

# Stereoscopic and Multiple-Perspective Video Streaming System

<sup>1</sup>Goran Petrovic                      <sup>1,2</sup>Peter H.N. de With  
<sup>1</sup>Eindhoven University of Technology   <sup>2</sup>LogicaCMG Netherlands  
P.O. Box 513                              P.O. Box 7089  
5600MB Eindhoven                      5605JB Eindhoven  
The Netherlands                        The Netherlands  
g.petrovic@tue.nl                      P.H.N.de.With@tue.nl

## Abstract

3D-video systems allow a user to perceive depth in the viewed scene and to display the scene from arbitrary viewpoints interactively and on-demand. This paper presents a prototype implementation of a 3D-video streaming system using an IP network. The architecture of our streaming system is *layered*, where each information layer conveys a single coded video signal or coded scene-description data. We demonstrate the benefits of a layered architecture with two examples: (a) stereoscopic video streaming, (b) monoscopic video streaming with remote multiple-perspective rendering. Our implementation experiments confirm that prototyping 3D-video streaming systems is possible with today's software and hardware. Furthermore, the performance tests indicate that our system compares well to other systems presented in recent 3D-video streaming research.

## 1 Introduction

*3D-Television (3D-TV)* systems allow a user to perceive depth in the viewed scene [1]. To render the depth effect, either head-mounted glasses or an auto-stereoscopic display are used. 3D-Television system concepts can also be extended to multi-perspective viewing of a video scene. A *multiple-perspective video* system allows a scene to be displayed from arbitrary viewpoints (angles) interactively and on-demand. Thus, a special case of multiple-perspective viewing is *stereoscopic* viewing, where a scene is reconstructed from two nearby viewpoints simultaneously, and shown with the help of a specialized display device (e.g., stereoscopic display). In the sequel, we refer to stereoscopic and multiple-perspective video signals jointly as “*3D-video*” and make clear distinctions where appropriate.

3D-Video systems require a new system architecture and components for capturing, coding, transmission and rendering of stereoscopic and multiple-perspective video signals [1]. This paper presents the design and a prototype implementation of a 3D-video streaming system employing an IP network. In other words, our paper focuses on 3D content transmission and delivery aspects, while making the following two assumptions. First, we assume that suitable multi-camera capturing systems can be constructed, as exemplified in recent work on this topic (e.g., [2] [3] [4] [5]). Second, we believe that the state-of-the-art video coding standards (MPEG-4, H.264) – although not specifically

tailored to the compression of 3D-video data – are at least readily applicable for fast prototyping and building operational streaming systems. Armed with these two assumptions, we set out to implement an efficient 3D-video streaming architecture while addressing the *key practical trade-off in 3D-video streaming*: the balance between the *server computation load* and the *network bandwidth*.

The remainder of the paper is structured as follows. Section 2 surveys experimental 3D-video streaming systems in the recent literature. Section 3 motivates our design choice, and briefly introduces the layering concept and the view interpolation algorithm. In Section 4 we detail on the software components, their integration in the system and the end-to-end system performance. We conclude in Section 5 by highlighting the main points of the presented work.

## 2 Related work

Our work builds upon recent research in several application areas related to networked 3D-video systems, most notably stereoscopic video streaming over the Internet, tele-immersion and multi-view client-server systems.

*Stereoscopic video transmission* over the IP networks was first considered in the *Smile!* teleconferencing system [6]. *Smile!* is an end-to-end system for capturing, encoding, transmission and display of stereo streams. The system features separate JPEG coding of the two streams, standard protocols for real-time transmission [7] and active shutter-glasses for display. The main focus of this work is on the signalling extension to the RTCP [7]. This extension is necessary for the association of the two streams as belonging to the same session and distinction between them within the session. More recent system studies employ state-of-the art video codecs for compression (e.g., H.264) and investigate the bitrate savings achievable when encoding the two streams in different qualities or spatio-temporal resolutions [8] [9]. Matusik and Pfister [2] present an end-to-end system for multi-view stereoscopic rendering of a dynamic scene. The camera streams are encoded using MPEG-2, and rendered on a multi-projector system with lenticular screens. The focus of their work is on the projector system design and rendering, rather than on the encoding and transmission aspects.

