A Standards-Based, Flexible, End-to-End Multi-View Video Streaming Architecture

Engin Kurutepe*, Anıl Aksay†, Çağdaş Bilen†, C. Göktuğ Gürler‡,
Thomas Sikora*, Gözde Bozdağı Akar†, A. Murat Tekalp‡
*Technische Universität Berlin, Berlin, Germany
†Middle East Technical University, Ankara, Turkey
‡Koç University, Istanbul, Turkey

Abstract— **In this paper we propose a novel framework for the streaming of 3-D representations in the form of Multi-View Videos (MVV). The proposed streaming system is completely standards based, flexible and backwards compatible in order to support monoscopic streaming to legacy clients. We demonstrate compatibility of the proposed system with various possible encoding schemes and operating scenarios. In the current implementation, the MVV's in the server are compressed using a simplified form of MVC with negligible loss of compression efficiency and streamed using Real Time Streaming Protocol (RTSP), Session Description Protocol (SDP) and Real Time Protocol (RTP) to the clients. We describe our extensions to SDP and discuss a preliminary RTP payload format for MVC. The clients in this implementation perform basic error concealment to reduce the effects of packet losses and decode MVC in near-real-time. The modular clients can display decoded 3-D content on a multitude of 3-D display systems.**

I. Introduction

The most important promise of the Internet based media delivery over traditional broadcast systems is the dimension of asynchronicity provided to the users. The ability to watch content at the desired time instead of being bound by schedules is very valuable. This was quickly realized and there has been great amount of research on the Internet-based on-demand streaming of audio and video content. This field has already begun producing successful commercial applications such as various Internet radio stations and on-demand video services such as YouTube and Joost.

On the other hand 3-D video is a very new topic. Although 3-D illusions had people's imagination since the 19th century, the computing and storage capabilities of common computers have only recently reached a necessary level to allow 3-D video applications. As a result there has been very interesting research work on the capture, representation and rendering of real-life scenes under various research programs such as ATTEST project[1] and 3DTV NoE[2] in Europe or FTV[3] project in Japan. These research programs are beginning to produce mature results which can be used to provide interactive 3-D entertainment. However, there has been relatively little research on the transmission aspects of 3-D video.

Single view video plus depth representation is standardized under MPEG as a result of ATTEST work[4], which embeds the compressed depth map to the auxiliary information field of each frame at the cost of increasing the bitrate by about 10-20%. This representation allows generation of stereo video from a single video plus the associated depth map. However, the range of motion offered by such a single view method is very limited due to occlusions. Hence, the need for multi-view representations which fill the disoccluded regions using pixels from other views.

For the streaming of multi-view representations two extreme operating conditions can be imagined: streaming all views in a highly compressed MVC[5] bit-stream with little possibility of random access or encoding all views independently and streaming only the required views. Since MVC bit-stream is H.264/AVC compliant both of these extremes can be streamed over the Internet using existing single view streaming standards. However, there are very good reasons for developing new methods to stream multi-view videos over the internet. Firstly, both extremes can be suboptimal as the rate-distortion results presented in [6] suggest. Secondly, and more importantly, streaming MVC streams over existing single view methods prevents the system from utilizing various multi-view related information and possibilities. Such possibilities are still an open research question and can range from selective streaming such that each client gets only required views to obtaining different views from different sources in a Peer-to-Peer streaming setting to make use of path diversity.

Therefore, our motivation in this paper is to develop a streaming framework for Multi-View Videos (MVV), that will allow flexible and standards based streaming of MVV under various operating scenarios. The proposed system is intended to be a basis for future MVV streaming research and, therefore, is designed to be as flexible and extensible as possible. We will both describe our design decisions and provide details of our current implementation where appropriate.

The remainder of this paper is organized as follows: In Section II, we present overall features of the proposed framework and set it into context. In Section III, we discuss the compatible encoding schemes for the framework and give details of the current implementation. In Section IV, we discuss SDP and RTP related particulars of the system and introduce two new extensions. In Section V, we detail our decoding, error concealment and display implementations and explain how these could be extended. And finally in Section VII, we draw our conclusions and discuss future directions of research.

