MedVis: A Real-Time Immersive Visualization Environment for the Exploration of Medical Volumetric Data

Rui Shen, Pierre Boulanger
Department of Computing Science
University of Alberta, Edmonton, AB, Canada
{rshen,pierreb}@cs.ualberta.ca

Michelle Noga
Department of Radiology & Diagnostic Imaging
University of Alberta, Edmonton, AB, Canada
mnoga@ualberta.ca

Abstract

This paper describes the Medical Visualizer, a real-time visualization system for analyzing medical volumetric data in various virtual environments, such as autostereoscopic displays, dual-projector screens and immersive environments such as the CAVE. Direct volume rendering is used for visualizing the details of medical volumetric data sets without intermediate geometric representations. By interactively manipulating the color and transparency functions through the friendly user interface, radiologists can either inspect the data set as a whole or focus on a specific region. In our system, 3D texture hardware is employed to accelerate the rendering process. The system is designed to be platform independent, as all virtual reality functions are separated from kernel functions. Due to its modular design, our system can be easily extended to other virtual environments, and new functions can be incorporated rapidly.

1. Introduction

Medical imaging modalities such as *Computed Tomography* (CT) and *Magnetic Resonance Imaging* (MRI) produce high-quality 3D data that radiologists use to diagnose various health problems of patients. However, due to technological limitations, it is still commonplace for radiologists to observe volumetric data as a set of slices printed on films viewed in front of a white diffusing light source. Even though this practice is well accepted in the medical community, it only utilizes a fraction of the data provided by the imaging systems. One important rational for using such system is that radiologists want to observe the data set without prior interpretations by a computer that may hide important structures that only a human can interpret as important. Several systems, from non-commercial software (*e.g.,* ParaView) to commercial products (*e.g.,* AVS/Express), have been developed for this purpose. However, the visualization and manipulation stay in the 2D space, *i.e.,* users can only see the volumetric data as a 2D image on a screen and the feedback of an operation on the data is still a 2D image. Although volume rendering is supported in current systems, radiologists still cannot directly examine 3D volumes in stereo. This problem can be resolved by using virtual reality (VR) techniques. The effectiveness of VR technology in medicine has been proven in many applications [11]. Hence, to assist existing diagnostic procedure by creating a new insight through 3D visual representation, we have developed the *Medical Visualizer* (MedVis), a medical visualization system that provides real-time high-quality volume rendering and interaction with 3D medical data in virtual environments (VEs).

Our main contributions in this paper are:

- Developing a cross-platform medical data visualization system capable of dealing with various display modalities. Radiologists can use this system to interactively explore medical volumetric data in stereo in various non-immersive or immersive VEs.
- Incorporating new GPU-based acceleration techniques of volume rendering that we developed earlier into the system. Remarkable speedups compared to traditional techniques are observed from real experiments.
- Developing user-friendly interface both for a standard PC environment and for virtual environments. Users can carry out various interactions with the data set in realtime.
- Object-oriented programming paradigm and modular design concept are employed throughout the development of MedVis, which maximizes the extensibility and portability of the system.

2. Related Work

Various medical data visualization systems have been proposed for virtual environments. Zhang *et al.* [16] de-

vise a system for visualizing diffusion tensor MRI data sets of brains in a CAVE environment. The brain is rendered as iso-surfaces, but due to a large amount of decimation required to obtain an interactive rendering speed, many details are lost in the final images. Lapeer *et al.* [9] introduce the ARView, a generic software framework for stereoscopic augmented reality microsurgery. However, it only supports non-immersive single-screen VR systems. Its transfer function editor only provides the ability to adjust the mapping of a scalar value to each color channel separately but without a reference color palette, which makes it very difficult to generate the desired mapping.

Kratz *et al.* [8] integrate perspective direct volume rendering into a virtual reality system developed in their lab. However, no VR user interface is provided for manipulating the transfer functions on the fly. Although they argue that a transfer function editing widget could be easily added, this inability to interactively manipulate the transfer functions impairs the effectiveness of analyzing the data as it relies on prior information for display. In addition, this system only supports non-immersive VEs, and the extensibility of the system to immersive VEs such as CAVE is not addressed. In contrast, Kniss *et al.* [7] apply hardware-accelerated volume rendering and VR techniques to visualizing multi-field medical data sets in immersive VEs. However, the design of the transfer function manipulating interface uses the so-called data-centric methods as mentioned in [10], which tend to be fairly counter-intuitive.

