AN INTERACTIVE 3D VISUALIZATION MODEL BY LIVE STREAMING FOR REMOTE SCIENTIFIC VISUALIZATION

Eriko Touma, Satomi Hara, Mari Kurumi, Yuri Shirakawa, Chisato Ishikawa, Masami Takata
Graduate School of Humanities and Sciences, Nara Women's University, Nara-city, Japan

Takeshi Horinouchi, Kazuki Joe
Research Institute for Sustainable Humanosphere, Kyoto University, Uji-city, Japan
Graduate School of Humanities and Sciences, Nara Women's University, Nara-city, Japan

Keywords: 3D Visualization, Live Streaming, GPGPU, Geophysical Fluids.

Abstract: Recent improvement of high-end GPUs has made it possible to perform real-time 3D visualization such as volume rendering and 3D contour plot for scientific data locally. A web browser based remote 3D visualization by visualization servers is attractive, but data transfer overhead prevents from performing interactive operations. We propose an interactive remote 3D visualization model by live streaming for geophysical fluids research. In this model, we use live streaming flash media for the web browser based operations keeping minimum quality of data analysis and minimum bit rate for live streaming of flash media. Preliminary experiments with a prototype system validate the effectiveness of our proposing model.

1 INTRODUCTION

Visualization is an effective and attractive tool for analyzing numerical data of scientific phenomena. To find out a peculiar phenomenon from huge amount of numerical data, visual understanding is sometimes more efficient than just numerical analysis by computers. Thus, visualization techniques are to be indispensable in the fields where huge numerical data analysis is ineluctably faced. In recent years, even researchers without deep knowledge of 3D visualization theory can make use of 3D visualization such as volume rendering and 3D contour by using 3D visualization software. For the actual and effective use of 3D visualization, they must prepare a high-end computer and install required software packages and/or plug-ins of which number is sometimes considerably large. These limitations prevent new users of scientists from using 3D visualization for their research. If 3D visualization environments were easily available for any researchers such as desktop-publishing tools, the number of new users of 3D visualization would increase drastically. Considering current client-server environments or grid computing, it is natural to think of 3D visualization servers: any remote users can use 3D visualization servers through a 3D visualization web browser.

Web3D technologies, which have been paid attention to in recent years, are modeling based, i.e. polygon based rendering. It is mostly used for product advertisement and is not suitable for scientific visualization, which is volume based, i.e. voxel based rendering. Regardless to say, the volume based rendering is more computationally intensive than the modeling based rendering. This is the reason of having the 3D visualization servers. Although the improvement of enhanced low cost PC is remarkable, it is still far from the performance of real time volume rendering, for examples.

To get interactive 3D visualization of scientific data, the response time of 3D visualization is essential rather than the resultant quality. Just the 3D visualization servers are inefficient because of data transfer overhead. In short, typical 3D visualization servers are not suitable for real time 3D visualization in scientific research. We are interested in an interactive 3D visualization environment via a web browser for geophysical fluid science in which huge data must be treated.

In this paper, we propose an interactive 3D visualization model by live streams for remote scientific visualization via a web browser. By using the proposed model, users can apply volume rendering and 3D contour to their target data located in a remote database

Touma E., Hara S., Kurumi M., Shirakawa Y., Ishikawa C., Takata M., Horinouchi T. and Joe K. (2008).
AN INTERACTIVE 3D VISUALIZATION MODEL BY LIVE STREAMING FOR REMOTE SCIENTIFIC VISUALIZATION.
In *Proceedings of the Third International Conference on Computer Graphics Theory and Applications*, pages 409-414
DOI: 10.5220/0001095404090414

via the web browser. Visualized objects displayed on the browser can be observed from any direction with any scales interactively. GPGPU is adopted for the calculations of real-time volume rendering and 3D contour plot. The proposed model is implemented as a 3D visualization function for Gfdnavi introduced in section 2, which is a tool to produce web-based data services for geophysical fluid sciences. The 3D visualization function is based on our 3D visualization implementation model(E. Touma and Joe, 2007) which enables huge data to be analyzed easily, quickly, and efficiently.

