Karsten Müller, Philipp Merkle, and Thomas Wiegand

Compressing Time-Varying Visual Content

Generating 3-D scene representations

The problem of sampling and representing three-dimensional (3-D) visual content is one of the fundamental challenges in the area of 3-D signal processing. The basic goal is an efficient representation of a 3-D scene that allows the navigation around that scene, obtaining different viewpoints through resampling. To solve this problem, various representations of 3-D content have been proposed: Lightfields [1], multiview representations [2], two-dimensional (2-D) images with depth or disparity information [3], and volumetric or wireframe models [4]. These representations fill a range between image-based and geometry-based methods, which correspond to different methods of sampling the 3-D scene using cameras. Since these representations are rather diverse, possible compression approaches also differ significantly. A block diagram of such compression methods is shown in Figure 1.

Here, image-based representations are either 2-D single or multiview video (including lightfields), while geometry is represented by depth (or disparity) data or 3-D mesh data. Each image and geometry representation has its underlying adapted compression algorithm. This article

Digital Object Identifier 10.1109/MSP.2007.905697

1053-5888/07/$25.00©2007IEEE

investigates compression approaches for 3-D scene representations, where image and geometry are combined. The approaches exemplified in this article mostly focus on work in which the authors have participated.

In recent years, the complete chain from content recording to display technologies for 3-D has been developed. 3-D content is generated by multicamera setups or via 3-D modeling programs and needs to be transmitted and distributed very efficiently. Therefore, compression of 3-D content is an essential topic within the 3-D chain. Thus, compression technologies for different 3-D representations are also considered in standardization committees, such as International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Moving Picture Experts Group (MPEG) or International Telecommunication Union (ITU-T) Video Coding Experts Group (VCEG). Single-view and multiview video coding, as well as depth-based coding, are currently being worked on in the Joint Video Team (JVT), a collaboration between MPEG and VCEG, while 3-D and mesh coding is being standardized in MPEG's 3-D Graphics Compression (3DGC) group.

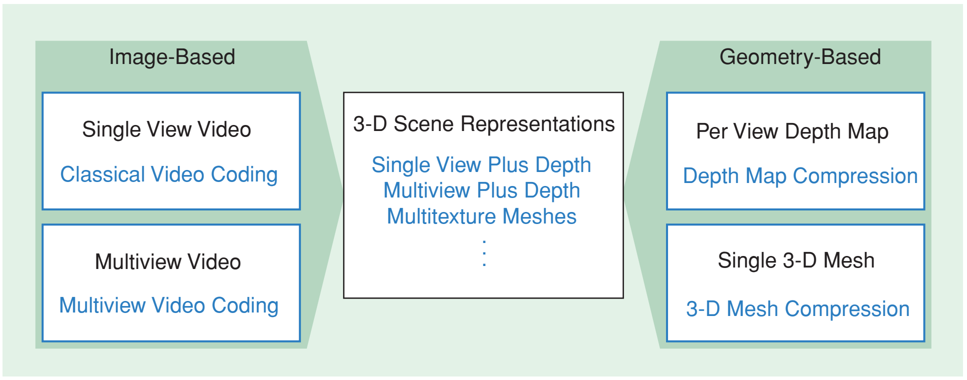
In the next section, single-view and multiview video coding is presented. The combination of single-view or multiview video with depth maps representing the underlying geometry is described. Finally, 3-D mesh coding is presented.

MULTIVIEW VIDEO CODING

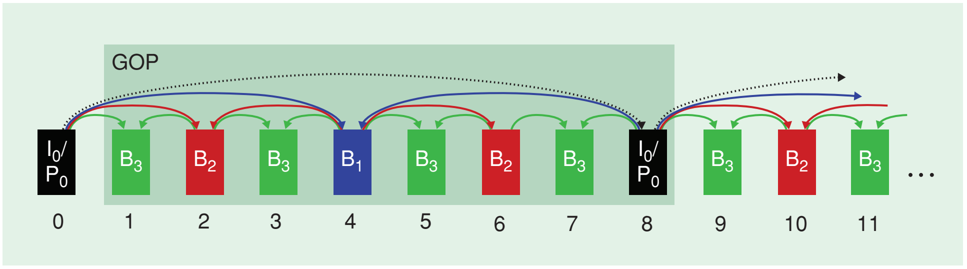
The most efficient algorithm for single-view video compression is currently H.264/advanced video coding (AVC) [5], [6]. Hence, it typically serves as a starting point for modern video coding and multiview video coding, where a 3-D scene is captured by a number of cameras. Since some of the cameras share common content, a coding gain can be achieved with multiview coding, in comparison to single-view coding, when exploiting the statistical dependencies between the camera views in addition to temporal statistical dependencies within each sequence. Multicamera settings can range in practice from simple one-dimensional (1-D) linear and arched arrangements to complex spatial position distribution patterns. However, any camera arrangement can be reordered into a 1-D linear array, where the spatially neighboring camera relations are maintained. Thus, the underlying multiview coding structure is still that basic 2-D temporal/inter-view array with appropriate neighborhood dependencies. This approach was followed by multiview compression proposals that finally led to a project for an amendment of H.264/AVC for multiview video coding (MVC).

