UC Davis
IDAV Publications

Title
Interactive Multi-Volume Visualization

Permalink
https://escholarship.org/uc/item/28n6b1h0

Authors
Wilson, Brett
Lum, Eric
Ma, Kwan-Liu

Publication Date
2002

Peer reviewed

eScholarship.org Powered by the California Digital Library
University of California

Interactive Multi-Volume Visualization

Brett Wilson, Eric B. Lum, and Kwan-Liu Ma

Department of Computer Science
University of California at Davis
One Shields Avenue, Davis, CA 95616, U.S.A.
{wilson,lume,ma}@cs.ucdavis.edu,
WWW home page: http://www.cs.ucdavis.edu/~ma

From: P.M.A. Sloot et al (Eds.): ICCS 2002, LNCS 2330, pp. 102–110, 2002.
©Springer-Verlag Berlin Heidelberg 2002. http://www.springer.de/comp/lncs/index.html

Abstract. This paper is concerned with simultaneous visualization of two or more volumes, which may be from different imaging modalities or numerical simulations for the same subject of study. The main visualization challenge is to establish visual correspondences while maintaining distinctions among multiple volumes. One solution is to use different rendering styles for different volumes. Interactive rendering is required so the user can choose with ease an appropriate rendering style and its associated parameters for each volume. Rendering efficiency is maximized by utilizing commodity graphics cards. We demonstrate our preliminary results with two case studies.

1 Introduction

Volume rendering has been accepted as an effective method for visualizing physical phenomena or structures defined in 3-space. The advent of texture hardware algorithms [2, 5] and special hardware like the VolumePro [14] make possible interactive volume rendering which is very attractive to many disciplines. While previous volume visualization algorithms were mostly designed for looking at one volume at a time or for animating time-varying volume data [11]. the ability to simultaneously visualize multiple volumes becomes increasingly desirable for many application areas.

In medical research and practice, many imaging modalities including X-ray imaging, Doppler ultrasound, CT scans, MRI scans, etc. are available to physicians for making better diagnoses and surgical plans. While generally the images from different modalities are looked at individually or side by side, it would often beneficial to see two types of data sets simultaneously by spatially superimposing them. For example, it is helpful to visualize functional data overlaid on anatomical data.

In scientific computing, to gain a better understanding of the intrinsic properties of certain physical or chemical process, scientists often try to simulate and study different aspects of the process. The capability to visualize different variables simultaneously describing the same spatial domain and to determine their correlations is thus desirable. For example, in a multi-disciplinary computing environment, several engineering analysis programs, such as structural and flow solvers, run concurrently and cooperatively to perform a multi-disciplinary design. The goal may be to identify the relevant design

variables that can explain the causes of a particular phenomenon, like vortices in a flow field.

Multimodality volume visualization in medical imaging generally must proceed with a registration step because volumes from different modalities almost always have either different resolutions or various degree of distortions. Registration can be as simple as performing a set of linear transformations but can also be a very complex process involving extensive human intervention [1, 15].

Similarly, in many scientific or engineering studies, simulation data and experimental data of different resolutions or on different kinds of computational mesh structures need to be looked at together. Resampling is often done to match one resolution to the other. Registration/resampling by itself can present many challenges. The work reported in this paper focuses on the visualization problems after registration/resampling has been done.

We aim to develop techniques that can generate effective visualizations which reveal both correspondence and distinction between multimodality volume data sets. Our approach to the multi-volume visualization problem is to use:

- highly interactive rendering,
- mixed rendering styles, and
- user-controlled data fusion.

Interactive rendering allows the user to freely change rendering and visualization parameters, as well as data fusion schemes. Rendering different volumes with different styles, if done appropriately, may enhance perception of shape, structure, and spatial relationship. The data fusion problem here is to determine how to display multiple data values defined at the same spatial location. For example, a CT image and MRI image from the same patient can be combined to show both bones and fat clearly in a single image. We feel that the user should be allowed to select a particular method to combine images according to the properties under study. The rest of the paper discusses these topics. Finally, two different data sets are used to discuss the techniques and processes we have developed to achieve more efficient multi-volume visualization.

2 Interactive Volume Rendering

Highly interactive volume rendering can be achieved by using either graphics hardware-assisted methods or a parallel computer. We are particularly interested in utilizing consumer PC graphics cards like the Nvidia GeForce 3. For large data sets, a cluster of PCs is used to distribute both the data and the the rendering calculations.

