Image-space Free-viewpoint Video

Stephan Würmlin, Edouard Lamboray, Michael Waschbüsch
Peter Kaufmann, Aljoscha Smolic[†], Markus Gross

Computer Graphics Laboratory
ETH Zürich, Switzerland

[†]Fraunhofer Institute for Telecommunications
Heinrich-Herz-Institute, Germany

Abstract

Image-space free-viewpoint video (FVV) is a framework for representation and coding of sparse multi-view video and subsequent re-rendering from arbitrary viewpoints. It is based on the fundamental concept of storing all information describing a scene's visual appearance in multi-channel video images. Each pixel's channels define different attributes of discrete point samples of observed surfaces. In this paper, we show how image-space FVV can be implemented with standard video coding tools and ready-available video coding methods can thus be reused. Besides, this representation has been adopted as extension of the MPEG-4 AFX standard. In addition to the representation, we also introduce a novel video coding technique that allows random access and progressive transmission. We evaluate this new codec in the context of FVV and compare it to state-of-the-art image and video codecs like MPEG-4 and JPEG-2000. We present results based on real-world data proving the suitability of the chosen approach.

1 Introduction

In the face of recent research activity, the field of dynamic visual media enabling free viewpoint selection draws more and more attention. The prospect of free navigation regarding time and space in streams of visual data might even represent the next major step in terms of interactivity, allowing for virtual replays just as for the well-known freeze-and-rotate effects. This motivates to face the challenges in designing a representation that enables capturing and processing of dynamic scenes and subsequent rendering from arbitrary viewpoints. 3D-television [16] marks a first line of recent research, aiming at view-independent video for dynamic scenes but in a very limited range only. That is, users might experience changes in parallax but no fly-around effects are possible. The concept of free-viewpoint video, on the other hand, allows for truly free navigation in the spatial range of captured data, i.e. in the range covered by the acquisition cameras. The format we present is a well qualified representation for FVV. Because it encodes a scene's complete appearance in a stream of images using multiple channels we refer to it as being an image-space representation. The different channels record all relevant attributes of a scene being acquired by exploiting the notion of point samples. That is, all channels of each video pixel describe attributes of discrete 3D point samples. An image-space FVV representation features many advantages as compared to other descriptions. Firstly, it may be understood as a unified representation also amenable to topological changes of the scene's geometry—quite contrary to approaches based on mesh and texture information that require handling of heterogeneous types of data and do not allow changes in topology. Secondly, being an image-space representation, well proven conventional video coding schemes are applicable—a fact reflected by the MPEG industry standard supporting our representation as an extension of MPEG-4 AFX. Moreover, since our representation incorporates geometrical scene knowledge in terms of point samples we have to deal with less acquisition cameras for even broader viewing ranges as compared to purely image-based approaches in the spirit of Light Fields [13]. Note that the latter technique may successfully be extended as appropriate representations for 3D-television. Besides all advantages given, our image-space FVV framework is able to deal with every static or dynamic 3D data set which can be completely described by a set of 2D views. Furthermore, the framework can smoothly be extended to an arbitrary camera setup where only a few cameras see the object at a time.

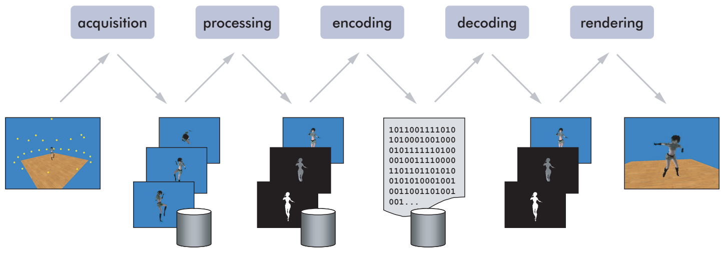

Figure 1: Overview of the image-space free-viewpoint video pipeline.

2 Background and Related Work

In free-viewpoint video, multi-view video streams are used to re-render a time-varying scene from arbitrary viewpoints. There is a continuum of representations suited for different acquisition setups and applications. Purely image-based representations [13] need many densely spaced cameras. This constraint can be relaxed by adding more and more geometry in terms of depth maps which lead to depth image-based representations, e.g. [1]. On the other end there are model-based representations which describe the object or scene by a time-varying triangular mesh with additional video textures, e.g. [5].

