

# CODING OF 3D MESHES AND VIDEO TEXTURES FOR 3D VIDEO OBJECTS

K. Mueller, A. Smolic, P. Merkle, M. Kautzner, and T. Wiegand  
Image Processing Department  
Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut  
Einsteinufer 37, 10587 Berlin, Germany  
{smolic/kmueller/merkle/kautzner/wiegand}@hhi.de

**Abstract**— We present a complete system for efficient 3D video object extraction, representation, coding, and interactive rendering. 3D video objects provide the same functionalities as virtual computer graphic objects but also depict motion and appearance of real world moving objects. They can be viewed interactively from any direction and can be integrated in complete 3D scenes together with other virtual or real world elements. In this work, 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. For efficient compression of such dynamic 3D meshes, we present a novel algorithm that significantly outperforms available standard approaches. The corresponding video textures are preprocessed taking the object's shape into account and coded using H.264/AVC. The presented results show that a complete and efficient transmission system for 3D video objects can be built.

## I. INTRODUCTION

The term “3D video object (3DVO)” describes a new representation format for visual media that allows free navigation around real world dynamic objects by choosing arbitrary viewpoints and viewing directions. 3DVOs are an extension of classical computer graphic objects towards representing motion and appearance of real world moving objects. This is illustrated in Fig. 1. It shows 3 rendered views at 3 different time instances from 3 different virtual viewpoints. These are snapshots from a virtual camera fly around the object as it moves. The complete video as well as other examples can be downloaded from [1].

Different representation and rendering formats have been proposed for 3DVOs. A straightforward solution is to use 3D meshes for geometry and to combine this with view-dependent texture mapping using the original camera views (polyhedral visual hulls) [2][3]. 3D meshes are widely used in computer graphics and are very efficiently supported by hardware and standard software APIs. Also international standards, such as VRML and MPEG-4 [4], support 3D meshes.

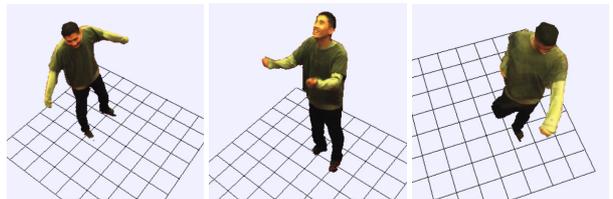


Fig. 1: Virtual camera fly, rendered views at 3 different times from 3 different virtual viewpoints.

Other possible representation formats would be image-based visual hulls [5], point-based [6], or volumetric representations [7]. However, these formats are not sufficiently supported by common graphics hardware, software, and standards. A sub-group of MPEG called 3DAV investigates standardization in this area [8]. An overview on interactive 3D video representation and coding can be found in [9].

For our system we use dynamic 3D meshes with associated video textures due to compatibility and interoperability reasons. The 3D geometry of dynamic real world objects is generated using a visual hull reconstruction algorithm from multi-camera video signals, as described in section II. This includes a new algorithm for generation of time-consistent meshes which are beneficial for geometry compression. For photo-realistic rendering we apply view-dependent texture mapping using the original camera views as described in section III. This algorithm was adopted into the MPEG-4 computer graphics part AFX (Animation Framework eXtension) [10].

Data representation and compression is described in section IV. Our approach is embedded in a MPEG-4 framework. For compression of dynamic 3D meshes we have developed a new algorithm that significantly outperforms available MPEG-4 tools. Video textures are encoded using H.264/AVC, which is the most efficient video codec available [11]. To further increase compression performance we apply shape-oriented preprocessing that is specifically adapted to coding of 3DVOs.

In section V we report experimental results on geometry and video texture coding that verify the effectiveness of the proposed techniques.

## II. GEOMETRY RECONSTRUCTION

The acquisition of 3DVOs 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.

