Real-Time Streaming of Point-Based 3D Video

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

Abstract

Free-viewpoint video is a promising technology for next-generation virtual and augmented reality applications. Our goal is to enhance collaborative VR applications with 3D video-conferencing features. In this paper, we propose a 3D video streaming technique which can be deployed in telepresence environments. The streaming characteristics of real-time 3D video sequences are investigated under various system and networking conditions. We introduce several encoding techniques and analyze their behavior with respect to resolution, bandwidth and inter-frame jitter. Our 3D video pipeline uses point samples as basic primitives and is fully integrated with a communication framework handling acknowledgment information for reliable network transmissions and application control data. The 3D video reconstruction process dynamically adapts to processing and networking bottlenecks. Our results show that a reliable transmission of our pixel-based differential prediction encoding leads to the best performance in terms of bandwidth, but is also quite sensitive to packet losses. A redundantly encoded stream achieves better results in presence of burst losses and seamlessly adapts to varying network throughput.

1. Introduction

In recent years, there has been an increasing interest in generating free-viewpoint video sequences from multiple camera views. Apart from purely image-based approaches [11, 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. Today, the robust generation and transmission of real-time 3D video is still a challenging problem. Figure 1 shows exemplary frames generated in real-time by our 3D video system.

Our research on real-time 3D video systems is motivated by our interest in novel immersive projection and acquisition environments for telepresence [7]. At ETH, we developed the blue-c, two networked virtual reality portals consisting of a CAVE-like environment, augmented by an array of cameras and an active lighting system. Thus, blue-c combines the simultaneous acquisition of multiple video streams with advanced 3D projection technology.

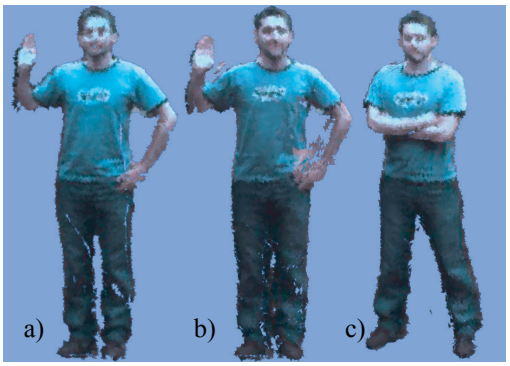

Figure 1: Three examples from a 3D video sequence: a) and c) result from a reliable transmission to the rendering node, b) is the result after a lossy transmission of a redundantly encoded stream.

Both portals can be used for 3D video acquisition and, as depicted in Figure 2, in full operation mode they enable networked collaborative applications enhanced by 3D video-conferencing features.

Even though most efforts in rendering and compression of 3D data focus on meshes, we opted for point samples as the basic primitive in our 3D video representation. In our opinion, point samples can be considered as a straightforward generalization of 2D video pixels into 3D space [27]. Furthermore, a general approach handling dynamic objects must allow for topology changes, and a topology change on a 3D mesh is a costly operation and hard to achieve in real-time. As demonstrated in Figure 1, already the simple arm movements from Figure 1a to Figure 1c lead to a topology change in the 3D data set representing the human person.

In particular, our 3D video system is composed of 16 camera nodes, which acquire images and perform 2D image processing. The resulting information is streamed to a reconstruction node, which computes the actual 3D representation of the observed object. Camera and reconstruction nodes are at the same physical location and are connected in a local area network. The 3D video data is then streamed to a rendering node, which, in a real-world telepresence application, runs at a remote location. As explained in Section 2, rendering and reconstruction nodes need to share a common data structure, and, depending on

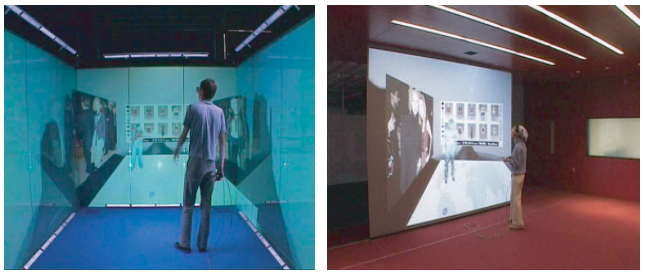

Figure 2: The two blue-c portals in action.

the 3D video streaming procedure, this data structure must satisfy different consistency requirements. The overall system architecture is depicted in Figure 3.

The main contribution of this paper consists in proposing a communication framework for distributed real-time 3D video reconstruction and rendering and in analyzing the transmission of the subsequent streams with respect to changing networking conditions.

After a short discussion of related work, Section 2 summarizes the components of our 3D video system and introduces the supported operation modes. Section 3 describes the communication software. Section 4 presents performance characteristics of the system with respect to different networking conditions.

1.1. Related work

3D video representations and systems can still be considered as an emerging technology. In the past, several systems for real-time and off-line reconstruction of dynamic objects have been developed [3, 15, 16, 19, 26, 28], but the encoding and compression of the resulting 3D video data remains largely unexplored. In fact, most systems reconstruct and render the object on the same node. Such a setup, however, does not correspond to the situation one encounters in telepresence or video-conferencing systems.

