3D VIDEO OBJECTS FOR INTERACTIVE APPLICATIONS

A. Smolic, K. Müller, P. Merkle, M. Kautzner, and T. Wiegand

Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Image Processing Department
Einsteinufer 37, 10587 Berlin, Germany
phone: +49 30 31002 232, fax: +49 30 3927200, email: smolic@hhi.de

ABSTRACT

In this paper we present a 3D scene representation with standardized components to be used in interactive applications. For this representation we also show efficient coding of 3D geometry and textures as well as a 3D reconstruction system for creating 3D video objects (3DVOs). Similar to computer graphics objects, 3DVOs provide functionalities, like free scene navigation and animation. In contrast, they describe real world appearance and natural motion. The presented object description combines a 3D mesh model with a number of original video textures. These videos are weighted in the final object rendering according to the particular point of view. For coding the object meshes over time, we present a novel algorithm that exploits spatial and temporal dependencies in the mesh sequence and outperforms comparable coding methods. For the multi-texture coding, we preprocessed the video textures w.r.t. their shapes and applied H.264/AVC, the MPEG-4 state-of-the-art video coder.

1. INTRODUCTION

3D video objects (3DVOs) offer a new representation format for visual media that allows free navigation around real world dynamic scenes 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. 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. For geometry reconstruction, the original camera views are used to obtain polyhedral visual hulls [7], [8]. 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 [2], support 3D meshes.

Other possible representation formats would be image-based visual hulls [9], point-based [18], or volumetric representations [1]. 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 [14]. An overview on interactive 3D video representation and coding can be found in [13].

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 2. For photo-realistic rendering we apply view-dependent texture mapping using the original camera views. The texture weighting is carried out in the final scene rendering, considering the particular point-of-view that is interactively selected by the user. This algorithm was adopted into the MPEG-4 computer graphics part AFX (Animation Framework eXtension) to create a view-dependent multi-texture object description [3].

Data representation and compression is described in section 3. Our approach is embedded in the 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 [4]. To further increase compression performance we apply shape-oriented preprocessing that is specifically adapted to coding of 3DVOs. Here, we also report experimental results on geometry and video texture coding that verify the effectiveness of the proposed techniques.

2. OBJECT RECONSTRUCTION

The following section gives a short overview of the geometry reconstruction process, which starts with the acquisition of 3DVOs. These video objects typically rely on a multi-camera setup as shown in Fig. 1.

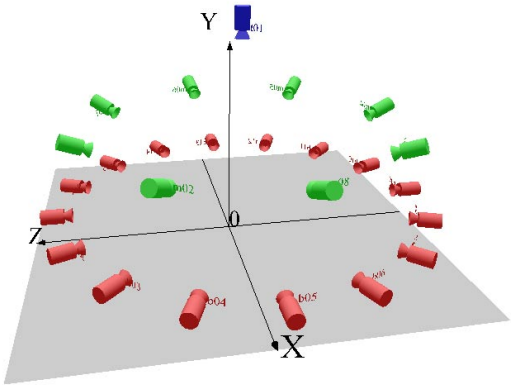

Fig. 1: Multi-camera setup for 3DVO acquisition.

In general, the quality of rendered views increases with the number of available cameras. However, equipment costs

and often 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. 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 [16].

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 [10]. The 3D volume containing the object is reconstructed from the silhouette images using an octree-based shape-from-silhouette algorithm [12]. After visual hull approximation the object's surface is extracted from the voxel model by applying a marching cubes algorithm [5] and represented by a 3D mesh. Then the object's surface is smoothed applying a first order neighborhood smoothing [15]. Finally the number of surface triangles is drastically reduced using a standard edge-collapsing algorithm, which mainly analyzes normal vectors of adjacent faces. The transformation steps from the octree model to the reduced wireframe surface model are shown in Fig. 2.

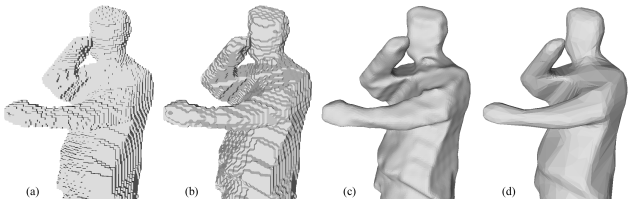

Fig. 2: Surface transformation: voxel model (a), marching cubes (b), first order neighborhood smoothing (c) and reduced mesh (d).

