Interactive Solution For Distributed Collaborative Visualization

Asma Al-Saidi
The Communication and Information Research Center (CIRC)
Sutan Qaboos University
P.O Box 17,
P.C 123, Muscat
Sultanate of Oman

Abstract

Interactive distributed visualization is an emerging technology with numerous applications. However, many of the present approaches to interactive distributed visualization have limited performance since they are based on the traditional polygonal processing graphics pipeline. In contrast, image-based rendering uses multiple images of the scene instead of a 3D geometrical representation, and so has the key advantage that the final output is independent of the scene complexity, and depends on the desired final image resolution. These multiple images are referred to as the light field dataset. In this paper we propose an on-demand solution for efficiently transmitting visualization data to remote users/clients. This is achieved through sending selected parts of the dataset based on the current client viewpoint, and is done instead of downloading a complete replica of the light field dataset to each client, or remotely sending a single rendered view back from a central server to the user each time the user updates their viewing parameters. The on-demand approach shows stable performance as the number of clients increases because the load on the server and the network traffic are reduced. Furthermore, detailed performance studies show that the proposed on-demand scheme outperforms the current local and remote solutions in terms of interactivity measured in frames per second. Finally, we conclude that the design of our 3D visualization system, based on image-based rendering coupled with an on-demand transmission model, has contributed to the field, and is a good basis for the future development of collaborative, distributed visualization systems.

.

Keywords: *Distributed Visualization; Distributed Systems; Client/Server Distributed Applications; Image-Based Rendering; Remote Visualization.*

.

1. Introduction

While scientific visualization plays a key role in the exploration, phase of large complex datasets and facilitates the understanding and analysis of such datasets [1], providing an effective visualization solution is challenging to achieve due to the fact that the increasing complexity of datasets often exceeds the rendering capabilities of local processors. Such challenges had previously limited visualization to only local access on powerful machines, or remote access to powerful machines via dedicated high throughput networks. Furthermore, with recent advances in e-science applications, an increasing demand has arisen among researchers for a means to share data and perform analysis remotely. Consequently, visualization systems have started to evolve from standalone systems to distributed systems, both to exploit remote resources and to serve geographically remote users, thereby allowing them to collaborate and share visualizations

In the past, the visualization of large datasets was limited to local users with powerful processing hardware, or remote users with dedicated networks. With the recent development of high-performance local area networks, a variety of distributed applications has emerged. However, current distributed visualization systems have either not provided a generic solution or have experienced a performance bottleneck in terms of the size of the dataset, and/or the number of concurrent users.

To construct an interactive remote visualization system two key challenges must be met and overcome. First, to ensure smooth navigation the distributed visualization system must be capable of delivering between 10-15 frames/second to each user connected to the system [2]. Such a high interactive frame rate is generally difficult to achieve for systems with low to mid-range rendering hardware. The second issue is the limitation of the network resources, such as low bandwidth and high latency.

.

2. Literature Survey

Visualization systems could be classified into two categories:

1. Stand-alone visualization systems, sometimes refereed to single-user environments allow users to visualize and analyze a dataset in a single machine. Such system normally requires an expensive graphics workstation that may not available to many organization or users. Examples of such systems are VisIt [3], ParaView [4] and VTK[5].

2. Remote visualization means interactive viewing of three-dimensional scientific data sets over the network. Because scientific data sets are in the gigabyte size range (or larger), it is difficult to send the entire data set over the network. Extraction, processing, network latency and rendering times add up and make the proposition of near real-time interactive visualization a challenge. Moreover, the client may have a limited amount of memory and CPU power for viewing and interacting with the data.
   Furthermore general-purpose visualization systems such as RAVE [6] (The Resource-Aware Visualization Environment) is a distributed, Grid-enabled collaborative visualization environment that supports automated resource discovery across heterogeneous machines. Rather than commandeering an entire machine, RAVE runs as a background process using Web Services, thus enabling resource usage to be optimised and shared between users. RAVE supports a wide range of machines, from hand-held PDAs to high-end servers with large-scale stereo, tracked displays. The local display device may render all, some or none of the data set remotely, depending on its capability and present loading. This enables individuals to collaborate from their desks, in the field, or in front of specialised immersive displays. However, RAVE has problems with scaling in terms of the number of users and the size of the dataset. Despite the fact that advances in computer graphics rendering techniques, such as ray tracing [7] are able to generate highly realistic images, they are still too slow to be used for 3D real-time applications using local desktop systems, and limited network bandwidth that cannot support the transfer of large datasets where interactive frame rates are essential. A parallel processing version of a ray tracer using a large number of processors, such as a 60 CPU Silicon Graphics Origin 2000 [8], can provide a real-time solution for large computerized tomography scan datasets [9]. Another example is the Visapult system which utilizes specialized hardware [10], and performs a high speed parallel rendering process for a massive dataset (1-5Tb).

