A Hybrid Ray Tracer for Rendering Polygon and Volume Data

Marc Levoy, University of North Carolina at Chapel Hill

Abstract

Volume rendering is a technique for visualizing sampled functions of three spatial dimensions by computing 2D projections of a colored semi-transparent volume. 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 aliasing of polygonal edges at modest computational expense, a form of selective supersampling is employed. To avoid errors in visibility at polygon-volume intersections, volume samples lying immediately in front of and behind polygons are given special treatment. The cost, image quality, and versatility of the algorithm are evaluated using data from 3D medical imaging applications.

Tag phrase for table of contents and magazine spine: Volume rendering

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 [5, 16] and 3D scan-conversion of the polygon data to yield a binary voxel array [10]. Both of these approaches require a binary classification of the volume data. Since classification is performed on a sampled function, small features are often incorrectly classified, leading to artifacts (such as spurious or missing surfaces) in generated images.

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 [6], and extension of a conventional ray tracer to handle volume data [8]. 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 based on volume rendering, a technique for displaying sampled fields by approximating the transmission of light through a colored semi-transparent volume. Previous volume rendering techniques include the slice-by-slice method of Drebin et al. [4], the cell-by-cell method of Upson and Keeler [22], the sample-by-sample method of Westover [23], and the ray tracing methods of Levoy [12], Sabella [21], and Upson and Keeler [22]. The present algorithm is an extension of the author's volume ray tracer [12] to handle both polygon and volume data.

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 [12], 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 using a Phong lighting model [19], 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 [20]

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

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 $C = \hat{C}_{out}(N) / \alpha_{out}(N)$.

Selective supersampling of polygon data

Ray tracing is a point sampling process and therefore susceptible to aliasing artifacts. Adaptive supersampling based on measures of local image complexity is commonly employed to reduce these artifacts [24, 11, 3, 2, 9, 17, 18]. In the present algorithm, the image plane is divided into abutting square sample regions one pixel on a side 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.

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 edges, polygon-polygon intersections, and polygon-volume intersections. Assuming that the volume data is properly filtered during resampling (e.g. using trilinear interpolation as just described), supersampling is not required in the first case. 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

$$C = \sum_{n=1}^{N} \left[C(n)\alpha(n) \prod_{n'=1}^{n} (1 - \alpha(n')) \right] \tag{2}$$

where contribution n is either a volume sample or a ray-polygon intersection point. By setting $C(n) = 0$ for all volume samples, 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.

Computing visibility at polygon-volume intersections

Compositing at evenly spaced sample locations along a viewing ray approximates the visibility of a volume 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 a subset of these slabs, obscuring some portion of 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 incorporated 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 [7]

$$\alpha_V = 1 - e^{-d_V t_V} \tag{3}$$

where $0 < \alpha_V < 1$ and $\alpha_V = 1$ signifies complete attenuation. Solving equation (3) for opacities α_F and α_B in terms of opacity α_V and thicknesses t_F and t_B gives

$$\alpha_F = 1 - (1 - \alpha_V)^{t_F/t_V} \tag{4a}$$

and

$$\alpha_B = 1 - (1 - \alpha_V)^{t_B/t_V}. \tag{4b}$$

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 volume data 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.

(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 subpixel coverage geometry from slab to slab. Carpenter's A-buffer [1] exemplifies this approach. In the present algorithm, we instead supersample the slab by casting more rays as discussed in the previous 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 just 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.

Implementation and results

The algorithm described in this paper has been implemented in the C language on a Dec 3100 with 24MB 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 areas of high image complexity. For sample regions larger than one pixel 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 earlier. 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 [14, 15].

Figure 3 shows a 300×300 pixel rendering of 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. Computation of voxel colors and opacities took about 1 minute on the Dec 3100. Construction of a pyramid of binary volumes from the array of voxel opacities took another 1 minute. Ray tracing of the preprocessed data to generate figure 3 took 105 seconds. The number of rays cast per pixel ranged from 1 ray per 16 pixels up to 16 rays per pixel with an average rate of slightly less than 1 ray per pixel. For comparison, the volume data without the polygons can be rendered in 32 seconds, and the volume and polygon data can be rendered together but without selective supersampling in 34 seconds.

Figure 4a shows a detail from figure 3. The image is seen to be generally free from aliasing artifacts. The slight striping noticeable in the bony tissue is due to the anisotropic spatial sensitivity of the CT scanner. A visualization of where rays were cast is shown in figure 4b. Each 4×4 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, polygon-polygon intersections, and polygon-volume 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. The data is a $256 \times 256 \times 109$ voxel magnetic resonance (MR) scan of a live human subject. A polygonally defined treatment volume (in purple) and three treatment beams (in red) have been added using the hybrid ray tracer. A portion of the volume has been clipped away and the raw MR values have been mapped onto the exposed surfaces. This is a sample visualization, not a clinical study; the MR data and treatment plan were taken from two different patients.

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 this paper 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 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, the algorithm produces images free from aliasing artifacts. This method underlies a proposed real-time volume rendering workstation supporting the display of both polygon and volume data [13].

