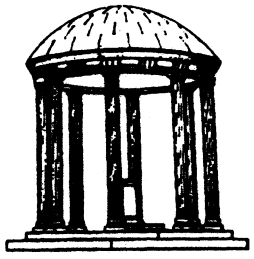
# Methods for Direct Visualization of 3D Volume Data

**TR91-011**
**February, 1991**


**Terry S. Yoo**
**Ulrich Neumann**
**Henry Fuchs**
**Stephen M. Pizer**
**Tim Cullip**
**John Rhoades**
**Ross Whitaker**

**Medical Image Display Group**
**Department of Computer Science**
**Department of Radiation Oncology**
**The University of North Carolina**
**Chapel Hill, NC 27599-3175**


*The research reported herein was carried out with the partial support of NIH grant number P01 CA47982.*
*Submitted to Information Processing Medical Imaging.*

# Methods for Direct Visualization of 3D Volume Data

*Terry S. Yoo*[1], *Ulrich Neumann*[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

We present evidence for the power of interactive volume visualization and describe capabilities that are needed to support this approach. These capabilities include

1) real time control of viewing properties and image features to be presented combined with fast rendering for immediate feedback;

2) powerful means of selecting regions of emphasis;

3) a natural user interface controlling the viewing, display, and region selection parameters.

We describe a sophisticated graphics hardware platform, Pixel-Planes 5, advanced rendering algorithms on this platform, and an interactive region selection system capable of both sculpting and semantic classification. We combine these elements to create a high quality environment for the exploration of 3D volume data. We have achieved frame rates of 2 to 20 frames per second of a 128x128x56 data set at a resolution of 640x512 pixels. These methods and systems allow the presentation and appreciation of complicated internal structures of the body in around 3 minutes from the loading of the 3D image data.

## 1. Introduction

According to the Oxford English Dictionary, diagnosis is the process of determining by examination the nature and circumstances of a diseased condition. By its very definition, diagnosis is an exercise of exploration. However, exploring volumetric data is a difficult task; a physician cannot assimilate all of the information from a single view of the data. Diagnosis using volume data becomes a two part process: navigating the data, and extracting information by interacting with the viewing parameters. Moreover, both the navigation and visualization control should be as natural as possible.

Since data exploration is a feedback process, effective volume visualization techniques require display updates at interactive rates (multiple frames per second) to get meaningful comprehension. The researcher or physician is actively exploring the information and making changes in the viewpoint, seeking things not readily apparent. 3D perception is essential to proper understanding; the effects of motion are among the strongest cues available and are important to correct and quick perception. Interaction is critical to speed the clinician's ability to focus on detail. To achieve a natural interface that encourages and enables exploration, the response to viewpoint changes must be immediate to show whether the change enhances perception or detracts from it. A perfect display can respond to aspect and portion requests in real time.

Effective visualization methods should also include an arsenal of tools that help the clinician to select both what to display and how to display it. Data selection tools pare away detail extraneous to a particular view of the

image Such tools include: sculpting tools, syntactic classifiers, and semantic specifiers. Sculpting and clipping controls remove unwanted portions of the image. Syntactic controls use image properties to enhance or de-emphasize image features, and semantic controls allow the viewer to use knowledge about all possible scenes to quickly define particular regions of importance.

Viewing controls should include interactive handles on light directions, specularity, color, opacity, etc. to assign properties to objects for maximum comprehension of shape, form and extent.

This paper presents our current research on interactive volume visualization It describes the display engine, Pixel-Planes 5, used to create the interactive high quality graphics environment for this research. We present the variations of volume rendering that are being explored, followed by a presentation of classification/selection methods. We include a discussion of the implementation and our preliminary findings regarding the different methods. We include images and results from volume studies that have been performed using these tools and some that are still in progress.

We do not include a general survey of all visualization techniques here. There are other groups exploring surface rendering along with advanced segmentation techniques. Because volume rendering is a superset of surface rendering, it allows more flexible control of the real time image. In spite of its speed disadvantage the work presented here has centered around volume rendering.

## 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 [Fuchs]. The machine is an experimental heterogeneous architecture suitable as a platform for a wide range of parallel algorithms research.