3. System Design and Implementation

MedVis has integrated the advantages of many of the systems found in the literature and incorporated improved functions, as MedVis supports enhanced GPU-based volume rendering, allowing for various real-time interaction modalities, and the ability of extensibility and portability. Currently, MedVis has a desktop version that can easily fit on a radiologist's workplace, and an immersive CAVE version, in which a radiologist can explore the volumetric data in a more natural way.

3.1. GPU-Based Volume Rendering

Volume rendering has been proven useful in many medical applications, such as to help in diagnosis [6]. Unlike surface rendering, which depends on the assumption that the important structures in medical images can be segmented using simple algorithms, direct volume rendering bypasses the intermediate geometric representation and directly renders the volumetric data set based on its scalar values. This allows a radiologist to visualize the fine details of medical data as he/she is the one who by law must make the final interpretations. Considering the advantages for radiology of direct volume rendering over standard surface rendering, this is why direct volume rendering is employed in MedVis as the main rendering algorithm.

To guarantee a high rendering speed that allows for real-time interactions with little degradation to the image quality, the graphics processing unit (GPU) is employed to accelerate the rendering process. This rendering algorithm was developed as part of our previous work [13]. The whole rendering process is illustrated in Figure 1. Most of the operations are performed in GPU. The proxy polygons that textures are mapped to are generated in the vertex shader of the GPU. Furthermore, using the idea of level-of-detail, the sample interval is adjusted according to the size of the volume in the world coordinate system and the distance from the viewpoint to the volume. The combination of the proxy polygon generation in the vertex shader and the adaptive sample interval doubles the rendering speed compared with previous methods, which use CPU to generate the proxy polygons with a fixed sample interval. For detailed algorithm description and performance comparison, please refer to our previous paper [13].

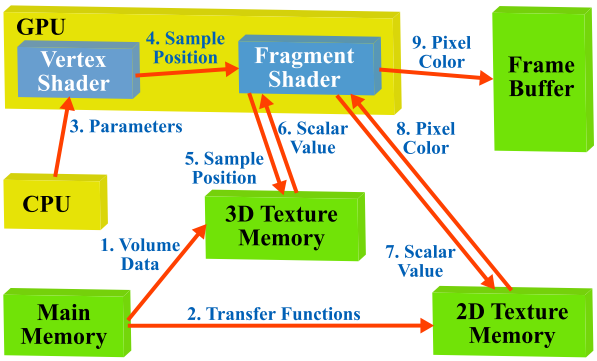


Figure 1. The dataflow between the CPU and the GPU.

In addition to the normal scalar color mapping and scalar opacity mapping, gradient opacity mapping is also employed to amplify the boundaries between different materials. After the gradient at each voxel is evaluated using the finite difference method, its magnitude is mapped to an opacity α'. The final opacity of a voxel is $\alpha' \times \alpha$, where α is the opacity obtained from the scalar opacity mapping.

3.2. Hardware Setup

3.2.1 Desktop Version

In the desktop configuration, MedVis runs on a consumer PC with an autostereoscopic display. We use an 18-inch DTI autostereoscopic display (on the left) to deliver glassless stereoscopic imagery and a normal display (on the

right) to display the control interface, as shown in Figure 2. When the stereo mode of the DTI display is enabled, with a special illumination plate placed behind the LCD screen, the strips of the left and right images are interlaced across the screen. Each image of the stereo pair is delivered to the corresponding eye. As for interaction, a 2D mouse is used to manipulate the volume or the control panel. Since this version only requires to add one autostereoscopic display to a normal PC setup, it easily fits in a radiologist's workplace.

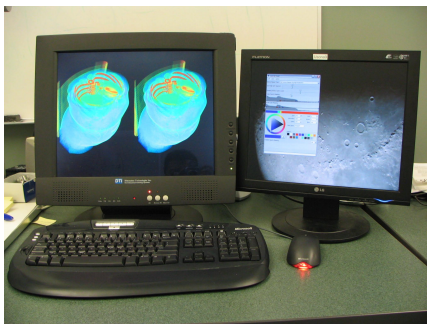


Figure 2. MedVis desktop hardware setup.

