Achieving Direct Volume Visualization with Interactive Semantic Region Selection

Ulrich Neumann[1], *Terry S. Yoo*[1], *Henry Fuchs*[1,2], *Stephen M. Pizer*[1,2], *Tim Cullip*[2], *John Rhoades*[1], *Ross Whitaker*[1]

[1] Department of Computer Science
University of North Carolina
Chapel Hill, NC 27599-3175, USA

[2] Department of Radiation Oncology
School of Medicine
University of North Carolina
Chapel Hill, NC 27599-7512, USA

Abstract

Interactive direct visualization of 3D data requires fast update rates and the ability to extract regions of interest from the surrounding data. We have implemented Multi-Valued Classification volume renderers on a general purpose multiprocessor graphics platform (Pixel-Planes 5) that are faster than any yet reported (2 -15 frames per second). We have joined one of them with a sophisticated system for interactive semantic region selection, to our knowledge the first such combination. The result is a system that provides fast visualization of complex information in three dimensional data.

Parallel volume rendering yields rates that make interactive control of image viewing possible for the first time. We have achieved rates as high as 15 frames per second by trading some function for speed, while volume rendering with a full complement of ramp classification capabilities is performed at 1.4 frames per second. These speeds have made the combination of region selection with volume rendering practical for the first time. Semantic driven selection, rather than geometric clipping, has proven to be a natural means of interacting with 3D data. Internal organs in medical data or other regions of interest can be built from preprocessed region primitives. We have applied the resulting combined system to real 3D medical data with encouraging results. The ideas presented are not just limited to our platform, but can be generalized to include most parallel architectures. We present lessons learned from writing fast volume renderers and applying of image processing techniques to viewing volumetric data.

1. Introduction

Visualization is more than just producing complex images of higher dimensional data, it is also the comprehension of the information within the data. Traditional approaches to representing 3D information using volume rendering have often been hampered by a lack of an environment that is capable of engendering natural exploration of the image.

We joined two separate research efforts in the work described here. Fast volume renderers were being constructed as part development work for the Pixel-Planes 5 graphics machine, and interactive segmentation using hierarchical descriptions of grey scale images is an on-going image processing project. The need for greater control of viewing parameters in volume rendering created a bridge between the two projects. The successful combination of the volume rendering and interactive semantic region classification has enhanced the strengths of each of the two techniques. Current work is generating excitement, not only among enthusiasts of volume rendering and image processing, but also of potential users in the medical community.

This paper presents the variations of parallel volume rendering that are being explored on Pixel-Planes 5 (a brief overview of the machine is included) and discusses their strengths and weaknesses. We follow with a description of the region of interest selection methods and the interactive tools we use. Finally, we demonstrate the unique flexibility and power of combining volume rendering with region of interest selection techniques by applying them to medical imaging.

This paper is expands and elaborates on [Yoo, Neumann, *et. al.* 90]. Presented here for the first time are discussion of the issues surrounding load balancing a parallel volume rendering approach, the tradeoffs between speed and image quality, and implementation details of the visualization system.

2. Graphics System

In order to provide adequate feedback, one needs a powerful graphics system. Another group in our department headed by Henry Fuchs and John Poulton provided us with such a machine in the summer of 1990 when they produced Pixel-Planes 5 (Figure 1) [Fuchs, *et al.* 89]. The machine is an experimental heterogeneous architecture suitable as a platform for a wide range of parallel algorithms research.