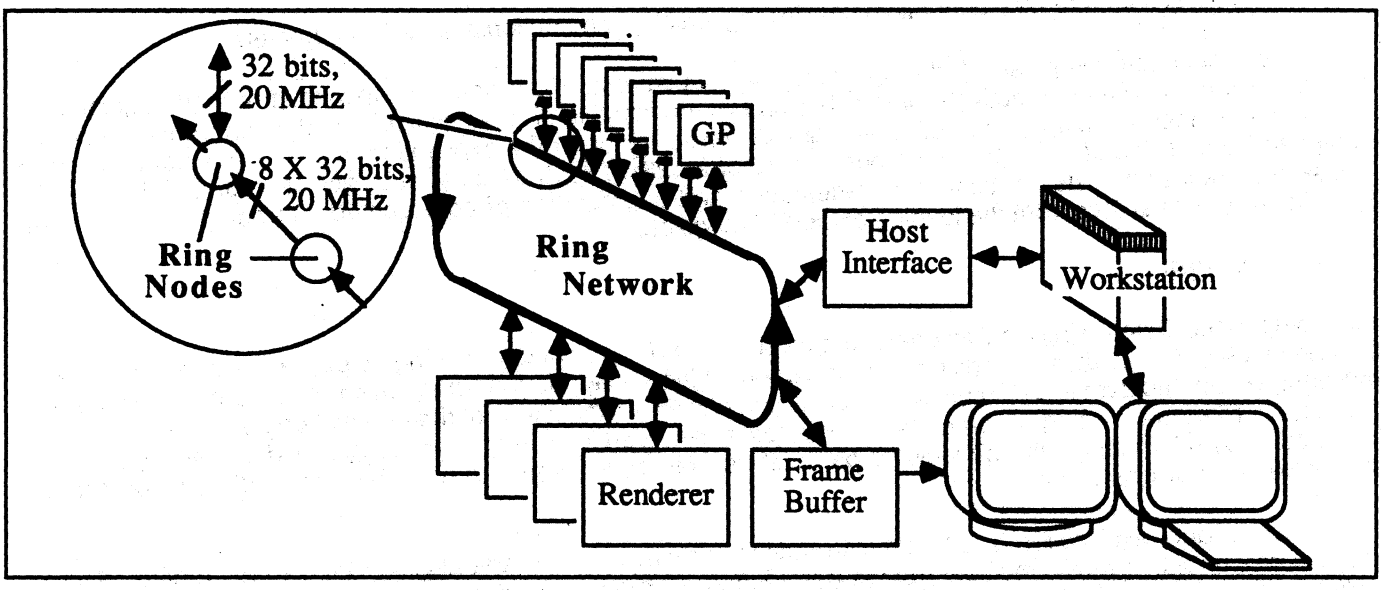



Figure 1: Overview of Pxpl5 System, showing Ring Network and Ring Devices that include Graphics Processors (GP's), Renderers, Host Interface, and Frame Buffer.

The system provides both MIMD and SIMD parallelism. MIMD parallelism is provided by the Graphics Processor boards, 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 sent from the Graphics Processors. In the volume rendering program, classification and shading of voxels is done by a SIMD algorithm, and screen space subdivision is done in a MIMD fashion for parallelizing the raycasting.

The flexibility of Pixel-Planes 5 becomes apparent in its ability to merge polygon and volume rendering. Using Levoy's [Levoy] approach, ray casting is used to merge polygons and volume data.

The major board level design elements, shown in Figure 1, are:

- **Graphics Processor (GP):** 40MHz Intel I860™ general purpose microprocessor each with 8MB of memory. The fully configured system contains 32 GP's (16 boards with two GP's per board.)
- **Renderer:** 128x128 SIMD array of pixel processors with its own 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. The fully configured system contains 16 Renderer boards.
- **Frame Buffer:** 1280 x 1024 x 24bits double-buffered, with display refresh controller.
- **Host Interface:** supports communications to/from a UNIX workstation.
- **Communication Ring:** high bandwidth message passing general purpose inter-board communication link. A bandwidth of 160 megawords per second is provided by 8 time sliced 20 MHz channels .