In [27], we propose a point-based system for real-time 3D reconstruction, streaming and rendering which does not make any assumptions about the shape of the reconstructed object. Hence, we pursue a more general approach to the problem of 3D video than in [3], and, unlike approaches based on animated 3D meshes [9], we are able to handle changes in the topology of the reconstructed object.

In the past, many efficient compression algorithms for meshes have been developed [8, 20], whereas compression for point representations is still in the fledglings stage. Rusinkiewicz and Levoy presented Streaming QSplat [22], a view-dependent progressive transmission technique for a multi-resolution rendering system, which is based on a hierarchical bounding sphere data structure and splat rendering [21]. In [1], Botsch et al. use an octree data structure for storing point sampled geometry and they show that typical data sets can be encoded with less than 5 bits per point. Lee et al. developed a progressive encoding scheme for isosurfaces using an adaptive octree and fine level placement of surface samples [13]. They achieve similar performance for coding connectivity and geometry information. However, the above techniques efficiently encode large but static data sets.

Briceno et al. propose to reorganize the data from dynamic 3D objects into 2D images [2]. This representation allows for high compression rates using standard video coding techniques, but the transformation from 3D to 2D space appears to be too complex for real-time applications.

2. 3D video pipeline

A point-based representation describes a 3D object as a set of point samples where each sample has a collection of attributes, e.g. position, color, and surface normal vector. From a general point of view, a point data structure can be dynamically updated over time by the following operators: INSERT adds a new point sample into the representation; UPDATE changes one or several attributes of an existing point sample; DELETE erases a point sample from the representation.

2.1. 3D video processing

Figure 4 depicts the processing steps of our 3D video system. *N* previously calibrated cameras grab images of the same scene from different angles. A hardware trigger guarantees consistent image acquisition. In each camera image, the static background is subtracted first. The silhouettes covering regions of connected foreground pixels are determined. We limit the number of contour edges and improve the speed of the reconstruction algorithm by using piecewise linear contour segments [28]. In our current implementation, we use a pixel-based differential update scheme, which exploits the spatio-temporal inter-frame coherence, i.e. pixels which change from background to foreground are interpreted as insert operations and already inserted pixels are analyzed with respect to color changes. Currently, we use 16 cameras and each camera has a dedicated host for image acquisition and the previously described 2D image processing. Note that the texture is scanned according to a linear pixel sampling pattern, which also allows for load balancing in case of performance or transmission bottlenecks.

The reconstruction process transforms the 2D pixels into 3D point samples using the geometry information provided by the silhouettes. We use a variant of the image-based visual hull algorithm [16]. 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. In particular, we distinguish between UPDATECOLOR and UPDATEPOSITION operations. A detailed description of the differential update scheme can be found in [27].

At the remote site, the point sample operators update a data structure which is used for rendering the 3D object to screen. If we exploit the spatio-temporal inter-frame coherence, the data structures at the acquisition and at the rendering site need to be consistent. As explained in Section 2.3, some strategies may require a totally consistent distributed data structure (hard synchronization), other strategies content themselves with a soft synchronization, which guarantees consistency over time, but not at every single time instant.

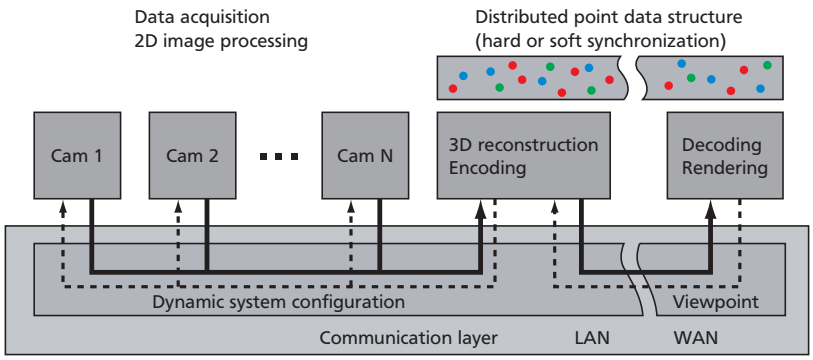

Figure 3: 3D video system architecture.

2.2. Dynamic system control

For performance and algorithmic reasons it is reasonable to take into account viewpoint feedback during the computation of free-viewpoint video. The framework proposed by the MPEG committee for 3D video suggests a backward channel for viewpoint transmission [25].

In our system, the rendering node transmits the current viewpoint to the reconstruction node which then computes a new system configuration for the next frame. The system configuration describes which cameras need to provide silhouette and texture information for an optimal 3D shape given the current viewpoint. Furthermore, taking into account performance measures from the reconstruction process of the previous frames and monitoring the quality of the 3D video transmission, the 3D video frame rate can be improved by downsampling the texture information using the image sampling pattern. Thus, reducing the number of 3D video operators allows to overcome networking bottlenecks.

2.3. Operation modes

The popular MPEG video compression standard defines three different types of video frames: I-frames are coded without any references from past frames, they are self-contained; P-frames use motion-compensated prediction from the past I- or P-frame; B-frames use bidirectional prediction from the most recent and the closest future I- or P-frame [23].

