Predictive-DCT Coding for 3D Mesh Sequences Compression

Rachida Amjoun, Wolfgang Straßer

Graphical-Interactive Systems
Wilhelm Schickard Institute for Computer Science
University of Tuebingen
Sand 14, 72076 Tuebingen, Germany
email: {amjoun,strasser}@gris.uni-tuebingen.de
www: http://www.gris.uni-tuebingen.de

Abstract

This paper proposes a new compression algorithm for dynamic 3d meshes. In such a sequence of meshes, neighboring vertices have a strong tendency to behave similarly and the degree of dependencies between their locations in two successive frames is very large which can be efficiently exploited using a combination of *Predictive* and *DCT* coders (PDCT). Our strategy gathers mesh vertices of similar motions into clusters, establish a local coordinate frame (LCF) for each cluster and encodes frame by frame and each cluster separately. The vertices of each cluster have small variation over a time relative to the LCF. Therefore, the location of each new vertex is well predicted from its location in the previous frame relative to the LCF of its cluster. The difference between the original and the predicted local coordinates are then transformed into frequency domain using DCT. The resulting DCT coefficients are quantized and compressed with entropy coding. The original sequence of meshes can be reconstructed from only a few non-zero DCT coefficients without significant loss in visual quality. Experimental results show that our strategy outperforms or comes close to other coders.

First presented at the 2nd International Conference on Computer Graphics Theory and Applications 2007, extended and revised for JVRB

Keywords: Animation, animated mesh compression, clustering, local coordinate frame, predictive coding, DCT

1 Introduction

Animated objects are frequently used in e-commerce, education and movies and are the core of video games. The animation in these applications can be either generated using motion capturing systems or simulated by sophisticated software tools like Maya and Max 3D.

The most common representation of animated three dimensional objects is the triangle mesh which consists of the geometric information describing vertex positions and connectivity information describing how these vertices are connected. 3D animation consists then of a sequence of consecutive triangle meshes.

As animation becomes more realistic and more complex, the corresponding frame meshes become bigger and bigger, consuming more and more space. It is therefore indispensable to compress the animation datasets. Key-frame animation is one of the most famous and dominant animation representations used in the industry to represent the animation compactly. A set of key frames are chosen to describe certain important key poses in the animation sequence at certain times. Then all frames in between are generated using interpolation techniques. For such applications, even the number of key-frames can be very large, requiring a large memory space and need for effective compression techniques.

The current coders are dedicated to compress the triangular meshes of fixed connectivity so that the connectivity needs to be encoded, stored or transmitted once, then the geometry coding comes into play.

There are several criteria by which developed coding techniques can be distinguished. One of these criteria is if the approach considers the entire sequence where the coherency is globally exploited by using the principal component analysis (PCA) transform or frame by frame where the coherency is locally exploited by using for example predictive coding.

In PCA based coding, the global linear behavior of the vertices through all frames is approximated in terms of linear space. The animation sequence can be reduced to a few principal components and coefficients. The efficiency of this technique increases when the datasets are segmented or clustered, so that each group is individually encoded by PCA. This type of method supports progressive transmission. The drawback of this approach is it is computationally expensive.

In predictive methods, for each frame, the difference between the predicted and the current locations is encoded with very few bits. These approaches are simple, not expensive, lossless and well suited for real-time applications. The drawback of these methods is that they don't support progressive transmission.

Affine transformations well approximate the behavior of sets of vertices relative to the initial position (the first frame, eventually the I-frame). This type of method is very effective for animations based on motion capturing, if the mesh is well partitioned into almost rigid parts, since the vertices are attached to the bones and move according to their representative joints.

Therefore, exploiting the coherence in this animation and finding the transformation that best matches each group of vertices is easier than finding a transformation that approximates each part in deformed meshes (like a cow animation). The drawback of this technique is that it can be computationally expensive depending on the splitting process or the affine transformation optimization.

In this paper, we propose a new compression algorithm based on predictive and DCT transform in the local coordinate systems.

The method is inspired from video coding. We first split the animated mesh into several clusters (similar to macroblocks in video coding) using a simple and efficient clustering process [AS07]. Then, we perform a prediction in the local coordinate systems. Finally, we transform the resulting delta vectors (between the predicted and the original vertex locations) of each cluster in each frame into the frequency domain using Discrete Cosine Transform.

2 Related Work

During the last decade, extensive research has been done on static mesh compression, producing a large number of schemes (see, e.g., [Ros04] or [AG05] for comprehensive surveys of the developed techniques). While research still focuses on efficient compression for huge static meshes [IG03] animated meshes have become more and more important and useful every where. However, the current techniques for the compression of sequences of meshes independently are inefficient.

Lengyel [Len99] suggested the decomposing of the mesh into submeshes whose motions are described by rigid body transformations. The compression was achieved by encoding the base submeshes, the parameters of the rigid body transformations, and the differences between the original and the estimated locations. Zhang et al. [ZO04] used an octree to spatially cluster the vertices and to represent their motion from the previous frame to the current frame with a very few number of motion vectors. The algorithm predicts the motion of the vertices enclosed in each cell by tri-linear interpolation in the form of weighted sum of eight motion vectors associated with the cell corners. The octree approach is later used by K. Mueller et al. [MSK$^+$05] to cluster the difference vectors between the predicted and the original positions. Very recently, Mamou et al. [MZP06] proposed skinning based representation. In their algorithm, the mesh is also partitioned, then each submesh in each frame is associated an affine motion and each vertex is estimated as a weighted linear combination of the clusters motions. Finally, the prediction errors are compressed using a temporal DCT coding.

In prediction techniques, assuming that the connectivity of the meshes doesn't change, the neighborhood in the current and previous frame(s) of the compressed vertex is exploited to predict its location or its displacement [YKL02, IR03, SO06]. The residuals are compressed up to a user-defined error. Ibarria and Rossignac [IR03] introduced two predictors to exploit the inter-frame coherence. They propose that one traverse the mesh triangles in an order [Ros01]

suitable for the various predictors. The first proposed space-time predictor is the *Extended Lorenzo Predictor* (ELP), a prefect predictor for a subset of the mesh undergoing pure translation from the previous frame. It uses a parallelogram prediction to exploit spatial coherence, and then performs temporal prediction on the spatial details. The second predictor is the *Replica Predictor* which replicates perfectly the local geometry undergoing any combinations of translations, rotations, and uniform scaling. The Replica predictor expresses the location of the vertex relative to the the locations of three vertices of an adjacent triangle as local coordinate system. The vertex location in the new frame is estimated by its relative coordinates from the previous frame. Similar predictor is introduced by Stefanoski and Ostermann [SO06]. It is a perfect predictor that preserves the angle between the reference triangle and the new triangle.

