Remote Control of Virtual Environments Using Image Streams

Ivo Haulsen
Thomas Jung
Daniel Tuchtenhagen

German National Research Center for Information Technology,
Institute for Computer Architecture and Software Technology,
Rudower Chaussee 5, D-12489 Berlin, Germany,
Tel. +49/30/6392-1779, Fax -1805,
E-mail ivo,tj,daniel@first.gmd.de

Abstract

This paper presents a scenario for the remote control of virtual environments using low-bandwidth networks. Through visual feedback from digital video streams, a low-end personal computer can be used as a "control terminal" for visual supercomputing applications. A wavelet-based video codec allows about four frames per second to be generated with a common video resolution via a single 64kbps ISDN B-channel. As a result, users with "low-end equipment" can participate in distributed heterogeneous virtual environments.

Keywords: Teleoperation, Visual Supercomputing, Distributed Virtual Environments, Video Compression, ISDN

1 Background

Teleoperation and Cooperative Work in Virtual Environments are becoming increasingly important.. Current high-end graphics workstations and high-bandwidth networking environments support the development of "virtual products" to optimize planning and development processes.

Moreover there is interest in integrating small and medium-sized enterprises (SMEs) to exploit the benefits of these new information and communication technologies. At present, most SMEs use personal computers. Since the cost of high-bandwidth connections is high, SMEs are connected via ISDN only.

1.1 Distributed Virtual Environments

There are already a number of Distributed Virtual Environments (DVE) (e.g. [1, 2, 3, 4, 5, 6, 7]). In these, parts of the application normally have to share

the three-dimensional-scene database via a network. In single-user environments (e.g. VRML2.0 scenes), the whole scene usually has to be downloaded for visualization on a local computer.

If there are multiple users participating, a DVE has to be consistent for these different users. While investigating the virtual environment, information has to be exchanged between the multiple instances of the virtual scene. For example, the positions of the users' avatars have to be updated via the network.

Since the network loads required for data exchange in such environments are high, high-bandwidth networks (ATM) are used in most cases[1, 2, 5].

The availability of low-priced, hardware-based 3D-graphics acceleration allows local rendering of complex three-dimensional scenes. In view of continuing performance differences between personal computers and high-end graphics supercomputers, remote access to computing and rendering power via networks is still of interest.

For example, a visual supercomputing application like a virtual wind tunnel [8] cannot run on a personal computer because it lacks the computing power needed to simulate the flow field. An application of this sort involves highly modifying a geometrical scene (e.g. thousands of particles in a flow field). This is why the real-time update of a three dimensional scene database is not always possible via a single ISDN B-channel. Visual coupling using image streams is helpful for these kinds of virtual environments.

1.2 Video Streaming

Video streaming has many applications (e.g. videoconferencing, teleoperation, telerobotics and digital television broadcasting). While high-bandwidth networks sometimes allow the transmission of uncompressed video data, the use of low-bandwidth media requires video compression. The choice of the video-coding algorithm usually depends on the medium's bandwidth. While MPEG-1 and MPEG-2 are designed for bandwidths around 1.5 Mbit/s for low bandwidth ISDN connections with p*64 kbps (p=1..30), video conferencing standards like H.261 and H.263 are more appropriate. All these methods are based on the discrete cosine transform. Alternatively, a wavelet-based method can be used[9].

In principle, single images (intraframe coding) or differential images (interframe coding) can be compressed. The use of differential images may be sufficient in videotelephony where a much of the image comprises an invariable background. Moving objects and cameras require motion compensation to take advantage of image coherences. While the compression and the decompression of full and differential images can done approximately in the same time (symmetric coding) motion-compensation algorithms require considerable computational effort on the coding side. Thus, parallel computation can be used to enable real-time coding (e.g. [10]).

2 Remote-Control Scenario

In the following sections, a remote-control scenario is presented in which a low-end personal computer (e.g. a notebook PC) is coupled with a high-end workstation cluster or a parallel supercomputer (host) via a single 64kbps ISDN B-channel.

