3DIVE: AN IMMERSIVE ENVIRONMENT FOR INTERACTIVE VOLUME DATA EXPLORATION

Michael Boyles
Indiana University
Indianapolis, USA

Shiaofen Fang
Indiana University Purdue University Indianapolis
Indianapolis, USA

ABSTRACT

This paper describes an immersive system, called 3DIVE, for interactive volume data visualization and exploration inside the CAVE virtual environment. Combining interactive volume rendering and virtual reality provides a natural immersive environment for volumetric data visualization. More advanced data exploration operations, such as object level data manipulation, simulation and analysis, are supported in 3DIVE by several new techniques: volume primitives and texture regions are used for the rendering, manipulation, and collision detection of volumetric objects; the region based rendering pipeline is integrated with 3D image filters to provide an image-based mechanism for interactive transfer function design; a collaborative visualization module allows remote sites to collaborate over common datasets with passive or active view sharing. The system has been recently released as public domain software for CAVE/ImmersaDesk users, and is currently being actively used by a 3D microscopy visualization project.

1. Introduction

Volume visualization is concerned with the abstraction, rendering and manipulation of volume data sets, such as medical imaging scans and samples of computed/simulated systems. Due to the 3D nature of the volume data sets, desktop interface does not provide sufficient 3D sense and experience for effective volumetric data exploration. Virtual reality based immersive environments offers a much more powerful and intuitive 3Dinterface for volume-based visualization applications. Although a number of commercial and research volume visualization systems have previously been developed [1, 2], the use of virtual reality in volume visualization has not been fully exploited. The Crumbs package [3], from the NCSA, relied on the CAVE virtual reality environment for display and interaction. Crumbs was primarily developed for the exploration of biological datasets. The user could drop “crumbs” inside the dataset and later explore along the constructed spline path. It also offered slicing planes, color map editor and access to remote supercomputers where image processing could be performed. Similar to Crumbs is a system called CAVE5D [4]. This is basically a CAVE implementation of the Vis5d visualization system [5]. CAVE5D supports any _le formatted for Vis5d including topography, map, iso-surface and multiple trajectory sets. The newest version also supports time-varying scalar data. These existing systems are, however, mainly focused on the immersive data viewing process. More advanced data exploration operations, such as object level data manipulation, animation, analysis and collaboration, have not been studied.

In this paper we describe an environment for volume visualization and exploration called 3DIVE (3 Dimensional Interactive Volume Explorer). 3DIVE employs a region-based approach for volume rendering and data exploration. It is built upon the following characteristics:

- *Immersive.* 3DIVE uses the CAVE virtual environment for interactive, stereoscopic display and manipulation so as to o_er the most realistic and intuitive interactions possible. Inside the CAVE, a user may walk around or even bend down and look under the volume data, and using an input device, the user can grab a portion of the data set and move it by twisting their own arm or hand. A 2D menuing system allows for easy control of parameters and state changes.

- *Region-Based.* In 3DIVE, objects are considered as regions of data sets and represented as collections of volumetric primitives. This allows the user to manipulate and visualize individual portions of the data set, and apply different transfer functions or data filters to individual objects.

- *Data filtering*. In 3DIVE, 3D image processing procedures can be chained together to form a data filtering sequence, which can then be applied as a transfer function of the volume rendering process. Thus, the users no longer need to pre-process their data prior to viewing, and will be able to interactively choose the appropriate filters for desired visualization effects.

- Collaborative. Collaboration allows multiple people to share the same data sets and collaborate within the same visualization environment. 3DIVE was designed with collaboration in mind. Users can easily join/leave a session, share views and discuss their opinions via a separate audio link. 3DIVE offers participants both active and passive roles in the collaboration process through a token sharing mechanism.

Figure 1 provides an overview of the 3DIVE system. The three main components include: (1) visualization, (2) virtual interface, and (3) collaboration. Section 2 describes the region-based volume visualization approach, including the rendering algorithm, data filtering by image processing and a volumetric collision detection algorithm. The immersive interface and interaction tools using the CAVE system are discussed in Section 3. Section 4 describes the collaboration module, followed by the implementation details and closing remarks.

