Rate-Distortion-Optimized Predictive Compression of Dynamic 3D Mesh Sequences

Karsten Müller, Aljoscha Smolic, Matthias Kautzner, Peter Eisert and Thomas Wiegand[1]

Keywords

3D Mesh, Dynamic Mesh Coding, Animation Coding, DPCM, Spatial Clustering, RD-Optimization

Abstract

Compression of computer graphics data such as static and dynamic 3D meshes has received significant attention in recent years, since new applications require transmission over channels and storage on media with limited capacity. This includes pure graphics applications (virtual reality, games) as well as 3DTV and free viewpoint video. Efficient compression algorithms have been developed first for static 3D meshes, and later for dynamic 3D meshes and animations. Standard formats are available for instance in MPEG-4 3D Mesh Compression for static meshes, and Interpolator Compression for the animation part. For some important types of 3D objects, e.g. human head or body models, facial and body animation parameters have been introduced. Recent results for compression of general dynamic meshes have shown that the statistical dependencies within a mesh sequence can be exploited well by predictive coding approaches. Coders introduced so far use experimentally determined or heuristic thresholds for tuning the algorithms. In video coding rate-distortion (RD) optimization is often used to avoid fixed thresholds and to select the optimum prediction mode. We applied these ideas and present here an RD-optimized dynamic 3D

[1] Correspondence address:
Karsten Müller
Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut,
Image Processing Department
Einsteinufer 37, D-10587 Berlin, Germany
Phone : +49-30-31002-225 Fax : +49-30-392-7200 email : kmueller@hhi.de

mesh coder. It includes different prediction modes as well as an RD cost computation that controls the mode selection across all possible spatial partitions of a mesh to find the clustering structure together with the associated prediction modes. The general coding structure is derived from statistical analysis of mesh sequences and exploits temporal as well as spatial mesh dependencies. To evaluate the coding efficiency of the developed coder, comparative coding results for mesh sequences at different resolutions were carried out.

1 Introduction

Efficient compression of computer graphics data becomes more and more important, since applications that require timely transmission over channels and storage on media with limited capacity become more and more popular. So far computer graphics data are mainly downloaded in advance and stored on spacious hard discs. Even real-time applications such as multi-user games rely on downloading almost all data and only exchanging some parameters in real-time, e.g. over the internet.

Nowadays, mobile phones are small computers with good graphics capabilities, though limited disc space and a limited channel capacity. Real-time graphics systems such as 3D visualized navigation systems require real-time transmission of tremendous amounts of data. Further, future applications such as 3DTV and free viewpoint video [25] make use of 3D graphics data to be transmitted in real-time and/or stored on media with limited capacity.

One type of such computer graphics data are polygonal 3D meshes as surface representations of 3D objects with 3D points and connectivity. Typically these are stored and transmitted in text or simple binary format posing a tremendous waste of capacities. First compression research for such data was focused on compression of static 3D mesh geometry in the format of vertex positions (points in 3D x, y, z) and connectivity (information about how to connect vertices to form surface patches) [22], [9]. Improvements on connectivity compression in the “Edgebreaker” coder from [22] were introduced in [8], [12], [27]. An overview on recent static mesh compression advances can be found in [2].

An important pre-condition for successful and wide application of digital media is the availability of international standard formats, providing interoperability while still enabling competition among equipment and content providers. ISO MPEG, as one of the international

standardization bodies, has recognized the importance of efficient 3D mesh compression and included a tool called 3D Mesh Compression (3DMC), for static meshes exploiting spatial dependencies of adjacent polygons, in MPEG-4 [14]. This standard is the first real multimedia format including video, audio, graphics, text, etc. in a single specification.

Later research was extended to compression of animated meshes and mesh sequences with changing vertex positions over time. In [20] a decomposition of the vertex position matrix is suggested, allowing a better decorrelation of different types of temporal mesh deformations. The deformation is described by special animation parameters, representing affine motion or free-form deformation. The residual between real mesh deformation and estimated animation parameter deformation is then coded. The temporal deformation is separated into low and high frequency motion - an approach that was later used in [24] to introduce a multi-resolution approach for dynamic mesh coding. Dynapack [11] analyzes spatial and temporal dependencies using a predictor to exploit similarities of neighboring points. Spatial dependencies are also exploited in [28], where mesh connectivity is transformed into special codes for coding vertex valences, i.e. how many edges coincide at a 3D vertex. Besides this prediction, 3D points are compressed directly and not represented by substitutes, an approach that is taken in [1]. Here, principal component analysis (PCA) of the geometry covariance matrix is carried out to reduce spatial correlation. By applying linear predictive coding to PCA components, a fast algorithm was introduced in [19]. A different approach was taken with Geometry Images [7] and Geometry Videos [4], where 3D geometry is transformed into 2D images and 3D vertices and normal vectors with *x*-, *y*- and *z*-coordinates are coded by R, G and B values of the image. Here, sophisticated mesh cutting needs to be applied to find a suitable mesh-to-image-mapping.

If motion within a sequence of meshes is approximately linear over certain periods of time, it can be represented by a few key meshes. Then only the key meshes need to be encoded and the intermediate meshes can be generated by interpolation. This drastically reduces the data rate. A corresponding algorithm that further exploits temporal statistical dependencies for compressing the key meshes by prediction was introduced in [17]. This approach was adopted as an extension of the computer graphics part of MPEG-4 called Animation Framework eXtension (AFX) [15] for dynamic mesh coding. This tool is called Interpolator Compression (AFX-IC). In [31], a spatial clustering algorithm for motion vectors was

introduced, predicting motion vectors within the object's bounding cube by tri-linear interpolation of the cube's corner vectors, which further reduces the amount of data to be encoded.

In this paper we present a new approach for the compression of dynamic 3D meshes that combines [17] and [31] into a unified and optimized framework. Here, we concentrate on the coding of mesh motion information of successive frames, while the first frame of a mesh sequence is coded, using static state-of-the-art coding approaches, e.g. 3DMC for wireframe models. The reason for this separation originates from the mesh sequence properties with constant connectivity: While the first mesh of a mesh sequence requires coding of all attributes, like vertices and connectivity, all following meshes only need to be coded by their motion information. Furthermore, by coding the first mesh separately, our differential coding approach can easily be applied to other time consistent description forms of dynamic 3D objects, e.g. point or point splat representations [29], where the first frame of a 3D sequence is coded differently, while the motion information of successive meshes remains the same. In our coding framework for 3D motion vectors, different coding modes are available for each single vertex motion vector, or for appropriate clusters of motion vectors. We further developed an efficient mode control based on rate-distortion (RD) optimization that always selects the best possible prediction mode among all possible modes. Our system uses standard MPEG-4 components and syntax (3DMC and AFX-IC) as far as possible and extends them as appropriate, and can thus be easily integrated into a future extension of MPEG-4.

The paper is organized into the sections Coding Structure and Coding Results. In Section 2, we describe the basic temporal DPCM-structure and RD-optimization. Furthermore, spatial subdivision and clustering methods are described, followed by arithmetic coding. In Section 3, we first discuss common error measures used for 3D mesh compression. Then we show how a certain rate point is selected by the RD-optimization process. Finally, in subsection 3.3 comparative coding results between our optimized coder and state-of-the-art dynamic mesh coding are presented.

2 Coding Structure

In dynamic mesh coding, often a sequence of time-consistent meshes is considered, where each single mesh of a sequence has the same number of vertices and all meshes share a common connectivity. An example of an animated mesh or mesh sequence is shown in Fig. 1.

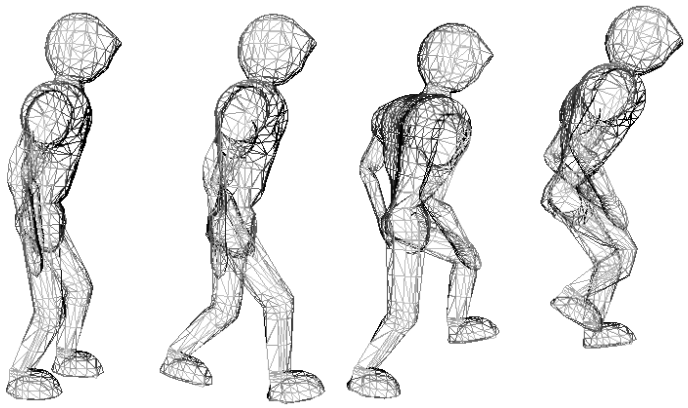

Fig. 1: Different meshes of the level 2 Humanoid sequence sharing a common connectivity.

