Distributed Collaborative Visualization on Mobile Devices Using Interactive Video Streaming Techniques

Maciej Panka[1], Michal Chlebiej[2], Krzysztof Benedyczak[2,3], and Piotr Bała[2,3]

[1] UCNTN Nicolaus Copernicus University, Torun, Poland
maciej.panka@umk.pl
[2] Faculty of Mathematics and Computer Science Nicolaus Copernicus University, Torun, Poland
[3] ICM, University of Warsaw, Warsaw, Poland

Abstract. Remote visualization using mobile devices has been a challenge for distributed systems for a long time. Large datasets, usually distributed on different servers require high network bandwidth and significant computational power for effective, real time rendering. The problem is getting more complex when data are visualized in collaborative environment, where every user can interactively participate in rendering session.

In this paper we present a distributed system we have developed for the interactive visualization of remote datasets on variety of mobile devices such as laptops, tablets and cell phones. In our system mobile users can join sessions, where they can collaborate over remote data in real time. Every user can watch presentation or can become presenter. If needed, users can individually manipulate the data without affecting rest of participants.

During these sessions all the data are generated on dedicated rendering servers, compressed on-the-fly by the encoding machines using video codec and progressively sent to participants as video streams. Every video stream is dynamically adapted to individual capabilities of users' devices and their network bandwidth. Our system works in a distributed environment, where every machine serve different functionality, like data storage, frames rendering or video compression. Successive parts of processed data are streamed between different servers in real time to achieve highly interactive visualization with minor latency. Based on this model we took off most of the computational power from client's application so it can be run on almost any kind of modern mobile device. We were also able to achieve very high video quality and frame rates. System can work with 2D, 3D and even animated 3D data, all of them being processed remotely in real time. At the end of this paper we present some preliminary results of performance test we have obtained using sample multidimensional datasets.

Keywords: distributed data visualization, mobile visualization, mobile collaboration, distributed video streaming.

R. Wyrzykowski et al. (Eds.): PPAM 2011, Part II, LNCS 7204, pp. 191–200, 2012.

1 Introduction

Pervasive computer technologies have redefined group collaboration long time ago. Modern collaboration techniques rely mostly on the Internet and mobile communication systems, which provide almost unlimited access to the global information resources. Presently available solutions for computer-based cooperation could be divided into four categories depending on a place-time relation of participating users, as presented in [1] (same time - same place, same time different place, different time same place, different time different place). The collaboration, which involves simultaneous presence of all users is usually called synchronous communication. Otherwise it is called asynchronous communication. Most popular examples of today's asynchronous systems are email and discussion boards. On the other hand, exemplary synchronous techniques would be instant messaging, chat, shared boards, teleconferencing or videoconferencing.

Modern collaboration techniques could also be effectively adopted into the scientific activities. Progressive globalization causes, that many scientific projects involve cooperation of researchers from different departments, universities or even countries. With the use of the Internet and computer technologies the collaboration is much easier today then it used to be in the past. However, in some situations it is still a challenge to virtually bridge the distance between participants, especially when the collaboration is centered on the large datasets, distributed among storage centers around the globe. Many modern scientific experiments and simulations are so complex, that they must be realize on dedicated computer clusters. Obtained data volumes are usually so large that they cannot be easily transferred between distant users, and must be stored on the dedicated servers.

One of the biggest challenges in this area is an effective visualization of these data through the Internet, without the need of downloading and rendering it on local computers. The problem is even bigger when the visualization should be realized on mobile devices, which still don't have enough computational power to effectively render complex data in real time. In this paper we present a framework, which confronts this problem allowing mobile visualization of remote datasets in a distributed collaborative environment. Our framework consists of a distributed server side application, as well as the client's application, which could be run on thin, mobile devices. According to the users' demands, server application reads the data from adequate storage area, renders it in real time, compresses using a video codec and broadcast to client using a dedicated streaming protocol. Remote users receive these streams, decode them and display on the screen. Moreover, distant users can collaborate over the remote data in real time, with the use of a built-in teleconferencing functionality and a shared, interactive video area. If needed, they can also manipulate remote data without affecting rest of the session participants.