Note that the predicted P- and B-frames achieve the highest compression rates, but a sequence encoded exclusively with prediction frames can only be rendered correctly if the receiver retains the complete data stream. Problems may occur in many situations, e.g. parts of the data stream are lost during network transmission or the sender and the receiver are started asynchronously. Moreover, prediction errors accumulate over time. In the following, we use the nomenclature from 2D video coding for describing three different modes of our 3D video pipeline.

I-mode (Full-Reconstruction). The 3D object is completely reconstructed in each frame, no information from previous frames is used. Hence, the complete 3D data needs to be recomputed and transmitted. The resulting stream is composed of I-frames only and is highly redundant.

P-mode (Reliable-Prediction). This setup keeps track of all the previous frames and only computes and transmits changes in geometry and color. In this case, reconstruction and rendering nodes need to share a totally consistent data representation and hence a reliable data transmission is required (hard synchronization). Speaking in MPEG terms, the data stream consists exclusively of P-frames.

R-mode (Redundant-Prediction). This setup too, exploits differential information over several input frames, but also reconstructs parts of the 3D representation at regular intervals, such that the complete 3D data is regularly recomputed over time. In this case, no reliable transmission is required and errors due to an inconsistent shared data representation between reconstruction and rendering nodes only occur temporarily (soft synchronization).

The experimental results of Section 4.3 suggest that, in real-time, only a limited number of point samples can be computed per frame. A traditional use of I- and P-frames is thus not recommendable. The R-mode circumvents this problem by equally distributing the computation and transmission of redundant data over several frames.

Hence, a characteristic parameter of the 3D video pipeline is the recomputation frequency f_g of the geometry information. In the I-mode, f_g is identical to the acquisition frame rate. Apart from the aperiodic geometry recomputations triggered by the spatial consistency conditions of the prediction scheme [27], f_g is a user-defined parameter in the R-mode and equals to infinity in the P-mode.

Further note that only the P-mode requires a reliable transmission between the reconstruction and the rendering node. The subsequent quality and performance trade-offs will be discussed in Section 4. Finally, we do not exploit prediction based on future frames because we want to minimize the latency of the real-time streaming system in teleconferencing applications.

3. Communication framework

The distributed virtual reality platform of the blue-c uses a communication framework supporting all data types which occur in a networked virtual environment. For the 3D video subsystem in particular, an efficient real-time streaming scheme is essential.

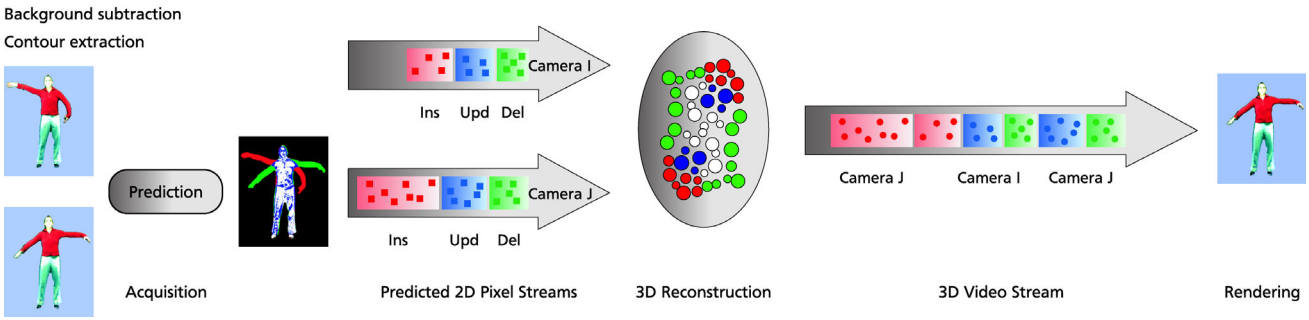

Figure 4: Overview of the 3D video processing steps.

3.1. Data transmission

The communication layer API offers channel interfaces for transmitting and receiving data. The shared application data is organized in message objects which provide an interface for writing their payload into a transmission buffer. At the receiver, the transmission buffer is parsed, the messages are decoded and a callback mechanism informs the application of the arrival of a new message of a given type. Flow control and retransmission algorithms are applied to the transmission buffers only. A single transmission buffer may contain a set of messages of the same type or a collection of messages of different types. If the message size is greater than the size of a transmission buffer, a single message can be fragmented and distributed in multiple transmission buffers. All buffers are handled by memory pools such that all memory allocation is performed during start-up. The communication software also handles issues of cross-platform interoperability, e.g. little-/big-endian conversions.

Real-time streaming data is transmitted by a communication channel consisting of a forward channel for payload data and a backward channel for control information and application feedback. This approach is inspired by the RTP/RTCP protocol suite, where RTP is used for unreliable streaming of payload data and RTCP is used for periodical exchange of control messages, indicating packet loss rates, packet jitter values and possibly application specific data [24].

