Data Streaming in Telepresence Environments

Edouard Lamboray* Stephan Würmlin Markus Gross
Computer Graphics Laboratory
Swiss Federal Institute of Technology (ETH)
Zurich, Switzerland
lamboray@cyfex.com *{wuermlin, grossm}@inf.ethz.ch*

In this paper, we discuss data transmission in telepresence environments for collaborative virtual reality applications. We analyze data streams in the context of networked virtual environments and classify them according to their traffic characteristics. Special emphasis is put on geometry-enhanced (3D) video. We review architectures for real-time 3D video pipelines and derive theoretical bounds on the minimal system latency as a function of the transmission and processing delays. Furthermore, we discuss bandwidth issues of differential update coding for 3D video. In our telepresence system – the blue-c – we use a point-based 3D video technology which allows for differentially encoded 3D representations of human users. While we discuss the considerations which lead to the design of our three-stage 3D video pipeline, we also elucidate some critical implementation details regarding decoupling of acquisition, processing and rendering frame rates and audio/video synchronization. Finally, we demonstrate the communication and networking features of the blue-c system in its full deployment. We show how the system can possibly be controlled to face processing or networking bottlenecks by adapting the multiple system components like audio, application data and 3D video.

Keywords: H.4.3 [Communications Applications]: Computer conferencing, teleconferencing, and videoconferencing. I.3.2 [Graphics Systems]: Distributed/ network graphics. I.3.7 [Computer Graphics]: Three-dimensional graphics and realism

1. Introduction

The use of spatially-immersive projection environments as portals for telepresence applications offers unprecedented possibilities for collaboration and communication of multiple users, located at distant sites. Along with data acquisition and rendering, a spatially-immersive portal for telepresence also raises many technical challenges for data communication. The time-varying state of a distributed collaborative virtual reality application needs to be transmitted to remote sites, respecting the delay tolerances for interactive user communication.

* The author is now with Cyfex AG, Zurich, Switzerland.

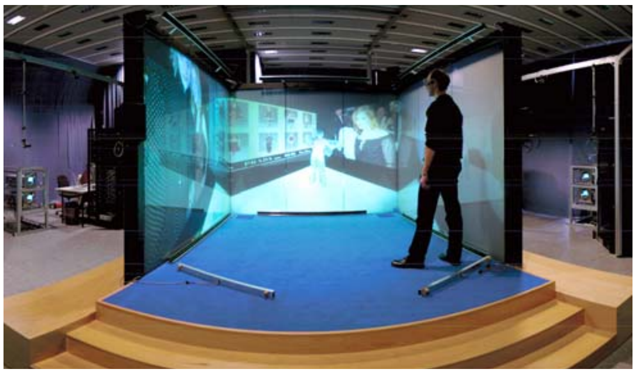
Figure 1: A blue-c CAVE-like portal.

The data generated by the system does not only include application data, but also audio/video streams for teleconferencing. A special novelty of our telepresence system – the blue-c – is the visual representation of the user [5]. The users are rendered based on 3D geometry data, the appearance attributes being extracted from multiple live video cameras.

Moreover, the design of our system is motivated by the guideline to use an attribute-centric approach and thus, to avoid approaches based on a priori models. On the one hand, the use of a priori models for describing real-world or synthetic data often allows for a very efficient data representation. On the other hand, the rendering of the data is also limited by the data model. In fact, model-based approaches do not allow for motions, actions or deformations which are not described by the model in the first place. And allowing for those alterations results in adapting – if not completely changing – the underlying model. In most cases, this task cannot be performed without modifying the system's implementation. Concerning the manipulation of objects in the virtual environment, we intend to reproduce the modifications at the remote sites as naturally and continuously as they are initially performed.

Furthermore, porting a collaborative virtual reality telepresence system from an experimental setup in a well controlled and well behaved laboratory into a production application requires that the system is able to adapt to various networking and application conditions. Thus, we investigated possibilities for application adaptation and fault-tolerance during adverse networking conditions and performance bottlenecks.

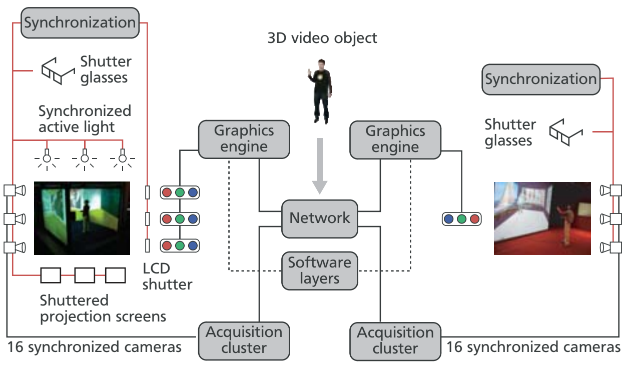

Figure 2: Hard- and software components of the blue-c system.

At ETH Zurich, we investigate novel immersive projection and acquisition environments for telepresence [5]. During the blue-c project, two networked virtual reality portals, consisting of CAVE-like environments, augmented by an array of cameras, were developed. In full operation mode, the two portals enable networked collaborative applications enhanced by 3D video conferencing features. Figure 2 depicts the components of the blue-c system, including the combined acquisition and projection hardware based on shuttered projection screens, the acquisition and rendering hard- and software and the distributed software architecture. The mainly graphical application data is handled by a distributed shared scene graph [13].

This paper is organized as follows: After a short discussion of related work, we analyze the data streams in networked virtual environments in Section 2. In Section 3, we discuss 3D video architectures for teleconferencing and in Section 4 we present the design of the system we implemented. In Section 5 we present application simulation results from our telepresence environment.

1.1. Related work

Cruz-Neira et al. demonstrated the first CAVE, a surround-screen projection-based virtual reality environment [2]. Raskar et al. presented an unprecedented system allowing for simultaneous per-pixel depth extraction and projection in an office environment [16]. Their Office of the Future demonstrated the potentialities of the smart combination of real-time computer vision and computer graphics technologies for telepresence applications. Subsequent examples of combining simultaneous acquisition of live 2D or 3D video streams and full-wall projection displays are the Teleport environment [3], the 3D video conferencing system by Kauff and Schreer [7], and the National Tele-Immersion Initiative [17].

Free-viewpoint video can be computed by extracting geometry and texture information from a set of concentric views of the same object. We will refer to this geometry-enhanced video streams as *3D video*. Matusik et al. investigated real-time reconstruction and rendering using the visual hull algorithm [12]. This approach is also used by Prince et al. in their augmented reality enhanced video conferencing system [15].

A variety of networked virtual environment architectures have been proposed in the literature, but only few researchers investigated the characteristics of the data to be shared and transmitted. In the NPSNET system, objects in the virtual world are modelled as entities having state information, e.g. attributes like location, velocity, color, orientation [11]. Audio and video streaming has been integrated into the MASSIVE architecture and changes in the scene are communicated following the event communication paradigm [4]. Leigh et al. investigated whether differentiated service mechanisms are suitable for tele-immersive applications [10]. In that context, they classified the data streams of tele-immersive applications with respect to the networking characteristics and requirements.