The rest of the paper is organized as follows. In section 2, we introduce Gfdnavi, a tool to construct a data server for geophysical fluid sciences. In section 3, we propose a web browser based 3D visualization function for Gfdnavi. The model of interactive 3D visualization via a web browser is the key concept in our proposal to be described in section 4.

2 GFDNAVI

A group of geophysical fluid scientists has recently developed a software package named Gfdnavi(T. Horinouchi and Shiotani, 2007)[1]. It is a tool to produce web-based data services for geophysical fluid sciences. The package of Gfdnavi includes a special-purpose web-server program, so one can start up web-based data services just by installing it. It also runs with commonly-used web servers such as Apache. Therefore, it is suitable both to use within local hosts and to operate servers to make data publicly available. Also, it is suitable to share data among a group of scientists. Gfdnavi creates a metadata database of numerical data automatically by scanning a tree of local directories under the top directly that a Gfdnavi-server maintainer specifies. It supports a few file formats frequently used in geophysical fluid sciences. Gfdnavi enables its clients to search, analyze, visualize, and download data with web browsers. It is implemented by using the web development framework Ruby on Rails. This framework naturally enforces the Model-View-Controller (MVC) architecture to the products constructed with it, where the model typically represents a relational database. The metadata database of Gfdnavi archives the structure of the directory tree. It also supports searches by text and spatio-temporal parameters such as searches using global maps.

In Gfdnavi, metadata extraction from numerical data is done through GPhys(T. Horinouchi and Takehiro,), a class library in the Ruby language to handle multi-dimensional data of geophysical fluids. It abstracts data formats based on a data model close to the one used in NetCDF, thereby enabling unified access to data in multiple formats such as GRIB and GrADS.

A Gfdnavi user first selects numerical data either from the directory tree or from the metadata search. Then the data can be visualized. Currently, it supports visualization up to 2D. Gfdnavi does not require any plug-ins, so a user can use it only by using a web browser.

Currently, a Gfdnavi server provides data on local disks. However, a development is underway to support search and use of data across multiple Gfdnavi servers. It is envisioned to create an overlay network of Gfdnavi to realize virtual unification geophysical fluid data access across the network.

3 3D VISUALIZATION FUNCTION FOR GFDNAVI

As described in the previous section, Gfnavi provides just 2D visualization functions. Using 3D visualization, it is possible for geophysical fluid scientists to understand a lot of information at a time intuitively. For this purpose, we implement the 3D visualization functions in Gfdnavi. In this section, we explain the design concept of 3D visualization function for Gfdnavi with some details of required elemental technologies.

For the appropriate implementation, we enumerate our design concept below.

i) Efficient analysis of huge data
ii) Simple operations without any manual
iii) OS-independent
iv) High performance visualization
v) Web browser based operations
vi) Interactive operability

We use our 3D visualization implementation model (E. Touma and Joe, 2007) for the implementation of 3D visualization functions to satisfy i) and ii). We do not explain the detail of the implementation model because of lack of space.

For the concept iv), we take the advantage of GPGPU. It is natural that the surplus computing ability is used for a part of calculation which should be performed by CPUs, if GPUs perform programmable pipeline operations. Nowadays, GPGPU is rapidly used for various visualization tasks. We also use GPGPU for Gfdnavi 3D visualization functions: volume rendering and 3D contour plot.

[1] http://www.gfd-dennou.org/arch/davis/gfdnavi/

The 3D visualization function of Gfdnavi should also provide a web browser based user-interface, and it satisfies the concept v). The 3D visualization results are displayed by Flash animation to satisfy iii). In our implementation of web browser based 3D visualization functions, target objects should be rotated, zoomed and moved by simple mouse operations, regardless to say, interactively. It satisfies the concept vi).