2.1 General System Architecture

Running on the host is a so called "Embedded Server for Rendering Architectural scenes" (ESRA), which is connected to multiple architectural workplaces. Architects who are modelling new buildings send them to ESRA, which puts the models into a common scene database. This database is used for both the virtual environment and high quality renderings.

The virtual environment can be distributed over a workstation cluster including an SGI Impact for the rendering or a visual supercomputer (MANNA/VISA [11, 12]). The DVE can be controlled either by a user sitting at the host or by an architect from a remote workplace using visual feedback from image streams (see Fig. 1).

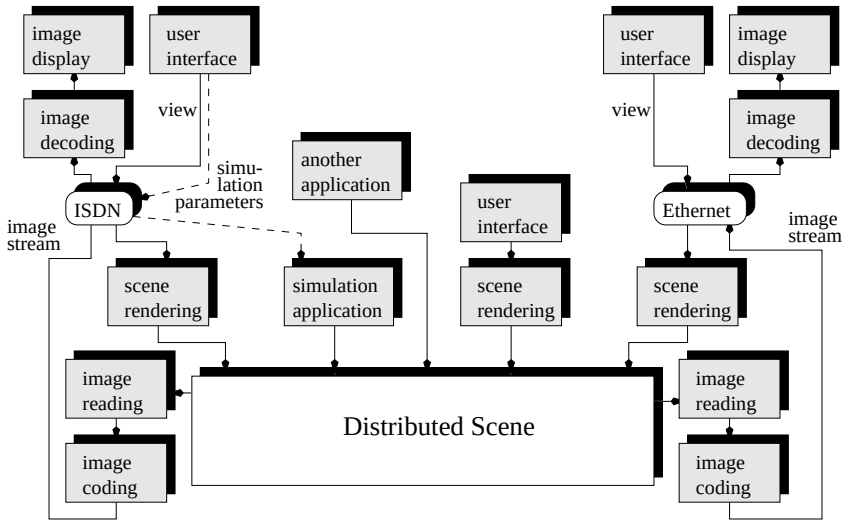

Figure 1. Architecture of the ESRA system

2.2 Test Scenario

In the following tests, a single user sitting at a remote workplace controls a virtual environment running on a single workstation (Sun Sparc). Rendering of

the moderately-sized scene is done on another workstation (SGI Impact). The simplification (Fig. 2) is made to isolate the main problems relating to the remote-control scenario.

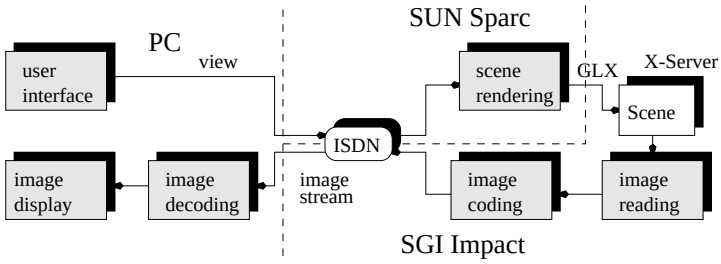

Figure 2. Test scenario

After transmitting the complete scene (in VEML format) from the remote PC to the host, the virtual environment (VE) is started. The VE, running on the SUN Sparc Workstation, sends GLX-calls (OpenGL extension for X11) to the X-Server of the SGI Impact. The rendered images are read locally from the frame buffer via an OpenGL-call, after guaranteeing[1] that the full image is available.

On the SGI workstation, the images are compressed and sent back to the remote PC. The PC decompresses and displays these images. Visual feedback gives the architect the information needed to generate new camera positions with the input device, there then being transmitted to the host. This data is used for generation of the next image.

For the following tests, we used a moderately-sized model of our institute's building, featuring areas in one colour, textured areas based on photographs, and synthetic textures like "grass" and "stones"[2]. This kind of model is chosen to avoid adaptation to a special image type (e. g. with large areas in one colour). Testing the remote-control scenario with other types of models resulted in similar performance values.