Here, all meshes share a common connectivity and only the 3D vertices change position. The introduced coding scheme was developed to exploit spatial or intra-mesh as well as temporal or inter-mesh dependencies of such mesh sequences. For real-world 3D data, constant connectivity may only be guaranteed over a limited time period, since scene content or the topology of 3D objects may change. In such cases, mesh sequences are terminated and reinitialized with new 3D attributes, like vertices and connectivity. Similar to 2D video coding, such sequences are called “Group of Meshes” or GOMs, where the first mesh is called an intra or I-mesh, followed by predictive or P-meshes. The resulting hybrid coding structure is presented in detail in the following sections.

2.1 Basic DPCM Structure

Based on the motion data of mesh sequences, we have developed a Differential 3D Mesh Compression (D3DMC) that operates on the difference or motion vectors. Fig. 2 shows a block diagram of the encoder.

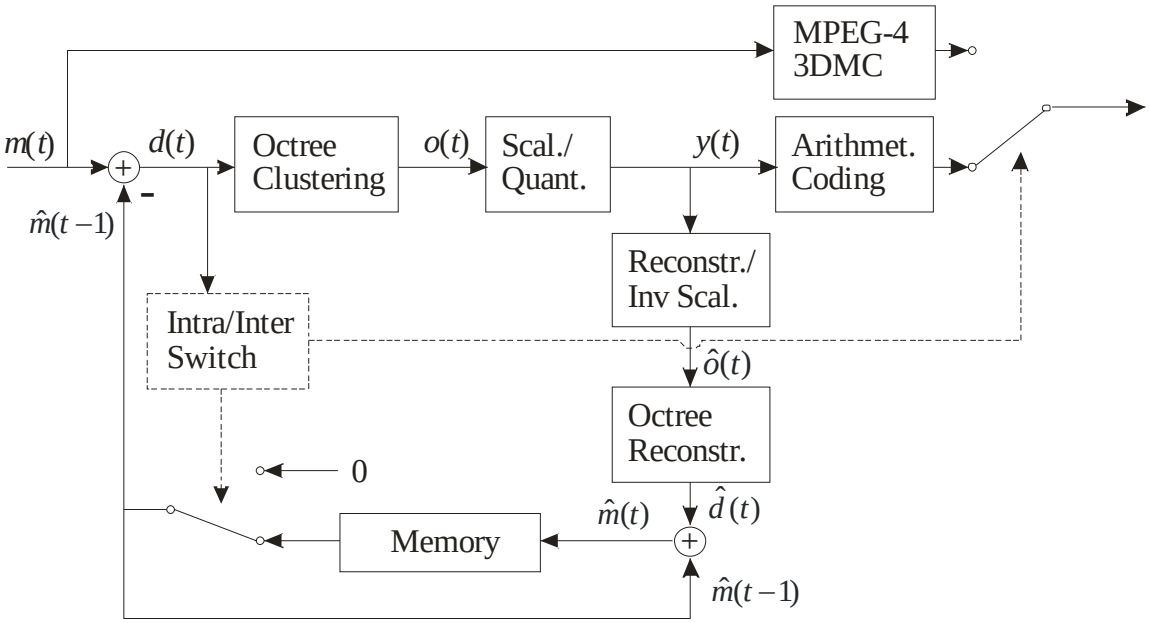

Fig. 2: Block diagram of D3DMC encoder.

The main structure is based on a DPCM-loop with 1st order predictor. The block diagram contains MPEG-4 3DMC as fallback mode that is enabled through the Intra/Inter switch that is fixed to either one for each 3D mesh of a sequence. This Intra mode is used for instance when the first mesh (I mesh) of a Group of Meshes (GOM) is encoded, i.e., when no prediction from previously decoded meshes is used. Additionally, I meshes can be used by the encoder in any other case, e.g., when the prediction error in D3DMC becomes too large. This mode provides backward compatibility to 3DMC and ensures that D3DMC can never be worse than 3DMC. The newly introduced predictive mode for mesh coding (P meshes) consists of the following steps:

1. The previously decoded mesh is subtracted from the current mesh to be encoded. This step can only be done if time-consistent meshes with a common connectivity are available. In case of a change of connectivity a fallback to 3DMC via the Intra/Inter switch is done and the recursive process is reinitiated. Only the difference signal between original and prediction, i.e. the difference vectors, is further processed. This backward prediction scheme also ensures the suppression of error drift that occurs in forward prediction structures [18].

2. Spatial clustering is applied to the difference vectors, in order to compute only very few representatives for a number of vectors. This algorithm uses an octree structure with tree pruning based on reconstruction error between original and reconstructed motion vectors with large cell sizes in spatial regions of homogeneous motion and small sizes for outliers. The result is a number of substitute vectors representing the motion vectors and octree structure information. In the RD-optimized version (see below), three different modes for spatial clustering are available and selected according to RD-calculation.
3. The substitute vectors are passed to an arithmetic coder using context-adaptive binary arithmetic coding (CABAC) [21] to efficiently adapt to the signal statistics.

2.1.1 Spatial Clustering of Difference Vectors

While a DPCM coding structure is used to exploit temporal dependencies between successive meshes, motion vector clustering is used to exploit spatial dependencies. The main concept of spatial clustering used here is to represent a number of similar motion vectors by only a few substitution vectors to achieve good compression results. This trilinear interpolation is illustrated in Fig. 3.

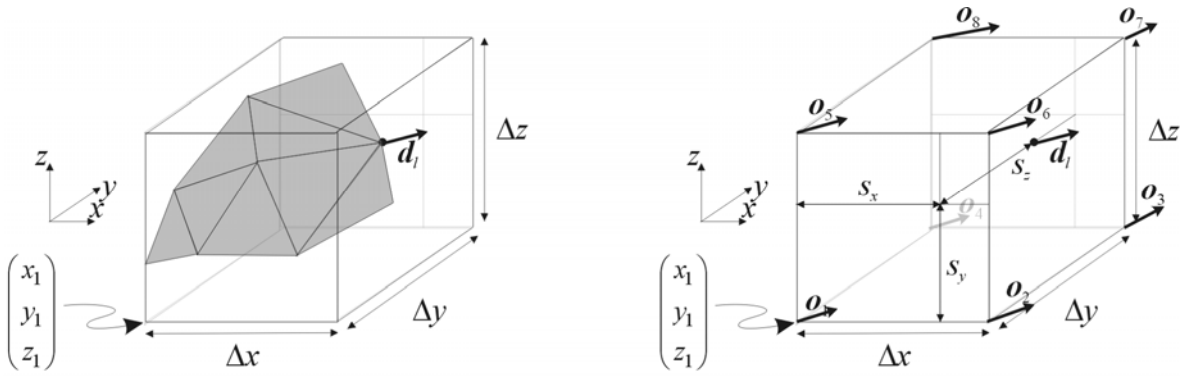

Fig. 3: Trilinear interpolation of 3D motion vectors.

The algorithm is also described in detail in [31]. Here, all motion vectors $\boldsymbol{d}_l$ of vertices within a volume are represented by the motion vectors $\boldsymbol{o}_1 \ldots \boldsymbol{o}_8$ of the eight corner vertices. These corner vectors are calculated (as described in section 2.1.2) from all motion vectors within a spatial cell, according to their vertex positions. In Fig. 3 right, all distances for the calculation are shown with respect to the first corner vertex with motion vector $\boldsymbol{o}_1$, considering the

distances s_x, s_y, s_z for the vertex with shown motion vector $\boldsymbol{d}_l$ relative to the vertex of $\boldsymbol{o}_1$ and the cell size Δx, Δy, Δz.

Of course this clustering introduces an error. A decoder that receives only the representative motion vectors cannot recover the exact motion vectors. We therefore use an error measure e_d to decide on further cell subdivision, which is the normalized sum of Euclidian distances between the original motion vectors $\boldsymbol{d}_l$ and their respective reconstructions $\tilde{\boldsymbol{d}}_l(t)$ from the corner motion vectors within the volume under consideration:

$$e_d = \frac{1}{L}\sum_{l=1}^{L}\left\|\tilde{\boldsymbol{d}}_l(t) - \boldsymbol{d}_l(t)\right\|. \tag{1}$$

2.1.2 Spatial Octree Subdivision

With this error measure the mesh under consideration can be clustered using an octree subdivision algorithm. Such algorithms are also applied in fine-scale 3D voxel reconstruction from 2D images [6], [23]. The octree subdivision scheme as an accelerated method for 3D reconstruction was introduced in [26]. This subdivision scheme allows a hierarchical volume partitioning that corresponds to the spatial statistics of 3D motion vectors. Here on one hand, large areas of homogenous motion of neighboring vertices exist that can be grouped into one large octree cell. On the other hand, small areas of diverse motion and single outliers exist, which require fine partitioning for grouping into individual cells.

