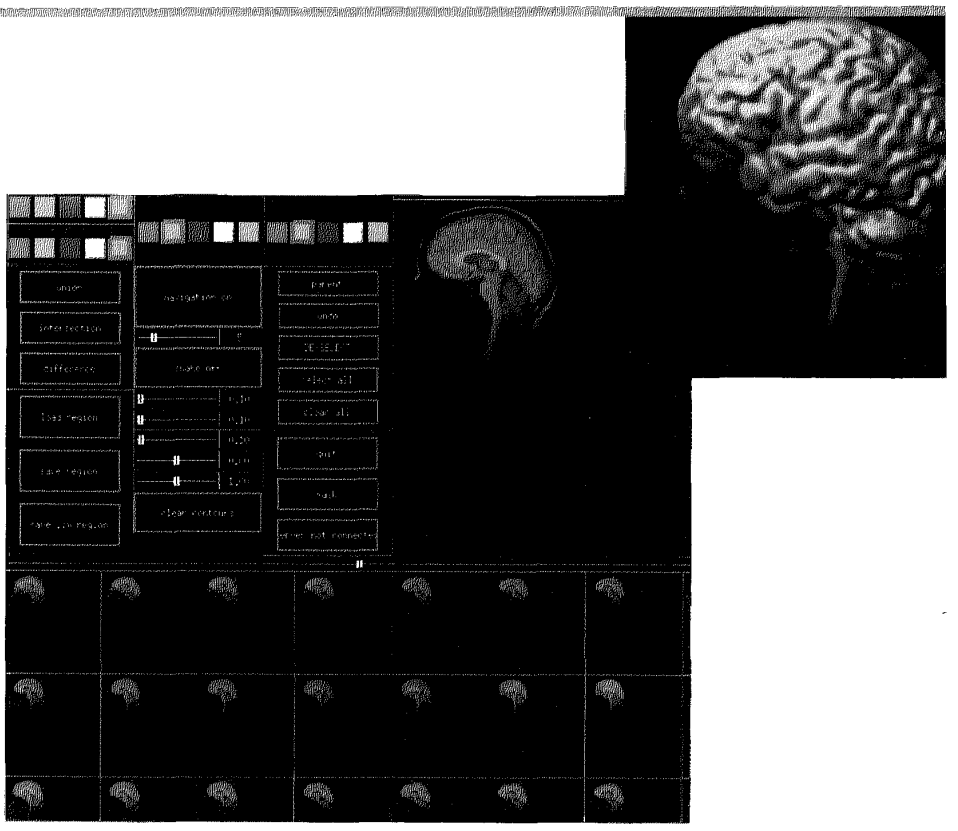

# Direct Visualization of Volume Data

Terry S. Yoo, Ulrich Neumann, Henry Fuchs, Stephen M. Pizer,
Tim Cullip, John Rhoades, and Ross Whitaker
University of North Carolina at Chapel Hill

*A combination of segmentation tools and fast volume renderers provides an interactive exploration environment for volume visualization.*

Visualization requires more than just the production of complex images of higher dimensional data: Users must be able to comprehend the information within the data. Traditional approaches to representing 3D information using volume rendering often lack an environment that lets users explore an image. Here we discuss several techniques that help make volume visualization more user driven. They include mechanisms that distribute volume data across multiple processors, as well as image compositing techniques and solutions to representation problems in the selection and display of subregions within bounding volumes. Our discussion is comprehensive and trades depth for breadth.

In this article we describe our approach to direct volume visualization: the interactive control of images rendered directly from volume data coupled with a user-controlled semantic classification tool. We present the variations of parallel volume rendering being explored on the Pixel-Planes 5 system at the University of North Carolina at Chapel Hill. (We include a brief overview of the machine.) Then we describe the region-of-interest selection methods and the interactive tools we use. Finally, through medical imaging applications, we demonstrate the flexibility and power of combining volume rendering with region-of-interest selection techniques.

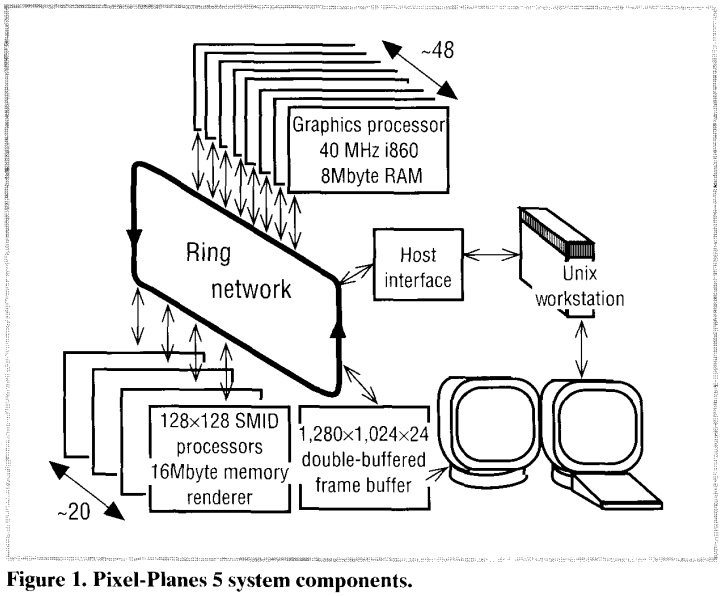


**Figure 1. Pixel-Planes 5 system components.**

## Interactive classification

We distinguish two early stages in volume visualization as *segmentation* and *classification*. By segmentation we mean the division of a data set into often nonoverlapping subvolumes that have some cohesive meaning unto themselves. Classification, on the other hand, is the process of labeling the various portions of an image by type. In many cases, data must be segmented into cohesive regions before labeling. If classification happens at the voxel level with each voxel labeled by type (in the case of medical data, by tissue type), the data has been segmented down to the individual voxels, the smallest cohesive regions possible.

