A distributed delivery model for 3D-video streams

Goran Petrovic
Eindhoven University of Technology
Den Dolech, 5600 MB
Eindhoven, The Netherlands
g.petrovic@tue.nl

Dirk Farin
Eindhoven University of Technology
Den Dolech, 5600 MB
Eindhoven, The Netherlands
d.s.farin@tue.nl

Peter H. N. de With
Eindhoven University of Technology and LogicaCMG
Den Dolech, 5600 MB
Eindhoven, The Netherlands
P.H.N.de.With@tue.nl

ABSTRACT

3D-video systems allow a user to perceive depth in the viewed scene and to interactively display the scene from multiple viewpoints. This paper presents an architecture for the scalable delivery of 3D-video streams over IP-networks, where we contribute in two aspects. First, we model 3D-video as a multi-stream application, where each stream (or *layer*) carries a single coded video signal or coded scene-description data. Second, we propose an end-to-end delivery model for 3D-video applications, which leverages a distributed system architecture to reduce the bandwidth and processing cost at the server and the end-hosts.

Keywords

Multi-media streaming, 3D-video, Streaming-CDN

1. INTRODUCTION

The emergence of 3D-TV and immersive-teleconferencing applications has spawned interest in building large multi-camera recording systems [26]. In such a system, a number of consumer-grade cameras are used to synchronously record a scene from multiple viewpoints. In a *multiple-perspective* viewing scenario [7], a scene can be displayed from different viewpoints (angles) interactively and on-demand. A user either selects a new viewpoint, or his movements are continuously tracked and the displayed content automatically adjusted. *Stereoscopic video* is a special case of multiple-perspective viewing, where the depth effect is rendered with the help of a specialized display device (head-mounted glasses or an auto-stereoscopic display). For brevity, we refer jointly to both multiple-perspective and stereoscopic applications as *3D-video* and make clear distinctions where appropriate.

The need to serve a large number of video streams for a single 3D scene, coupled with the processing for interactive viewpoint adaptation, may require new trade-offs in the design of multimedia servers, clients and delivery architectures. This paper focuses on the end-to-end delivery architecture for 3D-video streaming. Specifically, we address the problem of delivering 3D-video to a large and highly heterogeneous client-base in today's IP networks in a scalable fashion. In doing so, we assume that suitable multi-camera recording systems can be constructed, as exemplified in recent work on this topic (e.g., [26] and the references therein). Moreover, our view is that the state-of-the-art video coding standards (MPEG-4/H.264) – although not specifically tailored for the compression of 3D-video data – are at least readily applicable to fast prototyping and building operational streaming systems. Armed with these two assumptions, we address the key design trade-off in 3D-video streaming architectures: *the balance between the computation load and the network bandwidth.* To this end, we propose a distributed delivery architecture where processing modules extend a generic Streaming-CDN to incorporate various forms of 3D-video rendering. A Streaming Content Delivery Network (Streaming-CDN) is a wide-area overlay network of streaming servers/proxies, which provide control over the content stored at individual network nodes as well as the traffic between the nodes.

The remainder of the paper is structured as follows. Section 2 surveys the recent related work and puts our work in context. Section 3 presents our model for the end-to-end 3D-video delivery architecture. In Section 4 we report on an ongoing implementation of the 3D streaming and rendering algorithms within the proposed delivery model. Section 5 highlights the main points of the presented work and suggests directions for future work.

2. BACKGROUND AND RELATED WORK

When multiple synchronized cameras are used to record a scene, interactive viewpoint changes can be directly supported by switching to the desired camera stream. However, to support *continuous viewing while changing the viewpoint*, the number of physical cameras may grow impractically large. For this reason, most 3D-video systems render *virtual viewpoints* at locations between the original viewpoints. To ensure a seamless transition and a similar image quality across all views, virtual views have to be automatically constructed from a number of selected original camera streams. *View interpolation* in the context of 3D-video refers to a set of rendering algorithms which render virtual views of the scene by blending a number of original views. These algorithms either render virtual views directly, or assume some form of scene description (e.g., a geometric model) in order to generate those views more efficiently [21].

IMMERSCOM 2007, October 10-12, 2007, Verona, Italy.