II. System Overview

In this paper we propose a novel server-client streaming architecture for multi-view videos. The proposed system is built on existing standards and can support several different streaming scenarios, which cover a wide range of possible 3-D applications:

- N-View Streaming for advanced multi view display systems.
- Selective Stereo Streaming and user head tracking in order to provide free viewpoint experience.
- Stereo Streaming with static two views and fixed viewpoint.
- Video plus depth streaming for static stereo viewing.
- Mono-view conventional 2-D Video Streaming for legacy clients.

The server contains the MVV representation in encoded form. A version of Multi-View Coding (MVC) with a simplified spatial reference structure, as described in Section III, is employed in order to provide easier random access within a stream and VCR-like play controls. However, the proposed streaming framework can also work with other encoding schemes, such as independently encoded streams without inter-view dependencies or MVC base layer with independently encoded enhancement streams such as described in [6].

The server and clients use RTSP[7] to negotiate and initiate on-demand 3-D streaming. When a client first contacts the server an SDP[8] announcement describing the available MVV representation is sent from the server. Our proposed framework extends the standard SDP in order to support MVV's as described in Section IV-B. The client determines the streams required according to current streaming scenario and initiates streaming over RTP[9] for those streams as described in Section IV-A.

The received RTP packets are decoded and displayed by the client. The modular client proposed in this paper currently supports various display systems such as N-view lenticular sheet displays, parallax based stereo displays[1] and polarized projection systems and can further be extended to support other display technologies.

III. Multi-View Video Encoding

A. Supported Standards

H.264/AVC Part 10 is the state-of-the-art video coding standard for monoscopic video [10]. Most of the representations of 3D Video are coded using variants of this codec. Simulcast coding uses several streams all encoded by H.264/AVC independently. Video-plus-depth (VPD) representation is coded by independent coding of video and depth signal by H.264/AVC with small information about depth data embedded into the high-level syntax. MPEG specified a corresponding container format "ISO/IEC 23002-3 Auxiliary Video Data Representations", also known as MPEG-C Part 3. MVV is coded by exploiting temporal and inter-view redundancy by interleaving camera views and coding using a hierarchical manner with some MVC specific tools like illumination and color compensation, improved disparity estimation and coding and some high-level syntax changes. MVC is decided to be an amendment (Amendment 4) to H.264/AVC which is scheduled to be finalized in early 2008[11].

[1]Such a display is found on Sharp Actius AL3D Notebook

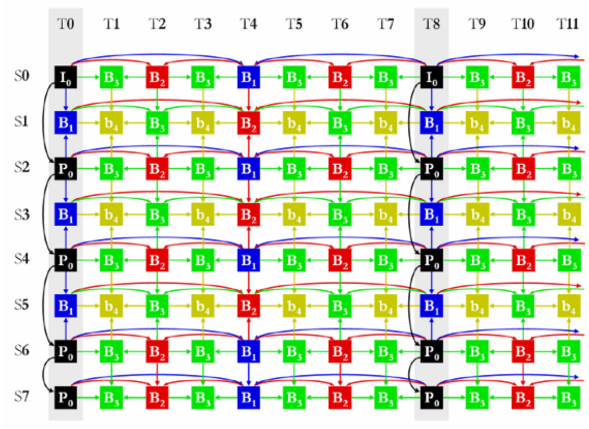
Fig. 1. Standard MVC prediction structure

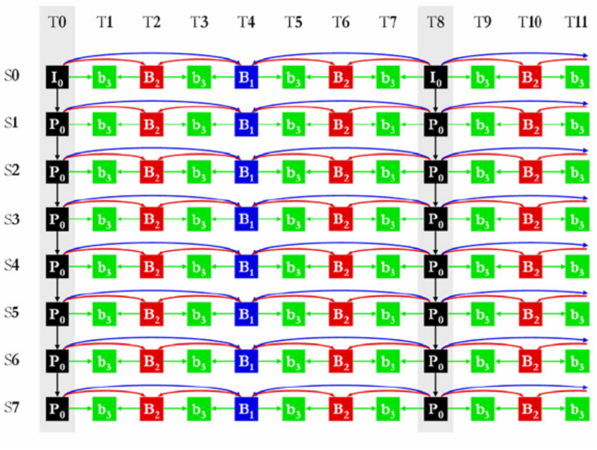
Fig. 2. Simplified MVC prediction structure

B. Multi-View Coding Encoder Implementation

MVC encoder used in our system is JMVM 3.0.2 [12]. This is the reference software for MVC. It uses prediction structure of hierarchical B pictures for each view in temporal direction as shown in Figure 1 [13].