Segmentation and classification are often performed together in a preprocessing stage, but it is possible to do them separately in different stages. Our system provides a powerful segmentation preprocessing stage and then allows the user to control classification interactively while viewing the data with a volume renderer. By giving the user classification and labeling control at rendering time, we provide a flexible means of specifying the visualization.

In addition to the distinction between segmentation and classification, we distinguish two types of classification: *syntactic* and *semantic*. Syntactic classification uses only the information contained within the data set to perform the segmentation or classification. Available information includes voxel intensity, local gradient information, and connectivity.

Local properties of the data often do not reveal information regarding morphological features within the image, and geometric tools are sometimes unwieldy. Therefore, 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 practical for defining features of 3D data, so our system gathers voxels of similar nature into primitive collections that then become the basis of the shape-driven classification. The user performs classification in a second pass by collecting these region primitives into coherent volumes that encompass features in the data set. The way these primitive groupings relate to global structure is most effectively determined by a knowledgeable user, who requires a graphical language of sensible fundamental regions and a means to interact with them.

We contend that effective volume visualization requires this level of control. Control of the labeling should rest in the hands of expert users who apply their knowledge of the information in the visualization. Effective visualization is possible only when the user who explores the data controls the view.

## Display

Researchers exploring advanced graphics architectures at the University of North Carolina at Chapel Hill provided us with a platform for our research. Completed in the summer of 1990, Pixel-Planes 5[1] is a heterogeneous graphics architecture using both MIMD and SIMD parallelism.

As Figure 1 shows, the Pixel-Planes 5 system has multiple i860-based Graphics Processors and multiple SIMD pixel-processor arrays called Renderers. Each Renderer is a 128 × 128 array of pixel processors capable of executing a general-purpose instruction set. Graphics Processors send Renderers operation-code streams that are executed in SIMD fashion. Renderers also have a quadratic expression evaluator that can be configured to occupy any screen position. Special operation codes evaluate the function

$$Q = Ax + By + C + Dx^2 + Exy + Fy^2$$

in the Renderer for each pixel's unique $x$, $y$ location. The coefficients A through F are part of the instruction stream from the graphics processors.

The Graphics Processors, the Renderers, a frame buffer, and a workstation host communicate over an eight-channel one-dimensional ring network whose aggregate bandwidth is 5.12 Gbits per second. Renderers supplement their working memory with a secondary memory or backing store that may contain large quantities of user data as well as completed rendered images. Renderers also have backing store ports that allow data movement in and out of secondary memory under Graphics Processor control.

## Volume renderers

A fundamental operation in any volume renderer is the reconstruction of a continuous function from discrete samples. The efficiency with which this is done has a direct impact on rendering speed. One aspect of our current research is a comparative analysis of the speed and quality of multipass shear, splatting, and trilinear reconstruction techniques.

Parallel approaches to volume rendering are complicated by the fact that data distributed throughout the system's memory spaces must be combined in depth order during image rendering. This partial sorting of potentially large volumes of data can tax the communications network, resulting in poor performance. An important element in minimizing the expense of this operation is the data distribution itself.

This section describes the evolution of a parallel algorithm that achieves both fast update rates and good image quality. The final algorithm is the culmination of several disparate efforts. Each incremented effort resulted in a product whose performance in some way fell short of the performance needed for the visualization described in this article.

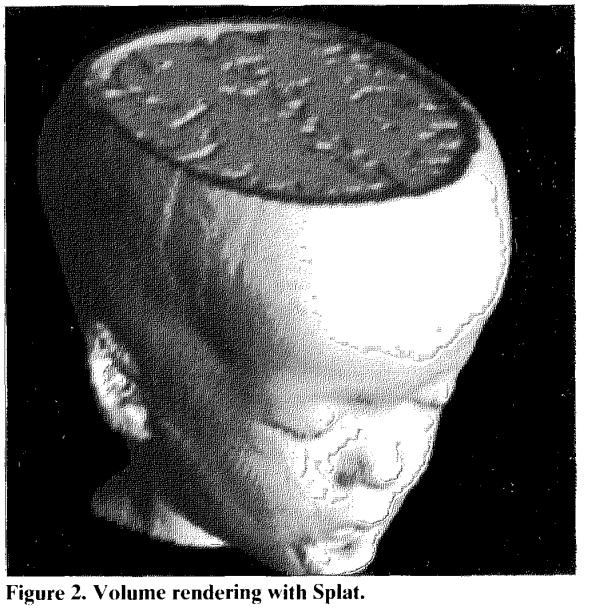

**Figure 2. Volume rendering with Splat.**

### VRN

VRN, one of the first applications to run on Pixel-Planes 5, is an implementation of multivalued classified volume renderers.[2,3] Its unique aspect is its exploitation of the Renderers' SIMD parallelism to perform the classification and Phong shading of the voxels. The algorithm distributes data among the Renderers' backing store memory. The Renderers perform elementary shading and syntactic classification, and transmit shaded blocks of data to the Graphics Processors for ray casting on a demand basis. A designated master processor creates a block hit list that allows the sending of shaded blocks to all the Graphics Processors that handle rays passing through that block—whether or not the processors have already requested them. VRN incorporates several techniques[4] to speed the ray casting on the Graphics Processors: alpha cutoff, adaptive sampling, and incremental octrees.