*Immersive teleconferencing* systems typically employ multi-camera recording systems at each participant's site and some form of 3D scene reconstruction such that arbitrary views of the scene can be rendered at remote sites. The Coliseum system [3] reconstructs a *rough 3D model* of the foreground object (*Image-Based Visual Hull (IBVH)*). Multiple-perspective views of the scene are rendered locally, encoded and transmitted as conventional 2D video streams. A standard MPEG-4 encoder is used for real-time compression of the rendered video streams and UDP serves as the transport protocol. A related conferencing system is that of Kauff and Schreer [10], which relies on *3D image warping* to create the correct perspective views rather than building 3D scene models. The similarity of the two systems is that they both render the views locally and send only the resulting 2D-video over the network. A radically different approach to immersive teleconferencing is presented in the work of Towles *et al.* [11]. In this system, the entire 3D scene representation is transmitted to the receiver. A

set of “3D-cameras”, each consisting of three conventional video cameras is used to record the scene. The output of such a 3D-camera consists of a 2-D video stream and a 3D scene-geometry information in the form of a depth stream. The aggregate data set is streamed over a high-bandwidth WAN network (Internet2). Follow-up work focuses on the issues of compression [12] and congestion control [13] in the multi-stream transport. More recently, Yang *et al.* [14] propose a framework for QoS adaptation in a multi-stream immersive environment based on unequal importance of the individual streams. Lamboray *et al.* [4] consider similar issues in the context of streaming and rendering of point-based scene representations.

In *multi-view client-server systems* (or equivalently, *Free-Viewpoint Television* [1]), a remote scene can be displayed from arbitrary viewpoints interactively and on-demand. The setup of Kimata *et al.* [15] consists of a large number of closely spaced cameras so that arbitrary perspective views can be created by view interpolation in the *ray-space*. When the receiver requests a viewpoint change, the server transmits only the views required to create the desired virtual viewpoint. A similar system, supporting unicast and multicast of multiple viewpoints and a number of multi-view special effects, is presented by Lou *et al.* [5]. Without view interpolation, the set of available viewpoints in their system is limited to the original cameras. Kurutepe *et al.* [16] present a system for efficient delivery of multi-view video over IP networks based on peer-to-peer overlay multicast. In our earlier work [17], we propose a framework for layered transmission of multi-stream data and also suggest the application of network overlay architectures for higher server and network efficiency. Additionally, the proposed framework introduces virtual viewpoint creation (view interpolation) as a new functionality to be implemented in a network overlay, e.g., in a group of collaborating receivers.

### 3 System design and architecture

The architecture of our streaming system is *layered*, where each information layer conveys a single coded video signal or coded scene-description data. The generic layered framework is presented in our earlier work [17], and its benefits are twofold. Firstly, the information layering is a well-known design guideline to deal with heterogeneity of receivers in the system. This way, the receivers can select the number of layers to receive based on their preferences or capabilities. As a result, representation scalability is achieved, where the presentation quality depends on the selected number of layers and can be extended dynamically and on-demand. Secondly, having the 3D content organized in separate layers can lead to a higher bandwidth efficiency in practical streaming scenarios. For example, broadcasting all the available viewpoints in multiple-perspective video is inefficient, considering that some of them will be requested frequently, and others not at all. A better design is to schedule viewpoint transmissions interactively, as a response to the actual user interest.

Our streaming system ensures a synchronized transmission of the encoded layers over an IP network to the receiver. The receiver application includes the components for synchronous layer decoding, stereoscopic signal generation and display. In the sequel, we describe how the two key aspects of 3D-video streaming, namely the stereoscopic and the multiple-perspective streaming, can be implemented within this framework.

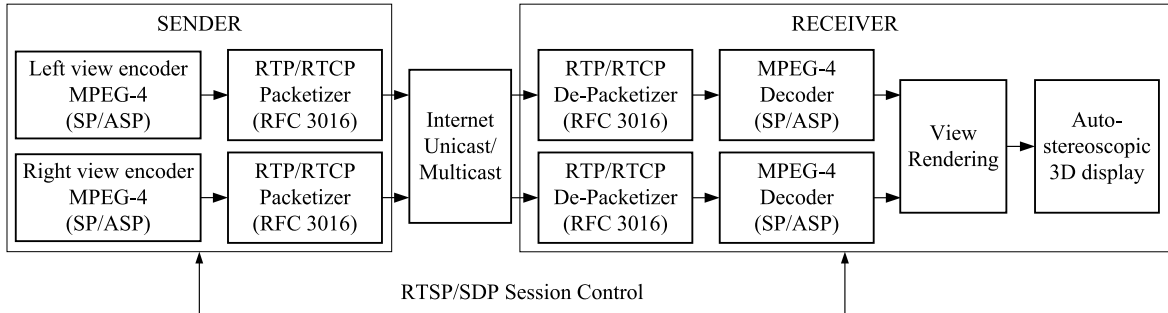


Figure 1: Example of a dual-layer stereoscopic video streaming system.