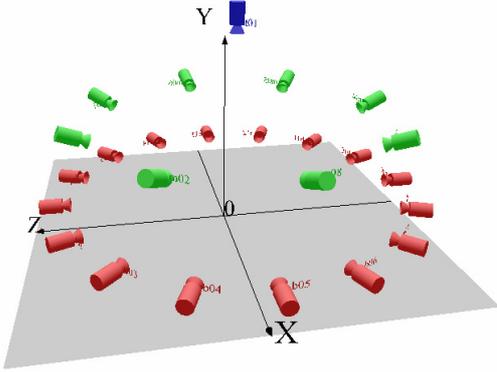


Fig. 2: Multi-camera setup for 3DVO acquisition.

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 [12].

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 [13]. The 3D volume containing the object is reconstructed from the silhouette images using an octree-based shape-from-silhouette algorithm [14]. After visual hull approximation the object’s surface is extracted from the voxel model by applying a marching cubes algorithm and represented by a 3D mesh [15]. This process is repeated for every time instant resulting in a sequence of 3D meshes, describing motion and deformation of the object’s geometry over time. Therefore this sequence of 3D meshes contains large amounts of statistical dependencies in the temporal domain, which can be exploited for compression as described in section IV.

In principle it is possible to use time-consistent models to represent a 3D geometry over time. Such time-consistent models use the same mesh-connectivity over time, only the 3D position of the vertices changes. However, we have found experimentally that due to deformations and imperfections of the reconstruction process, the

time-consistency constraint can only be exploited over a limited time of e.g. 0.5s. Note that we are not using any a priori assumptions about the object’s geometry, which could ease the use of time-consistent meshes (see e.g. [16]). We therefore use a time-consistent mesh for a group of meshes (GOM) of 11 time instances (approx. 0.5s). The mesh of the middle frame (i.e. the 6<sup>th</sup>) is reconstructed independently as described before. The meshes 1-5 and 7-11 of each GOM are reconstructed using the available meshes as reference. The connectivity is the same for all meshes in a GOM.

For all dependently reconstructed meshes we first extract a highly overrepresented 3D mesh (e.g. 25000 vertices) by constraining the marching cubes algorithm appropriately. Then we match the vertices of the temporally nearest available mesh (e.g. 1000 vertices) to the overrepresented point cloud of the actual geometry. For each vertex of the available mesh we compute one correspondence within the point cloud using neighborhood criteria, including vertex position and normal features. Thus we end up with a time-consistent mesh with the same connectivity and translated vertices.

This process introduces a fixed delay of 6 frames. Still this algorithm is a simple and straightforward solution. Future work will include a more sophisticated extraction of time-consistent meshes using 3D motion estimation.

## III. RENDERING WITH VIEW-DEPENDENT TEXTURE MAPPING

For photo-realistic rendering, the original videos are mapped onto the reconstructed geometry. Natural materials may appear very differently from diverse 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.

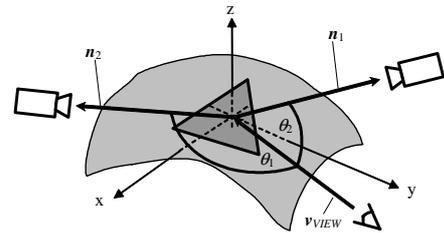


Fig. 3: Weighted virtual view interpolation.

Fig. 3 illustrates the idea. The textures are projected onto the geometry using the calibration information. For each projected texture a normal vector  $\mathbf{n}_i$  is defined that points into the direction of the original camera. For generation of a virtual view into a certain direction  $\mathbf{v}_{VIEW}$  a weight  $w_i$  is calculated for each texture, which depends on the angle  $\theta_i$  between  $\mathbf{v}_{VIEW}$  and  $\mathbf{n}_i$  [17]:



used to efficiently adapt to the signal statistics. First the probability density function (pdf) of the data was analyzed, finding a Laplacian distribution to be superimposed by further small peaks at varying positions due to the clustering algorithm. Therefore, CABAC is well suited, since it combines algorithms that fit these pdf-features: Firstly, the binarization uses unary/ $k^{\text{th}}$ -order Exp-Golomb codes to not only assign small code words to most frequent symbols but also fit to certain outliers (i.e. limit the code word length) that frequently occur. Furthermore, the unary as well as the Exp-Golomb part of the code are calculated from the data to minimize overall code length. The second important feature of CABAC is its usage of multiple context models to better fit the input signal statistics. Thus the algorithm can adapt to changing statistics, since 3D meshes exhibit a variety of global and local motion that significantly modify the corner vector distribution.