An undesirable aspect of this approach is the latency in a Graphics Processor's access to the data needed for a ray. Much of the data set must be transmitted over the communications network every frame. Load imbalance among Graphics Processors is significant and difficult to correct. These shortcomings (among others) limit the speed of this approach to about three frames per second for a $128 \times 128 \times 128$ data set using 20 Graphics Processors and eight Renderers.

### VOL, Splat, and DIVO

Researchers at UNC made several incremental steps toward improved volume rendering on our multicomputer architecture. The effort centered around overcoming the bottlenecks in the VRN algorithm by ray casting into unshaded volumes and performing shading at rendering time. The volume renderers known as VOL, Splat, and DIVO demonstrate advances in accelerated shading, reconstruction of images from a minimal number of samples, and ray casting in a distributed data environment.

VOL circumvented the data distribution problem by duplicating the full data set at each processing node and attacked fast shading. This programming effort demonstrated the efficiency of lookup table techniques to speed shading operations. VOL performs syntactic classification and Phong shading by using lookup tables indexed by the raw data, its gradient, and gradient magnitude.

Splat generated techniques for parallel screen space image reconstruction from volume data. It is based on splatting, a feed-forward mechanism rather than a ray-casting approach to volume rendering.[5] Neumann constructed a parallel implementation of splatting on Pixel-Planes 5 in the spring of 1991.[6] This volume renderer uses the Renderer quadratic expression evaluator to efficiently generate the reconstruction filter kernels needed for splatting. Figure 2 shows example output from Splat. This approach demonstrated an efficient screen interpolation method later used in DIVO and subsequent volume renderers.

If data is distributed and ray casting is parallelized among separate Graphics Processors, the system must reconstruct the ray segments into coherent rays. DIVO was an attempt to address the issues surrounding data distribution and ray reconstruction. It introduced the idea of pipelining by assigning ray casting to one group of Graphics Processors and assigning compositing and screen interpolation to another. DIVO distributes the data blocks among the ray-casting Graphics Processors.

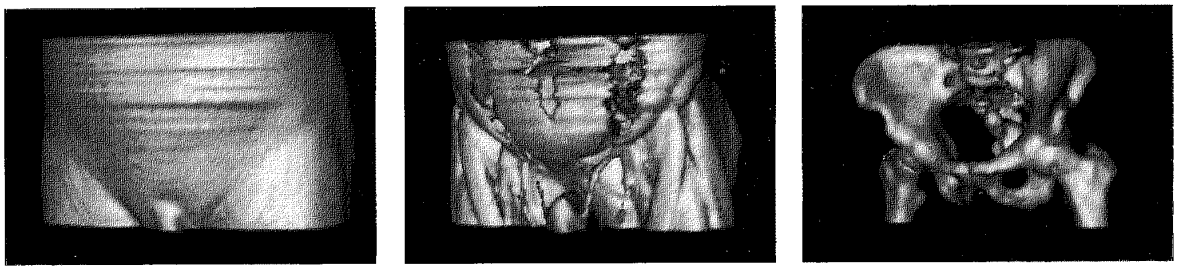

**Figure 3. Three different intensity classifications of a CT scan: skin surface (a), musculature (b), bone (c).**

In DIVO, a subset of the Graphics Processors casts rays on a regular screen grid through the Graphics Processors' data blocks. Each Graphics Processor computes an <RGBα> tuple for the ray fragments through its data block. The tuple arrays are written to the Renderers' data memory. The Renderers composite the many fragments and distribute the final ray colors to the interpolation Graphics Processors. A binary space-partition tree determines the proper order for compositing. The interpolation Graphics Processors use either software or a modified splatting technique and the Renderers to do screen interpolation. The advantages of using the Renderers are that this approach is actually faster than bilinear interpolation on the Graphics Processors, and higher order filter kernels can be used, producing less offensive artifacts for the coarsely sampled images generated while the user navigates through the data set. Kinetic depth perception is much better using splatting because of the less objectionable artifacts.

DIVO accommodates large data sets (greater than 256 × 256 × 256). The system performs well, except that the ray-casting Graphics Processors are not evenly loaded because the data cubes have varying amounts of nonzero data in them.

### VVEVOL

The VVEVOL renderer is the culmination of the ideas and techniques developed in the four earlier approaches. VVEVOL uses the lookup table method for syntactic classification and Phong shading. It performs progressive refinement from a coarse density of one ray per 8 × 8 screen pixels to a final density of one ray per pixel. One set of Graphics Processors supports pipelined ray casting, and another supports compositing and interpolation. The compositing Graphics Processors assemble ray fragments into a final ray color. Finally, the Renderers translate the rays into splat kernels using the Renderer quadratic expression evaluators to perform parallel screen interpolation.