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. Therefore, we apply view-dependent texture mapping that more closely approximates natural appearance when navigating through the scene. For every video texture, we calculate view-dependent weights from the camera vector of each associated texture and the current viewing direction. Here, the cosine term is calculated from these two vectors and than transformed into a normalized texture weight, as shown in [17]. This way, smooth texture weighting is achieved at intermediate viewpoints between original camera positions and exclusive weighting of single textures at a viewpoint equal to their associated original camera position.

3. REPRESENTATION AND CODING

MPEG-4 already provides a rich framework for interactive and 3D audio-visual media, including representation and compression tools for natural audio, video, and computer graphics. These elements can be combined to build a large variety of multimedia applications and systems.

Since view-dependent texture mapping is now supported by AFX, our representation format for 3DVOs is compatible to MPEG-4. Furthermore, we have developed a new coding algorithm for dynamic 3D meshes that outperforms available MPEG-4 tools and we apply the most efficient available video coding standard H.264/AVC together with shape-oriented preprocessing. An overview of 3D video object representation and underlying coding is shown in Fig. 3.

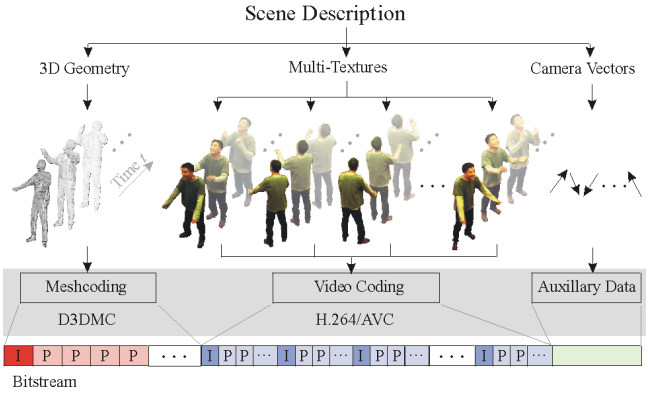

Fig. 3: Description and coding of 3D Video Objects with geometry and multi-texture components.

3.1 Coding of dynamic 3D meshes

In [11] we present the structure of D3DMC, a differential mesh coder, which compresses time-consistent meshes with common connectivity. A number of time-consistent meshes of video objects are referred to as group-of-meshes or GOM with an I mesh containing the common connectivity and associated P meshes with 3D coordinates only. D3DMC uses MPEG-4 static mesh coding (3DMC) for the I mesh and a DPCM structure with motion vector clustering, similar to [19], and context-adaptive binary arithmetic coding (CABAC) [6] for the P meshes. This approach guarantees better compression results, than coding each mesh separately with static 3DMC, as shown in Fig. 4. The example shows the reconstruction error for the "Doo Young" sequence.

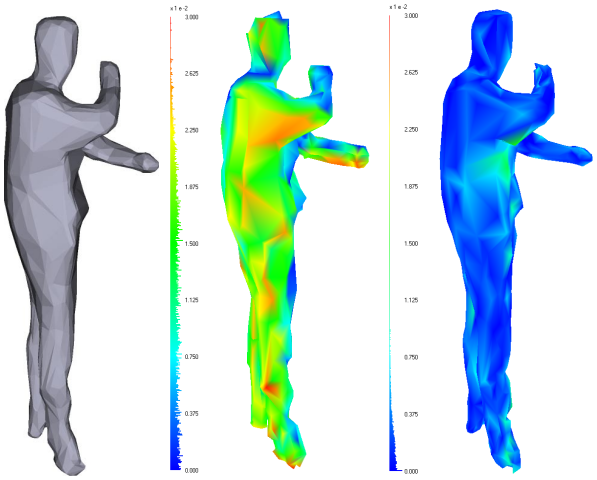

Fig. 4: Original mesh (left) and reconstruction error using 3DMC at 274 kBit/s (middle) and D3DMC at 253 kBit/s (right).

Here, the reconstruction error represents the point-to-face Hausdorff or Euclidian distance between two meshes. In Fig. 4 left, the original mesh is shown while the other images show the reconstruction error for 3DMC and D3DMC GOM11 (GOM with 11 time-consistent meshes) respectively. The colors (see histogram to the left of the error images, where blue represents a small error, red a large error) indicate the amount of reconstruction error. Both color codes have a maximum value of 0.03. Here, the reconstruction error for D3DMC has dropped to values smaller then 0.0075, as indicated by the blue and green colors in Fig. 4 right.