Besides representations one has to distinguish between online and offline applications. Matusik et al. [14, 15] focused on real-time applications, e.g. 3D video conferencing. However, they did not address the coding of their representations. Gross et al. [7] used a 3D video system based on a point sample representation [21] for their telecollaboration system *blue-c*. Their temporal coding scheme is in a sense a depth-map compression but with a focus towards real-time encoding. Mulligan and Daniilidis [17] also target telepresence. They compute geometric models with multi-camera stereo and transmit texture and depth over a network. Offline free-viewpoint video systems typically process triangular meshes. Carranza et al. [5] employ an a-priori shape model which is adapted to the observed outline of a human. Therefore, only motion parameters and video images need to be transmitted. However, this system is only able to capture pre-defined shapes and no scene acquisition is possible. Geometry videos [3] are another interesting representation for compression of dynamic 3D objects. Geometry is reorganized into 2D images before deploying conventional video compression. However, the mapping of a 3D mesh to a 2D image is non-trivial. The 3D video recorder by Würmlin et al. [22] handles point-sampled 3D video data which is encoded using hierarchical space partitioning. But the reported bit rates are well above our target rates.

3 Image-space Free-viewpoint Video

In this paper, we present a free-viewpoint video system where data is represented in image- or camera-space. Consequently, we can employ coding schemes from image or video compression. The image-space framework extends depth image-based representations towards a point-based representation. This means that although the data is represented in image-space each pixel basically corresponds to a discrete point sample of the captured surfaces. Therefore, the point samples constitute a dynamic point cloud of the objects in the scene. This allows for arbitrary topology changes which is not trivially handled using other representation, e.g. triangular meshes. For our free-viewpoint video system multiple video streams are synchronously recorded. After capturing we process the images to have not only color information for each pixel but also depth, and optionally normal and splat information. The latter two attributes are used for high-quality point rendering. These multi-attributed video images are then encoded using image or video coding schemes and stored to disk. During playback free-viewpoint video sequences can be streamed and decoded over the network or from local disk and displayed on screen using high-quality point-based rendering. For this purpose, our framework allows for view-dependent decoding and blending of the different camera images.

3.1 Data Acquisition and Processing

Acquisition of real-world data for free-viewpoint video typically captures multiple concentric or parallel video sequences of the same scene. All cameras have to be calibrated beforehand, i.e. the intrinsic and extrinsic parameters of each camera are known. The video sequences of the multiple cameras are recorded with synchronized cameras. When acquiring 3D video objects we first perform a background segmentation algorithm to calculate for every input frame an image mask telling which pixels belong to the object of interest, i.e., are foreground pixels, and which pixels belong to the background.

After the input images have been processed by a 3D reconstruction algorithm, each input frame provides for each foreground pixel a depth value describing, in combination with the camera calibration parameters, the geometry of the object, and a color value. Currently we use a shape-from-silhouettes method based on the image-based visual hulls algorithm [15] to extract the depth information. For each foreground pixel, a surface normal vector and a splat size can be stored as optional attributes. This data can be computed for example by the approach of [18]. In general, with the data representation presented in the next section it is possible to encode any attributes describing the visual appearance of an object or scene. Given a specific rendering scheme or target application, any subset of the above pixel attributes might be sufficient.

3.2 Data Representation

A key component for efficient compression and rendering of free-viewpoint video is the underlying data representation. An image-space data representation allows to use conventional video coding algorithms for compressing the time-varying data. In our free-viewpoint video coding framework we basically structure all point attributes in separate images or separate channels of a (video) image. This representation was adopted by the MPEG committee as an extension to the MPEG-4 *Animation Framework eXtension* (AFX) for representing static and dynamic point-sampled geometry [10].

AFX aims at providing 3D formats for naturally looking scene objects. In the following, we present the *depth image-based representation* (DIBR) which is the standardized format for the image-space data representation. Version 1 of DIBR introduced depth images as representation for still and animated 3D objects [8]. A 3D object or scene is represented as a set of reference images completely covering its visible surface. This data is usually accompanied by some kind of information about the object geometry. To this end, each reference image comes with a corresponding depth map, an array of distances from the pixels in the image plane to the object surface. Rendering is achieved by either forward warping or splat rendering. But with Version 1 of the specification of DIBR nodes no high-quality rendering, as suggested in Section 3.3, can be achieved.

