Representation, Coding, and Rendering of 3D Video Objects with MPEG-4 and H.264/AVC

A. Smolić, K. Mueller, P. Merkle, T. Rein, M. Kautzner, P. Eisert, and T. Wiegand
Image Processing Department
Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institute
Einsteinufer 37, 10587 Berlin, Germany

***Abstract*—3D video objects provide the same functionalities as virtual computer graphics objects but depict the motion and appearance of real world moving objects. They can be viewed interactively from any direction and integrated in complete 3D scenes with other virtual and real world elements. So far, related work only considered extraction, representation, and rendering methods. Compression and transmission has not yet been studied in detail and combined with the other components into one complete system. In this paper, we present such a complete system for efficient 3D video object extraction, representation, coding, and interactive rendering. Data representation is based on 3D mesh models and view-dependent texture mapping using video textures. The geometry extraction is based on a shape-from-silhouette algorithm. The resulting voxel models are converted into 3D meshes that are coded using MPEG-4 SNHC tools. The corresponding video textures are preprocessed taking the object's shape into account and coded using an H.264/AVC codec. The presented results illustrate that based on the proposed methods a complete transmission system for 3D video objects can be built.**

***Keywords*—3D video objects; free viewpoint video; visual hull; shape-from-silhouette; 3D mesh coding; H.264/AVC.**

I. Introduction

Free viewpoint video is a novel representation format for visual media that allows the user to navigate freely within dynamic real world scenes by choosing arbitrary viewpoints and view directions. This functionality is well-known from computer graphics, computer games and virtual reality applications. However, in these cases, most of the scenes and objects are either purely computer generated or contain static 2D views of real world objects represented by still pictures or moving textures. In contrast free viewpoint video targets at real world scenes and objects as captured by real imagery.

Typically scenes and objects are captured using synchronized multi-camera systems. This can be dome-like settings as illustrated in Fig. 2, but also parallel settings and other types of camera arrangements. The video signals are processed and converted into a certain format that allows generation of virtual intermediate views and thus enables free navigation. Of course the range of navigation is restricted by the camera arrangement, and in general the overall quality of rendered views increases with the number of available cameras.

A 3D video object is a special case of free viewpoint video. Here the representation does not include the whole scene but is restricted to a certain object, typically related to physical objects such as humans. 3D video objects rely on a certain geometry representation with associated texture, which are both changing over time. They provide the same functionalities as classical computer graphics objects (free navigation, integration in complete scenes), but depict the motion and appearance of real world moving objects. This is illustrated in Fig. 1. Here 4 rendered views at 4 different time instances from 4 different virtual viewpoints are shown. These are snapshots from a virtual camera fly around the object as it moves.

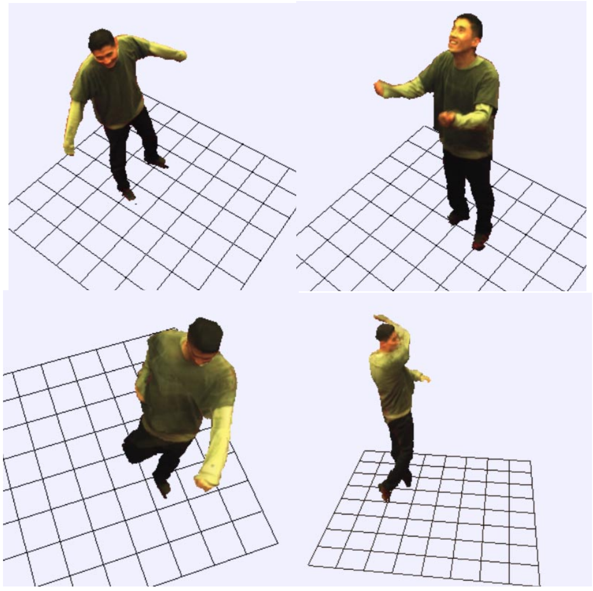

Fig. 1: Virtual camera fly, rendered views at 4 different times from 4 different virtual viewpoints.