2. Data streams in telepresence environments

2.1. Data stream classification

Using the results from previous research on networked virtual environments, we propose to distinguish the following data streams in telepresence environments:

- *Conventional 2D video and audio data*: These data streams have been thoroughly investigated in the past. Their deployment within networked virtual environment applications is similar to their usage in conventional video conferencing applications.
- *Avatar data*: Contains the information required for describing an avatar's state. This information typically has a predefined format and a fixed size.
- *3D Video*: Unlike conventional video, 3D video contains geometric information of the respective object or scene. This data thus allows for free-viewpoint and stereoscopic rendering.
- *Application updates*: Data which is generated interactively by the users while working with the application. We distinguish updates which lead to visual changes and updates which only concern internal application states but do not lead to visual changes. In summary, these data updates are generated through *user interaction* with the scene and the application. If the underlying scene data structure is a distributed shared scene graph, the scene graph update messages fall into this category.
- *Initial application data*: Data which leads to a predefined state in the application. A predefined starting point is needed at application start-up time, but can also be required at several intermediate stages. This data also includes the initial scene description.
- *Awareness management data*: Fixed format data describing where a user is focusing on in the virtual environment, e.g. a user's current position and viewing direction.
- *System control data*: Fixed format data describing internal system states.

Data which is of a predefined format and which does not depend on the user interaction is predictable and, in some sense, well-behaved for transmission over computer networks. The data streams falling into this category are awareness management and system control data, possibly also avatar state information and, to some extent, audio data.

Conventional 2D as well as the novel 3D video formats present the difficulty that the required bandwidth depends strongly on the deployed coders. Since most coders use motion prediction, the short-term bandwidth depends heavily on the user's motion and hence the streams present a medium or high burstiness. For video transmission in general, a regular update rate is more important than a lossless transmission. High packet loss rates however lead to prominent visible artifacts.

Application updates, or scene graph messages, also present a potentially medium or high burstiness and are not easy to predict, since they completely depend on the user interacting with the application. In many applications, their average bandwidth requirements are expected to be significantly lower than those for video streams. However, efficient collaborative applications only tolerate short-term differences in the distributed shared data. Thus, a consistent data representation needs to be guaranteed through error resilient transmission of the application data. The scene description and application initialization streams are not critical because of their burstiness, since they do not present strict requirements with respect to latency or jitter.

In summary, we can distinguish three major data classes: *bulk data*, *sporadic events* and *real-time streaming* data. Each of these categories has its specific characteristics and must be handled with an appropriate communication technology. Table 1 classifies the previous data streams into these categories.

Table 1: Data stream classification.

Data category	Examples
Bulk data	Scene description, initial application data
Sporadic events	Avatar, application updates (punctual user interaction), system control data
Real-time streaming data	Audio, video, 3D video, awareness management, application updates (continuous user interaction)

Among the above data types, those which directly lead to a visual or aural output are more critical than others. This is especially true, if the visualized motion is intuitively understood as a continuous movement by the human observer.

2.2. Bandwidth

The system bandwidth B_S of a telepresence system is the achievable bit rate between the acquisition site and the rendering site. It thus includes limitations imposed by applications, hosts and communication networks. Since a telepresence application is an inherently multi-stream application, the total bandwidth is the sum of the bit rates of all individual streams b_i, i.e.

$$B_S = \sum_i b_i .$$

In a typical blue-c application, each site produces a single 3D video stream at the bit rate b_{3dv}, a high-quality audio stream at b_{audio} and an application update stream at b_{app}. Thus, B_S can be approximated by

$$B_S = b_{3dv} + b_{audio} + b_{app} .$$

The bit rate of the awareness management and system control data streams are of a lower order of magnitude and are not really important for bit rate calculations. In a reasonably controlled environment – as it is the case between our two blue-c portals, installed at distant locations on the ETH campus – the maximum network bandwidth is in the order of tens or even hundreds of megabits per second.

2.3. Latency and jitter

The *latency* or *delay* is the time elapsing between the emission of a data block at the sender, and its arrival at the receiver. In his review of virtual reality applications, Brooks counts latency – inter-system latency, as well as latency due to communication over a computer network – among the major problems of virtual reality applications [1].

If we analyze the latency of a system, it is important to notice that there exists a threshold under which delays are not perceivable. A second threshold defines the range in which interaction among experienced users is still possible, but where the user performance starts degrading. The user performance is generally measured in a usability study which analyzes how fast or how accurate a well-defined task can be executed. If the latency becomes too important, no reasonable interaction is feasible anymore. The absolute values of these thresholds depend on the task that needs to be accomplished. During a networked ball playing experiment, first problems in real-time interaction are reported for delays of 500 milliseconds, and interaction stops completely when the delays approached 1 second [18]. In the voice transmission community, it is generally admitted that communication performance drops dramatically if round-trip delays approach 500 milliseconds [8].

In general, jitter – the variation of the latency – is admitted to be more harmful than latency. Park and Kenyon confirm this statement by analyzing the user performance in collaborative virtual environments under various latency and jitter conditions [14]. They report no significant difference between experiments with a latency of 200 milliseconds without jitter and a latency of 10 milliseconds with jitter, i.e. a long latency without jitter has a lower effect on performance than short latencies with substantial jitter. The experiments of Vaghi et al. show that experienced users develop intuitive strategies to cope with system latency, but these strategies fail if the latency is not constant [18].

3. Real-time 3D Video

3.1. System architectures

There exist two main approaches for generating *free-viewpoint* or *3D video*:

- *Image-based* 3D video approaches generate the virtual views from a dense set of camera images by interpolating between neighboring views [6];
- *Geometry-based* 3D video approaches compute a geometric 3D reconstruction from a sparser set of camera images [12].

Deploying both approaches for teleconferencing leads to two different solutions. The image-based approach either requires at the rendering site the original set of camera images or a new image constructed from a subset of the input images and rendered for the respective viewpoint. The image-based approach thus requires the transmission of one or several 2D video streams.

In the geometry-based approach, either the camera streams are transmitted to the rendering site, where also the geometry computations are executed, or a truly geometry-enhanced video, containing e.g. per-pixel depth data, is transmitted. Figure 3 illustrates the two options for real-time 3D video architectures.

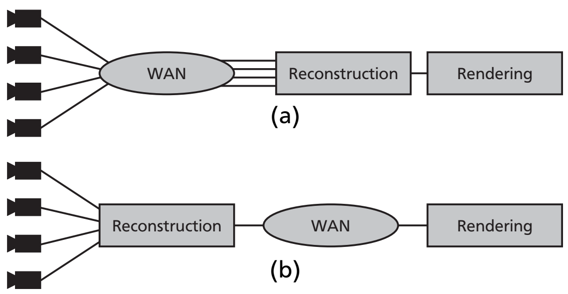

Figure 3: Architectures for a real-time 3D video streaming pipeline with acquisition and rendering at remote sites.