For Version 2 of the specification of DIBR nodes we extended the initial concepts for high-quality point-based rendering. For this purpose, the definition of depth images were extended by the fields *normal*, *splatU* and *splatV*. Since the latter two fields are constrained to 8 bits, *splatMinMax* specifies the minimum and maximum splat vector lengths. Thus, both *splatU* and *splatV* fields have to be scaled to the interval defined by the *splatMinMax* field. The *normal* field specifies the normal vector for each pixel in the texture. The normal vector should be assigned to the object-space point sample derived from extruding the pixel with depth to 3-space (by using the camera calibration parameters already included in the depth image). The *splatU* and *splatV* fields specify the local tangent plane and reconstruction kernel needed for high-quality point rendering. The next section will explain how we use these three fields to enable high-quality point rendering. If the fields are not specified, the decoder can calculate them on-the-fly during rendering. Refer to [8, 10] for detailed information on DIBR Version 1 and Version 2 nodes. Figure 2 illustrates our representation combining multiple *depth image*-nodes describing the different attributes.

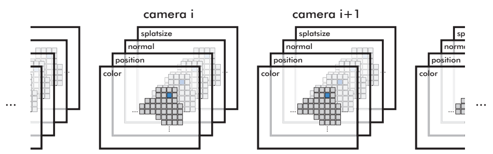

Figure 2: Illustration of the image-space data representation for free-viewpoint video.

An example of all fields can be found in Figure 3. Note that the silhouette or shape can be coded implicitly in the alpha channel of the texture field which then can be used for shape-adaptive coding. Explicit shape coding by coding separate images is also possible.

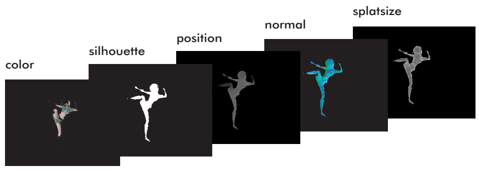

Figure 3: Example of a DIBR node.

When using the MPEG video format the normal and splat fields can be encoded in the auxiliary images. MPEG-4 video allows for a maximum of three auxiliary images, each auxiliary image consisting of an 8 bit grayscale image. The normal codecs proposed in [6, 2] allow to quantize surface normals on 3 to 15 bit words. Würmlin et al. observed that a precision of 7 bits is sufficient for many applications [22]. For circular splats, the splat sizes can also be efficiently quantized on 8 bit words. Thus, if an explicit encoding of surface normal vectors and splat sizes is required, these attributes can be encoded using two of the three optional auxiliary images. However, as for the depth values, the statistics of the normal and splat size codewords are different from the statistics of color data in video sequences. Thus, we cannot expect the same performance from the MPEG compression algorithms for normal and splat size data than for color data. Furthermore, the lossy encoding does not allow to reproduce exactly the same codewords at the decoder. Hence, the MPEG encoding of normal vectors and splat sizes may lead to artifacts in the rendering.

3.3 Free-viewpoint Video Rendering

High-quality rendering is based on the notion of point-sampled surfaces as non-uniformly sampled signals. Point-sampled surfaces can be easily constructed from the DIBR nodes by projecting the pixels with depth into 3-space. The discrete signals are rendered by reconstructing and band-limiting a continuous signal in image space using so called resampling filters [24]. To this end, we have to associate a 2D reconstruction kernel r_k with each sample point p_k. The kernels are defined in a local tangent frame with coordinates u and v at the point p_k, as illustrated in Figure 4. The tangent frame is defined by the splat and normal fields of the DIBR structures Version 2. The kernels are rendered as overlapping elliptical disks spanned by the u and v axes of the tangent frame. To get a continuous approximation of the surface the overlaps are blended in image space by associating a gaussian alpha texture with each kernel.