A multiview coder basically consists of multiple single-view coders. Therefore, they both use similar temporal coding structures, where a sequence of successive pictures is coded as intra (I), predictive (P), or bipredictive (B) pictures. For I pictures, the content is only predicted from the current picture itself, while P and B picture content is also predicted from other temporal reference pictures. With the concept of multiple reference pictures, a selection of pictures used as a reference can be made [7]. One approach for further improving coding efficiency is the use of hierarchical B pictures [8], where B pictures can be references for other B pictures. Thus, a B picture hierarchy is created, sometimes denoted as $B_1 \dots B_N$ for the first, up to the Nth hierarchical level, as shown for a hierarchy with three levels in Figure 2. Here, the distance between I/P pictures is 8, so that we refer to a group of pictures (GoP) size of 8.

For multiview video coding, the single-view concepts are extended [9]. Here, a current picture in the coding process can have temporal as well as inter-view reference pictures for motion-compensated prediction. An example of a multiview coding structure with five linearly arranged cameras and GoP size of 8 is shown in the complex prediction structure in Figure 3. This figure illustrates how the advantages of hierarchical B pictures from Figure 2 are combined with inter-view prediction, without any changes regarding the temporal prediction structure. For view "Cam 1," the prediction structure is identical to single-view coding.



[FIG1] Combining image-based and geometry-based representations. Each image, geometry, and combined representation (black) has its compression method (blue).



[FIG2] Gaining coding efficiency by staged arrangement: Hierarchical reference picture structure for temporal prediction.

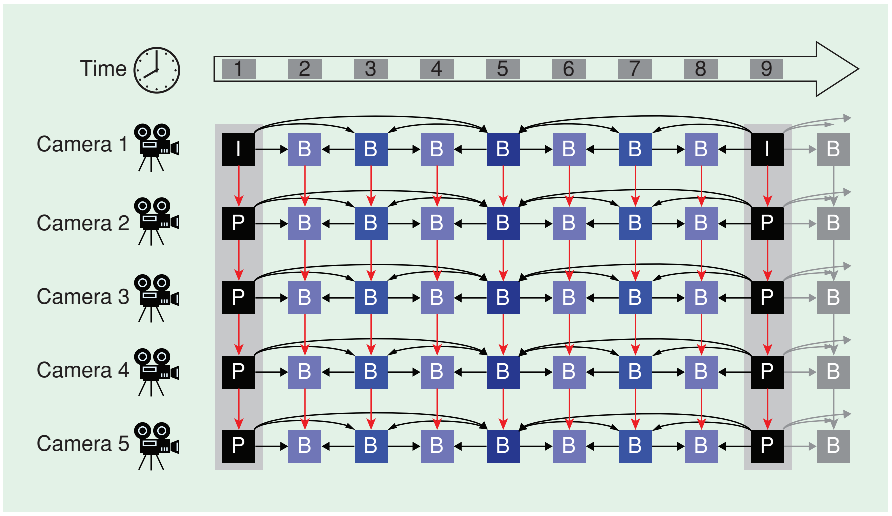
In the experiments conducted by MPEG, multiview coding gained up to 1.6 dB in peak signal-to-noise ratio (PSNR) with respect to simulcasting single-view coding using hierarchical B pictures [10], as shown in Figure 4. Here, the average gain across all MPEG test sequences was 0.9 dB.

Ongoing work concentrates on technologies to improve the exploitation of statistical inter-view dependencies. One technology under investigation is illumination compensation, since different cameras in a multiview setting can introduce different colors for identical scene content, e.g., due to different reflections from a surface. In such cases, illumination compensation is not only required for better coding efficiency, but much more so for a possible 3-D scene reconstruction from all views. Here color homogeneity across all views is required to avoid color artifacts if an object surface is reconstructed from different views. However, it should also be noted that a variety of illumination effects exist which cannot be equalized across views, e.g., specular surface reflections that can occur in single views only.

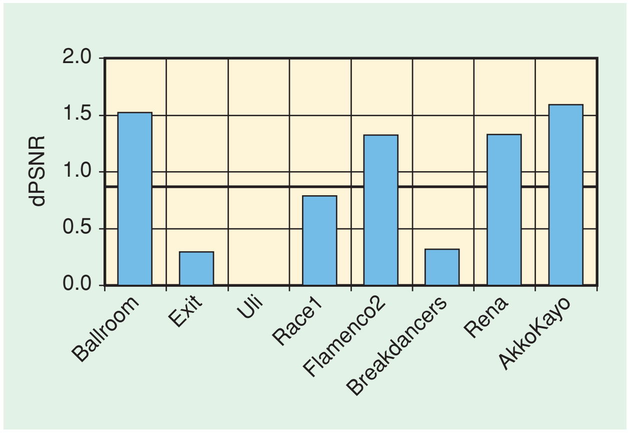
CODING OF VIDEO+DEPTH REPRESENTATIONS

In the previous section, we described how 2-D video content of multiple cameras can be efficiently compressed. Therefore, without any further scene analysis processing, a decoded MVC stream typically only allows for selecting from the original camera views when navigating around the 3-D scene. However, for seamless navigation in between the original views, additional scene information is required. One approach is to provide depth information for each view, as shown in Figure 5.

