses of the volumes lying in front of and behind the perpendicular plane be denoted t_F and t_B respectively. The opacity α_V of a rectangular volume of homogeneous material can be computed from its density d_V and thickness t_V using the exponential relation (Johns and Cunningham, 1983)

$$\alpha_V = 1 - e^{-d_V t_V} \tag{2}$$

where $0 < \alpha_V < 1$ and $\alpha_V = 1$ signifies complete attenuation. Solving equation (2) for opacities α_F and α_B in terms of opacity α_V and thicknesses t_F and t_B gives

$$\alpha_F = 1 - (1 - \alpha_V)^{t_F/t_V} \tag{3a}$$

and

$$\alpha_B = 1 - (1 - \alpha_V)^{t_B/t_V}. \tag{3b}$$

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. Each ray-polygon intersection results in the addition of one polygon contribution to the affected ray and the splitting of one volume contribution into two. For notational convenience later in the paper, we denote the augmented number of contributions affecting a ray as N'.

(2) *The polygon intersects the front face of the slab, the rear face, or both.* An example is shown in figure 2d. In this case, the polygon contributes to more than one slab along the ray, although it only intersects the ray once. An exact solution of the hidden-volume problem would require carrying pixel coverage information from slab to slab. In the present algorithm, we instead supersample the slab by casting more rays as described in the next section. Increasing the number of rays effectively subdivides the slab along its width and height as shown in figure 2e. Within each subslab, we apply the trapezoidal hidden-volume solution already described. As the number of subslabs increases, a larger percentage will contain intersections only along their side faces. In the limit, this method converges to the exact solution. In practice, a few extra rays suffice to eliminate objectionable artifacts.

4. Selective supersampling of polygon data

Ray tracing is a point sampling process and therefore susceptible to aliasing artifacts. In the present algorithm, aliasing may arise from several distinct causes. In scenes containing volume data alone, trilinear interpolation between data voxels, coupled with sampling rates of one ray per pixel and one or two ray samples per voxel, generally suffices to avoid aliasing. In scenes containing polygon data alone, this sampling rate is insufficient to avoid aliasing, which typically appears as staircasing along polygon edges, polygon-polygon intersections, and shading highlights. Scenes containing polygon-volume intersections may also alias, depending on the angle between the polygon and the viewing ray. For a volumetrically defined object sliced by a polygon seen nearly on edge, a sampling rate of one ray per pixel usually results in staircasing along the curve of intersection between the polygon and the volumetrically defined object.

Adaptive supersampling based on measures of local image complexity is commonly employed to reduce aliasing artifacts in ray tracers (Whitted, 1980; Lee et al., 1985). In the present algorithm, the image plane is divided into square sample regions and rays are cast from the four corners of each region. If the range of colors returned by the four rays is greater than some threshold, the region is divided into four subregions and more rays are cast. Subdivision continues until the range of colors falls below the threshold or the size of the region reaches some specified minimum. Since it is unnecessary to cast more than one ray per pixel in scenes containing only volume data, initial sample regions are two pixels on a side and enclose four rays each.

Given the small number of samples drawn in each initial region, it is difficult to reliably distinguish between low-frequency image fluctuations due to volume data and high-frequency fluctuations due to polygon data. To reduce the number of incorrect subdivision decisions and hence the number of unnecessary rays cast, the present algorithm computes for each region a second set of sample colors that contain contributions only from polygon data. Solving equation (1) for ray color C and expressing the solution as a depth-sorted list of N' contributions gives

$$C = \sum_{n=1}^{N'} \left[C(n)\alpha(n) \prod_{n'=1}^{n} (1 - \alpha(n')) \right] \tag{4}$$

where contribution n is either a volume sample (see section 2), a volume sliver (see section 3), or a ray-polygon intersection point. By setting $C(n) = 0$ for all volume samples and slivers, we obtain for each ray $\mathbf{u}$ a new color $C_P(\mathbf{u})$ that contains contributions only from polygon data, but attenuated by passage through the volume data. The resulting image looks like a polygonally defined scene viewed through non-homogeneous black smoke. If the range of C within a sample region exceeds the threshold, but the range of C_P does not, then the observed color difference is due to volume data rather than a geometric event (such as a polygon edge) and no subdivision occurs. If the range of C_P exceeds the threshold, then the observed color difference is due to a geometric event and subdivision occurs.