2.1 Hardware-accelerated volume rendering

Specialized volume rendering hardware such as the VolumePro [14] has been available for realtime volume rendering for several years. Recent lower-cost consumer graphics cards such as the Nvidia GeForce 3 also support volume rendering through the use of volumetric textures [5]. To render a volume using volumetric textures, a series of planes parallel to the screen plane are drawn. These planes intersect the volumetric texture,

which is transformed using OpenGL's texture transformation matrix, and the graphics hardware interpolates the texture in three dimensions. The closer together the planes are drawn, the more accurate (but slower) the resulting volume rendering becomes. This method is limited by the low amount of memory typically available on consumer graphics cards. For example, the GeForce 3 has 64 MB of memory, limiting the volume texture size to $256 \times 256 \times 256$. For higher-resolution data, parallel approaches must be used [12].

A more widely-supported method for hardware-accelerated volume rendering is to map large numbers of 2-d textures onto axis-aligned parallel planes [2]. Generally, 8-bit indexed-color textures are used to minimize memory usage and to allow realtime modification of the colormap and transfer function by palette modification. This method produces images with more visual artifacts than the 3-d texture method, and sets of slices for each axis must be kept to prevent the slices from appearing edge-on. However, 2-d textures are somewhat faster than 3-d textures on current hardware, and can be straightforwardly paged in and out of video memory, so data sets can be displayed which exceed the memory of the video card (although with a significant performance penalty).

Realtime lighting can enhance geometry and structure of data. We currently implement two methods for lighting in hardware, both of which encode normal information in a separate set of textures. The first method encodes the normal at each data point as an 8-bit index, indicating one of 256 direction vectors uniformly distributed in 3-space. These textures are drawn in a separate pass using a palette which maps each normal index to a specular light value for the current view vector. The second method uses the texture-shader features of the Nvidia GeForce 3. The normal vector at each point is encoded in the three color values of an RGB texture. The GeForce 3 interpolates these values for each pixel and computes the dot product with the current view vector. The result is used as a lookup into a texture that encodes a reflection map. This method provides better precision than the first method at a cost of memory usage.

2.2 Parallel volume rendering

Many software algorithms for parallel volume rendering have been developed [9, 10, 13]. In this work, we intend to use hardware-accelerated rendering. One significant limitation of volume rendering using consumer PC graphics hardware is the relatively small amount of video memory. For example, the Nvidia Geforce 3's 64 megabytes of video memory is shared between the frame buffer and texture memory. It is very desirable to fit the volume being rendered entirely in texture memory in order to avoid swapping data into the graphics card from main memory over the relatively slow graphics bus. By subdividing the volume spatially and distributing it across a cluster of PCs equipped with graphics cards, it is possible to render significantly larger volumes into the aggregated video memory of the entire cluster. In addition to the larger amounts of texture memory provided by a PC cluster, performance improvements also result from the combined fill-rate of multiple graphics cards.

A PC cluster we have built recently consists of 17 PCs. The 16 node PCs are connected with a 100-base-T fast Ethernet while the connection to the host PC, which is used for final display and user interface control, is through the cluster's switch with

a gigabit Ethernet. Each PC has an AMD Athlon 1.3 Ghz processor, one gigabyte of PC133 SDRAM and a GeForce 3.

Our current implementation of the renderer subdivides and distributes the volume using k-d tree subdivision. Each node PC resamples it's subvolume to a size of $256 \times 256 \times 256$ regardless of the dimensions of the actual volume. For smaller volumes this permits higher order resampling to be done in software prior to the tri-linear interpolation done in hardware. During rendering, for each frame, every node on the cluster renders its subvolume and composites the resulting subimage using binary-swap [13] with the final image being sent to the host for display. Presently, using 8 of the node PCs, we are able to render $512 \times 512 \times 512$ voxels to a 512×512 window at about 1-2 frames per second. The current implementation can switch between high and lower resolution modes based user's need for interactivity. For example, rendering a scaled down version of the data instead like $256 \times 256 \times 256$ voxels, over 10 frames per second can be achieved.

3 Mixing Rendering Styles

When rendering multiple volumes, it is important that visual cues be present to help differentiate the volumes. This can be achieved by varying the color, lighting, as well as rendering style used for each of the different volumes. For example, Hauser et al. [7] show how to efficiently perform both maximum intensity projection and direct volume rendering to generate effective visualizations. In addition, the mixing of photorealistic and non-photorealistic rendering styles can be particularly effective in not only differentiating the two volumes, but also in drawing focus to features of interest.