Many precedent real-time 3D video systems reconstruct and render the 3D video object on the same node. Such a setup, however, does not correspond to the situation one encounters in telepresence or video conferencing systems, since the acquisition and rendering sites are at physically remote locations. Hence, if we opt for a geometry-based approach and the architecture of Figure 3b, an efficient and robust streaming for the time-varying 3D data needs to be devised. This architecture however, also presents a couple of advantages. We expect the streaming of 3D data to be more bandwidth efficient than the streaming of the sum of the 2D video streams it was generated from. Moreover, the 3D reconstruction requires synchronized 2D video data which is easier to realize if all contributing nodes reside in a local area network, than if the 2D streams must be transmitted over a wide area network. Finally, the transmission of a 3D geometry-enhanced video stream frees the rendering and application nodes from the extensive processing required for the generation of the 3D user representation.

3.2. Latency constraints

In this section, we discuss the theoretical latencies for the various design options of real-time 3D video streaming systems. Let us introduce a variable Δt describing the one-way transmission delay of the communication channel and Δf describing the period of the camera frame rate. The latency of the 3D video system can be modelled using the communication delay and the processing delay. In a real-time system, the processing must not be longer than the period of the camera frame rate, i.e. the processing time per frame is bounded by Δf.

A pipeline architecture allows to stretch the processing time per frame over several nodes at the cost of a higher system latency. We propose a three-stage processing pipeline with an acquisition stage, a geometry processing stage and a rendering stage, see Figure 3. In this case, we can use two time slots for processing and the available processing time becomes $2\Delta f$. Thus, the accumulated processing latency in both pipelines of Figure 3 is of $2\Delta f + \Delta r$, where Δr is the rendering time. The minimum value for Δr is 16 milliseconds. The new approximation of the one-way system latency Λ thus becomes

$$\Lambda = \Delta t + 2\Delta f + \Delta r . \qquad (1)$$

We assume that two stages of the pipeline reside in the same LAN and we neglect the transmission delay between these two stages.

A different situation occurs during *camera hand-overs*, i.e. the renderer's virtual viewpoint changes such that the set of cameras contributing to the 3D video stream changes. In this case, the two-way system latency is the important characteristic value. We now analyze the two-way latency for a single stage and a pipelined system, with and without back-channel transmission of the renderer's virtual viewpoint.

- *S – single stage system without back-channel*:
  Since all video streams are permanently at the renderer's disposal, this architecture leads to the smallest possible delay. From Equation (1), $\Lambda_S = \Delta t + \Delta f + \Delta r$. However, this design does not enable a prefiltering of the data at the acquisition site and hence requires also the highest transmission bandwidth.
- *S+ – single stage system with back-channel*:
  In this case, the new virtual viewpoint first needs to be transmitted to the acquisition site. Moreover, we expect an average time of $0.5\Delta f$ before the cameras grab new images. Thus, the expected latency becomes $\Lambda_{S+} = 2\Delta t + 1.5\Delta f + \Delta r$.
  We observe that Λ_{S+} is almost twice as high as Λ_S, but in this configuration the required bandwidth is only K/N times the bandwidth of system S, since only the K video streams required for the final rendering are transmitted. In the real-time 3D video pipeline which we describe in Section 4.1, $N = 16$ and $K = 2$ or $K = 3$.

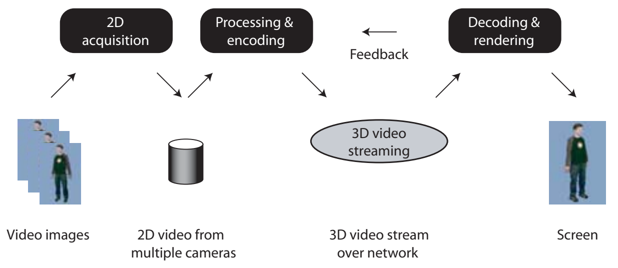

Figure 4: Pipelined processing of a 3D video stream. The feedback line illustrates the back-channel transmission of the virtual viewpoint.

- *P – pipelined system (3 stages) without back-channel*: From Equation (1), $\Lambda_P = \Delta t + 2\Delta f + \Delta r$. The comments on the required bandwidth of system S remain valid.
- *P+ – pipelined system (3 stages) with back-channel*: Similar to S+, we must take into account the transmission time of the virtual viewpoint and the average time until the next image grabbing. We thus obtain $\Lambda_{P+} = 2\Delta t + 2.5\Delta f + \Delta r$.

Figure 4 illustrates the pipelined processing of a 3D video stream and Figure 5 depicts the above system latencies as a function of the camera frame rate and for various transmission delays. We observe quite high latencies at the low frame rates which are essentially provoked by the relatively high value of Δf. Although the system inherent latency remains quite high, the system becomes practicable at rates of 10 to 15 frames per second in most configurations. The configurations with $\Delta t = 75\text{ms}$, e.g. a transatlantic connection between ETH Zurich and the US West Coast, lead to a considerable latency which is due to the high value of Δt. The distinction between S/P and S+/P+ however, is only relevant for analyzing delays during camera hand-overs because of virtual viewpoint changes. For describing the one-way latency of e.g. the user performing a movement and the respective rendering, the S and P curves are representative.

From the previous discussion, we conclude that real-time 3D video at 10 or 15 fps can be used in teleconferencing applications. A three-stage processing pipeline provides sufficient computation time and still allows for interactive communication. For long-distance communication however, the latency constraints imposed by the rules of interactivity for multi-user communication systems are difficult to meet. Moreover, all previous results are of a theoretical nature and suppose lossless communication channels. Thus, the practical system latencies will be larger, eventually because of packet loss or sub-optimal system implementations.

A common technique to prevent jittering in the rendering frame rate after a networked transmission is to use an extra buffer at the renderer which then can be used for smoothing the playback frame rate. Unfortunately, the extra buffer also introduces an extra delay. Since the system inherent latency of a 3D video pipeline does not leave much time for extra delays, we suggest to reduce jittering by controlling the frame rate at all pipeline stages and by adapting the data processing if performance or transmission problems arise. This strategy however, does not allow for a complete elimination of jitter.

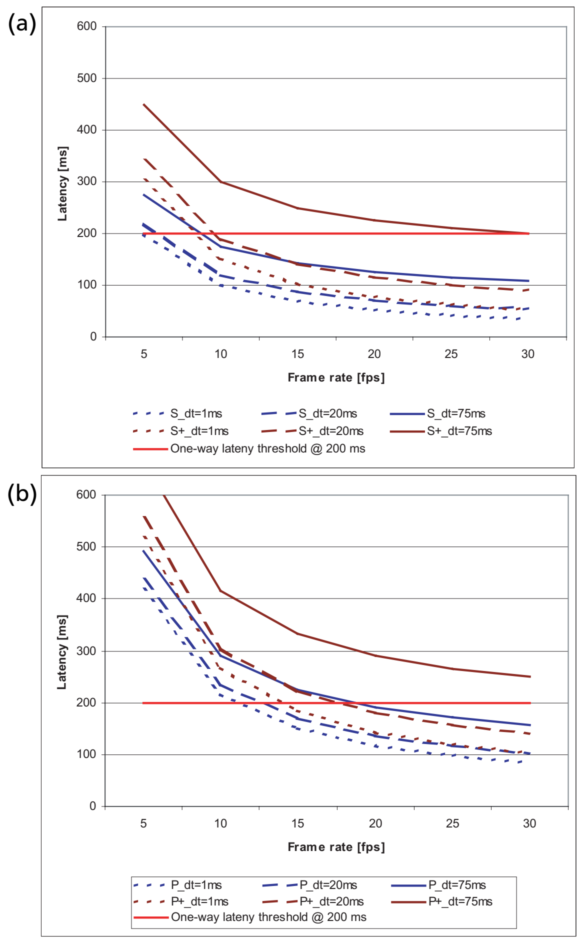

Figure 5: System latency of various real-time 3D video pipelines as a function of the camera frame rate for various transmission delays. The latency threshold at 200 ms guarantees an acceptable round-trip latency for duplex human communication. a) Single stage system; b) 3-stage pipelined system.