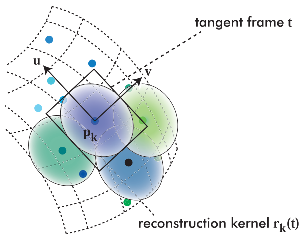

Figure 4: Illustration of local tangent planes and reconstruction kernels for high-quality point rendering.

Because only a part of the surface is generally visible from the current viewpoint there is no need to decode and render all points from all available camera images. Due to anisotropic material properties of the object, the texture provided by a single acquisition camera is only accurate for views close tho the real acquisition viewpoint. Therefore, we apply a view-dependent rendering technique by only displaying the points from those so-called *rendering-active cameras* that are closest to the current virtual viewpoint. We determine the set of rendering-active cameras on the fly whenever the view changes. To avoid discontinuities during viewpoint transitions, which would appear as popping artifacts in the rendered video, we associate a weight with each camera describing the impact of its points to the rendering. Thereby, we achieve a smooth blending by using the weight to scale the amplitude of the gaussian alpha mask of each reconstruction kernel.

One way to compute the camera weights is given by Unstructured Lumigraph Rendering [4]. However, we noticed in our experiments that this approach does not produce the best possible results for our sparse camera setting. Due to the highly non-linear blending, the transitions between two rendering-active cameras is not seamless. We found that a weight computed by the dot product between the virtual and real viewing directions produces smoother results. In a concentric camera setup, the simple consideration of the viewing directions is sufficient. Figure 5 and the accompanying video illustrate camera weighting.

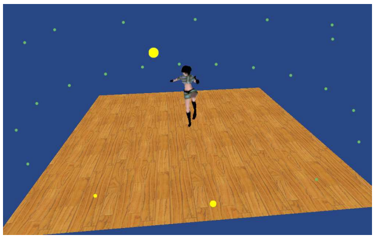

Figure 5: Illustration of view-dependent camera weighting. Green dots indicate the inactive and yellow dots the rendering-active cameras. The size of the circle corresponds to the camera rendering weight for the virtual viewpoint of this rendering.

3.4 Coding Requirements

The 3DAV ad-hoc group (for 3D audio/visual) of MPEG defined requirements for free-viewpoint video [9]. Our data representation fulfills some of these requirements directly. Besides straightforward requirements the most important are multiple views, integration with SNHC computer graphics, disparity information, occlusion handling, and different display types. Partial or view-dependent decoding of the data is directly possible due to the nature of the data representation. Since the number of cameras can be large and thus a huge data volume is available it is not realistic to decode and render all cameras in real-time. Hence, before decoding we have to choose a subset of the cameras which is needed to render an image for a given virtual viewpoint. For this purpose we employ the algorithm of [21], i.e. given a viewpoint and the camera calibration data, we compute the contributing cameras and read the data accordingly before decoding. To this end, a feedback channel is required. Furthermore, a coding method for image-space free-viewpoint video should address the following features for requirements like scalability, random access, and interactivity:

- **Multi-resolution.** Scalability and progressivity with respect to resolution. This feature can be achieved using a progressive encoding of the data.
- **Multi-rate.** Scalability with respect to time, i.e., the playback of the sequence is possible at a different frame rate than the recording frame rate. Backward playback should also be possible.
- **Random access.** A codec can access a specific frame in constant time.
- **Full-frames.** Besides handling video objects a codec should also handle full-frame images to represent scenes.

The random access property is essential for unconstrained free-viewpoint video as defined by Lamboray et al. [12]. They propose to distinguish between *constrained* and *unconstrained* free-viewpoint video. In the constrained case, the 3D data can be rendered from any possible direction after real-time decoding, but either only small viewpoint changes are allowed during rendering, or discontinuities in rendering are tolerated in presence of large viewpoint changes. In the unconstrained case, such discontinuities are minimized during rendering and the spatio-temporal viewpoint trajectory during playback can be arbitrary.

In addition, the codecs must enable encoding of all attributes for the image-space framework, i.e., color, depth, optional normal and splat size. Furthermore, a codec should be able to decode multiple multi-attributed video frames in real-time. As side information for each camera intrinsic and extrinsic calibration parameters need to be available at both encoder and decoder.