3.1 Nonphotorealistic rendering

Non-photorealistic rendering involves the application of techniques used by artists for the creation of computer generated imagery. These techniques have been applied to scientific visualization for the creation of imagery that can be more meaningful than that generated with more traditional photorealistic techniques. Non-photorealistic rendering is usually associated with the representation of surfaces, but has also been applied to direct volume rendering [3, 12]. Through the application of several non-photorealistic rendering techniques it is possible to accentuate key features in a volume, while de-emphasizing structures that might obstruct or detract from those features.

An artist rarely depicts shading by simply making colors darker, but instead relies on variations in both light intensity as well as color temperature or tone to indicate illumination. For example, shadows are often shown in cooler blues, while directly illuminated regions are shown with warmer yellows and reds. Gooch et al. [6], describe how tone shading can be applied to the rendering of surfaces for the creation of illustrations. We implement tone shading by modulating the rendered volume with a volumetric tone texture. This texture consists of a paletted texture of gradient directions with a palette that varies from cool to warm depending on the lighting of each gradient direction as calculated using a standard Phong shading model.

Silhouette rendering consists of adding dark lines around an object and can be very effective in enhancing fine structures. They can also aid in depth perception when viewing overlapping structures of similar color. Silhouette rendering is accomplished in a

second rendering pass that modulates the transfer function with a texture that is dark and highly opaque in those regions with a normal perpendicular to the view, and highly transparent in those regions parallel to the viewer. This texture is obtained by using a paletted texture with gradient directions and specifying a palette such that each normal has an opacity that increases with degree silhouette should be shown.

Through the variation of color based on distance, depth perception can be improved [4]. In particular, color can be varied in temperature as well as intensity depending on the distance from the viewer. We implement this non-photorealistic rendering style by modulating each textured polygon drawn in the rendering processes with a color that is selected based on the distance from the viewpoint. The user is able to interactively specify how color is manipulated based on distance using a transfer function styled interface that maps distance to a color and opacity map. Thus, closer objects can be mapped to warmer colors compared to cooler distant objects, improving perception of the depth relationship between these objects.

4 Data Fusion Schemes

When rendering multiple volumes simultaneously, we must decide how to treat multiple values defined at the same spatial location. There are basically three approaches to this data fusion problem:

1. using one of the values based on some criterion
2. using one value which is weighted by a function of some or all of other values
3. using one value for each color channel

An example for the first approach is the alternating sampling used in [8] for rendering two volumes. The second approach gives us more freedom. For example, we could use the opacity transfer function for one volume to enhance or de-enhance some aspects of the other volume. This is similar to the common practice of volume visualization in which gradient magnitude is used to enhance boundary surface. Furthermore, a weighted sum might be used with scalings that reflect a desired property, such as distance from the viewer [12]. The third approach is probably the simplest for one to implement and to verify its results. While it is limited to visualization of three or fewer volumes, in practice we hardly need to see more than three volume simultaneously.

We have implemented a suite of combining operations based on these three approaches. Most importantly, our system allows the user to interactively select a particular way to present multimodality data. We found this capability aids to the multi-volume data exploration process.

5 Results

We have been studying multi-volume visualization using data sets generated from medical imaging, biological data imaging, and computational fluid dynamics simulations. In the following sections, case studies on a mouse data set and a plant microbiology data set are presented to summarize our initial experience. Alternating sampling was used for data fusion for both data sets.

5.1 Case study I: a mouse data set

We use two volumes from small animal imaging: a Positron Emission Tomography (PET) data and an MRI scan of a mouse head. Each of these data sets consists of 47 slices at 256×256 pixels per slice. Prior to visualization, the data was cropped to remove a significant amount of empty space, and resampled to 100 slices at 256×256. On a single PC, we can render to 500×500 pixels window without lighting at 25 frames per second, with one volume lit at 7 frames per second, and with both volumes lit at 1 frame per second.

The left image in Figure 1 shows high density regions of the PET data set in green, indicating the region of higher brain activity. The blue areas show a narrow range of values present in the MRI data set. This allows some anatomical structure to be shown while keeping most of the PET scan data visible. White specular lighting is applied to the MRI data to help define its shape.