2.3 Image Coding

The coded images are transmitted via a single ISDN B-channel with a bandwidth of 64 kbps. We used TCP/IP (running PPP at both ends of the ISDN connection) to simplify the routing between the remote PC and the host.

For the image coding and decoding, a wavelet-based software video codec by the Berlin College of Technology and Business Studies was provided. Orig-

[1] A glFinish call from the SUN

[2] Synthetic textures generated by a noise function usually cause lower compression rates.

Quality parameter	30	60	90	120	150	180
Compression rate	17.9	27.5	37.2	46.8	56.2	63.8
Bytes per frame	4268	2784	2055	1633	1360	1199
Time for coding per frame in milliec.	46	39	34	30	28	27
Time for decoding per frame in milliec.	46	39	34	31	28	27

Table 1. Compression and performance depending on q (QCIF resolution)

inally, the codec was developed for teleconferencing applications. It supports different image sizes, as well as monochrome and coloured modes. Currently, the codec supports interframe and intraframe coding; motion compensation is under construction. We chose this codec because of its adaptability to the special requirements of the application. Moreover, it outperforms the MPEG-4 Video Verification Model (VM 5.1) in both subjective and objective quality [9].

In videoconferencing applications, monochrome images are often used to reduce bandwidth requirements. However, architects prefer coloured images. Also it is our experience that, to obtain higher frame rates, architects are willing to accept reduced image resolution.

Unlike in teleconferencing applications, the generation of interframes does not produce a reasonable reduction in bandwidth requirements. In teleconferencing applications, "real" cameras are normally mounted. As the background is constant, the coding of differential images reduces the required bandwidth per frame. However, virtual cameras, controlled by architects in a walk-through, move very quickly between two consecutive frames. Image coherency can only be used if it is possible to apply motion compensation. Since the motion-compensation algorithm for the wavelet codec is not currently available, only intraframe coding was used in the following tests.

2.4 Image Quality

The compression rate and image quality can be controlled by a single parameter, called q. The higher the value, the higher the compression. Figure 4 illustrates the relation between the image quality of the codec's output and the value of q[3]. The images are in QCIF format and are converted to a grey-scale postscript format for printing. Table 1 shows the compression rate (original size divided by compressed size), the compressed size per image and the time needed for compression. While the subjective image quality for CIF images is the same as for QCIF images, the compression rate for CIF is usually higher.

3 Results

The first step was to measure the execution time of single operations in the remote-control scenario. The computational effort required for most of these

[3] We used half resolution and double q-value for u and v.

Platform	Operation	Time per frame
SGI Impact	read framebuffer, RGB to YUV	0.008s
R4400 250 MHz	compression	0.142s..0.086s (q=30..160)
ISDN 64kbps	transmission	1.5s..0.3s (q=30..160)
Notebook	decompression	0.28s..0.16s (q=30..160)
Pentium 120MHz	YUV to RGB, display	0.03s

Table 2. Average Time per Frame (CIF, 352x288)

operations scales with the image size.[4] We chose CIF resolution, which is widely used in teleconferencing applications.

The operations on the different computing platforms (PC, SUN workstation and SGI workstation) in a remote control scenario have to be done in parallel to avoid latencies. As shown in Table 2 the operations on the SGI Impact have no major impact on performance within the overall system. If the compression rate is increased to accelerate image transfer via the ISDN B-channel, the time for compression and decompression actually decreases.

The main bottleneck is the limited bandwidth of the ISDN connection. Neglecting the overhead for the TCP/IP-protocol and the operating system, in Table 2 it is assumed that exactly 64kbit of compressed image data per second can be transmitted. With normal video resolutions it is possible to visualize about four frames per second. This frame rate can easily be supported by the current scenario (SGI with R4400 and PC with Pentium 120). If it is possible to double the effective bandwidth of the ISDN connection[5], the switch to a Pentium 200MHz processor and an SGI with a R10000 processor, for example, would allow also the doubling of the frame rate.