The communication framework also offers a collection of codecs. Typically the data is entropy encoded before network transmission. We use a computation efficient variant of arithmetic coding for this purpose [14].

3.2. Reliable streaming

As suggested in Section 2.3, the P-mode requires a loss-free, in-order data transmission. The conventional TCP protocol offers this functionality, however, its deployment in real-time systems is critical. Its retransmission and congestion avoidance algorithm achieves reasonable performance in best effort networks for data transmissions which are not critical with respect to latency or jitter. If a packet loss is detected, TCP reduces the data rate at the network level. However, this behavior inhibits its use in real-time applications if the rate of data generated by the application is not reduced accordingly. Hence, an efficient communication pipeline must allow for coordinated flow control at the network level and adaptation at the application level [4]. Note that packet loss does not only occur in saturated network links or error-prone wireless channels. In case of an asymmetric setup between sender and receiver, i.e. a slow receiver communicating with a faster sender, packet losses occur because the receiving host is not capable of processing the data fast enough. Also in this situation it makes sense to use adaptation schemes at the application level which allow to minimize packet loss and maximize end-to-end data throughput. In recent years it has been shown that TCP can be outperformed by more elaborate retransmission schemes which are implemented using the unreliable UDP protocol. Keshav and Morgan propose the use of selective retransmission with packet-pair flow control and achieve goodput values of 90% in overloaded networks [12]. Reliable Blast UDP, which uses UDP for data transmission and TCP for signalling and aggregated acknowledgements, has been proposed as a technique for bulk data transfers in high-speed or dedicated networks [10].

The reliable real-time data streaming in our system is implemented using selective retransmission. Since the dynamic configuration features at the application level of our system need low latency state information from the receiver, we decided to emit backward channel messages on a packet-per-packet basis. Each backward channel message carries either a cumulative acknowledgement or the sequence number of the first missing packet. In presence of a negative acknowledgement, the missing packet is retransmitted. Similar to [12], the detection of a lost packet does not lead to the complete retransmission of the sending window. As suggested in [10], the feedback channel could easily be augmented by a bitmap tally of missing packets.

3.3. Service API

Our communication framework also includes a number of services which are based on the CORBA standard for distributed object computing [5]. The associated *Naming Service* is used for locating distributed objects. The *Time Service* offers synchronized timestamps across the network. The *Notification Service* can be used for distributing sporadic events across interested nodes. The connection management of the communication channels is based on the CORBA *Audio/Video Streaming Service*. Our imple-

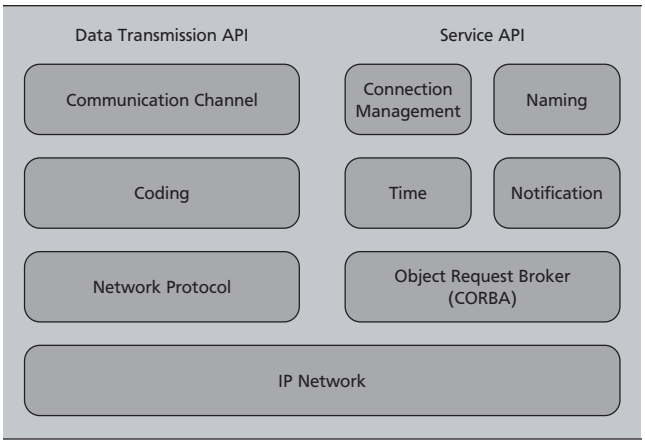

Figure 5: Communication framework architecture.

mentation is based on the TAO/ACE toolkit (http://www.cs.wustl.edu/~schmidt/TAO.html). The communication framework architecture is depicted in Figure 5.

3.4. The 3D video system

Figure 6 shows how the 3D video system uses various services offered by the communication framework. The camera nodes run a daemon which allows for the remote start-up and shut-down of applications. The reconstruction node identifies the camera nodes using the *Naming Service* and remotely starts the acquisition applications. The transmission channels between camera and reconstruction nodes are configured using the *Connection Management Service*. This service also keeps track of the state of the connections, and hence allows to deal with camera node failures. The pre-processed data from the camera nodes, i.e. silhouette and texture information, is streamed over reliable transmission channels. The associated backward channels carry the dynamic system configuration which is determined for each frame at the reconstruction node. Furthermore, the distribution of a global frame ID enables the synchronization of the running 3D video process with late joining camera nodes.

4. Streaming experiments

4.1. Experimental setup

During our streaming experiments, we simulate various networking conditions for different operation modes of our 3D video pipeline. We recorded a synchronized input sequence and stored each frame as JPEG image. During a test run, we synchronize the camera nodes using the hardware trigger, but overwrite the camera image with the data from the pre-recorded sequence. This procedure allows for a close simulation of the dynamic behavior of the real-time 3D video system, but also allows to use identical input data for all experiments. If not stated, the experiments run with an acquisition frame rate of 8 frames per second.

