Design for a Real-Time High-Quality Volume Rendering Workstation

Marc Levoy

Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175

ABSTRACT

A design is presented for a workstation capable of rendering arbitrary mixtures of analytically defined geometry and sampled scalar fields of three spatial dimensions in real-time or near real-time. The design is based on *volumetric compositing*, a relatively new approach to visualizing scalar fields by computing 2D projections of a colored semi-transparent volume. Geometric primitives are added by filtering and 3D scan-converting them into the dataset prior to rendering. Speedups totaling four orders of magnitude over published volumetric compositing techniques are obtained through a combination of algorithmic improvements and hardware implementation on Pixel-planes 5, a massively parallel raster display engine incorporating custom logic-enhanced memory chips. A preliminary design is also presented for a two-handed volume-of-interest data exploration tool.

KEYWORDS: Visualization, voxel graphics, volume rendering, volumetric compositing, parallel architecture, 3D scan-conversion, adaptive refinement, ray tracing.

INTRODUCTION

Voxel-based techniques for displaying sampled scalar fields of three spatial dimensions, also known as *volume rendering* techniques, have been in use for about 10 years and have been applied in a variety of scientific, medical, and engineering disciplines. Kaufman has written an excellent survey of architectures designed for rendering voxel data [Kaufman 86a], including his own CUBE (CUbic frame Buffer) system currently under development [Kaufman 88a]. Most of the machines he surveys start by thresholding the incoming data to yield a set of 0-voxels and a set of 1-voxels. While reducing memory requirements and image generation time, this binary classification procedure invariably leads to false positives (spurious objects) or false negatives (erroneous holes in objects), particularly in the presence of small or weakly defined features.

To avoid these problems, researchers have begun exploring *volumetric compositing*, a variant of volume rendering in which a color and an opacity is assigned to each voxel, and a 2D projection of the resulting colored semi-transparent volume is computed [Levoy 88a, Drebin 88, Sabella 88, Upson 88]. The key improvement offered by volumetric compositing is that it eliminates the necessity of making a binary classification of the data, thus providing a mechanism for display of poorly defined features. This improved image quality comes, however, at a substantial cost in image generation time. The fastest volumetric compositing system at this writing is probably the Pixar Image Computer. Using a four channel SIMD processor [Levinthal 84] and Pixar's ChapVolumes software package, the Image Computer can generate high-quality images in tens of seconds or minutes, depending on the size of the dataset.

I propose a volume rendering workstation based on the image-order (i.e. ray tracing) volumetric compositing algorithm described in [Levoy 88a]. Since that paper was written, I have obtained speedups of two orders of magnitude for many datasets by taking advantage of various forms of coherence. Three such optimizations are summarized here and described in detail elsewhere [Levoy 88b, Levoy 89].

The first optimization is based on the observation that many datasets contain coherent regions of empty voxels. A voxel is defined as empty if its opacity is zero. In [Levoy 88b], I encode this coherence using a hierarchical spatial enumeration represented by a pyramid of binary volumes. The pyramid is used to efficiently compute intersections between viewing rays and regions of interest in the 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. In [Levoy 88b], I stop tracing each ray when its opacity reaches a user-selected threshold level. This allows me to ignore much of the data, particularly for scenes composed mainly of opaque surfaces.

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, reported in [Levoy 89], 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. Images are formed from the resulting non-uniform array of sample colors by interpolation and resampling at the display resolution. Alternatively, successively more refined images can be generated at evenly spaced intervals of time by casting more rays, adding the resulting colors to the sample array, and repeating the interpolation and resampling steps.

In the present paper, I describe techniques for obtaining speedups of another two orders of magnitude by optimizing the algorithm for several common types of animation sequences and implementing it on Pixel-planes 5 [Fuchs 89], a raster display engine under development at the University of North Carolina and scheduled for completion during the summer of 1989. Ray tracing rendering algorithms have been developed for a number of parallel machine architectures including the Connection Machine [Delaney 88] and the MPP [Dorband 87], but none of these implementations support the display of volume data. Although Pixel-planes 5 was not explicitly designed for volume rendering, its flexibility makes it surprising well suited to the task. Such a workstation would provide update rates of between 1 and 20 frames per second for datasets of useful size and complexity.