3.1 Latency

If the visual feedback corresponding to the last input (e.g. mouse move) is to be presented to users before a new user input is accepted, all operations have to be done sequentially and the frame rate will decrease. On the other hand, if camera control is decoupled from the visual feedback, the time delay between the user's input and visual feedback may increase until navigating becomes impossible. A fixed number of camera positions is therefore sent in advance, before the system is waiting for a returned image. Sending one additional camera position at the beginning usually causes a doubling of the frame rate. If two additional positions are sent in advance the speedup sometimes increases.

Figure 3 shows an "ideal" scenario in which two additional positions result in a tripling of the frame rate[6]. A sufficiant number of additional camera positions

[4] An exception here is the time needed for coding and decoding, which depends not only on the image size, but also on the image details per image size.

[5] Currently, the driver for our AVM PCMCIA-ISDN-card only supports a single 64kbps B-channel.

[6] In the direction PC to host, there is a great deal of free capacity that can be used to

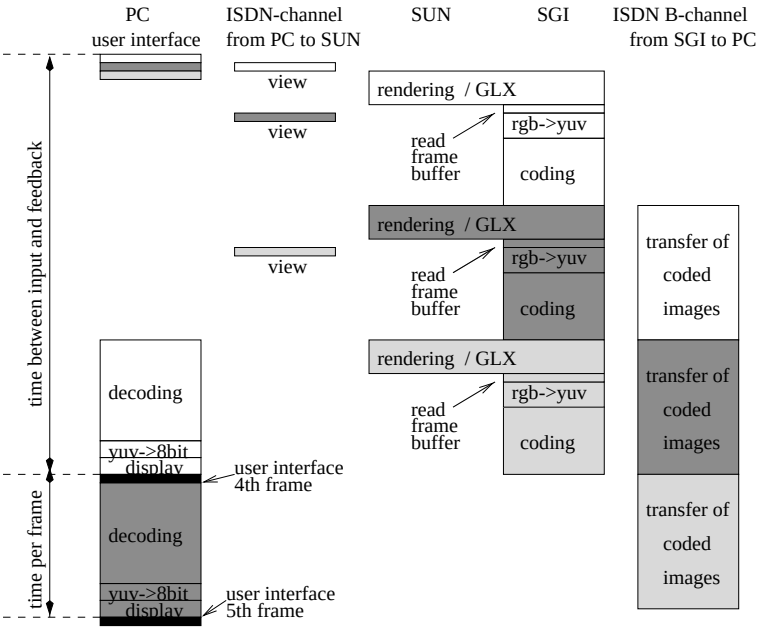

Figure 3. Control scenario with two images sent in advance

at the beginning does not cause a time delay for the visual feedback. Only the frame rate will improve.

3.2 Walking Through the Scene

We let a user walk through the scene several times to evaluate system performance, measuring the frame rate and the transferred bytes. Values did not vary substantially from cycle to cycle. After about 10 cycles, with some hundred frames each, we changed the image resolution and/or the value of the quality parameter q. The average values of the walk-through cycles are given in Table 3.

As can be seen, the transfer rate is mostly near to the limit of 64 kbps. Where a value is clearly below this, the decoding performance of the Pentium 120 processor is the bottleneck.

---

transmit information from the remote workplace (e.g. information about a user's avatar).

Image size	80x64		160x128		320x240		352x288	
Quality	Frames/s	kbps	Frames/s	kbps	Frames/s	kbps	Frames/s	kbps
30	8.0	61.0	2.2	62.0	1.0	62.5	0.8	62.5
90	11.0	59.5	4.7	60.0	2.1	62.0	1.8	62.5
150	14.0	57.5	6.6	62.0	3.0	58.0	2.8	62.0
210	16.5	59.5	9.5	57.0	4.0	56.5	3.8	61.5

Table 3. Average walk-through times

4 Conclusion and Future Work

A remote-control scenario for the interactive manipulation and inspection of virtual environments via a single ISDN B-channel was presented. We achieved an average frame rate of about four frames per second, walking through a moderately-sized, textured architectural scene in CIF resolution.