Main prediction structure shown in Figure 1 is quite complex introducing a lot of dependencies between images and views. These dependencies make use of the redundancies present in both spatial and temporal directions to reduce the bitrate, however they also impose many restrictions in decoding and packet loss sensitivity. An alternative simplified structure is presented in [14], and shown to be very close to the main prediction structure in terms of overall coding efficiency. In this simplified

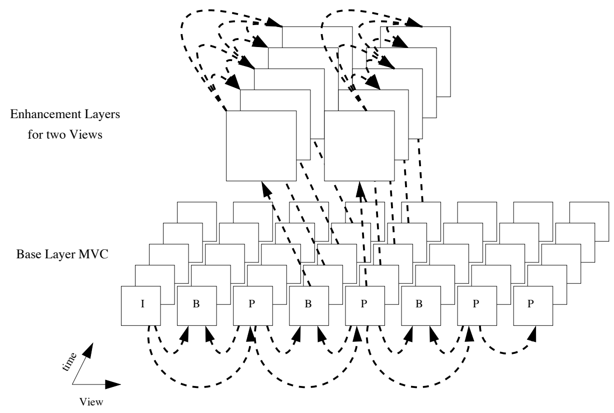

Fig. 3. Scalable MVC coding reference structure. Only two enhancement layers shown for the sake of simplicity

prediction structure the temporal prediction using hierarchical B-pictures remains unchanged when compared to original MVC prediction structure, but spatial references are only limited to anchor frames, such that spatial references are only allowed at the beginning of a group of pictures (GOP) between I and P pictures as shown in Figure 2. Since the proposed system intends to support different types of clients, this simplified prediction structure is adopted due to its more flexible decoding properties. Due to the lack of dense spatial references in this prediction structure, the multi-view video server or media aware network elements (MANE) further down the network can easily adapt the MVC stream to the needs of the clients and/or network conditions by leaving out the unneeded NAL units. Therefore, each client receives only the necessary NAL units in order to decode their requested views. A Multi-view client will receive all the packets, whereas a stereo client would receive only two requested streams and the anchor frames for the other streams which are references for the requested two streams. Similarly a mono client would receive only a single requested stream plus the necessary anchor frames for decoding. On the other hand a legacy client which does not understand the extensions described later in this paper would only receive the independently encoded and H.264/AVC compatible stream from the complete MVC representation.

In the case where the system is also required to support dynamic selective streaming clients, which track user movements to predict the required streams, the encoding structure proposed in [6] can be used. This scheme as shown in Figure 3 consists of a lower bit-rate base layer MVC, which is received by all clients, in addition to enhancement layers for each view, which independently coded from each other. When this coding scheme is employed monoscopic clients only receive the base stream from the base layer MVC plus optionally the corresponding enhancement stream. Stereo clients receive base layer MVC plus two enhancement layers of their choice. In this encoding scheme, the views in the base layer provide a kind of an insurance when wrong enhancement streams are streamed due to errors in the head tracking system. And finally Multi-view clients receive the base layer MVC and all enhancement layers.

IV. MVV Streaming

A. H.264/AVC over RTP

MVC builds upon the scalable extensions of H.264/AVC standard, therefore in order to stream MVC over RTP, existing work and standards should be employed as the basis any novel system in order to obtain a flexible and backwards compatible solution.

The streaming of H.264/AVC streams over RTP is standardized by the IETF in RFC 3984[15], which defines the RTP header usage and necessary packetization rules for H.264/AVC. RFC 3984 is perfectly applicable for the streaming of MVC bit streams as a whole, because they are H.264/AVC compatible. However, this approach leaves out the possibility of adapting the MVC bitstream to the receiver's needs and capabilities.