Dynamic load balancing of the ray-casting Graphics Processors is accomplished by dividing them into several groups. The data set is divided among groups as approximately equal slabs. Each group of Graphics Processors is responsible for a full screen of rays cast through the group's slab. Each Graphics Processor within a group has the same slab subset of the whole volume, thus allowing the system to assign rows of rays to Graphics Processors on a demand basis. Within a group, therefore, the Graphics Processor loads are balanced. The imbalance between groups is typically small because usually there are only a few groups. Memory and data set size dictate the number of groups.

VVEVOL can render shaded images from 128 × 128 × 56 volume data at 20 frames per second, and 192 × 192 × 128 volume data at 11 frames per second. Semantically selected regions specified by the user are rendered as surfaces within the volume.[3,7]

The evolution of volume renderers on Pixel-Planes 5 has generated new thoughts about the trade-offs made in sampling and image reconstruction, as well as the sorting and distribution of volume data throughout a parallel architecture. We have analyzed the costs of communication and the importance of load balancing in a generic MIMD architecture with local memory at each computing node. Schroder and Salem, in their discussion of memory motion on the Connection machine,[8] emphasize that these are concerns throughout all parallel architectures and merit further investigation.

## Classification

Our research handles classification both syntactically and semantically. Many volume renderers are equipped with gradient magnitude and intensity magnitude classification tools that allow the user to select and display various data set elements that have consistent properties. Two of the volume renderers described in the previous section allow alternate data input for connection with semantic visualization interfaces, adding user-supervised semantic classification to their capabilities.

### Syntactic classification: One pass

We implement syntactic classification in two forms. In the first, a series of linear (or other parametric) matching functions lets the user map intensity values to display parameters. Gradient mapping is widely used in our systems; we refer to this class of methods as ramp classification. The second type of classification is purely geometrical and involves in its sim-

plest form a series of clipping surfaces used to pare away extraneous portions of the data set. These two approaches – intensity or gradient windowing and geometric sculpting – often seem natural to users.

In ramp classification, many volume data sets have natural divisions along intensity values, local density, or gradient magnitude. An intensity-value histogram will show significant divisions among the different classifiable groups of voxels. The example in Figure 3 shows three views of a human pelvis acquired using X-ray computed tomography with tissue density as the classification parameter. It is natural to think of bone as denser than muscle and muscle as denser than skin, so ramp methods work well for visualization of this type of data.

One caution about this method: These classifying divisions are often subject to changes in the parameters of the image acquisition system and are not reliably reproducible in arbitrary images, even of the same original scene. This was an important motivation for providing interactive classification control at rendering time.

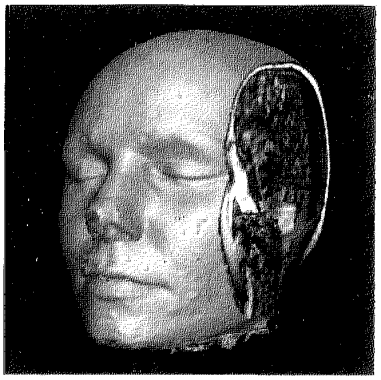

**Figure 4. Geometric sculpting as classification.**

Geometric sculpting tools are also natural and powerful implements in visualization. The cutting plane is particularly valuable in medical visualization. Medical imaging systems that compute 3D reconstructions from planar images present the tomographic data as a series of slices. Figure 4 shows the use of a cutting plane to reproduce a particular section or slice of cranial anatomy within the context of the surrounding tissue.

For this presentation to fit well with existing diagnostic paradigms, some manipulation of the cutting plane surfaces is required. Most visualization systems render the exposed surface using normal interpolation and shading techniques. This may appeal to students and pathologists who often view prepared slices and sections of actual tissue. However, most radiologists are disturbed by this approach and wish to see the original tomographic data texture-mapped onto the exposed surface. Their common view of the data is through film reproductions of computed tomographic slices of a patient, and they require some assurance of a correlation between the volume rendering and the original data. If these user interface issues are resolved, the type of visualization we describe here may become an extension of the diagnostic tools in use today.

As powerful and rapid as these methods can be, they cannot distinguish between the different bones of the leg and different muscle groups of the pelvis, or follow the contours of separate lobes of the brain. For that we turn to the image analysis techniques that contribute the semantic tools for classification.

## Semantic classification: Two passes

Our approach to semantic classification is to perform the labeling process in multiple steps. First, in a preprocessing step,our system segments the image into a large number of mathematically cohesive volumes using strictly syntactic information. Second, it automatically generates a hierarchy to connect the primitive regions with a directed acyclic graph. The goal is to parse an image into cohesive elements and then provide the user with an intuitive language to describe its regions. Finally, at rendering time, the user interactively traverses the graph, grouping primitive regions into classifiable subsets.

We try to oversegment an image and leave the parameterization of the disjoint segments until rendering time. Using the hierarchy, we shorten the human interaction time required to group the elements into classifiable regions.