While this frame rate is acceptable for inspection of virtual scenes, for the manipulation of highly modifying objects and scenes the frame rate must be at least doubled. This can be achieved by using an additional ISDN B-channel or by increasing the compression rate. The next version of the wavelet-based codec will support motion compensation. Since the motion-compensation algorithms will probably not be fast enough on a single processor parallelization must be considered.

As the quality parameter remains constant throughout a walk-through cycle, the frame rate, which depends on the image complexity, varies. The next step will be for the quality to be chosen automatically by the system to ensure the required frame rates. Moreover, the quality should be adapted to the velocity of the camera to allow higher frame rates for fast camera moves at lower image qualities.

In the given scenario the only way to interact is navigating through the scene. In the next version at least all input features of VRML2.0 should be supported.

Additionally, multicast scenarios should be explored in which one user controls the walk-through, and another gets the visual feedback only. Finally, the performance of the remote-control system must be evaluated in distributed versions of ESRA and for different application types.

5 Acknowledgements

We wish to thank Hanc Cycon for providing the excellent video codec, Alexander Haderer und Mark Palkow for their technical support, Oliver Wagner for implementing parts of the system, and Gabriele Layer-Jung and Phil Bacon for polishing up the English text. This work is partially funded by the Berlin Senat. Administration for the Economics and Business under grant no. VB1-7.6.-6.08.

Bibliography

1. Wim Lamotte, Eddy Flerackers, Frank Van Reeth, Rae Earnshaw, and Joao Mena De Matos. Vidnet: Collaborative 3D Visualization and VE over ATM Networks. *IEEE Computer Graphics and Applications*, pages 66–75, 1997.

2. Jose Miguel Salles Dias, Ricardo Galli, Antonio Carlos Almeida, Carlos A. C. Belo, and Jose Manuel Rebordao. mWorld: A Multiuser 3D Virtual Environment. *IEEE Computer Graphics and Applications*, pages 55–65, 1997.

3. Alex Pang and Craig Wittenbrink. Collaborative 3D Visualization with Cspray. *IEEE Computer Graphics and Applications*, pages 33–41, 1997.

4. Valerie D. Lehner and Thomas A. DeFanti. Distributed Virtual Reality: Supporting Remote Collaboration in Vehicle Design. *IEEE Computer Graphics and Applications*, pages 13–17, 1997.

5. Michael R. Macedonia and Stefan Noll. A Transatlantic Research and Development Environment. *IEEE Computer Graphics and Applications*, pages 76–82, 1997.

6. Michael T. Gardener and Phillip Amburn. Simulation-Based Remote Debriefing for Red Flag Missions. *IEEE Computer Graphics and Applications*, pages 30–39, 1997.

7. C. Carlsson and O. Hagsand. DIVE - A Platform for Multi-User Virtual Environments. *Computers & Graphics*, 17(6):663–669, November/December 1993.

8. Alexander del Pino, Thomas Jung, and Thomas Stickdorn. Interactive Flow-Field Evaluation on a Distributed-Memory Architecture. In *Proceedings of Graphicon 1996*, July 1996.

9. D. Marpe and H.L. Cycon. Efficient Pre-Coding Techniques for Wavelet-Based Image Compression. In *Proceedings of PCS'97*, Berlin, 1997.

10. A. C. P. Loui, A. T. Ogielski, and M. L. Liou. A Parallel Implementation of the H.261 Video Coding Algorithm. In *Proc. of the IEEE Workshop on Visual Signal Processing and Communications*, pages 80–85, 1992.

11. W. K. Giloi, U. Brüning, and W. Schröder-Preikschat. MANNA: Prototype of a Distributed Memory Architecture With Maximized Sustained Performance. In *Proceedings Euromicro PDP96 Workshop*. IEEE - CS Press, 1996.