For state-of-the-art coding of dynamic meshes or mesh animation, usually MPEG-4 AFX Interpolator Compression (AFX-IC) is used for all P meshes within a GOM. Therefore, the dynamic compression of D3DMC is compared to AFX-IC. For the dynamic mesh comparison with AFX-IC, the average distortion error (D_A) is used, which measures the temporal frame distance in addition to the spatial Hausdorff distance. As an example we present results for the synthetic "Chicken Crossing" sequence, which provides a sequence of 400 time-consistent meshes (GOM400) with 3030 vertices. The results for D3DMC are compared with AFX-IC in Fig. 5.

For efficient data representation, we use the coordinate interpolator syntax. Here only a subset of all P meshes is used as keyframes, while all other P meshes are reconstructed during rendering by linear interpolation between the associated keyframes. The results in Fig. 5 are presented for all 400 meshes used as keyframes and a subset of 170 keyframes, which were selected from the entire sequence by minimizing the overall reconstruction error of linear interpolated meshes.

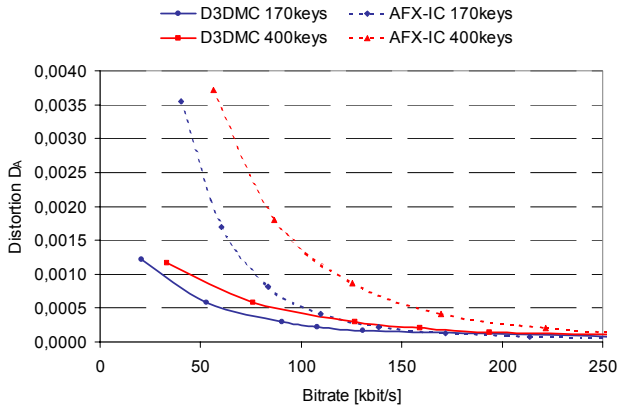

Fig. 5: Distortion over bit-rate for D3DMC and AFX-IC with 170 and 400 keyframes (GOM400), Chicken Crossing sequence

Comparing D3DMC and AFX-IC shows similar results for both cases (red and blue curves): D3DMC performs better than AFX-IC especially for lower bitrates. Here, the bitrate is reduced by 50% at a distortion measure of 0,001. Comparing the different keyframe numbers for D3DMC (solid curves), a decrease of 30% can be achieved by representing the sequence with 170 instead of the full 400 keyframes.

Fig. 6 visualizes the reconstruction error. The original mesh is shown on the left, AFX-IC in the middle and D3DMC on the right. Both color codes have a maximum value of 0.078 to allow better comparison. The color histograms show, how the reconstruction quality has improved for D3DMC in comparison to AFX-IC at the same bitrate.

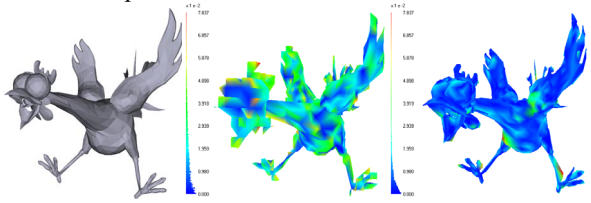

Fig. 6: Original mesh (left) and reconstruction error using AFX-IC at 125 kBit/s (middle) and D3DMC at 126 kBit/s (right).

3.2 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. 7. 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, since surface triangles of the 3D object geometry are only associated by 2D texture coordinates within the object region. Finally, the video is encoded using standard H.264/AVC syntax.

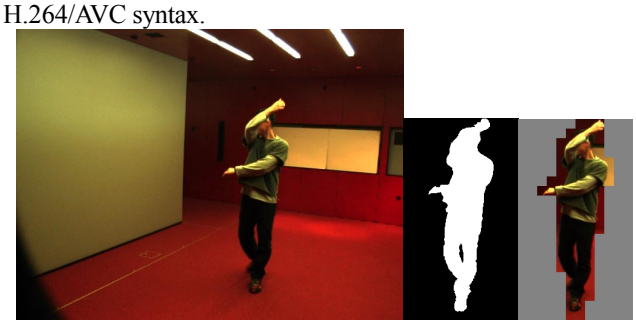

Fig. 7: Shape-oriented preprocessing for H.264/AVC.