5. Implementation and results

The algorithm described in this paper has been implemented in the C language on a Sun 4/280 with 32MB of main memory. The implementation employs several techniques that take advantage of spatial coherence to reduce rendering time. The first technique consists of constructing a hierarchical spatial occupancy enumeration (i.e. a pyramid of binary volumes, or complete octree) to speed up the search for non-empty (non-transparent) voxels along viewing rays. The second technique technique uses an opacity threshold to adaptively terminate ray tracing. Early termination of a ray reduces both the number of voxels that must be resampled and the number of ray-polygon intersections that must be shaded. The third technique consists of casting a sparse grid of rays, less than one per pixel, and adaptively increasing the number of rays in regions of high image complexity. For regions larger than two pixels on a side, subdivision decisions are based on contributions by both volume and polygon data. For smaller regions, decisions are based on polygon data alone as described in the preceding section. When all regions have been processed, an image is formed by bilinearly interpolating between the available colors and resampling at the display resolution. Combining all three optimizations, savings of more than two orders of magnitude over brute-force ray tracing have been obtained for many datasets (Levoy, 1990a; Levoy, 1990b).

Figure 3 shows a 380 × 380 pixel rendering of a simple test environment consisting of three mutually perpendicular polygons embedded in a 256 × 256 × 113 voxel computed tomography (CT) study of a human head. Computation of voxel colors and opacities took about 2 minutes on the Sun 4. Construction of a pyramid of binary volumes from the array of voxel opacities took about 1 minute. Ray tracing of the preprocessed data to generate figure 3 took 80 seconds. The number of rays cast per pixel ranged from 1 ray per 16 pixels up to 4 rays per pixel with an

average rate of slightly less than 1 ray per pixel. By contrast, the volume data without the polygons can be rendered with the same level of refinement in about 50 seconds.

Figure 4a shows a detail from figure 3. The image is seen to be generally free from aliasing artifacts. The slight aliasing noticeable in the bony tissue is due to insufficient bandlimiting during CT scanning and is not a result of the rendering process. A visualization of where rays were cast is shown in figure 4b. Each 2×2 block of pixels in this visualization corresponds to one pixel in figure 3. White pixels in figure 4b correspond to cast rays in figure 3, and black pixels correspond to samples filled in by interpolation. As expected, the number of rays per unit area is lowest in the interiors of homogeneous regions and highest along polygon edges and polygon-polygon intersections.

Figure 5 shows the effect of mapping a texture onto the embedded polygons. At each point of intersection between a ray and a polygon, a mapping is performed from object space to a 2D texture array. The texture is resampled and used to modify the shading calculations for the polygon. Assuming that the texture is properly filtered during resampling, it is not necessary to increase the number of rays cast in the interior of textured polygons. To prevent the addition of textures from triggering unnecessary supersampling, their contribution is omitted during computation of $C_P(\mathbf{u})$. Note that a 3D texture array could be used in place of the 2D texture, thus simultaneously visualizing two 3D datasets.

Figure 6 suggests one possible way in which these techniques might be applied to the problem of radiation treatment planning. A polygonally defined treatment volume (in purple) and treatment beam (in blue) have been added to a color rendering of the CT dataset used to generate figure 3. A portion of the CT data has been clipped away to show the 3D relationships between the various objects.

6. Conclusions

A hybrid ray tracer for rendering mixtures of polygon and volume data has been presented. The algorithm operates directly on the original data representations, thereby eliminating the conversion artifacts exhibited by other methods. The algorithm is designed to be efficient and to avoid aliasing artifacts.

