Volume Rendering by Adaptive Refinement

Marc Levoy

June, 1988
(revised January, 1989)

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

Abstract

Volume rendering is a technique for visualizing sampled scalar functions of three spatial dimensions by computing 2D projections of a colored semi-transparent gel. This paper presents a volume rendering algorithm in which image quality is adaptively refined over time. An initial image is generated by casting a small number of rays into the data, less than one ray per pixel, and interpolating between the resulting colors. Subsequent images are generated by alternately casting more rays and interpolating. The usefulness of these rays is maximized by distributing them according to measures of local image complexity. Examples from two applications are given: molecular graphics and medical imaging.

Key words: Volume rendering, voxel, adaptive refinement, adaptive sampling, ray tracing.

1. Introduction

In this paper, we address the problem of visualizing sampled scalar functions of three spatial dimensions, henceforth referred to as *volume data.* We focus on a relatively new visualization technique called *volume rendering*[1] in which a color and an opacity is assigned to each voxel, and a 2-D 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 methods are its superior image quality and the ability to generate images without explicitly defining surfaces. The technique's principal drawback is its cost. Since all voxels participate in the generation of each image, rendering time grows linearly with the size of the dataset.

The complexity of volume rendering can be reduced by taking advantage of coherence present in the sample array. Methods based on hierarchical spatial enumeration and adaptive termination of ray tracing are presented in (Levoy 1988b). Although the amount of time saved depends on the data and the viewing angle, savings of more than an order of magnitude have been observed. Despite these improvements, the time required to render a 256 x 256 x 256 voxel dataset using currently available workstation technology is still measured in tens of seconds or minutes. To be useful in an interactive setting, reductions of at least another order of magnitude are desirable.

[1] 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 (1989) 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.

This paper presents an image-order (i.e. ray tracing) volume rendering algorithm in which image quality is adaptively refined over time. Bergman et al. (1986) have written an excellent survey of refinement techniques for polygonal environments. These techniques have three principal characteristics: they distribute work according to where it makes the most difference, they form intermediate images from partial information, and they minimize the amount of work discarded after formation of each image.

Conventional ray tracing algorithms are not well suited for use in an adaptive refinement system. Although numerous techniques exist for distributing rays according to local image complexity (Whitted 1980; Lee et al. 1985; Dippe and Wold 1985; Cook 1986; Kajiya 1986), these techniques cast large numbers of rays per pixel, resulting in long image generation times. They also complete the rendering of one pixel before moving on to the next, precluding the display of images from partial information. These limitations are necessary when rendering analytically defined objects; undersampling of such scenes produces objectionable aliasing artifacts.

Volume data, unlike analytically defined objects, is assumed bandlimited. Additional blurring is often introduced during acquisition as in the case of computerized tomography (CT) and electron density maps obtained from X-ray diffraction data (Herman 1980; Glusker 1985). Certain amorphous phenomena occurring in astronomy and physics are even more bandlimited (Upson 1986). This property allows us to generate subjectively acceptable images from partial information and therefore to display images as they are being refined.

In the present algorithm, an initial image is generated by casting a uniform but sparse grid of rays into the volume data, interpolating between the resulting colors, and resampling at the display resolution. A sampling rate of one ray per four pixels, corresponding to a data sampling rate of one ray per four voxels, is typical. Subsequent images are generated by discarding interpolated colors, casting more rays, and repeating the interpolation and resampling steps. Recursive subdivision based on color differences is used to concentrate these additional rays in regions of high image complexity, and recursive bi-linear interpolation is used to form images from the resulting non-uniform array of colors. The approach is similar to that described by Whitted, but extended to allow sampling rates of less than one ray per pixel and modified to provide a mechanism for progressive refinement.