4 REMOTE 3D VISUALIZATION BY LIVE STREAMING

3D visualization was a computationally intensive task sometimes requiring supercomputers. As the improvement of GPU technologies, now it comes to be almost an interactive task for reasonably small local data. During the design of 3D visualization function of Gfdnavi, we have faced a difficult implementation problem. As explained in section 2, Gfdnavi is a tool for client-server or grid computing environments. Adding 3D visualization functions to Gfdnavi with a web browser based method, we need to determine who visualizes target data. In the case that a server performs the computation, the server needs to transfer a lot of resultant snap shot images to clients rapidly. Obviously there is a problem of data transfer overhead to perform the interactive remote 3D visualization. On the other hand, when a client (a web browsing PC) performs the computation, the client must be equipped with a latest GPU as well as GPGPU application environments. Furthermore, the client must download the whole target data in advance of the computation. Since the purpose of Gfdnavi is to provide geophysical fluid scientists with various analysis functions as well as data, the former approach of server rendering is preferable.

4.1 3D Visualization Model for Remote Data

In our method, visualization runs on a visualization server, which means server side rendering. The visualization results are delivered to client users as live streaming movie. In other words, client users can observe the visualization results as real time animation via a web browser. When a point of display area in the web browser is clicked or dragged by mouse, an operation command is transmitted to the visualization server. The visualization server reflects the transmitted operation command so that it changes visualization results: rotate, move and zoom, interactively. In this way, interactive 3D visualization for remote data by streaming distribution is obtained.

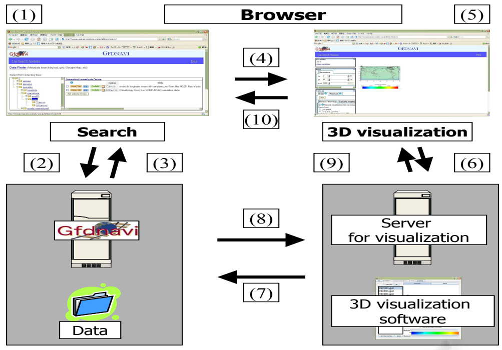

Figure 1: The flow of the interactive 3D visualization process.

Figure 1 briefly explains the flow of the interactive 3D visualization process. A user accesses to a Gfdnavi server via a web browser. The Gfdnavi server contains data for geophysical fluid research, which can be browsed from the data search page shown as (1) of figure 1. When the user chooses the data that the user wants to analyze, an operation command is send to the Gfdnavi server as shown in (2). In (3), the Gfdnavi server sends the information about the data that the user chose. The information is displayed in (1). When the user executes 3D visualization, the user goes to the page of a visualization server as shown in (4) and (5). The information of data chosen and the request of 3D visualization by the user are sent to the visualization server at (6). The visualization server asks the Gfdnavi server for the data and receives the data as shown in (7) and (8). After downloading the whole data, 3D visualization is applied to the data according to the user's requests. The visualization server takes snap shot images of the 3D visualization results at a regular interval. An animation clip is generated from these snap shot images to be distributed as a streaming Flash file at (9). In the browser of (5), the user can observe the visualization result by animation delivered from the visualization server in live streaming. When the user clicks and drags by mouse on the stream animation display area, the information of the mouse position and operation is sent to the visualization server as a command to request the corresponding visualization process such as rotating, moving, and zooming. The resultant visualization is delivered back to the client in live streaming, namely interactively. In the same way, the user can request a visualization method of 3D contour plot or volume rendering as well as parameter setting in real time. When the user wants to change data, the user goes back to the data search page shown as (10).

4.2 Advantage of the Model

In our proposing model, geophysical fluid scientists can observe the results of 3D visualization via a web browser without installing any visualization packages and with the benefit of high-end 3D visualization resources. Live streaming by Flash animation is OS-independent. Although the installation of a plug-in is required for watching the Flash animation, the Flash player is a popular plug-in and easy for users to install. Actually, most web browser users have installed the Flash player plug-in because Flash media is widely common. The benefit of streaming media is that clients need not download the movie data into their hard disks. It saves client users from considering about the storage capacity to watch as long movies as they want.