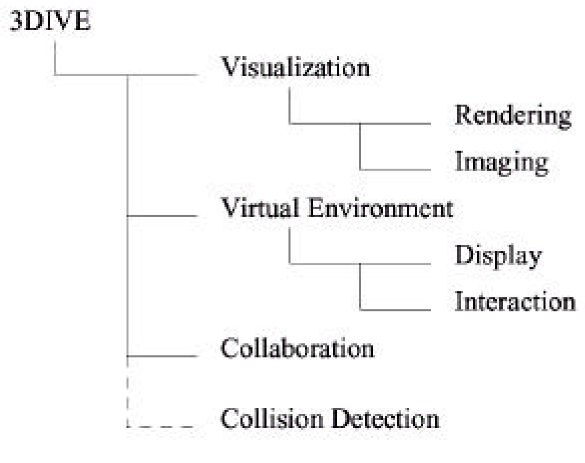

Figure 1: 3DIVE Overview

2. Region-Based Volume Visualization

A volumetric object is essentially a region of a volume data set. It can be represented by a collection of volume primitives, typically tetrahedra, defined within the volume data set space. When an object's volume primitives are geometrically transformed , the data content (the volume data originally defined within the primitives) will be carried along with the primitives. This can be realized using a 3D texture mapping mechanism, similar to the 3D texture mapping implementation in SGI's Volumizer [6]. I.e. the vertices of the primitives will carry texture coordinates, which do not change when the primitives are geometric transformed. Since any 3D region can be tessellated into tetrahedra that geometrically approximate the region, this approach can essentially represent any volumetric object. At present, such objects are defined interactively by the user through some simple region cutting tools. In principle, however, objects obtained from 3D segmentation can also be represented this way.

Conceptually, volumetric objects are independent entities that have their own properties including color maps, geometric transformations, data filters (image processing operations) and opacity scales. Using 3D texture mapping, only one copy of the data set needs to reside in the system even if multiple instances (with different transfer functions and transformations) of the same region can be generated and rendered simultaneously. This would be a useful tool for comparing the results of different parameter settings. Furthermore, regions or objects defined from different data sets can also be visualized, combined, and manipulated together. Various blending functions can be used to create desired volume overlapping effects. One practical application of this tool is to visualize multi-parameter image volumes from confocal microscopy, in which several different proteins within the same specimen are imaged simultaneously, each in a specific color of fluorescence. In this case, multiple data sets are generated for the same set of physical objects, which need to be visualized both separately and combined.

2.1 RENDERING ALGORITHM

Our algorithm is similar to Volumizer in that it also uses a slicing mechanism in conjunction with 3D texture mapping to facilitate the coloring of voxel information [7, 8]. However, instead of depth sorting the resulting polygons, our algorithm borrows concepts from the 2D scanline procedure (found in [9]) and constructs and renders polygons slice-by-slice.

The slicing process proceeds by moving a z-plane, the plane parallel to the viewing plane, with a fixed step size. Within each slice, the intersecting polygons between the z-plane and the volume primitives will be computed and rendered by 3D texture mapping. The process of computing the intersecting polygons can be considered as a 3D extension of the 2D scan conversion algorithm.

To determine which primitives (if any) a slice intersects, we borrow the concepts of the *y-bucket* list and *active list* from the 2D scanline algorithm, and construct a *z-bucket* list that contains the closest (denoted as *min*) and farthest (denoted as *max*) vertex of each primitive in the viewing coordinate system. These vertices are then sorted using a standard quicksort. The *active list* is a dynamic list of primitives that keeps track of all primitives that intersect the current slice. Using the z-bucket list, a primitive is added to or removed from the active list whenever a max or min is encountered respectively. Figure 2 shows a diagram of the process and below that is a procedure that summarizes the algorithm.