The fully configured Pixel Planes 5 system contains 16 Renderer boards, and 32 Graphics Processors (16 boards with two GP's per board.)

## 3. Variations of 3D Visualization Techniques

For many years, most of our 3D visualization research has centered around interactive techniques. Using Pixel-Planes 5, we made volume rendering at interactive rates a reality. In addition, we are adding other methods to our research effort. In particular, interactive computed radiographs have begun to show promise as a visualization method.

This section presents two methods of 3D visualization implemented on Pixel-Planes 5: computed radiographs and volume rendering. We present not only the general structure of the techniques, but we also outline the implementation and provide information regarding the interactive speed of the display methods. Two approaches to volume rendering are also presented.

### 3.1. Computed Radiography

Computed radiographs are the synthetic reprojection of x-rays through a CT data set to obtain an image that simulates an actual x-ray film image. This is not a new technique; however, we have been working to create these images in real time. Traditionally, radiation treatment planning uses a real x-ray of the patient taken from the beam's eye view to determine if the beam has been aligned properly. This x-ray can show how the beam's outline lies in relation to important organs and the tumor. The x-ray is taken, as a safety check, after the beam orientation and treatment plan have been determined.

Our radiation oncology department has developed a computer simulation package called the Virtual Simulator [Sherouse] which allows interactive computer simulation of the treatment planning process. We may simulate a radiograph by casting rays through the CT data set from the beam's eye view onto a simulated film plane. This gives the physician the flexibility to try many different beam orientations and view the results without the patient being exposed to a real x-ray for each orientation. Until recently, computed radiography has been slow, taking up to 30 minutes on a typical workstation. Therefore, it was not available during the interactive planning phase, but only as a last step after the beam had been placed.

Using Pixel-Planes 5, we are now able to produce computed radiographs at interactive rates as high as 20 frames per second. This capability offers several options not previously available. First, we can now view the process as dynamic computed fluoroscopy rather than as static computed radiography. This allows the physician much greater flexibility to explore non standard radiation beam orientations in real time. Secondly, the interactive capability alleviates a major weakness of traditional x-rays; the collapsing of 3D data into a 2D plane without surfaces occluding one another makes it very difficult to discern the 3D structure of the scene. With real time interaction we gain back much of the 3D information simply by dynamically moving the view point. The lack of surface occlusion can be advantageous, since important structures are not hidden from view. Finally, the fact that radiation oncologists are historically used to viewing radiographs makes this visualization technique very natural to them.

### 3.1.1. Implementation

The basic technique used in computed radiography is to cast rays through the 3D data set, sample the data along the rays, and finally projecting that information on a 2D viewing plane. In computed radiography, the attenuation coefficients sampled along a ray are simply summed. Since computed radiography does not require the shading, and each GP has enough local memory to hold the entire 3D data set (up to a 256x256x64 CT dataset), we simply replicate the data at each GP, letting each GP be responsible for rendering a portion of the output image. The major communication flow occurs between the GP's and the frame buffer and only involves the transfer of the 2D output image.

In order to mimic the image of an x-ray we use the attenuation equation:

$$\text{intensity} = 1 - e^{-(u_1 x_1 + \ldots + u_n x_n)}$$

where:

$u_i$ : linear attenuation coefficient for sample i along the ray.

$x_i$ : path length of the ray between sample i and i+1.

For a given ray, the step size between samples is constant so we can factor out the $x_i$'s, therefore the sampling along a ray only involves the summation of the attenuation coefficients. The attenuation coefficients are obtained from the original CT numbers by a table lookup that represents the following equation:

$$u_i = (1000 \cdot CT_i \cdot u_w + u_w) \cdot R(CT_i)$$

$CT_i$ : CT value for sample i.

$u_w$ : linear attenuation coefficient for water.

R(CT) : bimodal conversion ratio dependent on the classification of the CT as bone or water (converting the total attenuation coefficient to a photoelectric attenuation coefficient)

In order to achieve high frame rates, image quality is sacrificed during motion by only casting one ray for every 8x8 block of pixels and interpolating the values for the intermediate pixels across the image. Once the motion is stopped, a more refined picture is computed within 1/5 second (one ray per 4x4 pixel block) or 3/4 second (one ray per 2x2 pixel block). The highest frame rate is achieved by using nearest neighbor sampling along a ray, while the refined images use interpolation of surrounding voxel values to compute ray samples.