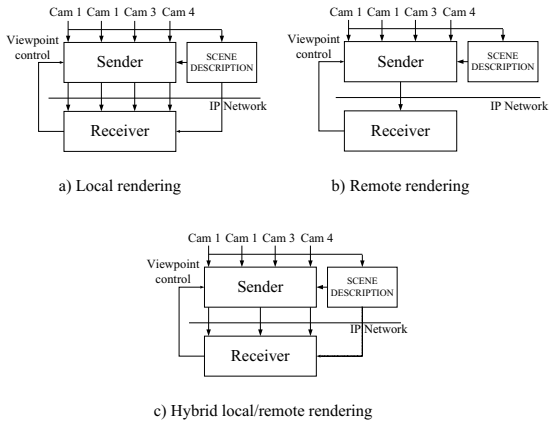

Figure 1: Distributed 3D-video delivery models.

Different architectural options exist for implementing view interpolation in 3D-video systems. In general, either the sender renders the desired virtual viewpoints and streams them to the receivers, or the sender transmits a number of original streams to each receiver to perform the interpolation. To better understand possible design options and analyze the differences between them, we borrow a categorization from computer graphics research. Martin [14] gives an overview of the approaches for partitioning the rendering load between the server and the clients in networked graphics applications. We adopt the same general concept, but confine ourselves to considering real-world 3D scenes as opposed to synthetic scenes. According to the classification in [14], there are three broad categories of distributed rendering approaches.

Local rendering (Fig. 1a). A delivery model where a 3D-video sender transmits all available camera streams to the receiver is common in *immersive teleconferencing* systems [22], which employ multi-camera set-ups at each participating site. The design choice is dictated by the application – to create a continuous immersive feeling, all available streams must be rendered at the same time. Efficient compression [11], congestion control [16] and adaptive streaming strategies [24] are all important to maximize the user Quality of Service (QoS). Our objective is to explore a different application space. Instead of focusing on immersive (and computationally expensive) scene rendering for a single user, our main goal is to enable interactive exploration of a remote scene for a large number of concurrent users.

Remote rendering (Fig. 1b). This delivery model is common in computer graphics for implementing complex rendering tasks. The model is particularly useful if a client does not have the resources to render the full model, while rendering a simplified model is unacceptable. In this case, a powerful remote server renders the model and sends the resulting high-quality images to the client [9]. For bandwidth efficiency, these images are compressed at a sufficient quality.

Hybrid rendering (Fig. 1c). Hybrid methods balance the rendering load between the client and the server. The server renders a part of the scene, while the client combines those images with the locally rendered scene parts [25].

Discussion

The design choice among the alternative architectures is application-dependent and illustrates a trade-off between the computation load at the server (and the client) and the network bandwidth. If the sender implements the view interpolation, bandwidth and rendering costs at the receiver will not be larger than in conventional 2D streaming. However, serving a large number of such requests concurrently may quickly consume computational resources at the sender. This is a serious concern for the scalability of 3D-video systems. Similarly, if the view interpolation is implemented at the client, multiple video streams and scene description data need to be delivered to the client in real-time. The associated bandwidth and rendering costs may be prohibitively high for a large fraction of the today's client base. Still, this delivery model has already been considered for multi-view stereosopic 3D-TV applications [15]. System optimizations to reduce the bandwidth cost include a combination of unicast and IP multicast for the delivery of multiple viewpoints [13]. The lack of global Internet support for IP-multicast can be alleviated using a peer-to-peer overlay network for multi-view video delivery [12]. Without view interpolation, the set of available viewpoints in these systems is limited to the original cameras. Our work is complementary in that our system proposal includes a view interpolation module, which is efficient and flexible enough to be employed at either the sender, or at an intermediate node (e.g., proxy/peer), or at the receiver.

3. SYSTEM PROPOSAL

3.1 Selective and layered streaming

Our design decisions are based on two observations related to the 3D-video rendering process. First, the quality of interpolated virtual frames depends on the number of original frames that are blended together, as well as the availability of a suitable scene description. Second, in real-world scenarios, some scene viewpoints will be transmitted frequently, and others not at all, entirely driven by the user interest. Therefore, an on-demand transmission scheme will have a lower bandwidth cost than a flat scheme that transmits the entire scene to every user. Furthermore, bandwidth optimizations for partial scene transmission are best implemented at run-time, for every user independently. For example, a user navigating to a specific part of the scene will receive the camera streams and geometry streams for that part of the scene only.