In this paper, we present a complete system for 3D video objects including compression. It uses conventional 3D mesh models and view-dependent mapping of multiple textures as acquired from real cameras. The 3D geometry is reconstructed using a shape-from-silhouette algorithm that calculates a voxel approximation of the visual hull [1]. The voxel model is converted into a 3D mesh using a marching cubes algorithm and mesh reduction [2]. The geometry is coded using 3D mesh coding as standardized in MPEG-4 SNHC [3]. The multiple video textures are compressed using H.264/AVC [4]. For that we developed a special shape-oriented preprocessing of the video sequences that significantly improves texture coding. For view-dependent texture mapping with unstructured Lumigraph rendering [5] we developed a new tool that was accepted as extension to MPEG-4 AFX [6].

This paper is organized as follows. In the next section, we briefly describe the process of geometry extraction from multiple video streams and rendering using view-dependent texture mapping. Coding and preprocessing is described in section 3. Section 4 presents experimental results.

II. Geometry Reconstruction and Rendering

Acquisition of 3D video objects typically relies on a multi-camera setup as shown in Fig. 2. In general, the quality of the rendered views increases with the number of available cameras. However, equipment costs and often the complexity costs required for processing increase as well. We therefore consider a classical tradeoff between quality and costs by limiting the number of cameras and compensating this by geometry extraction.

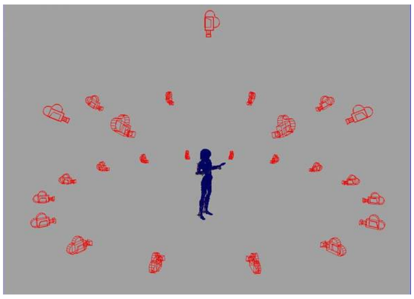

Fig. 2: 3D video object acquisition.

The first step of our algorithm consists of deriving intrinsic and extrinsic parameters for all cameras that relate the 2D images to a 3D world coordinate system since our geometry extraction and rendering algorithms require knowledge of these parameters. These parameters are computed from reference points using a standard calibration algorithm [7].

In the next step, the object to be extracted is segmented in all camera views. For that we use the combination of an adaptive background subtraction algorithm and Kalman filter tracking. The results of this step are silhouette videos that indicate the object's contour for all cameras. For details please refer to [8]. The 3D volume containing the object is reconstructed from the silhouette images using an octree-based shape-from-silhouette algorithm [9]. After visual hull approximation the object's surface is extracted from the voxel model by applying a marching cubes algorithm and represented with a 3D mesh [2].

For photo-realistic rendering, the original videos are mapped onto the reconstructed geometry. Natural materials may appear very different from different viewing directions depending on their reflectance properties and the lighting conditions. Static texturing (e.g. interpolating the available views) therefore often leads to poor rendering results. We have developed an algorithm for view-dependent texture mapping that more closely approximates natural appearance when navigating through the scene.

As illustrated in Fig. 3, the textures are projected onto the geometry using the calibration information. For each projected texture a normal vector $\boldsymbol{n}_i$ is defined pointing into the direction of the original camera. For generation of a virtual view into a certain direction $\boldsymbol{v}_{VIEW}$ a weight is calculated for each texture, which depends on the angle between $\boldsymbol{v}_{VIEW}$ and $\boldsymbol{n}_i$. The weighted interpolation ensures that at original camera positions the virtual view is exactly the original view. The closer the virtual viewpoint is to an original camera position the greater the influence of the corresponding texture on the virtual view.

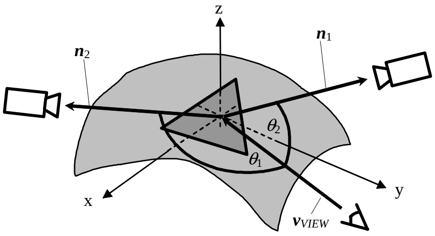

Fig. 3: Weighted virtual view interpolation.

θ_i are the angles between the virtual viewing direction and the camera directions $\boldsymbol{n}_i$. First individual weights w_i are calculated as described in [10]:

$$w_i = \begin{cases} \dfrac{\cos\theta_i}{1-\cos\theta_i}, & \cos\theta_i \neq 1 \\ Float\_Max, & \cos\theta_i = 1 \end{cases} . \qquad (1)$$

If one angle approaches zero, the weight approaches infinity. All other weights can be neglected in this case. In order to avoid overlighting the weights are normalized such that they always sum up to one:

$$a_i = \frac{w_i}{\sum\limits_{\forall i} w_i} . \qquad (2)$$

This weighted interpolation approximates realistic intermediate views enabling realistic virtual camera flights through the scene.