In the basic structure the subdivision is performed top-down, starting with one single volume that contains the entire mesh at a certain time instant together with the motion vectors for each individual vertex. Although any volume, which contains the entire object, can be selected, mostly the object's bounding cube is taken. In the first subdivision step, the initial volume is split into eight octants, as shown in Fig. 4.

For each octant, a set of representative vectors as in Fig. 3 is computed from the contained motion vectors. At the beginning, the trilinear reconstruction of the motion vectors $\boldsymbol{d}_l(t)$ from the representative vectors $\boldsymbol{o}_m(t)$ with associated weights $w_{m,l}(t)$ is formulated first (see [31]):

$$\boldsymbol{d}_l(t) = \sum_{m=1}^{8} w_{m,l}(t)\boldsymbol{o}_m(t). \tag{2}$$

Here, a set of L equations exist for all motion vectors $\boldsymbol{d}_l(t)$ within a spatial cell, with $l = 1 \ldots L$. From this set, a matrix equation $\boldsymbol{d} = \boldsymbol{W} \cdot \boldsymbol{o}$ is created with weighting matrix $\boldsymbol{W}$, which contains the known positions of all motion vectors by means of their tri-linear weights. Thus, the representative vectors $\boldsymbol{o}_m(t)$ are calculated either from the pseudo inverse of $\boldsymbol{W}$, since $\boldsymbol{W}$ is a non-square matrix: $\boldsymbol{o} = \left[\boldsymbol{W}^T \boldsymbol{W}\right]^{-1} \boldsymbol{W}^T \cdot \boldsymbol{d}$, or by singular value decomposition and inversion of the decomposed matrices: $\boldsymbol{o} = \boldsymbol{V} \cdot \boldsymbol{diag}\left(\frac{1}{x_i}\right) \cdot \left(\boldsymbol{U}^T \boldsymbol{d}\right)$, where $\boldsymbol{W} = \boldsymbol{U} \cdot \boldsymbol{diag}(x_i) \cdot \boldsymbol{V}^T$.

Then the error measure is computed for the calculated representatives with equation (1). If the error is below a certain threshold, the representatives for the octant are determined and further encoded. If the error exceeds the threshold, the octant is further subdivided into 8 sub-octants, and the process is repeated recursively, as illustrated in Fig. 4. The complete process stops when the complete initial volume is processed and all representatives are determined.

A significant drawback of the basic structure is the need to specify a threshold to control the algorithm. It has to be determined experimentally, but does not allow direct control of resulting bit rate and quality. This is overcome in the RD-optimized extension as described in section 2.2.

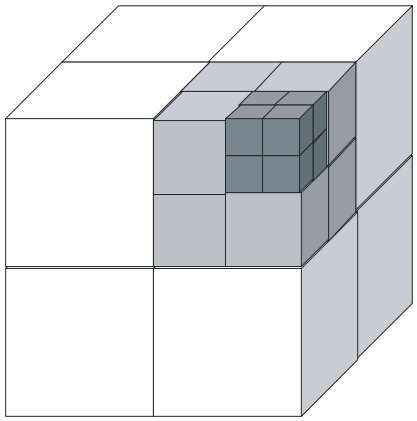

Fig. 4: Level 3 Selective Spatial Octree Subdivision.

2.1.3 Context-Adaptive Arithmetic Coding

The substitute vectors are passed to an arithmetic coder using context-adaptive binary arithmetic coding (CABAC) [21] to efficiently adapt to the signal statistics. Analysis of the probability density function (pdf) of the motion vector data has shown a Laplacian

distribution that is superimposed by small peaks at varying positions due to the clustering algorithm, as shown in the distribution function in Fig. 5.

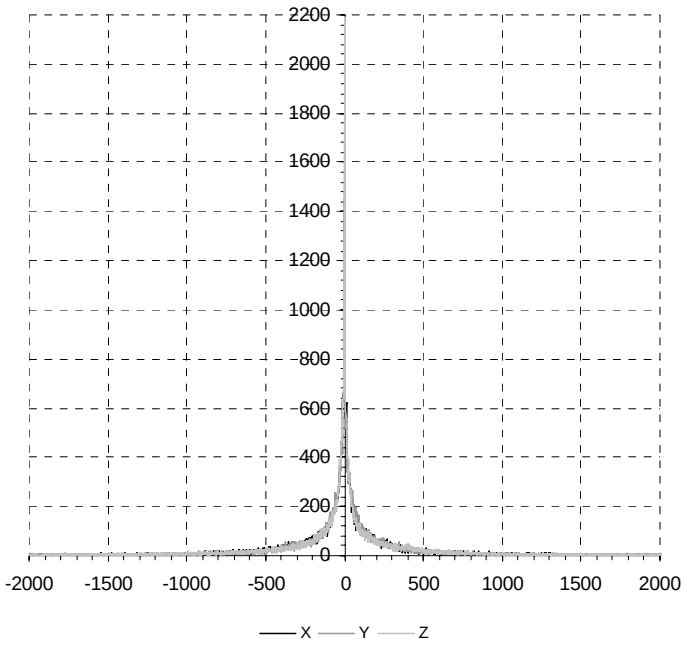

Fig. 5: Distribution of *x*-, *y*-, and *z*-coordinates of octree vectors.

For CABAC, the binarization uses unary/k^{th}-order Exp-Golomb codes to not only assign small code words to most frequent symbols but also to deal with certain outliers that frequently occur. Furthermore, the unary as well as the Exp-Golomb part sizes are calculated from the data to minimize overall code length. Thus the algorithm can adapt to changing statistics, since 3D meshes exhibit a variety of global and local motions that significantly modify the corner or substitute vector distribution An important feature of CABAC is its usage of multiple probability models to better fit the input signal statistics i.e. that for each element of the unary part of the unary/k^{th}-order Exp-Golomb codeword a different probability model is applied. These probability models adapt to the most frequent code words. Two additional probability models are used to encode the resulting octree structure. One probability model adapts to bits representing nodes and leaves of the octree the second probability model adapts to bits which give evidence if a node or leaf of the octree contains data to be encoded.

For the RD-optimization, CABAC becomes a part of the optimization process, where rate values need to be calculated for the mode and merge decision of each spatial cell. Due to the context adaptation, CABAC depends on the processing order of the octree pruning approach. Here, we follow the octree branch ordering, which usually starts at one corner of the 3D-model. This approach does not require additional subdivision structure information. To provide the minimum rate at least for the selected processing path, all previous quantized motion vectors and structure information are used for CABAC coding, including the data from all previous meshes. Although this approach might not deliver the best overall coding across all meshes, it still presents an optimum under the imposed constraint of keeping the data in temporal mesh order.

2.2 RD-optimized Coding Structure

The basic structure from 2.1 already performs very well in comparison to AFX-IC as shown in the results section 3. It is basically the work presented in [31] extended by efficient arithmetic coding (CABAC). However the octree pruning relies on a predefined error threshold. Further, we have found that additional prediction modes can improve the overall compression performance significantly, which are the following:

1. **Direct Coding** of differential vectors. This prediction mode is applied, if the motion vectors within the currently analyzed spatial volume are very different.
2. **Mean Replacement** of all motion vectors by their mean vector. This mode prediction is selected, if all motion vectors within a volume are very homogeneous.
3. **Trilinear Interpolation** of all differential motion vectors from the 8 corners. This prediction mode is selected, if the motion vectors exhibit a moderate smooth variation across the considered volume.