We propose to stream MVC over RTP in a similar fashion to [15]. However, instead of sending the MVC stream as a single H.264/AVC stream over RTP, we break up the stream into several parts. NALUs belonging to different views are streamed over different RTP port pairs as if they were separate H.264/AVC streams. Although only the stream for the independently encoded view will be decodable by itself, this scheme allows the receiver to select which parts of the whole MVC stream to receive. The server neither imposes any restrictions nor issues any warranties on the decodability of the packets received by the client: the client is responsible for selecting a correct set of streams such that the received packets can be decoded. In other words, this means that a mono client who is interested in watching a view, must request the stream for that view and any other streams which are referenced by the NALUs in the stream for that view in order to receive a correctly decodable set of NALUs. The information required in order to select the correct set of streams should be provided before the start of the streaming session through out of band methods. An example of such a method is detailed in the following section.

Additionally, in order to save further bandwidth, instead of sending whole reference streams the server can also send only the referenced NALUs in the reference streams. Due to the simplified MVC prediction structure employed by the proposed system, this can be achieved by only sending the anchor frames. We call such a stream an anchor-only stream. It should be noted, however, for the optimal disk access performance of the server, the anchor-only streams are best stored in separate anchor-only files which increase the number of total stored bits for the whole MVC representation at the server but have no effect on the transmitted bitrate. Any available anchor streams should also be declared before the streaming session begins, as described in the following section for instance.

B. *SDP Extensions for Multi-View Video*

The Session Description Protocol (SDP) is one of the standardized methods to announce on-demand content available on a server. It provides a simple and extensible framework to describe the detailed properties of the content, enabling the initialization of the data connection and decoder structures at the client. An SDP announcement consists of a list session-level section followed by zero or more media-level sections. In addition to media dependent information such as RTP clock rate, sequence number and timestamp base values, the media-level sections contain necessary information to start RTSP streaming and establish the RTP/RTCP connections to the server.

In the proposed system, each view contained in the MVV is declared as a separate media-level section in the SDP announcement in order to enable the receivers to freely select the views they would like to receive. Although the MVV is compressed to a single file after MVC encoding, we propose to break this single file into separate streams for each view on the disk. This has no effect on the total file size and facilitates fast disk access to the NAL units belonging to different views. Obviously, most of these streams will not be independently decodable due to the inter-view dependencies generated by MVC compression. In order to define such dependencies in SDP announcements a dependency variable has already been suggested in [16], which introduces two different types of dependencies for layered streams to handle either hierarchically layered streams such as SVC or non-hierarchical layered streams generated by multiple description coding approaches. However, in a multi-view video stream there can be further types of dependencies which should be declared explicitly. Here we introduce an extension to [16] with three new types of references between streams.

First new type of dependency is the MVC decoding dependency, which is analogous to the layered SVC stream dependency case in [16]. This dependency is declared as `a:depend=mvc ID_LIST` for all streams part of the MVV representation, where `ID_LIST` is a space separated list of view ID's, where a MVV with 9 views could have view ID's from 0 to 8. The view IDs in this list correspond to the immediate dependencies of the stream in question. In order to make sure that the received NALU's are indeed decodable, a client needs to obtain the all dependencies of those reference streams in a recursive fashion as well.

The second and third dependency types assign depth and anchor streams to their corresponding views, with the help of the view ID's. Although there is no syntax based decoding dependency between such streams, there is a clear semantic dependency, as a decodable depth stream, for instance, is of little value without the video stream it belongs to. These types of dependencies are declared as `a:depend=KEYWORD ID`, where `KEYWORD` is `depth` and `anchor` for depth and anchor streams respectively and `ID` is the view ID of the corresponding view.

This method of declaring dependencies is completely backwards compatible. In accordance with original SDP specification, a client which does not understand the proposed SDP attributes, it simply ignores them. However, such a client might request and fail to decode dependent streams due to its inability to use the dependency information. Similarly, a 3-D client can still display in 2-D if it is connected to a conventional video server with no support for the proposed SDP attributes.