3.3. 3D Video in blue-c

In the blue-c system, the underlying user representation is a 3D point sample cloud. With this data representation, we can establish a direct mapping from a pixel in 2D image space to a point sample in 3D space. Thus, we avoid interpolation and alignment artifacts which frequently occur in mesh-based representations, where geometry and texture are treated as heterogeneous data types. Furthermore, we do not rely on complicated and restricting a priori models, e.g. avatars or articulated human-body models.

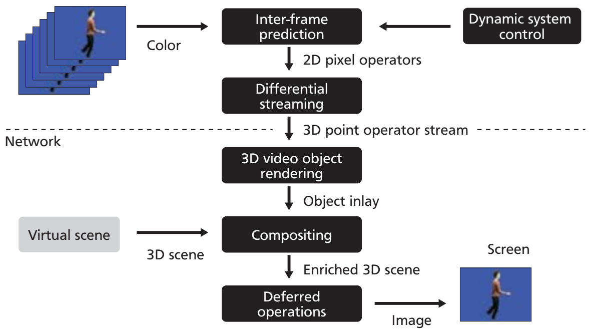

Figure 6: 3D video transmission and rendering.

Our real-time 3D video system uses a pixel-based differential update scheme, which exploits the spatio-temporal inter-frame coherence. We detect pixels which, within two consecutive frames, change from background to foreground or vice versa. The pixels which are in the foreground in both frames are analyzed with respect to attribute changes, e.g. geometry and color. The reconstruction process transforms the 2D pixels into 3D point samples using the geometry information provided by the silhouettes. For this purpose, we use a variant of the image-based visual hull algorithm [12]. In each frame, the reconstructed 3D object can be described by a stream of point sample operators which insert, delete or change the attributes of individual point samples. Since our differential update scheme is image-based, we obtain an efficient load-balancing by using the system architecture of Figure 3b.

At the receiver node, the 3D video stream is rendered from the current virtual viewpoint only. Thus, the back-channel communication of the observer's virtual viewpoint allows to optimize the 3D video reconstruction and streaming for the respective view.

Figure 6 illustrates the processing steps which lead from 2D images to a 3D video object integrated into a virtual scene. A detailed description of our differential update scheme, the dynamic viewpoint adaptation and the transmission of real-time 3D video can be found in [9, 19].

3.4. Bandwidth constraints

In this section, we focus on the operator framework of our real-time 3D video pipeline. We evaluate various configurations of the differential update scheme for geometry attributes and discuss the consequences of the subsequent bandwidth and quality trade-offs.

For this purpose, we simulate the real-time processing of a 3D video sequence of 100 pre-recorded frames. During this experiment, the dynamic viewpoint adaptation is disabled and, hence, no progressive sampling is performed [19]. This leads to significantly higher bit rates than in a real-time session since all pixels are processed in each frame. In this experiment, we evaluate the update operator for the point sample positions. The positional update threshold denotes the maximum absolute depth difference between clusters in subsequent

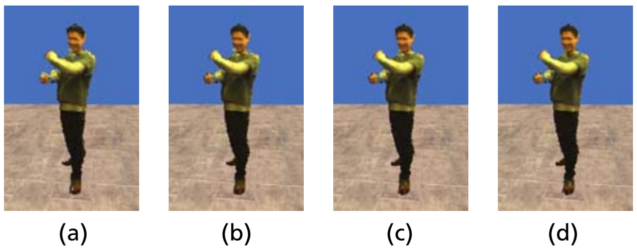

Figure 7: Rendered images in differential 3D video evaluating positional updates. a) threshold 0.04, b) threshold 0.02, c) threshold 0.01, d) threshold 0.005.

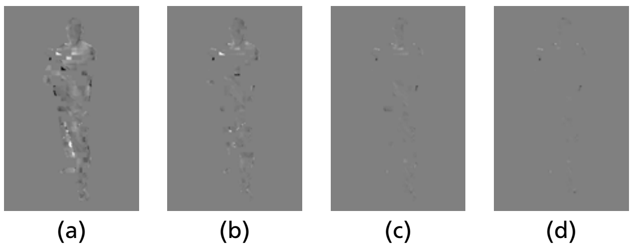

Figure 8: Difference images for positional updates evaluation comparing correct and differentially updated depth images, from a camera contributing to the view in Figure 7, magnified by a factor of 5. a) threshold 0.04, b) threshold 0.02, c) threshold 0.01, d) threshold 0.005.

frames. Because of the metric calibration of our system, a threshold of, e.g., 0.04 denotes a maximum depth difference of 4 cm.

Figure 7 shows rendered images with a varying positional update threshold. The difference images in Figure 8 illustrate the block artifacts in the object geometry which appear if the threshold is too high.

Table 2 and Figure 9 summarize the quality and performance measures of different positional updates configurations. We compare the correct and differentially updated depth images and average the result over all frames of the sequence. We see that the quality of the differential depth images is very high for all evaluated thresholds. But for thresholds over 0.02 on average more than half of the point samples get updated in each frame. This leads to a high bit rate in the real-time stream. Nevertheless, even with a threshold of 0.04 and an update of 25%-30% of the points per frame, a decent image quality can be achieved.

4. The blue-c communication architecture

The blue-c system relies on a CORBA-based communication layer. This allows for a straightforward use of remote method invocations for system control and data transmissions which are not performance critical. Furthermore, we use CORBA services such as the *Naming* and *Audio/Video Streaming Service* for connection management. The real-time streaming data, such as 3D video, audio and application data, is directly transmitted via IP sockets. For this purpose, we developed a mes-

Table 2: Results from positional updates in differential 3D video. The PSNR values and bit rates for different positional update thresholds compare correct and differentially updated depth images at 15 fps, from a camera contributing to the view in Figure 7.