III. Representation, Rendering, and Coding with MPEG-4 and H.264/AVC

Previous work on 3D video objects was mainly concentrated on extraction and rendering. Coding and transmission issues have not been considered in detail so far. To investigate these issues, MPEG has established a working group called 3DAV [11]. MPEG-4 is a suitable framework for 3D video objects since it already provides state-of-the-art components for computer graphics and video representation and coding. However, view-dependent texture mapping as described in the previous section is not supported so far. We therefore did propose the technology as a new part of the Animation Framework eXtension (AFX) of MPEG-4. This part of the standard specifies advanced computer graphics tools [12]. Our proposal for view-dependent texture mapping was adopted for an upcoming amendment since the technology is not only suitable for 3D video objects but for other computer graphics applications as well.

For coding of the 3D meshes, MPEG-4 already provides efficient tools [3] that can readily be used for our 3D video object representation. For coding of the dynamic textures that are in fact video sequences we investigated several video codecs. The most advanced open standard codec is the recent H.264/AVC [4], which is a joint standard of ITU-T VCEG and MPEG. It outperforms all other available standard codecs but only supports standard rectangular video.

However, for rendering of 3D video objects at the decoder, we don't need to transmit the complete rectangular video as acquired by the cameras. Only the area covered by the object of interest needs to be transmitted. Therefore, the video sequences are preprocessed prior to encoding as illustrated in Fig. 4. We cut out a bounding box that completely contains the object and that has dimensions in pixels that are multiples of the macroblock size (16x16). Within the bounding box all empty macroblocks that do not contain the object are set to a constant value of 128. This preprocessing ensures that only the minimum information that the decoder needs to render the 3D video object is actually encoded. Note that we do not need to transmit the shape information since it is already given at the decoder by the 3D mesh model. Finally, the video is encoded using standard H.264/AVC syntax.

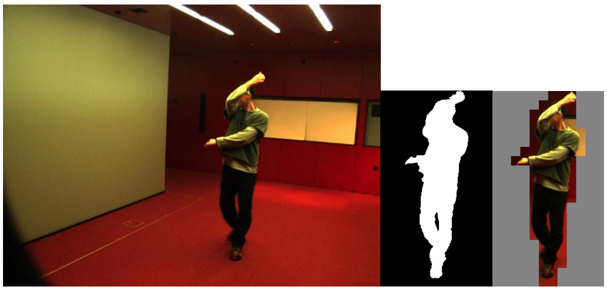

Fig. 4: Preprocessing for H.264/AVC coding.

Additionally, the camera calibration parameters have to be transmitted once to the decoder as side information, which means 10 float values per camera and session.

IV. Experimental Results

The experiments were performed with several data sets made available to the MPEG 3DAV group. As illustrated Fig. 2, humans in motion were captured in a dome-like multi-camera setting. The geometry was reconstructed as explained in section 2. The resulting wire frames were encoded using MPEG-4 mesh coding, which allows varying the number of bits used per vertex. The resulting bitrates using the Doo Young data set (16 sequences, 640x480) are shown for some settings in TABLE I (2000 vertices/mesh). Examples of decoded meshes are shown in Fig. 5. We have found that increasing the number of bits per vertex above 8 bits does not significantly increase visual quality, when the texture is mapped onto the mesh.

Current MPEG-4 mesh coding only allows encoding the meshes for each frame separately (corresponding to I-frame coding in video). However, the meshes describe the same object moving over time, and therefore the compressed data still contain a huge amount of temporal statistical dependency. Our future research will therefore be to develop predictive coding for moving and deforming 3D meshes. We also plan to constrain the geometry reconstruction in order to produce more time-consistent meshes, since this process also still works independently for each time instant.

TABLE I. Bitrates for Geometry in kbit/s

bit/vertex	6	7	8	10
bitrate [kbit/s]	556,42	685,33	831,08	1088,25

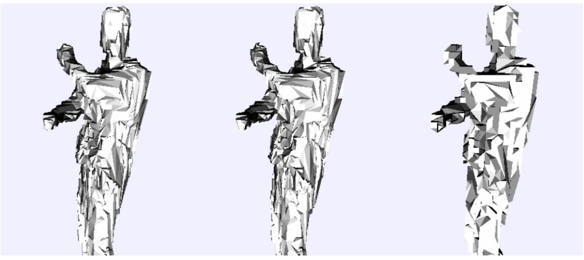

Fig. 5: Original mesh (left) and decoded meshes at 8 (middle) and 6 (right) bit per vertex.