In our model, the process of 3D visualization, movie generation of visualization results, and live streaming distribution is performed on a visualization server. In this process, it is unavoidable that dropping frames sometimes occur because of regular sampling of snap shot images. During scientific data analysis with 3D visualization, the essential requirement is to guarantee that scientists surely recognize the display changes caused by their operations. In our model, a display of a 3D visualization result does not change unless a new command is issued. Therefore, dropping frames are not a critical problem for our model unlike stream distribution of animation which always changes their display.

4.3 Implementation

4.3.1 Interactive 3D Visualization for Remote Data

The 3D visualization functions executed in visualization servers include volume rendering and 3D contour plot by GPGPU with Cg as a shader language. The Cg language can be used from OpenGL APIs. For volume rendering and 3D contour plot, we use VTK, which supports OpenGL as low level APIs. We develop the 3D visualization functions in Cg so that they are callable from any VTK program[2] as well as Gfdnavi. Taking the advantage of collaboration with GPGPU, the 3D visualization functions of the visualization server will achieve high performance visualization.

We here explain the performance issue. Namely, how fast GPGPU based 3D visualization is? As for volume rendering by GPGPU, it has been reported that the frame rate of displaying 166MB volume data consisting of a (512,512,333) array on an Nvidia Quadro FX 3400 GPU is 15.3fps(F. Schulze and Hadwiger, 2007). In the case of 3D contour plot, it has been reported that the frame rate of displaying 588MB volume data consisting of a (700,700,600) array on an ATI Radeon X800 with 256 MB memory is 28.7fps(B. von Rymon-Lipinski and Keeve, 2005). Although the frame rates change by the complexity of the volume data, considering the reported GPUs are relatively old, the real-time 3D visualization (30fps) is almost possible on the latest high-end GPUs as described in the next subsection.

In our model, the visualization server receives operation commands from client users, and the 3D visualization functions are invoked by the commands. Receiving commands, if necessary, the visualization server needs to get target data from a DB server (may be equivalent to a Gfdnavi server). The data dealt with geophysical fluid research mainly consists of four dimension arrays, and is so large in size. Downloading the whole data in advance of 3D visualization to the visualization server, the waiting time may sometimes irritate client users. To solve the problem, we use the Gphys library of Gfdnavi, which clips the three dimension domain specified by users. Clipping just the necessary data, the data size is reduced to shorten the data transfer time.

In our implementation, we adopt a double transfer scheme assuming that the network bandwidth between the visualization server and the Gfdnavi server is enough high such as two nodes in a PC cluster with a high speed interconnection network. According to a data transfer request, when the target data is larger than the pre-defined size, Gphys is invoked to clip and performs partial data transfer. At the same time, the whole data is transferred to the visualization server in background. This background data transfer is expected to complete until the client user makes another clip request after several visualization operations. If so, the new clip is obtained in local, i.e. from the pre-loaded whole data in the visualization server. If not so, the requested clip is transferred from the Gfdnavi server again keeping the background whole data transfer. In this way, we ensure the interactive operation for web based client users keeping from the huge data transfer overhead.

4.3.2 Live Streaming

3D visualization results are taken as snap shot images with 320×240 at every $\frac{1}{30}$ seconds. An mpeg animation is generated from those images at every sec-

[2] We know VTK5.2 will officially support Cg and GLSL, but we don't like to wait for the future release.

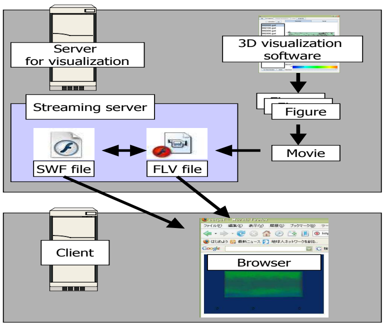

Figure 2: The flow of the live streaming distribution.