These studies have led to several welcome but unexpected results. The embedding of polygons in volume data 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, polygonal constructs may prove useful as diagnostic tools in the study of volume data.

A number of improvements to these methods can be suggested. The selective supersampling method described in section 4 is not entirely successful in eliminating unnecessary ray casting. In particular, a volumetrically defined object silhouetted against a polygonally defined backdrop appears to the algorithm like a polygon-volume intersection. To distinguish between these two cases, it would appear necessary to establish 3D sample regions and to examine the contributions made to local image complexity by each region. This approach is currently under investigation. In another vein, the pyramid of binary volumes reduces the cost of tracing of rays through segments of volume data lying between successive polygon intersections. This method does not, however, reduce the cost of computing intersections between rays and polygons. Among the solutions currently under consideration are a separate data structure to handle polygon data (e.g. bounding volumes or spatial subdivision) or a single data structure to represent both polygons and volume data.

Although the algorithm presented in this paper demonstrates the efficacy of operating directly on the original data representations, this does not imply that artifact-free images cannot be generated by methods employing representation conversions. For example, the author has developed a rendering algorithm based on 3D scan-conversion with analytic anti-aliasing. Polygons are shaded, filtered, sampled, and combined with the volume data. The resulting composite dataset is then rendered using published methods. If the polygon data is sufficiently bandlimited prior to sampling, this method also produces images free from aliasing artifacts (Levoy, 1989a). Images generated using this algorithm have appeared in (Fuchs et al., 1989), and the algorithm forms the basis for a proposed real-time volume rendering workstation supporting the display of both polygon and volume data (Levoy, 1989b).

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 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, NCI grant P01-CA47982, and IBM.

References

Fuchs, H., Kedem, Z.M., and Uselton, S.P. (1977), "Optimal surface reconstruction from planar contours," *Communications of the ACM* 20(10):693-702.

Fuchs, H., Levoy, M., and Pizer, S. (1989), "Interactive Visualization of 3D Medical Data," *IEEE Computer* 22(8):46-51.

Goodsell, D.S., Mian, S., and Olson, A.J. (1989), "Rendering of Volumetric Data in Molecular Systems," *Journal of Molecular Graphics* 7(1):41-47.

Johns, H. and Cunningham, J. (1983), *The Physics of Radiology*, Charles C. Thomas.

Johnson, E.R. and Mosher, C.E.,Jr. (1989), personal communication.

Kaufman, A. (1987), "An Algorithm for 3D Scan-Conversion of Polygons," *Proc. Eurographics '87*, ed. G. Marechal, Elsevier Science Publishers B.V., North-Holland, pp. 197-208.

Kaufman, A. (1988), personal communication.

Lee, M.E., Redner, R.A., and Uselton, S.P. (1985), "Statistically Optimized Sampling for Distributed Ray Tracing," *Computer Graphics* 19(3):61-67.

Levoy, M. (1988a), "Display of Surfaces from Volume Data," *IEEE Computer Graphics and Applications* 8(3):29-37.

Levoy, M. (1988b), "Rendering Mixtures of Geometric and Volumetric Data," Technical Report 88-052, Computer Science Department, University of North Carolina at Chapel Hill.

Levoy, M. (1989a). *Display of Surfaces from Volume Data*, Ph.D. dissertation, University of North Carolina at Chapel Hill.

Levoy, M. (1989b), "Design for a Real-Time High-Quality Volume Rendering Workstation," *Proc. Chapel Hill Workshop on Volume Visualization*, University of North Carolina, pp. 85-92.

Levoy, M. (1990a), "Volume rendering by adaptive refinement," *The Visual Computer* 6(1). In press.

Levoy, M. (1990b), "Efficient Ray Tracing of Volume Data," *ACM Transactions on Graphics*. In press.

