Multi-Chart Geometry Video: A Compact Representation for 3D Animations

Khaled Mamou, Titus Zaharia, and Françoise Prêteux
Groupe des Ecoles des Télécommunications
Institut National des Télécommunications / ARTEMIS Department
9, rue Charles Fourier 91011 Evry, France
<Khaled.Mamou, Titus.Zaharia, Françoise.Preteux>@int-evry.fr

Abstract

This paper introduces a new compression scheme, so-called Multi-Chart Geometry Video (MCGV), for 3D dynamic meshes with constant connectivity and time-varying geometry.

The core of the proposed method is a piecewise affine predictor coupled with a Multi-Chart Geometry IMage (MCGIM) representation of the residual errors. The mesh is first partitioned into vertex clusters whose motion can be accurately described by a unique 3D affine transform. The prediction errors are represented as MCGIMs which are compressed by using standardized image encoders such as JPEG and MPEG-4.

The performances of our encoder are objectively evaluated on a data set of six animation sequences with various sizes, geometries and topologies, and exhibiting both rigid and elastic motions. The experimental evaluation shows that the proposed MCGV achieves up to 60% lower compression distortions than the geometry video approach, while outperforming (with 30% to 94% lower distortions) the RT, MPEG-4/AFX-IC, D3DMC, PCA and Dynapack techniques.

1. Introduction

The growing industries of CGI (*Computer Generated Imagery*) films and video games extensively exploit dynamic 3D content.

Most often, such content is represented as highly memory consuming animated 3D meshes, that need to be efficiently stored, transmitted and visualized over various types of fixed or mobile networks and terminal devices.

The bone-based animation (BBA) technique adopted by the MPEG-4 AFX (*Animation Framework eXtension*) standard [3] provides an intuitive and compact motion representation of such dynamic meshes, expressed in terms of 3D pose parameters associated with a hierarchical animation skeleton.

The skinning model is transmitted only once, for the first frame. Then, solely the animation parameters are transmitted for each frame.

The MPEG-4/AFX approach requires the creation of advanced skinning models, involving the partition of the mesh vertices into regions of influence associated with bones, the specification of weighting coefficients for each mesh vertex, the integration of additional bones for simulating complex, non-rigid motions…

Such skinning models are manually created, with the help of complex 3D authoring environments (such as Maya, 3DS Max…). This tedious modelling process requires both technical and artistic skills, and involves a huge amount of human and financial resources. For this reason, content creators are reluctant to the BBA technology, which requires the transmission of such expensive skinning models that may be reused by tierce for generating new 3D content. The animation industry currently disregards such a framework, which is not designed for managing intellectual property issues.

Therefore, a key-frame representation is usually adopted by the industrials to ensure content protection. Here, neither the skinning model nor the animation parameters are transmitted. Instead, the animation sequence is stored as a set of consecutive 3D meshes representing key-frames. The intermediate frames are derived by applying interpolation procedures.

Key-frame representations model generic animations and handle the content protection issue. In addition, they are completely independent from the underlying animation technique used for generating the content (physical simulations, morphing and skinning techniques, motion capture, motion cloning/retargeting approaches…), and thus provide a generic format for consistent and multi-platform 3D animation. However, the main drawback of such representations is related to the huge amount of data required since even for short sequences of a few minutes, several thou-

sands of key models are needed. Elaborating then efficient coding techniques for such memory-consuming representations becomes a highly challenging issue.

This paper proposes a new compact image-based representation for 3D dynamic meshes, so-called *Multi-Chart Geometry Video (MCGV)*.

Section 2 presents a critical overview of the state of the art of dynamic mesh compression techniques. Section 3 describes the proposed MCGV compression scheme and details its main stages: motion segmentation, piecewise affine motion compensation and multi-chart geometry image construction. MCGV's performances are objectively evaluated and discussed in Section 4. Finally, Section 5 concludes the paper and opens perspectives of future work.

2. Related Work

Let us first recall the concept of time-dependent geometry compression, introduced in [11].

Let $(M_i)_{i\in\{0,\ldots,F-1\}}$ be a sequence of meshes (F stands for the number of frames) with a constant topology T, and a time-varying geometry, denoted by $(G_i)_{i\in\{0,\ldots,F-1\}}$. Here, the mesh geometry G_i at frame i is represented as a vector of size $3 \times V$ (with V being the number of vertices), which includes the three x, y and z coordinates of the whole set of mesh vertices.