Here, the color video signal is associated with the corresponding depth signal. This depth information is either additionally recorded by special cameras or estimated from the 2-D views. Additionally, camera parameters can be transmitted as side information requiring only a few bits per view. With the camera parameters, a depth-based decoder knows the position of each camera in the original scene and can provide seamless navigation within the range covered by the original cameras if the 3-D scene has been correctly sampled [11]. Here, the depth information is used to convert the 2-D image sample into 3-D points.



[FIG3] Multiview coding structure with hierarchical B pictures for both temporal and inter-view prediction (red arrows).



[FIG4] Multiview coding gains in comparison to simulcasting single-view coding. On average, a coding gain of 0.9 dB is achieved.

CODING OF SINGLE-VIEW VIDEO+DEPTH REPRESENTATIONS

A special scenario for depth-based coding is a video+depth representation, a derivative of classical stereo coding, where two cameras record scene content that can be used for presenting 3-D content on stereoscopic displays. In conventional stereo video, two views are transmitted. Similar to MVC, temporal and inter-view dependencies between the left and right view are exploited. In the video+depth representation, only one view associated with depth information is transmitted [12]. This depth information of the scene is scaled and quantized to fit a gray-value monochromatic image at 8 b or 256 values per pixel, as illustrated in Figure 5. In this case, depth data is scaled perspectively, i.e., the inverse depth $1/z$ is quantized uniformly to provide a finer depth quantization for near objects and coarser quantization for distant objects. Future coding approaches may also choose higher depth resolutions as used in computer graphics applications to represent more details. The depth data is used to reconstruct the second view at the decoder. To allow for correct stereoscopic viewing, the depth data range needs to be converted back to the range of original scene depth information. Therefore, MPEG specified a corresponding container format, called "representation of auxiliary video and supplemental information," for video+depth data [13]. Since depth data can typically be easier compressed than color data, a coding gain can be achieved in comparable to classical stereo video coding. One problem occurs at depth discontinuities in connection with traditional coding, where high signal frequencies are usually omitted. This leads to edge distortions in the reconstructed view, as explained in detail in the next section. Due to missing information in the second view, another problem occurs at depth discontinuities, where areas are occluded in the transmitted view while being visible in the omitted view. One method to mitigate the problem is to fill such areas with spatially neighboring content.

CODING OF MULTIVIEW VIDEO+DEPTH REPRESENTATIONS

If each view of a multiview sequence is associated with depth information, the approach is called a N-view + N-depth representation. In the current coding approaches for this representation, depth is treated as monochrome 2-D video where the depth values are quantized according to the appropriate luminance range, i.e., 8 b resulting in 256 distinct depth values. Although depth data is treated as monochrome video, it typically differs statistically from the luminance component of the color video content. While color video typically contains detailed texture information, depth data typically contains large homogeneous regions of slow-changing values, as shown in the depth map in Figure 5. Additionally, object boundaries may cause abrupt changes in depth values. Therefore, depth data mostly contains very low and very high frequencies. In regular video coding, high signal frequencies are omitted at higher compression rates, leading to an increasingly blurry visual appearance. While this effect only leads to slight visual degradation in regular videos, the effects for depth coding are much more severe. An example of N-view + N-depth coding results is shown in Figure 6.

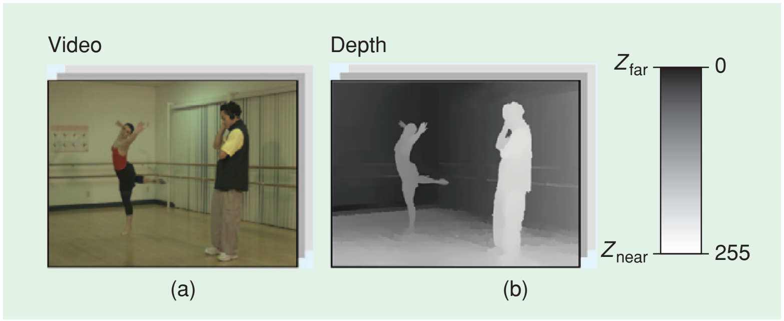
The diagram shows PSNR-Y values for a camera path through a linear camera arrangement. Each value represents the temporal average over the entire sequence. Here, intermediate views are generated from original depth data, as well as from coded depth data, and both are compared to calculate PSNR-Y values. The graphs show a considerable decrease in reconstruction quality for intermediate views, where wrong depth values lead to distortions. For original views, depth values have no influence at all and PSNR-Y results are therefore identical to pure video coding results. Furthermore, intermediate view quality further decreases with higher depth compression rates as shown for the increased view/depth coding ratio from 1:1 to 2:1 and 4:1. The associated visual results are presented in Figure 7. The depth information can be used to generate new intermediate views between original camera positions, as well as to generate a 3-D scene representation. In both cases, wrong depth values lead to projection errors of the color information, causing strong visual distortions during navigation [11], as shown in Figure 7.

Scene parts may even drift apart. This problem becomes especially severe at object boundaries: here the high-frequency depth discontinuities and sharp edges are highly visible. In addition, object boundaries very often indicate a separation line between two scene objects of different color. After reconstruction, depth edge smoothing erroneously leads to an area between these objects with a variety of depth values and different colors along the boundary.