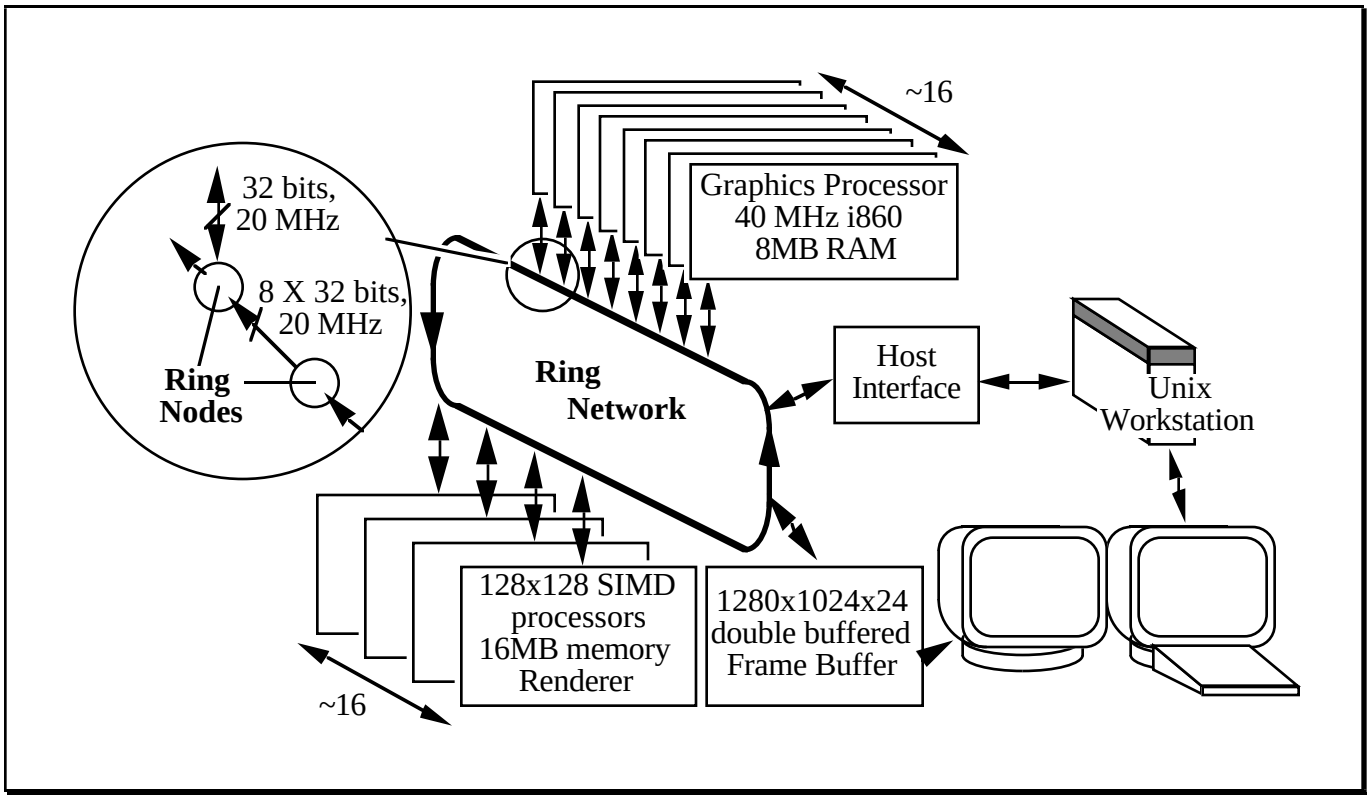


Figure 1: Overview of Pixel-Planes 5 System, showing Ring Network and Ring Devices that include Graphics Processors, Renderers, Host Interface, and Frame Buffer.

The system provides both MIMD and SIMD parallelism. MIMD parallelism is provided by the graphics processors, each of which contain independent code and data stores. SIMD parallelism is provided by the renderer boards, each of which executes a single instruction stream in parallel for 128x128 one bit processing elements. The instruction streams for the renderers are typically created by, and sent from, the graphics processors. The flexibility of having both types of parallel organizations available has resulted in the implementation of several different volume rendering algorithms. Two of those are discussed within this paper. In one approach, classification and shading of voxels is done with a SIMD algorithm, and the parallelization of raycasting is done through MIMD screen space subdivision. In the other, only MIMD screen space parallelism is used.

The communication ring is a high bandwidth general purpose inter-board communication link. A sustained bandwidth of 160 megawords per second is provided by 8 time-sliced 20 MHz channels.

The graphics processors (GPs) are 40MHz Intel I860™ general purpose microprocessors, each with 8MB of memory. Each GP has its own ring port for communication.

The renderer is a single board with a 128x128 SIMD array of pixel processors and its own instruction controller. The array may be positioned arbitrarily in screen coordinates, and through a hardware multiplier tree evaluate quadratic expressions in screen space. Each pixel processor has 208 bits of local memory and 4096 bits of fast access backing store. A fully configured system contains 16 renderer boards. Each renderer has two ring ports; one for its instruction stream, and one for backing store data access and loading.