Lorensen, W.E. and Cline, H.E. (1987), "Marching Cubes: A High Resolution 3D Surface Construction Algorithm," *Computer Graphics* 21(4):163-169.

Porter, T. and Duff, T. (1984), "Compositing Digital Images," *Computer Graphics* 18(3):253-259.

Whitted, T. (1980), "An Improved Illumination Model for Shaded Display," *Communications of the ACM* 23(6):343-349.

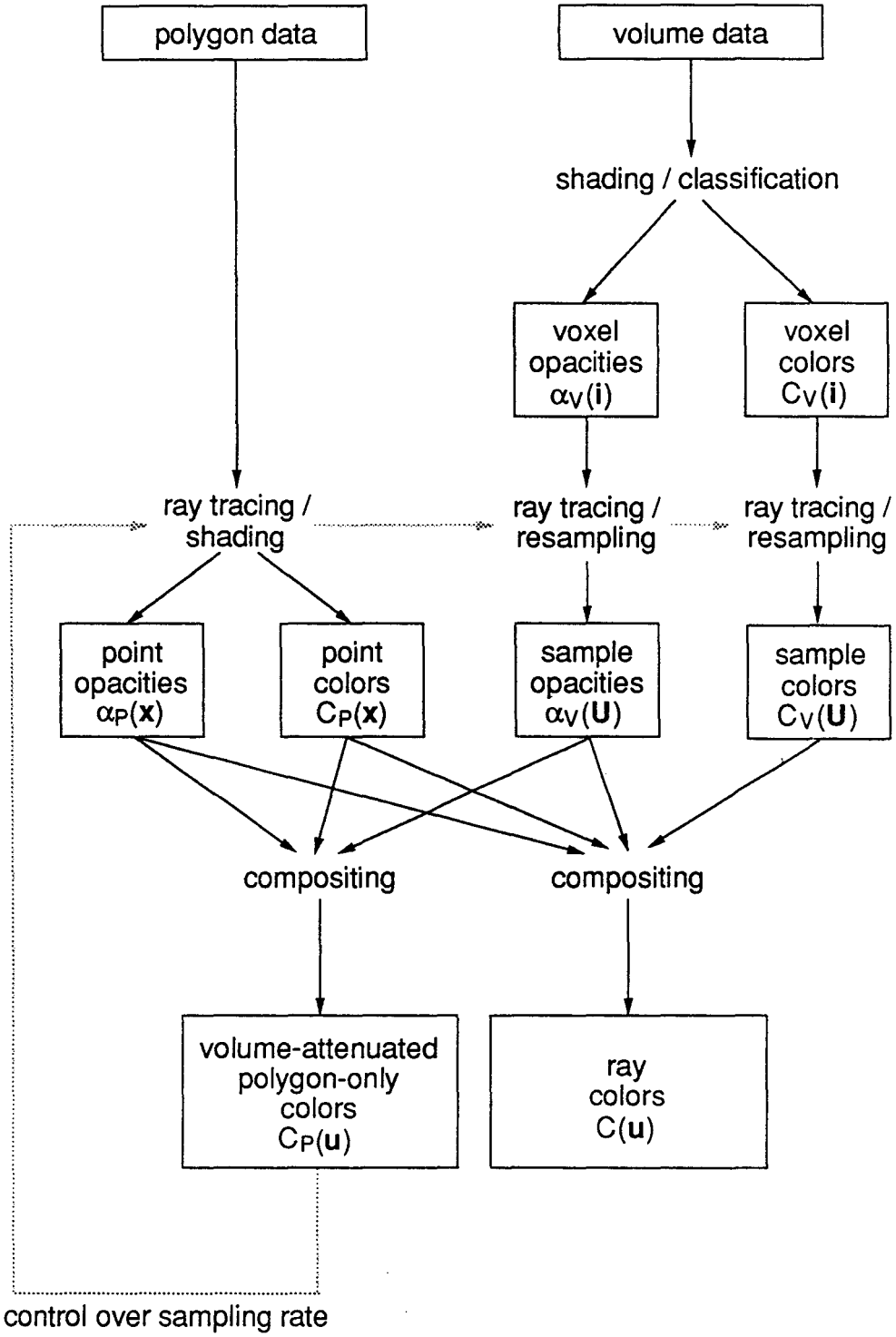
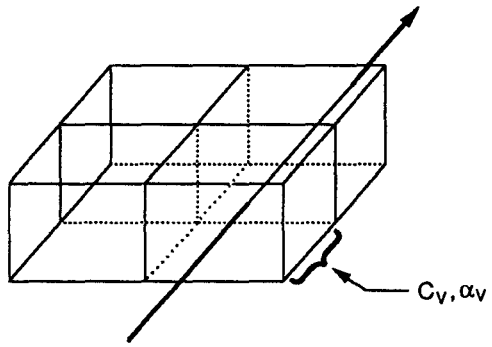
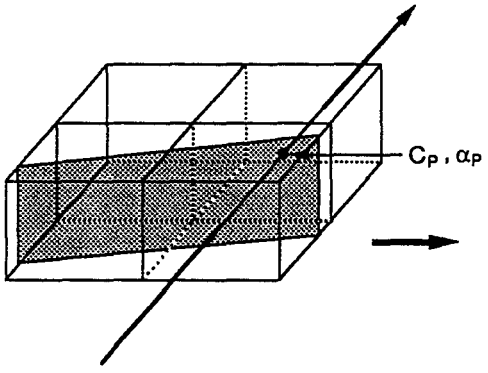


