Immersive 4D Volume Visualization in CAVE

Yuki Yamaura[1], Youhei Masada[1], Akira Kageyama[1], and Kouhei Yamada[1]

[1]Graduate School of System Informatics, Kobe University, Japan

1 Introduction

CAVE is a room-sized immersive virtual reality system developed by Electronic Visualization Laboratory at University of Illinois, Chicago [1]. A viewer in CAVE room, which is surrounded by screens on which stereo images are projected, can observe virtual 3 dimensional (3D) objects from any position, angle and direction through head tracking technology. The viewer can interact, in addition, with the objects using a potable controller, called wand, which is also under the tracking control. Not only for the demonstration of virtual 3D objects, it has also grown in its importance as a tool for visualization of simulation data aimed at the scientists, and has been used in various research fields [2]

The amount of simulation data generated on supercomputers becomes larger year by year with increasing supercomputer specifications, such as processing power and storage capacity. As scientists gain access to more powerful computers, they attempt to tackle larger, more complex problems. Progressive approaches are then required for the visualization of such huge and complex simulation data in CAVE. For the more efficient data analysis and visualization, it is reported in this paper that we had developed a 4D volume rendering library especially for displaying time-evolving, sequential, 3D scalar data in CAVE.

2 3D Texture Slicing

Among some visualization methods for time-evolving 3D scalar data [3], we choose 3D texture slicing technique which is one of volume rendering methods. A standard algorithm of 3D texture slicing is summarize as follows:

1. conversion of 3D scalar data into 3D texture data with RGB and Alpha using a transfer function we defined in preprocess.
2. generation of a number of slice planes perpendicular to the viewer's line of sight.
3. mapping of 3D texture onto the planes and drawing it in back-to-front order.

There exists two procedures to transfer time-evolving 3D scalar data processed by 3D texture slicing method from CPU RAM to GPU memory. It would be the simplest that all the time-sequential 3D data is transferred at first to GPU memory and then is cached for the visualization. Since all the data must be stored in GPU, a large size of memory is required for this procedure. In contrast, we can take an another approach in which 3D data at each time step is transferred one after the other when needed. The latter one is adopted in our 4D volume rendering library because it can save GPU memory and then be applicable to huge times-sequential data generated on massive supercomputers.

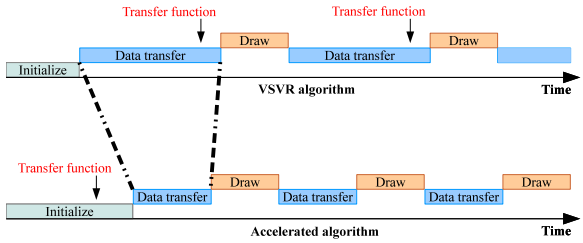

Fig. 1. Comparision between VSVR algorithm and our new algorithm.

3 Implementation: 4D Volume Rendering

A Very Simple Volume Rendering code, VSVR, which is developed by Thomas Lewiner on the basis of basic OpenGL features for the efficient visualization of 3D scalar data [4], is adopted as a reference source code to implement our fast volume rendering library.

Two new codes are installed into VSVR to make it work with CAVE systems and to get faster rendering of 3D scalar data.

3.1 Immersion

The original VSVR is specified to commonly-used 2D displays, and thus supposes at all times a situation in which viewer's line of sight is perpendicular to the display. The slice plane is then set consistently to be parallel to the screen. However in CAVE, it is not the case. Since the angle between viewer's line of sight and a normal vector of screens is changed at every moment, we should make the slice plane being always perpendicular to the line of sight, not to the screen, in CAVE systems. By taking in real time the information of viewer's line of sight using the head tracking system, the direction of slice planes is regulated incessantly in our rendering procedures.

We use CAVELib to display volume data in CAVE systems. Since a program with CAVELib (CAVE application) is generally multi-processed or multi-threaded, we have to synchronize visualization data generated by multiple processes/threads. For an instantaneous synchronization, we use a double buffering technique.

3.2 Pixel Buffer Object and Rendering Algorithm

VSVR uses glTexture3D function not only for transferring data from CPU to GPU, but also for converting scalar data to texture data with RGB, and Alpha. This function is easy to use, but takes a long time to transfer data. We thus take an alternative approach using Pixel

Buffer Object (PBO), instead of this function, to accelerate rendering speed. Since PBO stores pixel data into the buffer object, we can transfer 3D data faster to and from a graphics card through direct memory access. When adopting PBO, we have to convert scalar data to texture data, in other word, to apply a "transfer function" by ourselves. The scalar data is transformed at first into initialization block in our library to reduce rendering time.

The new algorithm we proposed for realizing a fast 3D volume rendering of time-sequential data in CAVE is organized as follows:

1. conversion of 3D scalar data into 3D texture data with RGB and Alpha using a transfer function by ourselves.
2. transfer of 3D texture data from CPU RAM to OpenGL memory as PBO and generation of 3D texture from PBO.
3. generation of a number of slice planes perpendicular to the viewer's line of sight.
4. mapping 3D texture to the planes and drawing it in back-to-front order.
5. forwarding a time step to the next.