Limitations to Visualization
As in all areas of science and research there are currently limitations on our ability to visualize data. Computers give rise to two main limitations:
1. The hardware may be unable to process data fast enough.
2. The software may be unable to provide useful and efficient algorithms.
More complex modeling and simulation leads to higher quality images, however, they also lead to longer rendering times. Even with today’s most advanced computer running today’s most powerful rendering algorithms, it is still fairly easy to distinguish between a synthetic photograph of a scene and an actual photograph of that same scene. In recent years, a new approach to computer graphics has been developed: Image-Based Rendering (IBR). Instead of simulating a scene using some approximate physical model, novel images are created though the process of reconstruction. Starting with a database of source images, an image is constructed by querying the database for information about the scene. IBR hasThe potential of providing a more realistic
representation for very complex scenes at much faster rates than classical geometrical rendering. Units. Visualization system used IBR has shown performance independent of how much complex the dataset [11]. However, sending each viewpoint cause the overloading the network. In this paper we introduce system based on Image-based rending with mechanism of sending partial dataset instead of single image.

3. System Architecture

3.1 System Overview

We propose a generic distributed visualization system that presents a solution which is independent of 3D model complexity and relies only on the final output image resolution (see Fig 1). Our system provides a distributed collaborative environment, where multiple concurrent users visualize a remotely located 4D dataset. The developed system applies a multi-user client-server model and takes advantage of the availability of low-cost network equipment to provide an inexpensive visualization solution. There are three core phases that comprise our system architecture.
1 In the data acquisition phase the 4D light field dataset is generated for a targeted object. The 4D light field dataset is generated in a pre-processing step using either a multi-array camera [12] or gantry for real objects [13]or it is created for a synthetic object using a modified ray tracing algorithm

2 The second phase involves the rendering process. During this phase, the nearest images to the requested view are interpolated to produce the desired view. In order to be able to view a 3D model on a 2D display the rendering process must create a 2D output image based on the description of the viewer's position in 3D space. The light field is created by sampling from a 2D array of camera positions, represented by (u,v), and a pixel position (s,t) within the selected image. The rendering in this case simply combines and resamples the closest available images. Interpolation algorithms are applied to create the best estimate of the output image [14]. The simplest interpolation method is the nearest neighbor algorithm where a pixel in the output image is computed as the value of the nearest mapped pixel in the source image. Since there is little calculation involved in this interpolation method, it is the fastest. The next resampling method is bilinear interpolation where the destination pixel is computed by combining the linear interpolation along two orthogonal axes. Although this interpolation method produces smoother results than the nearest neighbor method, it tends to require more computation as it involves more data points from the surrounding neighborhood. In quadrilinear interpolation, destination pixels are generated by combining linear interpolation with respect to all four axes. This approach gives high fidelity results compared with the other approaches, but its higher computation time makes it an appropriate choice only for highly interactive systems running on high-end device. In our implementation we make use of quadrilinear interpolation as it gives a good tradeoff between the quality of the image and the computation time.

3 The final phase is when 3D viewing takes place. In the viewing scenario visualization participants (clients) run an instance of the viewing process in which the independent 3D viewing positions are created locally and processed remotely in the rendering server. Each visualization client or viewer is working independently. The concurrent interaction between the server process and the different viewer processes. As the user runs the viewer process it will connect to the server process. After the user selects a new viewing parameter set, a new request is sent to the server where the rendering request is processed. The client will then receive the updated framebuffer and a new view will be reloaded and displayed. The server executes the viewer’s viewing requests based on their arrival time. Clients send their viewing requests separately to the rendering server. The rendering server processes the requested views concurrently and sends the resulting framebuffers to the clients, as shown in Fig 1.