#### IV.2. Video texture coding

For coding of dynamic textures, which are in fact video sequences, we have investigated several video codecs: MPEG-4 Core Profile and H.264/AVC Main Profile. The latter does not support variable shape coding. However, for rendering of 3DVOs at the decoder, we don't need to transmit the complete rectangular video. Only the area covered by the object of interest needs to be transmitted. Therefore the video is preprocessed prior to encoding as illustrated in Fig. 5. For that, we extracted the bounding box that completely contains the object. The width and height of the bounding box is an integer multiple of 16 to fit entire macroblocks. Within the bounding box all empty macroblocks are set to a constant value of 128. 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. 5: Shape-oriented preprocessing for H.264/AVC.

### V. EXPERIMENTS

The experiments were performed with several real and synthetic test data sets.

#### V.1. Coding of dynamic 3D meshes

We first present results for the synthetic ‘‘Chicken Crossing’’ sequence. This is a sequence of 400 time-consistent meshes with 3030 vertices produced by animation of a

chicken model (Fig. 7). As reference these meshes were encoded using standard 3DMC, which allows varying the number of bits used per vertex. Then the meshes were encoded using D3DMC. For that we varied the size of the GOM and the bitrate (by variation of the error threshold for clustering). For quality evaluation we use an average mean squared error (MSE) measure between original and decoded vertices averaged over all vertices of a sequence.

Fig. 6 compares the MSE over bitrate for 3DMC and D3DMC. For D3DMC we show a GOM of 10 and a GOM of 400, which means that only the first mesh is INTRA coded. The results prove a significant increase of compression performance. For GOM400 we get the same quality at 10-20% of the bitrate. Still for GOM10 we get the same quality at 15-25% of the bitrate compared to standard 3DMC. As expected the gain increases with the GOM size.

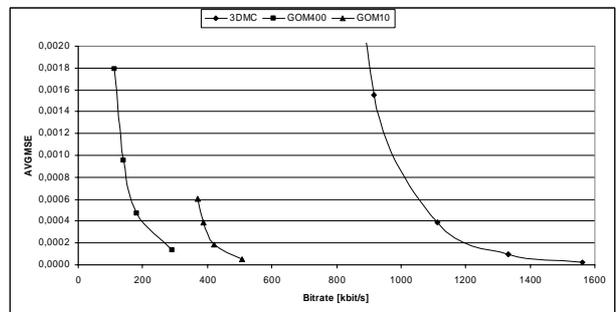


Fig. 6: MSE over bitrate for 3DMC and D3DMC, Chicken Crossing sequence.

Some visual examples are shown in Fig. 7. Left is the original mesh at a certain time from a certain viewpoint. The middle image shows the corresponding decoded mesh using 3DMC coded at 915 kbit/s. The right image shows the same mesh decoded using D3DMC coded at 507 kbit/s.



Fig. 7: Original mesh (left) and decoded meshes using 3DMC at 915 kbit/s (middle) and D3DMC at 507 kbit/s (right).

The same comparison was done for the meshes reconstructed from the real world data set Doo Young (16 sequences, 640x480, 25Hz) as explained in section II. As explained before we use a fixed GOM size of 11 for real world sequences. The comparison of quality over bitrate is shown in Fig. 8. Again D3DMC clearly outperforms 3DMC. We get the same quality at 30-75% of the bitrate. The gain decreases with the bitrate.

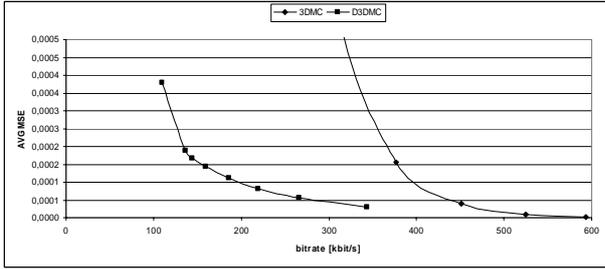


Fig. 8: MSE over bitrate for 3DMC and D3DMC, Doo Young sequence.

