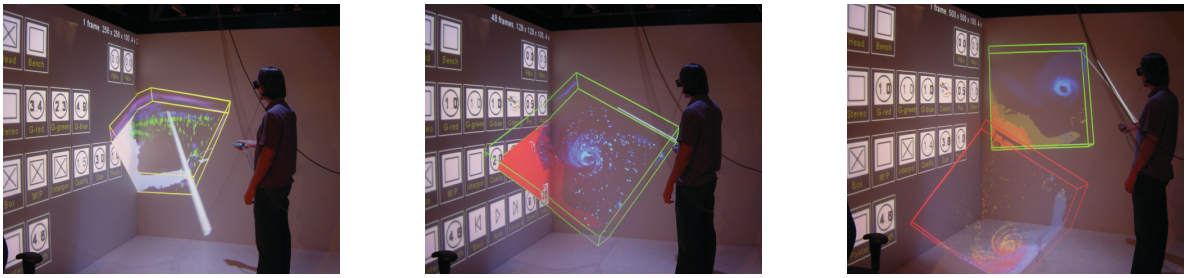
User Friendly Volume Data Set Exploration in the Cave

Jürgen P. Schulze Andrew S. Forsberg

Department of Computer Science, Brown University, Providence, RI
{schulze|asf}@cs.brown.edu



(a) (b) (c)

Figure 1: Our software in the Cave with three methods of data fusion: (a) multiple variables in one data set (ICE, RAIN, SNOW); (b) animation (CLOUD); (c) multiple data sets side-by-side (CLOUD01 and CLOUD48).

ABSTRACT

We are presenting an interactive visualization system to explore volume data sets like the contest's hurricane data in a CAVE-like immersive virtual environment. We use an an immersive environment because we hypothesize that immersive viewing (i.e., via a head-tracked stereo tool) will give a better sense of 3D structures. Furthermore, when viewing an *animation* of complex 3D data, we expect that a better appreciation of each frame and its relation to adjacent frames can be achieved with head-tracked stereo viewing. Data fusion can be done in different ways: up to four independent variables can be shown in a single image, time dependent data can be shown as real-time animations, and multiple data sets can be displayed concurrently. All visualizations can be viewed and interacted with at interactive frame rates. The focus of our work is on the user interface and real-time visualization.

1 INTRODUCTION

We first summarize our contributions to the contest's tasks: interactivity, exploration, and visualization of multiple characteristics. Then we explain the user interface elements and the data processing in more detail, present performance numbers, and explain soft- and hardware. Accompanying images and movies, along with more details, can be found at: http://www.cs.brown.edu/people/schulze/vis04-contest/.

2 TASKS ACCOMPLISHED

In this section, the methods our system offers to accomplish the three major tasks of the contest are presented.

2.1 Interactivity

Our visualization system runs at interactive frame rates of about 10+ fps most of the time, except when data sets are loaded from disk, or the floating point mapping changes. All other interactions like navigation, changes of transfer functions, resizing, region of interest mode, and clipping planes happen instantaneously. The data sets are displayed using texture based direct volume rendering. Modifications of transfer functions require graphics hardware support for fragment shaders.

The frame rate that our software runs at depends mainly on how much screen space the data sets occupy, i.e., how many fragments are being rasterized (pixel fill rate limited). The data set reconstruction quality can be specified by the user with a widget from within the Cave, which directly influences the frame rate.

2.2 Exploration

Our system offers the following software tools for interactive exploration. This section is an abbreviated version of the corresponding one on the web page—please refer to the latter for more details and screen shots.

Transfer functions: In data sets with one scalar value per voxel, the color mapping and the mapping from scalar values to opacity can be changed with two dials and a button. Colors are selected by cycling through seven pre-defined color maps. Opacity is set with a ramp function of which the slope and position can be specified by the user. In data sets with two to four scalar variables for each voxel, the intensity of the first three variables (which are shown in red, green, and blue, respectively) can interactively be changed. The fourth data channel can be mapped to an arbitrary color with dials for its red, green, and blue components.

Concurrent display of multiple data sets: Multiple data sets can be put side-by-side to help discover differences. The limit on the number of data sets displayed is the graphics card's texture memory.

Region of interest: This mode limits the rendering region to a sub-volume, and a wire frame indicates the full volume. The ROI can be moved and resized by moving and rotating the wand.

Clipping plane: Instead of visualizing the full 3D data set, only an opaque, user-specified plane through it is rendered. The plane can be freely rotated and positioned with the wand.

Markers: Cone-shaped markers can be placed in the data set to serve as references, for instance to help find features after changes of visualization parameters. The markers can be moved and oriented arbitrarily by the user.

Reconstruction mode: From within the Cave, the user can change between alpha blending and MIP. MIP gives a fast overview about regions of different intensity in the data set. Alpha blending gives a more realistic representation of a translucent object.