In PCA based approaches, Alexa et al. [AM00] used PCA to achieve a compact representation of animation sequences. Later, this method is improved by Karni and Gotsman [KG04], by applying second-order Linear Prediction Coding (LPC) to the PCA coefficients such that the large temporal coherence present in the sequence is further exploited. Sattler et al. [SSK05] proposed a compression scheme that is based on clustered PCA. The mesh is segmented into meaningful clusters which are then compressed independently using a few PCA components only. Amjoun et al. [ASS06, AS07] suggest the use of local coordinates rather the world coordinates in the local PCA based compression. They showed that the local coordinate systems are more compressable with PCA than the world coordinates.

Guskov et al. [GK04] used wavelets for a multi-resolution analysis and exploited the parametric coherence in animated sequences. The wavelet detail coefficients are progressively encoded. Payan et al. [PA05] introduced the lifting scheme to exploit the temporal coherence. The wavelet coefficients are thereby optimally quantized. Briceno et al [BSM+03] transform the mesh sequences into geometry images which are then compressed using standard video compression.

3 Overview

The local coordinate system has an important property that can be very useful for compression of animation. It exhibits a large clustering over time and the locations of the vertex tend to form a cluster around one position (over all frames). Regardless what kind of deformation the vertices undergo, i.e. rotation, or translation or scaling or combination of all three relations, the vertices will generally keep their positions, at least between two successive frames.

Our technique uses this property to guide the clustering process and to perform a predictive coder.

Basically, our algorithm consist of four steps:

1. **Clustering process:** The vertices are clustered into a given number of clusters depending on their motion in the LCFs. Indeed, the vertex should belong to the cluster where its deviation in the LCF through all frames is very small compared to the other LCFs. Thereby, the efficiency of the prediction through a time increases. Moreover, the clustering will preserve the global shape when DCT coding is performed (spatially) in each cluster.

2. **Lossless coding of LCFs:** The locations of the vertices that contribute to the construction of the LCF of each cluster should be losslessly encoded. In order to ensure that the decoder could use the same LCF, we decode and reconstruct the LCF to be used during the compression of the remaining vertices.

3. **Predictive coding:** This step allows the reduction of space-time redundancy. It is performed on the local coordinates rather than the world coordinates, which makes the coding more efficient. It allows the prediction errors to tend to be very small. The powerfulness of the predictor strongly relies on the clustering process. If the vertex is associated with a LCF whose motion is not similar to its motion then the local coordinates of the vertex will have a large variation over all frames and the prediction will produce large delta vectors.

4. **Transform-based coding or DCT:** For further compression, the coordinates of delta vectors are represented as 1D signals then transformed into frequency domain using DCT, producing uncorrelated coefficients. These coefficients are more compressable with the entropy coding than delta vectors. Moreover, many coefficients of low values can be zeroed without significant loss in visual quality.

To avoid error accumulation that may occur, we simulate the decoding process during encoding to

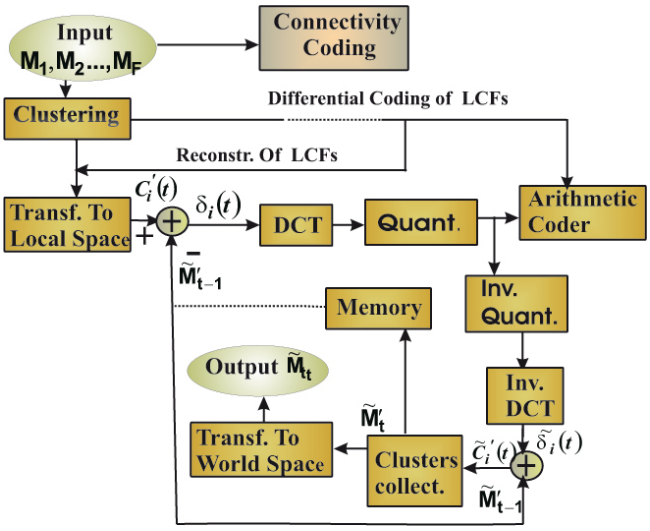

Figure 1: Overview of the compression pipeline

make sure that during the encoding, we use exactly the same information available to decoding algorithm. After the compression of each frame we should substitute the original vertex locations by the decoded locations.

4 Compression Pipeline

Given a sequence of triangle meshes $M_f, f = 1, ..., F$ with V vertices and F frames (meshes), we encode the first frame separately from the rest of the frames in the sequence using static mesh compression [GA03].

An overview of the whole compression pipeline is illustrated in Figure 1.

4.1 Local Coordinate Frames

4.1.1 Seed Triangles Selection

The first step in our algorithm is to find N seed triangles upon which we construct the LCFs.

We select N seed vertex using the far distance approach [YKK01]. The first seed is selected as the vertex corresponding to the largest euclidian distance from the geometrical center of all vertices in the first frame. The next seeds are selected one after the other until all N seeds are selected whereas the next seed is selected to be the vertex with the farthest distance from the set of already selected seeds.