4 Image and Video Coding

A great variety of image and video coding methods are available today, further are under development. Among those, we concentrate on standard formats developed by MPEG and JPEG, since we target interoperable systems and these standards represent the state of the art in image and video coding. We selected a few standard codecs which are described in more detail below and which satisfy our specific requirements best. *Scalable video coding* has also been widely studied [20, 23], resulting for instance in a standard known as MPEG-4 FGS. However, the compression efficiency is very limited compared to it's non-scalable counterparts and the level of scalability is also quite restricted. *H.264/AVC* is a recently finalized standard developed jointly by ISO/MPEG (MPEG-4 Part 10, Advanced Video Coding) and ITU/VCEG and represents the state of the art in video coding. However, it only supports conventional rectangular video and therefore shape information has to be encoded additionally. Our early tests proved that H.264/AVC is not competitive to shape-adaptive MPEG-4 for 3D video objects. Therefore, we refrained from taking H.264/AVC into our evaluation. In Section 4.2 we review a non-standardized coding scheme which however supports progressive encoding and full random access.

4.1 Image and Video Coding Standards

MPEG-4 Core Profile. The MPEG-4 Core Profile is the state of the art codec with support for arbitrarily shaped video objects. Shape is an inherent requirement of our system as described before and, hence, we employ the shape coding part (MPEG-4 Shape Only Object). For random access support we apply intra-only coding, i.e. all frames are coded in intra mode, without any temporal prediction. In our comparison we denote the intra-only coding in MPEG-4 as *MP4-I*.

JPEG-2000. J2K represents the state of the art in still image coding. It provides highest compression efficiency and full spatial and quality scalability, due to wavelet transform coding. Therefore J2K is the reference method for our evaluation satisfying all requirements. As mentioned before we combine it with MPEG-4 shape coding.

4.2 Average Coding

Since the conventional video coding schemes—based on temporal prediction and motion compensation—do not optimally support the coding of unconstrained free-viewpoint video, we investigate the use of temporally averaged information in reference frames and encode the difference between the original frame and the reference frame in the corresponding delta frames. The reference frame thus contains a prediction value for every foreground pixel for the respective attribute and thus provides already a good approximation of the target frame. The remaining differential information in the delta frame should allow for a better compression than the original frame. In our experiments we averaged information over 5 frames. Within this approach, the codec can access each frame in constant time and thus fulfills the desired properties for unconstrained free-viewpoint video coding.

Progressive Average Coding (PAC). Apart from the silhouettes which require lossless encoding, e.g. lossless MPEG-4 binary shape coding can be used [11], the free-viewpoint video average coder can be implemented using progressive image coders. We implemented progressive attribute codecs for the depth and color channels based on the embedded zero-tree wavelet algorithm [19]. In this case, we encode the depth information in one grayscale image. The color data is encoded in the YUV 4:2:0 format and appropriate ratios of the respective channels are decoded in order to match the desired bit rate. Progressive average coding is introduced in [12].

5 Comparison and Evaluation

We evaluate our free-viewpoint video coding framework using a real-world data sequence which was recorded in our acquisition stage comprising 16 cameras. The sequence was recorded at the resolution of 640x480 pixels and at 25 frames per second with a total length of 250 frames. As can be seen in Figure 7, our acquisition stage is part of a telepresence environment. In this multipurpose installation the cameras are embedded behind glass panels. The narrow spyholes in the wainscot explain the dark areas in the lower left corners of the camera images in Figure 7. In our test sequence the user moves in a range of approximately 2 meters. Since we use a uniform 8-bit quantization of the depth values, the sole quantization error is in the order of ± 4 millimeters. Figure 5 and the accompanying video shows additional results using a synthetic test sequence provided by MPEG. The input images of this sequence have a resolution of 320x240 processed at 25 frames per second with a total length of 200 frames.

Figure 10 a) illustrates view-dependent decoding of reference camera images. We locked the camera selection for the virtual viewpoint of Figure 7 and rotated the viewpoint by approx. 90 degrees. It can be seen that we only decode and render surface points needed for the virtual viewpoint. However, the decoded representation can carry redundant information when the same surface point is visible in all decoded views. Elimination of such samples would introduce high frequencies (holes) in the images and thus result in lower coding efficiency. Figure 10 b) shows a visualization of the point sample representation. We scaled down the splat sizes of the rendered surface points for Figure 7. No connectivity between the points is needed. But for calculating local tangent planes and reconstruction kernels at each point sample the corresponding normals are required. Figure 10 c) shows the estimated surface normals.