The rest of this paper is organized as follows. Section 2 briefly describes related works covering mobile visualization in a distributed collaborative environment.

In the sections 3 and 4 we describe in details main functionalities of the system, its architecture and technologies that we have used to implement it. Section 5 presents system's performance test results obtained during a sample collaborative session. Section 6 concludes this paper and draws up further work.

2 Related Work

There are few different approaches to the remote visualization problem. First category involves systems, which force data rendering on clients' devices. Many solutions which are based on this model compress all the data on the server and transfer it progressively to distant users using different kind of 3D objects streaming techniques, like progressive meshes or levels of details [2, 3, 25–27]. Other commonly used approaches are based on the Virtual Reality Modeling Language, which is a text file standard for 3D scenes representation [4, 5]. Although, one of the biggest challenges when using VRML standard is the size of generated files, which causes some serious limitations in a data network transfer. References [6] and [7] propose different compression techniques, which reduce this latency.

Second category involves systems, where all the data is rendered on dedicated server and transferred to client as a digital image sequence. References [8] and [9] introduce exemplary solutions based on the image streaming techniques, which make use of the lossless compression algorithms, like LZO or BZIP. The authors of [10] and [11] presented possibilities of using JPEG 2000 compression, which increases the overall image compression level. They also showed, that with the use of the Motion JPEG 2000 it is possible to visualize animated 3D data, although there are more sophisticated techniques available in this area, like for example VNC [12] or by means of a dedicated video codec [13, 14].

The idea of collaborative visualization of remote data has been described in details in [1]. Reference [15] proposes exemplary solution to this problem by means of establishment of dedicated rooms, equipped with the specialized videoconferencing hardware and software. However, a much ubiquitous solution would be to use participants' individual computers during the collaborative sessions, including modern mobile devices. Exemplary systems based on the VRML text files were introduced in [6], [16], [17] and [18]. The authors of [19] and [20] introduced sample frameworks for a collaborative visualization, which render all the data on a server and send it to every connected user as a sequence of digital images. The references [21], [22] and [23] additionally made use of dedicated video codecs.

In this paper we propose a distributed visualization system, which derives directly from the video streaming techniques, allowing thereby an effective and fully interactive data visualization on different kind of mobile devices, including tablets and cell phones.

3 System Overview

3.1 Remote Collaboration

The system consists of the client and server applications. Server modules are responsible for session management, clients' collaboration, data rendering and video encoding. On the other hand, client's application's main job is to receive a video stream, decode it and display on the screen. During collaborative sessions clients can additionally communicate using a built-in teleconferencing module.

The first user, which connects to the server, becomes a session presenter. Successive users can either start their own sessions or they can join previously established one. The presenter of each session can load the remote data and manipulate it in real time using his mouse or touch gestures. As the response to user's requests server's application loads the data from the source, renders it, compresses using a video codec and broadcast to every user participating in the session. By default all the other session participants can only passively display the video stream on their screens, listening to the presenter's lecture at the same time. At any point during the session, the presenter can pick up any connected participant, giving him temporary ability to manipulate the data and to speak to other users. From this moment both the presenter and selected user can discuss among other participants using teleconferencing module. Additionally, during the whole session users can individually manipulate the remote data, having a chance to analyze it from different angles, without affecting other participants.

3.2 Interactive Visualization

Our system can work with a 2D, 3D and animated 3D data, all of them being processed on the server and broadcasted to the users in real time. When the presenter changes his view angle, the remote data is dynamically processed by the adequate servers, which generate successive frames and compress them using a video codec. Video frames are streamed to all participants in real time, giving them effect of a zoom, move and spin animations [Fig. 1].