We associate with each seed one of its incident triangles and call this triangle the seed triangle. We denote the three vertices of seed triangle of k-th cluster in the f-th frame as $(\mathbf{p}^f_{k,1}, \mathbf{p}^f_{k,2}, \mathbf{p}^f_{k,3})$

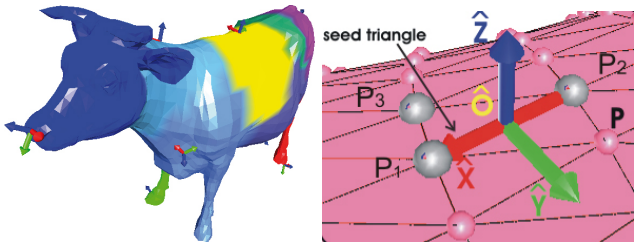

Figure 2: Illustration of the local coordinate frames

4.1.2 Local Coordinate Frame Construction

We assume that each cluster is initialized with the three vertices of the seed triangle. Each cluster $\mathcal{C}_k$ has its own LCF defined on the seed triangle $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ as illustrated in Figure 2. The origin $\mathbf{o}$ is the center of one of its three edges (typically $(\mathbf{p_1}, \mathbf{p_2})$), the x-axis (red arrow) points down the edge $(\mathbf{p_1}, \mathbf{p_2})$, the y-axis (green arrow) is orthogonal to the x-axis in the plane of the seed triangle and the z-axis is orthogonal to the x- and y-axis.

The transformation of a point $\mathbf{p}$ to its local coordinate system $\mathbf{q}$ can be accomplished by an affine transformation with a translation $\mathbf{o}$ and a linear transformation $\mathbf{T}$ ($\mathbf{T}$ is an orthonormal matrix):

$$\mathbf{q} = \mathbf{T}(\mathbf{p} - \mathbf{o})$$

In our algorithm, for each frame f $(1 \leq f \leq F)$ and for each cluster $\mathcal{C}^f_k$ $(1 \leq k \leq N)$, we have $\{\mathbf{T}^f_k, \mathbf{o}^f_k\}$ computed from the points of the seed triangle $(\mathbf{p}^f_{k,1}, \mathbf{p}^f_{k,2}, \mathbf{p}^f_{k,3})$.

4.2 Motion in LCF based Clustering

The clustering process starts with several seed triangles upon which the LCFs are constructed. Then the clustering is obtained by assigning the vertices to the seed triangle in whose LCF they have minimal local coordinate deviation across the F frames. The clustering process consists of the following steps:

1. Initializes the N cluster $\mathcal{C}_k$, $k = 1, ..., N$, to be empty. All vertices are unvisited.

2. Initializes the clusters with the three vertices of their seed triangles upon which the LCFs are constructed.

3. Given an unvisited vertex $\mathbf{p}^f_i$, we do the following:

- Transform its world coordinates into the N LCFs constructed in each frame f, so: $\{\mathbf{q}_{1,i}^f, \mathbf{q}_{2,i}^f, ..., \mathbf{q}_{N,i}^f\}$, where $f = 1, ..., F$.
- Compute the total deviation (motion) of the vertex between each two adjacent frames f and $f-1$ in euclidian space:

$$\theta_{k,i} \quad = \quad \sum_{f=1}^{F} \|\mathbf{q}_{k,i}^f - \mathbf{q}_{k,i}^{f-1}\|^2$$

  $\theta_{k,i}$ represents the total motion of the vertex i in the LCF associated with the cluster k. A small value means that the vertex position has motion that is similar to $\mathcal{C}_k$. Thus the vertex should belong to the cluster k for which the deviation is very small, note k_{min}:

$$k_{min} \quad := \quad argmin_{1 \leq k \leq N}\{\theta_{k,i}\}$$

- We iterate over all vertices, adding the unvisited vertex whose local coordinates are almost invariant in the *LCF* to the cluster $\mathcal{C}_k$ and store its local coordinates for the next step (compression).

  The iteration stops if no more candidate vertices exist. When a vertex is added to a cluster, it is marked as visited. We end up with N clusters that have V_k vertices each. The results of the clustering technique can be seen in Figure 3

4.3 Differential Coding of LCFs

Generally, our approach first transforms the world coordinates of each vertex into local coordinate frame of its cluster. Then, it performs the compression. At reconstruction, the local coordinates are decoded then transformed back to world coordinates. A lossy compression of the vertices of the seed triangle may damage the coordinate frames at the decoding step and as a result, the transformed local coordinates will be damaged. Therefore, the LCF of each cluster should *losslessly* encoded.

We assume that the LCFs of the first frame is already encoded. For each frame and for each new LCF, we encode the locations of their three vertices with the differential encoding. We subtract their coordinates in previously encoded frame from its current coordinates. We quantize the prediction differences, we apply the arithmetic coder to the resulting integers and we update the current locations with the decoded locations.

Figure 3: Results of the clustering process: dance with 14, dolphin with 9, chicken with 10 and cow with 6 clusters. Each cluster is colored differently and encoded separately

4.4 Spatial-Temporal Predictive Coding

Once the segmentation process is finished, and all LCFs are decoded (during the coding), the prediction assumes that the current point does not change relative to the LCF of its cluster. So, for each new point $\mathbf{p}_{k,i}^f$ in the cluster $\mathcal{C}_k^f$ of the frame f, one transforms its world coordinate into local coordinates $\mathbf{q}_{k,i}^f$. Then, one predicts its location from the decoded local coordinates of its location in previous frame $f-1$ by:

$$\mathbf{pred} \quad = \quad \tilde{\mathbf{q}}_{k,i}^{f-1}$$

The delta vectors are computed:

$$\delta_{k,i}^f \quad = \quad \mathbf{q}_{k,i}^f - \mathbf{pred}$$

Unlike the current predictive animated mesh compression techniques [YKL02, IR03, MSK+05] where the delta vectors are encoded in world coordinate frame, here they are computed in the local coordinates.

4.5 DCT Coding

After prediction, we represent the x,y,z coordinates of the delta vectors of each cluster C_k^f as 1D separate signals of length $V_k - 3$ ($V_k - 3$ is the number of vertices in the cluster $\mathcal{C}_k$, minus the three vertices of seed triangle) and encode them with DCT coding.

For each cluster we have three signals:

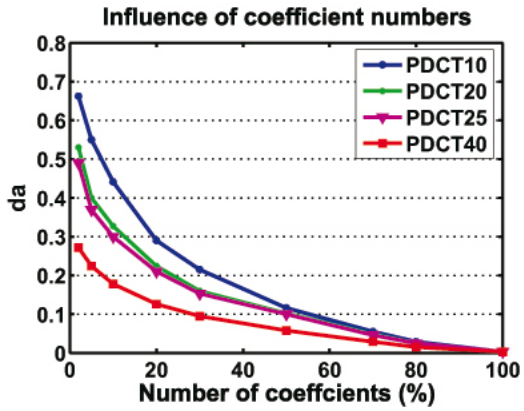

Figure 4: Influence of different numbers of zeroed DCT coefficients (%) on the reconstruction quality using different number of clusters.