Base on the above observations, we believe that 3D-video streaming systems are best implemented using a layered content representation scheme [17]. In this representation, each information layer conveys a single video stream or a scene-description stream. The receiver can select the number of different layers to receive, based on its preferences or capabilities. The layering concept is a well-known guideline for the design of adaptive streaming systems (e.g., the RLM scheme [18]), and has been considered for multi-view scene transmission [12]. However, in view of the heterogeneity in today's IP-networks and very specific 3D-video rendering requirements, a pure end-to-end adaptive strategy may not be sufficient. We assert that a distributed delivery architecture of today's Streaming-CDNs provides a suitable

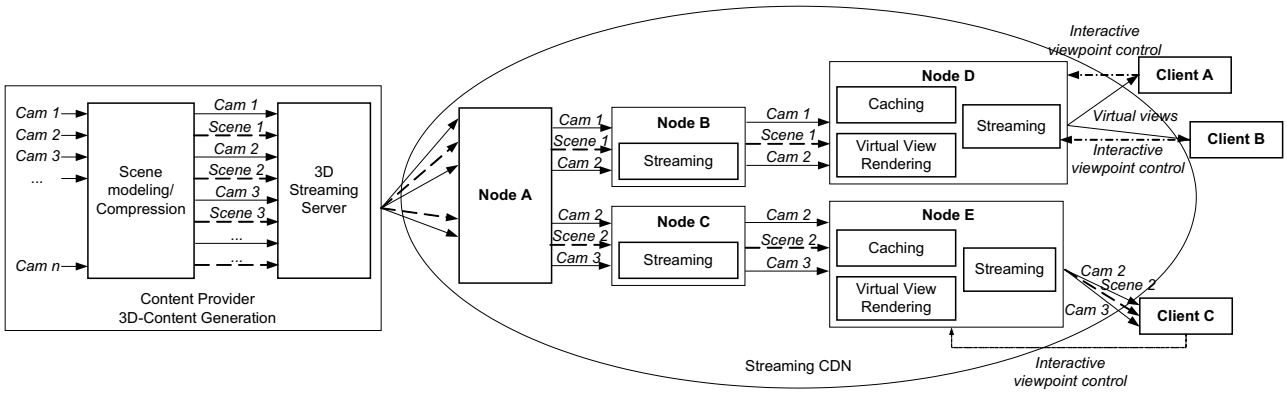

Figure 2: Distributed 3D-video delivery architecture.

model for implementing large-scale 3D-video streaming systems. Therefore, we propose to extend a Streaming-CDN with processing modules specific to 3D-video to incorporate the remote (Fig. 1b) or the hybrid rendering architecture (Fig. 1c) for the benefit of resource-constrained receivers. Such extensions and benefits they provide with respect to reducing the server bandwidth, the server processing load and latency are described next.

3.2 Streaming-CDNs

Many of today's large-scale commercial streaming systems are implemented over Streaming Content Delivery Networks (Streaming-CDNs) [3] [10]. A Streaming-CDN is a wide-area overlay network of streaming servers/proxies. Being privately managed, Streaming-CDNs provide control over the content stored at individual servers as well as the traffic between them. Their distributed architecture allows to apply network-wide load balancing, replication, caching and pre-fetching strategies, otherwise impossible with centralized cluster architectures. The benefits are direct for both the content providers and the clients. For providers, Streaming-CDNs provide significant bandwidth savings. Instead of serving one stream per client, a single stream needs to be sent to a Streaming-CDN, redirecting every incoming client to one of its servers. Ideally, the server will be selected according to the network metrics such as RTT to the client, topological proximity, or congestion level, thus improving the QoS delivered to the client [10]. In addition to generic functions (pre-fetching and caching), some Streaming-CDN servers implement stream-specific processing (transcoding and optimized packet scheduling) as a part of the client adaptation strategy [23].