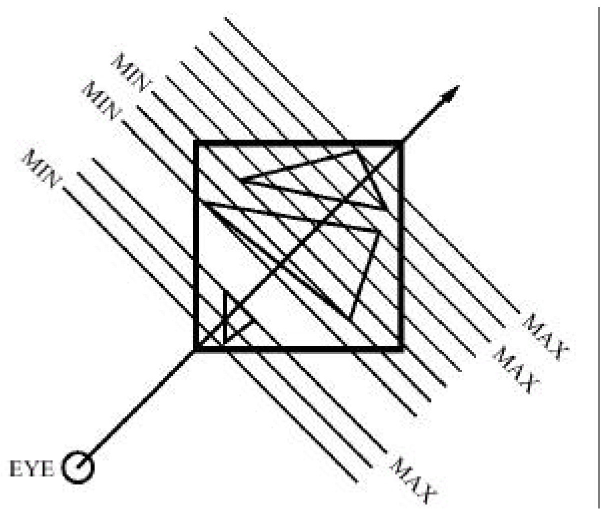

Figure 2: Volume Primitive Rendering

```
PROCEDURE volumeRender
{
        Create an empty list of real numbers called the z-bucket list
        Create an empty list of primitives called the active list

        Compute the viewing normal
        Compute the Z-plane step size

          for (each primitive)
          {
                    // Using viewing coordinate system
          min = depth of near corner of volume
                    max = depth of far corner of volume

          Add min and max to the z-bucket list
        }
          Sort z-bucket list in increasing order
          Set current point to last value in z-bucket list

        do
          {
                    Grab the last value (the largest) from z-bucket list

          while (current point <= current value)
          {
                    if (current value is max)
                              Add its owner to active list
                    else if (current value is min)
                              Remove its owner from active list
                    Set current value equal to next value in list from z-bucket list
          }
```

```
                Compute a Z-plane orthogonal to viewing normal using current point
                for (each primitive in active list)
        {
                Enable clipping planes and other properties for that primitive
                Determine the intersection of Z-plane and volume
                Draw Z-plane with 3D texture map
                Using a blending function, composite this plane in framebuffer
                }

        Compute current point by subtracting step size along normal
      }while (no values in z-bucket list)
}
```

2.2 DATA FILTERING

One of the novel features of 3DIVE is its ability to perform data filtering using 3D image processing operations from within the application on a per-region basis. In this version of 3DIVE, a temporary data buffer will be used for each object to store a local copy of the data for image processing operations. By storing both a global copy and a local copy of the data, a user can switch between the original data set and the newly processed data set without delay. In addition, the user will experience no loss in rendering speed when moving a processed regions throughout the data set. The actual image processing operations are implemented using SGI's ImageVision API [10], which provides fast responses for most 3D image processing operations. This approach has two disadvantages: (1) if each regions requires its own copy of the data set, memory can quickly run out and (2) our image processing relies on ImageVision which is specic to SGI machines. It should, however, be mentioned that 3DIVE was implemented as an immersive environment (currently only for CAVE-like systems). These systems inherently require a large amount of main memory and most only run on high-end SGI machines.

2.3 COLLISION DETECTION

Volumetric collision detection is very important for a region-based system in which collisions between volumetric objects often need to be avoided or specially treated. It should be mentioned that detecting collisions between the geometries of the volumetric primitives is not sufficient since the actual volumetric objects are defined by the data values within the primitives. There exist many possible applications, the most prominent being virtual surgery, where the touching and penetration of tissues, bones, and surgical tools all require specific actions.

The slicing algorithm described earlier for volumetric primitive rendering was the basis for a volumetric collision detection framework. This framework provides a mechanism for collision detection between volumetric regions. The collision detection test, implemented as a two-step process, is performed slice-by-slice using various blending functions and framebuffer tests. The first step, the global test, is performed once per slice. Using the logical OR blending function and the glMinMaxEXT() function, this step determines whether or not a collision between at least two regions has occurred. If a collision was detected, the identity test further investigates until the exact region identities are discovered. For a more detailed discussion of the algorithms, the reader is referred to [11].

The volumetric collision detection procedure described here is not directly integrated within the 3DIVE interface. This is because that the collision detection algorithm has a running time proportional to the screen resolution, and therefore is particularly and unnecessarily expensive for projection-based virtual reality equipment. It would be much more suitable for head-mounted displays with lower display resolutions.

3. Virtual Environment

Virtual reality offers several advantages over the traditional 2D display/interaction paradigm. The use of magnetic trackers allows the user to navigate and view the environment stereoscopically in real-time. Using the same trackers, the user can also interact in 3D through varied input devices. This combination of real-time viewing and more natural interaction is an excellent addition to any volume visualization environment.

3DIVE was implemented for the CAVE Automatic Virtual Environment [12, 13][1]. A standard CAVE consists of multiple walls, a floor, and possibly a ceiling. The graphics are generated using a high-end multi-pipe SGI machine and displayed via rear-projection for each wall. The user wears light-weight active-matrix shuttering glasses to achieve the stereoscopic effect and uses the wand, basically a 3-button 3D mouse/pointer, for input (selection and manipulation).

3.1 INTERACTION

As described by Gabbard and Hix [14] in their taxonomy of usability characteristics in virtual environments, interaction is key to a successful virtual environment. 3DIVE uses the standard wand along with a software menuing system to achieve real-time, well-organized interaction. 3DIVE's interaction paradigms can be broken down into two categories: (1) wand actions and (2) menu actions. We de_ne wand actions to be volumetric manipulations that include all selection and movement of regions. Menu actions refer to those actions achieved only through the 2D menuing interface incorporated within 3DIVE. Locomotion is not necessary and hence not allowed.

