REMOTE VISUALIZATION USING 2D ARRAY OF IMAGES

Asma Al-Saidi
aalsaidi@squ.edu.om
Oil And Gas Research Center
Sultan 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 a 3D visualization system, based on image-based rendering for efficiently transmitting visualization data to remote users/clients. This is achieved through applying Light field (LF) rendering and the resulted image is remotely sent from a central server to the user each time the user updates their viewing parameters. Furthermore, performance study shows that the proposed scheme outperforms the current local solution in terms of interactivity measured in frames per second.

KEYWORDS

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

1. LITERATURE SURVEY

Stand-alone visualization systems, sometimes refereed to single-user environments allow users to visualize and analyze a dataset in a single machine. Such systems normally requires an expensive graphics workstation that may not available to many organization or users. Examples of such systems are VisIt [1], ParaView[2] and VTK[3].

While 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.

Remote visualization can be classified into categories: specialized systems and general purpose visualization systems.

1.1 Specialized Systems

Despite the fact that advances in computer graphics rendering techniques, such as ray tracing [4] 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 [5], can provide a real-time solution for large computerized tomography scan datasets [6].
Another example is the Visapult system which utilizes specialized hardware [7], and performs a high speed parallel rendering process for a massive dataset (1-5Tb).

1.2 General Purpose Systems

The general purpose solutions should enable visualization sharing and enhance collaboration, which would allow non-constrained visualization solutions, instead of the existing visualizations solutions which are specific to groups of people and domains.
An example of such systems is RAVE [8] (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

1.3 Image-Based Rendering (IBR)

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 has the potential of providing a more realistic representation for very complex scenes at much faster rates than classical geometrical rendering. Units
The most well-known IBR representation is Light Field (LF) rendering [9] Research using similar approaches has also been carried out independently at Microsoft, and named Lumigraph [10]. LF rendering is flexible in the nature of images. The rendering process involves the combining and resampling of the closest available images for the given viewpoint, and interpolation algorithms are applied to create the best estimate of the output image.

2. APPLICATION CONTEXT

In general, the visualization system could be used for any static or time dependent field. To exemplify the category of application we have selected the medical domain, where trainee physicians and surgeons need to develop a deep and detailed understanding of the structure and variation of human anatomy. Traditionally this has been acquired through hands on experience with a cadaver. In recent years a number of factors including strengthening health and safety requirements, diversification of curricula, geographical distribution of learners (from undergraduate through to senior practitioners), and a need to work with multiple examples to experience variations in anatomy, have made this increasingly impractical and ineffective. In the future we anticipate extending the usage of the system to encompass diagnosis, medical training, and surgery planning using real-time visualization in a distributed environment. Our system can be applied also to non-medical domains and could cover various field such as 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.

3. 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 preprocessing step using either a multi-array camera [11] or gantry for real objects [12] 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 [13]. 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. Figure 1: System Overview. In the Data Acquisition phase a 4D dataset is created. The Rendering phase performs interpolation across multiple images. In the Viewing phase viewpoint queries are sent and the received output frame buffers are loaded for display.
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.

4 EXPERIMENTAL RESULTS AND DISCUSSIONS

The testbed environment uses commodity hardware consisting of 32 standard PCs running Linux Fedora 7. Each node has a 2.8 GHz processor, 2 GB of RAM, and an nVidia Corporation G70 graphics card (GeForce 7800 GS) with a clock speed of 375 MHz and 256 MB of RAM. The server has 2 GB of RAM, and is a 64-bit Intel Pentium Dual CPU machine with 3.4 GHz processors, and an ATI Radeon X1300 graphics card with a clock speed of 450 MHz and 256 MB of RAM. Communication is over a 100 Mb/s local Ethernet network. The datasets used represent different views of part of a human skeleton (originally obtained from http://www.3dcafe.com/). To determine the performance of the system we used four different light field dataset dimensions. These datasets all have a (u, v) array of 16 x 16 camera positions so there are 256 images for each dataset, and for each position we have image sizes of 128 x 128, 256 x 256, 512 x 512, and 640 x 640 pixels. These correspond to image sizes in kilobytes (KB) of 16KB, 64KB, 256KB, and 400KB. These image sizes reflect the range of image sizes commonly supported by current cameras and scanners. The navigation path (a sequence of cursor positions) is recorded from human interaction with the visualized object to emulate a real-life navigation experience, and we make sure that these parameters are used in all experiments.

4.1 Effect Of The Number Of Concurrent Visualization Clients

Initially each client starts a connection with the server. On the server side the server forks a new thread for each of the individual client connections so that all the connection requests can be handled concurrently. We have examined the effect of the number of visualization clients for local rendering, remote rendering (see Fig 2). The remote rendering achieved 38-45 frames per second while the local rendering has only abled to achieve 1-2 frames per seconds.

![Figure 2. Time for different numbers of viewers for 256x256 dataset resolution.](image)

4.2 Effect Of Different Datasets Sizes

The overall impact of the communication time (see Fig. 3) varies according to the dataset size and the number of visualization clients. At small resolutions (128 x 128 and 256 x 256) the remote rendering case shows a higher interactivity rate due to the small time interval taken by the rendering process (averages of 2.9 ms and 14.2 ms). For a large dataset (512 x 512) remote rendering is able to produce (10-17) frames per second for different viewers no. With 640x640 the achieved frame rate is (2-10). With these results users with larger viewers have to adjust their output to be 512x512 and for small number of viewer (4 or less) the users can view their output in 640x640 format.
5 CONCLUSIONS

We have described the software architecture for using an on-demand 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 the existing local and remote transmission models. Our results show that a light field rendering system enables constant frame rates 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. Also the system enabled 32 concurrent users to navigate the dataset in interactive frame rate.

The applicability of the system covers a large number of application areas, where interactive collaboration could enhance and accelerate navigation and discovery.

6 REFERENCES

1. VisIt, https://wci.llnl.gov/codes/visit/