3.3 3D-video over Streaming-CDNs

Our proposed three-tier architecture for the delivery of 3D-video streams over IP-networks is illustrated in Figure 2. It consists of a 3D-video server, a network of intermediate nodes, and the clients. The design is based on a generic Streaming-CDN and is extended with 3D-video rendering modules. In the sequel, the most important system aspects are discussed.

3.3.1 3D-Server

The 3D-server is a part of the content provider's network, located at the point of connection to a Streaming-CDN. Its back-end is functionally decomposed into scene recording, 3D-scene modelling and compression stages. For ease of exposition, we assume that the scene is recorded with an array of static, fully calibrated cameras. The raw camera frames and their calibration data are first processed at the scene modelling stage, which is discussed next. The compression of the resulting scene representation is covered in Section 4.

Scene modelling for 3D-video refers to a range of algorithmic approaches for the recovery of physical scene description from its camera recordings. These include extraction of scene geometry (3D-surfaces), surface reflectance properties and modelling of light sources [20]. We consider scene geometry extraction only, as our focus is multiple-perspective viewing under original, static lighting. Depending on the scope of geometric information recovered for a scene, the models can be broadly classified as local or global. Global approaches reconstruct a geometric model consistent with all input cameras and continuously update it over time (e.g., dynamic 3D-wireframe mesh models). Due to a high computation complexity, current global systems focus on individual scene objects [21]. Local geometric models only describe the scene from a subset of viewpoints (e.g., depth maps, inter-camera occlusion relationships). These reconstruction methods have a lower complexity, and the hardware for certain local geometry measurements such as depth, is already available [5]. A local, view-dependent geometric model matches well with selective scene transmission model. Therefore, our resulting 3D-video scene has a multi-stream representation, consisting of the video streams and the local scene-geometry streams.

3.3.2 Distributed 3D-video delivery

A Streaming-CDN for 3D-video delivery is an overlay network of nodes that cache, process and forward 3D-streams (Fig. 2 illustrates a simplified architecture). In this subsection, we attempt to apply a Streaming-CDN to the design of our 3D-video streaming system. It will become apparent that this approach will yield a number of benefits for the server and for the clients.

Streaming-CDN connects to a 3D-video server via one of its edge nodes (*Node A*). This node serves as the entry point for every 3D-stream requested by a client connected to one of the streaming edge-servers (*Nodes D and E*). For clarity, we consider a small session with only three clients concurrently navigating a region of the scene spatially delimited by

Cameras 1 and *3*.

Clients first join the session by sending a request to the 3D-server. The server replies with a list of available camera viewpoints, their calibration parameters and associated geometric description. After that, the client is redirected to a suitable streaming edge-server (*Client A* and *Client B* to the *Node D*, whereas *Client C* is redirected to *Node E*). The streaming edge-servers act as agents for their clients, fetching the additional streams as needed to support client interaction with the scene. Here, we assume that the extracted scene-geometry streams provide enough local information for the view interpolation algorithm to support a continuous blending between the corresponding original streams.

Reducing the 3D-server bandwidth cost. *Clients A* and *B* are navigating the scene region between *Camera 1* and *Camera 2*, while *Client C* is currently at a position between *Camera 2* and *Camera 3*. The 3D-server transmits the required video streams and their associated geometry streams. Note that each stream is transmitted only once, while the Streaming-CDN replicates the streams where needed. This way, the server-bandwidth cost depends on the number of active viewpoints, but is independent of the number of clients. Effectively, this scenario illustrates a multicast delivery implemented at the application layer. In view of the large number of streams compared to conventional 2D streaming, the benefits of efficient delivery schemes in 3D systems will be substantial.

Accommodating client heterogeneity. The remote rendering (Section 2) is an effective method to enable 3D-video on resource-constrained devices. Streaming-CDNs and their distributed architecture provides the opportunity to implement this approach in a scalable fashion. In this case, each node only handles a fraction of the total number of viewers. The main idea is to equip a number of streaming edge-servers with specialized rendering facilities and configure them for on-demand remote rendering. These servers can be strategically placed in a network – e.g., at the gateway between the network's wired and wireless parts. In Figure 2, *Node D* performs the viewpoint adaptation by remote rendering on behalf of the *Clients A* and *B*.