The state-of-the-art open standard codec with shape support is currently MPEG-4 Core Profile [3]. We therefore encoded the selected test set with MPEG-4 in arbitrary shape mode at different bitrates as reference (using Microsoft reference software). Then we encoded the complete (rectangular) sequences with H.264/AVC (JVT reference software with similar settings to MPEG-4 test). The PSNR was evaluated only within the object shape in both cases. In all our experiments H.264/AVC outperformed MPEG-4 significantly for several dB, even if the bits used for shape by the MPEG-4 codec are subtracted. A typical example result for one sequence is shown in Fig. 6. For such sequences with relatively static background as shown in Fig. 4 (changes only due to noise, shadows and lighting effects) H.264/AVC is much more efficient even if the complete video is encoded.

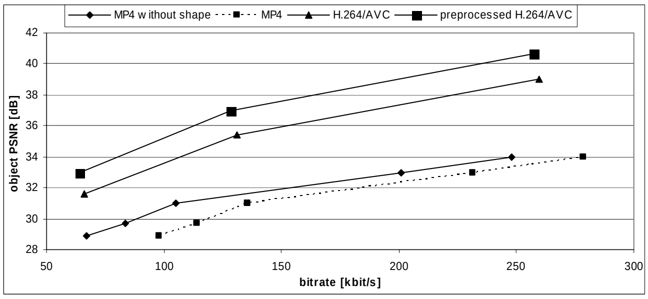

Fig. 6: PSNR within object for several codecs over bitrate.

However, encoding the background with H.264/AVC is a waste of bits in our application scenario. We therefore applied preprocessing as described in Section 3 to all the test sequences and encoded the outcome with H.264/AVC. A result is also included in Fig. 6. We get a significant gain for the mean object PSNR of up to 2.6 dB compared to H.264/AVC without preprocessing. The gain depends on the content and most of all on the size of the resulting bounding box. TABLE II shows the results for the selected subset of 8 cameras. We get the largest gain for small bounding boxes.

For large bounding boxes we get small gains and even losses. The mean gain increases with the bitrate from 0.99 and 1.13 to 1.31 dB for 64, 128 and 256 kbit/s/view respectively. These results show that even such simple preprocessing can significantly improve the results for video object coding. Our future research will therefore be a further improvement through better integration of shape-adaptive coding algorithms into H.264/AVC.

TABLE II. OBJECT PSNR GAIN USING PREPROCESSED VIDEO

Sequence number	**Size of bounding box**	**PSNR gain @ 64 kbit/s/view**	**PSNR gain @ 128 kbit/s/view**	**PSNR gain @ 256 kbit/s/view**
11	352x336	1,04	1,25	1,46
13	352x320	0,45	0,67	1,05
14	416x432	-0,18	0,15	0,22
16	416x448	-0,25	-0,01	0,28
2	192x336	1,36	1,51	1,65
4	192x320	2,61	2,54	2,39
7	224x352	1,7	1,57	1,77
9	176x320	1,2	1,37	1,68

Fig. 7 compares details of virtual rendered views in the middle between original views. The left example was rendered with uncompressed textures. The other images show examples with textures coded at 128 and 64 kbit/s/view illustrating the degradation of image quality with reduced bitrate.

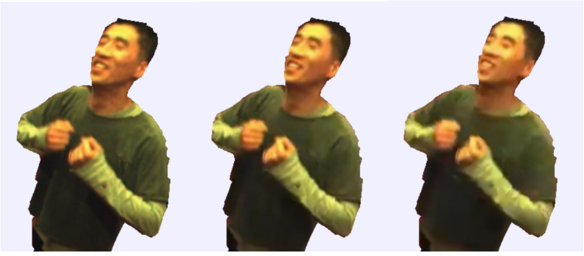

Fig. 7: Details of rendered views uncoded, 128 and 64 kbit/s/view.