Figure 2: Computed radiographs... sequence in real time

### 3.2. Volume Rendering

In contrast to the simple raycasting and compositing calculations of computed radiographs, volume rendering produces a shaded image from a 3D array of data samples. Three classes of volume rendering/modeling techniques are: Surface Representation [Lorenson], Binary Classification [Herman][Kaufman], and Multi-valued Classification (MVC) [Drebin][Levoy]. 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 needs.

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] to yield the shaded R, G, B values.

Resampling of the shaded array is required to project the volume onto a 2D view plane. Ray casting with linear interpolation is the method we use, but only one of many [Drebin] [Westover] possible methods.

Compositing [Porter] after each resampling step computes opacity and color accumulation. Resampling must proceed front-to-back or back-to-front for compositing to work correctly.

#### 3.2.1. Replicated Data Implementation

The high frame rates achieved with computed radiography has led us to consider using its GP-only approach to volume rendering. We have developed a volume renderer that replicates the data set at each GP and achieves update rates of 15 frames/second. In this approach, the GP's are responsible for voxel shading. To reshade the voxels rapidly when the light or view point changes, an efficient approach to shading must be used. As a preprocessing step, each voxel has its normal computed and stored 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 each frame using a Phong shading model. This results in only $2^{13}$ shading computations rather than the $2^{21}$ that would be needed for a 128 cubed data set if the voxels were shaded directly.

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

8 bits : original CT or MRI data

13 bits : normal encoding

8 bits : gradient magnitude

3 bits : currently unused.

This look up table shading maintains fast update rates for a volume renderer with a standard gradient based opacity shader. The currently unused bits may be used to perform simple classification schemes in the future. We are still investigating the strengths and weaknesses of this approach as compared to the distributed data volume renderer described next.

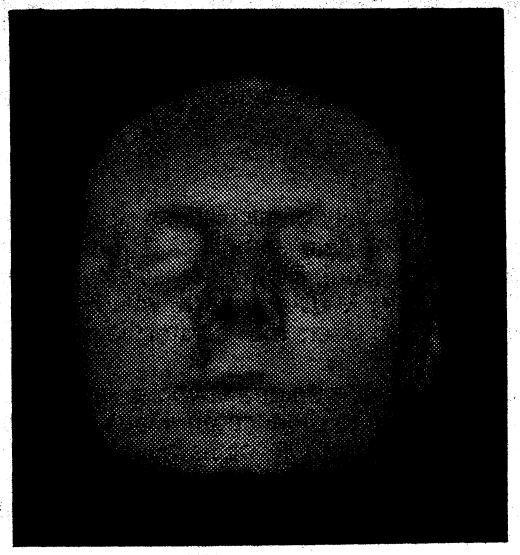


Figure 3: Image produced using table driven volume renderer

### 3.2.2. Distributed Data Implementation

A general purpose volume rendering system was developed in the Summer and Fall of 1990 for the Pixel-Planes 5 system by John Rhoades and Ulrich Neumann, based on a design concept by Marc Levoy [Levoy]. Its major difference with respect to the system previously described is that the data set is distributed among the renderers; it is not replicated throughout the system. Therefore, this system can display 3-D data sets up to 256x256x256 voxels in size. Additionally, it does classification and phong shading of the data using the renderers. The data set position as well as several lights can be manipulated via joysticks. It provides a flexible local classification scheme for converting intensity and intensity gradient magnitude to color and opacity also under joystick control. A moveable cutting plane is provided. The system can also input selection masks from an external source, such as IHE, to highlight selected voxels. The current system can update the display at about 1.4 frames per second with a 128x128x128 data set using 7 Renderers and 16 GP's.

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. The renderers store the data base in their backing store, perform conversion to color and opacity, and transmit shaded voxels (with color and opacity) to the GP's. The GP's are divided between a single master GP and a number of slave GP's. The master GP controls the renderers and manages the slave GP's. The slave GP's perform the ray casting algorithm on their assigned

portions of the image and transmit the image to the frame buffer. Typical hardware configurations contain 16-24 GP's and 7-18 Renderers.