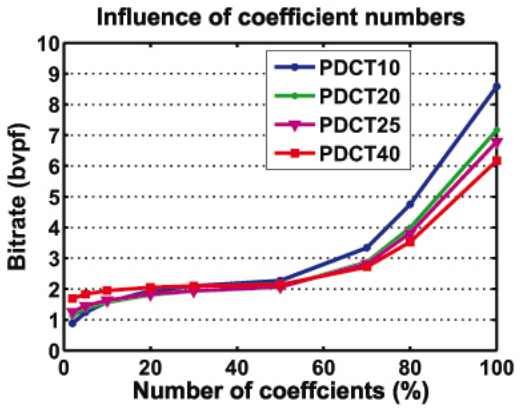

Figure 5: Influence of different numbers of zeroed DCT coefficients (%) on the bitrate using different number of clusters.

$$\begin{aligned}\mathbf{X}_k^f &= \{x_{k,4}^f, x_{k,5}^f, ..., x_{k,V_k}^f\}\\ \mathbf{Y}_k^f &= \{y_{k,4}^f, y_{k,5}^f, ..., y_{k,V_k}^f\}\\ \mathbf{Z}_k^f &= \{z_{k,4}^f, z_{k,5}^f, ..., z_{k,V_k}^f\}\end{aligned}$$

where $k \in 1, ..., N$ and $f \in 1, ..., F$.

For the whole sequences, the number of signals we obtain is $N \times 3 \times F$. We transform each signal vector into the frequency domain using 1D DCT to obtain a more compact representation. Simple 1D DCT is defined as:

$$\mathcal{X}_{k,l}^f = \alpha(l) \sum_{i=4}^{V_k} x_{k,i}^f cos(\frac{\pi(l-4)(2(i-4)+1)}{2(V_k-3)})$$

for $l = 4, ..., V_k$, and $\alpha(l)$ is defined as:

$$\alpha(l) = \begin{cases} \frac{1}{\sqrt{V_k-3}} & \text{for } l = 4 \\ \sqrt{\frac{2}{V_k-3}} & \text{for } l \neq 4 \end{cases}$$

The inverse DCT is similarly defined as:

$$x_{k,i}^f = \sum_{l=4}^{V_k} \alpha(l) \mathcal{X}_{k,l}^f cos(\frac{\pi(l-4)(2(i-4)+1)}{2(V_k-3)})$$

where $i = 4, ..., V_k$.

After DCT transform, the majority of signal energy concentrates on the low frequencies and little on the high frequencies. Hence the high frequencies (insignificant coefficients) can be zeroed yielding a significant reduction in the overall entropy and the signal can then be represented by few high value coefficients without significant distortion. Note that the high frequencies close to zero can also be set to zero automatically using quantization module only.

In our algorithm, we arrange the DCT coefficients from high to low values to easily set the coefficients to zero from bottom to a certain number of coefficients depending on the compression rate and the desired quality.

4.6 Quantization and Arithmetic Coder

The low frequency coefficient (high values) correspond to the coarse details of the cluster while the high frequency coeffients (low values) correspond to the fine details. On the other hand the human eye can perceive the coarse details much more accurately than the fine details. This means that if we use a coarse quantization or set the low value coefficients to zero, the cluster will still retain an acceptable visual quality and we will obtain better compression ratio.

In this version of the algorithm, we uniformly quantize the coefficients to a user specified number of bits per coefficient. Typically, we use a number between 8 and 12 bits, depending on how many DCT coefficients we zeroed. The more coefficients that are zeroed, the more coarser the quantization is, and that better the compression will be at the expense of visual appearance. The finer details can be preserved when only a finer quantization is used and few coefficients are thrown away. For example, if 50% of coefficients have zero values then we use 10 bits quantization. If 90% we use 8 bits only.

One might possibly improve on the present quantization approach by introducing different levels of quantization in each cluster. The high frequencies can be coarsely quantized while the low frequencies can finely quantized.

Note that, the delta vectors of the first frame are encoded using 12 bits quantization while the delta vectors of the LCFs in the whole sequence are quantized to 16 bits.

For further compression the resulting integer values are well encoded with an arithmetic coder [WNC87].

4.7 Reconstruction

To reconstruct the original data cluster, we simply dequantize the coefficients and perform the inverse DCT to find out the delta vectors and add these latter to the predicted location from the perviously decoded frame to recover the original local coordinates. Then, we transform them to world coordinates.

5 Experimental Results

To show the efficiency of our coding *PDCT*, we measured the number of bits per vertex per frame (bpvf). And we used a metric *da* similar to [SSK05] to measure the distortion in the reconstruction animation with regard to the original animation. We used four animations generated in different ways: chicken (3030 vertices, 5664 triangles and 400 frames), cow (2904 vertices, 5804 triangles and 204 frames), dolphin (6179 vertices, 12337 triangles and 101 frames) and dance (7061 vertices, 14118 triangles and 201 frames) sequences.

We compared the compression performance of our algorithm against several techniques: the static mesh compression technique of Touma and Gotsman TG [TG98], the wavelet (AWC) of Guskov and Khodakovsky [GK04] and TLS of Payan and Antonini [PA05], Dynapack of Ibarria and Rossignac [IR03], angle preserving predictor (maverg+angle) of Stefanoski and Ostermann [SO06], PCA of Alexa and Müller, KG (LPCA+PCA) and LPC of Karni and Gotsman [KG04], CPCA of Sattler and al. [SSK05] and RLPCA of Amjoun and Strasser [AS07].

Influence of Cluster Numbers: The number of clusters N is an important compression parameter that affects the compression performance. The bigger this number is, the smoother the shape reconstruction will be and the lower the bit rate that is obtained. If this number is too small, the vertices of the same cluster may behave differently relative to their LCF. Thereby, the prediction in the LCF becomes poor yielding poor compression. In opposite, If N is big, the variation of the vertex relative to the LCF of its cluster becomes smaller and the prediction is more effective.