Table 1 summarizes again all evaluated coding schemes with their supplied coding requirements for image-space FVV. Figure 8 illustrates the average PSNR values obtained from the evaluated coding schemes. Note the different scales of the PSNR values. The experimental results from color and depth coding are presented for various bit rates. We devised per camera target bit rates of 128, 256, 384 and 512 kbps and allocated the available rate such that color uses twice as much bits than depth. The shape is encoded separately and accounts for approximately 25 kbps per camera. The rendered images in Figure 9 are composed of two reference cameras, so the total bit rate for these images are 306, 562, 818 and 1’074 kbps.

	Multi Res	Multi Rate	Random Access	Full Frames
MP4				✓
MP4-I		✓	✓	✓
PAC	✓	✓	✓	
J2K	✓	✓	✓	✓

Table 1: Supplied coding requirements of evaluated codecs.

As can be seen clearly in Figure 8, MPEG-4 Core Profile is superior to all other codecs because of its shape-adaptive nature. This gets also reflected in the rendered FVV images in Figure 9 e) and i). If neither progressive decoding nor multi-rate or random access features are desirable this is the best choice for image-space FVV. If multi-rate and random access is needed, e.g. for slow motions or backward play, MPEG-4 intra-only coding should be considered as can be seen in the PSNR values in Figure 8 and Figure 9 d) and h). An interesting observation is the fact that intra-only coding of depth values performs better than motion-compensated coding. Both MPEG-4 codecs can handle full-frame video but for full-frame intra-only coding it is assumed that JPEG-2000 (J2K) will be superior. Progressive average coding (PAC) and J2K both have random access and progressive decoding capabilities. For 3D video objects PAC performs better as can be seen again in Figure 8.

Rendered images with J2K coding exhibit significant visual artifacts at the object boundary. This is due to coarsely reconstructed depth maps which, combined with discontinuity at the shape boundary, lead to substantial artifacts in the decoded depth maps as illustrated in the difference images in Figure 6 a). The same effect is also visible in the difference image of the decoded color image in Figure 6 d). PAC demonstrates similar artifacts but the codec can handle the boundary better because zero difference between the average frame and the delta frame means a value of 128 and we employ padding with 128 in the delta frames. The errors for the MPEG-4 codecs appear to be uniformly distributed as can be seen in Figure 6 c) and f) which leads to better visual results in the rendered FVV images.

6 Conclusions and Future Work

We presented image-space free-viewpoint video as a representation and data format for re-rendering multi-view video data from arbitrary viewpoints. Since the representation is adopted as an extension of the MPEG-4 AFX standard and coding is possible with standard MPEG-4 video codecs the image-space free-viewpoint framework is possible with available MPEG-standardized technology. Using special codecs further features like progressiveness and random access to the video data can be employed. We believe that free-viewpoint video gives novel possibilities regarding replay, editing and interacting with multi-view video data. Future work includes investigating coding of normal and splat fields and optimal bit allocation for the different attributes of the representation. Furthermore, shape-adaptivity as employed by MPEG-4 proves to be much superior in terms of rate distortion. Investigation of shape-adaptive JPEG-2000 and H.264/AVC would be interesting. In addition, an algorithm determining the optimal window length for average coding should be developed.

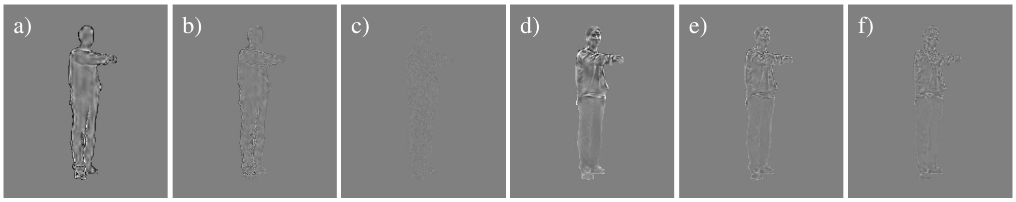

Figure 6: Difference images between uncompressed and compressed reference video images, from a camera contributing to the view in Figure 9. Depths at total bit rate of 512 kbps, magnified by a factor of 5: a) J2K, b) PAC, c) MP4. Colors at total bit rate of 128 kbps, magnified by a factor of 2: d) J2K, e) PAC, f) MP4-I.