Data value zoom: The mapping of floating point data values to integers, which is required by the graphics hardware, can be changed from within the Cave. This acts as a zoom function in the data domain.

Pointer: By default, the user interacts with the system with a virtual laser pointer long enough to reach through the Cave. An optional type of laser pointer with variable length and a red tip is available. This is useful to point to features of a data set when more than one user are in the Cave.

2.3 Multiple Characteristics

Multiple variables of the hurricane data sets can be displayed concurrently in the Cave. The three major methods that our system supports to accomplish this are:

Multiple variables in one data set: By storing different variables in different color channels, up to three variables can be displayed in a single image (see Figure 1a). This visualization technique works especially well if most voxels contain only one modality.

All time steps of a variable shown as an animation: By concatenating multiple time steps, the temporal development can be visualized (see Figure 1b). Our software provides VCR-like controls in the Cave to start and stop the animation, as well as single step forward and backward, playback speed, and direct time step selection.

Multiple data sets side-by-side: By rendering multiple data sets in the same environment, the user can directly compare differences in the data sets (see Figure 1c). As many data sets as fit into texture memory can be displayed concurrently.

Any combination of the above methods is possible: Animations of different variables can run in parallel, multiple variables can occur in an animated data set, multiple data sets with multiple variables can be displayed, and multiple animations of data sets with multiple variables can run in parallel.

3 User Interface

Our user interface consists of three main elements: widgets on the left wall of the Cave, pop-up menus for context specific settings, and gestures as shortcuts or for special functions.

3.1 Wall Widgets

The most frequently used functions, as well as functions that are global to the system, are located on the left wall of the Cave. We placed them in the plane of the projection screen, thereby allowing both head-tracked and non-head-tracked viewers to clearly view the icons at all times. There are three basic types of widgets: action buttons, check boxes, and dials. The first two are triggered with a click of the left wand button, dials are manipulated by holding the left button down and twisting the hand.

3.2 Pop-Up Menu

The system offers two pop-up menus. They are triggered by pressing the right wand button. The menus show up at a predefined height and distance, and in the direction of the user's head. When the user clicks on the Cave background, the file browser pops up: it shows all available data sets, each with a preview image. When the user pushes the right button when pointing at a data set, a pop-up menu with widgets for region of interest, clipping plane, and marker mode appears. Both pop-up menus disappear once the user has made a selection.

3.3 Gestures

Some of the user interaction is done in implicit ways without the use of widgets. Navigation is done by pointing on a data set, clicking the left button, and dragging the data set around—the middle button always provides this functionality as a shortcut. The trackball on the wand rotates the selected data set around its center. By clicking the left wand button when the wand is near the eyes, a snapshot of the front wall is stored on disk as a file—this is useful for the users to take home effective views on data sets. We implemented a shortcut method for dials which we call "widget snarfing": the system memorizes the most recently used dial and manipulates it when the user clicks on the Cave background and twists the hand.

4 Data Processing

We converted all hurricane data sets to our own binary volume format (Virvo XVF), which allows storing some of the data set specific information like the physical minimum and maximum values in a file with the volume data. Animations can be stored in a single file as well, for easier file handling. The data values are stored as floating point values. Any mapping to integer values happens when the data are loaded to texture memory. For interactive frame rates the entire data set must fit into texture memory.

5 Rendering Performance

To give an idea of our system's rendering performance, we ran two tests. In the first test we used the CLOUDf48 dataset at $500 \times 500 \times 100$ voxels and 8 bits/voxel, rendering full screen on the front wall with 1024×768 pixels. We achieved 7.5 fps when the volume was reconstructed from 714 textured polygons.

Using a multi-channel data set with 24 bits/voxel—8 bits for each of three variables—we get 6.5 fps with a $256 \times 256 \times 100$ voxels data set, reconstructed with 376 textured polygons.

6 Implementation and Hardware

Our software system runs on a four node rendering cluster of Dell PCs with dual 3.0 GHz Pentium CPUs and 1 GB RAM, and Nvidia Quadro 3000g graphics boards. The four images in the Cave have 1024x768 pixels each and are rendered in frame sequential stereo, each wall is driven by one PC.

Additionally, we can run the system on a larger cluster of rendering nodes (e.g., 48-nodes) to boost the performance. Unfortunately, our 48-node cluster has outdated graphics cards that do not support pixel shaders so while it provides higher frame rates, the complete functionality of our system is not available yet. We hope to upgrade this system in the near future.

Our software is based primarily on C++ based in-house libraries to drive the Cave. We are using the OpenSceneGraph API as an interface to the graphics elements. With this API we integrated a volume rendering node, which is based on the Virvo library [1].

References

[1] J.P. Schulze, U. Wössner, S.P. Walz, and U. Lang. *Volume Rendering in a Virtual Environment*. Proceedings of the Fifth Immersive Projection Technology Workshop (IPTW'01) and Eurographics Virtual Environments (EGVE'01), Springer Verlag, pp. 187–198, 2001.