Mean Replacement is beneficial if vectors within an octant are homogeneous. In that case only one instead of 8 representatives needs to be encoded. Direct Coding is beneficial for very heterogeneous vectors. This mode is exclusively used in AFX-IC. Also Zhang and Owen use a direct mode in their recent work [30]. However, they switch between direct mode and trilinear interpolation on a global basis mesh by mesh. In our approach we allow individual modes for each individual cell. Trilinear Interpolation is used as in the basic

structure, however, in the RD-optimized version there is no need for a predefined threshold anymore. Instead we vary the Lagrangian parameter λ and obtain the minimum value of the cost function $D_A + \lambda R$ for all octree motion vector quantizations between 5 and 11 bits and compute rate R and distortion D_A for any possible subdivision (quantization below 5 and above 11 bits did not yield any minimum cost values and can therefore be omitted). Therefore, a minimum distortion is obtained at the associated bit rate and vice versa. By varying λ, associated optimal RD-pairs are obtained to create the final RD-curve across different bitrates and distortions.

Having different modes available implies the need for a mechanism to select the best one for each single subdivision. Here, the RD cost function is analyzed for each single volume, starting from the fully subdivided volume, where each spatial cell only contains 8 vectors. This minimum number is required, since trilinear interpolation uses 8 corner vectors, such that this mode becomes similar to direct coding at this subdivision level in terms of the number of motion vectors. Therefore, only Mean Replacement and Direct Coding need to be evaluated at this level. The objective for each spatial cell is to decide, whether the cell should remain at the current level or being merged with its neighboring cells. Therefore, the described motion vector quantization is carried out for the 8 separate cells, as well as the merged cell for all modes. The minimum $D_A + \lambda R$ cost value of all possibilities determines the merging decision as well as the optimal mode for a cell. The analysis continues up to the single volume, where the entire object's motion vectors are contained. Finally, the optimal octree pruning scheme with the appropriate spatial clustering modes for the remaining cells has been determined for a given reconstruction quality. In this search strategy basically all possible mode combinations for all subdivision are calculated, and the best is selected. Here, best possible performance is ensured; however, computational complexity is immense as well. Our future work will include strategies to speed-up this process by appropriately constraining the search.

The general structure of the RD-optimized coder still applies a DPCM structure, as shown in the basic structure in Fig. 2. The new elements introduced before have been integrated as shown in Fig. 6 top in the block "RD-Optimization". Here, the arithmetic coder was included in this block to be able to calculate the final rate of the coded bit stream for RD-optimization. A detailed graph of this block is shown in Fig. 6 bottom.

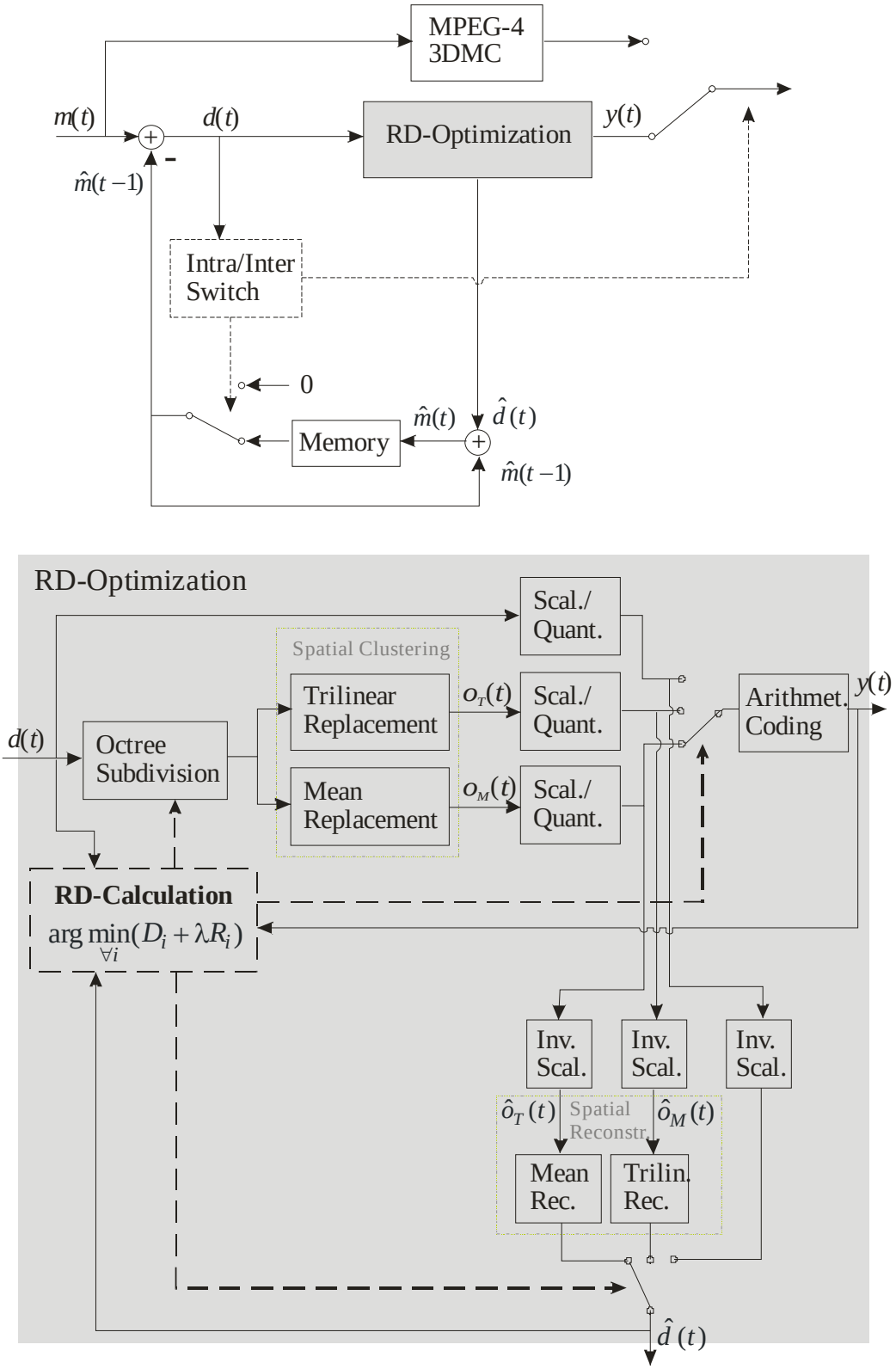

Fig. 6: Coding structure for the RD-optimized Dynamic 3D Mesh Coder (*top*) and detailed structure of RD-optimization (*bottom*).

Here, the analysis-and-reconstruction block from the fixed D3DMC, where only octree clustering together with trilinear interpolation is applied, is extended towards the RD optimization block. The RD optimization takes all differential vectors $\boldsymbol{d}(t)$ and outputs the arithmetically coded data $y(t)$ of a current mesh. For best compression results, context adaptive arithmetic binary coding (CABAC) [21] is used, which adapts to the mesh difference vector statistics over time. CABAC was also successfully applied to video compression and was standardized for H.264/MPEG4-AVC [16]. Furthermore, the RD-optimization also outputs the estimated differential vectors $\hat{\boldsymbol{d}}(t)$ for the following mesh at time t+1.

As already described, the RD-optimization contains 3 different prediction modes, which are controlled and selected by the central "RD-Calculation" block by varying the Lagrangian parameter λ and quantizing the motion vectors for each λ. Since the mode selection is based on the RD decision, it requires the current distortion and rate. For that, distortion is calculated from input and predicted vectors $\boldsymbol{d}(t)$ and $\hat{\boldsymbol{d}}(t)$ respectively, while the actual rate is taken from the arithmetically coded data $y(t)$. Therefore, the arithmetic coder needs to be included into the RD analysis loop. With these inputs, the RD calculation block calculates the encoder settings and controls the octree subdivision block for the appropriate octree pruning scheme as well as the mode selection via the signal flow switches.

3 Coding Results

For the coding efficiency evaluation of the developed RD-optimized D3DMC, comparative experiments with state-of-the-art compression technology were carried out. Furthermore, we also compared our RD-optimized version against the basic structure to evaluate, to which degree the introduced changes contribute to possible coding gains. For the evaluation, a special 3D error measure is used, which considers spatial displacement as Euclidian distance as well as different temporal distances between successive key meshes. In the case of evenly distributed mesh sequences, only the Euclidian distance would be sufficient, however for sequences that where reduced to a subset of key meshes prior to coding, the different temporal distances also need to be considered. Key mesh sequences are considered here,

since they where also used for the AFX-IC benchmark coding. The error measure and final coding results are introduced below.