The cost of computing each image in a refinement sequence is equal to the sum of the costs of recursive subdivision, ray tracing, and recursive interpolation. Of these, only ray tracing would be required if intermediate images were not displayed. In the current implementation, this cost dominates the others by two orders of magnitude. The amount of work expended displaying intermediate images that does not also contribute toward the final image is therefore minimal.

Using this algorithm, crude images of many datasets can be obtained in few seconds. Gradually better images are obtained at intervals of a few seconds each, culminating in a high quality image in less than a minute.

2. Description of algorithm

Figure 1 summarizes the flow of data in the algorithm. It begins with a 3D array of scalar-valued voxels. The array is shaded and classified using techniques described in (Levoy 1988a) to yield a color and an opacity for each voxel. Parallel viewing rays are then traced into the array from an observer position as shown in figure 2. The image plane is divided into square sample regions measuring ω_{max} pixels on a side, and rays are cast from the four corner pixels of each region. Each ray is divided into equal-size sample intervals, and a color and opacity is computed at the center of each interval by tri-linearly interpolating from the colors and opacities of the nearest eight voxels. The resampled colors and opacities are then composited from front to back to yield a color for the ray.

If the range of colors returned by the four rays in a sample region is less than some ε, no further processing is performed on the region. Otherwise, the region is divided into four subregions and more rays are cast. Subdivision continues until the range of colors falls below ε or the size of the region reaches some ω_{min} where $1 \le \omega_{min} \le \omega_{max}$.

When all sample regions have been processed, an image is formed by interpolating between the available colors and resampling. To insure continuity despite the non-uniform distribution of colors, a recursive technique similar to the algorithm employed during ray tracing is used. The image plane is again divided into square regions measuring ω_{max} pixels on a side. Pixels are interpolated at the midpoints of the four sides and at the center of each region. The region is then divided into four subregions and the process is repeated. Subdivision continues until the region contains a single pixel.

When all pixels have been filled in, the resulting image is displayed. To continue the refinement process, the image is cleared of all interpolated colors, the level of detail is raised by decreasing ω_{min}, ω_{max}, or ε, the image plane is again divided into sample regions, and ray tracing begins anew.

Since the colors associated with traced rays are retained throughout the refinement process, some means is needed to avoid tracing the same rays repeatedly. In the current algorithm, this is implemented by maintaining a flag *F* for each pixel. Rays are traced only from pixels whose flags are clear. Once a ray has been traced from a given pixel, its flag is set. It is worth noting that the retension of colors has the useful side effect of allowing rays to be shared by adjacent sample regions. This reduces the number of rays that must be traced, thereby improving the efficiency of the algorithm.

Some means is also needed to distinguish pixels whose colors are interpolated from pixels whose colors are computed by ray tracing. This is implemented in the current algorithm by maintaining another flag *G* for each pixel. Colors are interpolated only at pixels whose *F* and *G* flags are both clear. Once a pixel has been interpolated, its *G* flag is set. After display of each intermediate image, pixels whose *G* flag is set are cleared before additional rays are cast.

The refinement process alternates between casting rays and forming images, terminating when $\omega_{max} = 1$, when the user changes a rendering parameter, or when the observer moves.

3. Implementation and results

To demonstrate the performance of this algorithm, two case studies are presented. The first is a 123 x 123 x 123 voxel portion of an electron density map of cytochrome B5. Using methods described in (Levoy 1988a), an isovalue surface was selected for display. Using the algorithm described in section 2 of this paper, a four-frame adaptive refinement sequence was then generated. The resulting images are shown in figure 3 with the sequence running from top-left to bottom-right. Each image measures 256 x 256 pixels.

Performance statistics for this sequence are given in table 1. For each image, the table gives values for the three level-of-detail parameters, incremental and total ray counts, and incremental and total elapsed times. Timings are for a Sun 4/280.