Visual results for the real data set are shown in Fig. 9. The bitrate for D3DMC is slightly higher than for 3DMC, however, the visual reconstruction quality is significantly improved.

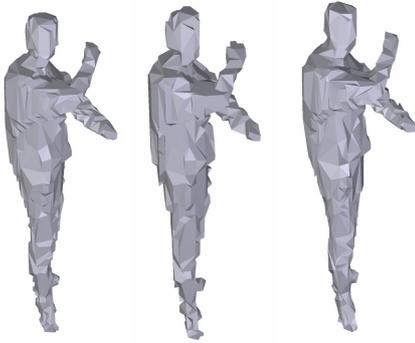


Fig. 9: Original mesh (left) and decoded meshes using 3DMC at 307 kbit/s (middle) and D3DMC at 343 kbit/s (right).

### V.2. Video texture coding

The state-of-the-art standard codec with shape support is MPEG-4 Core profile [4]. We have therefore encoded the test set with MPEG-4 Core profile in arbitrary shape mode at different bit rates (using Microsoft reference software). Then we encoded the complete sequences with H.264/AVC Main profile (JVT reference software with similar settings). The PSNR was evaluated only within the object shape in both cases. In all our experiments H.264/AVC Main profile outperformed MPEG-4 Core profile significantly for several dB, even if the bits used for shape by MPEG-4 Core profile were subtracted.

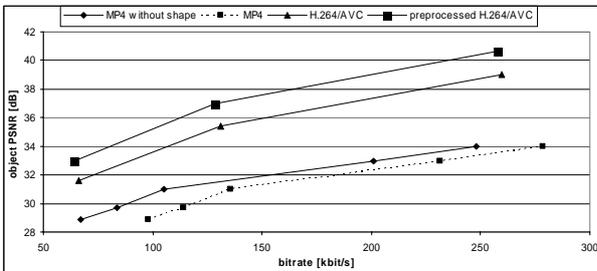


Fig. 10: PSNR within object for several codecs over bitrate.

A typical example result for one sequence is shown in Fig. 10. For such sequences with relatively static background as shown in Fig. 5 (changes only due to noise, shadows and lighting effects) H.264/AVC is much more efficient even if the complete video is encoded.

However, encoding the background with H.264/AVC is a waste of bits in our application scenario. We therefore applied preprocessing as described above to all the test sequences and encoded the results with H.264/AVC Main profile. Then we evaluated the PSNR gain within the object compared to coding the complete video with H.264/AVC Main profile as described before. We again get a significant gain for the mean object PSNR of up to 2.6 dB. The gain depends on the content and most of all on the size of the resulting bounding box. Table 1 shows the results. 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 include a better integration of shape-adaptive coding algorithms into H.264/AVC.

Table 1: Object PSNR gain using preprocessed video.

Seq. #	Bounding box	PSNR gain @ kbit/s/view		
		64	128	256
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

### V.3. Combination of geometry and texture coding

Decoded geometry and texture are finally combined for rendering with view-dependent texture mapping. Compression of 3DVOs inherently includes splitting the available overall bitrate between geometry and texture that has to be optimized for a particular application at hand. Also the number of textures transmitted needs to be determined against the bit rate per view [14]. We performed exhaustive experiments around this issue that cannot be reported within the scope of this paper. An example is illustrated in Fig. 11. From the data set we selected 8 cameras for texture mapping (table 1). The left image shows a rendered virtual view of the 3DVO using uncoded geometry and texture. The middle image shows the same view with geometry coded at 343 kbit/s and video texture coded at 1 Mbit/s (i.e. 128 kbit/s/view). The right image gives the same view using 110 kbit/s for geometry and 0.5 Mbit/s for video texture. This example illustrates the degradation of quality with decreasing bitrate.