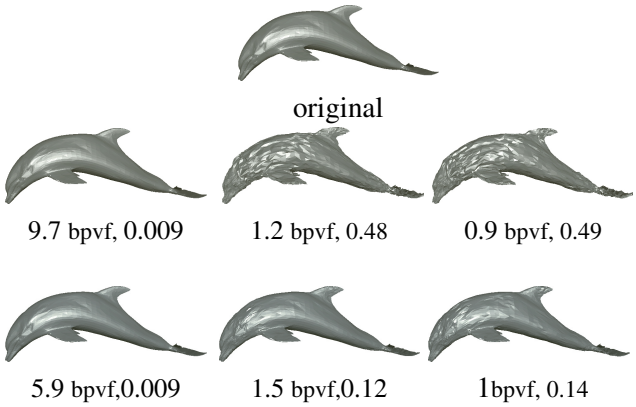

Figure 6: Reconstruction frame 60 of dolphin sequence, original mesh (top arrow), using 10 clusters (middle arrow) and 40 cluster(bottom arrow). From left to right: using different numbers of non-zero coefficients (%) and quantization levels: (100%,12 bits), (2%,12 bits) and (2%,8 bits), at various bit rates in bit per vertex per frame and decoding error (*da*).

Figures 5 and 4 illustrates the curves DCT coefficients/bitrate and coefficients/error *da* for different numbers of clusters.

Figure 9 also shows the rate-distortion curves for different animations at different numbers of clusters: dolphin using 10 and 40 clusters, chicken using 10, 25, 40 and 60 dance using 10, 20 and 40 clusters. We observe that 40 clusters provide better error quality and bit rate than using 10 or 20 clusters. When the number of clusters becomes very large (typically 60 clusters for chicken animation), the bit rate becomes worse, as illustrated in figure 9 (f), because the sequence of the local coordinate frame of each cluster should be losslessly encoded.

Influence of DCT Coefficients: To find the influence of the number of DCT coefficients on the rate and on the reconstruction of animation, we have run our coding on different resolution. Figure 5 and 4 show the results of the number of these coefficients percent for chicken animation. When more coefficients are discarded, better compression (Figure 5) is achieved at the expense of the reconstruction quality (Figure 4).

The effect of the cluster and coefficient numbers

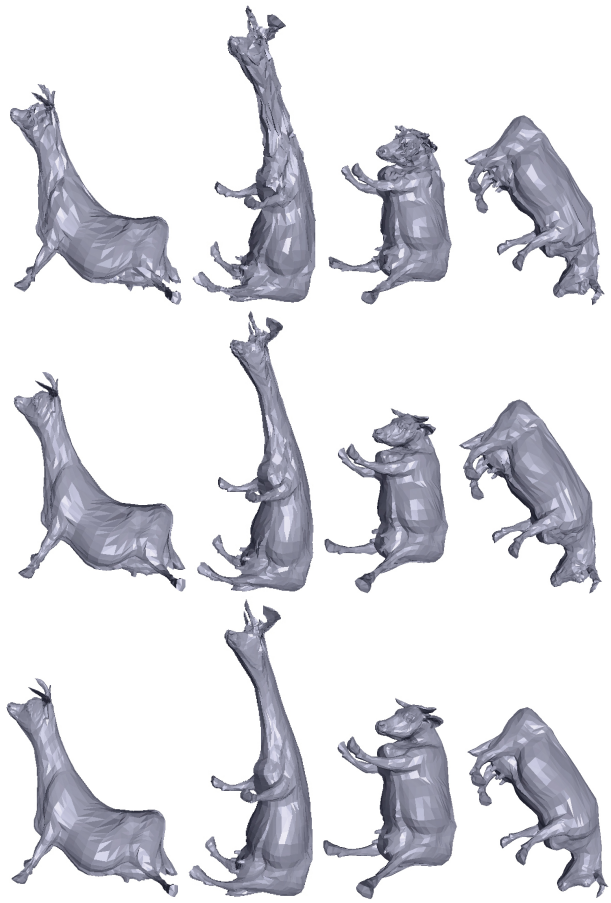

Figure 7: Reconstruction sample frames of cow animation using different quantization levels. From top to bottom: 6, 8, 12 bits.

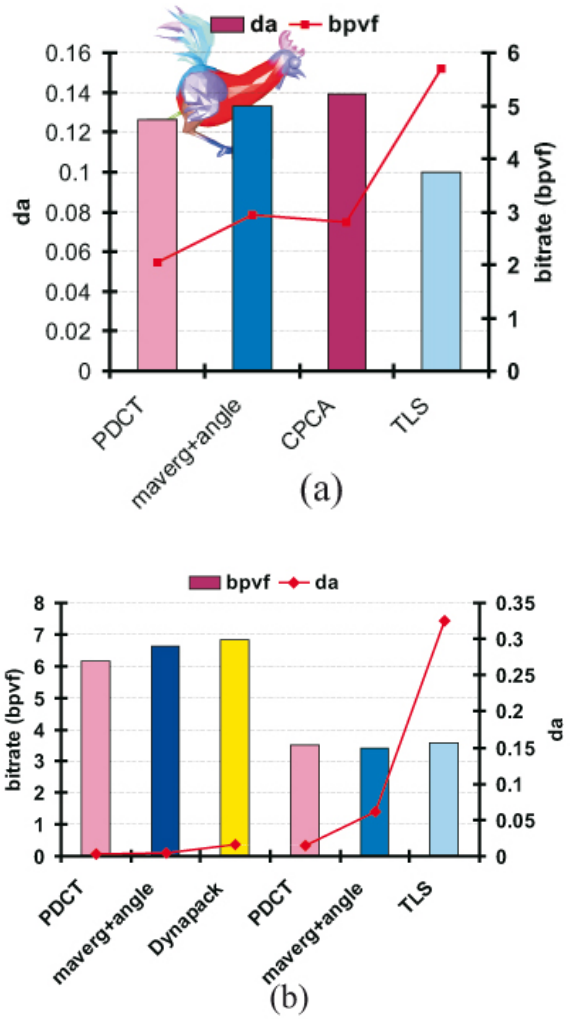

Figure 8: Comparison of our method with different compression algorithms at almost similar bitrates (a) and at similar reconstruction error (b) (chicken sequence).

can also be seen in Figures 6 and 10.