Many scientific problems require sampled functions and analytically defined objects to appear together in a single visualization. Kaufman (with others) has described algorithms for 3D scan-conversion of polygons [Kaufman 87b], polyhedra [Kaufman 86b], and cubic parametric curves, surfaces, and volumes [Kaufman 87a], and proposes to implement these techniques on his CUBE architecture. His use of a binary voxel representation should give rise to artifacts in the generated images, although he has apparently worked out a solution to this problem [Kaufman 88b]. A number of other SIMD and MIMD architectures are suitable platforms for implementing parallel 3D scan-conversion, but no implementations have yet appeared in the literature.

I have developed and described elsewhere two methods for rendering mixtures of 3D scalar fields and polygonally defined objects [Levoy88c]. The first method employs a hybrid ray tracer. Rays are cast through the ensemble of volumetric and geometric data, and samples of each are drawn and composited in depth-sorted order. To avoid errors in visibility, volumetric samples lying immediately in front of and behind polygons require special treatment. To avoid aliasing of polygonal edges, adaptive supersampling is employed. The second method involves 3D scan-conversion, but geometric primitives are filtered prior to sampling. and the resulting ensemble is rendered using volumetric compositing. No particular care need be taken in the vicinity of sampled geometry, and no supersampling is required. If polygons are sufficiently bandlimited prior to sampling, this approach produces images free from aliasing artifacts.

The proposed workstation would utilize my 3D scan-conversion method. In the present paper, I describe an implementation on Pixel-planes 5 that promises a speedup of more than three orders of magnitude over the sequential algorithm, yielding scan-conversion rates of about 1000 arbitrary-sized polygons per second.

User interfaces for existing volume rendering systems are constrained by the inability to generate images in real-time. Feedback during selection of rendering parameters is usually provided by *meta-visualizations* such as 2D plots of color and opacity versus input value, wire-frame representations of viewing frustrums and motion paths, etc. These ancillary displays complicate the user interface and alienate prospective users. Teaming a computer technician with each user is not a satisfactory alternative, particularly in the medical field. Such intermediaries inhibit the frequent and informal experimentation that leads to insight.

Since my proposed workstation would perform volume rendering in real-time or near-real-time, these meta-visualizations could be omitted or relegated to a supporting role. Sequences of volume rendered images would serve as feedback to the user of changes made in rendering parameters. A two-handed volume-of-interest data exploration tool exemplifying this approach is presented in the present paper.

OPTIMIZATION FOR ANIMATION SEQUENCES

Figure 1 summarizes the volume rendering algorithm that would run on the proposed workstation. It begins with a 3D array of scalar-valued voxels. We first classify and shade the array to yield a color and an opacity for each voxel as described in [Levoy 88a]. Parallel viewing rays are then traced into the array from an observer position. 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.

This algorithm consists of several steps: shading, classification, ray tracing, resampling, and compositing. Each step is controlled by user-selectable parameters and produces as output a sampled scalar or vector-valued volume. For animation sequences in which only a subset of the controlling parameters change from frame to frame, these intermediate results can be stored in arrays, and only those calculations whose parameters change need be repeated on each frame.

A common type of sequence is one in which the object and light sources are fixed and the observer moves. In this case, the color and opacity of each voxel can be held invariant, substantially reducing image generation time. This simple optimization is discussed in [Levoy 88a] and is implicit in the approach of [Drebin 88]. The method produces incorrect specular highlights, but users seldom notice the error. This method also hampers users' ability to distinguish changes in surface orientation from changes in surface albedo, a more serious drawback. If the light sources move relative to the object, voxel colors must be recomputed on each frame, but voxel opacities are still invariant.

Another common type of sequence is one in which voxel colors are held invariant and voxel opacities change. For example, users frequently ask for some means of highlighting and interactively moving a 3-D region of interest. The notion of treating the voxels lying inside a defined region differently from the rest of a dataset has been explored extensively in [Hoehne88b]. In the context of volume rendering, one way to highlight such a region is to increase the opacity of voxels lying inside in the region and to decrease the opacity of voxels lying outside the region. In some cases (such as figure 6) it is preferable to perform the inverse transformation, decreasing the opacities of voxels lying inside the region of interest.