3.2.2 Immersive CAVE Version

Our CAVE is a $10' \times 10' \times 8'$ cube with rear-projected front, right and left screens, as shown in Figure 3(a). The MedVis CAVE version runs on a cluster, where each cluster node handles one of the three screens. Stereoscopic images are displayed on these screens and merged together to form a 3D virtual world. The current interaction configuration employs an InterSense IS-900, a 6-DOF (degree of freedom) inertial-acoustic motion tracking system. A head tracker (Figure 3(b)) is placed on the user's head or the shutter glasses so that the correct images can be displayed for the current view. A tracked wand (Figure 3(c)) with 4 buttons is used by the user as a controller for different interactions.

(a) CAVE.

(b) Head tracker.

(c) Wand.

Figure 3. MedVis CAVE hardware setup.

3.3. Interaction Modalities

3.3.1 Desktop Interaction

In the desktop interaction mode, the transfer functions can be adjusted through a 2D widget, and for all the transfer function changes, the rendered volume is updated correspondingly in realtime. Figure 4 shows the control panel.

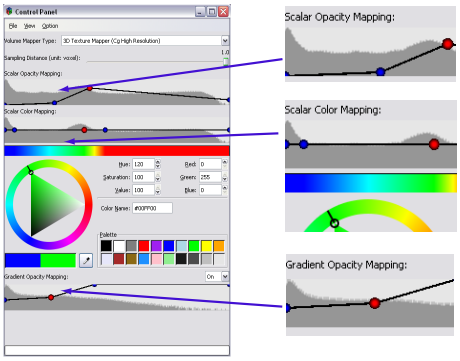
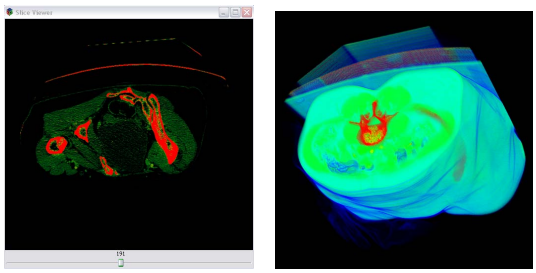


Figure 4. The desktop control panel.

The middle region of the control panel holds the transfer function editor. The top part of the widget controls the scalar opacity mapping. Its background shows the log-based histogram of the current scalar data. The foreground is the piecewise linear opacity transfer function represented as a combination of lines and spherical control points that determine the shape of the transfer function. The middle part controls the scalar color mapping, and the bottom part controls the gradient opacity mapping. The logarithmically scaled opacity transfer function proposed by Potts and Möller [10] is employed instead of traditional linear transfer functions used in previous volume rendering systems. Based on the observation that the change of the transfer function in the lowest five to ten percent of the range often results in the most visible differences in the images, the transfer function opacities are scaled logarithmically to provide a more intuitive and direct response. A linear function on the logarithmic scale is actually exponential on the linear scale, which produces more perceptible changes on the rendered images.

A mouse is used to simulate a joystick or trackball for transforming the displayed volume (*e.g.,* rotation, panning). The joystick style performs transformation based on the position of the mouse, while the trackball style performs transformation based on the magnitude of the mouse motion. The volume can be re-sliced along the current viewing direction and the slices are displayed in a separate window. Figure 5(a) shows the slice viewer with slices generated from the pelvic volume displayed in Figure 5(b). We define those interactions in the InTml framework [4] to guarantee

easy re-targeting to other VEs. InTml is an XML-based language developed at our lab for the description of complex VR application systems. It allows to create formal model of interactions that is hardware-independent and component-based, allowing for easy re-targeting and code re-use.



(a) (b)

Figure 5. Re-slicing along the view direction.

3.3.2 CAVE Interaction

In the CAVE interaction mode, the transfer function editor is a 3D extension of the 2D widget used in the desktop interaction mode. Figure 6 shows the 3D transfer function editor. The bottom part controls the scalar opacity mapping. The middle part is the 3D palette based on the HSV color model. The top part, together with the 3D palette, controls the scalar color mapping.