### 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) 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 GP makes a global assignment list of macros to slave GP's. A macro contains 8x8x8 voxels.

(4) The slave GP's commence ray casting to create their image tiles. When a slave GP needs a macro which it doesn't have, it sends a "fetch macro" request to the master GP. The master GP responds by instructing the render that has the macro data to classify and shade the backing store sector containing the macro. Then the master GP instructs the renderer to transmit all the macros in that backing store sector to the slave GP's 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 are ignored. One backing store sector holds 32 macros.

(5) After the slave GP's have completed all their ray casting, they fill in the screen pixels via screen space linear interpolation. Then they send their image tiles to the frame buffer and notify the master GP that they have finished.

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

### Optimizing the Ray Casting

The slave GP's 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 skip the rays quickly over transparent regions, and to avoid performing the trilinear interpolation algorithm in transparent regions. The renderers assist in computing the octtree by marking macros which contain only transparent voxels. The octtree helps a lot, as typically 2/3 of the macros are completely transparent. Rays are discontinued when they accumulate enough opacity such that further processing would yield little color change ("alpha cutoff"). Rays are cast adaptively. At first a coarse grid of rays is cast. Then, if the adjacent values are sufficiently dissimilar, more rays are cast. The final image is formed by bilinear interpolation of the ray-cast pixels. The use of the cutting plane adds no cost, but in fact makes the system faster. The cutting plane is implemented by starting the rays at the cutting plane instead of the data set edge. Typical time for a slave GP to render a 128x128 tile, for a 128x128x128 data set with coarse refinement, is 100-240 ms.

### Controlling the Renderers

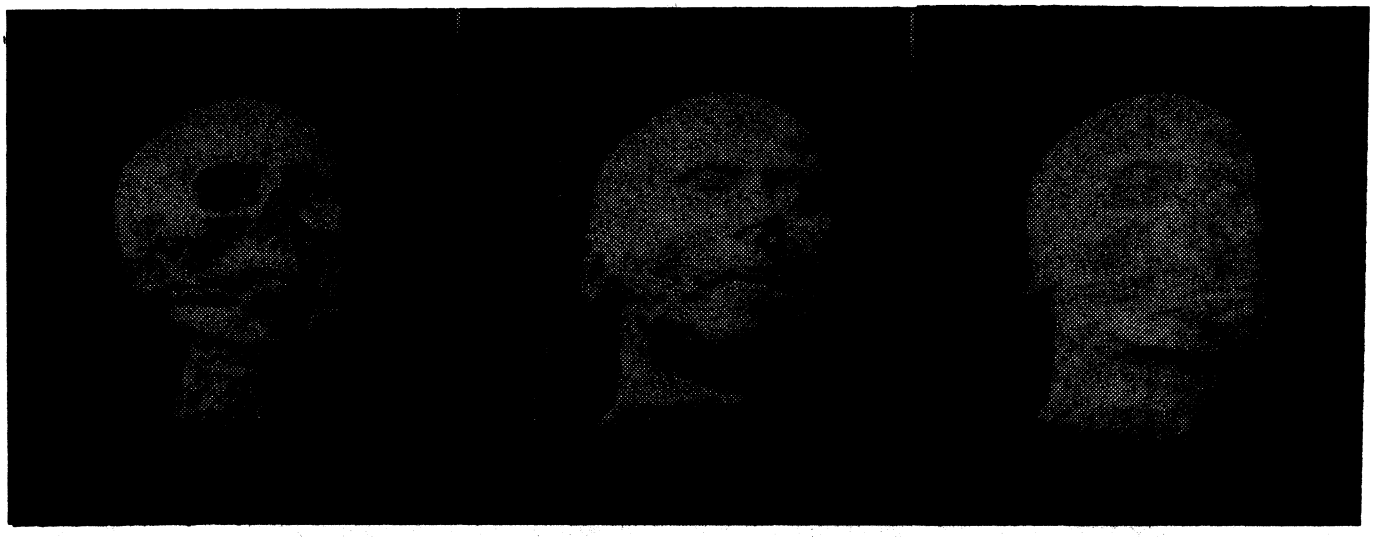
The master GP implements a finite state automaton for each renderer to make the task of controlling all the renderers in parallel, while receiving asynchronous fetch requests, manageable. The main states are:

1. Loading backing store
2. Classifying and Shading
3. Sending macros to slave GP's
4. Waiting on slave GP acknowledgement
5. Done

For a 128x128x128 data set a renderer typically spends about 120 ms shading and 100 ms sending macros to the slave GP's. The other states together typically take 30 ms.