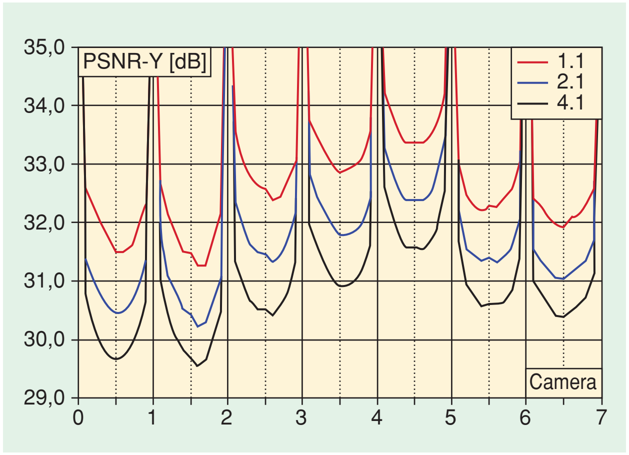
The preservation of depth edges requires coding of depth data at about one-third of the color information, although the depth PSNR values are much higher than the color values [9]. Current efforts try improving on this by taking the statistical properties of depth signals into account. Moreover, future research is also likely to concentrate on combined coding and reconstruction methods which analyze overall reconstruction quality already during encoding.

3-D MESH CODING

While multiview and depth-based 3-D representations often only provide limited user navigation, mesh-based free viewpoint video is typically a representation form with free navigation, where any viewpoint is possible. As a consequence, this approach consists of a time-varying 3-D geometry and original textures from each camera or a selected subset of cameras. These texture sequences can either be kept as original camera sequences or merged into one single texture sequence which contains the entire object surface from all cameras. In both approaches, the textures are projected onto the 3-D mesh. In the first case, each separate texture is simply projected back from the 2-D camera plane onto that portion of the 3-D object where it originally came from. Here, view-dependent texture weighting



[FIG5] Depth maps are an extension to classical video. Shown here is an example for a video+depth data representation, including color and accompanying depth with depth range scaling.



[FIG6] Results of objective evaluation of a virtual camera path for compressed N-view+N-depth for ballet sequence with 1:1, 2:1, and 4:1 view/depth coding ratio.

can be applied after decompression and reconstruction at the user's side to improve visual rendering quality. Especially if camera sequences differ in their lighting conditions, a pure texture blending approach with original cameras would cause illumination artifacts. In the second case, the single merged texture is projected onto the entire object.

For the geometry, animated triangle meshes are utilized. Typically, these meshes are originally created through a complex 3-D reconstruction process from the original camera views, e.g., [4]. Since the representation with geometry and textures is twofold, its compression also consists of two parts: static and dynamic mesh compression for the time-varying 3-D geometry and 2-D video coding for the texture sequences. Although the underlying compression algorithms are rather diverse, they can be integrated into a single compression framework, as shown by MPEG-4 systems, where a hierarchical scene description with different types of audiovisual components exist, each with its associated optimal compression algorithms.

GEOMETRY CODING

In the area of 3-D geometry compression, different approaches have been introduced. Mostly, geometry compression was developed for 3-D triangle meshes, since this 3-D representation form is widely used in a variety of graphics applications and computer graphics hardware was optimized towards such representations. First, coding approaches were developed for static meshes, whereas later compression methods for dynamic meshes were added to cover time-varying 3-D meshes. For textures that are mapped onto such 3-D meshes, very basic compression methods have been implemented directly onto graphics cards in order to minimize texture buffer sizes. For transmission purposes, the MPEG-4 standard provides compression tools for animated meshes.

For the actual 3-D geometry coding, a number of algorithms have been proposed mainly for wireframe representations, as they are widely used in computer graphics. In the case of mesh-based free viewpoint video, animated mesh sequences are used. The sequence contains a first mesh with 3-D vertex positions as well as connectivity information, which defines polygonal patches or 3-D faces from the vertices. For better compression results, the following meshes in the sequence share this connectivity, and thus need only to be represented by their vertex positions or 3-D motion. In analogy to video coding, the first mesh is called intra (I) mesh and the remaining ones are called predictive (P) meshes. In this way, mesh coding algorithms can also exploit temporal dependencies in a mesh sequence. The coding principles for the basic components of a mesh sequence are shown in Figure 8, where the sequence is divided into static and dynamic mesh compression for I and P meshes, respectively.

The three main components are 3-D vertices, faces, and motion. The associated underlying coding principles and spatial, topological, and temporal decorrelation are shown together with common coding approaches which are based on these principles.