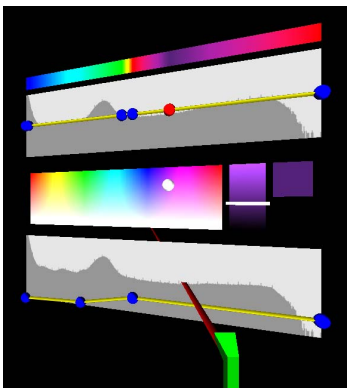


Figure 6. The CAVE control panel.

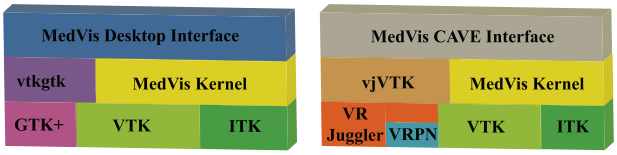
Although the appearance of the editor changes, the operations on the transfer functions remain similar to those in the desktop interaction mode. However, as the widget is placed in a 3D environment, the interaction is a little more complicated than the 2D case. A wand is used instead of a 2D mouse. A red line is drawn from the wand's position and along its direction. This red line serves as a 3D pointer as well as a controller for interactions. A thin bounding box of the transfer function editor is updated whenever the position of the editor changes. The point on the editor that the 3D pointer is aiming at is determined by the intersection between the bounding box and the infinite red line. The interaction style for transforming the displayed volume resembles the joystick style in the desktop version.

3.4. Software Architecture

MedVis is built using several open-source toolkits and follows the object-oriented programming paradigm. Figure 7 gives a high-level block diagram of the visualization pipeline. Medical data acquired from CT or MRI is usually stored in DICOM format. The DICOM image series are read in memory by the *reader* and organized in an internal structure, which is then passed to the *renderer*. The *interactor* handles user input and exchanges states with the renderer to adjust the rendered images.

Figure 7. MedVis's visualization pipeline.

As shown in Figure 8(a)(b), the whole system is designed in a modular fashion and consists of three major elements: the kernel module, the desktop interface module and the CAVE interface module. The kernel module deals with all VR-independent operations, such as volume rendering; the desktop interface module exports the kernel functions to the PC-based VR systems, like autostereoscopic displays; and the CAVE interface module extends the visualization pipeline into the immersive CAVE.



(a) (b)

Figure 8. MedVis software architecture.

Different toolkits are integrated together to support basic visualization and interaction, upon which more sophisticated techniques and algorithms are applied. All the toolkits are cross-platform and widely used in their respective areas. The *Insight Toolkit* (ITK) [15] serves as the interface between the DICOM format images and MedVis visualization classes. Due to its well-structured visualization pipeline and extensibility, the *Visualization Toolkit* (VTK) [12] is chosen as a basic visualization layer, upon which the MedVis rendering model is built. One major issue with VTK is that it

does not support stereo rendering. Therefore, we have extended VTK interactor classes to coordinate the rendering and other operations on the pair of stereo images. We also incorporated our rendering algorithm [13] into the VTK visualization pipeline.

The desktop interface of MedVis is built on the *Gimp Toolkit* (GTK+) [1] as VTK does not provide any graphical user interface (GUI). With *vtkgtk* [5], an interface for using VTK within a GTK+ widget in the X Window system, a VTK render window can be embedded into a GTK+ render window. We have extended vtkgtk to work in the Windows environment and added new functions. The CAVE interface of MedVis is built on *VR Juggler* [2], a virtual platform for the creation and execution of immersive applications. As MedVis visualization model is based on VTK pipeline, *vjVTK* [3] is employed to enable the use of VTK within VR Juggler. The *Virtual Reality Peripheral Network* (VRPN) [14] is used to connect the tracking devices with VR Juggler in order to provide a uniform hardware interface for MedVis CAVE version.

Due to this modular design, MedVis can be easily extended to support other VR setups. The implementation of MedVis is platform-independent. Although MedVis is currently running under Windows, its cross-platform nature allows it to run under Linux, IRIX, *etc.* with minimal modifications.