UpdatePos threshold	PSNR [dB]	Total bit rate [Mbps]	Bit rate UpdatePos only [Mbps]
0.040	35.4	4.8	2.1
0.020	37.8	6.3	3.5
0.010	40.6	8.1	5.3
0.005	44.1	10.1	7.4

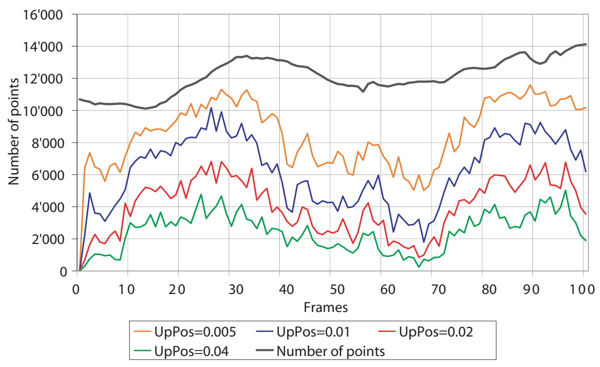

Figure 9: Number of positional updates (UpPos) in differential 3D video for various positional update thresholds. The data is extracted from a single camera contributing to the virtual viewpoint used in Figure 7.

sage streaming protocol based on UDP. Alternatively to the connectionless and unreliable UDP transmission, the message streaming protocol offers a reliable mode in which all messages are delivered without loss and in order.

Both modes include a forward and a backward channel, which is inspired by the RTP/RTCP protocol suite and which is extensively used for 3D video streaming [9]. The original RTCP packet format, however, was conceived to carry statistical data describing the transmission quality. It could be used to carry the steady and low latency feedback that we require for our 3D video system, but the overhead of using RTCP packets becomes pretty high. Thus, we decided to develop a special protocol for real-time streaming in the blue-c communication layer.

4.1. 3D video system

As suggested in Section 3.2, our 3D video system is composed of three classes of computing nodes for acquisition, reconstruction and rendering. Each camera has a dedicated node which performs 2D image processing operations, i.e. background segmentation, contour extraction and image differentiation. They run at the pace of an hardware camera trigger and our current system for combined projection and acquisition allows for acquisition rates of up to 10 frames per second for 640×480 images. The reconstruction node collects the data from the acquisition nodes and transforms it into a stream of 3D point operations. Those update the rendering data structure at the receiver side. The use of separate threads for networking and graphics at the renderer decouples the rendering frame rate from the reconstruction frame rate.

In order to generate a consistent 3D video stream, every operation issued by a camera node needs to be processed at the reconstruction node. In our implementation, we achieve this feature by running the reconstruction node asynchronously from the camera nodes, and thus we decouple the reconstruction frame rate from the acquisition frame rate. However, we need to deal with scheduling and synchronization issues at the reconstruction node, which are explained in the following paragraphs.

The reconstruction node collects data from all camera nodes. The camera data consists of contour information and texture data. The contour data per frame is described in a list of image coordinate vectors and the texture information describes pixel-based insert, delete or color update operations. A performance improvement of the CPU-consuming geometric computations can be achieved with a multi-threaded implementation of the reconstruction node. Hence, the data processing of the diverse camera nodes is distributed on separate threads. In fact, the reconstruction process needs the contours from many different cameras, but all texture processing is completely independent for each camera. Thus, the concurrent access to the contour data structure only needs to be protected during the write operation whenever a new contour arrives. Figure 10 illustrates the scheduling and multi-threading of camera and reconstruction nodes. It is important to notice that a consistent 3D video object representation and an efficient processing requires all delete operations from one camera client to be executed before the insert operations of the same. The camera clients support this operation flow by first transmitting contour data, then delete and update operations and, finally, insert operations.

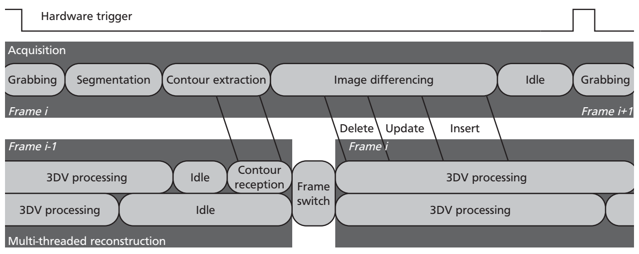

Figure 10: Scheduling and multi-threading of camera and reconstruction nodes.

Another critical section exists at the single communication channel from the reconstruction to the rendering node. For algorithmic reasons, it is important to maintain the order of the operations generated by the same camera, but interleaving the operations from different cameras is not critical. Hence, each processing thread of the reconstruction node has its own queue for storing 3D operations. Every time a new operation is pushed into the queue, the thread tries to get write access to the communication channel. On success, all operations contained in the respective queue are forwarded to the transmission channel. Thus, asynchronous access to the transmission channel for multiple threads is provided.

The camera weights, which are determined by the texture blending algorithm of the dynamic viewpoint adaptation [19], are used for load-balancing the contributing cameras over the multiple threads. More load is to be expected from a high camera weight. At every frame switch, the reconstruction node checks the camera weights and redistributes the processing load over the available threads. In our concrete system with a dual-processor reconstruction node and up to three active cameras, this simply means that the camera with the highest weight is processed by one thread and the two remaining cameras share the second thread.

Note that every camera node, even those which are not used for the current virtual viewpoint, transmits at least an empty set of contours for every frame. This strategy allows the reconstruction node to check if all clients are still synchronized. The acknowledgement message of contour data contains the new state information for the corresponding acquisition client. The reconstruction node detects a frame switch while receiving contour data of a new frame. At this time, state computations, which are only necessary once per frame, are triggered.

4.2. Audio/Video synchronization

In a 3D video conferencing system, we typically have multiple video acquisition nodes. The audio samples are thus not necessarily acquired by the same node than the corresponding video and a technique for synchronizing audio and video streams is required.

In our system, we can use the same strategy for consistent timestamping of audio and video packets than we use for synchronizing the camera frames on the multiple acquisition nodes.

For a consistent 3D representation, it is important to correctly associate the corresponding contours and textures of many different cameras. It is true that a hardware camera trigger guarantees a synchronized image grabbing, but since all cameras may not start up at exactly the same time, the frames still need to be synchronized by a unique distributed frame ID. If all camera nodes have their own frame counter, which is incremented every time an image is grabbed, the camera which was started first will have the highest counter value and hence provide the highest frame ID. The reconstruction node determines the highest frame ID for each frame and forwards this frame ID as the master frame ID to all camera clients, together with the camera state information. The camera clients check if they are synchronized with the master frame ID and adapt their own frame counter to the master frame ID. With this strategy, all cameras synchronize their frame IDs and hence an asynchronous camera start-up becomes possible.

Finally, if the audio acquisition process is integrated into the same camera acquisition loop, the audio packets can be marked with the same consistent timestamps than the video frames and thus also a synchronized audio/video playback becomes possible at the rendering site.

5. Application simulations

In this section, we analyze the overall performance of our communication system. For this purpose, we use the three main components, i.e. the real-time 3D video pipeline, the audio conferencing module and the distributed shared scene graph. We record the real-time performance of the system during an application simulation which is detailed in the following section.

5.1. Simulation setup

In the following experiments, we run the system in a local loop using the configuration of Figure 11. An SGI Onyx 3200 (hostname `pacific`) runs all receiver application processes and the `atlantic` Linux cluster runs all sender processes. The 3D video stream is generated using the full real-time setup, but the real camera image is replaced by the corresponding frame of a pre-recorded sequence. Thus, we can produce the same data for various experiments. The acquisition frame rate of the camera nodes is 10 fps. The dual processor host `atlantic2` is used as reconstruction node and computes the 3D video stream for a constant virtual viewpoint. A duplex speech transmission is simulated between the `pacific` and `atlantic3` nodes. The dual processor Linux node `atlantic1` acts as application host and simulates a duplex application data stream with `pacific`.