### Performance

The current frame rate of 1.4 frames per second isn't really adequate for 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. We find it difficult to obtain good load balancing among the slave GP's, and operating system overhead for message passing among the GP's is higher than expected. The ring bandwidth is sufficient for our desired level of performance, so more efficient low level communications routines must be developed. The use of a single master GP introduces latency and thus can be 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 find ways to speed up the algorithm. Then we expect to devote some serious effort to optimizing the code.



bone skin bone+skin

Figure 4: Three Views Using Local Classification Techniques

## 4. Viewing and Classification

Control of the viewing parameters extend to more than just moving light sources and affecting the color of objects depicted in the image. In order to best understand volumetric data, a visualization program should also allow the user to emphasize specific elements of the image.

We discuss control of the viewing and classification parameters. We divide classification properties into two broad categories: local or syntactic classification methods in which properties of the image determine the orientation of the viewpoint, and global or semantic methods where knowledge of anatomy and pathology allow the selection of morphological features to be distinguished. Both approaches are discussed here as well as their inherent advantages and built-in limitations. As before, we include implementation information and performance measurements.

### 4.1. Viewing Controls

The data set can be translated, rotated, and scaled. Up to three light sources are provided. The light sources are selected to be directional or ambient. Controls are provided for direction and intensity of each color. Lights do not move when the data set is rotated (i.e., lights are attached to the user space, not the data set space.) The amount

of image refinement is selectable (i.e., how many rays are cast). A higher level of refinement is performed when the image isn't changing.

There is a cutting plane which can be translated and rotated. Its effect is to make all the voxels on one side completely transparent, so the user can cut away part of the data set to see inside. Voxels are displayed on the cutting plane Phong shaded, but we are working on displaying a grey scale slice on the cutting plane, with the rest of the image Phong shaded. We find it difficult to determine the orientation of the cutting plane, and therefore plan to add an indicator to the display showing where the cutting plane intersects the data set bounding box.

### 4.2. Local Classification Techniques

The user's interface to the volume renderer is via a control panel for inputting numerical values and options, joysticks for interactive control. The interface provides controls for classification and display options. When the system is started, all the control values are initialized by reading a configuration file. Thereafter the user can change the control parameters either by manipulating joysticks or typing values into the control panel.

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 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 via 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.

The inherent limitations of local classification arise if the area or organ being viewed has a similar intensity and intensity 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.

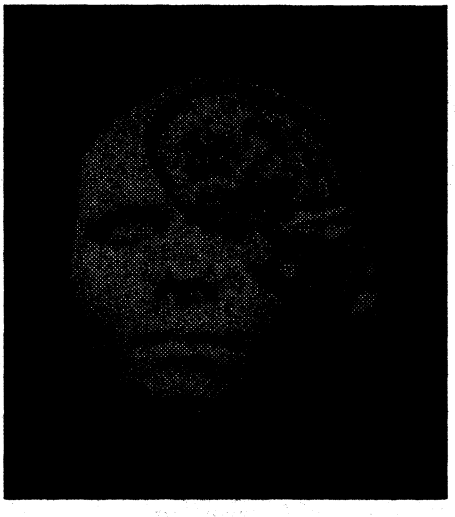


Figure 5: head + cutplane

### 4.3. Global Classification Techniques

Since local properties of the data do not reveal information regarding morphological features of the image, it is often useful to apply some 'semantic' groupings of a given data set. The way in which these groupings relate to global structure are most effectively determined by a knowledgeable user. What is required is a language of sensible fundamental regions and a means of interacting with them.

We have shown that providing a user with a segmented dataset and an interface with which to combine and edit regions can greatly decrease the effort of specifying regions of interest. In 2D images, we perform the segmentation by forming a hierarchy of sub-regions within the image using one of many techniques [Pizer]. 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 are extending 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 and editing the images using the resulting descriptions and a 3D editor.