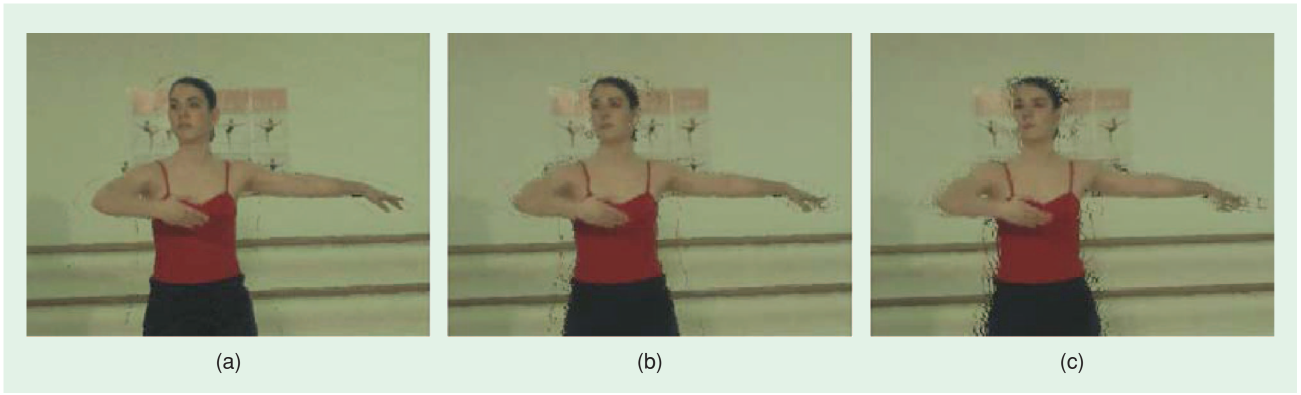
STATIC MESH CODING

For the I mesh, a static mesh coding approach is required, such as face-based Edgebreaker [14] or MPEG-4 3-D mesh compression (3DMC) [15], edge-based FaceFixer [16], or vertex-based methods [17]. An overview of static mesh compression approaches can be found in [18]. The principles of static mesh compression are spatial and topological decorrelation. A static compression algorithm has to code the mesh data types associated with static scene content, such as 3-D vertices, faces, texture, coordinates, color attributes, appearance, etc.

For mesh sequences, the most important components of an I mesh are the 3-D vertices and faces. One example is 3DMC, which was standardized in MPEG-4. 3DMC uses topological surgery [19] to cut the mesh into connected triangular stripes. In this form, the mesh can be treated as a graph and its spanning tree can be encoded through a traversal of mesh elements. The static mesh coder also supports progressive transmission, i.e., the received bit stream can be terminated at any position and a reconstruction is possible from partially received streams. Similar, a base mesh can already be reconstructed from the first parts of the bit stream, while further parts increase the reconstruction quality. Thus, incremental rendering is supported and the reconstruction can always start immediately after receiving the first parts of the bit stream.

DYNAMIC MESH CODING

For P meshes, dynamic compression approaches are used, exploiting temporal as well as spatial dependencies. Here, a



[FIG7] Results of subjective evaluation with original and compressed *N*-view+*N*-depth for Ballet sequence: (a) original, (b) 1:1, and (c) 4:1 view/depth coding ratio.

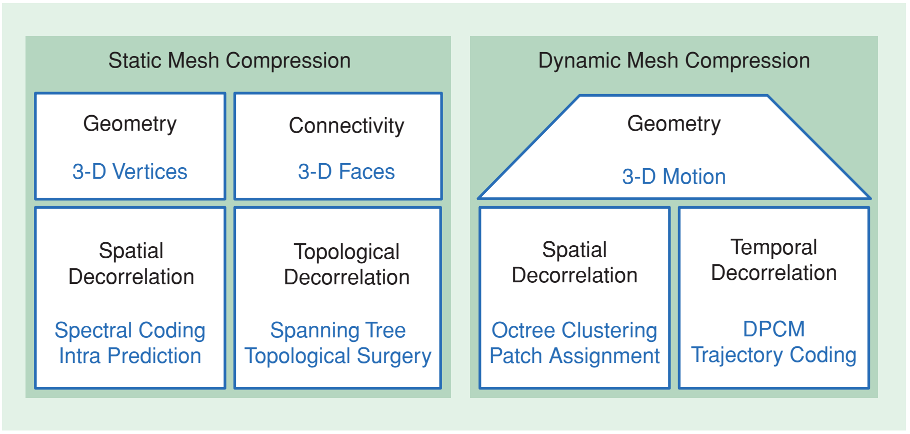
number of different approaches exist, e.g., approaches based on principal component analysis [20] and the associated linear prediction [21], the vertex-based method Dynapack [22], interpolator compression [23], or surface patches with residual error transform coding [24]. There are rate-distortion-optimized frameworks, based on temporal differential pulse code modulation (DPCM) and spatial octree clustering [25] and a number of other methods.

Exploitation of Spatial Statistical Dependencies

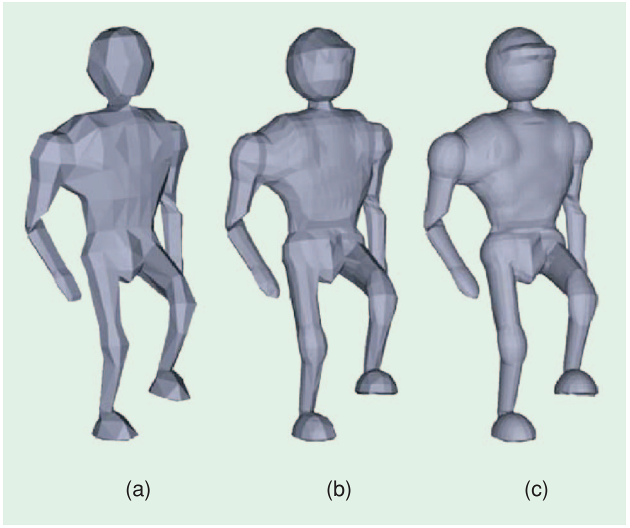
The exploitation of spatial statistical dependencies is typically achieved by spatial clustering of motion vectors, like octree clustering [25], [26] or patch assignment [24]. The first approach applies a regular hierarchical structure to the motion vectors by splitting the initial bounding cube of a mesh into eight octants, creating eight new subcubes in the next level. Each subcube contains a set of motion vectors. The statistics are analyzed and, if all enclosed motion vectors have similar direction, then they can be replaced efficiently by a few substitution vectors, e.g., the eight corner vectors of the cube or a single average motion vector. In the case of very irregular motion, this cube is further subdivided into eight octants, and the process continues until either certain motion vector homogeneity is reached or a subcube only contains a minimum number of motion vectors, typically 8 or below. At this lower limit, the vectors are coded directly. The octree splitting process can be controlled via a minimum reconstruction error [26], where the intended motion vectors are interpolated from their representative vectors and compared to the original, uncompressed motion vectors. Another method uses a rate-distortion framework where octree splitting is controlled by varying a Lagrangian parameter λ [25]. In this method, the functional $D + \lambda R$ is minimized for all possible octree subdivisions from the single bounding cube down to the finest split with a minimum of enclosed motion vectors in each cube. Distortion D and rate R are calculated for a range of substitute vector quantizations—e.g., 1–16 b/vector component—and the optimal setting of subdivision structure/quantization is obtained, which yields the minimum cost value.