The output of a two-dimensional data is a typical digital image. Depending on the data source, these images could have very large resolutions, transcending typical screen sizes of mobile devices. Our system dynamically crops and resizes every image, fitting it to the individual capabilities of users' devices. The session presenter can zoom in / out and move to the different parts of an image by selecting its successive regions on the screen of his device. According to the user's actions these events are transferred to the server application, which crops adequate parts of an input image and places them on an encoding sequence. Successive frames are compressed and streamed to the users one by one, giving the effect of an image motion. Every session participant receives his own video stream, individually customized to the capabilities of his device, including a screen size and network parameters. That means, that depends on the number of concurrent connections every server could encode in parallel many different video streams during a single visualization session. The visualization of a 3D data additionally involves rotation

of a remote object over the X and Y axes. As a response to the presenter's rotation events, adequate server application generates successive frames in real time, which are later encoded and progressively broadcasted to users. During the visualization of an animated 3D data all the above techniques work in a loop, giving thereby an effect of data evolution in time.

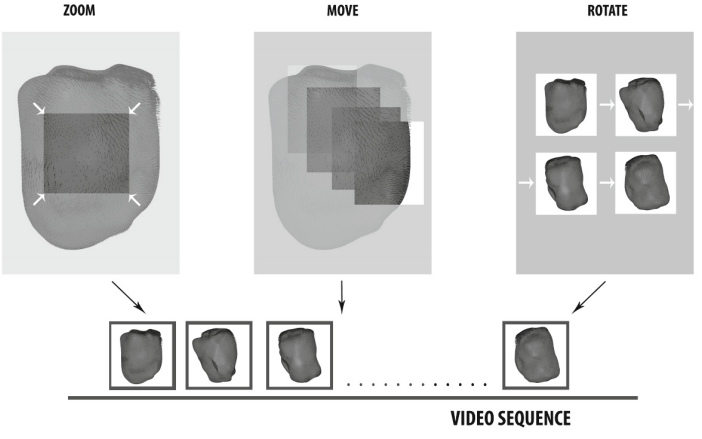

Fig. 1. Successive frames of processed data are encoded as a sequence, and broadcasted to all participants as an independent video streams, individually customized to their devices

4 System Architecture

The server side application consists of three modules: a session manager, data renderer and video encoder. To achieve the highest possible performance level, each of them should be run on a different machine, so they could process their tasks in parallel. For a better load balancing during multiuser sessions, the rendering and encoding modules, which are the most CPU consuming parts of the system, should be additionally distributed between servers [Fig. 2].

Every joining user starts his session by connecting to the manager module. The session manager stores the information about other servers available in the system, together with their current load, including avarage CPU / RAM usages and a number of concurrent users. Every server measures these parameters and sends them to the session manager periodically. Based on this information the session manager is able to find the less laden servers at the moment (encoding and rendering modules), and sends their IP addresses to the newly connected presenter. With the use of these parameters the presenter is able establish the connection with the selected encoder, which from this moment controls the rest of the visualization computes. The further participants who join this session establish their connections only with the session manager module, which also acts as a video proxy between the encoding module and clients' applications.

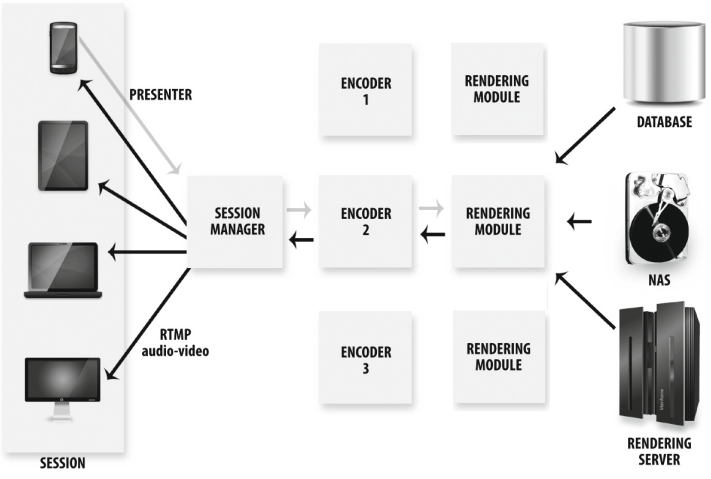

Fig. 2. A general schema of the system's communication workflow. To achieve highly interactive visualization, the session manager, encoder and rendering modules exchange partial data progressively, and transfer them to distant users as a live video stream.