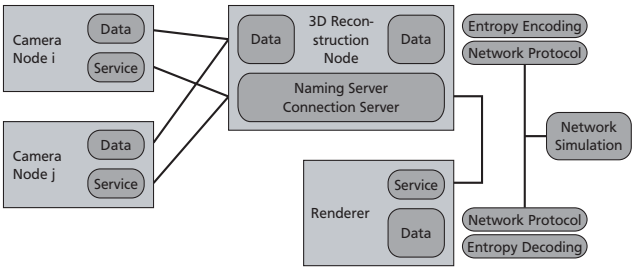

Figure 6: Services for real-time 3D video.

In order to limit the number of degrees of freedom in the experimental setup, we run all experiments for the same static viewpoint. Using the load balancing and camera blending algorithm described in [27], the selected viewpoint requires texture information at 100% and 70% from two cameras respectively. Dynamic viewpoint changes naturally lead to an increase of network traffic and to a lower resolution at the rendering node.

Our multi-threaded image based visual hull implementation runs on a dual processor machine with two AMD AthlonMP 2400+ CPUs. The rendering node is a 1.8 Ghz Pentium4 machine equipped with an NVIDIA GeForce4 Ti200 graphics accelerator. All nodes are interconnected in a Fast Ethernet local area network at 100 Mbps.

Note that we simulate in our experiments only packet losses for packets carrying 3D video data, i.e. packets that are transmitted from the reconstruction node to the rendering node. No loss is simulated for control messages carrying acknowledgement and system data. The consequences of missing backward channel messages are twofold: the perceived system delay increases, since the difference between the local viewpoint and the computed representation increases, and the retransmission of lost data packets is potentially delayed. The overall system functionality however is not impaired in case of missing feedback messages. Also no losses are simulated in between camera and reconstruction nodes, since these nodes share a dedicated local area network.

4.2. Quality assessment

The community of free-viewpoint and 3D video research is still in lack of an overall framework for quality assessment. A first step has been made by the MPEG-3DAV subgroup in its report on exploration experiments [18]. However, no approach reflecting all possible errors which might occur during the reconstruction, coding and transmission of a 3D video stream has yet been proposed. Ideally the error metric distinguishes between reconstruction and coding errors and thus allows for a fair comparison of reconstruction algorithms and encoding schemes.

In this paper, we compare the operation modes of our system with respect to the average number of points they maintain in the remote 3D data structure. Of course, this argument does not take into account the accuracy of the 3D points, but gives a reasonable first approximation. Furthermore, our experience has shown that the absolute numbers

for resolution and bandwidth are quite dependent on the test scenarios, i.e. viewpoint changes and motion of the person to be reconstructed.

In order to compare the coding efficiency of the different representations, we compare the average number of bytes required for coding one point sample. The coding efficiency C can be computed as

$$C = b/(n \cdot f),$$

where b is the average bandwidth in megabytes per second, n the average number of points per frame and f the frame rate.

A lossy transmission inevitably leads to artifacts in the rendered 3D video object, see Figure 9. Within our framework of INSERT, UPDATE and DELETE operations, the artifacts can be classified into different patterns. Missing UPDATE or DELETE operators lead to ghosting artifacts around the person or incorrect texturing of the point samples. Floating points can be eliminated in the rendering process by an outlier detection which can be calculated without much effort during dynamic point density estimation [27]. Visible artifacts due to incorrect texturing can be reduced by blending the point samples during splat rendering. Holes in the representation due to missing INSERT operations can also be alleviated to some extent by the point density estimation, but big holes as shown in Figure 9c cannot be completely eliminated in real-time.

4.3. Lossless network

In the first experiment, we did not simulate any packet losses, and hence all bottlenecks reside on the reconstruction and rendering nodes. The curves in Figure 10 show cumulative probability distributions, where $P(x>X)$ expresses the probability of x being larger than X. We see that the extensive geometry computations required by the I-mode considerably limit the number of points per frame, i.e. the resolution of the reconstructed object is rather low. On average, the P-mode achieves a better resolution than the R-modes, which can be explained by the fact that the P-mode does not perform any redundant geometry computations. On average, the bandwidth requirements for the R-modes are three times as high as for the P-mode. However, the R-mode allows to trade-off bandwidth against reconstruction accuracy by decreasing f_g. In this example, f_g^{-1} equals 1 and 1.5 seconds respectively. Also note that the P-mode is especially favorable if the viewpoint remains static. Table 1 summarizes the results of this experiment.

4.4. Bandwidth-limited network

We simulate a bandwidth-limited network using a constrained packet rate channel model, i.e. the throughput of the transmission channel is limited by a deterministic service rate λ. This simple model simulates e.g. a network router connected to a low data rate link. A network with varying throughput can easily be simulated using a Markov-modulated rate process. In this case, the current service rate λ_i depends on the current state i in the Markov chain, see Figure 7. We used $\lambda_i \in \{2, 0.75, 1\}$ for the varying throughput experiment A and

Table 1: Characteristic parameters for the test sequence with no simulated packet loss. R^1 used f_g=1 Hz and $R^{1.5}$ used f_g=0.6 Hz. The computation intensive I-mode was also tested at 3 frames per second.