Influence of Quantization Level: Figure 7 illustrates the reconstruction samples of cow animation for different quantization levels. If a coarse quantization is used then the low value DCT coefficients will be zeros. Consequently, the fine details are lost and only the coarse details are detected.

Comparison to other Coders: Figure 9 illustrate the results of running of our coder on three animations compared with different methods. At first glance, we can see that our approach achieves a better rate distortion performance than the standard PCA, LPC, KG and TG for the three models. This result is obvious since the animation coding based on static techniques only exploit the spatial coherence and the linear prediction coding only uses the temporal coherence. Furthermore, the standard PCA only approximates the global linearity and is less effective for nonlinear animation.

For the CPCA and AWC algorithms, we achieve better or similar results. Figure 9 (a) shows that for the cow animation which contains extreme deformations, our method is significantly better than the KG method and comes close to the CPCA and to wavelet based methods (TLS and AWC).

For the chicken and the dolphin sequences, our method performs better than all the above methods, including the predictive techniques (Dynapack and maverg+angle). This improvement is due to the clustering of the model into rigid parts making the prediction more efficient in the local rather than the world, coordinates and to the further DCT coding which leads to a significant reduction in the overall entropy.

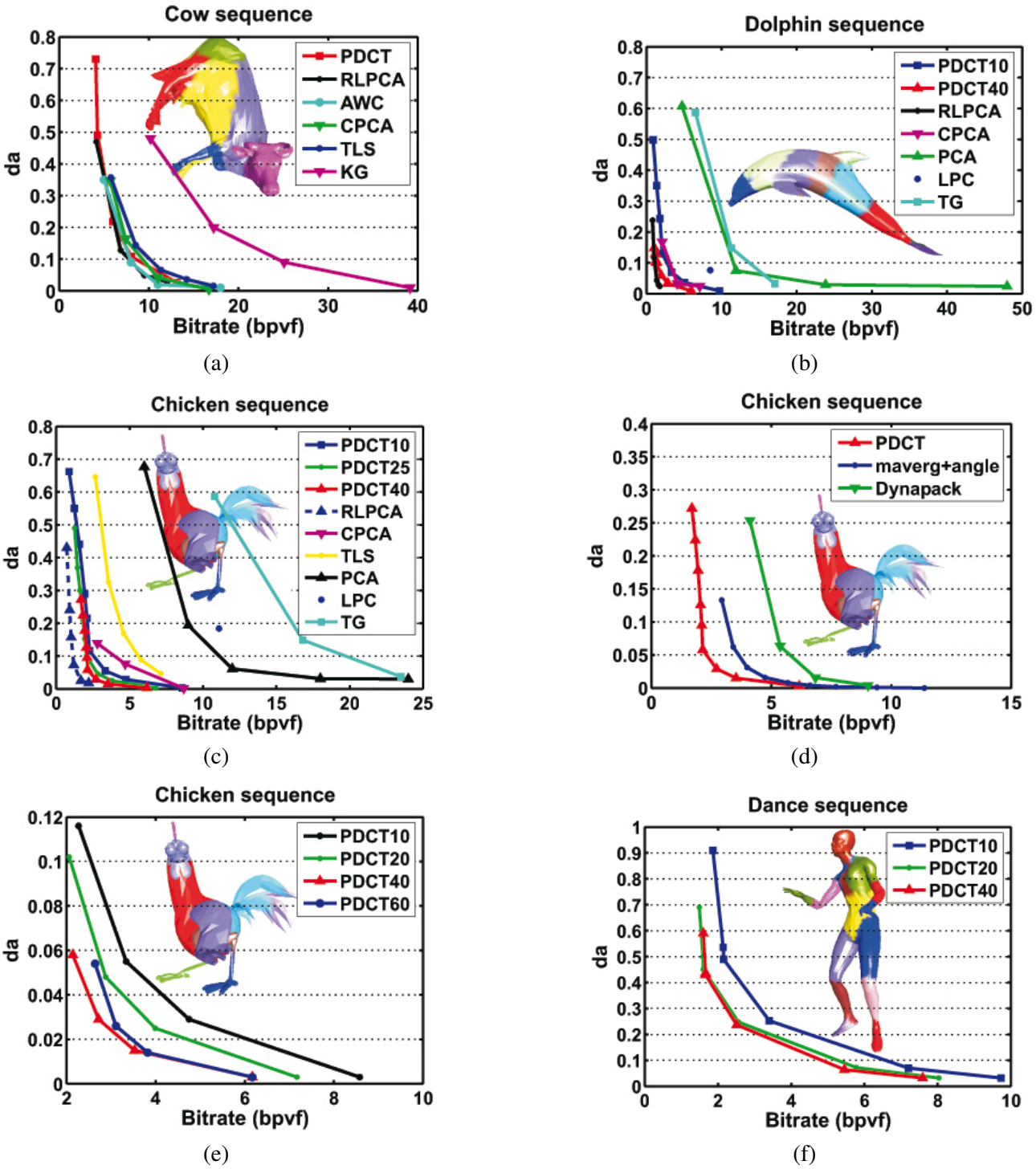

Figure 9: Rate distortion curves for the cow (a), dolphin (b), chicken (c), (d), (e), and dance (f) sequences.

The RLPCA method overcomes all other methods, including ours, for chicken animation while it comes close to our for the other models. Our method PDCT uses as similar clustering process as RLPCA scheme. However, the difference arises in the way of encoding the local coordinates. The RLPCA considers the entire cluster sequence and exploit the global coherence using PCA. While PDCT have to encode frame by frame using predictive and spatial DCT coding, the method is well suited for real-time compression.

In Figure 8, we compare our approach against several approaches. At (almost) similar quality (Figures 8 (a)), our coder archives gains up to 30% and 27%, over the angle preserving predictor (maverg+angle) and CPCA respectively. At (almost) similar numbers of bits (Figures 8 (b)), our approach obtains better animation quality from 28% up to 76% over maverg+angle, 81% over Dynapack (using Extended Lorenzo Predictor) and 95% over TLS.