By default, every new user is able only to watch the presentation passively, so there is no need for them to communicate with any of the encoders. The only situation when other users connect to the encoder is when they become temporary presenters or when they want to individually manipulate the remote data. The selection of the less laden server in that case is realized the same way as described earlier.

In a response to the presenter's activities (zooming, moving and rotating) client's application sends adequate events to the selected session encoder. Based on these events the encoder creates appropriate frame sequence and progressively sends it to session participants. During the two-dimensional data visualization, all the images are buffered directly on the encoder machine, as there is no need for permanent data rendering. However, working with the 3D and animated 3D data involves parallel image generation in the rendering module, which broadcasts them to the encoder one by one in real time (rotation and animation frames). The rendering module is completely transparent to the data source, and it could be easily adapted to cooperate with a database, storage drive or a dedicated rendering machine.

Regardless from the dimension of the data source, the encoder processes every frame on-the-fly, based on the current parameters of the presenter's view (crops and resizes), compresses them using a video codec and broadcast to the session manager, which republishes them to the rest of the session participants. To serve many users simultaneously the encoding module runs a multithread application, which in parallel compresses and broadcasts different video streams to session participants (different resolutions, bit rates and qualities of the outgoing video). We are not sending the videos directly to the session users, because practically

in most cases one stream would have many simultaneous recipients, which have devices with the same screen sizes and similar network bandwidth. This approach allowed us to limit number of the encoding threads being run in parallel, reducing thereby the encoding server's load.

Session users can communicate using a teleconference module, which has been built into the manager server's application. The audio data, which is transmitted from the presenter's device, behaves exactly the same way as other videos streamed to the session manager from the encoder module. Client's application acquires the data from user's microphone, compresses it and broadcasts to the session manager, which re-streams it to the rest of the session participants.

All the server side modules have been written in Java. The session manager's application has been deployed on the Wowza Media Server, which is currently one of the most powerful video streaming solution. The WMS assures communication with the sessions' participants and provides the video re-streaming solution to a variety of popular protocols, including RTMP, RTSP or HTTP.

Rendering module streams successive frames to the encoder using a TCP socket connection. During a typical visualization session both machines exchange large amount of data, so it is recommended that they communicate using a broadband connection. We have decided not to use any compression techniques at this point because they would require additional computational power from the rendering module. Encoding module compresses the video sequence and streams it to the session manager with the use of FFMPEG, which is currently one of the best open source solutions to handle digital video. Encoded video sequence is streamed to the session manager, and later to all users by means of the Adobe's Real Time Messaging Protocol.

5 Results

We have run a series of performance tests of the system. At first we have measured the video compression speed and CPU usage of the encoder. The session management module was run on an Intel Xeon 5050 3GHz, the rendering module on a dual core Intel Xeon X5355 2.6 GHz, and the encoding module on a quad core Intel Xeon E5420 2.5 GHz. All the servers were connected using 1 Gb network.

We have run four tests using different types and number of clients' devices, each of them having set different screen resolutions. That way we were able to measure the system's performance during simultaneous encoding of multiple video streams. At first we have tested its behavior using three different mobile devices: a laptop running Mac OS X, tablet and a cell phone, both equipped with the Android 2.2 system. Three other tests were run on a Windows Vista desktop computers, which involved 5 and 10 simultaneous users, each of them having different screen sizes. Additionally, 5 users test was run two times. The first test was run for a group of devices equipped with the small screens, and the second effort, which involved only a high resolution video encoding. Table 1 presents obtained data.