4. Results

4.1. Results for the Desktop Version

Figure 9 shows the rendering result of a CT-scanned abdomen (data size: 512x512x333) in MedVis desktop version. The left and right images shown in Figure 2 are merged together to form a 3D image. Running on a dual-core 2.0GHz computer with a 256MB-memory NVIDIA GeForce 7800 GTX graphics card, the system produces a 13Hz frame rate using the testing data in two 640x640 stereo viewports. VTK's GPU algorithm can only produce nearly the same frame rate by downsampling the data set to 10% of the original resolution, which results in a 256x256x128 subsampled volume. Lots of details are lost in the rendered images, as compared in Figure 10. The user can observe the rendered volume from any position, either inside or outside the volume by transforming it with a 2D mouse. The transfer functions and other control parameters can be adjusted by the user using a 2D mouse at runtime, and the changes to the volume are applied in realtime.

4.2. Results for the CAVE Version

Figure 11(a)(b) show the rendering results of an MRI-scanned heart (data size: 384x228x52) in MedVis CAVE

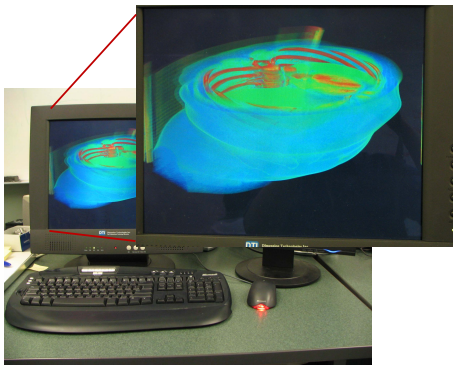
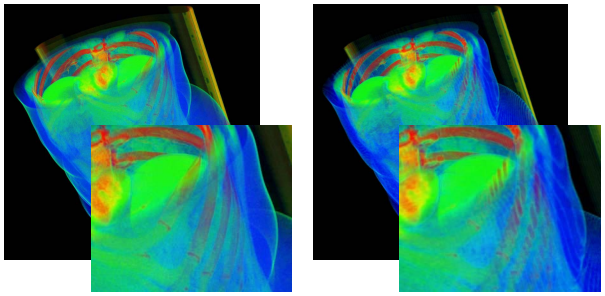


Figure 9. Volume-rendered 3D CT abdominal images on the DTI autostereoscopic display.



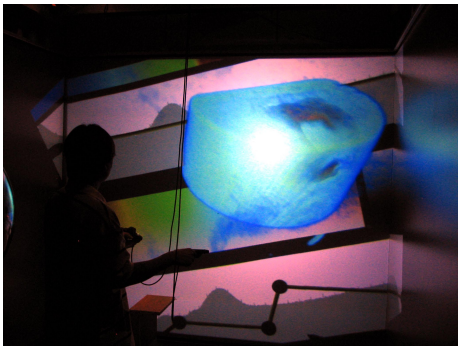
(a) Using MedVis. (b) Using VTK.

Figure 10. Comparison of the rendering results using MedVis and VTK.

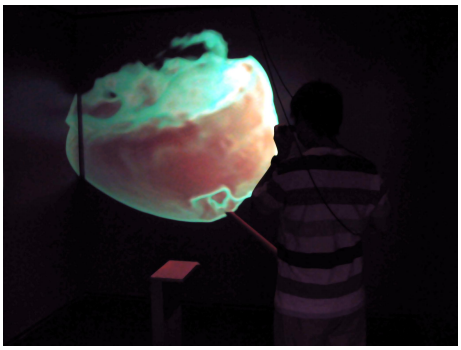
version. Each cluster node has a 2.4GHz CPU with a 256MB-memory NVIDIA Quadro FX 3000G graphics card. The system produces a 10Hz frame rate using the testing data on the three 1024x768 screens. VTK's rendering algorithm can only maintain half the rendering speed. Multiple users can stay in the CAVE simultaneously to analyze the data. The displayed volume can be transformed using the wand, or the user can walk around to observe it from different directions. The transfer function editor (shown in Figure 11(a)) can be turned on/off (shown/hidden) at any time, and the user can drag it using the wand and then place it in a convenient location. Like in the desktop version, the changes of the transfer functions are applied to the volume in realtime, *i.e.,* the user immediately gets the visual feedback of the effect of the current transfer functions. Figure 11(b) shows the visually segmented heart.