3.1.1 Wand Actions

Wand actions incorporate two distinct paradigms: (1) region selection and (2) region movement. 3DIVE implements region selection through the classical virtual hand [15]. By rendering a virtual hand at the tracked wand's position, 3DIVE can o_er a natural grabbing mechanism. In order to select a region, the user simply reaches out to the appropriate region and once intersected by the virtual hand, clicks the right button. Region-border coloring provides visual cues about the successful intersection. The biggest disadvantage to the virtual hand technique is its limited range. However, this is easily overcome in 3DIVE since all regions are always within reach due to the dataset's central location within the CAVE. De-selection is accomplished by clicking the same button when the virtual hand is not intersecting any region. A more detailed discussion concerning virtual hands and their extensions and usability in virtual environments can be found in [16].

Once selected, a region can be moved (that is, rotated or translated) simply by moving the wand while holding the middle button. All movements are performed within the region's coordinate system. If no region is selected, the middle button will move all regions with respect to the scene coordinate system.

3.1.2 Menu Actions

To facilitate global state changes or parameter adjustments, 3DIVE relies on a simple 2D menuing facility. Simple Cave Menuing System (SCMS) allows 3DIVE users to activate a familiar 2D widget set consisting of buttons, sliders and labels. The menuing system resides on a virtual pallet that can be

[1] Of course, the system also runs on ImmersaDesks and InfinityWalls without recompiling, as they rely on the same technology and software libraries.

opened or closed with the click of a button and positioned arbitrarily in the room. From the menu, users can perform either per-region or scene operations.

Per-region operations include:

- *Scaling.* This allows the user to perform scaling in the x-, y-, and z- dimension. Sliders are provided for each dimension to ensure accuracy.
- *Transfer Function Editing*. For intensity volumes, each of the four color tables (red, green, blue and alpha) may be specified through a graphical function editor. In addition, a minimum and maximum threshold capability has been implemented.
- *Data Filtering.* Nine operators have been implemented to provide per-region image processing. Up to three operators may be chained together for maximum efficiency. The user is also allowed to switch between the original volume and the filtered.
- *Capture Mode*. With capture mode disabled, the rendering for a given region is defined by the intersecting volume data and the region's geometry. The capture mode allows the user to pull-out a portion of the dataset. This facilitates the examination of the same region using different filtering/transfer function techniques.

Scene operations include:

- *Resolution Control.* This slider allows the user to control the number of approximating slices. A maximum resolution button is provided for simplicity.
- *Blending Function Selection*. Using these buttons, the user may select either mathematical addition or maximum projection for the blending phase of the volume rendering.
- *Slicing Tool Control.* The slicing tool provided by 3DIVE has four settings. It allows constrained viewing along any of the three axis along with the ability to attach a plane to the end of the virtual hand and render only the intersecting volume data. Any combination of these four modes may be enabled.
- *Snapshot.* This simply writes the current framebu_er image to disk as an RGB _le. In addition to the operations supported above, there is an extensive database facility which gives the user the ability tosave/load transfer functions, region data, or entire scene data. It can also serve as a temporary bu_er which implements a cut-and-paste mechanism between regions.

4. Collaboration

It is nearly impossible to discuss collaboration in a virtual world without _rst considering avatars. An avatar is simply a virtual representation of a physical person (currently logged into the environment). Most avatars are rendered using geometric structures that resemble a real human (i.e. head, hands, body, etc.). The most prominent motivations for avatar-based societies include: (1) participants are free to move and explore whatever they wish and (2) participants know what others are exploring and where they are located. The biggest disadvantage of this approach is the complexity of synchronizing views. What if a participant needs to see exactly what another person is seeing (i.e. view sharing)?