For the experimental implementation of stereoscopic video streaming, we currently instantiate two layers. Depending on the stereoscopic image-generation method available at the receiver end, either a *depth* stream and a *texture* stream (either left or right), or both the *left* and the *right* camera streams [17] are transmitted. In our system, the left and the right camera views are considered separate layers, and are independently encoded, stored and transmitted. The number of layers transmitted can be extended to include multiple viewpoints and such a configuration is referred to as a *multi-view streaming system*. A dual-layer system configuration including the left and the right camera streams, is shown in Figure 1.

With respect to multiple-perspective video streaming, we contribute as follows. *View interpolation* is required to display a scene from viewpoints between the original cameras and enable a continuous perspective change. Our multiple-perspective streaming system includes a view interpolation algorithm [18] which enables a continuous horizontal transition between pairs of user-selected viewpoints. The following layers are included for each pair of adjacent capturing cameras: (1) the left camera stream; (2) the right camera stream; (3) per-pixel disparity maps; (4) occlusion information (explicitly detected pixels visible in one of the streams only). Our current experimental system implementation is an interactive client-server application in which the server creates the interpolated views, encodes and streams them to the client in real time. To achieve real-time performance, the interpolation module partly relies on hardware acceleration, i.e., it uses the OpenGL API [19] to take advantage of the hardware rendering capabilities of today's PC graphics cards. Figure 2 depicts the architecture of our setup and its main components.

## 4 Implementation and Results

In this section we present our implementation of an experimental system for 3D-video streaming over IP networks. The system includes software modules to demonstrate both key aspects of 3D video streaming introduced earlier in this paper: (1) stereoscopic video streaming, (2) monoscopic streaming with remote multiple-perspective rendering. The current version of the system implements these two 3D-video modal-

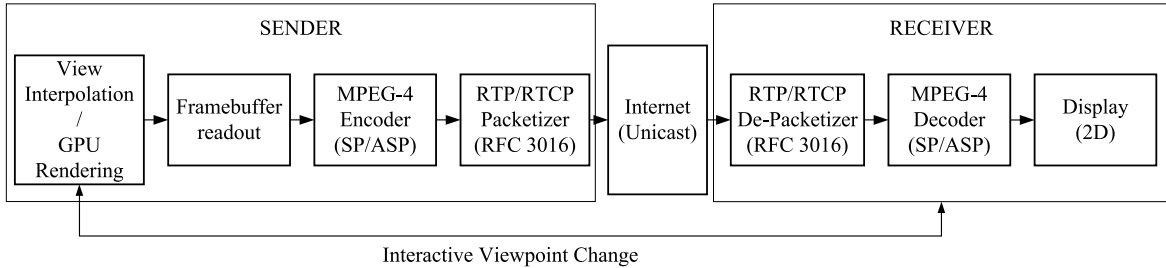


Figure 2: Multiple-perspective video streaming.

ities as separate applications. The combination of both modalities is ongoing and planned for the next implementation stage. In the sequel, we elaborate on the implementation details of both modalities and the main software components.

#### 4.1 Stereoscopic streaming

At the sender, the pre-recorded left and right camera views are independently encoded using an open-source MPEG-4 SP encoder [20], and stored in separate files. When a streaming session is started, the compressed bitstreams are read frame by frame, packetized in RTP packets using the open-source Live555 library [21], and sent to the receiver. According to IETF recommendations for carriage of MPEG-4 elementary streams (ESs) over IP networks [7] [22], the two streams are transmitted as separate RTP sessions, i.e., using different RTP/RTCP ports. The sender’s packet scheduler implements a round-robin discipline – it intersperses transmissions of the left and the right video stream on a frame basis.

The receiver system includes components for RTP depacketization, decoding and display. The RTP depacketizers assemble the received RTP packets into compressed MPEG-4 frames and pass them on to the corresponding left and right stream decoders. As both decoders run in the same thread, a special application-level coordination is implemented to ensure fair treatment of both streams. The display module receives pairs of decoded frames and combines them to create a *stereoscopic image*, in a format specific to the auto-stereoscopic display we use in the system [23].