5. Conclusion

This paper presents the Medical Visualizer, a VR system for visualizing volumetric medical data in various VR sys-



(a)



(b)

Figure 11. Volume-rendered MRI cardiac images in the CAVE.

tems, ranging from non-immersive to immersive systems. To effectively analyze the fine details in the data sets, real-time high-quality stereo volume rendering algorithm is incorporated and friendly user interface for interactive manipulation of the color and transparency classifications is developed. MedVis desktop version can be easily incorporated into radiologists's daily workflow, while MedVis CAVE version allows radiologists to explore the volumetric data sets in an even more natural way. Some radiologists have already shown great interest in using our system to analyze volumetric medical data. In the future, we will perform a formal user study with our medical partners to demonstrate the advantages of our system relative to the traditional film-based approach. Based on the modular design concept, we will also extend MedVis to work with other VR modalities (*e.g.,* with haptic feedback) that may provide more help to radiologists in understanding the data sets.

References

[1] The Gimp Toolkit. http://www.gtk.org/.

[2] A. Bierbaum, C. Just, P. Hartling, K. Meinert, A. Baker, and C. Cruz-Neira. VR Juggler: a virtual platform for virtual reality application development. In *VR '01: Proceedings of the Virtual Reality 2001 Conference*, pages 89–97, 2001.

[3] K. Blom. vjVTK: a toolkit for interactive visualization in virtual reality. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 17–19, 2006.

[4] P. Figueroa, M. Green, and H. J. Hoover. InTml: a description language for VR applications. In *Web3D '02: Proceeding of the seventh international conference on 3D Web technology*, pages 53–58, 2002.

[5] D. Grobgeld. vtkgtk. http://vtkgtk.sourceforge.net/.

[6] N. Hata, T. Wada, T. Chiba, Y. Tsutsumi, Y. Okada, and T. Dohi. Three-dimensional volume rendering of fetal MR images for the diagnosis of congenital cystic adenomatoid malformation. *Academic Radiology*, 10(3):309–312, 2003.

[7] J. Kniss, J. P. Schulze, U. Wössner, P. Winkler, U. Lang, and C. Hansen. Medical applications of multi-field volume rendering and VR techniques. In *Proceedings of the Joint Eurographics-IEEE TCVG symposium on Visualization*, pages 249–254, 2004.

[8] A. Kratz, M. Hadwiger, R. Splechtna, A. Fuhrmann, and K. Bühler. GPU-based high-quality volume rendering for virtual environments. In *Proceedings of international workshop on Augmented environments for medical imaging and computer aided surgery*, 2006.

[9] R. J. Lapeer, R. S. Rowland, and M. S. Chen. Pc-based volume rendering for medical visualisation and augmented reality based surgical navigation. In *IV '04: Proceedings of the 8th international conference on Information visualisation*, pages 67–72, 2004.

[10] S. Potts and T. Möller. Transfer functions on a logarithmic scale for volume rendering. In *GI '04: Proceedings of Graphics Interface 2004*, pages 57–63, 2004.

[11] G. Riva. Applications of virtual environments in medicine. *Methods of Information in Medicine*, 42(5):524–534, 2003.

[12] W. J. Schroeder, K. M. Martin, and W. E. Lorensen. The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In *VIS '96: Proceedings of the 7th conference on Visualization '96*, pages 93–102, 1996.

[13] R. Shen and P. Boulanger. Hardware-accelerated volume rendering for real-time medical data visualization. In *Proceedings of the third International symposium on Visual Computing*, pages 801–810, 2007.

[14] R. M. Taylor II, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. Vrpn: a device-independent, network-transparent VR peripheral system. In *VRST '01: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61, 2001.

[15] T. S. Yoo, M. J. Ackerman, W. E. Lorensen, W. Schroeder, V. Chalana, S. Aylward, D. Metaxes, and R. Whitaker. Engineering and algorithm design for an image processing API: a technical report on ITK - the Insight Toolkit. In *Proceedings of Medicine Meets Virtual Reality*, pages 586–592, 2002.

[16] S. Zhang, C. Demiralp, D. F. Keefe, M. DaSilva, D. H. Laidlaw, B. D. Greenberg, P. J. Basser, C. Pierpaoli, E. A. Chiocca, and T. S. Deisboeck. An immersive virtual environment for DT-MRI volume visualization applications: a case study. In *VIS '01: Proceedings of the conference on Visualization '01*, pages 437–440, 2001.