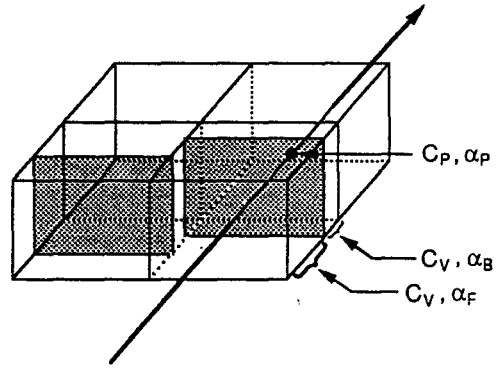
Figure 1: Overview of hybrid ray tracer for rendering polygon and volume data



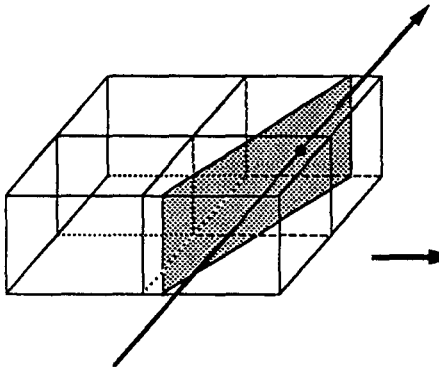
(a) Slabs of volume data along viewing ray



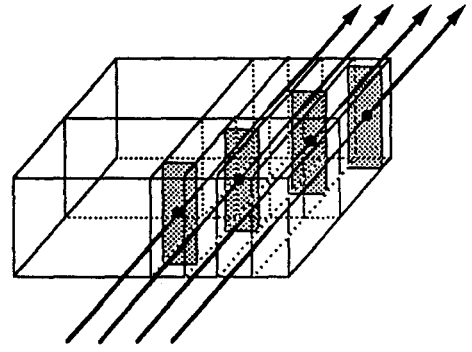
(b) Polygon intersects sides only



(c) Exact hidden-volume solution



(d) Polygon intersects front or back



(e) Supersampling coupled with (c)