The frame buffer is a 1280 x 1024 x 24 double-buffered design with a display refresh controller normally set for a 72 Hz vertical rate. It has a zoom mode which replicates one quadrant of pixels to yield an effective resolution of 640 x 512. Two ring ports are implemented to provide a high bandwidth into the frame buffer.

The host interface supports communication with a UNIX workstation. The workstation is the file server for the system and provides user control. One ring port is allocated to this function.

The complete 2 card cage Pixel-Planes 5 system contains 16 renderer boards, and 32 graphics processors (16 boards with two GP's per board). It may also be

configured as two independent half-sized systems (16 GP's and 7 renderers each). The results quoted herein are for one of these half-sized systems. The addition of a third cage is under consideration.

3. Parallel Volume Rendering

Volume rendering produces a shaded image from a 3D array of data samples. Three classes of volume rendering/modeling techniques are: Surface Representation [Lorenson, Cline 87], Binary Classification [Herman, Liu 79][Kaufman 90], and Multi-Valued Classification (MVC) [Drebin, Carpenter, Hanrahan 88][Levoy 88]. Using the MVC approach, volume data may be rendered as solid opaque objects, or as semi-transparent surfaces or gels; users may control the presentation to suit their need. Since MVC is superset of the other classes, it allows more flexible control of the real time image. In spite of its speed disadvantage, the systems presented have focussed around MVC.

In MVC volume rendering, data is treated as an array of point samples of a continuous 3D function. These points are shaded, resampled, and composited to produce an image. Shading is the process by which a data sample is converted to a color and opacity; an R, G, B, alpha four-tuple. A classification function ascribes an intrinsic color and opacity (alpha) to each point. The local gradient of the data provides a normal vector for each point. The intrinsic color and normal vector is used in a Phong lighting calculation [Phong 75] to yield the shaded R, G, B values.

Resampling of the shaded array is required to project the volume onto a 2D view plane. We employ ray casting with trilinear interpolation, but this is only one of many possible methods [Drebin, Carpenter, Hanrahan 88] [Westover 89].

Compositing [Porter, Duff 84] after each resampling step computes opacity and color accumulation by approximating an integral along the ray. Resampling must proceed front-to-back or back-to-front for compositing to work correctly. We composite front to back in order to take advantage of ray termination or alpha cutoff efficiencies [Levoy 90]. Rays terminate when they attenuate to opacity; further calculation is not required.

The next sections present two parallel volume rendering algorithms implemented on Pixel-Planes 5. The first presented is also chronologically the earliest implementation. The second version is the faster of the two, and in many respects grew out of the lessons learned from the first approach.

3.1. Distributed Data Implementation

A general purpose volume rendering system was developed in the Summer and Fall of 1990 for the Pixel-Planes 5 system, based on a design concept by Marc Levoy [Levoy 88]. The data set is distributed among the renderers; it is not replicated throughout the system. Our current allocation scheme uses 64 bits per voxel: 12 bits of raw scalar data, three 13 bit signed normals, and a 13 bit gradient magnitude. Using the 16 presently available renderers, this system can display 3D data sets up to 256x256x256 voxels in size.

Additionally, this system supports ramp classification and Phong shading of the data using the SIMD processor arrays in the renderers. The data set position as well as several lights can be manipulated via joysticks. A flexible ramp-based local classification converts intensity and intensity gradient magnitude to color and opacity also under joystick control. A moveable cut plane is provided. This system can input selection masks, from an external source, to highlight selected voxels. The update rate is about 1.4 frames per second with a 128x128x128 data set using 7 Renderers and 16 GP's to generate a 640 x 512 image.

Functions are assigned to the Pixel-Planes 5 hardware components as follows: The host controls the user interface, requests new frames to be drawn, and initially loads the data base into the renderer's backing store. The renderers access their backing store, perform conversion to color and opacity, and transmit shaded voxels (color and opacity) to the GP's. There is a single master GP; the remaining GPs are slaves. The master GP controls all the renderers and synchronizes the slaves. The slaves do the ray casting for their assigned portions of the screen and transmit their sub-images to the frame buffer.