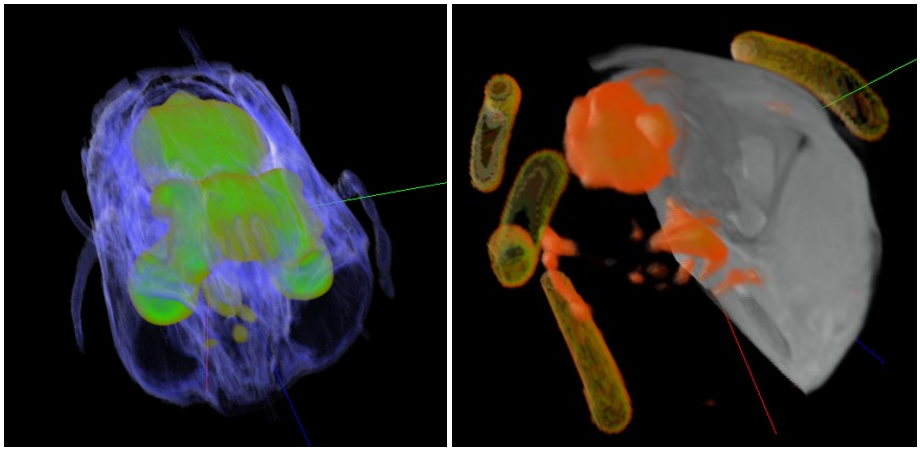

Fig. 1. Simultaneous visualization of PET and MRI mouse data.

The right image in Figure 1 shows high density PET data in red with specular highlighting. The large tube structures around the data are alignment markers to aid in image registration. The MRI data is shown in gray with a clipping plane to allow a portion of the PET data to be visible. This MRI data is more difficult to visualize because it consists of large regions of relatively constant value separated by small regions of other values. This makes it hard to show structure while keeping the image transparent enough to reveal deeper structure. In addition, noise causes specular lighting to be grainy and distracting.

In each case, the PET data was much easier to visualize than the MRI data. The PET data is smooth and present in large, mostly convex areas. This makes it easy to see desired structure by selecting a cutoff for displayed densities, and the smooth contours produce smooth, consistent specular highlights. By contrast, the MRI data has a lot of fine detail at many different densities. A partially transparent MRI image reveals too

many levels of detail to be useful, while a mostly opaque MRI image obscures other data we may want to see (such as the PET data).

5.2 Case study II: roots data set

In our second case study we examine a root data set generated using confocal microscopy. Two volumes were obtained based on the reflected light emission of two different fluorescences. The first volume consists of a root from the species Melilotus alb, which has the common name white clover. The other volume contains mycorrhizal fungi attached to the root. Each volume has $512 \times 512 \times 256$ voxels. The scientists we worked with were interesting in studying the symbiotic association that exist between the plant and fungus.

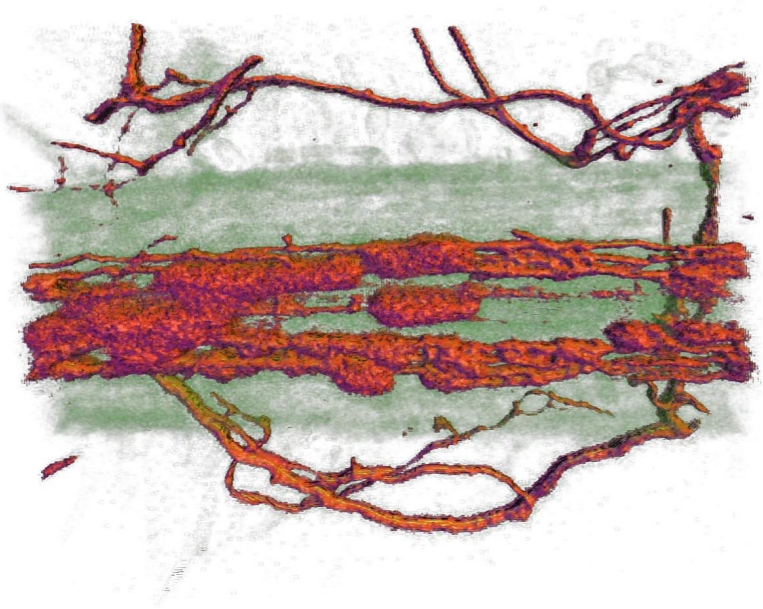

Fig. 2. Mixed-style rendering of plant root data from confocal microscopy. The red part displays the cover of the fungi and the green part shows the extent of root.

In Figure 2 we see a hardware rendered image of the root and fungus generated with our technique. The fungus, shown in red, is rendered using a transfer function that makes it appear opaque, allowing it's fine tubular structures to be readily visible. Lighting parameters have also been specified to make the shape more obvious. The root, rendered in green, is shown to give the fungus structure context. It is of less importance to the scientist and is therefore shown with relatively transparent opacity map. Lighting is less visible for the root since it would draw attention away from the fungus being studied.