Enhancing client QoS. Stream propagation through a well-managed Streaming-CDN is possible with a lower delay and higher reliability than in the public Internet [10]. Furthermore, a proper selection of the streaming edge-server results in a low latency between the edge server and the client. Due to the interactivity requirement, QoS in 3D streaming systems has the latency as an important parameter. Finally, the view interpolation algorithms, requiring a synchronized access to multiple streams, may significantly benefit from bounding the delay jitter.

In conclusion, our proposed architecture is efficient in bandwidth because of the multicast delivery aspect, and the selective streaming driven by the clients. Moreover, the network is modified with an extra stream-processing function, i.e., view interpolation, which is facilitated by distributing rendering capabilities to specific network nodes.

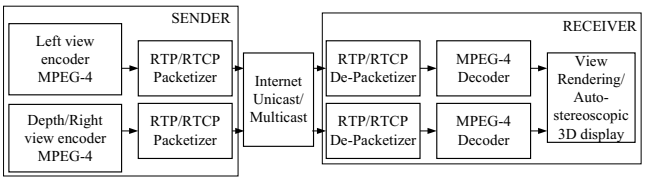

Figure 3: Example of a dual-layer stereoscopic video streaming system.

4. IMPLEMENTATION

In this section, we describe our implementation of the main building blocks of a 3D-video streaming system. The processing modules and protocols are designed to support an integration into a distributed delivery model of Section 3. More specifically, we present our choices for the data representation format, compression scheme, transmission protocols and the rendering algorithm. We emphasize that our implementation is based on ongoing work and is not yet as complete as the architecture presented in the previous section.

4.1 3D-Video format

3D-Video in our experimental system is represented as a set of original camera streams and *depth maps* - one for each original camera [27]. For every pixel in a video frame, a depth map conveys the distance between the camera plane and the nearest surface point in the scene. Using a single video frame and its depth map, multiple nearby perspective views can be rendered by re-projecting the pixel values driven by the depth map. For an overview of algorithms for automatic extraction of depth maps from video frames, the reader is referred to a recent survey [21].

4.2 Content coding

We currently rely on established coding standards to compress the data for transmission. Both the video streams and the associated depth streams are encoded using an MPEG-4 Simple Profile (SP) encoder. All camera and depth streams are encoded independently from each other. We acknowledge that in certain cases a higher compression gain would be possible if an inter-stream compression was performed in addition. This would be particularly effective for dense multi-camera systems, where the recordings of neighboring cameras largely overlap. However, the gain decreases for sparser camera arrangements [6]. A standardization activity investigating these issues is currently underway [21]. Once more efficient coding schemes become available, we will easily incorporate them in our layered transmission model. At the time of writing, we investigate the applicability of MPEG-4 AVC to our proposal. Lastly, an independent coding approach has its own benefits, especially from the system perspective. It allows a parallelization of the encoding process and a distribution over multiple processors [15]. Equally important is the backward compatibility with the legacy broadcast systems in the Internet. Namely, live streaming systems commonly employ real-time hardware encoders at the server back-end, and forward the compressed bitstreams to clients directly [3]. Therefore, the 3D-video representations can be supplied to the deployed coding systems as multiple independent streams.