For certain special types of sequences, additional optimizations may be possible. For example, if the light sources move relative to the object but the observer stays motionless, the depth along each viewing ray at which the first non-empty voxel is encountered does not change. This depth can be recorded in an array during generation of the first frame in a sequence and used to speed generation of subsequent frames. Hoehne reports success using a similar depth buffer in his own work [Hoehne88a]. If the shading model includes multiple light sources only one of which is moving, the contribution made by the stationary sources can be pre-computed and added on each frame to the contribution computed for the moving source (assuming that multiple scattering effects are ignored).

As a final note, the local gradient vector at each voxel is is a function only of the input data and does not depend on any of the controlling parameters. If this vector is precomputed for all voxels, calculation of new opacities following a change in classification parameters entails only generation of a new lookup table followed by one table reference per voxel.

IMPLEMENTATION ON PIXEL-PLANES 5

Pixel-planes is a raster graphics engine incorporating custom logic-enhanced memory chips and designed for high-speed rendering of 3D objects and scenes. Versions of this machine have been running in our laboratory for over five years and have been the subject of numerous published papers [Fuchs 81, Fuchs 82, Fuchs 85]. Pixel-planes 5 is a new design that promises to have unprecedented power and flexibility [Fuchs 89]. Briefly, it will consist of 32 independently programmable 20-MFLOP graphics processors (GP's), 1/4 million pixel processors (PP's) organized into 16 independently programmable renderers, a 512 x 512 pixel color frame buffer, and a 640 Mb/sec ring network. The machine is expected to become operational sometime during the summer of 1989.

Figure 2 summarizes the proposed implementation of my volumetric compositing algorithm on Pixel-planes 5. The shading and classification calculations for all voxels would be performed in the pixel processors (PP's). The current hardware plans call for 4K bits of backing store on each processor. Using this configuration, a processor can hold 8 bytes of information for each of 64 voxels, enabling us to render a 256 x 256 x 256 voxel dataset. Initially, each voxel's scalar value and gradient, which are invariants of the volume rendering process, are loaded into the processors. Shading and calculations are then performed in parallel using the 1-bit arithmetic units of the processors to yield a color and an opacity for each voxel. The time required to apply a color Phong shading model at a single voxel is estimated to be about 1 msec. Since each processor is assigned 64 voxels, the time required to classify and shade the entire dataset would be about 64 msec.

The ray tracing required to generate an image from a set of colors and opacities would be performed in the graphics processors (GP's). Each is assigned a set of rays, which it traces through object space. The GP's request sets of voxels from the PP's as necessary, perform tri-linear interpolation and compositing, and transmit the resulting pixel colors to the frame buffer for display. Since each GP is assigned several rays, it can context switch as necessary to keep busy while waiting for requested voxels to be transferred across the network.

The success of this approach depends on reducing the number of voxels that must be processed along each ray and by implication the number of voxels that must be moved across the network. Three strategies are planned. First, a hierarchical spatial enumeration of the volumetric dataset [Levoy 88b] would be installed in each GP. This data structure tells the GP which voxels are non-empty (non-transparent) and hence worth requesting from the PP's. Second, adaptive sampling [Levoy 89] would be used to reduce the number of rays required to generate an initial image. Last, all voxels received by a GP would be retained in a 16Mb local cache. If the observer does not move during generation of the initial image, the cached voxels would be used to drive successive refinement of the image. If the observer moves, many of the voxels required to generate the next frame might already reside in the cache, depending on how far the observer moves between frames.

The frame rate that could be expected from this system depends on what parameters change from frame to frame. Preliminary estimates suggest that for changes in observer position alone, sequences of slightly crude images (of the quality of figure 3) could be generated at 10 frames per second, and fully refined images (similar to figure 4) could be generated in about 1 second. For changes in shading or changes in classification that do not invalidate the hierarchical enumeration, the system should produce 20 crude or 2 refined images per second. This includes highlighting and interactively moving a region of interest by scaling voxel opacities as already described. If the user changes the classification mapping in such a way as to alter the set of non-empty voxels, the hierarchical enumeration must be recomputed. This operation would take several seconds.

PARALLEL 3D SCAN-CONVERSION

The technique I propose for fast 3D scan-conversion of polygons is an extension to three dimensions of the progressive refinement method currently used to display antialiased polygons on Pixel-planes 4 [Fuchs 85]. A group of voxels is assigned to each pixel processor (PP) as described in the previous section, the geometry of each polygon is broadcast across the ring network, and each PP is directed to determine whether the polygon falls within the voxels assigned to it (treating voxels as abutting cubes). To eliminate aliasing artifacts, the spatial location of the polygon is moved through a set of subvoxel-sized translations and re-rendered. When this process is complete, the number of times the polygon fell into each voxel is divided by the number of times the polygon was rendered to yield a fractional volumetric coverage for the voxel.