3.1 3D Error Measures

One of the common error measures in 3D mesh comparison is the Hausdorff distance [10]. For two meshes A and B to be compared, the minimal distance $d_{A,i}$ between all points $\boldsymbol{v}_{A,i}(t)$ of mesh A towards mesh B is calculated first:

$$d_{A,i} = \min_{\forall k} \left\| \boldsymbol{v}_{A,i}(t) - \boldsymbol{v}_{B,k}(t) \right\|. \tag{3}$$

From this, the directional Hausdorff distance $d_{A\to B}$ is obtained as the maximum of all single distances

$$d_{A\to B} = \max_{\forall i} \left(d_{A,i} \right). \tag{4}$$

In the next step, the algorithm is applied vice versa to obtain $d_{B\to A}$. In general $d_{A\to B} \neq d_{B\to A}$, such that the general Hausdorff distance $d_{A,B}$ is taken as the maximum of both directional distances:

$$d_{A,B} = \max\left(d_{A\to B}, d_{B\to A} \right). \tag{5}$$

In the case of time-consistent mesh sequences, which we also consider in this paper, a one-to-one mapping between mesh A and B is possible. Therefore a direct Euclidian distance measure can be applied, which gives a far better displacement representation. For mesh comparison, we used the average distance d_m:

$$d_m = \frac{1}{N} \sum_{i=1}^{N} \left\| \boldsymbol{v}_{A,i}(t) - \boldsymbol{v}_{B,i}(t) \right\|. \tag{6}$$

In the case of the Hausdorff distance, only the maximum error of any two vertices is represented, such that all other displacements are neglected.

The average Euclidian distance d_m or average root mean squared error (AVGRMSE) is used for mesh-to-mesh comparison, as it represents a common measure for mesh evaluation, as in “Mesh” [3] and “Metro” [5]; two tools that automatically calculate distances between 3D meshes. “Metro” additionally provides a visual comparison. Often, mesh sequences are only represented by a subset of key meshes with varying temporal distances between them. A distortion of two successive key meshes also influences distortion of all intermediate meshes

that need to be interpolated after decoding. Thus also the temporal distance between the meshes needs to be included in an error measure. Here, the area distance was introduced in [17], which was also used in the experiments below. The main idea here is to extend the 1D Euclidian distance to a 2D area distance measure D_A by adding the temporal distance as 2nd dimension. This area distance is first created separately for *x*-, *y*-, and *z*-component between any successive pair of key meshes within a sequence. The calculation for the *x*-component is shown in (7). Here, the error $D_A(x)$ is the sum of all single area distances $D_{A,n}(x)$ between adjacent key meshes at times t_n and t_{n+1}:

$$D_A(x)=\sum_{n=1}^{N-1} D_{A,n}(x) \text{ with } D_{A,n}(x)=\begin{cases}\frac{|d_{n+1}(x)|+|d_n(x)|}{2}\left(t_{n+1}-t_n\right) \text{ if } \operatorname{sgn}\left(d_n(x)\right)=\operatorname{sgn}\left(d_{n+1}(x)\right)\\ \frac{|d_{n+1}(x)|^2+|d_n(x)|^2}{2\left(|d_{n+1}(x)|+|d_n(x)|\right)}\left(t_{n+1}-t_n\right) \text{ else}\end{cases} \quad (7)$$

The single area distances $D_{A,n}(x)$ are calculated from the trapezoidal areas, defined by the spatial Euclidian distances $d_n(x)$ and $d_{n+1}(x)$ of the adjacent key meshes and the temporal distance t_{n+1} - t_n, as shown in Fig. 7.

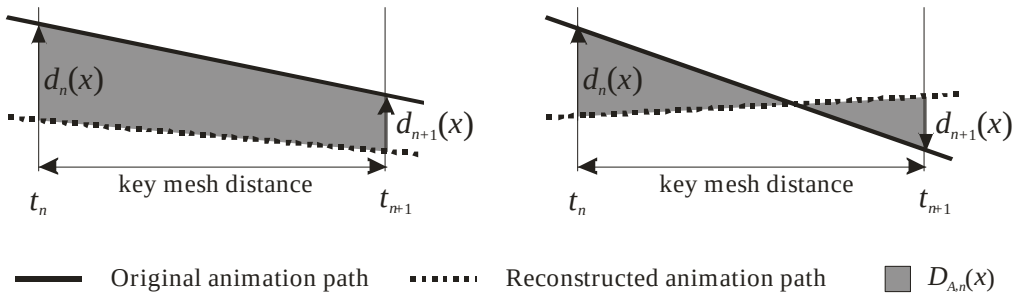

Fig. 7: Area Calculation for non-crossing and crossing original and reconstructed animation paths for *x*-component, leading to regular and twisted trapezoidal areas for $D_{A,n}(x)$ respectively.

Since the Euclidian distance is calculated from 1D components, it can also become negative and the signs of both distances $d_n(x)$ and $d_{n+1}(x)$ are used to specify, whether the area under investigation is a regular or twisted trapezoid and thus adapt the area calculation accordingly, as shown in equation (7) and the arrow directions for the signed distances $d_n(x)$ and $d_{n+1}(x)$ in Fig. 7. For the *y*-, and *z*-component, similar calculations are carried out. Finally, a normalized average distance $\overline{D}_A$ is calculated from the 3 separate area differences:

$$\overline{D}_A=\frac{D_A(x)+D_A(y)+D_A(z)}{3\left(t_{N-1}-t_0\right)\cdot d_{m,\max}}. \quad (8)$$

For normalization purposes, $\overline{D}_A$ is divided by the total temporal distance ($t_{N-1} - t_0$), as well as the maximum spatial distance in *x*-, *y*- and *z*-direction $d_{m,\max}$. The normalized average area

distance was also used in MPEG core experiments for 3D graphics compression technology and is described in [17] in more detail.

3.2 RD Point Cloud Creation

As already described, the coder performs RD optimization for all possible spatial subdivisions, starting from the finest spatial subdivision up to the entire bounding cube. This way the combination of subdivisions together with the appropriate prediction mode is determined. One example of an obtained RD point cloud is shown in Fig. 8. For visualization purposes, the distortion axis was logarithmically scaled to better show point resolution at low distortion errors, where one of the points is selected as the minimum RD costs at that data rate. Here, the points are vertically clustered around the quantization levels between 5 and 11 bits for the motion values. Due to this quantization, each cluster has its lower distortion bound, e.g. a minimum distortion of 0.004 for the 7-bit-cluster.

The final value is taken from that entire point cloud as the point with minimum rate-distortion value $D_A+\lambda R$. In the example in Fig. 8, this is the single leftmost point with $D_A = 0.0005$ from the 10-bit-cluster. Note, that the linear minimum RD function for the point selection in Fig. 8 is also drawn logarithmically due to the logarithmic distortion axis.

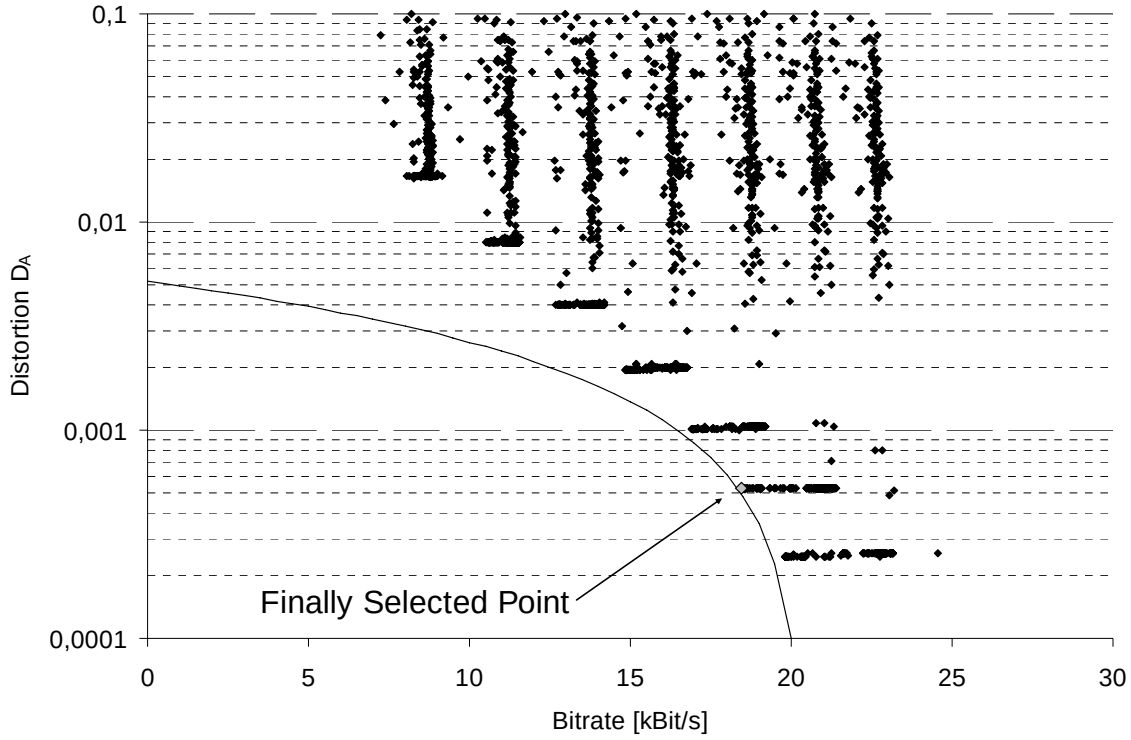

Fig. 8: RD point cloud (logarithmic scale) and finally selected rate point at a given maximum distortion (0.0006) and minimum bit rate.