### 4.3.1. Hierarchy Generation

We are using a reverse gravity watershed algorithm which chooses primitive regions based on local maxima in a 3x3x3 neighborhood. It then grows each region by examining successively lower (discrete) intensities, and includes adjacent voxels at each intensity with their respective regions. When a voxel at a given intensity touches more than one region, then the appropriate regions are joined at the next higher level in the hierarchy. This algorithm produces primitive regions and the accompanying hierarchy in a time proportional to the product of the number of discrete intensities and the number of voxels. On a DEC3100 the description of a 128x128x128 image with 256 gray levels can be produced in roughly 15 minutes.

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

### 4.3.2. The Interactive Editor

The 3D interactive hierarchy editor (3D IHE) provides the user with mouse-based interactions and a view of the multiple slices of the dataset. The interface contains three basic components. First, a full size view of one slice of the dataset. Mouse clicks and drags in this window allow the user to select/deselect primitive regions for classification. Although the interaction is with 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 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 methods. For example, we created descriptions of a 128x128x128 mri dataset of a human head, and choose the task of selecting the cerebellum 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 cerebral cortex can be selected in approximately 4 minutes.

Although the current 3D segmentation is very powerful it warrants further research. The growth of regions in 3D dimensions provides a great many directions in which regions can connect and can result in undesirable connections between regions. The 3D watershed is prone to 'leaking' so that two large distinct regions can connect because of a very small area of one slice. Also, watershed regions vary greatly in size. Some are so small that they are of little value to the user, and tend to unnecessarily increase the complexity of the hierarchy. Future work will look at alternative ridge based descriptions that could improve the efficiency of user interaction and region definition.

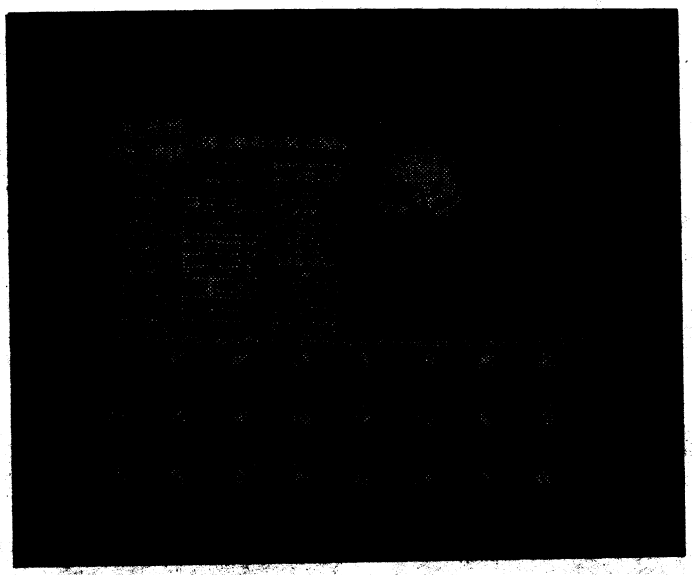


Figure 6: 3D IHE Console Window

### 4.3.3. The IHE to Pixel-Planes Connection

We combined 3D IHE with volume rendering to create a powerful visualization suite. We want to see immediate reactions to IHE selection in the volumetric display. The results are both engaging and provoking. We obtain better perception of the region selection process and subsequently aid the overall visualization of 3D volume images.

These two diverse systems are connected using standard Unix sockets via an ethernet, passing image masks and op-codes in TCP packets. The systems pass a voxel mask via the network. Sending a selection mask from the IHE application currently takes about 10 sec. The mask is used to emphasize selected voxels by modifying their color and opacity. The current design doesn't permit the selection mask to interact with the raw data. before local classification.

## 5. Assessment

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. Real-time control of viewpoint, dynamic change of viewpoint, and immediate feedback are so powerful that even the somewhat limited innate ability of fluoroscopic projection to communicate 3D comprehension is significantly enhanced. In addition, 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 the image data.

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. If the slices are to be superimposed, they must be restricted to object-based ROIs.