Table 1. Encoding speed averages measured in frames per seconds during four different collaborative sessions

Resolution	3 users	5 small screens	5 big screens	10 users
320 x 240	-	228	-	119
400 x 240	-	228	-	116
640 x 480	-	86	-	40
800 x 480	78	84	-	40
854 x 480	-	52	-	37
768 x 576	-	-	63	31
800 x 600	-	-	37	30
1024 x 600	57	-	36	26
1024 x 768	41	-	27	20
1366 x 768	-	-	26	19

The results show, that a single encoding server is able to effectively compress many simultaneous video streams, all set at different resolutions. From the users point of view the lowest acceptable encoding speed is 15 frames per second. Below that value the visualization looses its smooth and decreases the overall reception of the session. In our experiment all results were above the minimum fps, regardless from the targeted users device (desktop, tablet, cell phone). The CPU usages of the encoding server were 25%, 30%, 50% and 75% respectively, never reaching the maximum capability of the system.

During the experiment we have also measured the network communication latency between the clients and a session manager modules. We have also tested the overall delay between client's activity start (mouse or touch gesture), and the moment when the first visualized video frame was received from the server (including image generation in the rendering module, compression in the encoder and all the network transfers). Both tests were run for 5 and 10 simultaneous users connected to a single session, using single rendering and encoding machines. Every group was tested three times, using different screen resolutions of the clients' devices: 320x240, 800x480 and 1366x768.

The average measured RTMP communication latency was 0.05 ms and with this level it didn't affect the overall system efficiency. The overall encoding time of a single video frame was also gratifying. For the typical mobile devices' resolutions (320x240 and 800x480) the visualization's latency was lower than 100 ms in most cases. Only when 1366x768 resolution was set and 10 simultaneous users were connected to a single session, the encoding latency increased to 300 ms. However, even then it did not affect the overall comfort of the visualization session.

6 Conclusion and Further Work

In this paper we presented the system we have developed for a distributed collaborative visualization of remote datasets. The system allows for visualization of a multidimensional data on a variety of modern mobile devices. Distance users can join sessions and collaborate over the remote data in real time using a built-in teleconferencing system. They can also manipulate the data individually, without affecting rest of the participants. Obtained results of the performance tests showed, that thanks to the distributed architecture our system is able to effectively serve many simultaneous participants, even if they are using different devices.

In the future we are planning to experiment with the multicast and peer-to-peer communications, which we believe, in some cases should further increase the overall efficiency of our system. Beside the many laboratory tests that have been taken so far, one of the most important thing is also the evaluation of the users' perception of the system. The current results of the latency in communication between the servers and mobile users show, that the fluency and interactivity of the remote visualization could be nearly the same, as if the data were stored locally. We are planning to do much bigger researches in this area soon, once the final version of the application's user interface will be prepared.

References

1. Brodlie, K.W., Duce, D.A., Gallop, J.R., Walton, J.P.R.B., Wood, J.D.: Distributed and Collaborative Visualization, pp. 1–29. The Eurographics Association and Blackwell Publishin (2004)
2. Hu, S.: A Case for 3D Streaming on Peer-to-Peer Networks. In: Web3D 2006. The Association for Computing Machinery, Inc. (2006)
3. Sung, W., Hu, S., Jiang, J.: Selection Strategies for Peer-to-Peer 3D Streaming. In: NOSSDAV 2008 (2008)
4. Mosmondor, M., Komericki, H., Pandzic, S.: 3D Visualization of Data on Mobile Devices. In: IEEE MELECON 2004 (2004)
5. Lipman, R.R.: Mobile 3D visualization for steel structures. Automation in Construction 13, 119–125 (2004)
6. Engel, K., Ertl, T.: Texture-based Volume Visualization for Multiple Users on the World Wide Web, `http://www.vis.uni-stuttgart.de/ger/research/pub/pub1999/EGVE99.pdf`
7. Zhou, H., Qu, H., Wu, Y., Chan, M.: Volume Visualization on Mobile Devices, `http://www.cse.ust.hk/~huamin/pg06_mobilevis.pdf`
8. Constantinescu, Z., Vladoiu, M.: Adaptive Compression for Remote Visualization. Buletinul Universitatii Petrol-Gaze din Ploiesti LXI(2), 49–58 (2009)
9. Ma, K., Camp, D.: High Performance Visualization of Time-Varying Volume Data over a Wide-Area Network. IEEE (2000), `http://www.cs.ucdavis.edu/~ma/papers/sc2000.pdf`
10. Dragan, D., Ivetic, D.: Architectures of DICOM based PACS for JPEG2000 Medical Image Streaming. ComSIS 6(1) (June 2009)
11. Lin, N., Huang, T., Chen, B.: 3D Model Streaming Based on JPEG2000, `http://graphics.im.ntu.edu.tw/docs/tce07.pdf`