The goal of time-dependent geometry compression is to elaborate adequate and compact representations of the geometry sequence $(G_i)_i$, taking into account both spatial and temporal correlations.

The issue of dynamic 3D mesh compression has gained rapidly the interest of the scientific community as testifies the important number of recently developed works reported in the literature (see [12] for an overview).

In [11], Lengyel introduced the first clustering-based approach for encoding dynamic 3D meshes. The principle consists of splitting the mesh into sub-parts whose motion can be accurately described by rigid transforms. The object's motion is described by a set of rigid motion parameters associated with each cluster and the prediction residuals associated with each vertex. The Lengyel's approach is further improved in [5], where animation is expressed only in terms of rigid transforms (RT). Authors introduce a new weighted least square mesh segmentation algorithm which minimizes the number of clusters under a distortion bound criterion.

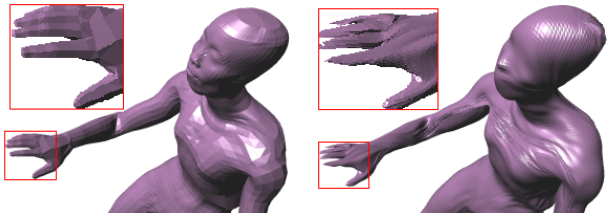
A different clustering-based compression scheme is proposed in [19]. Here, the motion field of each frame is represented as an octree structure and a set of associated motion vectors. An optimized version of this approach, so-called Dynamic 3D Mesh Coder (D3DMC) is introduced in [13]. Such clustering-based approaches are able to compactly describe a large category of motions. However, their major limitation is related to the segmentation procedure which is computationally complex. In addition, such algorithms generally cause disgraceful discontinuities at low bitrates at the level of the cluster borders, since different motion models are considered.

The Interpolation Compression (IC) scheme [8] recently adopted by the MPEG-4/AFX standard proposes a key-framing technique. The animated mesh is represented as a set of key-frames undersampling the initial sequence. The in-between frames are obtained by applying an interpolation procedure. Furthermore, the redundancies between successive key-frames are eliminated by applying a local spatio-temporal prediction scheme. The Dynapack approach [7] exploits a similar spatio-temporal prediction strategy with more elaborate predictors, so-called *Extended Lorenzo Predictor (ELP)* and *Replica*. The MPEG-4/AFX-IC and Dynapack techniques offer the advantages of simplicity and low computational cost, which makes them well-suited for real-time decoding applications. However, because of the underlying deterministic traversal of the mesh vertices involved, such approaches do not support more advanced functionalities such as progressive transmission and scalable rendering.

In [1], Alexa and Müller introduce a different class of approaches based on a principal component analysis (PCA) of the mesh deformation field. In [10], Karni and Gotsman enhance the PCA approach by introducing a second order temporal linear prediction. An additional refinement is introduced in [17], where authors propose to partition the mesh vertices into a set of clusters that are optimally adapted to the PCA representation. The PCA-based approaches are specifically adapted for long repetitive animation sequences of small meshes. However, they suffer from the high computational complexity (cubic with the number of the mesh vertices) of the singular value decomposition algorithm involved.

The Wavelet Animation Compression (AWC) approach [6] constructs irregular wavelets on the top of a progressive mesh hierarchy. The AWC achieves very low bitrates while fully supporting progressive transmission and scalable rendering functionalities. However, the AWC approach is well-suited only for large meshes with a few connected components.

The Geometry Video (GV) [4] technique exploits a mesh cut and a stretch minimizing parameterization [15] over a 2D square domain. Here, the initial mesh connectivity is completely discarded and replaced by a regular one, obtained by uniformly sampling the parametric domain. The resulting sequence of *geometry images* is then compressed by using traditional video encoding techniques. A motion compensation procedure based on a global affine motion model is first applied. The resulting prediction errors are then compressed by using a wavelet-based encoding



(a) MCGV (b) GV

Figure 1. MCGV (a) vs. GV (b): GV's remeshing procedure leads to loss of surface details and tangent plane discontinuities.

scheme.