6 Conclusion

In this paper we introduced a simple and efficient compression technique for dynamic 3D mesh based on predictive and DCT coding. First, the algorithm clusters the vertices into a given number of clusters depending on their motion in their LCF. This technique is simple and can be well adapted for different purposes. Second, the location of each new vertex in the current frame, is predicted from its location in the previous frame. The effectiveness of prediction coding depends strongly on the clustering process. Indeed, if the vertices are well clustered then the motion relative to the LCF between two successive frames tends to be zero. Third, the delta vectors are further encoded with DCT transform to reduce the code length since the entropy in frequency domain is smaller than the entropy coding of delta vectors. The resulting DCT coefficients are quantized and encoded with an arithmetic coder.

Experimental results show that our algorithm is competitive when compared to the state-of-the-art techniques. In this context, it is important to note that our coder is applicable to meshes and point-based models regardless of how the animation is generated. The drawback of the proposed approach is that it doesn't support progressive transmission. Moreover, for a very low and fixed number of coefficients, not all frames can be reconstructed at the same desired level of quality.

Future Improvement: The clustering used in our approach produces clusters of different sizes. Thereby, different numbers of DCT coefficients are produced. If one chooses a fixed number for all clusters then there may be too few coefficients to recover the clustered vertices at a desired accuracy and possibly too many coefficients for other clusters. Therefore, the selection of the number of significant coefficients and quantization level, is necessary to properly recover the original data of each cluster with a certain accuracy. Therefore, we plan to introduce a rate distortion optimization that trades off between rate and the total distortion, overcoming the aforementioned drawback. We also plan to develop temporal DCT in combination with predictive coding in local coordinates. This approach is more suitable for progressive transmission. For a large sequence of meshes, the animation may become more complex and the clustering can produce poor prediction for some successive frames. Therefore, we propose to cut the sequence into short clip and update the clustering for each new coming clip. The first frame of each clip should be encoded spatially as I-frame.

Acknowledgements We would like to thank Douglas Cunningham for proof-reading, Zachi Karni and Hector Briceño for providing us with the animated meshes and Mirko Sattler, Igor Guskov, Frédéric Payan and Nikolče Stefanoski for the results of their methods. The Chicken sequence is property of Microsoft Inc.

Citation

Rachida Amjoun and Wolfgang Straßer, *Predictive-DCT Coding for 3D Mesh Sequences Compression*, Journal of Virtual Reality and Broadcasting, 4(2007), no. 6, July 2008, urn:nbn:de:0009-6-14446, ISSN 1860-2037.

References

[AG05] Pierre Alliez and Craig Gotsman, *Recent Advances in Compression of 3D Meshes*, Advances in Multiresolution for Geometric Modelling serie, Springer, Berlin, 2005, ISBN 3-540-21462-5, pp. 3–26.

[AM00] Marc Alexa and Wolfgang Müller, *Representing Animations by Principal Components*, Computer graphics Forum **19** (2000), no. 3, 411–426, ISSN 0167-7055.

[AS07] Rachida Amjoun and Wolfgang Straßer, *Efficient Compression of 3D Dynamic Mesh Sequences*, Journal of the WSCG, vol. 15, 2007, C31, ISSN 1213-6972, pp. 99–107.

[ASS06] Rachida Amjoun, Ralf Sondershaus, and Wolfgang Straßer, *Compression of Complex Animated Meshes*, Computer Graphics International (Hangzhou, China), vol. LNCS, Vol. 4035, Springer, 2006, ISBN 3-540-35638-X, pp. 606–613.

[BSM+03] Hector M. Briceno, Pedro V. Sander, Leonard McMillan, Steven Gortler, and Hugues Hoppe, *Geometry videos: a new representation for 3D animations*, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2003, ISSN 1727-5288, pp. 136–146.

[GA03] Stefan Gumhold and Rachida Amjoun, *Higher Order Prediction for Geometry Compression*, International Conference On Shape Modelling And Applications, 2003, ISBN 0-7695-1909-1, pp. 59–68.

[GK04] Igor Guskov and Andrei Khodakovsky, *Wavelet compression of parametrically coherent mesh sequences*, Proceedings of the 2004 ACM SIGGRAPH/-Eurographics symposium on Computer animation (Grenoble, France), 2004, ISBN 3-905673-14-2, pp. 183–192.

[IG03] Martin Isenburg and Stefan Gumhold, *Out-of-core compression for gigantic polygon meshes*, ACM Transaction on Graphics **22** (2003), no. 3, 935–942, ISSN 0730-0301.

[IR03] Lawrence Ibarria and Jarek Rossignac, *Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity*, Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation (San Diego, California), Eurographics Association, 2003, ISSN 1727-5288, pp. 126–135.

[KG04] Zachi Karni and Craig Gotsman, *Compression of Soft-Body animation sequences*, Computer and Graphics **28** (2004), no. 1, 25–34, ISSN 0097-8493.

[Len99] Jerome Edward Lengyel, *Compression of time-dependent geometry*, Proceedings of ACM symposium on Interactive 3D graphics (Atlanta, Georgia, United States), ACM Press, 1999, ISBN 1-58113-082-1, pp. 89–95.

[MSK+05] Karsten Muller, Aljoscha Smolic, Matthias Kautzner, Peter Eisert, and Thomas Wiegand, *Predictive Compression of Dynamic 3D Meshes*, IEEE International Conference on Image Processing ICIP, 2005, ISBN 0-7803-9134-9, pp. 621–624.

[MZP06] Khaled Mamou, Titus Zaharia, and Franoise Prêteux, *A skinning approach for dynamic 3D mesh compression*, Computer Animation Virtual Worlds **17** (2006), no. 3-4, 337–346, ISSN 1546-4261.

[PA05] Frederic Payan and Marc Antonini, *Wavelet-based Compression of 3D Mesh Sequences*, Proceedings of IEEE ACIDCA-ICMI'2005 (Tozeur, Tunisia), nov 2005.