Using eight nodes of our PC cluster to render this data, we are able to achieve over three frames per second. Using a single PC, we would have to scale down the data to maintain the same interactivity.

6 Conclusions

Simultaneous visualization of volumetric data from multiple modalities is challenging because of the disparity in data resolutions, and the higher storage and processing requirements. In our preliminary study using data from medical imaging and plant microbiology, we address some of the most relevant issues including rendering models, data fusion, and high-performance rendering.

We have shown that multi-modality volume visualization can aid in the understanding of volumetric medical data. The additional information provided by simultaneous visualization allows areas such as tumors or active regions of the brain to be more precisely located relative to each other and to the anatomy of the animal.

We have also shown that appropriate shading models and use of transparency in multi-volume visualization provide information not available from the use of a single, traditional rendering model. The non-photorealistic rendering we used helps bring out information about the geometry and boundary of a structure, and the customizable transparency provides information about the relative positions and size of two or more structures, and allows the researcher to highlight the most important details of an image.

We intend to create a system framework for further study. Other future work directions include refining our mixed-style rendering model to derive more effective illustrations, conducting a comprehensive study of the data fusion problem, and studying other types of multi-volume data,

Acknowledgments

This work has been sponsored in part by NSF PECASE, NSF LSSDSV, and DOE SciDAC. The authors are grateful to Dr. Juan Jose Vaquero, Dr. Michael Green, the National Institutes of Health, and UCLA Ann Hirsch Laboratory for providing the test data sets.

References

1. Maintz, A. and Viergever, M.: A survey of medical image registration. Medical Image Analysis, vol. 2, no. 1. (1998) 1–36.
2. Cabral, B., Cam, N., and Foran, J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. 1994 Workshop on Volume Visualization (October 1994) 91–98.
3. Ebert, D. and Rheingans, P.: Volume illustration: non-photorealistic rendering of volume models, IEEE Visualization 2000 Conference. (October 2000) 195–202.
4. Foley, D. J., Van Dam, A., Feiner, S. K., and Hughes, J. F.: Computer Graphics: Principles and Practice. Addison Wesley, 1996.

5. Van Gelder, A. and Hoffman, U.: Direct Volume Rendering with Shading via Three-Dimension Textures, 1996 Symposium on Volume Visualization. (1996) 23–30.
6. Gooch, A., Gooch, B., Shirley, P., and Cohen, E.: A non-photorealistic lighting model for automatic technical illustration, SIGGRAPH '98 Conference. (July 1998) 447–452.
7. Hauser, H., Mroz, L., Bischi, G.-I., and Groller, E.: Two-level volume rendering - fusing MIP and DVR. IEEE Visualization 2000 Conference. (2000) 211–218.
8. Hastreiter, P., and Ertl, T.: Integrated registration and visualization of medical image data. Computer Graphics International (1998) 78–85.
9. Lacroute, P.: Real-time volume rendering on shared memory multiprocessors using the shear-warp factorization. 1995 Parallel Rendering Symposium (October 1995) 15–22.
10. Li, P. P., Whitman, S., Mendoza, R., and Tsiao, J.: ParVox – a parallel splatting volume rendering system for distributed visualization. 1997 Symposium on Parallel Rendering (October 1997) 7–14.
11. Lum, E. B., Ma, K.-L., and Clyne, J.: Texture hardware assisted rendering of time-varying volume data. IEEE Visualization 2001 Conference. (October 2001) 263–270.
12. Lum, E. B., and Ma, K.-L.: Hardware-accelerated parallel non-photorealistic volume rendering. International Symposium on Non-Photorealistic Animation and Rendering, Annecy, France (June 2002).
13. Ma, K.-L., Painter, J., Krogh, M., and Hansen, C.: Parallel volume rendering using binary-swap compositing. IEEE Computer Graphics & Applications. (July 1994) 59–68.
14. Pfister, H., Hardenbergh, J., Knittel, J., Lauer, H., and Seiler, L.: The VolumePro real-time ray-casting system. ACM SIGGRAPH '99 Conference. (1999) 251–260.
15. Woods, R. , Grafton, S., Holmes, C., Cherry, S., and Mazziotta, J.: Automated image registration: II. intersubject validation of linear and nonlinear models. Journal of Computer Assisted Tomography. (1998) 22:155–165.