12. D. Jackèl and H. Rüsseler. A Real Time Rendering System with Normal Vector Shading. In Wolfgang Straßer, editor, *9th EUROGRAPHICS Workshop on Graphics Hardware*, Oslo, Norway, 1991.

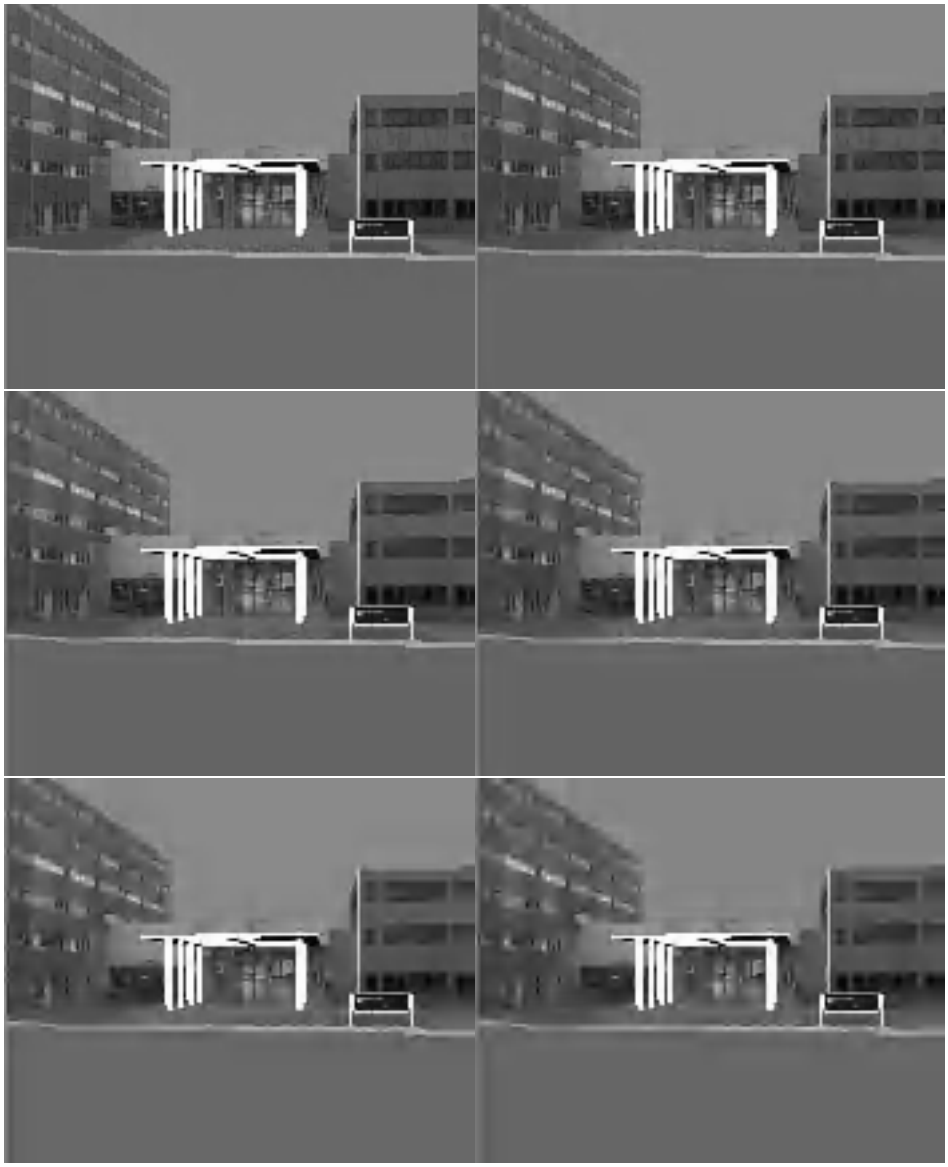

Figure 4. QCIF images with quality parameter from 30 to 100