We achieve real-time end-to-end system performance with both half-SDTV (Standard Definition broadcast TV,  $720 \times 288$ ) and VGA-type resolution sequences (e.g.,  $800 \times 600$ ), while running the receiver application on a commodity PC. The depth effect while viewing a 3D scene is compelling with both synthetic and real-world sequences (captured in our lab). The synchronization between the two streams is maintained over all the system components, providing a working end-to-end prototype for stereoscopic video streaming with glitch-free display.

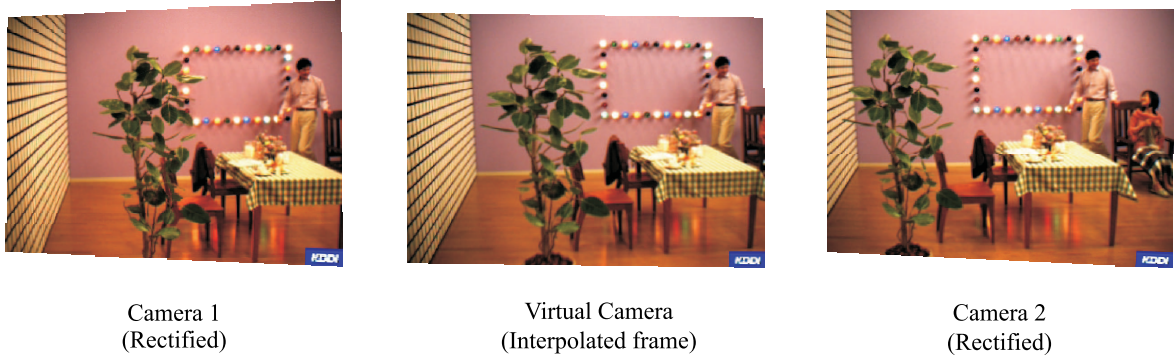


Figure 3: Virtual view interpolation.

## 4.2 Multiple-perspective streaming

The implemented multiple-perspective video streaming software allows a remote scene to be displayed from multiple viewpoints (angles) interactively and on-demand.

The viewpoint control to support client interactivity is possible through standard-PC input devices (a mouse or a keyboard). In our system, the input device events generated while moving the mouse pointer inside of the video display window, trigger the execution of the interpolation algorithm. As a result, *virtual-viewpoint frames* are rendered, corresponding to the current position of the pointer in display window coordinates. In the current implementation, we only handle locally-generated input device events, while an extension to the events coming in from the network (a client requesting the viewpoint change to be rendered by the server) is under development.

The rendered virtual-viewpoint frames are read back from the GPU frame buffer, and color-space conversion is performed for every frame (RGB to YUV). The resulting raw video frames are compressed and transmitted to the receiver in real-time using the same components as in Section 4.1. Figure 3 illustrates two original camera frames and an interpolated virtual frame between them.

As only a monoscopic video stream (containing both the virtual and the original video frames) is transmitted, the receiver application can be any media-player client with streaming support (e.g., [24]). Because our remote rendering design is geared towards enabling multiple-perspective video on resource-constrained clients, we have run tests with low-resolution multi-view sequences (e.g., CIF)). The sender (a 3GHz Desktop-PC with a state-of-the-art graphics card <sup>1</sup> renders the multiple-perspective video frames in real-time and streams them to the receiver (a 1.5GHz notebook-PC) which uses a VLC client [24] for decoding and display. The selection of the viewing perspective is done with a mouse pointer at the sender; the effects of which are displayed at the receiver after a short delay, thereby demonstrating the interactive properties of our system.

<sup>1</sup>Commercially available under Nvidia GeForce 7600GS.

## 5 Conclusion

3D-Video streaming systems require a new system design and new components for the recording, compression, transmission and rendering of 3D-video signals. We have contributed in two key aspects. First, we have designed and implemented a stereoscopic video streaming system, relying on open standards and open-source software where possible. Second, we have extended the state-of-the-art in 3D-video streaming research by incorporating an interactive view rendering and interpolation algorithm into our multiple-perspective streaming system. Finally, we have validated the layered system architecture based on our earlier architecture studies [17], in order to enable support for heterogeneous clients. Our current operational prototype demonstrates that highly heterogeneous clients can coexist in the system, ranging from auto-stereoscopic 3D displays to resource-constrained mobile devices.