## Syntactic segmentation as preprocessing

Here we introduce two algorithms to generate hierarchies in the preprocessing stage. We also describe the interactive tool that allows users to perform interactive classification. Since the geometry is often hard to conceptualize, we describe the algorithms as implemented on 2D images. Figure 5

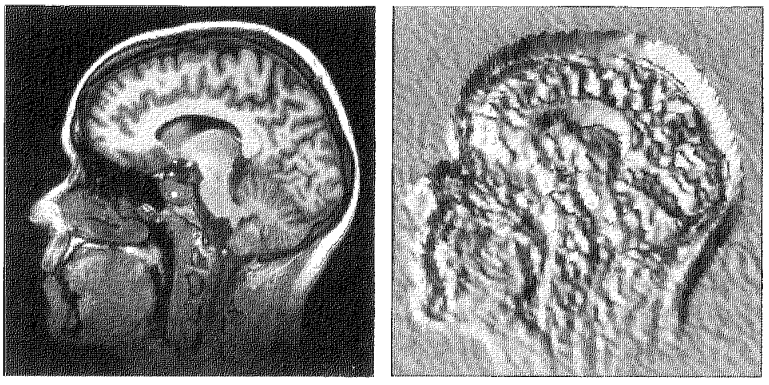

**Figure 5. Two conceptualizations of a 2D image.**

shows a 2D image with variable intensity at each pixel. If we consider the image as a surface of variable height (shown on the right), rather than a plane with variable intensity (as shown on the left), we see the image as an intensity mountain range, with peaks, ridges, valleys, and saddle-shaped gaps dividing the ranges of peaks.

In previous research with 2D images, we generated these region hierarchies using one of many techniques.[9] These methods characterize a 2D image as an intensity surface and define regions based on the geometric 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 a volume image as a 3-manifold in 4-space, and analyze ridges and valleys in the 3-manifold.

## Reverse-gravity watershed techniques

Consider a $d$-dimensional image as a function $I$: $R^d \Rightarrow R$, say $I(x)$, where $x \in R^d$. Let $M$ be the set of critical points $y$ for which $I(y)$ is a relative maximum. Given $y \in M$, the *reverse-gravity watershed* corresponding to $y$ is the set of points $x_0 \in R$ such that there is a path of steepest ascent from $(x_0, I(x_0))$ to $(y, I(y))$ along the graph of $I$. Intuitively, if we place a droplet of water on the intensity surface and allow it to flow against gravity, then the droplet will follow a steepest ascent path to a local peak on the surface. A concise mathematical description for such a region is the following: A point $x_0$ is in the reverse-gravity watershed of $y$ if there is a $T > 0$ such that the solution $x(t)$ to the differential equation

$$\frac{dx(t)}{dt} = \frac{\nabla I(x(t))}{\|\nabla I(x(t))\|}, t > 0, x(0) = x_0$$

satisfies $x(T) = y$. Here $\nabla$ is the gradient operator and $\| \|$ denotes the length of a vector. The right-hand side of the differential equation is the unit gradient vector for $I$, which points in the direction of steepest ascent.

Sometimes the following hierarchy-generation technique is called the *peak-flow method*. We can partition the domain for $I$ into reverse-gravity watersheds as a method of segmentation. A critical point yielding a relative minimum is arbitrarily assigned to a watershed on which it borders. Given this collection of primitive regions, we construct a hierarchy from it. The method we use is based on the concepts of blurring and scale space. The original image is blurred via a Gaussian kernel of standard deviation $s$, called a *scale*. The blurred image $J(x, s)$ is a solution to the partial differential equation

$$\frac{\partial J(x,s)}{\partial s} = s\nabla^2 J(x,s), \quad s > 0$$

with initial data $J(x, 0) = I(x)$. The operator $\nabla^2$ is the Laplacian.

As scale increases, the relative maxima of $J(x, s)$ are annihilated. In fact, at the limit, as scale becomes infinite, the image becomes constant. Annihilation occurs when two or more relative maxima merge into a single such point. In the region hierarchy, the watersheds associated with merging relative maxima are linked as siblings of a common parent. As scale increases, more mergings occur and more tree links are set. When the blurred image is finally constant, the final parent nodes of the hierarchy are linked to a single root.

## Ridge-flow techniques

We are developing a more sophisticated hierarchy-generation technique called the *ridge-flow method*. Instead of identifying regions associated with relative maxima, we construct regions based on the ridges of the intensity surface. For the intensity function $I(x)$, $x \in R^d$, we define *ridge points* to be those points $x_0$ such that the curvature of the level set defined by $I(x) = I(x_0)$ has a relative maximum at $x_0$.

It is difficult to work with the level sets of $I$, so in the implementation we have a measure of curvature at a point $x$ without reference to level sets. The measure is

$$c(x) = trace\left(\nabla \otimes f\frac{\nabla I(x)}{\|\nabla I(x)\|}\right)$$

where $\otimes$ is the standard tensor product between vectors. The tensor product yields a matrix, and $C(x)$ is the trace (sum of diagonal elements) of the matrix. We threshold this curvature measure to get a candidate set of ridge points, which in turn we thin to the appropriate size using methods of mathematical morphology.