Acknowledgements

The author wishes to thank Profs. Henry Fuchs, Stephen M. Pizer, Frederick P. Brooks, Jr., and Turner Whitted of the University of North Carolina 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 scan was provided by the Radiation Oncology Department at North Carolina Memorial Hospital, and the MR scan was provided by Siemens AG and edited by Julian Rosenman. Figure 6 was produced by Victoria Interrante. The texture map used in figure 5 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

[1] Carpenter, L., ‘‘The A-buffer, an Antialiased Hidden Surface Method,’’ *Computer Graphics,* Vol. 18, No. 3, July, 1984, pp. 103-108.

[2] Cook, R.L., ‘‘Stochastic Sampling in Computer Graphics,’’ *ACM Transactions on Graphics,* Vol. 5, No. 1, January, 1986, pp. 51-72.

[3] Dippé, M.A.Z. and Wold, E.H., ‘‘Antialiasing Through Stochastic Sampling,’’ *Computer Graphics,* Vol. 19, No. 3, July, 1985, pp. 69-78.

[4] Drebin, R.A., Carpenter, L., and Hanrahan, P., ‘‘Volume Rendering,’’ *Computer Graphics*, Vol. 22, No. 4, August, 1988, pp. 65-74.

[5] Fuchs, H., Kedem, Z.M., and Uselton, S.P., ‘‘Optimal surface reconstruction from planar contours,’’ *Communications of the ACM,* Vol. 20, No. 10, 1977, pp. 693-702.

[6] Goodsell, D.S., Mian, S., and Olson, A.J., ‘‘Rendering of Volumetric Data in Molecular Systems,’’ *Journal of Molecular Graphics,* Vol. 7, No. 1, March, 1989, pp. 41-47.

[7] Johns, H. and Cunningham, J., *The Physics of Radiology,* Charles C. Thomas, 1983.

[8] Johnson, E.R. and Mosher, C.E.,Jr., ‘‘Integration of Volume Rendering and Geometric Graphics,’’ *State of the Art in Data Visualization,* Siggraph ’89 tutorial, July, 1989.

[9] Kajiya, J.T., ‘‘The Rendering Equation,’’ *Computer Graphics,* Vol. 20, No. 4, August, 1986, pp. 143-150.

[10] Kaufman, A., ‘‘An Algorithm for 3D Scan-Conversion of Polygons,’’ *Proc. Eurographics ’87*, ed. G. Marechal, Elsevier Science Publishers B.V., North-Holland, 1987, pp. 197-208.

[11] Lee, M.E., Redner, R.A., and Uselton, S.P., ‘‘Statistically Optimized Sampling for Distributed Ray Tracing,’’ *Computer Graphics,* Vol. 19, No. 3, July, 1985, pp. 61-67.

[12] Levoy, M., ‘‘Display of Surfaces from Volume Data,’’ *IEEE Computer Graphics and Applications,* Vol. 8, No. 3, May, 1988, pp. 29-37.

[13] Levoy, M., ‘‘Design for a Real-Time High-Quality Volume Rendering Workstation,’’ *Proc. Chapel Hill Workshop on Volume Visualization,* ed. C. Upson, University of North Carolina, 1989, pp. 85-92.

[14] Levoy, M., ‘‘Volume rendering by adaptive refinement,’’ *The Visual Computer,* Vol. 6, No. 1, January, 1990. In press.

[15] Levoy, M., ‘‘Efficient Ray Tracing of Volume Data,’’ *ACM Transactions on Graphics*, Vol. 9, No. 3, July, 1990. In press.

[16] Lorensen, W.E. and Cline, H.E., ‘‘Marching Cubes: A High Resolution 3D Surface Construction Algorithm,’’ *Computer Graphics,* Vol. 21, No. 4, July, 1987, pp. 163-169.

[17] Mitchell, D.P., ‘‘Generating Anti-Aliased Images at Low Sampling Densities,’’ *Computer Graphics,* Vol. 21, No. 4, July, 1987, pp. 65-72.

[18] Painter, J. and Sloan, K., ‘‘Antialiased Ray Tracing by Adaptive Progressive Refinement,’’ *Computer Graphics,* Vol. 23, No. 3, July, 1989, pp. 281-288.

[19] Phong, Bui-Tuong, ‘‘Illumination for Computer-Generated Pictures,’’ *Communications of the ACM.* Vol. 18, No. 6, June, 1975, pp. 311-317.

[20] Porter, T. and Duff, T., ‘‘Compositing Digital Images,’’ *Computer Graphics*, Vol. 18, No. 3, July, 1984, pp. 253-259.

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

[22] Upson, C. and Keeler, M., ‘‘VBUFFER: Visible Volume Rendering,’’ *Computer Graphics*, Vol. 22, No. 4, August 1988, pp. 59-64.

[23] Westover, L., ‘‘Interactive Volume Rendering,’’ *Chapel Hill Workshop on Volume Visualization,* Chapel Hill, North Carolina, May, 1989, pp. 9-16.

[24] Whitted, T., ‘‘An Improved Illumination Model for Shaded Display,’’ *Communications of the ACM*, Vol. 23., No. 6, June, 1980, pp. 343-349.