## References

- [1] A. Smolic and P. Kauff, “Interactive 3D video representation and coding technologies,” *Proceedings of the IEEE*, vol. 93, no. 1, pp. 98–110, Jan. 2005.
- [2] W. Matusik and H.-P. Pfister, “3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes,” in *Proc. Comp. Graphics (SIGGRAPH’04)*, Aug. 2004, pp. 814–824.
- [3] H. Baker, N. Bhatti, D. Tanguay, I. Sobel, D. Gelb, M. Goss, W. Culbertson, and T. Malzbender, “Understanding performance in Coliseum, an immersive videoconferencing system,” *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, no. 2, pp. 190–210, May 2005.
- [4] E. Lamboray, S. Würmlin, and M. Gross, “Data streaming in telepresence environments,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 11, no. 6, pp. 637–648, Nov. 2005.
- [5] J.G. Lou, H. Cai, and J.Li, “A real-time interactive multi-view video system,” in *Proceedings of the 13th annual ACM international conference on Multimedia*, Nov. 2005, pp. 161–170.
- [6] M. Johanson, “Stereoscopic video transmission over the Internet,” in *IEEE Workshop on Internet Applications*, July 2001.
- [7] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, “RTP: A transport protocol for real-time applications,” *RFC 3550, IETF*, July 2003.
- [8] A. Aksay, S. Pehlivan, E. Kurutepe, C. Bilen, T. Ozcelebi, G. Bozdagi Akar, M. Reha Civanlar, and A. Murat Tekalp, “End-to-end stereoscopic video streaming with content-adaptive rate and format control,” *Signal Processing: Image Communication*, vol. 22, no. 2, pp. 157–168, Feb. 2007.
- [9] H. Kalva, L. Christodoulou, L. M. Mayron, O. Marques, and Borko Furht, “Design and evaluation of a 3D video system based on H.264 view coding,” in *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2006.

- [10] P. Kauff and O. Schreer, "An immersive 3D video-conferencing system using shared virtual team user environments," in *Proceedings of the 4th international conference on Collaborative virtual environments*, Sept. 2002, pp. 105–112.
- [11] H. Towles, S.-U. Kum, T. Sparks, S. Sinha, S. Larsen, and N. Beddes, "Transport and rendering challenges for multi-stream 3D tele-immersion data," in *NSF Lake Tahoe Workshop on Collaborative Virtual Reality and Visualization*, Oct. 2003.
- [12] S.U Kum and K. Mayer-Patel, "Real-time multidepth stream compression," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, no. 2, pp. 128–150, May 2005.
- [13] D. E. Ott and K. Mayer-Patel, "Coordinated multi-streaming for 3D tele-immersion," in *Proceedings of the 12th annual ACM international conference on Multimedia*, Oct. 2004, pp. 596–603.
- [14] Z. Yang, B. Yu, K. Nahrstedt, and R. Bajcsy, "A multi-stream adaptation framework for bandwidth management in 3D tele-immersion," in *16th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2006.
- [15] H. Kimata, M. Kitahara, K. Kamikura, and Y. Yashima, "System design of free view-point video communication," in *Proceedings of the Fourth International Conference on Computer and Information Technology (CIT04)*, 2004.
- [16] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "Interactive transport of multi-view videos for 3DTV applications," *Journal of Zhejiang University: Science A*, vol. 7, no. 5, pp. 830–836, 2006.
- [17] G. Petrovic and P.H.N. de With, "Near-future streaming framework for 3D-TV applications," in *International Conference on Multimedia and Expo (ICME)*, July 2006, pp. 1881–1884.
- [18] D. Farin, Y. Morvan, and P.H.N. de With, "View interpolation along a chain of weakly calibrated cameras," in *IEEE Workshop on Content Generation and Coding for 3D-Television*, June 2006.
- [19] "OpenGL," <http://www.opengl.org/>.
- [20] "FFmpeg Multimedia System," <http://ffmpeg.mplayerhq.hu/>.
- [21] "Live555 Streaming Media," <http://www.live555.com/>.
- [22] Y. Kikuchi, T. Nomura, S. Fukunaga, Y. Matsui, and H. Kimata, "RTP payload format for MPEG-4 audio/visual streams," *RFC 3016, IETF*, Nov. 2000.
- [23] "SeeReal Technologies," <http://www.seereal.com/>.
- [24] "VLC media player," <http://www.videolan.org/vlc/>.