		Operation Modes @ 8 fps			
	I@3fps	I	P	R^1	$R^{1.5}$
Average resolution [points per frame]	12734	4012	25922	15157	17748
Average bandwidth [megabit per second]	3.3	2.9	1.0	3.5	3.4
Peak bandwidth [megabit per second]	4.3	4.6	5.5	5.5	6.1
Coding efficiency [bytes per point]	10.8	11.3	0.6	3.6	3.0

$\lambda_i \in \{1.5, 0.75, 1\}$ in experiment B. All service rates are indicated in megabit per second. Furthermore, the transition probabilities $p_{0,1}$ and $p_{1,0}$ were higher in experiment B than in experiment A. Hence, experiment B simulated bandwidth variations at a higher frequency than experiment A. We did not simulate lower throughput than 0.75 megabit per second, since this appears to be the minimum required for our current 3D video representation at 8 frames per second.

The results in Figure 11 show that both P- and R-modes are capable of dealing with the simulated situation. However, the R-mode considerably reduces the inter-frame jitter at the times the network varies from a high to a low throughput. We also emphasize that the considerable difference in the resolution of the object, which is observable in the lossless transmission experiment between P- and R-modes, has vanished. The characteristic parameters of the test sequence are summarized in Table 2. The relatively high standard deviation of the inter-frame period indicates the irregular frame updates in the P-mode.

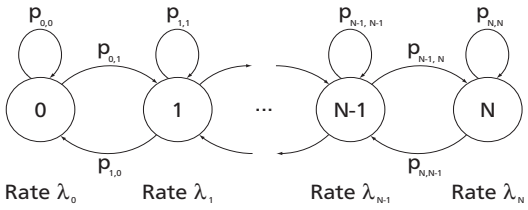

Figure 7: Markov-modulated rate process.

Table 2: Characteristic parameters for the test sequence with varying network throughput.

	Experiment A		Experiment B	
	P	R^1	P	R^1
Average resolution [points per frame]	16472	13535	11769	13272
Average inter-frame period [ms]	125	125	123	124
Standard deviation of the inter-frame period [ms]	159	53	149	51
Maximum inter-frame period [ms]	3023	381	2308	385

Table 3: Characteristic parameters for the test sequence with burst losses.

	Burst(95, 25)		Burst(98, 80)		Burst(99, 50)		Burst(99, 80)	
	P	R^1	P	R^1	P	R^1	P	R^1
Average resolution [points per frame]	10286	15095	10644	15292	15315	15381	12393	15411
Average inter-frame period [ms]	124	127	129	40	125	127	125	128
Standard deviation of the inter-frame period [ms]	162	53	212	50	119	50	157	59
Maximum inter-frame period [ms]	3365	806	2849	371	1589	346	1998	360

4.5. Burst losses

We simulate burst losses using a two-state Markov chain, see Figure 8. The average number of iterations in the two states is $1/(1-p)$ respectively. Table 4 gives an overview of the four experimental scenarios.

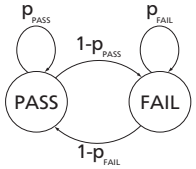

Figure 8: Two-state Markov chain modelling a Gilbert-Elliot packet erasure channel [6].

Table 4: Parameters of the burst loss experiments.

Experiment	Average number of packets		Description
	PASS	FAIL	
Burst(95, 25)	20	1.3	Numerous but short burst losses.
Burst(98, 80)	50	5	Less frequent but longer burst losses.
Burst(99, 50)	100	2	Occasional but short burst losses.
Burst(99, 80)	100	5	Occasional but longer burst losses.

We observe that in the P-mode, the average number of points per frame decreases even more than in the variable bandwidth experiment. The average number of points in the R-mode is of the same order of magnitude than for the lossless transmission. However, the most dramatic effect is again the important inter-frame jitter in the P-mode. Especially the subsequent loss of retransmitted packets in presence of frequent burst losses leads to a high inter-frame jitter. The R-mode is still capable of achieving a reasonable update rate at the renderer. Note that in the burst loss R-mode experiment, we do not adapt the reconstruction process according to the packet loss rate. The information which is lost during irregular bursts is naturally corrected by the redundant coding of the R-mode. However, temporal visual artifacts are present. The characteristic parameters for the test runs with burst losses are summarized in Table 3, the cumulative probability distributions are presented in Figure 12.

5. Conclusions and outlook

Our results show that real-time 3D video streams can be efficiently represented using points as basic 3D primitives, combined with an inter-frame prediction based on differential updates. This compact encoding, which relies on a reliable transmission, can be used in favorable networking conditions or if the receiver end permits buffering before replay. If the network is unstable with respect to available bandwidth or if frequent packet losses occur, a redundant encoding scheme guarantees better performance.

In a multicasting setup where multiple viewers should receive the same 3D video stream, the redundant encoding scheme is easier to deploy. The reconstruction node however, needs to take into account the possibly disparate viewpoints. If the viewpoints cover many different viewing directions, the resolution for every single viewing direction will decrease because of processing bottlenecks and each viewer will only render a fraction of the information contained in the multicasted stream.

In the future, we plan to develop better compression schemes for 3D video data. Furthermore, we would like to investigate adequate error metrics for 3D video compression.