Simple and accurate view sharing was our primary concern for 3DIVE's collaborative functionality. We wanted an interface that offered just enough collaboration for the task of volume visualization and manipulation. Since locomotion is not allowed, avatars would not have been beneficial. 3DIVE, therefore, employs a homogeneous view for all participants. A 3DIVE session consists of several people logged into the same 3DIVE server. One of those participants is selected as the master (usually the first to join). All other participants are labeled slaves. The master has total control of the environment through the normal 3DIVE interface (i.e. wand and menuing system). The slaves see exactly what the master sees. Through a simple click of the collaborative session menu, the master can turn over control to a slave. That slave then becomes the new master and control continues under his management. To allow local slave exploration, a slave may temporarily disconnect from the master's control. During this time, the slave is still connected to the session, but his actions are local to his scene. At a later time, this slave may return control to the managed session. The server then synchronizes his view with the master. Audio communication is accomplished via a separate audio link. This view-sharing paradigm is ideal for many applications such as virtual surgery or anatomical training where one or possibly many participants are present only to learn/observe or offer verbal guidance to another.

3DIVE's collaborative module uses a client-server methodology to facilitate communication. The server process has two responsibilities: (1) handle requests to join/leave a session and (2) store state data and filter it to the necessary participants. If connected to a session, the client processes are always in one of two states: (1) master state sends interface data and (2) the slave state receives interface data. For implementation, the CAVERNSoft library is utilized [17]. In short, CAVERNSoft is a hybrid approach to distributed shared memory using persistent database and high-performance networking technology. In CAVERNSoft, each client creates its own Information Request Broker (IRB) that maintains a local copy of the necessary data. Using keys, several IRBs may be linked together to form an interconnection of virtual environments. That data is then shared by CAVERNSoft using one of its many networking interfaces. The distributed shared memory paradigm (offered by CAVERN-Soft) works well for 3DIVE since all slaves basically just read the master's current state and duplicate it within its own process.

Figure 3 outlines the various components of the networked environment. The ColLib, VeLib, and VisLib represent the three core modules of 3DIVE which are in constant communication. The shared memory arena between the collaborative module (ColLib) and the CAVERNSoft IRB acts as the repository for interface/state data that eventually needs to be transmitted between participants. Once the data is given to the IRB, CAVERNSoft handles all network communication.

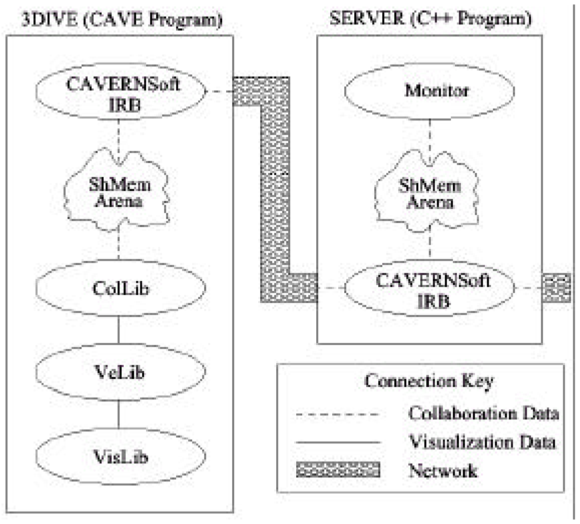

Figure 3: Collaboration Overview

5. Implementation

The previous three sections have described the algorithms used to implement the various components of 3DIVE. Figure 4 diagrams the relationships between these components and existing software libraries. VisLib is the largest of the modules. It is responsible for the region definition and manipulation. OpenGL [18] is used for the volume rendering while several operators of ImageVision [10] implement a subset of the data filtering possibilities. The VeLib is the virtual interface module. All display, wand and menu related code is implemented using the CAVELib [19] and SCMS. ColLib handles the collaborative functionality. The actual communication is accomplished using CAVERNSoft [17].

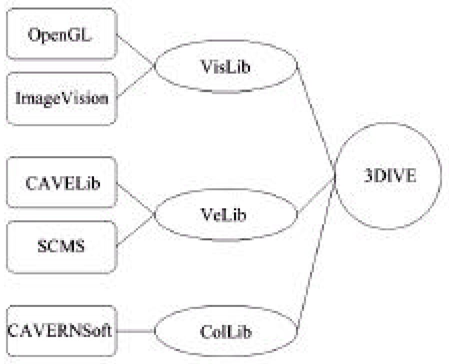

Figure 4: Software Libraries

3DIVE is currently supported on SGI machines (with 3D texture mapping hardware). 3DIVE has been used on both a CAVE and ImmersaDesk system. Using an ONYX2 with Reality Engine graphics and 64MB texture memory, we achieve an average of 6 to 12 FPS. Several factors contribute to the rendering speed; however, the most noticeable include (1) the current number of regions in the scene and (2) the necessary raster _ll requirements. Both are proportional to the frame rate. Figure 5 shows a 256 3 portion of the National Library of Medicine's Visible Human (male) dataset using three regions. The region outlined in blue is active and the scene control menu is shown on the left side. The region outside the red volume space was achieved using the capture feature described in Section 3.1.2. The FPS is shown in the upper left corner as 7.15.