Figure 2: Rendering of polygon embedded in volume data

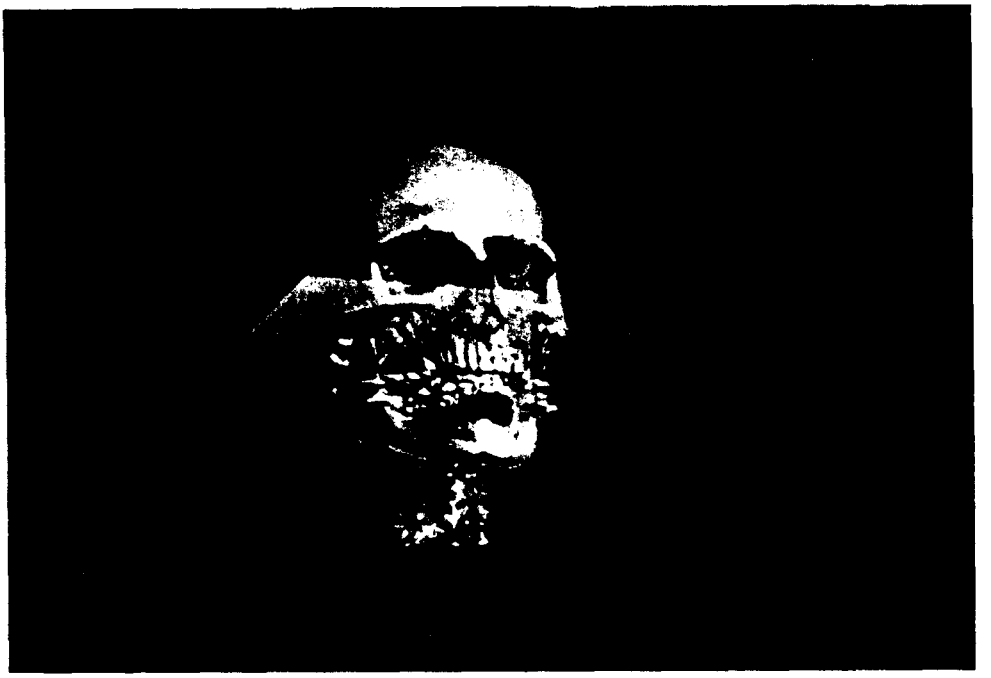


Figure 3: Color rendering of CT dataset and embedded polygons, generated using hybrid ray tracer