V. Real-Time MVV Decoding and Display

A. *Real-Time MVV Decoding*

We are currently using FFMPEG library [17] for real-time decoding of H.264/AVC streams. In order to decode MVC streams, we modified FFMPEG library with the appropriate changes.

First of all, DPB (Decoded Picture Buffer) size is increased since MVC prediction requires more pictures to be held in the buffer. There are some modifications in SPS (Sequence Parameter Set) to signal for the prediction structure and to signal the MVC encoding and modifications in NALU header to signal for the view ID and related other information. Since prediction of each frame and memory management depends on view ID of the frame, view ID tag is added to each frame in DPB and related functions such as prediction list generation, buffer management are modified accordingly. Support for new picture reordering commands is also implemented to modify the list more efficiently for inter-view prediction. Since MVC required the base-view to be standard compatible, view ID and related other information cannot be signaled in the NALU header of the base-view NALU. It is decided to have another NALU following each NALU of base-view with the required information which is called suffix NALU. Also for compatibility reasons, B-pictures that are marked as non-reference for the base-view, can be used for inter-view prediction and needs to be stored in DPB. Finally illumination compensation is added into the motion/disparity compensation module.

Although normal decoding requires all the views in the MVC stream, MVC decoder can decode only selected views according to the prediction structure. Monoscopic client needs only NALU of base view (S0) and decodes only base view. Similarly, stereo client decodes only first two views (S0 & S1) and multiview client decodes all of the views (S0 to S7).

After the packets received from the network, they are placed into a buffer. Before MVC packets are fed into MVC decoder, they need to be synchronized. All packets need to be ordered in decoding order according to both decoding time and decoding view. Decoding view order depends on the prediction structure of the encoder. For the scheme in Figure 1, decoding view order is S0-S2-S1-S4-S3-S6-S5-S7 and for the scheme in Figure 2, it is S0-S1-S2-S3-S4-S5-S6-S7. Also for the base-view (S0), suffix NALU is required to be after the NALU of this view.

In order to maintain the decoding order in both dimensions and detect packet losses, a buffer called "Tetris NALU Buffer" is used as shown in Figure 4. The name

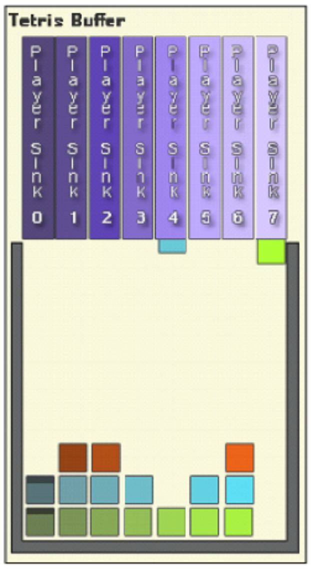

Fig. 4. The Tetris NALU Buffer. Player Sink objects are responsible for extracting packets for different views from the RTP stream(s).

comes from the fact that decoder is only fed with NALU packets, when the packets from each view is received. Similar to the game over in Tetris, buffer overflow occurs, when one of the packet is missing in the line and packets are filling some of the vertical line. In order to avoid buffer overflow, after a predefined timeout duration, half-full lines are fed into the decoder and error concealment methods are employed.

B. MVC Error Concealment

Error concealment methods implemented for the codec is based on non-normative error concealment algorithms for monoscopic H.264/AVC[18]. These algorithms are modified for the changed prediction structure in MVC.

Losses can be categorized into two: Slice losses and frame losses. Slice losses occur when frames are coded into multiple slices and some of the slices are lost. In such cases, using neighboring macroblock information like motion vectors and/or pixel values is employed [18]. Depending on the type of the frame, concealment approaches change as well.

In case of slice losses, the processing order is chosen to take the macroblock columns at the edge of the frame first and then move inwards column by column so to prevent concealment mistakes from propagating. For intra coded frames spatial concealment is applied. Pixels in the blocks are interpolated using a weighted average of the neighboring pixels as shown in Figure 5.