Each connected component of the ridge structure for the intensity surface is either an isolated point or a continuum of points that we can partition into curvilinear segments. Each such segment has an associated reverse-gravity watershed region, just as before. In this case, however, paths of steepest ascent are traversed until a ridge point is encountered. We also obtain some finer segmentations by partitioning the ridges at points where the intensity surface has a local minimum along the direction of the ridge.

The hierarchy-generation technique we use for this segmentation relies on identifying ridge-flank and ridge-ridge relationships. If a ridge lies on the flank of a larger ridge, then we create a parent-child link from the region of the large ridge to the region of the small ridge. If three or more ridges meet at a single junction, then we create the hierarchy links on the basis of collinearity of the joined ridges. For example, if three ridges are joined, we identify the two ridges that are nearly collinear (while traversing the ridges) as siblings. We consider their parent to be a sibling of the remaining ridge. The resulting hierarchy will be a forest of trees.

## Navigating the hierarchy

Our system lets users manipulate the regions of an image hierarchy. The 3D interactive hierarchy viewer (3D IHV) permits mouse-based interactions and provides a view of the multiple slices of the data set. We chose a slice paradigm as

the basis for navigating through 3D images because of its inherent ability to project away one level of dimension without loss of user comprehension.

Figure 6 shows the three basic interface components. A window presents a full-size view of one slice through the data set. Mouse operations in this window let the user select and deselect primitive regions for classification. Although the interaction is within one slice, the selected regions extend in three dimensions. Buttons in this window let the user move forward and backward in the unseen dimension 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 lets the user view different slices in the data set. With a frequency button, the user can subsample the range of slices for multiple simultaneous views of slices throughout the entire data set. The third component of the editor is a control panel that governs I/O operations and hierarchical movement.

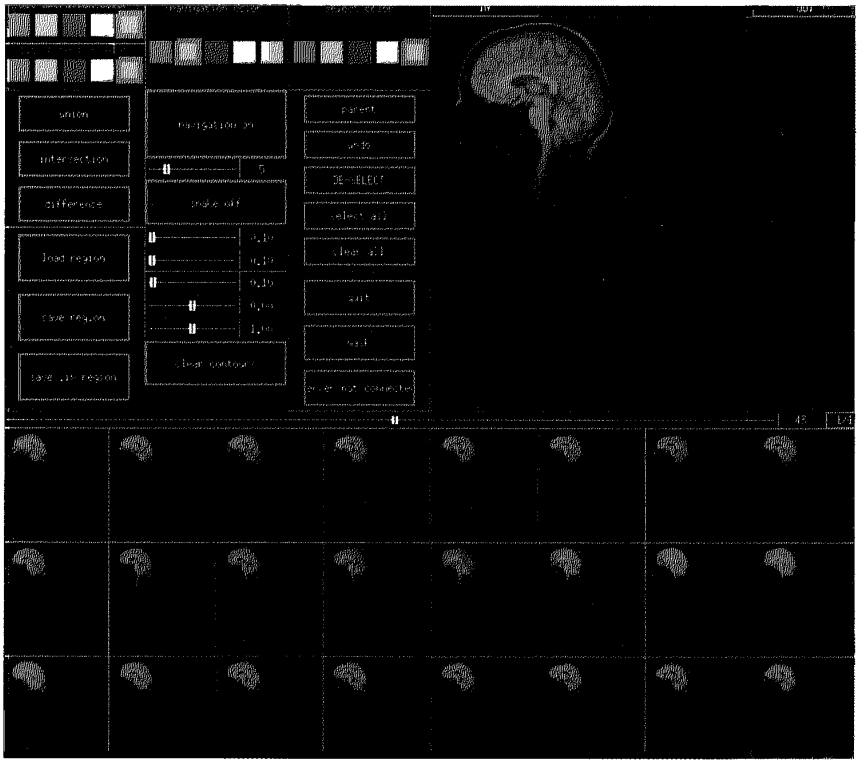


**Figure 6. 3D IHV console window.**

The results with 3D IHV show a dramatic improvement over the earlier slice-by-slice and voxel-painting methods. For example, we created descriptions of a $128 \times 128 \times 128$ data set of a human head and selected 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. 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 four minutes.

## IHV and Pixel-Planes

We combined 3D IHV with volume rendering to create a powerful visualization suite.[31] With segmentation performed by IHV, the user specifies regions of interest to the volume renderer. The user understands the region-selection process better and gains more from overall visualization of 3D volume images.

Using standard Unix sockets, we connected IHV and VRN via an Ethernet, passing image masks and operation codes in transmission-control protocol packets. Sending a selection volume from the IHV application currently takes about 10 seconds. The data is used during ray casting to emphasize selected voxels by modifying their color and opacity. The system currently does not permit the selected data to interact with the raw data before local classification. We will make such interaction possible in the future.

## Application results

Interactive semantic region selection is critical for visualizing medical data sets. Within a data set, many regions may perform different functions but have the same chemical composition or visual appearance; they may even be connected. Similarly, disconnected regions may have similar shape or physiology. For example, the different brain lobes are composed of the same tissue and are all interconnected, yet specialists often distinguish the various regions by the roles they play in cognition, sensing, and motor control. These distinctions are not easily communicated through a computer program that performs segmentation or classification. The best way to image this type of data is to provide appropriate control to users.