References

[1] Y. Bayakovski, L. Levkovich-Maslyuk, A. Ignatenko, A. Konushin, D. Timasov, A. Zhirkov, M. Han, and I. K. Park. Depth image-based representations for static and animated 3d objects. In *ICIP '02*, volume 3, pages 25–28, 2002.

[2] 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.

[3] H. Briceno, P. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos. In *Proceedings of ACM Symposium on Computer Animation 2003*, pages 136–146, 2003.

[4] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH '01*, pages 425–432, 2001.

[5] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In *SIGGRAPH '03*, pages 569–577, 2003.

[6] M. Deering. Geometry compression. In *SIGGRAPH '95*, pages 13–20, 1995.

[7] M. Gross, S. Würmlin, M. Näf, E. Lamboray, C. Spagno, A. Kunz, A. V. Moere, K. Strehlke, S. Lang, T. Svoboda, E. Koller-Meier, L. V. Gool, and O. Staadt. blue-c: A spatially immersive display and 3d video portal for telepresence. In *SIGGRAPH '03*, pages 819–827, 2003.

[8] ISO/IEC JTC1/SC29/WG11 (MPEG). FDIS of ISO/IEC 14496 Part 16: Animation Framework eXtension (AFX). Doc. N5397, Awaji Island, Japan, December 2002.

[9] ISO/IEC JTC1/SC29/WG11 (MPEG). Applications and requirements for 3DAV. Doc. N5877, Trondheim, Norway, July 2003.

[10] ISO/IEC JTC1/SC29/WG11 (MPEG). FPDAM1 of ISO/IEC 14496 Part 16: Animation Framework eXtension (AFX). Doc. N6986, Hong Kong, China, January 2005.

[11] A. K. Katsaggelos, L. P. Kondi, F. W. Meier, J. Ostermann, and G. M. Schuster. MPEG-4 and rate-distortion-based shape-coding techniques. *Proceedings of the IEEE*, 86(6):1126–1154, June 1998.

[12] E. Lamboray, S. Würmlin, M. Waschbüsch, M. Gross, and H. Pfister. Unconstrained free-viewpoint video coding. In *ICIP '04*, 2004.

[13] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH '96*, pages 31–42, 1996.

[14] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *EGRW '01*, pages 115–125, 2001.

[15] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH '00*, pages 369–374, 2000.

[16] W. Matusik and H. Pfister. 3D TV: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH '04*, 2004.

[17] J. Mulligan and K. Daniilidis. View-independent scene acquisition for tele-presence. In *International Symposium on Augmented Reality '00*, pages 105–110, 2000.

[18] M. Pauly, M. Gross, and L. Kobbelt. Efficient simplification of point-sampled geometry. In *VIS '02*, pages 163–170, 2002.

[19] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. *IEEE Transactions on Signal Processing*, 41(12):3445–3462, 1993.

[20] D. Taubman and A. Secker. Highly scalable video compression with scalable motion coding. In *ICIP '03*, volume 3, pages 273–276, 2003.

[21] S. Würmlin, E. Lamboray, and M. Gross. 3d video fragments: Dynamic point samples for real-time free-viewpoint video. *Computers & Graphics '04*, 28(1):3–14, 2004.

[22] S. Würmlin, E. Lamboray, O. G. Staadt, and M. H. Gross. 3D video recorder. In *Proceedings of Pacific Graphics 2002*, pages 325–334, 2002.

[23] Z. Zhang, G. Liu, and Y. Yang. High performance full scalable video compression with embeddd multiresolution MC-3DSPIHT. In *ICIP '02*, volume 3, pages 721–724, 2002.

[24] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002.