For inter coded frames, motion vectors of the neighboring blocks are used as a possible candidate for the motion vector of the lost block. Among the possible candidates and zero motion vector, a boundary matching algorithm (BMA) is applied to find the block that smoothen the boundary between the lost block and the neighbors. Selection of candidates and BMA is shown in Figure 6. When candidate motion vectors are selected, blocks with motion

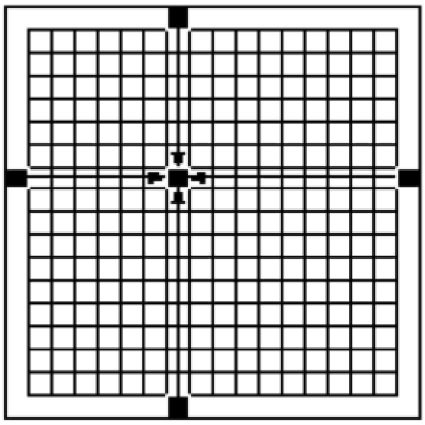

Fig. 5. Spatial concealment based on weighted pixel averaging

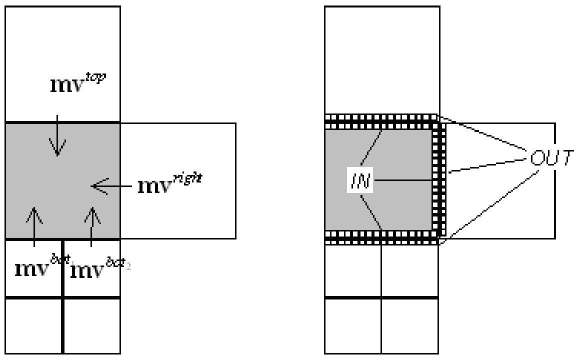

Fig. 6. Selection of the motion vectors for prediction

vectors that uses previous frames from the same camera is used. If no such block exists, then spatial concealment is used.

In case of frame losses, above algorithms do not apply since no neighboring block is available. In such cases, copying previous frame from the same camera is used.

C. MVV Display

Depending on the type of display, both the number of streams required and the way the images are displayed on the screen change. For N-view lenticular sheet displays and parallax based stereo displays, views from the same time instant needs to be combined to be shown on the display. This combination process is called "interzigging"[19], where N-views from WxH size images are combined to generate a single image of size W x H. $f(x, y, color\_channel)$ is a function defining for each pixel and each color channel which camera view will be used and changes according to the display type. For instance, if $f(50, 70, 1) = 5$, then the red component of the pixel in 50th column and 70th row will be taken from the 5th camera's red component of the corresponding pixel location. Interzigging module will combine N-views and generate the required output in RGB domain. Output of this module will be shown on the screen according to the timestamp of the image. The interzigging operation effectively reduces the resolution of each decoded image and combines them in such a way, that the resulting image is spatially multiplexed in an optimal fashion by the lenticular array and the user experiences the intended 3-D perception.For displays using shutter glasses no interzigging is necessary and the display module displays two images at different time instants. Similarly for polarized projection systems no interzigging is performed and images for left and right eyes are sent to corresponding and precisely aligned projectors which project the images to the same physical area on a polarization preserving screen.

VI. Results

An actual demonstration implementation has already been recently presented at IBC 2007 in Amsterdam. In the demo implementation, the Multi-View Server contained several multi-view sequences with and without depth maps. Three different clients with different display systems (Single View Legacy Client, Sharp Stereo Laptop with parallax barrier display and Multi-View Client with 9-view lenticular sheet display) connected to the server and requested a random sequence. In the case of stereo and mono clients, a random view from the multi-view sequence was selected as well. The reference streams for the requested streams were found using the dependency information contained in the SDP and those reference streams were requested in order to ensure decodability. We presented sequences both with and without separate anchor-only streams. The demonstrated multi-view client was able to decode and display 9-view multi-view sequences in real time on a standard high-end laptop.

VII. Conclusions

In this paper we have introduced a new framework for streaming of Multi-View Videos. The most important features of the proposed framework are its flexibility and standards compatibility. Therefore the system described in this paper can be implemented using currently existing standards with some well defined extensions. We hope that this methods will form the basis of future discussion and research. In addition to the streaming issues, we have also detailed encoding, decoding, error concealment and display methods for an example implementation of the proposed framework. We believe the proposed MVC payload format as separate H.264/AVC streams and SDP extensions for the declaration of different types of dependency relations between the streams contained in a multi-view sequence can prove to be good starting points for further standardization process.