A visualization of the *F* array showing where rays were cast is given in figure 4. Each white pixel in this figure corresponds to a single ray. Thus, any pixel in figure 3 whose corresponding pixel in figure 4 is white was computed by ray tracing, whereas any pixel in figure 3 whose figure 4 pixel is black was computed by interpolation. As expected, ray densities are highest along surface silhouettes where color differences are highest, and the overall number of rays increases from the first to the last frame of the sequence. Since $\omega_{max} = 1$ in the last frame, rays are cast from every pixel, and the *F* array is completely white.

The second dataset is a computerized tomography (CT) study of a human head and was acquired as 113 slices of 256 x 256 samples each. Using methods described in (Levoy 1988a), the bone surface was selected for display. A four-frame adaptive refinement sequence was then generated, the first frame of which is shown in figure 5 and the last frame of which is shown in figure 6. Each image measures 512 x 512 pixels. Performance statistics for this sequence are summarized in table 2.

4. Conclusions

A technique for rapidly visualizing sampled scalar functions of three spatial dimensions has been described. The algorithm casts fewer rays than conventional ray tracers, reflecting the bandlimited nature of volume data, and casts them in an order that allows display of intermediate images.

In this study, values for the three level-of-detail parameters ω_{max}, ω_{min}, and ε were selected manually. Generally speaking, high values of ω_{max} cause features to be missed, high values of ε cause features to be ignored even if they are not missed, and high values of ω_{min} causes features to be poorly resolved even if they are neither missed nor ignored. Inappropriate values for these parameters cause suboptimal presentation of the data as well as unequal intervals between successive frames in refinement sequences. Algorithms are needed that automatically select an optimum sequence of values based on the characteristics of a particular dataset.

5. 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 and to my reviewers for several useful suggestions concerning the manuscript. The electron density map was obtained from Jane and Dave Richardson of Duke University, and the CT scan was provided by the Radiation Oncology Department at North Carolina Memorial Hospital. This work was supported by ONR grant N00014-86-K-0680, NIH grant R01-CA39060, and IBM.

6. References

Bergman, L., Fuchs, H., Grant, E., and Spach, S. (1986), ''Image Rendering by Adaptive Refinement,'' *Computer Graphics,* Vol. 20, No. 4, pp. 29-37.

Cook, R.L. (1986), ''Stochastic Sampling in Computer Graphics,'' *ACM Transactions on Graphics,* Vol. 5, No. 1, pp. 51-72.

Dippe, M.A.Z. and Wold, E.H. (1985), ''Antialiasing Through Stochastic Sampling,'' *Computer Graphics,* Vol. 19, No. 3, pp. 69-78.

Drebin, R.A., Carpenter, L., and Hanrahan, P. (1988), ''Volume Rendering,'' *Computer Graphics* Vol. 22, No. 4, pp. 65-74.

Glusker, P.J. and Trueblood, K.N. (1985), *Crystal Structure Analysis,* Oxford University Press, Oxford.

Herman, G.T. (1980), *Image Reconstruction from Projections,* Academic Press, New York.

Kajiya, J.T. (1986), ''The Rendering Equation,'' *Computer Graphics,* Vol. 20, No. 4, pp. 143-150.

Lee, M.E., Redner, R.A., and Uselton, S.P. (1985), ''Statistically Optimized Sampling for Distributed Ray Tracing,'' *Computer Graphics,* Vol. 19, No. 3, pp. 61-67.

Levoy, M. (1988a), ‘‘Display of Surfaces from Volume Data,’’ *IEEE Computer Graphics and Applications*, Vol. 8, No. 3, pp. 29-37.

Levoy, M. (1988b), ‘‘Efficient Ray Tracing of Volume Data,’’ Technical Report 88-029, Computer Science Department, University of North Carolina at Chapel Hill.

Reynolds, A. (1989), Letter to the editor of *IEEE Computer Graphics and Applications*, Vol. 9, No. 2.

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

Upson, C. (1986), ‘‘The Visual Simulation of Amorphous Phenomena,’’ *The Visual Computer*, Vol. 2, pp. 321-326.

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

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