Acknowledgements

The authors would like to thank Wojciech Matusik for providing the IBVH source code, Michael Waschbüsch for proofreading the manuscript and all blue-c project members for the many fruitful discussions. This work has been funded by ETH Zurich as a “Polyprojekt” (grant no. 0-23803-00).

References

[1] M. Botsch, A. Wiratanaya, and L. Kobbelt. Efficient high quality rendering of point sampled geometry. In Proceedings of the 13th Eurographics Workshop on Rendering, pages 53–64, 2002.

[2] H. Briceno, P. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos. In Proceedings of ACM Symposium on Computer Animation 2003, July 2003.

[3] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In Proceedings of SIGGRAPH 03, pages 569–575. ACM Press / ACM SIGGRAPH, July 2003.

[4] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In Proceedings of ACM SIGCOMM ’90, pages 200–208, September 1990.

[5] The common object request broker: Architecture and specification, version 3.0. Object Management Group, July 2002.
[6] E. O. Elliott. Estimates of error rates for codes on burst-noise channels. Bell System Technical Journal, 42:1977–1997, September 1963.
[7] 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 Proceedings of SIGGRAPH 03, pages 819–827. ACM Press/ ACM SIGGRAPH, July 2003.
[8] S. Gumhold and W. Strasser. Real time compression of triangle mesh connectivity. In Proceedings of SIGGRAPH 98, pages 133–140. ACM SIGGRAPH, Addison Wesley, 1998.
[9] M. Gutierrez, F. Vexo, and D. Thalmann. A MPEG-4 virtual human animation engine for interactive web based applications. In Proceedings of IEEE International Workshop on Robot and Human Interactive Communication, pages 554–559, September 2002.
[10] E. He, J. Leigh, O. Yu, and T. DeFanti. Reliable blast UDP: Predictable high performance bulk data transfer. In Proceedings of IEEE Cluster Computing, pages 317–324, September 2002.
[11] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. In IEEE MultiMedia, volume 4, pages 43–54, January-March 1997.
[12] S. Keshav and S. P. Morgan. SMART retransmission: Performance with overload and random losses. In Proceedings of INFOCOM 97, pages 1131–1138, 1997.
[13] H. Lee, M. Desbrun, and P. Schroeder. Progressive encoding of complex isosurfaces. In Proceedings of SIGGRAPH 03, pages 471–475. ACM Press / ACM SIGGRAPH, July 2003.
[14] G. Martin. Range encoding: an algorithm for removing redundancy from a digitised message. In Video & Data Recoding Conference, Southampton, 1979. http://www.compressconsult.com/rangecoder/.
[15] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In Proceedings of Eurographics Workshop on Rendering, pages 115–126, 2001.
[16] 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.
[17] S. Moezzi, A. Katkere, D. Y. Kuramura, and R. Jain. Immersive video. In Proceedings of the 1996 Virtual Reality Annual International Symposium, pages 17–24. IEEE Computer Society Press, 1996.
[18] MPEG-3DAV. Description of exploration experiments in 3DAV. ISO/IEC JTC1/SC29/WG11 N5700, July 2003.
[19] 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.
[20] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. IEEE Transactions on Visualization and Computer Graphics, 5(1):47–61, 1999.
[21] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. In SIGGRAPH 2000 Conference Proceedings, ACM Siggraph Annual Conference Series, pages 343–352, 2000.
[22] S. Rusinkiewicz and M. Levoy. Streaming QSplat: A viewer for networked visualization of large, dense models. In Proceedings of the 2001 Symposium on Interactive 3D Graphics, pages 63–68. ACM, 2001.
[23] K. Sayood. Introduction to Data Compression. Morgan Kaufmann Publishers, 1996.
[24] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, January 1996.
[25] A. Smolic and H. Kimata. Applications and requirements for 3DAV. ISO/IEC JTC1/SC29/WG11 N5877, July 2003.
[26] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In Proceedings of the 13th ACM Eurographics Workshop on Rendering, June 2002.
[27] S. Wuermlin, E. Lamboray, and M. Gross. 3D video fragments: Dynamic point samples for real-time free-viewpoint video. In Computer and Graphics, Special Issue on Coding, Compression and Streaming Techniques for 3D and Multimedia Data, Elsevier Ltd, 2004. To appear.
[28] S. Wuermlin, E. Lamboray, O. G. Staadt, and M. H. Gross. 3D video recorder. In IEEE Pacific Graphics 2002 Proceedings, pages 325–334. IEEE Computer Society Press, October 2002.

a) b) c)

Figure 9: Examples from a 3D video sequence, showing different artifacts due to a lossy transmission. a) results from a lossless transmission, b) shows the effect of missing UPDATE and DELETE operations below the arm, c) shows the effect of missing INSERT operations in the right part of the body.