3DIVE is being (or has been) used in projects ranging from traditional medical imaging to interstellar gas cloud exploration. However, the most prominent project involves the Department of Nephrology at the Indiana University School of Medicine. These researchers are using 3DIVE to perform interactive 3D confocal microscopy imaging and visualization on epithelial kidney cells. Confocal images are very difficult to understand due to their large sizes, noisy and inaccurate signals, and highly complex structures. 3D has allowed the research staff to investigate their data interactively in true 3D. Figures 6 and 7 show two datasets collected and analyzed by the IU Department of Nephrology. The 512 x 512 x 64 dataset shown in Figure 6 was collected using 3-channel coloring from the confocal microscope. This scene also consists of three regions. They are rendered in slicing mode. The cyan outlined polygons represent the intersecting region space. Figure 7 provides a rendering of a 256 x 256 x 64 data-filtered golgi complex. Blurring and sobel edge detection algorithms were applied prior to range filtering and color table manipulation. The imaging menu environment is active on the left. The cursor's position on the menu shows where up to three operations may be chained together.

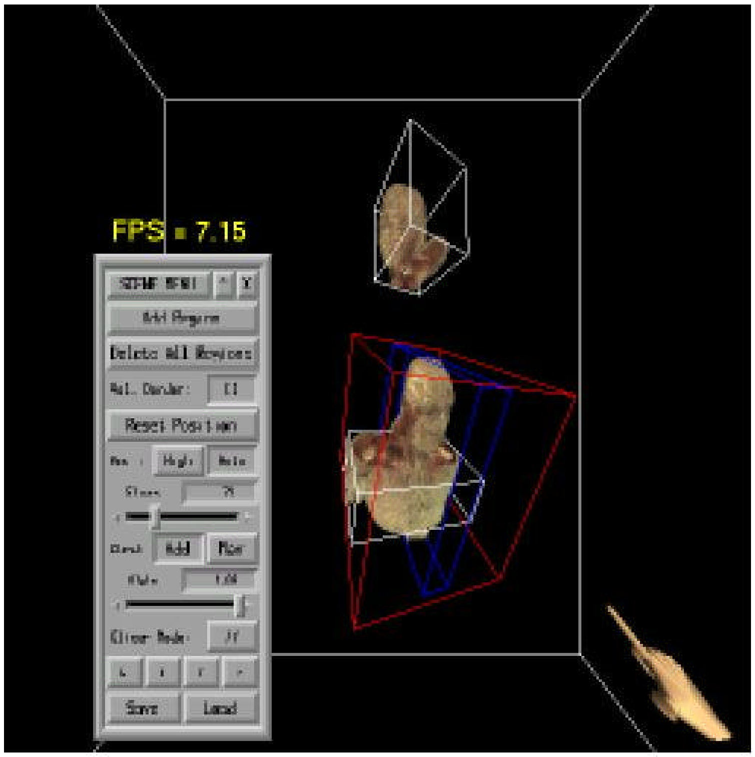

Figure 5: Visible Human Dataset

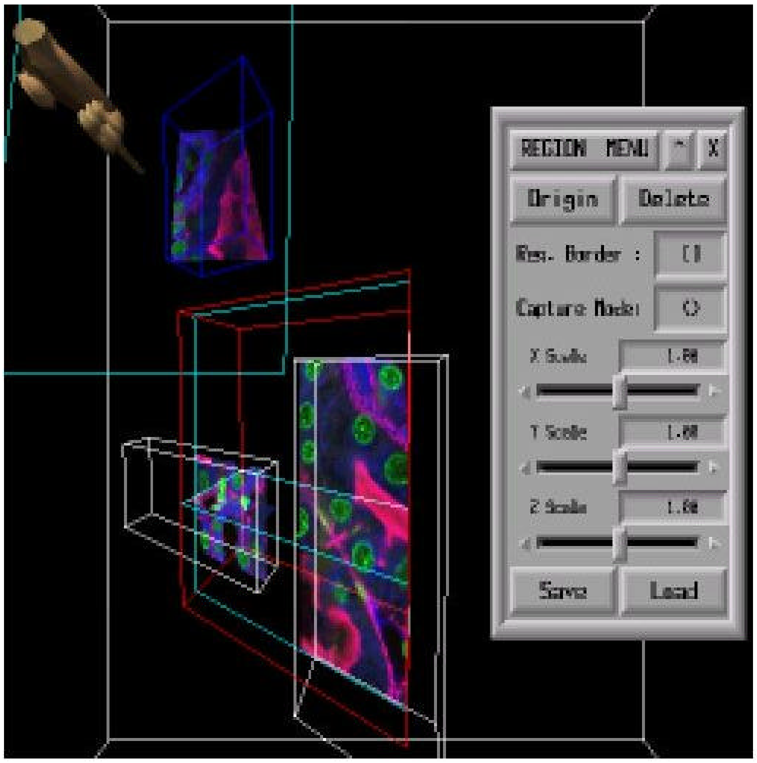

Figure 6: 3-Channel Colored Microtubles

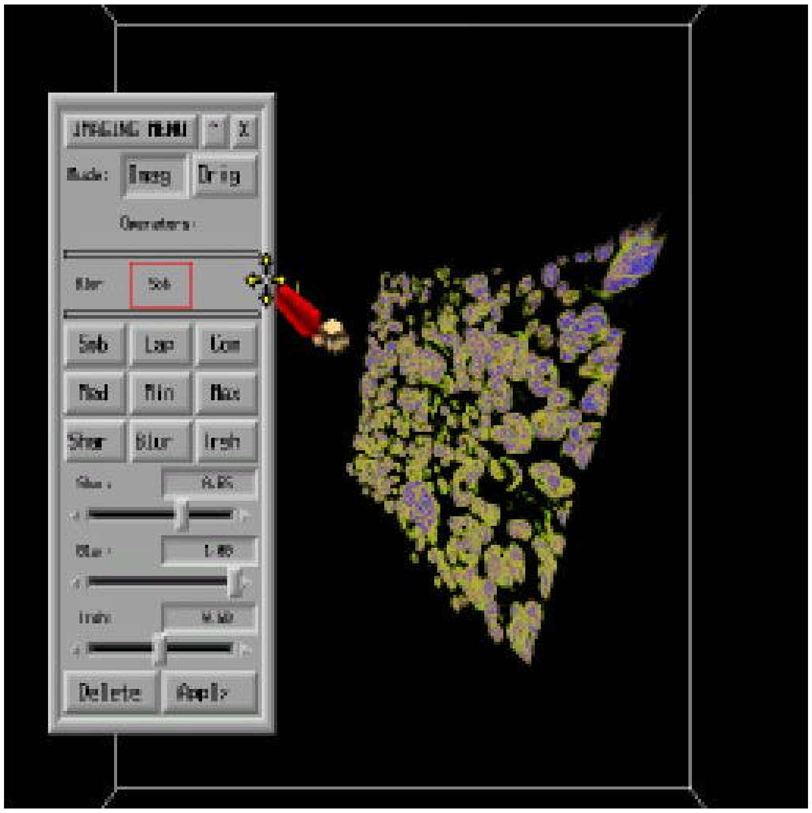

Figure 7: Filtered Golgi Complex

6. Conclusion

This paper described a novel system for the visualization of volume data inside a virtual environment. By combining the concepts of region-based manipulation, data filtering and collaboration in an immersive environment, researchers, scientists and doctors are able to explore their data in a more intuitive way which leads to higher productivity. Using 3DIVE data filtering is no longer a preprocessing stage. Instead, it can be performed in real-time from within the application. The collaborative component then allows researchers to explore and examine the results from geographically distributed locations.

3DIVE is a solid foundation for future work in immersive volume visualization systems. We are constantly investigating and adjusting 3DIVE's interface where necessary. Specifically, the menuing API is being expanded to include features such as disk access from within the application. This would allow users to load data sets without leaving a session. Another future work involves adding a tool for the arbitrary creation of volumetric regions and compensating the collision detection algorithm for high resolution displays. New concepts for tele-collaboration for volume visualization are also being explored. For more information, the reader is referred to the 3DIVE web site, www.avl.iu.edu/projects/3DIVE.

REFERENCES

[1] R. Avila, T. He, L. Hong, A. Kaufman, H. P_ster, C. Silva, L. Sobierajski and S. Wang. VolVis: A Diversi_ed Volume Visualization System. IEEE Visualization Proceedings, October, 1994, pp. 31-38.