12. Stegmaier, S., Magallon, M., Ertl, T.: A Generic Solution for Hardware-Accelerated Remote Visualization. In: IEEE TCVG Symposium on Visualization (2002)
13. Cheng, L., Bhushan, A., Pajarola, R., Zarki, M.: Real-Time 3D Graphics Streaming using MPEG-4,
http://vmml.ifi.uzh.ch/files/pdf/publications/3DMPEG4.pdf
14. Noimark, Y., Cohen-Or, D.: Streaming Scenes to MPEG-4 Video-Enabled Devices. IEEE Computer Graphics and Applications (January/February 2003)
15. Childers, L., Disz, T., Olson, R., Papka, M.E., Stevens, R., Udeshi, T.: Access Grid: Immersive Group-to-Group Collaborative Visualization, http://www.ipd.anl.gov/anlpubs/2000/07/36282.pdf
16. Knodel, S., Hachet, M., Guitton, P.: Visualization and Interaction with Mobile Technology. In: MobileHCI 2008 (2008)
17. Wang, M., Fox, G., Pierce, M.: Grid-based Collaboration in Interactive Data Language Applications,
http://grids.ucs.indiana.edu/ptliupages/publications/GridCollabIDL_ITCC2005.pdf
18. Manssour, I.H., Freitas, C.M.D.S.: Collaborative Visualization in Medicine. In: WSCG 2000 (2000)
19. Engel, K., Sommer, O., Ertl, T.: A Framework for Interactive Hardware Accelerated Remote 3D-Visualization,
http://www2.ccc.uni-erlangen.de/projects/ChemVis/VisSym2000.pdf
20. Lee, S., Ko, S., Fox, G.: Adapting Content for Mobile Devices in Heterogeneous Collaboration Environments,
http://grids.ucs.indiana.edu/ptliupages/publications/icwn03.pdf
21. Goetz, F., Domik, G.: Remote and Collaborative Visualization with openVISAAR, http://www.cs.uni-paderborn.de/fileadmin/Informatik/AG-Domik/publications/Remote_and_Collaborative_Visualization_with_openVisaar_VIIP_2003_.pdf
22. Engel, K., Sommer, O., Ernst, C., Ertl, T.: Remote 3D Visualization using Image-Streaming Techniques,
http://www.vis.uni-stuttgart.de/ger/research/pub/pub1999/ISIMADE99.pdf
23. Hereld, M., Olson, E., Papka, M.E., Uram, T.D.: Streaming visualization for collaborative environments,
http://www.mcs.anl.gov/uploads/cels/papers/P1512.pdf
24. Adobe Inc., Open Screen Project, http://www.openscreenproject.org/
25. Pajarola, R., Rossignac, J.: Compresed Progressive Meshes. IEEE Trans. Vis. Comput. Graph. 6(1), 79–93 (2000)
26. Chen, Z., Bodenheimer, B., Barnes, J.F.: Robust Transmission of 3D Geometry over Lossy Networks. In: Conf. on 3D Web Technology (2003)
27. Kim, J., Lee, S., Kobbelt, L.: View-dependent Streaming of Progressive Meshes (2004)