Control Flow

Rendering a frame is divided into the following phases:

(1) The host requests a new frame and transmits viewing, classification, and shading parameters to the master GP.

(2) The master GP assigns enough "image tiles" (currently 128x128 pixels each) to the slave GP's to cover the display screen, and transmits the viewing parameters to the slave GP's.

(3) The slave GP's respond by computing which chunks of voxels, called "macros", are visible through their image tiles, and sending a list of these to the master GP. The master makes a global list of macro assignments to slave GP's. A macro contains 8x8x8 voxels.

(4) The slaves commence ray casting to create their image tiles. When a slave needs a macro which it doesn't have, it sends a "fetch macro" request to the master GP. The master responds by instructing the renderer that has that macro to classify and shade the backing store sector containing the macro. Each backing store sector holds 32 macros or 128 x 128 voxels (each processing element handles one voxel).

(5) After shading is completed, the master instructs the renderer to transmit all the macros in the backing store sector to the slaves that need them (whether or not a fetch request has been received). Hence each voxel is classified and

shaded only once; fetch requests for macros already sent, or in process, are ignored.

(6) When the slaves have completed all their ray casting, they fill in their screen region pixels via bilinear interpolation. Then, they send their image tiles to the frame buffer and notify the master that they have finished.

(7) When all the slaves have finished, the master GP toggles the frame buffer and notifies the host that the frame is done.

Classification Interface

The volume renderer does only local classification. That is, the color and opacity assigned to a given voxel is a function of that voxel's intensity, gradient magnitude and gradient direction only. Classification is performed via piece-wise linear functions (ramps). Opacity is computed by passing the intensity and gradient magnitude through two ramp functions and multiplying the results. Color is computed by passing intensity through separate red, green, and blue ramp functions. The system is programmed to allow the ramps to have any number of segments, but the current user interface control panel only supports three-segment ramps. The user has interactive control over the center, width, and height of each ramp. The user can flip among multiple classifiers with a single keystroke, to facilitate comparing different views of the data. For example, with CT data, one classifier could be set to show skin and another bone, and the user can rapidly flip back and forth between them (Figure 2).

The inherent limitations of local classification become apparent if the region being viewed has a similar intensity and gradient to its background. MRI data, for instance, is difficult to view using only local classification. Clearly, we need some means of distinguishing anatomically distinct regions of the same organ (Figure 3).

Optimizing the Ray Casting

The slaves use a variety of techniques to make the image tile generation fast. As macros arrive, an octtree is incrementally updated. The octtree is used to quickly skip over transparent regions, thereby avoiding trilinear interpolation and compositing in transparent regions. The renderers assist in computing the octtree by tagging macros which contain only transparent voxels. The octtree effort is worthwhile [Levoy 88, 90], as typically 2/3 of the macros are completely transparent. Rays are terminated when they accumulate sufficient opacity and further processing would yield little color change ("alpha cutoff") [Levoy 90]. Rays are cast adaptively. At first, a coarse grid of rays is cast. Then, if the adjacent values are sufficiently dissimilar, additional rays are cast. The final image is formed by bilinear interpolation of the ray-cast pixels. The addition of a cut plane adds no run time cost, in fact it makes the system faster. The cut plane is implemented by starting the rays at the plane instead of the data set boundary. Typical time for a slave GP to render a 128x128 tile, for a 128x128x128 data set, with coarse refinement, is 100-240 ms.