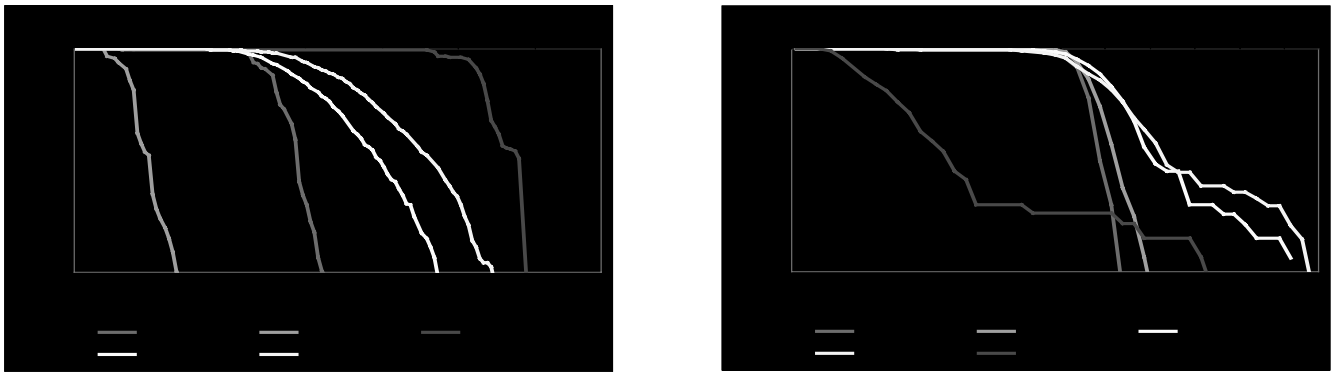

Figure 10: No simulated packet loss: a) number of points per frame; b) bandwidth in megabit per second.

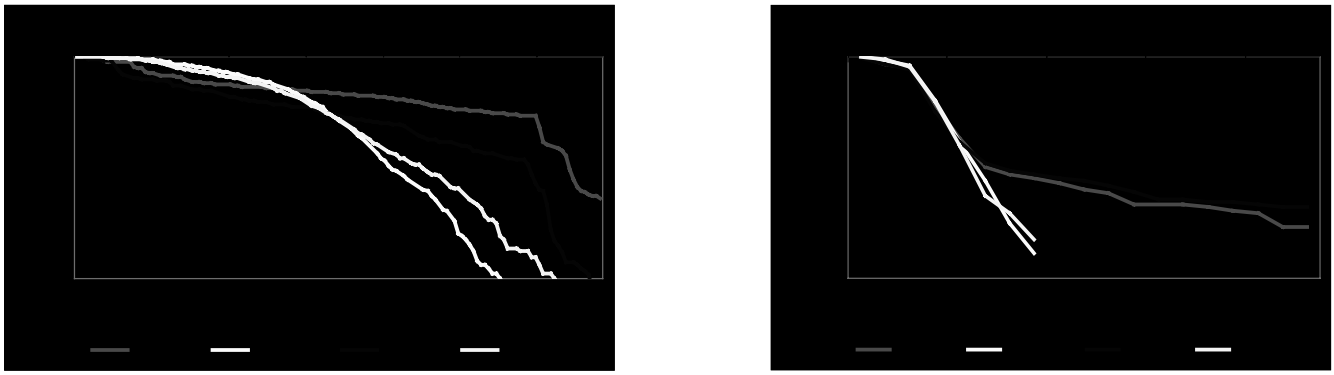

Figure 11: Limited bandwidth streams: a) number of points per frame; b) inter-frame period in milliseconds.

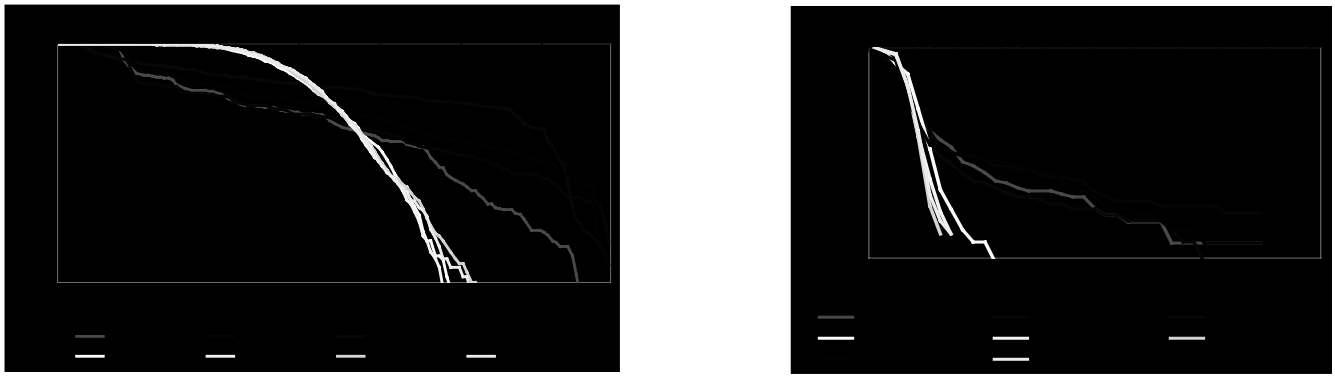

Figure 12: Streams with burst losses: a) number of points per frame; b) inter-frame period in milliseconds.