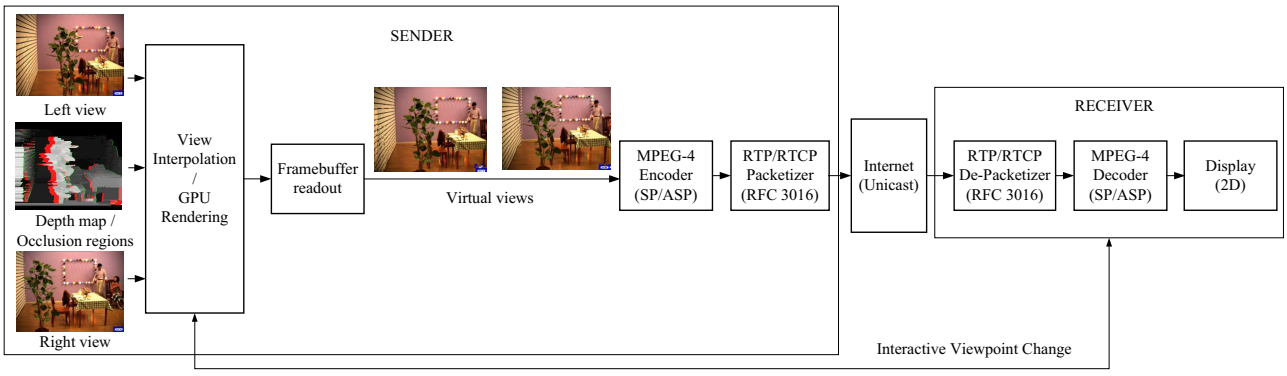

Figure 4: Multiple-perspective video streaming.

4.3 Layered streaming

For the real-time transport of 3D-video streams over IP networks, we employ unreliable transport service (UDP) and application-layer packetization [19]. According to IETF recommendations for the carriage of MPEG-4 elementary streams over IP networks [8], multiple encoded streams are transmitted as separate RTP sessions, i.e., using different RTP/RTCP ports. Therefore, our experimental 3D-video system is a multi-stream application where each compressed video/depth stream is transmitted in a separate RTP session (Fig. 3). The receiver application relies on RTP transport services (sequence numbers and time stamps) to ensure a synchronized decoding and display of the multi-stream data.

4.4 Virtual view rendering

The view interpolation algorithm integrated into our system [4] enables a *continuous horizontal blending* between pairs of original cameras. For each pair of adjacent cameras, the input to the algorithm includes: (1) the left camera stream, (2) the right camera stream, (3) per-pixel depth maps, and (4) occlusion information (pixels flagged as visible to one of the cameras only). The algorithm creates a virtual view at an arbitrary position between the cameras by weighting their respective contributions at every pixel location (Fig. 4). To achieve real-time performance, the view interpolation module partly relies on hardware acceleration, i.e., it uses the OpenGL API [1] to take advantage of the hardware-rendering capabilities of today's PC graphics cards.

4.5 Streaming prototype

Our prototype implementation demonstrates two aspects of 3D-video streaming: (a) stereoscopic video streaming, (b) monoscopic video streaming with remote multiple-perspective rendering.

For the implementation of stereoscopic video streaming, two layers are instantiated - either one *video* stream and its *depth* stream, or both the *left* and the *right* camera streams (Fig. 3). The streaming experiments were performed on a high-speed LAN in our department. We achieve real-time end-to-end system performance with both half-SDTV (Standard Definition broadcast TV, 720×288) and VGA-type resolution sequences (e.g., 800×600), while running the receiver application on a 3GHz desktop PC. The depth effect while viewing a real-world 3D scene is compelling with an auto-stereoscopic display [2].

Our current multiple-perspective streaming prototype is a remote rendering application in which the sender creates the interpolated views, compresses and streams them to the client in real time (Fig. 4). The depth maps and occlusion layers are computed in a pre-processing step and stored at the sender. The receiver navigates a remote scene by moving the mouse pointer inside of its video display window. The sender renders and streams a sequence of original and virtual frames, matching the client viewing parameters. As only a monoscopic video stream is transmitted, the receiver application can be any media-player with streaming support. The viewpoint change is effective with a delay of 500ms (mainly the buffering delay in the player software), thereby demonstrating the interactive properties of our system.

5. CONCLUSIONS AND FUTURE WORK

We have proposed a scalable model for the delivery of 3D-video streams over IP networks, leveraging a Streaming-CDN. The model accommodates heterogeneous clients by implementing 3D-video-specific functions in a distributed fashion. In our model, view interpolation can be implemented either at the server, or at an intermediate node, or at the client. To support such a delivery model, a layered content representation and streaming framework is developed. The layering in our system allows to aggregate the selected data at the node which performs the actual processing. Our current operational prototype demonstrates that highly heterogeneous clients can coexist in the system, ranging from auto-stereoscopic 3D displays to resource-constrained devices.