The main drawbacks of the GV approach are related to the remeshing procedure involved which may lead to (Figure 1):

- a loss of surface details due to an undersampling of the parametric domain,
- bad triangulations of the resampled meshes in the case of high distortion parameterizations,
- tangent plane discontinuities at the level of the cut.

In this paper, we propose a new compact representation for 3D animated meshes, so-called *Multi-Chart Geometry Video (MCGV)* which extends both the GV and RT approaches. The limitations of the GV technique are overcome by the MCGV approach, which (Figure 1):

- Preserves the initial mesh connectivity, and thus avoids remeshing related problems,
- Improves the motion compensation stage by applying a piecewise affine predictor, more suited for describing the motion of articulated characters,
- Exploits a low-distortion atlas of parameterization [20] instead of a unique and restrictive mapping on a 2D square domain.

With respect to the RT approach, the MCGV technique adopts a different strategy. Instead of multiplying the number of motion models, which may obviously lead to an oversegmentation of the dynamic mesh into small patches without any anatomical interpretation, a fixed number of patches is here preferred. In order to achieve high compression gains, an additional coding block of the motion compensation residual errors is adopted.

The proposed MCGV approach is described in detail in the next section.

3. Multi-Chart Geometry Video

The MCGV technique represents the geometry of a dynamic mesh by a piecewise affine geometry predictor and a set of Multi-Chart Geometry IMages (MCGIM) [16] storing the prediction errors. Figure 2 illustrates the MCGV construction process.

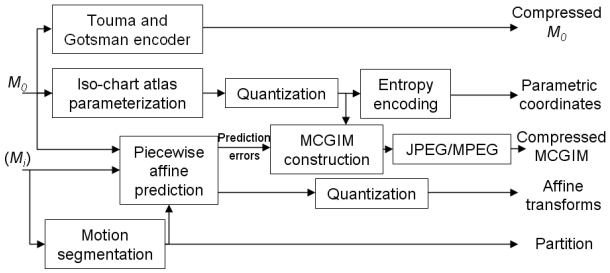


Figure 2. The MCGV construction process.

First, the mesh vertices are partitioned into groups exhibiting the same affine motion with respect to the first frame. This partition is then exploited in order to compute a piecewise affine predictor minimizing the prediction errors. The so-obtained errors representing the non-affine part of the motion are stored as 2D images. Here, a low-distortion atlas of parameterization is computed [20] and used to derive the MCGIMs for each frame. The MCGIMs are finally encoded by using standard 2D compression techniques such as JPEG and MPEG-4. The final bitstream consists of:

- the first frame compressed by using the Touma and Gotsman approach [18],
- the mesh partition,
- the quantized and entropy coded parametric coordinates of all vertices,
- the quantized coefficients defining the piecewise affine predictor, and
- the set of compressed MCGIMs.

Let us now detail the main blocks in this scheme, starting with the motion-based segmentation stage.

3.1. Motion-based segmentation

The objective of the segmentation stage is to obtain a partition $\pi = (\pi_k)_{k\in\{1,\dots,K\}}$ of the mesh vertices into K clusters whose motion can be accurately described by 3D affine transforms. The proposed approach, detailed herebelow, takes into account the motion of the mesh vertices over the whole set of frames in the sequence.

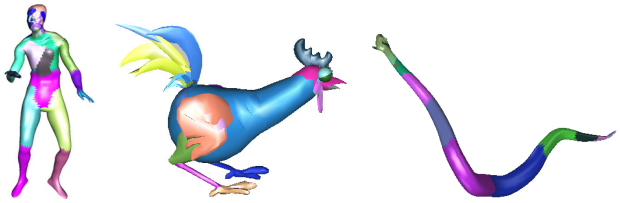
First, an affine transform A_i^v describing the affine motion of the local neighbourhood of each vertex $v \in \{1, ..., V\}$ at frame $i \in \{0, ..., F-1\}$ with respect to first frame is computed as follows:

$$A_i^v = \arg\min_A \left(\sum_{v \in v^*} \|A\chi_0^v - \chi_i^v\|^2 \right), \qquad (1)$$

where A is 4×4 matrix representing an affine transform, v^* is the third order neighbourhood of the vertex v (*i.e.*, vertices connected to v by a path consisting of at most three edges), and χ_i^v is a 4D vector representing the homogeneous coordinates of the vertex v at frame i.