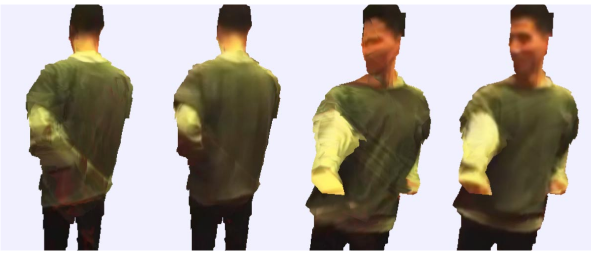

Fig. 8: Rendered views using 4 and 8 textures at same total texture bitrate of 512 kbit/s.

In general, the quality of view-dependent texture mapping increases with the number of textures used. On the other hand, also the data rate and the processing and rendering complexity increase. We therefore tested our algorithms with subsets including 4 and 8 video textures. Fig. 8 compares rendered views generated using 4 and 8 textures at the same total texture bitrate of 512 kbit/s. This means that the 4-view set was coded at 128 kbit/s/view and the 8-view set was coded at 64 kbit/s/view. Rendering artifacts decrease with the number of views, however, coding artifacts increase with the number of views. This illustrates that for coding of 3D video objects, a tradeoff has to be found regarding the number of views for optimum overall visual quality at a particular given bitrate.

V. SUMMARY AND FUTURE WORK

The presented system for 3D video objects covers the entire processing and transmission chain from cameras to interactive display including data compression. Our results indicate suitability of the tested approaches for the envisaged application. Predictive mesh coding to exploit temporal redundancy and constrained geometry reconstruction for time consistent mesh generation are subject to further research. Also improved shape-adaptive H.264/AVC coding will be studied in the future.

ACKNOWLEDGMENT

We would like to thank the Computer Graphics Lab of ETH Zurich for providing the Doo Young multi-camera data set.

REFERENCES

[1] P. Eisert, E. Steinbach, and B. Girod, “Multi-hypothesis, Volumetric Reconstruction of 3-D Objects from Multiple Calibrated Camera Views,” *Proc. ICASSP*, pp. 3509-3512, Phoenix, Mar. 1999.

[2] W. E. Lorensen, and H. E. Cline, “Marching Cubes: A high resolution 3D surface reconstruction algorithm,” *Proc. SIGGRAPH*, vol. 21, no. 4, pp 163-169, 1987.

[3] ISO/IEC JTC1/SC29/WG11, “Information Technology - Coding of Audio-Visual Objects, Part 2: Visual; 2001 Edition,” Doc. N4350, Sydney, Australia, July 2001.

[4] ITU-T Recommendation H.264 & ISO/IEC 14496-10 AVC, “Advanced Video Coding for Generic Audio-Visual Services,” 2003.

[5] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, “Unstructured Lumigraph Rendering,” *Proc. SIGGRAPH*, pp. 425-432, 2001.

[6] K. Mueller, and A. Smolic, “Study on View-Dependent Multitexturing for MPEG-4 AFX,“ ISO/IEC JTC1/SC29/WG11, MPEG03/M10152, Gold Coast, Australia, October 2003.

[7] R.Y. Tsai, “A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV camera and lenses,” *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 4, August 1987.

[8] K. Mueller, A. Smolic, M. Droese, P. Voigt, and T. Wiegand, “Multi-Texture Modelling of 3D Traffic Scenes,” *Proc. ICME*, Baltimore, MD, USA, July 6.-9. 2003.

[9] A. Smolic, K. Mueller, P. Merkle, T. Rein, M. Kautzner, P. Eisert, and T. Wiegand, “Free Viewpoint Video Extraction, Representation, Coding, and Rendering,” *Proc. ICIP*, Singapore, October 24.-27. 2004.

[10] D. Vlasic, H. Pfister, s. Molinov, R. Grzeszczuk and W. Matusik, “Opacity Light Fields: Interactive Rendering of Surface Light Fields with View-dependent Opacity”, *Proc. 2003 Symposium on Interactive 3D graphics*, pp. 65-74, 2003.

[11] A. Smolic, and D. McCutchen, “3DAV Exploration of Video-Based Rendering Technology in MPEG,” *IEEE Trans. on CSVT*, vol. 14, no. 9, pp. 348-356, March 2004.

[12] ISO/IEC JTC1/SC29/WG11, “Text of ISO/IEC 14496-16:2003/FDAM4,” Doc. N5397, Awaji, Japan, December 2002.