3.3 Comparative Coding Results

In the coding experiments, the RD-optimized D3DMC is compared against AFX-IC as well as against a fixed version of D3DMC, developed earlier. The latter comparison is carried out to find out, how the RD-optimization components alone contribute to the coding gain. Also the comparison against AFX-IC has to be investigated at a number of different rate points. As already described in Section 3.2, spatial subdivision structure, as well as prediction mode selection vary according to the given rate point for the RD-optimized D3DMC. An example for prediction mode selection is given in Table 1, where the number of spatial cells for three different area distances or distortions is shown, together with the percentage of mode selection.

Distortion D_A	Number of Spatial Cells	Direct Coding	Mean Replacement	Trilinear Interpolation

$2.0x10^{-3}$	4027	41.79%	24.44%	33.77%
$8.6x10^{-4}$	14715	73.53%	6.46%	20.01%
$5.5x10^{-4}$	58314	92.15%	0.87%	6.98%

Table 1: Number of spatial cells for three different resolutions and percentage of prediction mode selection, L3Humanoid_L3.

First, the number of spatial cells increases with lower area distortion. Concurrently, the percentage of mode selection shifts from relatively equal distribution towards a direct mode selection for 92% of the spatial cells for lower distortion and higher reconstruction quality, as already expected in the statistical mesh analysis.

Although the developed coder always selects the optimal prediction structure and thus combines the best technologies of AFX-IC and static D3DMC, it also requires slightly more bit rate, due to the signaling of spatial subdivision and prediction mode selection. Thus, the experiments also have to show, whether the advantage of RD-optimization can compensate this disadvantage. For AFX-IC, no spatial structure information needs to be transmitted, since this coder directly codes all differential motion vectors, i.e. operates completely in Direct mode. The fixed version of D3DMC requires spatial structure signaling, but no mode selection, since here only Trilinear Interpolation is carried out, similar to the approach in [31].

For the experiments, the “humanoid” test set with different resolutions was selected, which is an animated sequence of 399 frames at resolutions of 498, 1940 and 7646 vertices per mesh. The sequence is represented by 46 key meshes, which are also available for mesh coding and decoding. All other meshes are linearly interpolated during rendering from the **CoordinateInterpolator**-syntax from VRML [13], in which the animation is provided.

The first graph in Fig. 9 shows the results for the coarsest resolution with 498 vertices.

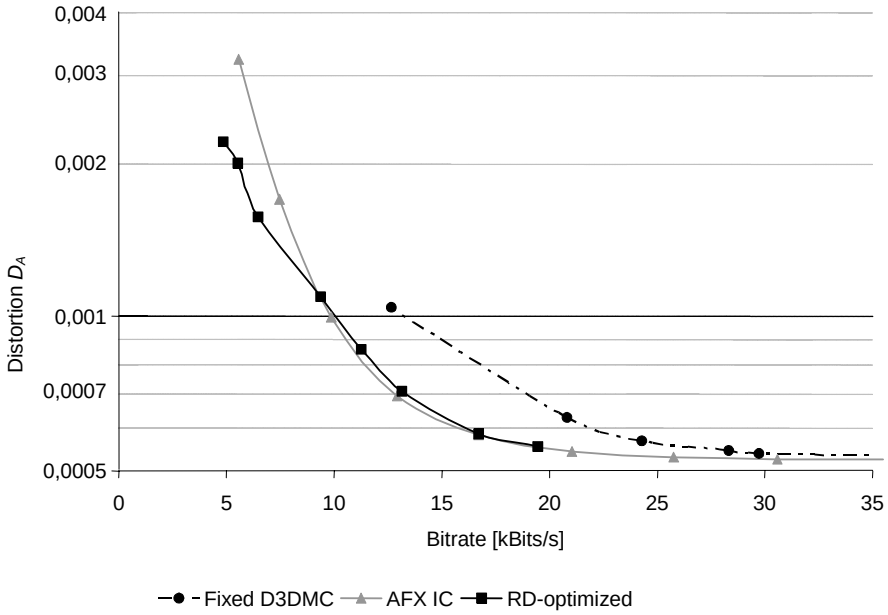

Fig. 9: Distortion D_A over bit rate for Fixed and RD-optimized D3DMC and AFX IC, L1Humanoid_L3, 46 key meshes, 498 vertices.

Here, the fixed D3DMC performed worse than the standard AFX IC, since a relatively large percentage (~34%) of the data rate is used for coding the spatial clustering structure. In comparison to that, the improved RD-optimized D3DMC performs similar to AFX IC at bitrates above 10 kbit/s and even better below. The improvement of the coder in comparison to the fixed version for low-resolution meshes mainly comes from the choice between different clustering modes, where a larger partition of the mesh is directly differentially coded, such that the tree structure for subdivision description is reduced. Since the RD-optimized coder always performs equal or better than any of the other two, the additionally required spatial structure and mode selection information could be compensated.

As an example, an analysis of the incremental coding gain for two intermediate coder implementation stages from Fixed D3DMC to the RD-optimized version is shown in Fig. 10. The first stage adds the RD-optimization, while the second stage adds Direct Coding as second prediction mode on top of it. Finally, Mean Replacement as third prediction mode is added, yielding the Full RD-optimized version.

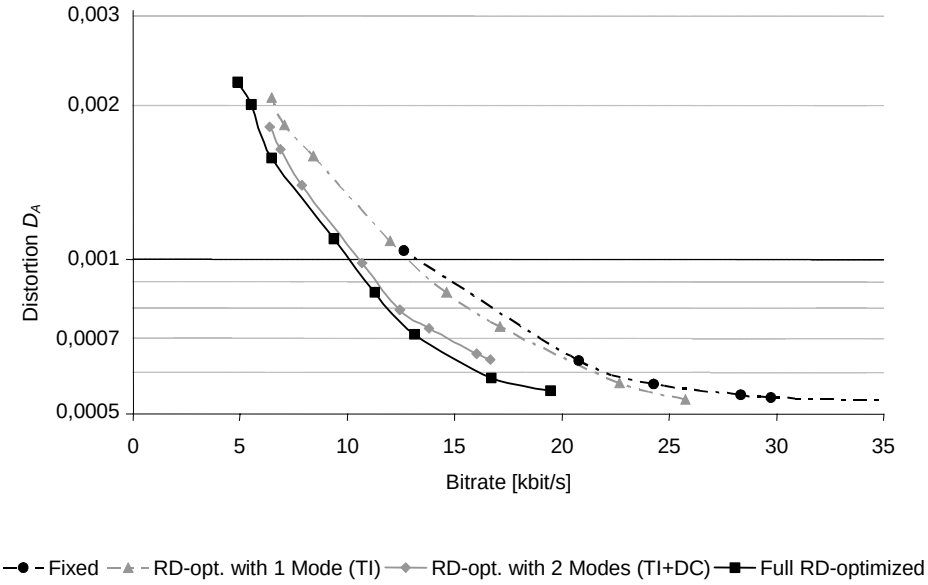

Fig. 10: Distortion D_A over bit rate for Fixed and RD-optimized D3DMC and intermediate steps: RD-optimized with Trilinear Interpolation mode (TI) and RD-optimized with Trilinear Interpolation and Direct Coding modes, L1Humanoid_L3, 46 key meshes, 498 vertices.