Then, the set of $(A_i^v)_{i \in \{0,...,F-1\}}$ is stored as a single vector $\alpha^v \in \mathbb{R}^{12 \times F}$ (since an affine transform is completely defined by 12 real-valued coefficients). Finally, the partition π is obtained by applying the k-means clustering algorithm [9] to the vector set $(\alpha^v)_{v \in \{1,...,V\}}$.

Figure 3 shows some segmentation results for the animated meshes "Dance", "Chicken" and "Snake". The number of clusters K was set here to 20. We note that for articulated characters the algorithm successfully recovers the corresponding body subparts. In all cases, the procedure yields clusters consisting of topologically connected vertices, even if no topological information was taken into account by the k-means clustering algorithm. Intuitively, this results shows that the piecewise affine motion modelling detailed in the next section is well suited for describing complex motions.



(a) "Dance" (b) "Chicken" (c) "Snake"

Figure 3. Motion-based segmentation of dynamic meshes with different motions, shapes and complexities.

3.2. Piecewise affine prediction

Once the partition of the dynamic mesh determined, for each frame i, the motion field is modeled as a set of K affine transforms denoted by $(H_i^k)_{k \in \{1,...,K\}}$. With the same notations as above, the affine transform H_i^k associated with patch k at frame i is defined as:

$$H_i^k = \arg\min_A \left(\sum_{v \in \pi_k} \|A\chi_0^v - \chi_i^v\|^2 \right). \qquad (2)$$

The set $(H_i^k)_{k \in \{1,...,K\}}$, together with the partition π provide a piecewise affine predictor of the frame i from the frame 0 defined as :

$$\forall\, v \in \{1, ..., V\}, \quad \widehat{\chi}_i^v = H_i^{k(v)} \chi_0^v, \qquad (3)$$

where $k(v)$ denotes the cluster to which the vertex v belongs.

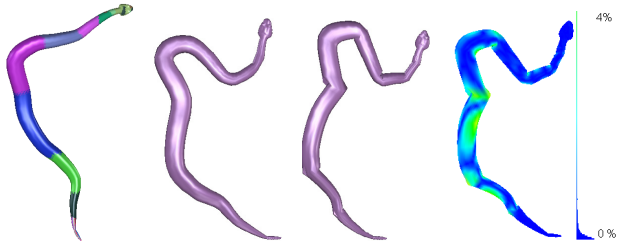
The prediction errors $e_i^v = (e_i^{v,x}, e_i^{v,y}, e_i^{v,z}, 0)^t$ are defined as:

$$\forall\, v \in \{1, ..., V\}, e_i^v = \chi_i^v - \widehat{\chi}_i^v. \qquad (4)$$

The least squares optimization problem described by Equation (2) leads to an over-determined system of linear equations, solved by using a pseudo-inverse-based method [14].

Figure 4 shows the original frame 36 of the "Snake" animation, its predicted version by using 20 clusters and the distribution of the prediction errors represented in pseudo-colors.

We note that the proposed predictor captures well the affine motion of the mesh clusters. The maximal prediction error represents here only 4% of the first frame's bounding box diagonal. We also note that the highest error values are obtained at the level of the cluster boundaries. This shows the limitations of the piecewise affine motion model which introduces inherent discontinuities at the frontier of adjacent patches.



(a) (b) (c) (d)

Figure 4. Piecewise affine prediction for the "Snake" animation: (a) original segmented frame 0; (b) frame 36; (c) frame 36 predicted from (a); (d) distribution of the prediction errors.

The so-obtained prediction errors are finally stored, for each frame, within a 2D image, as described in the next section.

3.3. Multi-chart geometry images

Representing the prediction errors associated with a 3D mesh as 2D images requires the construction of a mesh pa-

rameterization which maps the mesh surface onto a 2D domain.

In order to minimize the geometric distortions inherent to any mesh parameterization technique, we have adopted the iso-chart stretch minimizing technique introduced in [20].

The principle consists of constructing, rather than a global parameterization, an atlas of parameterization associated with a mesh partition into patches. The mesh partition is optimally derived by minimizing a parameterization distortion criterion. Each patch is mapped onto a 2D domain whose shape is optimized for reducing distortions. The set of 2D domains corresponding to all patches is finally packed within the unit square.

This principle is illustrated in Figure 5.a, for the "Dance" sequence.

The blank areas between patches correspond to some "don't care" parametric values, which do not have any corresponding points on the mesh surface.