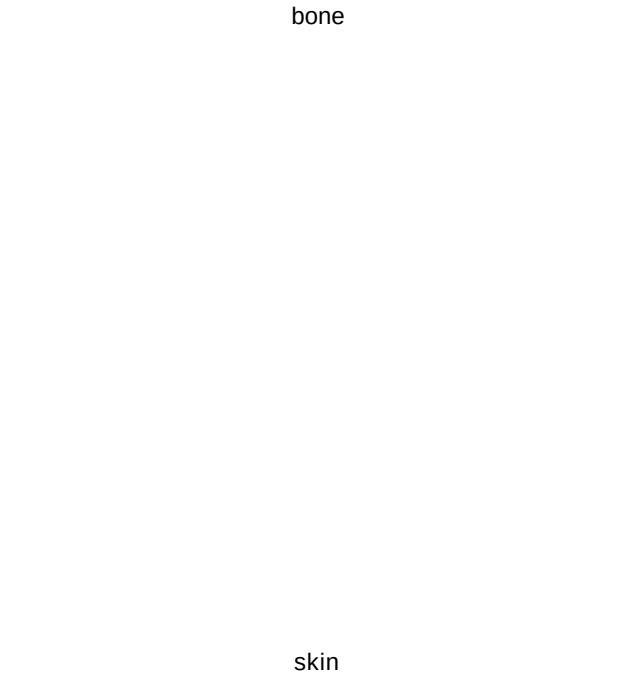
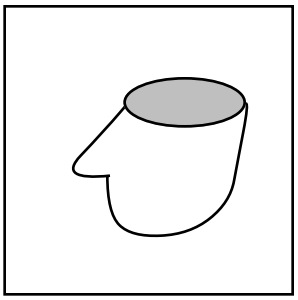


Figure 2: Two Views of CT Data Using Local Classification Techniques



head + cutplane

Figure 3: MRI data

Controlling the Renderers

The master GP implements a finite state automaton for each renderer to make the task of controlling them in parallel, while receiving asynchronous fetch requests, manageable. The main states, and the approximate time in each (per renderer, per sector) are:

100 us 1. Loading a backing store region (32 macros, or 16K voxels) into a renderer's pixel memory.

2-3 ms 2. Classifying and shading the entire region in parallel (SIMD).

100 us 3. Storing the region of shaded voxels in backing store for transmission to the slave GP's.

2-3 ms 3. Sending each of the 32 macros to all the slave GP's whose screen regions "see" them.

100 us 4. Receiving acknowledgement from the renderer that it is finished and available again.

Performance

The current frame rate of 1.4 frames per second isn't quite adequate for comfortable interactive viewing. The kinetic depth effect is lost and using the joysticks for control is awkward. Our goal is at least 5 frames per second with a 128x128x128 data set. Good load balancing among the slave GP's is difficult to obtain. Operating system overhead for message passing among the GP's is higher than we expected, so more efficient low level communications routines must be developed to lower the expense of short synchronization messages. The ring bandwidth is sufficient for our desired level of performance, our GP's are not yet fully utilizing it. The use of a single master GP introduces latency and thus is a bottleneck. Execution time profiles show the slave GP's idle about 30-50% of the time on average. We have installed software for recording time-stamped events and are currently using it to understand where the time is going and finding ways to speed up the algorithm. When the processor utilization is satisfactory, we expect to devote some serious effort to optimizing the code itself.

3.2. Replicated Data Implementation

The bottlenecks in the volume renderer described above, led us to explore parallel algorithms that avoid communication of data and control messages. Taking this pursuit to the extreme, we have implemented a volume renderer that replicates the data set at each GP and thereby achieves update rates of 15 frames per second. Due to GP memory limitations, the data set size is limited to 128x128x128, but this is sufficient for much of our work and demonstrates the capabilities of this approach. Classification and shading must now be performed in the GP's so an efficient approach to shading must be used. As a preprocessing step, each voxel has its normal computed and encoded as a 13 bit number: 6 bits for the X component, 6 bits for the Y component, and 1 bit for Z (since the normal is a unit length vector, the Z magnitude can be inferred from the X and Y components). This 13 bit normal is then used as an index into a shading table. This table is recomputed at the start of each frame using the current classification function and a Phong shading model. This results in only 2^{13} shading computations rather than the 2^{21} that would be needed for a 128 x 128 x 128 data set if the voxels were shaded directly.

This approach uses a 32 bit per voxel encoding scheme currently allocated as follows:

8 bits : original CT or MRI data
13 bits : normal encoding
8 bits : gradient magnitude
3 bits : currently unused.

This look up table shading method allows fast update rates for a volume renderer with a standard gradient based opacity shader (Figure 4). The currently unused bits may be used to perform simple classification schemes in the future.