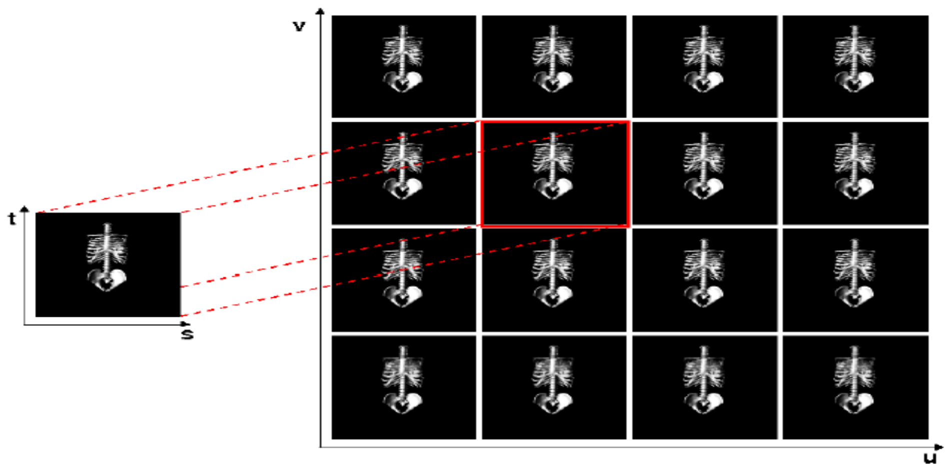

Figure1: 4D Light Field representation in terms of an array of images taken at different positions in the (u, v) plane.

3.2 The System

The developed system provides a distributed collaborative environment in which multiple concurrent users visualize a remotely stored 4D dataset. A multi-user client-server model is used, taking advantage of the availability of commodity off-the-shelf (COTS) computer hardware, software and network equipment. The main reason that our system is implemented only in software is that we targeted a generic system without the need for any special hardware. Furthermore, we aim to support for broad range of client-side platforms, ranging from a high performance desktop to a PDA for field scientists. There are two main components that comprise our system architecture, as shown in Fig. 2.

a) Client Module

The basic configuration of the client side of the system has the following components:

- The Client Manager Process is responsible for dealing with external network connections and internal process communication within the client. Its role can be summarized as follow:
  - Send user viewing parameters to the Rendering Process.
  - Check if the client cache has the required images

- Establish a connection port to the server.
- Send request for required image data image.
- Received required image data.
- Place a copy of the image data in the cache.
- Send the images to the Rendering Process.
- Send the resulting framebuffer to be displayed by the Viewer Process.
- Close the connection port after the user closes the Viewer Window.

- The Rendering Process locates the images nearest to the requested view and performs interpolation to produce the desired view.
- The Viewer Process is where 3D viewing takes place. As the user interacts with the Viewer Process, a corresponding set of viewing parameters is sent to the Client Manager Process for rendering. The resulting image is received from the Rendering Process, mapped onto the projection plane, and displayed using the QImage class of the Qt library [15].
- The Client Cache is filled with images based on recent viewing parameters within the 3D environment. This can be used to build a resource-aware layer to reduce the load from the server in order to increase scalability in terms of the number of participating clients. A further enhancement is to apply a prediction algorithm to pre-load images by predicting the navigation path of a user.

b) Server Module

The server has three main components:
_ The Server Manager Process handles new client connections using threads. When a new view request is sent to the server, the request is parsed, the viewing parameters are extracted, and the cache is checked. The Server executes the viewer's requests based on their arrival time (see Fig. 2). The client-server interaction model for our system represents collaboration in space and time, since participants could be located in different places at different times. Each individual viewer can join or leave the visualization at any time. Each visualization client or viewer independently controls their viewpoint requests. Each time a new request is received it checks the availability the related images in the cache and if not present fetches them from the dataset and copies them to the cache for possible future requests.

The role of the Server Manager Process can be summarized as follow:

- Accept a new connection with the joining visualization clients.
- Receive clients' data requests.
- Check Server cache for requested images, and if are found send them to the client.
- Upload any required images not found in the cache from the light field dataset and send them to the client.
- Update the Server Cache with the recently requested data images.

- The Light Field Dataset stores the 4D light field images.

c) The Server Cache stores in memory the images requested recently by the visualization clients. In collaborative visualization environments the users normally explore a section of the 3D scene in which they have a common interest, generating the same viewpoint for all clients.