Exploration implies a rich collection of operations, each with immediate feedback and a natural interface. Thus our 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. They also demonstrate the power of analog control, such as clipping and selection of opacity ramps, within ROIs selected by this interactive method. Finally, they demonstrate the great importance of fast 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.

Explorational control of many parameters is anathema to clinical users. This concern can only be overcome with a natural, powerful user interface. Our present collection of joysticks, with functions switchable from control panels, combined with X-window sliders and menus, is not satisfactory. We are working toward a far improved interface based on natural actions and perhaps a "chord" selector, organized by ROI, i.e., image object. For each control, immediate feedback on the image is critical.

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. This level of visualization is now possible in a couple of minutes by exploring parameter values using immediate feedback.

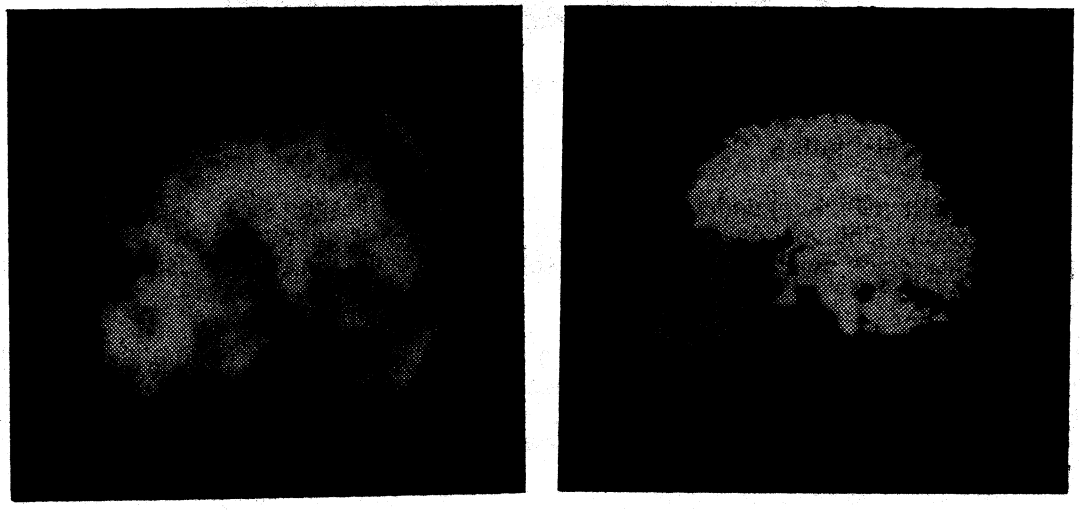


Figure 7: Comparison Views
With and without global classification

**A note to the review committee**

We are presently working on various aspects of this research. Much of what we are trying to accomplish should be complete by the time final submissions are due. If this paper is accepted and new results significantly augment the findings of the paper, we hope to include some of them in a subsequent draft or in the final presentation at the meeting.

Efforts currently underway include:

- variations of computed radiographs: synthetic reprojection of single photon emission computed tomography
- Splatting on Pixel-Planes 5 as another volume rendering technique
- exploring the table driven volume rendering version for viability in long term research
- incremental updates of volume rendering mask from IHE regions

## Acknowledgements

We would like to thank Bill Oliver for his help with the text and a 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. We also thank Julian Rosenman for his expert advice.

This research has been funded in part by the following grants
**Defense Advanced Research Projects Agency** ISTO order No. 6090
**National Science Foundation** Grant No. MIP-8601552
**National Institutes of Health** Grant No PO1 CA47982

## References

Drebin, Robert A. , et. al. 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., B. Oliver and S.M. Pizer, "Multiresolution Shape Descriptions and Their Applications in Medical Imaging." *Information Processing in Medical Imaging* (IPMI X, June 1987): 131-150, Plenum, New York, 1988a.

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, 1988b.

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.

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.

Sherouse, G. W., C. E. Mosher, K. Novins, J. Rosenman, E. Chaney. Virtual simulation: Comcept and implementation. In: The Use of Computers in Radiation Therapy, *Proceedings of the Ninth International Conference.* Schgeveningen, The Netherlands: North Holland Publishing Co.; 1987: 429-432.

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