4. Interactive Semantic Region Selection

Volumetric data often overwhelms viewers with non-essential information; a visualization program should also allow the user to emphasize specific elements of interest in the image. This requires a set of data selection tools to pare away extraneous detail. Such tools include: geometric, syntactic, and semantic classifiers. Geometrically based sculpting or clipping controls remove unwanted portions of the image. Syntactic controls use local image data properties to enhance or de-emphasize image features; ramp-based classification is an instance of syntactic control. Semantic controls use global set-membership of data points to allow the viewer to define particular regions of importance.

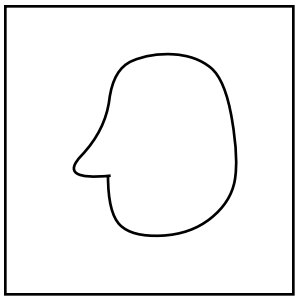


Figure 4: Image produced using table driven volume renderer

Since local properties of the data do not often reveal information regarding morphological features within the image, and since geometric classification is sometimes unwieldy, it is useful to apply some 'semantic' groupings of a given data set to isolate features of form. Selecting regions on a voxel by voxel basis is not a practical means of defining features of 3D data. We attempt to gather together voxels of similar nature into primitive collections of voxels which then become the basis of the shape driven classification. Further segmentation is then performed in a second pass by collecting together these region primitives in coherent volumes that encompass features in the data set. The way in which these primitive groupings relate to global structure are most effectively determined by a knowledgeable user. What is required is a graphical language of sensible fundamental regions and a means for users to interact with them.

4.1. Hierarchy Generation

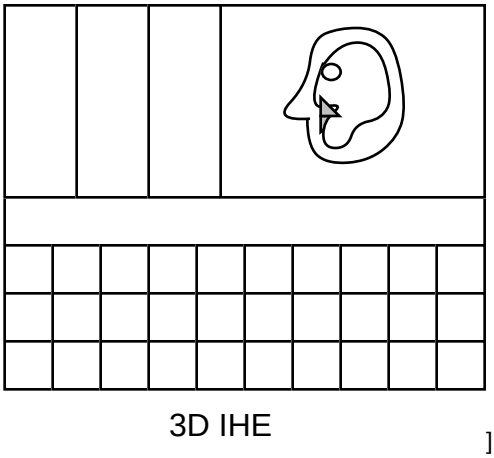
One means of representing related voxels in a 3D image is the creation of a hierarchical tree structure. Voxels with similar characteristics can be collected into primitive regions; these primitive regions can then be organized into a directed acyclic graph. These regions can be assembled using the knowledge of the graph hierarchy, or they can be arbitrarily grouped using set operations. This kind of hierarchy provides a handle for manipulating groups of voxels to describe portions of the data.

In previous research with 2D images, we generated these region hierarchies using one of many techniques [Pizer, Cullip, Fredericksen 90]. These methods characterize a 2D image as an intensity surface, and define regions based on the nature of the ridges and valleys in that surface. In our current work we extend these ideas for segmentation to three dimensions. We can treat the images as a three manifold in four space, and and analyze 'ridges' and 'valleys' in the 3-fold. We are now computing reverse gravity watersheds of 3D images. On a Decstation3100 the preprocessing necessary to create the hierarchy for a 128x128x128 image with 256 gray levels can be performed in roughly 15 minutes.

Our plans include the development of other hierarchy generation algorithms [Gauch, Pizer 88]. The shortcomings of reverse gravity watershed are known; it was however already available as a 3D segmentation scheme.

4.2. The Interactive Editor

We have developed a system to provide users with a means of manipulating the regions of an image hierarchy. The 3D interactive hierarchy editor (3D IHE) provides the user with mouse-based interactions and a view of the multiple slices of the dataset. We chose a slice paradigm as the basis for navigating three dimensional images because of its innate ability to project away one level of dimension without loss of user comprehension.