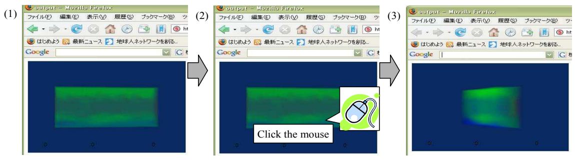

Figure 3: The experiment results.

ond using ImageMagick. Then, the mpeg data is converted to FLV using FFmpeg. In order to distribute the FLV data in a streaming form, we set up a Flash streaming server with red5. A Flash SWF player is created to run in a client web browser. ActionScript is used to call and play the FLV data that is in the visualization server. ActionScript also performs the acquisition of mouse event information on the Flash player. The other interface parts, buttons and sliders for parameter regulation, are implemented in JavaScript. The flow of the live streaming distribution is shown in Fig. 2.

In our system, the recommendation bit rate for watching the live streaming animation is 518K bps. The recommended bit rate is calculated by $BPP \times fps \times width \times height$, where BPP is 0.225 that is the right value for sports or music video. Note that the famous ”YouTube” adopts 500Kbps as the recommendation bit rate, so our recommendation bit rate is not a special (too expensive) case.

4.4 Experiments

To validate our model, we perform experiments using a prototype system. First, we show the effectiveness of interactive operation under some constraints. Then we show the high performance visualization of real-time volume rendering.

The prototype consists of two PCs connected by typical LAN. The client is a standard desktop PC while the server is a PC with Pentium D CPU (2.66GHz X 2) and 512MB memory. The client PC has a web browser with a Flash player where mouse events are acquired to send commands to the server. The server performs a simple volume rendering program written with VTK. The volume data is a small 3D array of 144 by 73 by 30 (2.48MB). Receiving a command from the client, the volume data is rotated to generate ten snap shot images as a BMP format with the size of 320 by 240 (230KB per image). Those BMP files are converted into an mpeg movie by ImageMagick, and the mpeg movie are also converted to an FLV movie (9KB for 10 frames assuming 500Kbps) by FFMPEG. After the creation of the FLV file, ten BMP files and one mpeg file are deleted. Then, the server sends a message to notify the client of the creation of the FLV file. When the client receives the message, the Flash player refreshes the display to load the new streaming FLV movie.

In the client, mouse events acquisition, command issuing and movie display are performed by ActionScript while display refresh is performed by JavaScript. The server and the client communicate in socket connection to exchange messages. All the tasks in the server are controlled by a Perl script.

Figure 3 shows the experiment results. A visualization result is displayed as Flash video in (1). By mouse dragging as shown in (2), a new visualization result is displayed in (3). In this way, the interactive 3D visualization by live streaming is shown to be possible.

In this experiment, the response time from a mouse click at the client to the re-display is 4.071 seconds. We explain each process time among the response time. It takes 0.103 seconds from the mouse click to the beginning of command issue to the server. In the server, ten executions of volume rendering require 1.46 seconds. Ten image files creation takes 0.21 seconds. Generating an FLV file from ten BMP files requires 1.133 seconds. The load time for the new stream is 0.14 seconds. The rest of 1.025 seconds is communication overhead.

The longest time consuming task is the volume rendering part of 1.46 seconds. Note that this volume rendering is performed in a CPU (not using GPGPU). To confirm the real-time 3D visualization, we perform the second experiment. Using GPGPU, we apply volume rendering to a volume data set with the size of 512 by 512 by 512 (134.7MB). We obtain the performance of 19.78fps and 4.77fps with Quadro FX4500 (512MB) and GeForce7300GS (512MB) , respectively. Using the latest high-end GPUs, we surely have faster 3D visualization than 30fps. Namely, we can expect ten times volume rendering for (512,512,512) volume data requires less than 0.3 seconds.