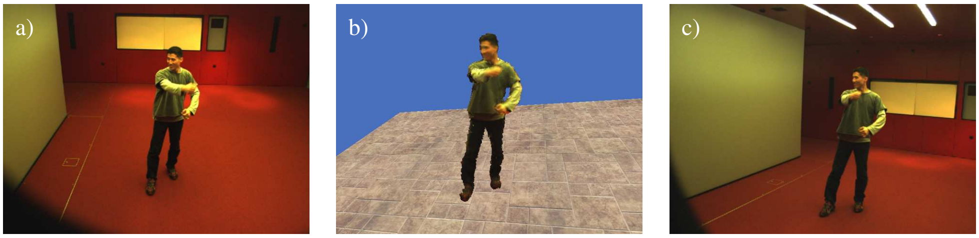

Figure 7: An image-space free-viewpoint video example. a) and c) are synchronized video images from different viewpoints, b) is an intermediate view rendered from uncompressed data.

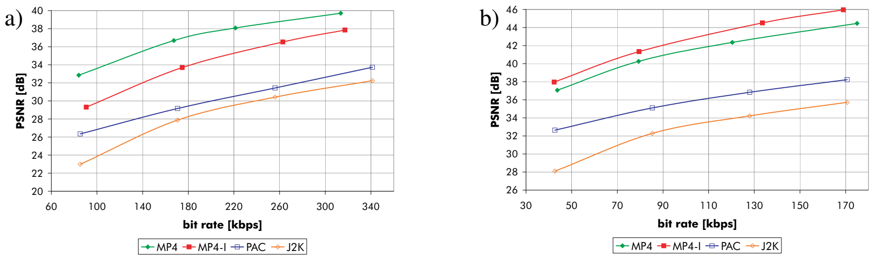

Figure 8: Results from our test sequences, PSNR values for devised bit rates: a) color values, b) depth values. Note the different scales of the PSNR values. Bit rates are given for coding of one attribute only.

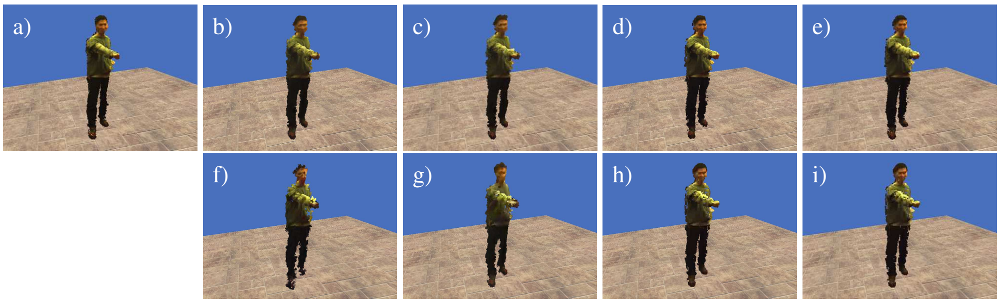

Figure 9: Rendered FVV images with different coding methods compared to image rendered from uncompressed data a). Total bit rate of 512 kbps per camera: b) J2K, c) PAC, d) MP4-I, e) MP4. Total bit rate of 128 kbps per camera: f) J2K, g) PAC, h) MP4-I, i) MP4.

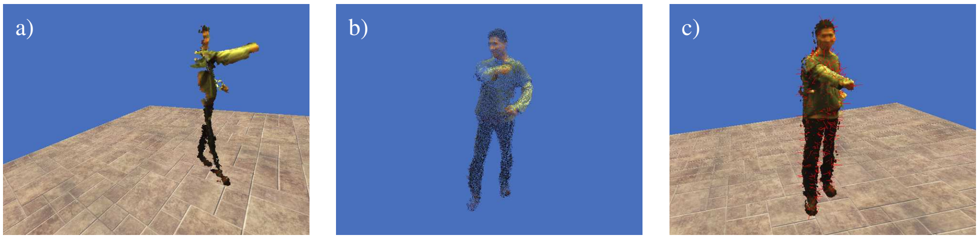

Figure 10: a) Locked camera selection and rotated viewpoint for image of Figure 7, visualizing the depth values and view-dependent rendering. b) Visualization of point sample representation by scaling down the splat sizes. c) Visualization of point sample normals.