To compute a Gouraud-shaded color and opacity for each voxel covered by a polygon, linear expressions representing the polygon's red, green, blue, and opacity components are broadcast across the network and evaluated in the PP's on behalf of each of their assigned voxels (now treating voxels as points in space as described in [Fuchs 85]. The opacities are then scaled by the volumetric coverage fractions, and the resulting colors and scaled opacities are combined with the colors and opacities already present in each voxel using volume matting operators [Levoy 88c].

Using a fully configured Pixel-planes 5 system, 3D scan-conversion rates of about 1000 arbitrary-sized polygons per second should be possible. Pixel-planes 5 also includes novel circuitry for fast evaluation of quadratic expressions [Fuchs 89]. This hardware would allow efficient scan-conversion of spheres and other curved surfaces as well as polygons.

TWO-HANDED VOLUME-OF-INTEREST TOOL

One of the user interface paradigms I am considering for my proposed volume rendering workstation consists of providing the user with two six-degree-of-freedom input devices (such as the Polhemus Navigation Sciences's 3SPACE tracker), assigning one of these *bats* (3D plural of mouse, term suggested in [Ware 88]) to control the position and orientation of a *volume-of-interest,* or 3D region of interest, and assigning the second bat to control the position and direction of a point light source. Using the proposed workstation, voxels inside (or outside) the volume-of-interest would be highlighted by scaling voxel opacities as described earlier, and user feedback would be provided by computing 20 crude volume rendered images per second on the proposed workstation.

One obvious addition to the above paradigm is interactive control over observer position. Both hands are already in use, but head position and orientation are still available as inputs to the system. As it happens, a head-mounted raster display system is already being developed in our laboratory and plans are in place for routing the output of Pixel-planes 5 into two liquid crystal video displays mounted on a helmet and visible to the user through half-silvered mirrors. The position and orientation information returned by a third Polhemus tracker affixed to the helmet would allow a user to move around and through a 3D scene as if it were actually in the room with them. Since Pixel-planes 5 is not expected to be capable of rendering a moving 256 x 256 x 256 voxel dataset at 30 frames per second as required for a head-mounted display system, either the size of the dataset must be reduced, or a speedup of another factor of 3 must be obtained through hardware or software speedups.

SAMPLE IMAGES

Figures 3 and 4 illustrate the trade-off between image quality and image generation time that could be expected from my proposed workstation. These two 512 x 512 images represent two frames from a progressively refined volume rendering of a 256 x 256 x 123 voxel magnetic resonance (MR) study of a human head. Tissues overlying the cortical surface were removed by editing the dataset manually. The apparent mottling of the facial surface is due to noise in the acquired data. Assuming that only the observer position and orientation change from frame to frame, figure 5 can be generated in 6 seconds on a Sun 4/280, and figure 6 requires an additional 57 seconds. Estimated timings for the proposed workstation are 1/10 second to generate figure 3 and an additional 9/10 second to generate figure 4. If the observer were stationary and only the light source or a 3D region of interest were changing, these timings would instead be 1/20 second and 9/20 second.

Figure 5 is a fully refined volume rendering of a 256 x 256 x 113 voxel computerized tomography (CT) study of a human head to which three mutually perpendicular polygons have been added using 3D scan-conversion with anti-aliasing. The cost of scan-converting a polygon on a sequential machine is proportional to its surface area. The large polygons shown here were scan-converted in about 1 second each on a Sun 4/280. Estimated timings for the proposed workstation are 1 msec per polygon.

Figure 6 illustrates one possible use of a 3D region of interest. In this example, the dataset used in figure 5 has been rendered in color to show both bone and soft tissue, and a polygonally defined tumor (in purple) and radiation treatment beam (in blue) have been added. The polygons were rendered using a hybrid ray tracer capable of handling both geometric and volumetric data [Levoy 88c], not the 3D scan-conversion technique discussed in this paper. To highlight the 3D relationships between the various objects, the opacities of all voxels inside a cube-shaped region above the right eye have been scaled down to nearly zero. To avoid aliasing artifacts, the transition from scaled to unscaled opacities is spread over a distance of several voxels. To further improve the visualization, the colors of all voxels in this transition zone have been recomputed as if the region of interest contained air rather than tissue. The effect of this extra step is to cap off anatomical structures where they enter the region of interest. The rescaling of opacities and recomputation of colors along region boundaries do not significantly slow down image generation since they are performed in parallel on the proposed workstation. Image generation times are therefore expected to be comparable to those given for figures 3 and 4.