[2] Hanspeter P_ster, Jan Hardenbergh, Jim Knittel, Hugh Lauer, and Larry Seiler. The volumepro real-time ray-casting system. Computer Graphics, SIGGRAPH'99, August 1999.

[3] Rachael Brady, John Pixton, George Baxter, Patrick Moran, Clinton Potter, Bridget Carragher and Andrew Belmonts. Crumbs: A Virtual Environment Tracking Tool for Biological Imaging. Proceedings of the IEEE Symposium on Frontiers in Biomedical Visualization, Atlanta, GA, October 30, 1995, pp. 18-25, 1995.

[4] CAVE5D User's Guide. www.ccpo.odu.edu/ cave5d/cave5dGuide.html, April 1998.

[5] W. Hibbard and D. Santek. The VIS-5D System for Easy Interactive Visualization. Proc. IEEE Visualization '90, 129-134, 1990.

[6] George Eckel. OpenGL Volumizer Programmer's Guide. Silicon Graphics, Inc., 1998.

[7] T. Cullip and U. Neumann. Accelerating Volume Reconstruction with 3D Texture Mapping Hardware. University of North Carolin Computerized Medical Imaging and Graphics Technical Reports, TR93-0027, 1993.

[8] O. Wilson, A. Van Gelder and J. Wilhelms. Direct Volume Rendering via 3D Textures. Technical Report UCSC-CRL-94-19, University of California, Santa Cruz, 1994.

[9] J. Foley, A.Van Dam, S. Feiner and J. Hughes. Computer Graphics: Principles and Practice. Addision Wesley, 1996.

[10] George Eckel, Jackie Neider, and Eleanor Bassle. ImageVision Library Programming Guide. Silicon Graphics, Inc., 1996.

[11] Michael Boyles and Shiaofen Fang. Slicing-Based Volumetric Collision Detection. To Appear in: ACM Journal of Graphics Tools.

[12] Carolina Cruz-Neira. The case: Audio-visual experience automatic virtual environment. Communications of the ACM, 1992. 35(6): pp. 65-72.

[13] Carolina Cruz-Neira, Dan Sandin and Tom Defanti. Surround-Screen Projection-Based Virtual Reality: the design and implementation of the CAVE. ACM SIGGRAPH 93, August, 1993.

[14] Joseph L. Gabbard and Deborah Hix. A Taxonomy of Usability Characteristics in Virtual Environments. Deliverable to O_ce of Naval Research from Department of Computer Science, Virginia Polytechnic Institute and State University, 1997.

[15] I. Poupyrev, M. Billinghurst, S. Weghorst and T. Ichikawa. Go-Go Interaction Technique: Non-Linear Mapping for Direct Manipulation in VR. Proceedings of the UIST'96, pp. 79-80, ACM, 1996.

[16] I. Poupyrev, S. Weghorst, M. Billinghurst and T. Ichikawa. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques. Computer Graphics Forum: EUROGRAPH-ICS'98 issue, 17(3), 41-52, 1998. 9.

[17] J. Leigh, A. Johnson and T. Defanti. CAVERN: A Distributed Architecture for Supporting Scalable Persistence and Interoperability in Collaborative Virtual Environments. Virtual Reality: Research, Development and Applications, Vol 2.2, pp. 217-237, 1997.

[18] Mason Woo, Jackie Neider and Tom Davis. OpenGL Programmer Guide Second Edition. Addision Wesley Developer's Press, 1997. [19] D. Pape, C. Cruz-Neira and M. Czernuszenko. CAVE User's Guide. Electronic Visualization Laboratory, University of Illinois at Chicago, 1997.

BIOGRAPHIES

Michael Boyles is lead analyst/programmer at the Advanced Visualization Laboratory of Indiana University. His general interests lie in the area of 3D computer graphics and visualization, but he has a particular interest in collaborative visualization, volume visualization, virtual reality and haptics.

Contact information:

Advanced Visualization Laboratory
Indiana University
799 W Michigan Street
Indianapolis, IN 46202
Phone: 317-278-3647
Fax: 317-274-0776
Email link: mailto: mjboyles@iupui.edu
Web: http://www.avl.iu.edu/~mjboyles/

Shiaofen Fang is Associate Professor in the Department of Computer and Information Science at Indiana University-Purdue University at Indianapolis, Indiana. His research interests are visualization, computer graphics, and geometric modeling.

Contact information:

723 W. Michigan St., SL280
Indianapolis, IN 46202
Tel: (317) 274-9731
Fax: (317) 274-9742
Email link: mailto:fang@cs.iupui.edu