Our SGI Onyx 3200 is equipped with eight 400 MHz MIPS R12000 processors and 4MB of main memory and runs SGI Irix 6.5. `atlantic1` is a dual processor AMD AthlonMP 1600+ PC with 512kB of main memory and `atlantic2` is a dual processor AMD AthlonMP 2400+ PC with 1MB of main memory. `atlantic3` is an AMD AthlonXP 1600+ single processor PC with 512kB of main memory. The PCs run Red Hat 2.4 Linux.

The *Naming Service* and the *Connection Server* run on `atlantic2`. In the system adaptation experiments, `atlantic16` runs an application simulation from which all participating hosts retrieve the current application state via CORBA remote method invocations. `atlantic3` and `atlantic16` do not act as camera nodes in the 3D video pipeline.

The setup of Figure 11 is not really full-duplex, inasmuch as the 3D video transmission from Portal B to Portal A is omitted. Since only dedicated computing nodes are involved in the 3D video generation and since we report the results for the application node at Portal B, which runs the same processes than in the true full-duplex setup, we nevertheless think that our experiments describe the correct system performance and that our test configuration can be qualified as a duplex setup.

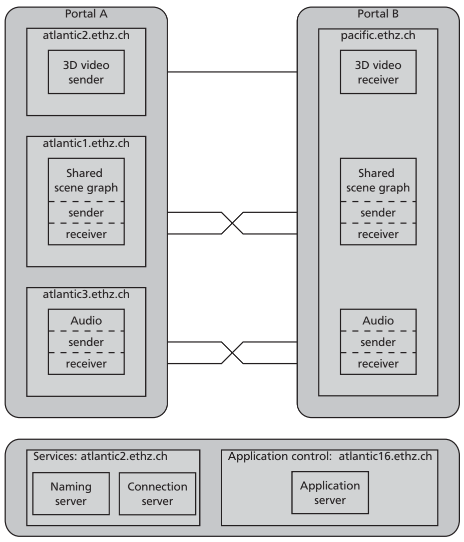

Figure 11: Experimental setup for a local loop system test.

In the following discussion, we analyze time intervals of length 0.5 seconds for the illustration of the bit rates. In order to distinguish long-term and short-term averages in the bit rate, we call the total average bit rate of the test sequences *mean bit rate* (MBR) and the short-term average, i.e. the mean bit rate on 0.5 second intervals, *peak bit rate* (PBR).

The application simulation for the system adaptation experiments demonstrates how the blue-c system can adapt to different situations during an application session and thus adapt to processing or networking bottlenecks. For each data stream, we foresee various states. The bit rate of the audio streams is controlled by the bit depth of the audio samples and by the sampling frequency. In our 3D video pipeline, we cannot change the camera input frequency, but the transmission bandwidth is reduced by lowering the resolution of the input cameras. The application updates are generated at 30 frames per second, the update messages having an average size of M bytes. For simulating a high user activity, we raise the average update message size to $3M$, but at the same time, we reduce the update rate to 15 frames per second. A low user activity is simulated with an average message size of M bytes, the update rate being 7 frames per second. During serious bottlenecks, we further reduce the application update rate to 1 frame per second. The bulk data transfer is implemented via continuous streaming of data using the same reliable protocol than for the application updates.

For the simulation of different loads and priorities of the involved data streams, we suggest the application scenario of Table 3. The timings in Table 3 are indicative and multiple state changes are stretched over an interval of several seconds.

Table 3: Application scenario.

Time	Audio	3D video	App. data	Bulk data	Description
0 s	high	high	normal	off	Application start-up.
10 s	normal	normal	off	on	A bulk data transfer starts.
20 s	high	normal	high	off	The bulk data transfer ends and is followed by a period of high user interaction.
35 s	high	high	normal	off	Normal user interaction.
50 s 52 s	high	normal high	high normal	off	Short period of high user interaction.
60 s 70 s	normal low	normal low	low overload	off	Simulation of a transmission bottleneck.
85 s 90 s	normal high	normal high	normal	off	Recovering from the transmission bottleneck.

5.2. Lossless local loop simulation

Figure 12 reports the results of the application simulation of Table 3, which ran in the local loop setup of Figure 11. From the curve describing the point resolution of the 3D video object, the different application periods, i.e. states of the 3D video transmission, can clearly be recognized. During the HIGH periods, the resolution is higher than 30k points. It drops down to 20k points during the NORMAL periods, and to 10k points during the LOW period. The peak bit rate of the 3D video stream is however not noticeably smaller during the HIGH and NORMAL periods, but during the LOW period the peak bit rate remains lower than 1 Mbps.

In presence of sharp changes in the point resolution, high peak bit rates appear in the 3D video stream. This seems to be reasonable in case of a point resolution increase, since the new point samples need to be inserted into the remote data structure. But the peaks pose a major problem for point resolution reduction. In case a downgrading of the point resolution is deployed as a reaction to transmission bottlenecks, this strategy first requires a high peak bit rate to face bandwidth problems. In fact, the peak bit rate results from the deletion of single point samples in our current implementation. The extension of the 3D video framework with a DELETE operation, allowing for the removal of groups of points, i.e. all points from a given input camera at a given sampling level, could solve this problem. The additional DELETE operator could also be deployed for fading out points during camera hand-overs where a similar behavior of the bit rate is observed.