Our current implementation of multiple-perspective streaming assumes that all the data required for view interpolation is locally available. A natural extension is to explore the scenario where view interpolation is performed on the data streamed from a remote location. In the context of the implemented view interpolation scheme, two video streams, one depth stream and a sequence of occlusion masks would have to be delivered to the view interpolation process in real-time. As our model assumes best-effort IP networks and an unreliable transport protocol (UDP), an interesting avenue for future research would be to explore adaptive streaming algorithms. The adaptation in a 3D-video system will have to apply optimizations that cut across its multiple layers of information.

6. REFERENCES

[1] OpenGL. `http://www.opengl.org/`.

[2] SeeReal Technologies. `http://www.seereal.com/`.

[3] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik. Video coding for streaming media delivery on the internet. *IEEE Trans. Circuits & Systems Video Technol.*, 11(3):269–281, Mar. 2001.

[4] D. Farin, Y. Morvan, and P. de With. View interpolation along a chain of weakly calibrated cameras. In *IEEE Workshop on Content Generation and Coding for 3D-Television*, June 2006.

[5] C. Fehn, P. Kauff, M. O. de Beeck, F. Ernst, W. A. IJsselsteijn, M. Pollefeys, L. V. Gool, E. Ofek, and I. Sexton. Evolutionary and optimised approach on 3D-TV. In *Proc. Int. Broadcast Conf. (IBC)*, pages 357–365, Sept. 2002.

[6] M. Flierl, A. Mavlankar, and B. Girod. Motion and disparity compensated coding for multi-view video. *Submitted to IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on Multiview Video Coding*, 2007.

[7] R. Jain and K. Wakimoto. Multiple perspective interactive video. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 202–211, May 1995.

[8] 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.

[9] D. Koller, M. Turitzin, M. Levoy, M. Tarini, G. Croccia, P. Cignoni, and R. Scopigno. Protected interactive 3-D graphics via remote rendering. *ACM Trans. Graphics*, 23(3):695–703, 2004.

[10] L. Kontothanassis, R. Sitaraman, J. Wein, D. Hong, R. Kleinberg, B. Mancuso, D. Shaw, and D. Stodolsky. A transport layer for live streaming in a content delivery network. *Proceedings of the IEEE*, 92(9):1408–1419, Sept. 2004.

[11] S. Kum and K. Mayer-Patel. Real-time multidepth stream compression. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 1(2):128–150, May 2005.

[12] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp. Interactive transport of multi-view videos for 3DTV applications. *Journal of Zhejiang University: Science A*, 7(5):830–836, 2006.

[13] J. 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*, pages 161–170, Nov. 2005.

[14] I. Martin. Hybrid transcoding for adaptive transmission of 3d content. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 373–376, July 2002.

[15] 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)*, pages 814–824, Aug. 2004.

[16] 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*, pages 596–603, Oct. 2004.

[17] G. Petrovic and P. de With. Near-future streaming framework for 3D-TV applications. In *IEEE International Conference on Multimedia and Expo (ICME)*, pages 1881–1884, July 2006.

[18] S. R.McCanne. Scalable compression and transmission of internet multicast video. *Ph.D. dissertation, University of California Berkeley*, 1996.

[19] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A transport protocol for real-time applications. *RFC 3550, IETF*, July 2003.

[20] H. Shum, S. Kang, and S. Chan. Survey of image-based representations and compression techniques. *IEEE Trans. Circuits & Systems Video Technol.*, 13(11):1020–1037, Nov. 2003.

[21] A. Smolic and P. Kauff. Interactive 3D video representation and coding technologies. *Proceedings of the IEEE*, 93(1):98–110, Jan. 2005.

[22] 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.

[23] S. Wee, J. Apostolopoulos, W. Tan, and S. Roy. Research and design of a mobile streaming media content delivery network. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2003.

[24] 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.

[25] I. Yoon and U. Neumann. Ibrac: Image-based rendering acceleration and compression. In *Eurographics*, volume 19, pages 321–330, 2000.

[26] C. Zhang and T. Chen. A self-reconfigurable camera array. In *Proceedings of the Eurographics Symposium on Rendering*, June 2004.

[27] L. Zitnick, S. B. Kang, M. Uyttendaele, S.Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graphics*, 23(3):598–606, Aug. 2004.