[Ros01] Jarek Rossignac, *3D Compression Made Simple: Edgebreaker with Zip&Wrap on a CornerTable*, Shape Modeling International Conference (Genovy, Italy), 2001, ISBN 0-7695-0853-7, pp. 278–283.

[Ros04] J. Rossignac, *Handbook of Discrete and Computational Geometry*, 2nd ed., ch. Chapter 54: Surface simplification and 3D geometry compression, pp. 1209–1240, CRC Press, 2004, ISBN 1584883014.

[SO06] Nikolce Stefanoski and Joern Ostermann, *Connectivity-Guided Predictive Compression of Dynamic 3D Meshes*, IEEE International Conference on Image Processing, oct 2006, ISBN 1-4244-0481-9, pp. 2973–2976.

[SSK05] Mirko Sattler, Ralf Sarlette, and Reinhard Klein, *Simple and efficient compression of animation sequences*, Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM Press, 2005, ISBN 1-7695-2270-X, pp. 209–217.

[TG98] Costa Touma and Craig Gotsman, *Triangle mesh compression*, Graphics Interface'98, Morgan Kaufmann,San Francisco, 1998, ISBN 1-55860-550-9, pp. 26–34.

[WNC87] Ian H. Witten, Radford M. Neal, and John G. Cleary, *Arithmetic Coding for Data Compression*, Communications of the ACM **30** (1987), no. 6, 520–540, ISSN 0001-0782.

[YKK01] Zhidong Yan, Sunil Kumar, and C.-C. Jay Kuo, *Error-resilient coding of 3-D graphic models via adaptive mesh segmentation*, IEEE Transactions on Circuits and Systems for Video Technology **11** (2001), no. 7, 860–873, ISSN 1051-8215.

[YKL02] Jeong-Hyu Yang, Chang-Su Kim, and Sang-Uk Lee, *Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction*, IEEE Transactions on Circuits and Systems for Video Technology **12** (2002), no. 12, 1178–1184, ISSN 1051-8215.

[ZO04] Jinghua Zhang and Charles B. Owen, *Octree-based Animated Geometry Compression*, Proceedings of IEEE on Data Compression, IEEE Computer Society, 2004, ISBN 0-7695-2082-0, pp. 508–517.

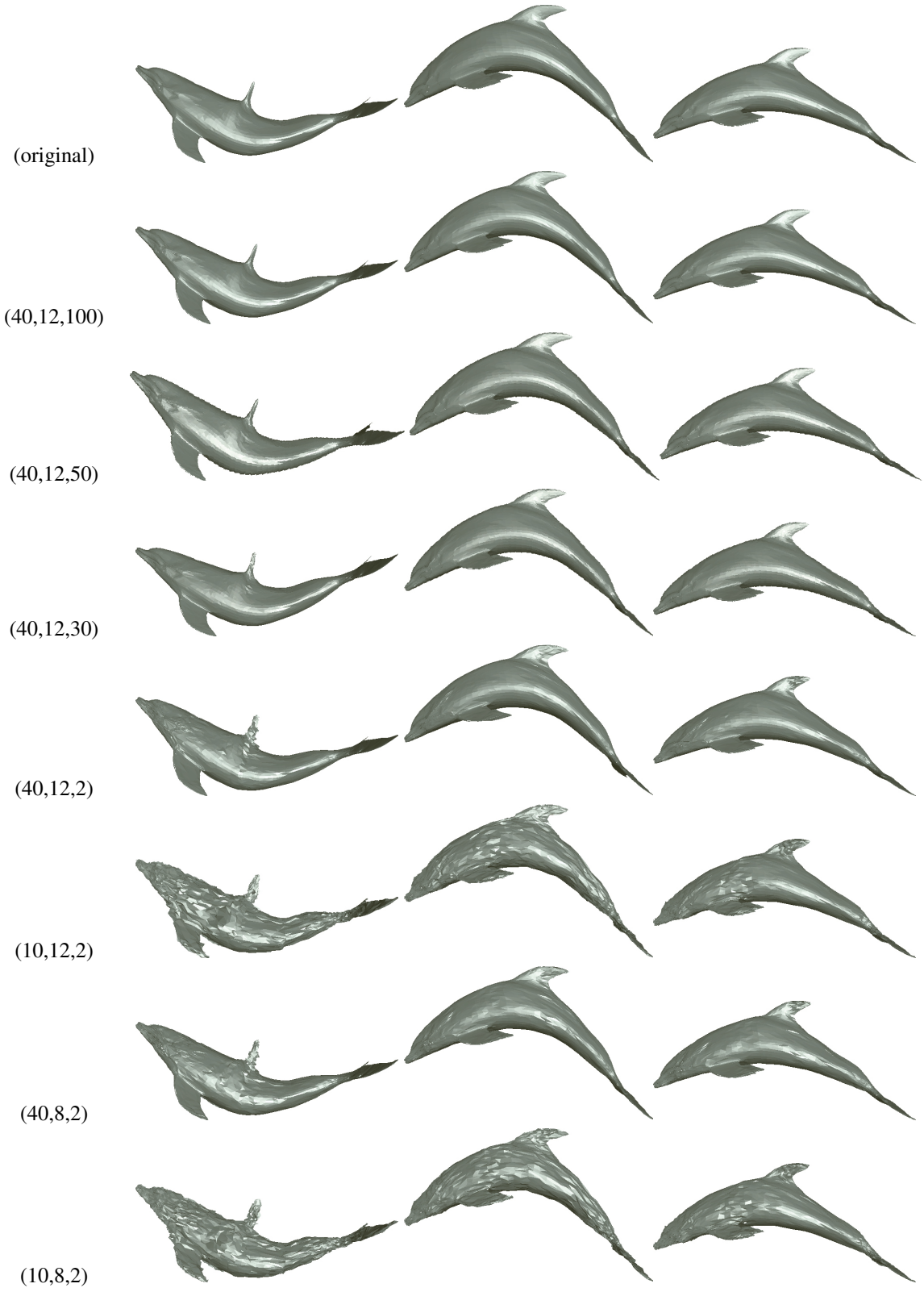

Figure 10: Reconstruction sample frames of dolphin sequence. The numbers in the first column are the number of clusters, quantization level and coefficient number (%).