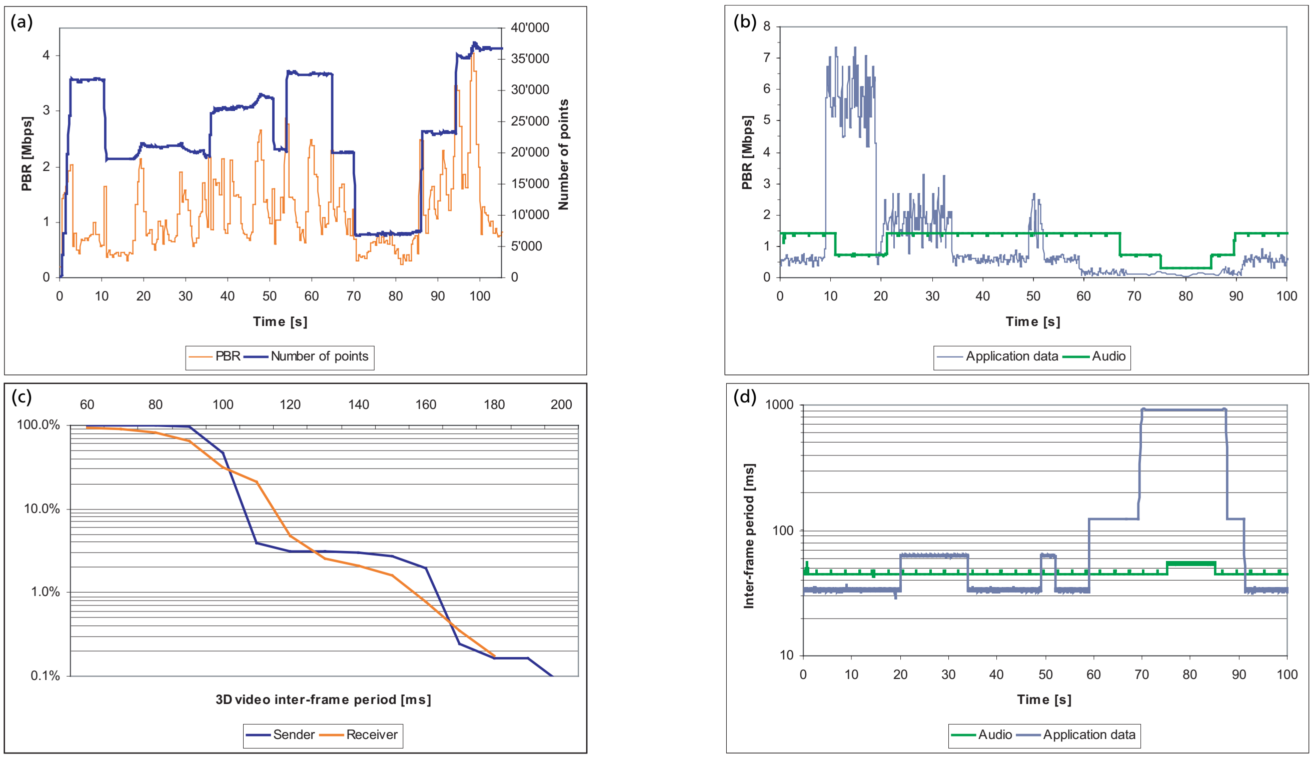

Figure 12: Local loop simulation at RZ: a) Number of points and peak bit rate of the 3D video stream; b) Peak bit rate of the application data and the audio stream; c) Cumulative probability distribution of the inter-frame period of the 3D video stream; d) Inter-frame period of the audio and application data streams.

Figure 12b shows the bit rate of the application data (application updates and bulk data transfer) and audio streams. They correspond to the behavior expected from the application simulation.

Figure 12c depicts the cumulative probability distribution of the inter-frame periods at the 3D video sender and receiver. We observe that the inter-frame period at the sender corresponds to the expected 100 ms over 90% of the time. But, also higher inter-frame periods, often around 160 ms exist. These correspond to the frames which need more extensive computations at the 3D video reconstruction node and which exceed the allocated time. The dynamic system control of the 3D video pipeline however is able to handle this situation and still achieves an average inter-frame period of 100ms. In Figure 12d, the inter-frame periods of the audio and application data streams are shown. Application and audio data streams show almost constant behavior within the respective state intervals. Note that several successive audio samples are grouped into one transmission buffer, and that we display in Figure 12d the time periods in between two buffer arrivals. We configured the audio transmission such that the sender frequency of the transmission buffers is almost constant. For the different audio modes, this means that a different number of audio samples is contained in the transmission buffers.

5.3. Lossy local loop simulation

In this experiment, we repeat the application simulation of Section 5.2 while simulating burst errors on all communication channels. We use the Gilbert-Elliott channel model with $p_{\mathrm{PASS}} = 0.99$ and $p_{\mathrm{FAIL}} = 0.50$.

While comparing Figure 12a and Figure 13a, we observe that the resolution of the 3D video object during a lossy transmission is not only influenced by the application simulation but also by the error bursts. Moreover, recovering from the sporadic errors requires again peaks in the bit rate, i.e. for reinserting the point samples of the higher sampling levels into the data structure. Figure 13c shows the behavior of the 3D video stream for the same application and network simulation while using the redundant mode of the 3D video streaming pipeline. In this case, the mean bit rate is higher, 4 Mbps versus 2.4 Mbps, because of the redundant information contained in the stream. Also the resolution of the 3D video object is not constant during periods of no or short packet losses and does not achieve the same maxima than in the reliable, but non-redundant mode. However, also the drop-off in resolution is less pronounced during the periods of error bursts.

The application data transmission ran with the same configuration than in the local loop simulation of Section 5.2. The effect of the noisy communication channels appears in the more noisy inter-frame periods, see

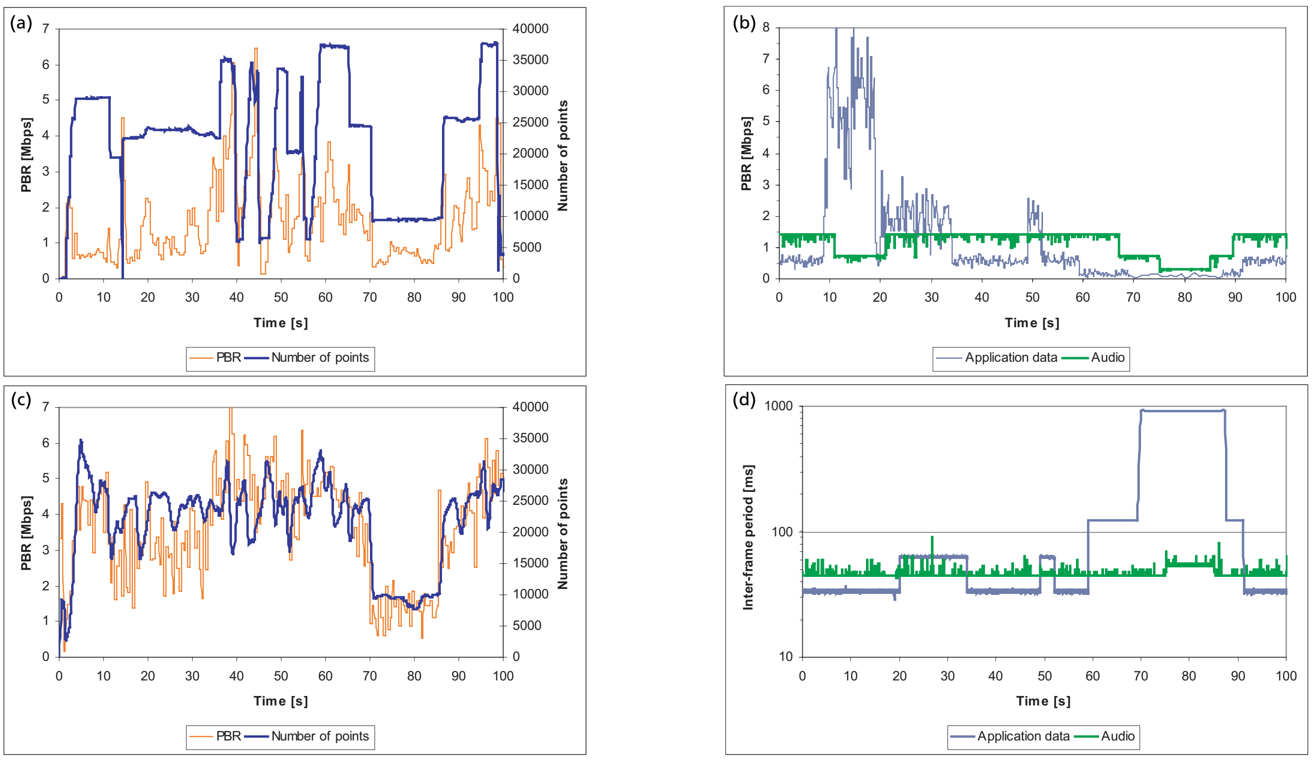

Figure 13: Local loop simulation with simulated burst losses: a) Number of points and peak bit rate of the 3D video stream in the reliable mode; b) Peak bit rate of the application data and the audio stream; c) Number of points and peak bit rate of the 3D video stream in the redundant mode; d) Inter-frame period of the application data and the audio stream.