(This cycle is repeated at every time step).

A schematic picture in Fig. 1 presents a new algorithm we proposed for the fast 3D volume rendering in CAVE in comparison with the algorithm adopted in VSVR.

4 Volume Rendering Applications in CAVE

Using our 4D volume rendering library, we have developed an application framework for CAVE, called Multiverse [5][6]. Muliteverse is itself a CAVE application and can launch several visualization programs. In Multiverse, we have employed our new volume rendering library in two visualization programs: SeismicWave program visualizes a simulation data of propagating seismic-wave, and CellDivision program is for a 3D time-sequential microscopic images of mouse embryo.

Fig. 2. The demonstrates of a CAVE application "SeismicWave" in which time sequential simulation data of seismic-wave propagation is displayed in VR space. The simulation data adopted in this application is defined on $256\times512\times160$ grids and has a total size of about 100Mbyte for each time [7]. Note that, in the SeismicWave program, hundreds of time-sequential data are continuously loaded and is smoothly exhibited in CAVE system using our fast volume rendering library.

5 Conclusion

We have developed a volume rendering library based on 3D texture slicing method for CAVE systems. This library has utilized several techniques for accelerating rendering speed, such as using the transfer function in preprocess and adopting PBO for data transfer. To verify the effectiveness of the new algorithm we employed in our volume rendering library, we have designed two applications to visualize large scale 3D scalar data in CAVE systems, that is SeismicWave and CellDivision. Our 4D volume rendering library is applicable to all the CAVE application developed on the basis of OpenGL. We believe that it would be helpful for efficient and effective data analysis in various research fields, and might boost scientific findings by using CAVE.

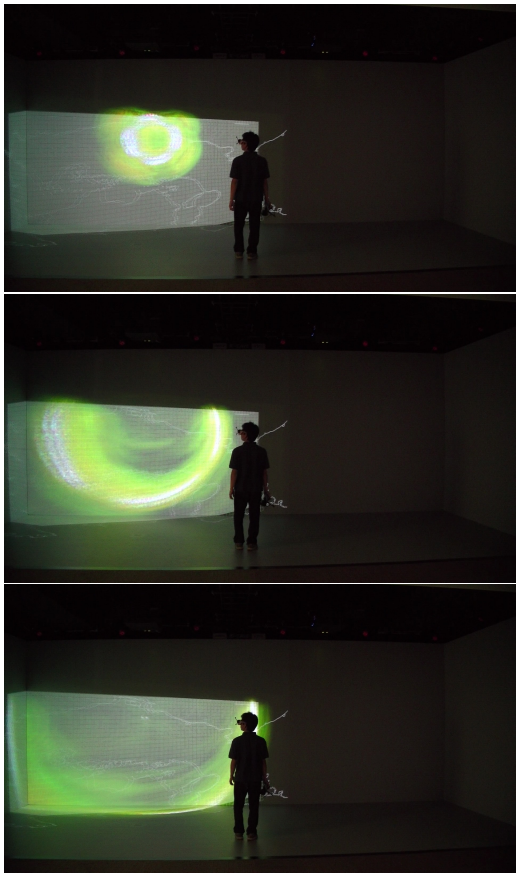

Fig. 2. Time sequence of snapshots of seismic-wave propagation simulation.

Acknowledgement

The seismic wave simulation data shown in Fig. 2 was provide by Prof. T. Furumura at University of Tokyo.

References

[1] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, Surround-screen projection-based virtual reality: the design and implementation of the CAVE, Proc. SIGGRAPH '93, 135-142, 1993

[2] Kageyama, A., Miyagoshi, T., & Sato, T., Formation of current coils in geodynamo simulations, Nature, vol.454, 1106-1109, 2008

[3] A. C. Telea, Data Visualization, A K Peters, Ltd., Florida, 2007

[4] T. Lewiner, VSVR: a very simple volume rendering implementation with 3D textures, http://zeus.mat.puc-rio.br/tomlew/tomlew_uk.php

[5] A. Kageyama, Y. Yamaura, D. Meno, Y. Masada, K. Yoshizaki, and K. Yamada, Application Launcher for CAVEs, International Conference on Modeling and Simulation Technology (JSST2011), Tokai Univ., Takanawa, Japan, 2011

[6] Y. Yamaura, D. Meno, A. Kageyama, Y. Masada, K. Yoshizaki, and K. Yamada, Development of an application launcher for CAVE systems, VRSJ the 16th Annual Conference, 69-72, 2011

[7] Furumura, T., B.L.N. Kennett and K. Koketsu, Visualization of 3-D wave propagation from the 2000 Tottori-ken Seibu, Japan earthquake: Observation and numerical simulation, Bull. Seism. Soc. Am., 91, 4, 667-682, 2003