The second method for spatial subdivision is patch assignment, where regions of similar motion are obtained [24]. In this process, a mesh sequence is iteratively coarsened by omitting 3-D vertices through edge collapsing. This method is carried out, until a certain global error measure between original and coarse mesh is reached. The remaining surface regions also cluster the object surface: All original surface polygons, lying in one coarse patch, are clustered together. Thus, in contrast to regular octree subdivision, the resulting regions are more adapted to the object's motion. The patches are finally described by the parameters of an affine motion model, such that all original motion vectors of a surface patch are replaced by an average motion, similar to regular clustering methods. Finally, a method known from skinning animation is applied, where the affine motion vectors of adjacent patches are weighted and linearly combined to obtain a motion vector field for the entire object.

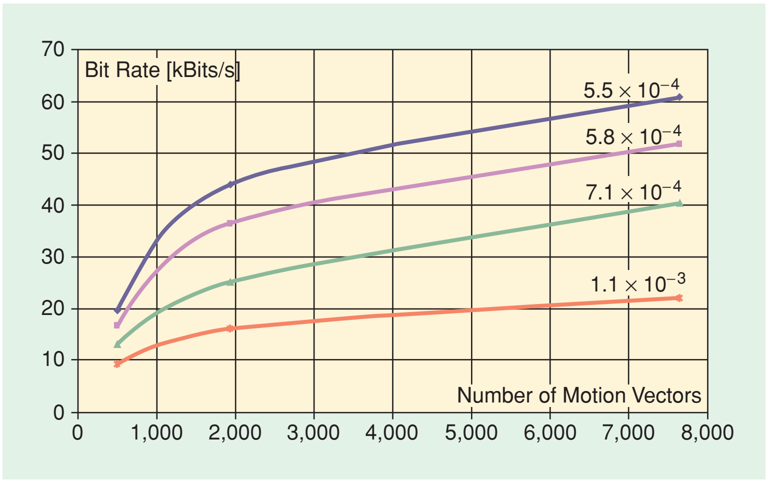
Both methods apply the same principle of motion vector bit rate reduction, replacing a number of similar motion vectors by fewer motion vectors. Of course, the efficiency of motion vector clustering depends on the degree of local motion. In general, it can be assumed that object motion follows a certain pattern such that neighboring motion vectors are similar. In practice, the number of vertices and faces in 3-D meshes has increased dramatically in the last decades to better approximate natural surfaces. Figure 9 gives an example from the "humanoid"



[FIG8] **Basic mesh-coding components for a free viewpoint video functionality: Geometry and connectivity primitives (top row) and underlying coding principle to achieve the associated decorrelation.**



[FIG9] **Mesh resolution has increased over the last decades: Representative single-mesh example from the "humanoid" animation sequence with (a) 498, (b) 1,940, and (c) 7,646 vertices.**



[FIG10] Gaining better compression at higher vertex numbers: Bit rate versus number of motion vectors at different distortions for the "humanoid" sequence, using hierarchical octree clustering for spatial decorrelation.

animation sequence, showing a single mesh with 498, 1,940, and 7,646 vertices or motion vectors, respectively.

Increasing computational power further strengthens this development. Therefore, compression for transmission becomes an important issue. Here, spatial clustering has one major advantage. If an object's surface is approximated by a much finer mesh, the number of vertices increases significantly. However, the degree of local motion remains and many more motion vectors are clustered per octree volume or surface, similarly enhancing coding efficiency. As a consequence, the bit rate for dynamic mesh compression only increases moderately (logarithmically, as indicated in recent results [25] and shown in Figure 10) with increasing number of motion vectors, which makes spatial clustering much more efficient at finer mesh resolutions. Figure 10 shows an example for the bit rates versus number of motion vectors for the different resolutions of the "humanoid" mesh in Figure 9. Here, the bit rate was evaluated for the "humanoid" mesh at three resolutions for four different distortions as a measure of mean-squared error in 3-D space. All curves show a logarithmic dependency of the bit rate versus the number of motion vectors, as described above.