]

Figure 5: 3D IHE Console Window

The interface contains three basic components (Figure 3). First, a full size view of one slice through the dataset. Mouse operations in this window allow the user to select/deselect primitive regions for classification. Although the interaction is within one slice, the selected regions extend in 3 dimensions. Buttons on this window allow the user to move forward and backward in the unseen dimension, in order to view adjacent slices. The second component is a series of scaled down images (24 in the current implementation) that allow the user to see the effects of region selection on a wide range of slices. A slider on this window allows the user to view different slices in the dataset. A frequency button allows the user to subsample the range of slices in order to provide views of slices throughout the entire dataset simultaneously. The final component of the editor is a control panel that provides a

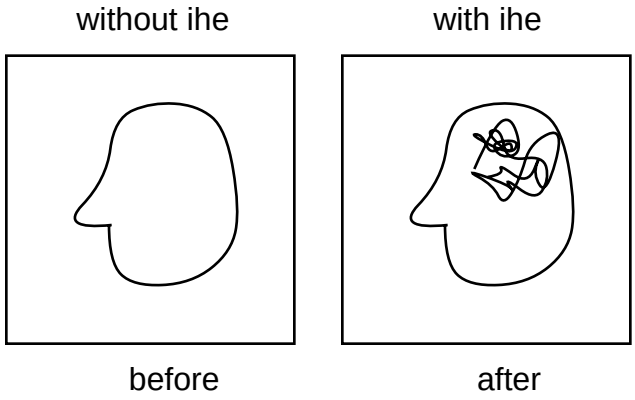


Figure 6: Comparison views with and without global classification
(left is without semantic segmentation; right is with IHE regions applied)

number of I/O operations, as well as hierarchical movement.

The results of 3D IHE show a dramatic improvement over the earlier slice by slice and voxel painting methods. For example, we created descriptions of a 128x128x128 dataset of a human head, and choose the task of selecting the a portion of the brain for volume rendering. Using 2D descriptions, and editing slice by slice (68 slices contained portions of interest), the task took approximately 40 minutes to complete. With the 3D description, the task required approximately 10 button presses, and took less than a minute. Using the same 3D description the entire brain can be selected in approximately 4 minutes.

4.3. The IHE to Pixel-Planes Connection

We combined 3D IHE with volume rendering to create a powerful visualization suite. segmentation performed using IHE can be used to specify regions of interest to the volume renderer. We obtain better perception of the region selection process and subsequently aid the overall visualization of 3D volume images.

The two systems are connected using standard Unix sockets via an ethernet, passing image masks and op-codes in TCP packets. Sending a selection mask from the IHE application currently takes about 10 sec. The mask is used during ray casting to emphasize selected voxels by modifying their color and opacity. The system currently does not permit the selection mask to interact with the raw data before local classification; something we will change in the future.

5. Application Results

Even with the limited, early version of this explorational 3D visualization system described above, we are finding that exploration of 3D data, with immediate feedback and on-line classification, qualitatively increases the user's ability to understand the information in a medical image. The ability to dynamically control opacity and regions of interest (ROIs) has the potential of changing 3D display from a postprocessing step to a primary way of viewing medical image data.

The real advantages that have been ascribed to volume rendering seem to lie particularly in the direct rendering from the original image data. If the user can focus on ROIs and on the dynamic selection of rendering parameters that optimize the visualization of surfaces, the actual means of the producing the final rendering, whether by surface selection and subsequent rendering, or by fuzzy classification and compositing, may not be central. An example of the power obtained through this approach is the selection of the cortex region in an MRI data set and then the dynamic rendering of the white matter by the selection of appropriate classification ramps determining either opacity or selection threshold (Figure 6). This level of visualization is now possible in a couple of minutes by exploring parameter values using immediate feedback.

Of course, viewing of medical images can never be divorced from viewing the contrasts in the original data. However, with the addition of dynamic selection of grey scale slices from the 3D data set, with possible superimposition of these slices on the cutplane, the grey scale viewing mode can be fully integrated into the 3D explorational viewing style.

6. Conclusions

In creating this system, we learned some valuable lessons. The requirement for immediate feedback makes speed a critical issue. By employing an approach that replicates data throughout a MIMD machine we can overcome communication bottlenecks at the expense of limited image size. Using lookup tables for normal and opacity values, we can reduce the number of required shading calculations by several orders of magnitude. This allows us to achieve the necessary speeds of 10-20

frames per second; the cost for these speeds is an acceptable loss of image quality.