Figure 13d. For the audio stream, we did not perform retransmissions of lost packets and hence the missing data packets produce audible noise during playback. These artefacts can however be partly avoided by redundant encoding of the audio data or by interpolations during playback. In Figure 13b, the lost audio packets are visible in the bit rate drop-offs of the audio stream.

5.4. Summary

From the experiments of this section, we conclude that our system behaves correctly in a controlled load environment. If the application foresees discrete operating states, the various data streams of the blue-c system can be configured such that they meet the targets of the application.

The results from Section 5.3 confirm that the redundant 3D video streaming mode outperforms the non-redundant mode if the communication channel is lossy. For the application data however, we still rely on a reliable transmission, and methods for a redundant encoding of the application data should be investigated in the future. Also the audio transmission can be improved by using packet loss recovery strategies.

Our two blue-c prototypes are located at 5 km from each other and are connected via the ETH backbone network, operating in controlled load. In this setup, the network bandwidth is abundant and delays are short, i.e. the transmission conditions are less severe than those simulated in Section 5.3. As we demonstrate through the experiments of this section, the blue-c system is however versatile enough to operate in different conditions. Video clips which illustrate the blue-c system can be downloaded from our project webpages blue-c.ethz.ch and graphics.ethz.ch/3dvideo.

6. Conclusions and outlook

From the discussion of Section 3, we conclude that for telepresence applications a three-stage pipeline for real-time 3D video presents a reasonable trade-off between system latency, processing power and practical implementation issues. The system experiments of Section 5 validate our system architecture and indicate how a telepresence system which foresees discrete operating states can adapt to network and processing bottlenecks.

The extension of the blue-c platform from a one-to-one to a many-to-many system raises some issues of scalability. 3D video multicasting can be implemented by the unreliable transmission of a redundantly encoded 3D video stream, combined with a modification of the dynamic viewpoint adaptation algorithm. A solution which does not require reliable multicasting of the attribute-centric application updates has yet to be devised. Finally, we would like to propose automatic application adaptation based on the system control data and monitoring at the network level.

Acknowledgements

We would like to thank Martin Näf for the distributed shared scene graph; Silke Lang and Doo-Young Kwon for the 3D video test sequences; Michael Waschbüsch for his help with the simulation environment; Thomas Gross for the fruitful discussions; and all blue-c project members. The blue-c project was funded by ETH grant No. 0-23803-00 as an internal poly-project.

References

[1] F. P. Brooks. What's real about virtual reality. *IEEE Computer Graphics and Applications*, 19(6):16–27, 1999.

[2] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Proceedings of SIGGRAPH*, pages 135–142, 1993.

[3] S. J. Gibbs, C. Arapis, and C. J. Breiteneder. TELEPORT - towards immersive copresence. *Multimedia Systems*, 7(3):214–221, 1999.

[4] C. M. Greenhalgh and S. D. Benford. Massive: A virtual reality system for teleconferencing. *ACM Transactions on Computer Human Interfaces*, 2(3):239–261, 1995.

[5] M. Gross, S. Wuermlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. V. Gool, S. Lang, K. Strehlke, A. V. Moere, and O. Staadt. blue-c: A spatially immersive display and 3D video portal for telepresence. In *ACM Transactions on Graphics, Proceedings of SIGGRAPH 2003 (Conference Issue)*, pages 819–827. ACM Press / ACM SIGGRAPH, July 2003.

[6] T. Kanade, P. W. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. In *IEEE MultiMedia*, volume 4, pages 43–54, January-March 1997.

[7] P. Kauff and O. Schreer. An immersive 3D videoconferencing system using shared virtual team user environments. In *Proceedings of the 4th international conference on Collaborative virtual environments*, pages 105–112. ACM Press, 2002.

[8] N. Kitawaki and K. Itoh. Pure delay effects on speech quality in telecommunications. *IEEE Journal on Selected Areas in Communications*, 9(4):586–593, May 1991.

[9] E. Lamboray, S. Wuermlin, and M. Gross. Realtime streaming of point-based 3D video. In *Proceedings of the IEEE Virtual Reality 2004 conference*, pages 91–98. IEEE Computer Society Press, March 2004.

[10] J. Leigh, O. Yu, D. Schonfeld, R. Ansari, E. He, A. Nayak, J. Ge, N. Krishnaprasad, K. Park, Y. joo Cho, L. Hu, R. Fang, A. Verlo, L. Winkler, and T. A. DeFanti. Adaptive networking for tele-immersion. In *Proceedings of the Immersive Projection Technology/Eurographics Virtual Environments Workshop (IPT/EGVE)*, pages 199–208, Mai 2001.

[11] M. R. Macedonia, M. J. Zyda, D. R. Pratt, P. T. Barham, and S. Zeswitz. NPSNET: A network software architecture for large scale virtual environments. *Presence*, 3(4), Fall 1994.

[12] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH 2000 Conference Proceedings*, ACM Siggraph Annual Conference Series, pages 369–374, 2000.

[13] M. Naef, E. Lamboray, O. Staadt, and M. Gross. The blue-c distributed scene graph. In J. Deisinger and A. Kunz, editors, *Proceedings of IPT/EGVE 2003*, pages 125–133. ACM Press, Mai 2003.

[14] K. S. Park and R. V. Kenyon. Effects of network characteristics on human performance in a collaborative virtual environment. In *Proceedings of the IEEE Virtual Reality 1999 conference*, pages 104–111, 1999.

[15] S. Prince, A. D. Cheok, F. Farbiz, T. Williamson, N. Johnson, M. Billinghurst, and H. Kato. 3-D Live: Real time interaction for mixed reality. In *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pages 364–371. ACM Press, 2002.

[16] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: a unified approach to image-based modeling and spatially immersive displays. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 179–188. ACM Press, 1998.

[17] A. Sadagic, H. Towles, J. Lanier, H. Fuchs, A. van Dam, K. Daniilidis, J. Mulligan, L. Holden, and B. Zeleznik. National tele-immersion initiative: Towards compelling tele-immersive collaborative environments. presentation given at Medicine meets Virtual Reality 2001 conference, January 2001.

[18] I. Vaghi, C. Greenhalgh, and S. Benford. Coping with inconsistency due to network delays in collaborative virtual environments. In *Proceedings of the ACM symposium on Virtual reality software and technology*, December 1999.

[19] S. Wuermlin, E. Lamboray, and M. Gross. 3D video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics, Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data*, 28(1), 2004.