Starting from the right curve for Fixed D3DMC in Fig. 10, the RD-optimization adds about 25% of the total coding gain. Both coder versions only use Trilinear Interpolation as prediction mode. The largest percentage of 60% of the total coding gain is achieved, if Direct Coding as second prediction mode is included. Including Mean Replacement as the third prediction mode leads to the final RD-optimized version of D3DMC and adds the last 15% of the coding gain.

In the second example at medium mesh resolution of 1940 vertices, the RD-optimized coder performs again better than the fixed version. Improvements are also obtained at higher data rates and lower distortion, where the fixed coder performed slightly worse above 55kBits/s than the standard AFX IC. Here, the RD-optimized D3DMC mainly codes the differential mesh motion vectors directly, thus omitting again a rather large and detailed subdivision structure. Both methods perform better at lower data rates, since here both D3DMC versions really benefit from the spatial clustering in combination with trilinear interpolation or mean replacement, where large spatial octree cells of differential motion vectors are coded by 8, respectively 1 substitution vector(s).

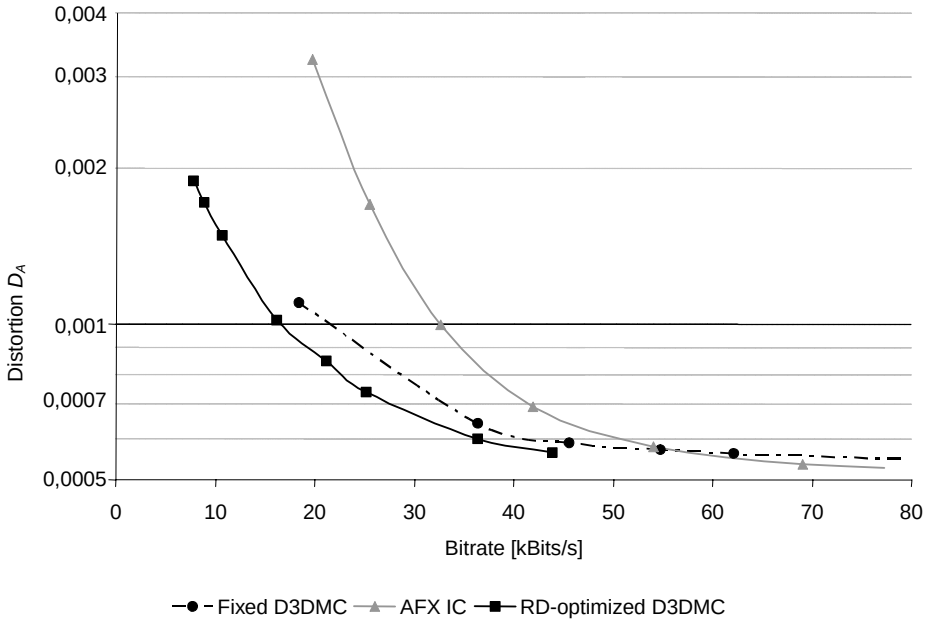

Fig. 11: Distortion D_A over bit rate for Fixed and RD-optimized D3DMC and AFX IC, L2Humanoid_L3, 46 key meshes, 1940 vertices.

In the third case for highest mesh resolution of 7646, the coding gain of both D3DMC versions against AFX IC becomes even larger, as even more motion vectors with similar statistical properties can be clustered together and coded by very few substitution vectors. Here, the fixed D3DMC, which always uses trilinear interpolation as spatial clustering, performs worse than the standard AFX IC for higher data rates. The RD-optimized D3DMC uses mainly direct coding, similar to AFX IC, as shown in the mode selection analysis above. Therefore, the fixed coder requires a very detailed spatial clustering and therefore a higher number of substitution vectors from the trilinear interpolation to be coded. Additionally, this detailed subdivision structure information needs to be coded, which contributes considerably to the bit stream. On the other hand, this structure information is reduced considerably for the RD-optimized D3DMC, due to the direct coding of large spatial areas.

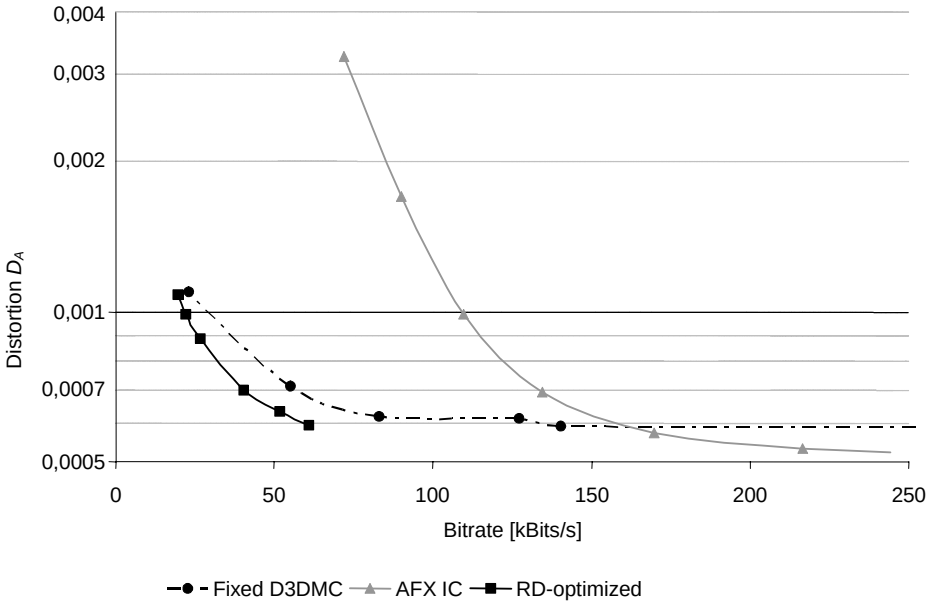

Fig. 12: Distortion D_A over bit rate for Fixed and RD-optimized D3DMC and AFX IC, L3Humanoid_L3, 46 key meshes, 7646 vertices.

Beside the RD-curves in the figures above, a visual example is shown in Fig. 13 (a) and (b) for two different resolutions at lower data rates to visualize the effects of clustering in these data ranges.

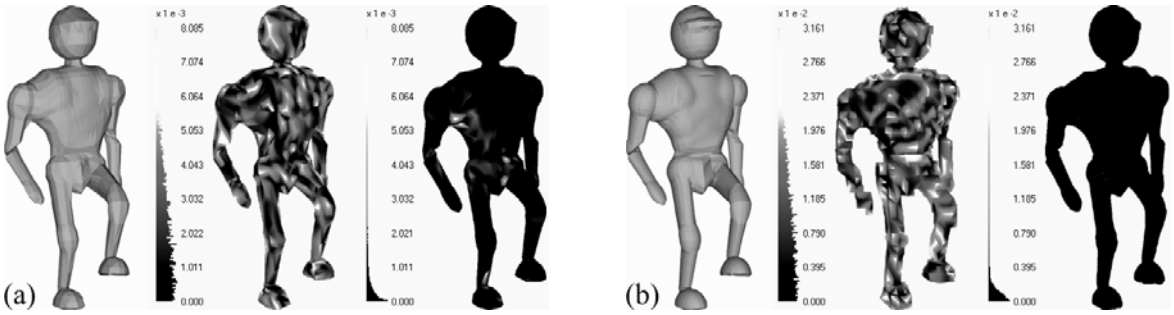

Fig. 13: Visual mesh reconstruction for 2 different resolutions, using the "Mesh"-Tool [3]: Original Mesh and reconstruction results for AFX-IC and D3DMC.
(a): 1940 vertices, AFX IC: 60.1 kBit/s, D3DMC: 62.7 kBit/s,
(b): 7646 vertices, AFX IC: 128.1 kBit/s, D3DMC: 86.4 kBit/s

On the left of each figure, the original mesh geometry is shown, followed by the reconstruction results for AFX IC and D3DMC. Here, the "Mesh"-Tool from [3] was used, and the difference images show the root mean squared error or spatial component of the area distortion measure to visualize the reconstruction error at a certain time instance. Both error images have been adapted to the same error scale to better highlight the differences between

the reconstruction methods. The lighter the color in the difference images, the larger is the reconstruction error. The scales are shown to the left of each error image and additionally include the error histograms. For the AFX IC reconstructed images, the mesh surface is rather distorted due to coarse quantization and the error histograms show a large error distribution. In contrast, D3DMC only has small reconstruction errors, and also a small error distribution at very small values. Here, the spatial clustering of D3DMC clearly outperforms a plain direct coding approach as used in AFX IC. Therefore, the difference between the two approaches becomes even larger for the higher resolution mesh sequence in Fig. 13 (b), where the data rate for D3DMC is only $^2/_3$ of AFX IC, compared to equal bit rates in Fig. 13 (a).