Acknowledgments

This work is supported by EC within FP6 by VISNET II Network of Excellence and 3DTV Network of Excellence under Grant 511568.

The authors would like to thank Atanas Boev from Technical University Tampere for his help with the interzigging process.

References

[1] C. Fehn, P. Kauff, M. de Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton, "An Evolutionary and Optimised Approach on 3D-TV," *Proc. of IBC*, 2002.

[2] L. Onural, T. Sikora, and A. Smolic, "An overview of a new european consortium: Integrated three-dimensional television–capture, transmission and display (3DTV)," *Proceedings of European Workshop on the Integration of Knowledge, Semantics and Digital Media Technology (EWIMT)*, 2004.

[3] M. Tanimoto, "Free viewpoint television-ftv," *Picture Coding Symposium 2004*, pp. 15–17.

[4] C. Fehn, "Depth-image-based rendering(DIBR), compression, and transmission for a new approach on 3 D-TV," *Proceedings of SPIE*, vol. 5291, pp. 93–104, 2004.

[5] K. Mueller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic, T. Oelbaum, and T. Wiegand, "Multi-view video coding based on H.264/AVC using hierarchical B-frames," in *Picture Coding Symposium 2006*. PCS, 2006.

[6] E. Kurutepe, M. R. Civanlar, and A. M. Tekalp, "Client-driven selective streaming of multi-view video for interactive 3DTV," *IEEE Transactions on Circuits and Systems for Video Technology*, 2007.

[7] H. Schulzrinne, A. Rao, and R. Lanphier, "Real time streaming protocol (rtsp)." [Online]. Available: http://www.ietf.org/rfc/rfc2326.txt

[8] M. Handley and V. Jacobson, "SDP: Session description protocol." [Online]. Available: http://www.ietf.org/rfc/rfc2327.txt

[9] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications." [Online]. Available: http://www.ietf.org/rfc/rfc3550.txt

[10] I. Rec, "H. 264 & ISO/IEC 14496-10 AVC, Advanced video coding for generic audiovisual services," *ITU-T, May*, 2003.

[11] A. Vetro, P. Pandit, H. Kimata, and A. Smolic, "Joint Draft 3.0 on Multiview Video Coding," *Joint Video Team, Doc. JVT-W209*, 2007.

[12] P. Pandit, A. Vetro, and Y. Chen, "JMVM 3 software," *ITU-T JVT-V208*, 2007.

[13] H. Schwarz, D. Marpe, and T. Wiegand, "Analysis of Hierarchical B Pictures and MCTF," *Multimedia and Expo, 2006 IEEE International Conference on*, pp. 1929–1932, 2006.

[14] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, "Comparative Study of MVC Prediction Structures," *ITU-T JVT-V132*, 2007.

[15] S. Wenger, M. M. Hannuksela, T. Stockhammer, M. Westerlund, and D. Singer, "RFC 3984: RTP payload format for H.264 video." [Online]. Available: http://tools.ietf.org/html/rfc3984

[16] T. Schierl and S. Wenger, "Signaling media decoding dependency in Session Description Protocol (SDP)," 2007. [Online]. Available: http://tools.ietf.org/wg/mmusic/draft-schierl-mmusic-layered-codec-03.txt

[17] Ffmpeg homepage. [Online]. Available: http://ffmpeg.sourceforge.net/

[18] V. Varsa, M. Hannuksela, and Y. Wang, "Non-normative error concealment algorithms," *ITU-T VCEG-N62*, vol. 62, 2001.

[19] L. Lipton and M. Feldman, "A new autostereoscopic display technology: The SynthaGram," *Stereoscopic Displays and Virtual Reality Systems IX, Andrew J. Woods, John O. Merritt, Stephen A. Benton, Mark T. Bolas, Editors, Proceedings of SPIE*, vol. 4660, pp. 229–235, 2002.