Figure 7 shows a visualization possible only with semantic control. The left image is a view of cerebral tissue produced

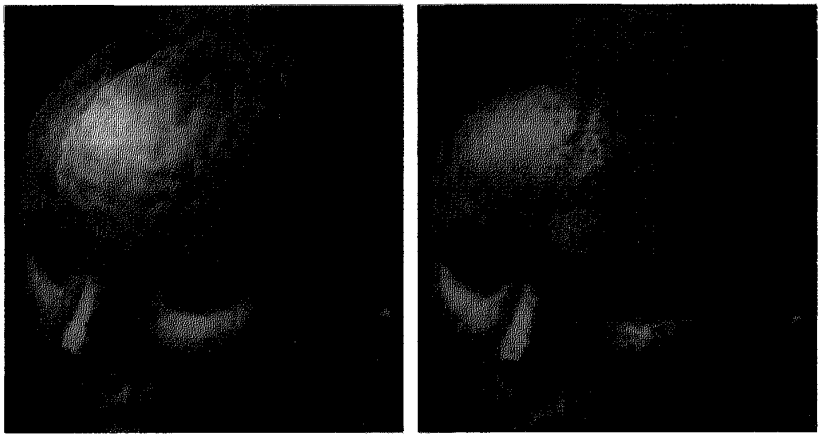

Figure 7. Comparison views with and without semantic classification.

using only geometric and syntactic selection. It is straightforward to classify all tissue of this type within this data set. However, it is impossible using these tools to distinguish the cerebellum, a wedge-shaped organ of the brain located in the rear section beneath the occipital lobe. This organ's tissue is similar to the tissue in the rest of the brain. Using the interactive semantic region selection techniques described earlier, it is simple to specify the organ of interest and separately classify it as different from the remaining tissue. The figure on the right is the same visualization after semantic information is applied.

We do not imply that syntactic tools have no value. On the contrary, semantic and syntactic tools in concert empower the user with visualization control unparalleled in its presentation capabilities. The images in Figure 8 show how tissue selected using the interactive hierarchy viewer (image on the left) can be further classified into different levels of tissue by the application of an intensity window or ramp. The image on the right shows a representation of only the white matter of the cerebral tissue of the data set previously selected on the left. The saggital rift (the division between the left and right brain hemispheres) becomes clear after syntactic classification is applied.

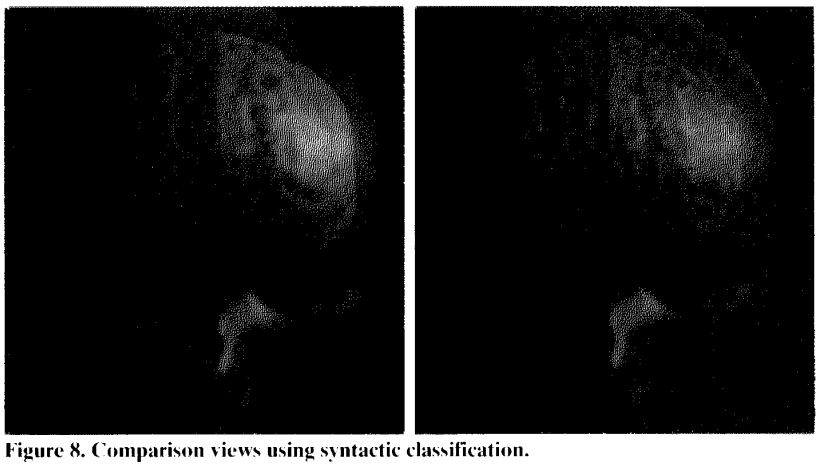

**Figure 8. Comparison views using syntactic classification.**

## Conclusions

The requirement for immediate feedback in volume visualization makes speed a critical issue. By concentrating on optimization of the rendering pipeline in parallel architectures, we are achieving the speeds necessary to provide a natural means to explore volume data. We take advantage of the human visual system's ability to integrate form from motion: Adaptive refinement reduces the number of rays cast per frame during motion of the image and increases the picture quality for still images. The cost for these speeds is some loss of image quality during motion.

Although the speed of hierarchy traversal is a limiting factor, our system demonstrates the advantages of user selection of 3D regions of interest when supported by precomputed sensible regions and immediate display feedback. Our approach reduces by an order of magnitude the time required to select particular regions of medical data. The system also demonstrates the power of interactive control of local classifiers such as clipping and opacity ramps within regions of interest selected by the interactive viewing tool.

Application of the combined systems to real images reveals their great potential for facilitating rapid comprehension of volume data. Even with the limited speeds of the present interactive system, we see the great importance of fast general-purpose parallel graphics systems such as Pixel-Planes 5, and the need for even faster systems with faster data access and rendering algorithms. The speed is not simply a convenience: It results in qualitative increases in user comprehension of data.

When such machines are more widely available and can support a wide range of segmentation and classification techniques at interac-

tive rates of up to 30 frames per second, volume visualization will be a much more powerful approach. When interactive semantic region definition with volume visualization can be achieved in real time on affordable platforms, we think it will be the standard means for viewing 3D data. ❑

## Acknowledgments

We 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, Qin Fang, Dave Eberly, and John Poulton for both their input and the significant contribution that their research represents.