Overall, the proposed optimized D3DMC approach outperforms fixed D3DMC as well as AFX IC. The main improvements against the fixed version are in the area of high-quality coding at higher data rates, where the fixed version had to allocate a rather large portion of the total bit rate for the subdivision structure. The overall better performance of the RD-optimized D3DMC comes from the combination and optimization of coding technology from both, fixed D3DMC as well as from AFX-IC. While fixed D3DMC only uses spatial clustering with trilinear interpolation, AFX IC is purely based on arithmetic coding of differential vectors. These modes are all included in the RD-optimized D3DMC, such that the improved coding results are not surprising.

4 Conclusions

In this paper we have presented a coding structure for dynamic mesh compression that utilizes spatial as well as temporal statistical dependencies of the mesh motion or differential vectors. Based on a statistical analysis of the spatial and temporal differential vector distribution, we introduced a basic DPCM coding structure similar to state-of-the-art hybrid 2D video coders. For exploiting the spatial dependencies, we introduced a spatial octree subdivision scheme together with different prediction modes: clustering by trilinear interpolation and mean replacement, where a number of similar adjacent motion vectors are represented by 8 respectively 1 substitution vector(s). Additionally, a direct coding mode is applied in cases of dissimilar vectors. The mode selection as well as subdivision structure is controlled by an RD-optimization to ensure that the best RD coding decision is taken for a

broad range of bit rates, respectively reconstruction qualities. Here, the prediction mode with minimal costs is selected, including all possible spatial subdivisions of the mesh sequence, starting from the smallest cell size and appropriate prediction mode up to the global bounding cube with the entire set of differential vectors.

The obtained results show that the RD-optimized D3DMC coder always performs superior in comparison to the fixed version, where spatial subdivision is controlled by a global error value, and AFX IC as the current standard for dynamic mesh compression. AFX IC only applies arithmetic coding to the differential vectors, which is equivalent to direct mode selection for the entire set of vectors. Although fixed D3DMC already performs better than AFX IC on average, there are still losses for very high data rates as well as for very low mesh resolutions due to the additional partitioning information that needs to be transmitted. These problems are solved by using different prediction modes and thus combining the advantages of both coders into one single design.

Future work will include further performance optimization through adding or replacing prediction modes. Moreover, since the mode selection results as found in our experiments indicate a certain pattern for high/low resolution meshes as well as high/low reconstruction quality, a preprocessing analysis about the mesh sequence characteristics might indicate, which restrictions could be imposed onto the RD-optimized D3DMC coder, in order to speed up processing. Additional work is also required for the first mesh or I mesh of a group of meshes, which still uses 3DMC. Here, further improvements in Intra mesh compression could lead to better overall coding results.

5 Acknowledgments

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

We would like to thank Alexandru Salomie from Vrije Universiteit Brussel for providing the Humanoid sequence.

References

[1] M. Alexa and W. Müller, “Representing Animations by Principal Components”, *Proc. Of EUROGRAPHICS* 2000, vol. 19, no. 3, pp. 411-418, 2000.

[2] P. Alliez, “Recent Advances in Compression of 3D Meshes”, *Proc. EUSIPCO 2005, 13th European Signal Processing Conference*, Antalya, Turkey, Sept. 2005.

[3] N. Aspert, D. Santa-Cruz, T. Ebrahimi, “MESH: Measuring errors between surfaces using the Hausdorff distance”, *Proceedings of the IEEE International Conference on Multimedia and Expo*, vol. I, pp. 705-708, 2002.

[4] H. Briceño, P. Sander, L. McMillan, S. Gortler, H. Hoppe, “Geometry videos: A new representation for 3D animations”, *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 136-146, 2003.

[5] P. Cignoni, C. Rocchini and R. Scopigno, “Metro: Measuring Error on Simplified Surfaces”, *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.

[6] P. Eisert, E. Steinbach, B. Girod, “Multi-hypothesis, Volumetric Reconstruction of 3-D Objects from Multiple Calibrated Camera Views”, *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 99*, pp. 3509-3512, 1999.

[7] X. Gu, S. Gortler, H. Hoppe, “Geometry images”, *Proc. 29th annual conference on Computer graphics and interactive techniques, ACM SIGGRAPH 2002*, pp. 355-361, 2002.

[8] S. Gumhold, “New Bounds on the Encoding of Planar Triangulations”, *Technical Report WSI-2000-1*, University of Tübingen, 2000.

[9] S. Gumbold, W. Strasser, “Real Time Compression of Triangle Mesh Connectivity”, *Computer Graphics Proceedings, Annual Conference Series, (Proc. of ACM SIGGRAPH 98)*, pp. 133-140, 1998.

[10] D. Huttenlocher, G. Klanderman and W. Rucklidge, “Comparing Images Using the Hausdorff Distance”, *IEEE Journal of Pattern Analysis and Machine Intelligence*, vol. 15, no. 9, pp. 850-863, 1993.

[11] L. Ibarria and J. Rossignac, “Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity”, *In: Proceedings of the 2003 ACM*

SIGGRAPH/ Eurographics Symposium on Computer Animation, pp. 126–135. Eurographics Association, 2003.

[12] M. Isenburg and J. Snoeyink, "Spirale Reversi: Reverse Decoding of the Edgebreaker Encoding", *In: Proceedings of 12th Canadian Conference on Computational Geometry*, pp. 247-256, 2000.

[13] ISO/IEC DIS 14772-1, "The Virtual Reality Modeling Language", April 1997.

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

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

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

[17] E. S. Jang, J. D. K. Kim, S. Y. Jung, M. J. Han, S. O. Woo and S. J. Lee, „Interpolator Data Compression fro MPEG-4 Animation", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 7, pp. 989-1008, 2004.

[18] N. S. Jayant, P. Noll, "Digital Coding of Waveforms" *Prentice-Hall Int.*, London, 1984.

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

[20] J. Lengyel, "Compression of Time Dependent Geometry", Symposium on Interactive 3D Graphics, pp. 89-95, 1999.

[21] D. Marpe, H. Schwarz and T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 620-636, 2003.

[22] J. Rossignac, "Edgebreaker: Connectivity Compression for Triangle Meshes", *IEEE Trans. on Visualization and Computer Graphics*, vol. 5, no. 1, pp. 47-61, 1999.

[23] S. M. Seitz and C. R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring", *Proc. Computer Vision and Pattern Recognition*, pp. 1067-1073, 1997.

[24] A. Shamir, V. Pascucci, "Temporal and Spatial Level of Details for Dynamic Meshes", *Proc. Of Virtual Reality Systems and Techniques*, pp. 423-430, 2001.

[25] A. Smolic, K. Müller, P. Merkle, M. Kautzner, and T. Wiegand: “3D Video Objects for Interactive Applications”, *Proc. EUSIPCO 2005, 13th European Signal Processing Conference*, Antalya, Turkey, Sept. 2005.

[26] R. Szeliski, “Rapid Octree Construction from Image Sequences” *CVGIP: Image Understanding*, vol. 58, No. 1, July, pp. 23-32, 1993.

[27] A. Szymczak, D. King, and J. Rossignac, “An Edgebreaker-Based Efficient Compression Scheme for Regular Meshes”, *Computational Geometry*, vol. 20 no. 1-2, pp. 53-68, 2001.

[28] C. Touma and C. Gotsman, “Triangle Mesh Compression”, *In: Proc. Graphics Interface 98*, pp. 26-34, 1998.

[29] S. Würmlin, E. Lamboray, 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.

[30] J. Zhang and C. B. Owen, “Hybrid Coding for Animated Polygonal Meshes: Combining Delta and Octree”, *International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume I*, pp. 68-73, 2005.

[31] J. Zhang and C. B. Owen, “Octree-based Animated Geometry Compression”, *DCC’04, Data Compression Conference*, Snowbird, Utah, USA, pp. 508-517., 2004.