The rendering speed of 0.7 sec with our distributed volume renderer is still one order of magnitude too slow; and the need of a few seconds to select a primary region from our quasi-hierarchical description or move to a parent in that description is not acceptable. Nevertheless, the present speeds demonstrate the power of user selection of 3D ROIs when supported by communication in terms of precomputed sensible regions and immediate display feedback. Using this approach, the time required to select particular regions of medical data has been shown to be reduced by an order of magnitude. The present system also demonstrates the power of interactive control of local classifiers such as clipping and opacity ramps within ROIs selected by the interactive editor.

Application of the combined systems to real images reveals their great potential for facilitating rapid comprehension of volume data. Even the limited speeds of the present interactive system demonstrate the great importance of fast general purpose parallel graphics systems, such as Pixel-Planes 5, and the need for even faster systems with even faster data access and faster rendering algorithms on these systems-- not simply as a convenience but because they allow qualitative increases in user comprehension of their data. In the future, when such machines that are capable of supporting a wide range of segmentation and classification techniques at interactive rates of up to 30 frames per second, and when they are more widely available, volume visualization will be a much more powerful approach. We expect that when interactive semantic region definition with volume visualization can be achieved in real time on affordable platforms, it will be the standard means for viewing 3D data.

Acknowledgements

We would like to thank Bill Oliver and Julian Rosenman for their expert advice and the physician's point of view. We are also indebted to Jeff Butterworth, Marc Levoy, and John Poulton for both their input and the significant contribution that their research represents.

This research has been funded in part by the following grants

Defense Advanced Research Projects Agency ISTO order No. 7510
National Institutes of Health Grant No PO1 CA47982
National Science Foundation Grant No. MIP-8601552

References

Drebin, Robert A. , Loren Carpenter and Pat Hanrahan. Volume Rendering. Proceedings of SIGGRAPH'88, *Computer Graphics* **22**, 4. August 1988, pp. 65 - 74.

Fuchs, Henry, John Poulton, John Eyles, Trey Greer, Jack Goldfeather, David Ellsworth, Steve Molnar, Greg Turk, Brice Tebbs, Laura Israel. "Pixel-Planes 5: A Heterogenous Multiprocessor Graphics System Using Processor-Enhanced Memories." Proceedings of SIGGRAPH '89*, Computer Graphics,* .**23**(3): 79-88. (ACM: New York), 1989.

Gauch, J. M. and S.M. Pizer, "Image Descriptions via the Multiresolution Intensity Axis of Symmetry." *Proc. 2nd Int. Conf on Cop. Vis.* (IEEE Catalog #88CH2664-1): 269-274, 1988.

Herman, G. T., H. K. Liu., "Three-Dimensional Display of Human Organs from Computed Tomograms." *Computer Graphics Image Process. ,* **9**: 1-21, 1979.

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

Levoy, M., "Efficient Ray Tracing of Volume Data." *ACM Transactions on Graphics ,* July 1990 **9**(3): 245 - 261, 1990a.

Levoy, M., "Volume Rendering by Adaptive Refinement." *The Visual Computer ,* February 1990 **6**(1): 2-7, 1990b.

Lorenson, William and Harvey Cline. "Marching Cubes; A High Resolution 3D Surface Reconstruction Algorithm". Proceedings of SIGGRAPH'87, *Computer Graphics* **21**, 4. July 1987, pp. 163-169..

Kaufman, Arie. Volume Rendering Architectures. SIGGRAPH'90 Course 11, notes for Volume Visualization Algorithms and Architectures. August 1990, pp. 189 - 198.

Phong, Bui-Thong. Illumination for Computer Generated Images. *Communications of the ACM*, **18**(6), June 1975, pp. 311 - 317.

Pizer, Stephen M., Timothy J. Cullip, Robin E. Fredericksen, "Toward Interactive Object Definition In 3D Scalar Images." *3D Imaging in Medicine* (NATO ASI Series F: Vol. 60): 83-105, Springer-Verlag, Berlin, 1990.

Porter, Thomas, Tom Duff. Compositing Digital Images. Proceedings of SIGGRAPH'84, *Computer Graphics* **18**, 3. July 1984, pp. 253 - 259.

Westover, Lee. Interactive Volume Rendering. Conference Proceedings, *Chapel Hill Workshop on Volume Visualization*. May 1989, pp. 9 - 16.