CONCLUSIONS

One of the principal lessons to be learned from the graphics literature is the importance of real-time motion to the comprehension of complex 3D scenes. This is particularly true for volume data, whose features lack the regularity of geometrically defined objects. The computer graphics literature also tells us that people are sensitive to and critical of aliasing artifacts in computer-generated images. In addition to their aesthetic impact, these artifacts often materially impair the ability of users to understand their data. This paper proposes a workstation for rendering volume data that addresses both of these issues. By combining volumetric compositing, fast image generation, and a simple user interface, it is hoped that routine use of volume rendering in the scientific and medical communities will become both feasible and fruitful.

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 Poulton, David Ellsworth, and Brice Tebbs for their help refining the Pixel-planes implementation. The MR study was provided by Siemens AG and edited by Juiqi Tang. 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.

REFERENCES

Delaney, H.C. [1988], "Ray Tracing on a Connection Machine," *Proceedings of the 1988 ACM/INRIA International Conference on Supercomputing*, St. Malo, France, July, 1988, ACM Press, pp. 659-667.

Dorband, J.E. [1987], "3D Graphic Generation on the MPP," *Processings of the 1987 ICS International Conference on Supercomputing*, International Supercomputing Institute, Vol. 1, pp. 305-309.

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

Fuchs, H. and Poulton, J. [1981], "Pixel-planes: A VLSI-Oriented Design for a Raster Graphics Engine," *VLSI Design*, Vol. 2, No. 3, 3rd Quarter, 1981, pp. 20-28.

Fuchs, H., Poulton, J., Paeth, A., and Bell, A. [1982], "Developing Pixel Planes, A Smart Memory-Based Raster Graphics System," *Proceedings of the 1982 MIT Conference on Advanced Research in VLSI*, Dedham, MA, Artech House, pp. 137-146.

Fuchs, H., Goldfeather, J., Hultquist, J.P., Spach, S., Austin, J., Brooks, F.P., Eyles, J., and Poulton, J. [1985], "Fast Spheres, Textures, Transparencies, and Image Enhancements in Pixel-Planes," *Computer Graphics*, Vol. 19, No. 3, July, 1985, pp. 111-120.

Fuchs, H., Poulton, J., Eyles, J., Greer, T., Goldfeather, J., Ellsworth, D., Molnar, S., Turk, G., Tebbs, B., and Israel, L. [1989], "A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," Technical Report 89-005, Computer Science Department, University of North Carolina at Chapel Hill, January, 1989 (submitted for publication).

Hoehne, K.-H. [1988], personal communication, February, 1988.

Hoehne, K.-H., Bomans, M., Tiede, U., and Riemer, M. [1988], "Display of Multiple 3D-Objects Using the Generalized Voxel-Model," *Medical Imaging II*, Proc. SPIE Vol. 914, 1988, pp. 850-854.

Kaufman, A. [1986], "Voxel-Based Architectures for Three-Dimensional Graphics," *Proceedings of the IFIP 10th World Computer Congress*, Dublin, Ireland, September, 1986, pp. 361-366.

Kaufman, A. and Shimony, E. [1986], "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. [1987], "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. [1987], "An Algorithm for 3D Scan-Conversion of Polygons," *Proc. EUROGRAPHICS '87*, Amsterdam, Netherlands, August, 1987, pp. 197-208.

Kaufman, A. and Bakalash, R. [1988], "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. [1988], personal communication, December, 1988.

Levinthal, A. and Porter, T. [1984], "Chap - A SIMD Graphics Processor," *Computer Graphics,* Vol. 18, No. 3, July, 1984, pp. 77-82.

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

Levoy, M. [1988], "Efficient Ray Tracing of Volume Data," Technical Report 88-029, Computer Science Department, University of North Carolina at Chapel Hill, June, 1988 (submitted for publication).

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

Levoy, M. [1989], "Volume Rendering by Adaptive Refinement," *The Visual Computer,* Vol. 5, No. 3, June, 1989 (to appear).