The second longest time consuming task is the creation of ten image files (0.21s) and one FLV file (1.133s). In this experiment, we use several existing

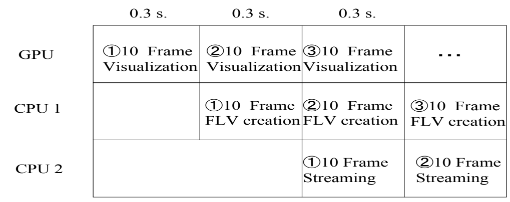

Figure 4: The pipeline chart for visualization process, FLV creation and live streaming.

applications for this task, so the data transfer is done via files. Re-implementing this task within memory, we estimate the whole process will be reduced to one over ten (0.13s).

The rest of time consuming task is communication overhead of 1.025 seconds. This delay can be surely improved because the synchronization operations are performed by a Perl script. Replacing this part with MPI synchronization functions, this overhead will be almost eliminated.

4.5 Discussions

The preliminary experiments indicate the following analyses.

1) Using the latest GPUs, ten times volume rendering for a (512,512,512) volume data set is estimated at 0.3 seconds.
2) Creation of ten frames FLV movie requires less than 0.3 seconds.
3) Assuming 500Kbps bit rate, the communication delay of ten frames FLV movie is also less than 0.3 seconds.
4) On the client web browsers, the detecting time for mouse events is 0.1 seconds by ActionScript.

If we use a visualization server with dual core CPUs as well as a high-end GPU, the delay between a client web browser and the visualization server is 0.3 seconds as shown in figure 4. In the case of mouse events, the delay is estimated at 0.4 seconds. Thus we conclude that we can develop an interactive 3D visualization system by live streaming for remote scientific visualization within the delay of 0.4 seconds, which is enough fast response time for geophysical fluid research.

5 CONCLUSIONS

In this paper, we proposed an interactive 3D visualization model using live streaming so that the 3D visualization is applied to scientific data located in other computers connected by the Internet. The model is to be implemented as 3D visualization functions for Gfdnavi, which is a tool to produce web-based services for geophysical fluid sciences. For this purpose, we described six design concepts for the implementation. Based on the design concepts, we explained the implementation methods using GPGPU and live streaming. To validate our implementation method, we developed a prototype system to perform preliminary experiments. The experimental results indicate that our interactive remote 3D visualization system by live streaming would have the delay of 0.3 seconds, provided with a high-end GPU and a dual core PC, while only 500Kbps bit rate streaming is enough for the remote 3D visualization. Our future work includes the actual implementation of our proposed system.

ACKNOWLEDGEMENTS

This work was partly supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, Grant-in-Aid for Scientific Research on Priority Areas, g i-explosionh(No 18049043).

REFERENCES

B. von Rymon-Lipinski, T. Jansen, N. H. and Keeve, E. (2005). Interactive visualization of large point isosurfaces using gpu-based decompression. In *Proceedings of the IEEE/Eurographics Symposium on Point-Based Graphics PBGf05*.

E. Touma, K. Noguchi, M. T. H. K. N. N. and Joe, K. (2007). 3d visualization system gateau for atmospheric science: Design concept and practical evaluation. In *The 2007 International Conference on Parallel and Distributed ProcessingTechniques and Applications*, volume II, pages 766–772. (more detail:E. Touma, M. Takata, K. Joe:A 3D Visualization Model for Information-Explosion in the Atomospheric Science Field,Information Processing Society of Japan: Mathematical modeling and Problem Solving (to appear),2007,(in Japanese)).

F. Schulze, K. B. and Hadwiger, M. (2007). Interactive deformation and visualization of large volume datasets. In *2nd International Conference on Computer Graphics Theory and Applications (GRAPP)*, pages 39–46.

T. Horinouchi, S. Nishizawa, C. W. Y. M. T. K. M. I. Y. H. and Shiotani, M. (2007). Development of gfdnavi: a new desktop/server tool for geophysical fluid database, analysis, and visualization. In *Proceedings of Data Engineering Workshop*. Japanese.

T. Horinouchi, R. Mizuta, D. T. S. N. and Takehiro, S. Gphys – a multi-purpose class to handle gridded physical quantities. URL: <http://ruby.gfd-dennou.org/products/gphys/>.