Figure 4a: Detail from figure 3 showing anti-aliasing

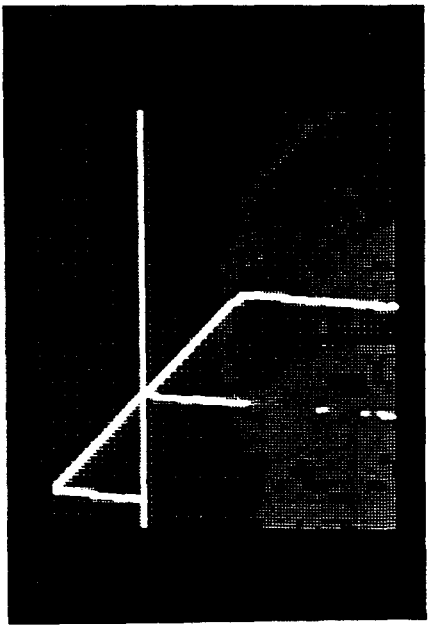


Figure 4b: Visualization of where rays were cast

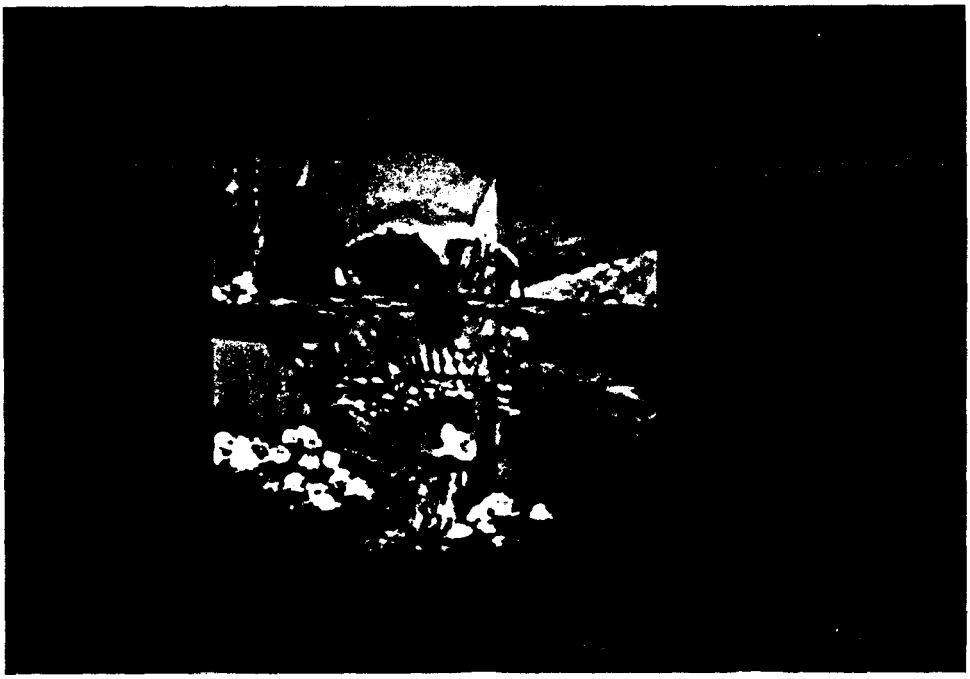


Figure 5: Color rendering of CT dataset and textured polygons, generated using modified hybrid ray tracer

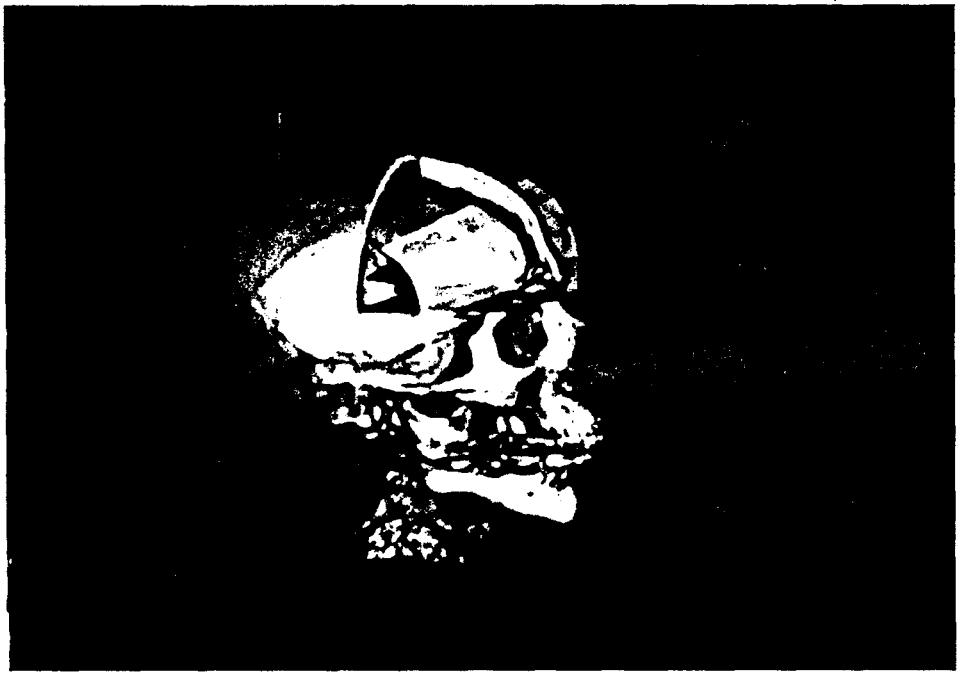


Figure 6: Color rendering of CT dataset showing bone, soft tissue, radiation target volume (purple), and treatment beam (blue)