Robertson, P.K. and O'Callaghan, J.F., "The Application of Scene Synthesis Techniques to the Display of Multidimensional Image Data," *ACM Transactions on Graphics,* Vol. 4, No. 4, October, 1985, pp. 247-275.

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

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

Ware, C. and Jessome, D.R. [1988], "Using the Bat: A Six-Dimensional Mouse for Object Placement," *IEEE Computer Graphics and Applications,* Vol. 8, No. 6, November, 1988, pp. 65-70.

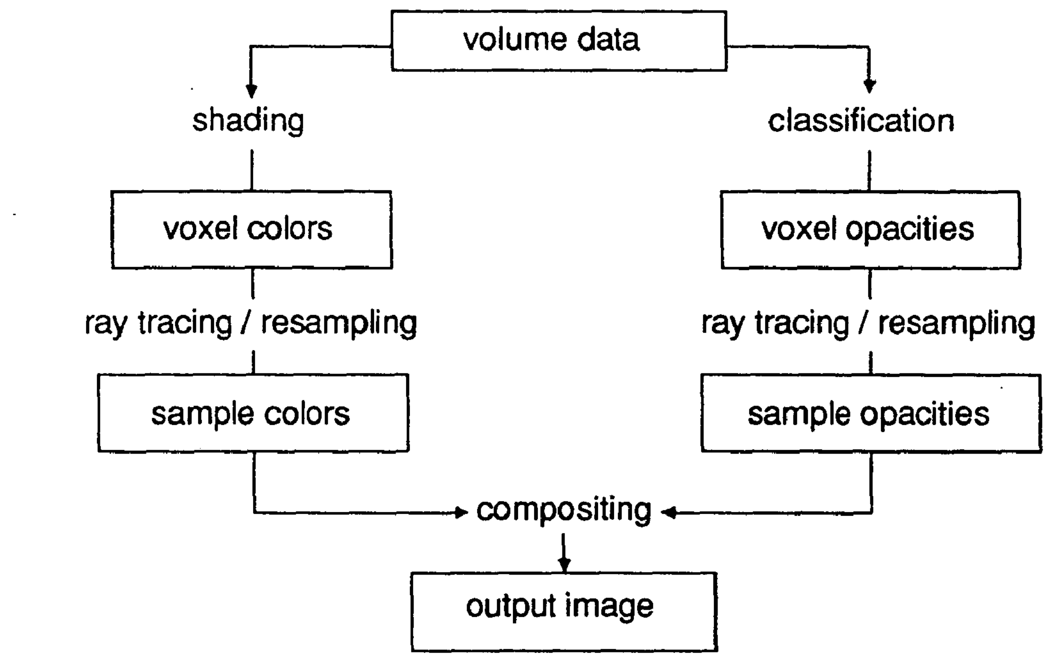


Figure 1 - Overview of volume rendering algorithm

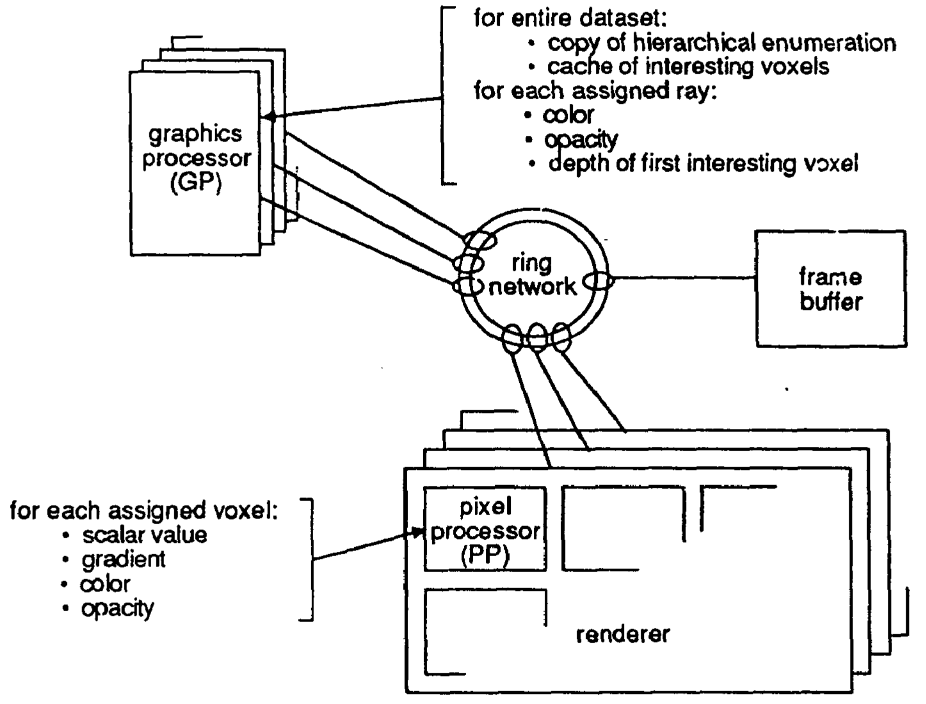


Figure 2 - Implementation on Pixel-planes 5

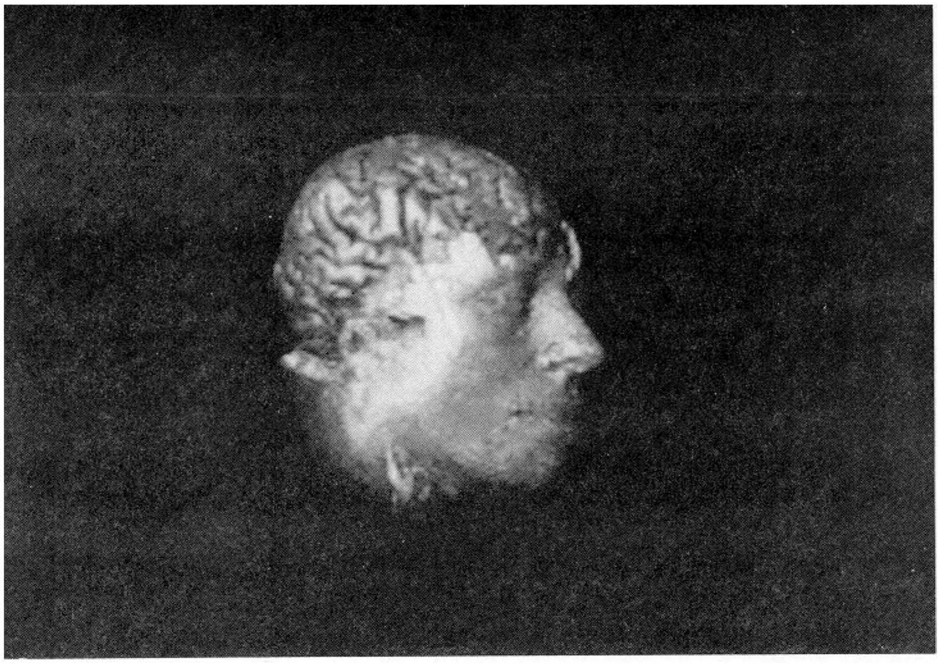


Figure 3 - Adaptively refined volume rendering of human head, estimated image generation time on Pixel-planes = 1/10 second