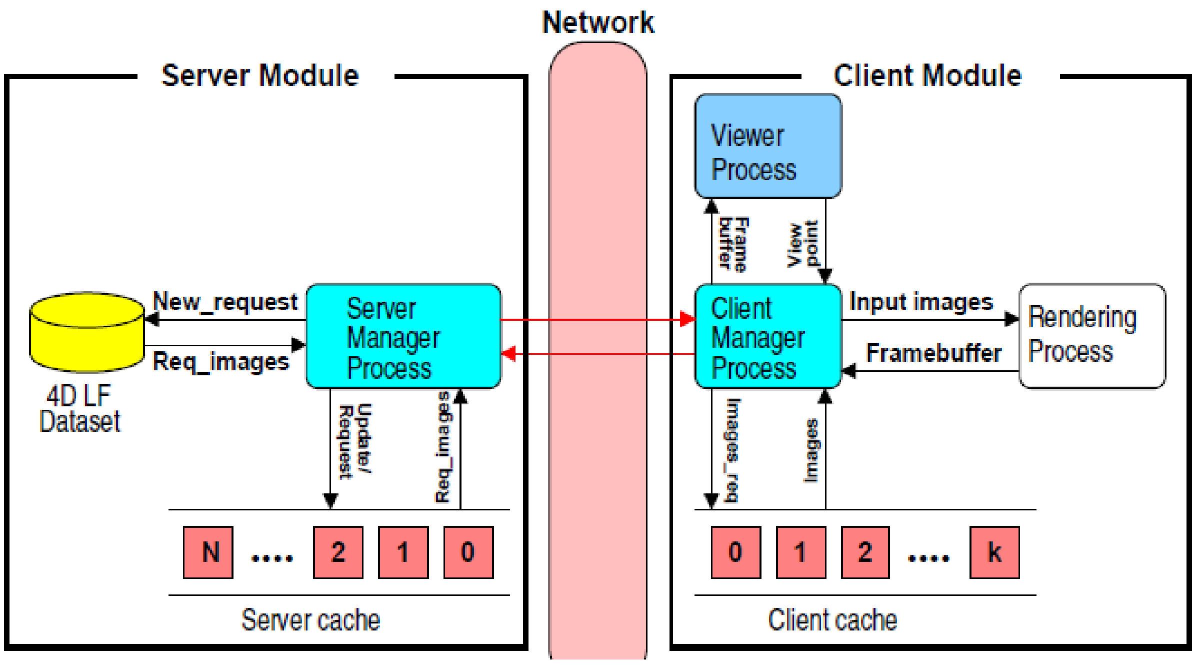

Figure 2: The architecture of the transmission model

4. Conclusions

We have described the software architecture for using a transmission model using light field rendering within a distributed collaborative environment. The system makes use of commodity networking, hardware, and software for distributed visualization. We have discussed sending partial dataset. Our results shows that a light field rendering system enables constant framerates independent of the scene complexity compared to geometry-based visualization, where the frame-rate depends on the complexity of the scene for the chosen viewpoint. The efficiency of the system increases as the scene complexity increases.
The applicability of the system covers a large number of application areas such as telemedicine, geologists and engineers wishing to view a large, complex model (such as an oil drilling platform), or any user of visualization wishing to view complex datasets that would otherwise overwhelm a graphics processor where interactive collaboration could enhance and accelerate navigation and discovery.

5. References

1. L. McMillan and G. Bishop, Plenoptic modeling: an image-based rendering system, SIGGRAPH ’95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (New York, NY, USA).
2. Micah Beck, Terry Moore, and James S. Plank, An end-to-end approach to globally scalable network storage, Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (New York, NY, USA), SIGCOMM ’02, ACM, 2002, pp. 339–346.
3. VisIt, https://wci.llnl.gov/codes/visit/
4. VTK: the Visualization Toolkit, http://www.vtk.org/,
5. ParaView, http://www.paraview.org/
6. I. J. Grimstead, N. J. Avis, and D. W. Walker, Automatic distribution of rendering workloads in a grid enabled collaborative visualization environment, SC ’04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing (Washington, DC, USA), IEEE Computer Society, 2004, p. 1.
7. A. Glassner, An introduction to ray tracing, First edition, Morgan Kaufmann,1989.

8. S. Parker,W. Martin, P.-P. J.Sloan, P. Shirley, B. Smits, and C Hansen, Interactive ray tracing, In Symposium on interactive 3D graphics, 1999, pp. 119–126.
9. S. Parker, M. Parker, Y. Livnat, P. P. Sloan, C. Hansen, and P. Shirley, Interactive ray tracing for volume visualization, IEEE Transactions on Visualization and Computer Graphics, vol. 5 (1999), no. 3, pp. 238–250, 1077-2626.
10. E.W. Bethel, Visualization dot com, IEEE Comput. Graph. Appl. vol. 20 (2000), no. 3, 17–20.
11. Asma Al-Saidi, Nick J. Avis, Ian J. Grimstead, and Omer F. Rana, Distributed collaborative visualization using light Field rendering, CCGRID, 2009, pp. 609–614.
12. The Stanford multi-camera array, http://graphics.stanford.edu/projects/array/,
13. The Stanford spherical gantry, http://window.stanford.edu/projects/gantry
14. R. E. Crochiere and L. R. Rabiner, Multirate digital signal processing, Prentice-Hall, 1983
15. Qt Library, http://qt.nokia.com.