This research has been funded in part by DARPA ISTO contract 7510, NSF grant MIP-9000894, NIH grant no. PO1 CA47982, NSF cooperative agreement ASC-8920219, and Vistanet cooperative agreement NCR-8919038 (DARPA, NSF, GTE Labs, Bell South, CNRI).

## References

1. H. Fuchs et al., "Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories," *Computer Graphics* (Proc. Siggraph), Vol. 23, No. 3, July 1989, pp. 79-88.
2. R. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," *Computer Graphics* (Proc. Siggraph), Vol. 22, No. 4, Aug. 1988, pp. 65-74.
3. M. Levoy, "Display of Surfaces from Volume Data," *IEEE CG&A*, Vol. 8, No. 3, May 1988, pp. 29-37.
4. M. Levoy, "Efficient Ray Tracing of Volume Data," *ACM Trans. Graphics*, Vol. 9, No. 3, July 1990, pp. 245-261.
5. L. Westover, *Splatting: A Parallel Feed-Forward Volume Rendering Algorithm*, doctoral dissertation (also Dept. of Computer Science Tech. Report TR91-029), Univ. of North Carolina, Chapel Hill, N.C., 1991.
6. U. Neumann, "Interactive Volume Rendering on a Multicomputer," *Proc. 1992 Symp. Interactive 3D Graphics*, Cambridge, Mass., pp. 95-100.
7. W. Lorensen and H. Cline, "Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm," *Computer Graphics* (Proc. Siggraph), Vol. 21, No. 4, July 1987, pp. 163-169.
8. P. Schroder and J. Salem, "Fast Rotation of Volume Data on Data Parallel Architectures," *Proc. Visualization 91*, IEEE CS Press, Los Alamitos, Calif., 1991, pp. 50-57.
9. S. Pizer, T. Cullip, and R. Fredericksen, "Toward Interactive Object Definition in 3D Scalar Images," *3D Imaging in Medicine*, NATO ASI Series F, Vol. 60, Springer-Verlag, Berlin, 1990, pp. 83-105.
10. T. Yoo et al., "Achieving Direct Volume Visualization with Interactive Semantic Region Selection," *Proc. Visualization 91*, IEEE CS Press, Los Alamitos, Calif., 1991, pp. 58-67.

**Terry S. Yoo** is a doctoral student in computer science at the University of North Carolina at Chapel Hill, where his research interests are in medical image processing. He worked for BBN Laboratories, AT&T Technologies, and MCNC before beginning the PhD program.

Yoo received his BA in biology from Harvard University in 1985 and his MS in computer science from the University of North Carolina in 1990. He is a site coordinator for the NSF Science and Technology Center for Computer Graphics and Scientific Visualization.

**Ulrich Neumann** is a doctoral student in computer science at the University of North Carolina. His research interests include computer graphics, image processing, and parallel systems. He worked and consulted in industry for nine years before enrolling at the University of North Carolina in 1988.

Neumann received his BSEE and MSEE from the State University of New York at Buffalo in 1977 and 1979. He is a member of the ACM and IEEE.

**Henry Fuchs** is Federico Gil professor of computer science and an adjunct professor of radiation oncology at the University of North Carolina. He is project leader for several research groups in high-performance graphics hardware, 3D medical imaging, and head-mounted display and virtual environments.

Fuchs received a BA in information and computer science from the University of California at Santa Cruz in 1970 and a PhD in computer science from the University of Utah in 1975. He recently received the 1992 NCGA Academic Award.

**Stephen Pizer** is Kenan professor of computer science and adjunct professor, radiation oncology, radiology, and biomedical engineering at the University of North Carolina. He heads the multidepartmental Medical Image Display Research Group. His research interests include human and computer vision, interactive 3D graphics, and contrast enhancement.

Pizer received a BA in applied mathematics from Brown University in 1963 and an MA and PhD in computer science from Harvard University in 1964 and 1967, respectively. He is an associate editor of IEEE Transactions on Medical Imaging and a senior member of the IEEE.

**Timothy J. Cullip** works in the Radiation Oncology Department of University of North Carolina Hospitals. His research interests include interactive rendering of dynamic volume data and real-time radiation dose computation.

Cullip received his BS in mathematics and MS in radiological physics from San Diego State University in 1979 and 1981. He received his MS in computer science from the University of Central Florida in 1986.

**John S. Rhoades** is a doctoral student in computer science at the University of North Carolina. His research interests include volume-rendering algorithms, texture algorithms, and modeling algorithms for curved surfaces. He worked at the NASA Johnson Space Center on postflight data reduction for the space shuttle and in the Mission Planning and Analysis graphics laboratory before becoming a full-time graduate student.

Rhoades received his BS in mathematics from Rice University in 1969 and his MA in mathematics from the University of Houston in 1976.

**Ross Whitaker** is a doctoral student in the Computer Science Department at the University of North Carolina. He is currently working on techniques for image segmentation that use nonlinear diffusion. He spent two years as a management consultant at the Boston Consulting group before enrolling in the graduate program at the University of North Carolina.

Whitaker received a BS in electrical engineering summa cum laude from Princeton University in 1986.

The authors can be reached at the Department of Computer Science, University of North Carolina, Chapel Hill, NC 27599-3175.