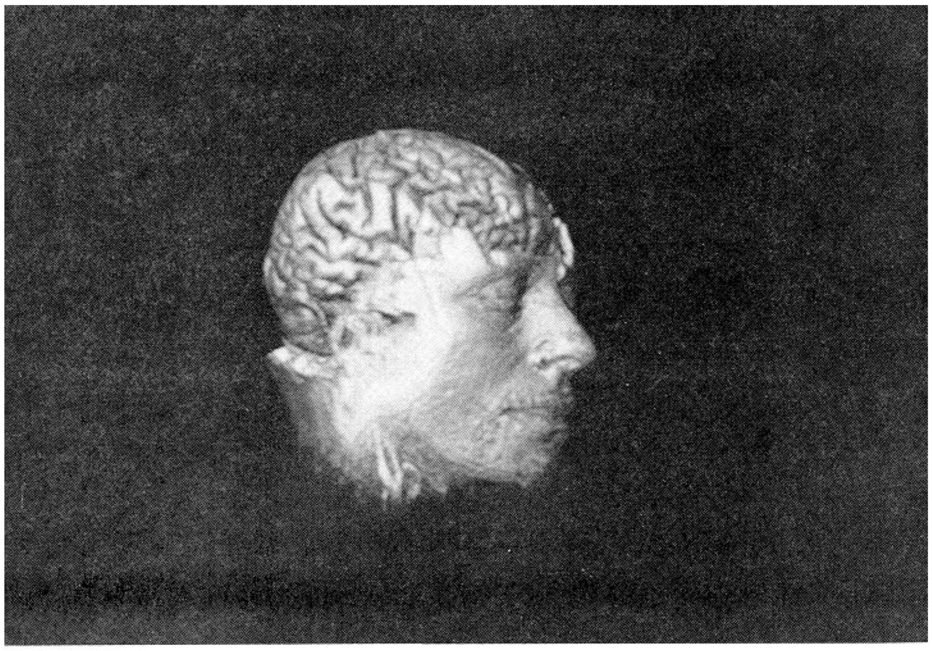
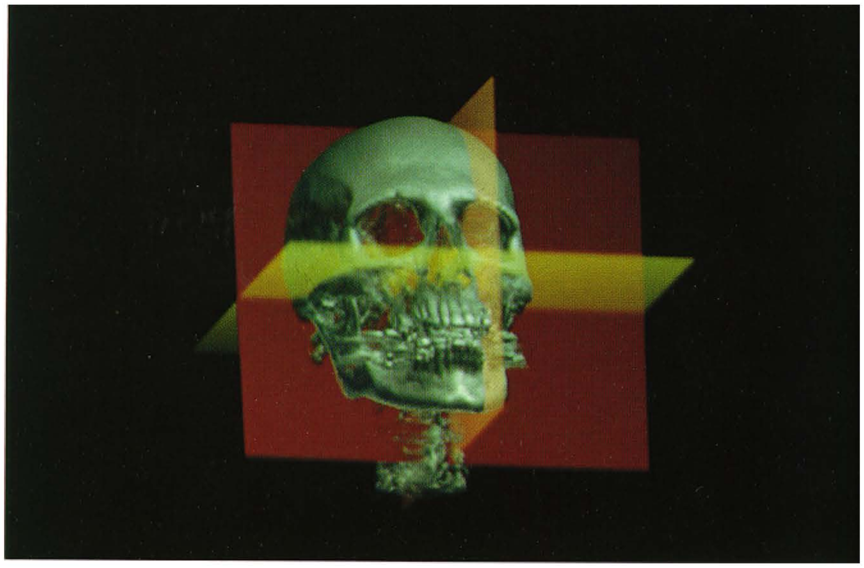
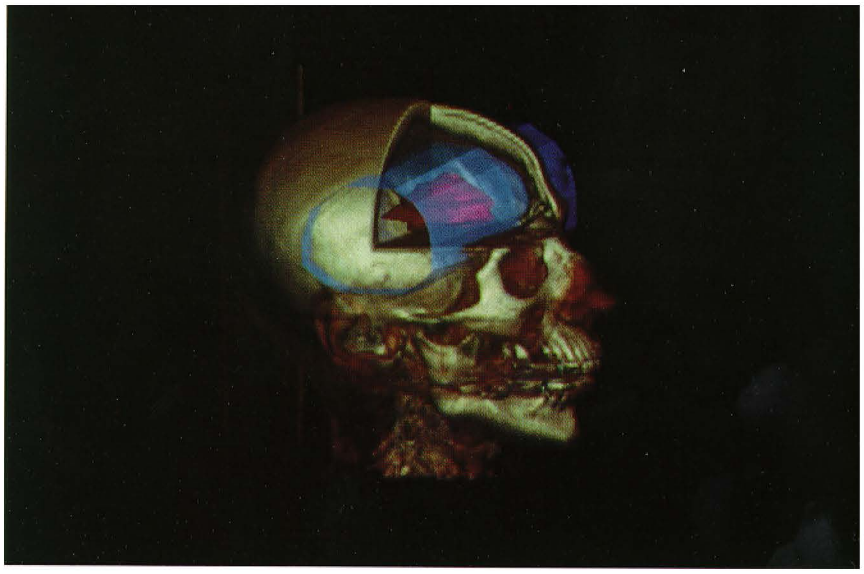


Figure 4 - Adaptively refined volume rendering of human head, estimated image generation time on Pixel-planes = 1 second



Levoy. Figure 5 - Volume rendering of human head with embedded polygons, estimated image generation time on Pixel-planes = 1 msec per polygon.



Levoy. Figure 6 - Volume rendering of human head with 3D region of interest, estimated image generation time on Pixel-planes = 1 second.