Exploitation of Temporal Statistical Dependencies

The exploitation of temporal statistical dependencies is the second principle in dynamic mesh compression. Similar to 2-D video coding, successive meshes have similar motion. This can be exploited by DPCM methods [25] or by temporal transform coding of residual errors [24]. These residual errors are the spatial displacement between original and reconstructed 3-D vertices of the mesh sequence, obtained during the skinning method described above. The three coordinates of the displacement vectors are coded separately by applying a 1-D discrete cosine transform. If the entire mesh sequence is known, residual error transform coding is the most efficient method of temporal decorrelation but introduces a considerable delay, i.e., the size of the transform. On the other hand, DPCM only causes a one-mesh delay, which is more desirable in streaming applications, but has so far shown some loss in coding efficiency.

The spatially and temporally decorrelated signal is finally entropy coded. Here, the most efficient way is to use adaptive arithmetic coding like context-adaptive binary arithmetic coding (CABAC), regardless of the previous decorrelation methods. This coding algorithm efficiently adapts itself to the signal statistics using context models and probability estimation to follow changing statistics. The binarization in CABAC follows the basic principle of entropy coding by assigning small code words to symbols with high probability and long code words to least probable symbols.

TEXTURE CODING

Textures or texture sequences can occur in the form of single or multiple textures and as both still image or video textures. Single textures can also be created from multiple camera views by combining them into one unwrapped texture. For this case, coding is carried out much like single-view coding or regular image and video coding. Single textures have a relatively small size in comparison to multitextures but lack the ability of natural dynamic lighting and illumination changes during rendering.

For multitexture sequences, compression algorithms similar to MVC are selected, depending on the degree of inter-view correlation between the camera views. Often, mesh-based free viewpoint video scenarios are recorded with a limited number of cameras distributed all around the scene to capture as much content as possible. Typically, inter-view dependencies between the cameras' views in this case are rather small such that, so far, MVC did not gain much compression efficiency in comparison to single-view coding.

In multitexture approaches, natural appearance and lighting conditions can be included and experienced during scene navigation. Each original camera view already includes the natural illumination of that scene, such that natural lighting is already provided but only at original camera positions. Here, approaches have been taken to provide smooth interpolation between original camera views, such as lightfield rendering [1], [2], lightfield mapping [28] or unstructured lumigraph rendering [29]. For the associated coding process, only the camera plane normal vectors need to be transmitted as auxiliary scene data. This has been included into the MPEG-4 multitexture coding [30].

CONCLUSIONS

For the variety of time-varying 3-D representations, efficient compression technologies are available which consider their specific properties. This article includes descriptions of multiview coding, depth-based coding, and 3-D mesh-based coding with textures. Although all compression methods are highly adapted to their corresponding 3-D representation methods, they all share some common coding principles. Uncompressed

3-D content requires at least N times the data rates of uncompressed video, if N is the number of cameras used in a multiview capture system. For depth-based and mesh-based free viewpoint representations, depth and geometry respectively require data rate as well. Consequently, for distributing 3-D content, efficient content adaptive compression is essential. Therefore, all presented compression methods are dealt with by standardization organizations. As an example, the MPEG-4 scene representation includes a variety of different 3-D scene representations, and the underlying compression standards have been mentioned along with the methods described in this article.

ACKNOWLEDGMENTS

We would like to thank Alexandru Salomie from Vrije Universiteit Brussel for providing the "humanoid" sequence and the Interactive Visual Media Group of Microsoft Research for providing the Ballet data set. Moreover, we are in particular thankful to Aljoscha Smolic.

AUTHORS

Karsten Müller (kmueller@hhi.de) received the Dr.-Ing. degree in electrical engineering and Dipl.-Ing. degree from the Technical University of Berlin, Germany, in 2006 and 1997, respectively. He has been with the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Berlin, since 1997, where he is currently project manager. His research interests are mainly in the field of representation, coding and reconstruction of 3-D scenes in free viewpoint video scenarios and coding, multiview applications, and combined 2-D/3-D similarity analysis. He has been involved in MPEG activities, where he contributed to visual MPEG-7 descriptors and MPEG-4.

Philipp Merkle received the Dipl.-Ing. degree in electrical engineering from the Technical University of Berlin, Germany, in 2006. He joined the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut (HHI), Berlin, Germany, as a student in 2003 and became a research assistant in 2006. He has been involved in several projects focused on multiview video coding, free viewpoint video, 3-D scene reconstruction, and augmented reality. His research interests include representation and coding of multiview video scenes, free viewpoint video, and 2-D and 3-D video-based rendering. He has been involved in ISO standardization activities, where he contributed to the development of the MPEG-4 multiview video coding standard.

Thomas Wiegand is the head of the Image Communication Group in the Image Processing Department of the Fraunhofer Institute for Telecommunications—Heinrich Hertz Institute Berlin, Germany. He received the Dr.-Ing. degree from the University of Erlangen-Nuremberg, Germany, in 2000. Since 1995, he is an active participant in standardization for multimedia. In October 2000, he was appointed as the associated rapporteur of ITU-T VCEG. In December 2001, he was appointed as the associated rapporteur/cochair of the JVT. In January 2005, he was appointed as associated chair of MPEG Video. His research interests include video processing and coding, multimedia transmission, semantic image representation, and computer vision and graphics.

REFERENCES

[1] M. Levoy and P. Hanrahan, "Light field rendering," in *Proc. ACM SIGGRAPH*, pp. 31–42, Aug. 1996.