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 with and without shape information (pink and dark blue curves). This shape information can be recovered from the 3D geometry by projecting the 3D mesh into each appropriate texture sequence. Therefore the dark blue curve in Fig. 8 is shifted to the left (shape bits subtracted), i.e. to lower bitrates, compared to the pink curve including shape bits.

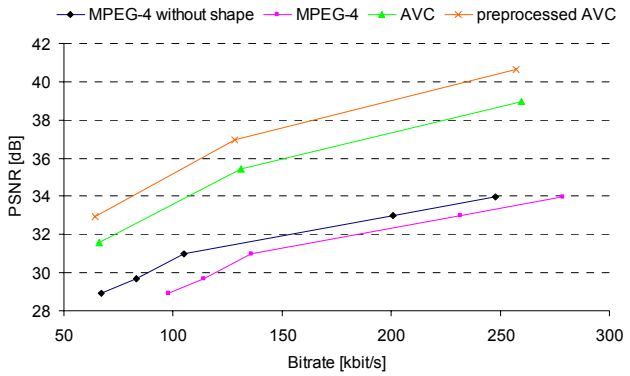

Fig. 8: Object PSNR for several codecs over bit-rate.

A typical example result for one sequence is shown in Fig. 8. For such sequences with relatively static background as shown in Fig. 7 (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

4. CONCLUSIONS

In this paper we have presented a standard-conform representation for 3D video objects in interactive applications. Furthermore, we presented an extraction algorithm to reconstruct 3D geometry from the original camera videos and combine the obtained meshes with texture information. For realistic rendering, we apply view-dependent multi-texturing. Beside the scene description, we also investigated coding methods for 3D geometry and textures. The presented predictive mesh compression D3DMC outperforms existing technology, whereas for the compression of video textures, H.264/AVC was applied to preprocessed data.

5. ACKNOWLEDGEMENTS

This work is supported by EC within FP6 under Grant 511568 with the acronym 3DTV.

We would like to thank the Computer Graphics Lab of ETH Zurich for providing the Doo Young multi-camera data set. Also 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] B. Goldlücke, and M. Magnor, "Real-time Microfacet Billboarding for Free-viewpoint Video Rendering", *Proc. ICIP*, Barcelona, Spain, Sep. 2003.

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

[3] ISO/IEC JTC1/SC29/WG11, "ISO/IEC 14496-16/PDAM1", Doc. N6544, Redmont, WA, USA, July 2004.

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

[5] 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.

[6] 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.

[7] 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.

[8] W. Matusik, C. Buehler, and L. McMillan, "Polyhedral Visual Hulls for Real-Time Rendering", *Proc. Eurographics Workshop on Rendering* 2001.

[9] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image-Based Visual Hulls", *Proc. SIGGRAPH 2000*, pp. 369–374, 2000.

[10] K. Müller, 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.

[11] K. Müller et al., "Predictive Compression of Dynamic 3D Meshes", *Proc. ICIP 2005, International Conference on Image Processing,* Genova, Italy, Sept. 2005.

[12] A. Smolic et al., "Free Viewpoint Video Extraction, Representation, Coding, and Rendering", *Proc. ICIP*, Singapore, Oct. 24.-27. 2004.

[13] A. Smolic, and P. Kauff, "Interactive 3D Video Representation and Coding Technologies", *Proceedings of the IEEE, Special Issue on Advances in Video Coding and Delivery*, vol 93, no. 1, Jan. 2005.

[14] 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.

[15] G. Taubin, „Curve and Surface Smoothing Without Shrinkage", *International Conference on Computer Vision (ICCV '95)*, pp. 852-857, 1995.

[16] 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.

[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. Symposium on Interactive 3D graphics*, pp. 65-74, 2003.

[18] S. Würmlin, E. Lamboray, and M. Gross, "3D video fragments: dynamic point samples for real-time free-viewpoint video", *Computers and Graphics*, vol. 28, no. 1, pp. 3-14, Elsevier Ltd, 2004.

[19] J. Zhang, and C.B. Owen, "Octree-based Animated Geomtery Compression", *DCC'04, Data Compression Conference*, Snowbird, Utah, USA, pp. 508-517, Mar. 2004.