Fig. 11: Rendered views of 3DVO, uncoded (left), 1.34 Mbit/s (middle), 0.61 Mbit/s (right).

## VI. CONCLUSIONS AND FUTURE WORK

The presented system for 3DVOs covers the entire processing chain from cameras to interactive display including data compression. We have presented a novel algorithm for compression of dynamic 3D meshes that significantly outperforms available technology. For video texture coding we use shape-oriented preprocessing and H.264/AVC. Our results indicate the suitability of the tested approaches for the envisaged application. Future work will include an advanced algorithm for time-consistent mesh extraction, rate-distortion-optimized mesh coding, better integration of shape-adaptivity into H.264/AVC, and integration into complete applications and systems.

### ACKNOWLEDGMENT

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

We would also like to thank Microsoft for providing the Chicken Crossing sequence (© Copyright 1996, Microsoft Corporation. The chicken character was created by Andrew Glassner, Tom McClure, Scott Benza, and Mark Van Langeveld. This short sequence of connectivity and vertex position data is distributed solely for the purpose of comparison of geometry compression techniques.).

### REFERENCES

- [1] <http://bs.hhi.de/~smolic/ICIP04.html>
- [2] W. Matusik, C. Buehler, and L. McMillan, "Polyhedral Visual Hulls for Real-Time Rendering", Proc. Eurographics Workshop on Rendering 2001.
- [3] T. Matsuyama, X. Wu, T. Takai, and T. Wada, "Real-Time Dynamic 3-D Object Shape Reconstruction and High-Fidelity Texture Mapping for 3-D Video", IEEE Trans. on CSVT, Vol. 14, No. 3, pp. 357-369, Mar. 2004.
- [4] ISO/IEC JTC1/SC29/WG11, "Information Technology - Coding of Audio-Visual Objects, Part 2: Visual; 2001 Edition", Doc. N4350, Sydney, Australia, July 2001.
- [5] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-Based Visual Hulls", Proc. SIGGRAPH 2000, pp. 369-374, 2000.
- [6] S. Würmlin, E. Lamboray, and M. Gross, "3D video fragments: dynamic point samples for real-time free-viewpoint video", Computers and Graphics 28 (1), pp. 3-14, Elsevier Ltd, 2004.
- [7] B. Goldlücke, and M. Magnor, "Real-time Microfacet Billboarding for Free-viewpoint Video Rendering", Proc. ICIP2003, Barcelona, Spain, Sep. 2003.
- [8] 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, Mar. 2004.
- [9] A. Smolic, and P. Kauff, "Interactive 3D Video Representation and Coding Technologies", to appear in Proc. of the IEEE, Special Issue on Advances in Video Coding and Delivery, scheduled Dec. 2004.
- [10] ISO/IEC JTC1/SC29/WG11, "ISO/IEC 14496-16/PDAM1", Doc. N6544, Redmont, WA, USA, July 2004.
- [11] ITU-T Recommendation H.264 & ISO/IEC 14496-10 AVC, "Advanced Video Coding for Generic Audio-Visual Services", 2003.
- [12] R.Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV camera and lenses", IEEE Jour. of Robotics and Automation, vol. RA-3, no. 4, August 1987.
- [13] 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.
- [14] A. Smolic et al., "Free Viewpoint Video Extraction, Representation, Coding, and Rendering", Proc. ICIP, Singapore, Oct. 24.-27. 2004.
- [15] 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.
- [16] J. Carranza, C. Theobalt, M. Magnor, and H.P. Seidel, "Free-Viewpoint Video of Human Actors", Proc. ACM SIGGRAPH 2003, pp. 569-577, San Diego, CA, USA, July 2003.
- [17] 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.
- [18] J. Zhang, and C.B. Owen, "Octree-based Animated Geometry Compression", DCC'04, Data Compression Conference, Snowbird, Utah, USA, pp. 508-517, Mar. 23.-25.2004.
- [19] D. Marpe, H. Schwarz, and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", IEEE Trans. on CSVT, vol. 13, no. 7, pp. 620-636, July 2003.