[2] P. Debevec, C. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry- and image based approach," in *Proc. ACM SIGGRAPH*, pp. 11–20, 1996.

[3] C.L. Zitnick, S.B. Kang, M. Uyttendaele, S.A.J. Winder, and R. Szeliski, "High-quality video view interpolation using a layered representation," *ACM SIGGRAPH and ACM Trans. Graphics*, pp. 600–608, Aug. 2004.

[4] S.M. Seitz and C.R. Dyer, "Photorealistic scene reconstruction by voxel coloring," in *Proc. Computer Vision and Pattern Recognition Conf.*, pp. 1067–1073, 1997.

[5] *Advanced Video Coding for Generic Audio-Visual Services*, ITU-T and ISO/IEC JTC 1, ITU-T Recommendation H.264 and ISO/IEC 14496-10 (MPEG-4-AVC), Version 1: May 2003, Version 2: Jan. 2004, Version 3: Sept. 2004, Version 4: July 2005.

[6] T. Wiegand, G.J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, July 2003.

[7] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 70–84, Feb. 1999.

[8] H. Schwarz, D. Marpe, and T. Wiegand, "Hierarchical B pictures," Joint Video Team, Poznan, Poland, doc. JVT-P014, July 2005.

[9] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Efficient compression of multi-view depth data based on MVC," in *Proc. IEEE 3DTV Conf.*, Sept. 2007.

[10] K. Müller, P. Merkle, H. Schwarz, T. Hinz, A. Smolic, T. Oelbaum, and T. Wiegand, "Multi-view video coding based on H.264/AVC using hierarchical B-frames," in *Proc. PCS Picture Coding Symp. 2006*, Beijing, China, Apr. 2006.

[11] P. Merkle, A. Smolic, K. Müller, and T. Wiegand, "Multi-view video plus depth representation and coding," in *IEEE Int. Conf. Image Processing (ICIP'07)*, San Antonio, TX, USA, Sept. 2007.

[12] C. Fehn, P. Kauff, M. Op de Beeck, F. Ernst, W. IJsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, and I. Sexton, "An evolutionary and optimised approach on 3D-TV," *Proc. Int. Broadcast Conf.*, pp. 357–365, Sept. 2002.

[13] ISO/IEC JTC1/SC29/WG11, "ISO/IEC FDIS 23002-3 representation of auxiliary video and supplemental information," Doc. N8768, Jan. 2007.

[14] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes," *IEEE Trans. Visualization Comput. Graphics*, vol. 5, no. 1, pp. 47–61, 1999.

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

[16] M. Isenburg and J. Snoeyink, "Face fixer: Compressing polygon meshes with properties," in *Proc. SIGGRAPH 2000*, pp. 263–270, July 2000.

[17] C. Touma and C. Gotsman, "Triangle mesh compression," in *Proc. Graphics Interface 98*, pp. 26–34, 1998.

[18] P. Alliez, "Recent advances in compression of 3D meshes," in *Proc. EUSIPCO 2005, 13th European Signal Processing Conf.*, Sept. 2005.

[19] G. Taubin and J. Rossignac, "Geometric compression through topological surgery," *ACM Trans. Graphics*, pp. 84–115, 1998.

[20] M. Alexa and W. Müller, "Representing animations by principal components," *Proc. EUROGRAPHICS 2000*, vol. 19, no. 3, pp. 411–418, 2000.

[21] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Elsevier Computer & Graphics 28*, pp. 25–34, 2004.

[22] L. Ibarria and J. Rossignac, "Dynapack: Space-time compression of the 3D animations of triangle meshes with fixed connectivity," in *Proc. 2003 ACM SIGGRAPH/ Eurographics Symp. Computer Animation*, 2003.

[23] E.S. Jang, J.D.K. Kim, S.Y. Jung, M.J. Han, S.O. Woo, and S.J. Lee, "Interpolator data compression for MPEG-4 animation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, pp. 989–1008, 2004.

[24] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning prediction scheme for dynamic 3D mesh compression," in *Proc. SPIE Conf. Mathematics of Data/ Image Pattern Recognition, Compression, and Encryption with Applications IX*, vol. 6315, pp. 631502, Aug. 2006.

[25] K. Müller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand, "Rate-distortion-optimized predictive compression of dynamic 3D mesh sequences," *Signal Processing: Image Commun. (Special Issue on Interactive Representation of Still and Dynamic Scenes)*, vol. 21, no. 9, pp. 812–828, Oct. 2006.

[26] J. Zhang and C.B. Owen, "Octree-based animated geometry compression," in *Proc. DCC'04, Data Compression Conf.*, pp. 508–517, 2004.

[27] D. Marpe, H. Schwarz, and T. Wiegand, "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 620–636, 2003.

[28] W.C. Chen, J.Y. Bouguet, M.H. Chu, and R. Grzeszczuk, "Light field mapping: Efficient representation and hardware rendering of surface light fields," in *Proc. ACM SIGGRAPH*, pp. 447–456, 2002.

[29] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "Unstructured lumigraph rendering," in *Proc. SIGGRAPH 2001*, pp. 425–432, 2001.

[30] ISO/IEC JTC1/SC29/WG11, "Text of ISO/IEC 14496-16/FDAM1 (morphing & textures)," Doc. N7400, July 2005.

[SP]