In our case, the atlas of parameterization is determined solely for the first mesh in the sequence. The initial partition derived from the motion-based segmentation stage is considered as initialization and refined as described hereabove for ensuring a low distortion parameterization.

The parametric unit square domain is then uniformly sampled on a (256×256) rectangular lattice in order to generate three MCGIMs (one for each of the x, y and z coordinates). For each frame, the MCGIMs store the prediction errors $e_i^{v,x}$, $e_i^{v,y}$ and $e_i^{v,z}$ (Figure 5.b).

However, directly compressing such MCGIMs would lead to poor compression performances because of the high frequencies generated by the frontiers between patches and blank areas. In order to make possible the efficient compression of such MCGIMs, we consider a padding technique which extrapolates the blank areas from the patch domains. Let us note that such a padding technique does not affect the reconstruction process since pixels in the blank areas can take arbitrary values, without any influence on the reconstructed mesh.

Since the MCGIM error images are further encoded with JPEG/MPEG encoders, the padding method proposed in this paper is specifically adapted to such compression schemes, which are based on a block DCT transform.

The images are first partitioned into (8×8) square blocks. The atlas of parameterization is then reorganized such that no (8×8) block includes pixels from different patches.

Each block $B = (B_{i,j})_{i,j \in \{1,\ldots,8\}}$ is composed of fixed-value pixels, corresponding to patch areas and free pixels, associated with blank regions. Let B^{free} and B^{fix} respectively denote the sets of free and fixed pixels in block B.

As a measure of smoothness of the block B we consider the function ψ defined as:

$$\psi(B) = \sum_{(p,q) \in \aleph_B} (B_p - B_q)^2, \quad (5)$$

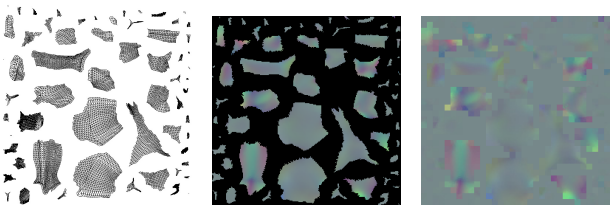
where $\aleph_B$ is the set of all couples of neighbour pixels of the block B (in 8-connectivity).

Minimizing $\psi(B)$ is equivalent to solving the following set of linear equations:

$$\forall p \in B^{free},\ 8B_p - \sum_{q \in p^* \cap B^{free}} B_q = \sum_{h \in p^* \cap B^{fix}} B_h, \quad (6)$$

where p^* stands for the neighbourhood of p.

The linear system (6) being sparse, symmetric, and positive defined, can be efficiently solved by using a gradient conjugate method [14]. Figure 5.c illustrates the smoothlly padded MCGIM obtained for the frame 48 of the "Dance" sequence.



(a) (b) (c)

Figure 5. (a) Atlas of parameterization, (b) MCGIM (63 components) and (c) smoothly padded MCGIM.

4. Experimental results

4.1. Evalution corpus

In order to be able to perform objective comparisons, we have considered a data set of six animation sequences, used by the majority of the works reported in the literature, and so-called "Dance", "Chicken", "Dolphin", "Humanoid", "Cow", and "Snake". Table 1 summarizes their properties, expressed in terms of numbers of vertices (V), frames (F), and connected components (CC).

The considered models offer a good variability in terms of both spatial and temporal sizes. Thus the number of vertices varies from 2904 (for the "Cow" model) to 9179 vertices (for the "Snake" model), while the number of frames ranges from 101 to 400. All models are composed of a unique CC, excepting the "Chicken" mesh, which includes 41 CCs, with an average of 73 vertices per CC.

Animation sequence	V	F	CC
"Cow"	2904	204	1
"Chicken"	3030	400	41
"Dolphin"	6180	101	1
"Dance"	7061	201	1
"Humanoid"	7646	154	1
"Snake"	9179	134	1

Table 1. Properties of the mesh sequences in the test data set.

4.2. Objective evaluation criteria

In order to be able to evaluate and compare the compression performances for mesh sequences with various number of frames and vertices, compression rates are expressed in terms of bits per vertex per frame (bpvf).

The compression distortions are measured by using the RMSE error [2] between initial and reconstructed (decoded) meshes. The RMSE error between two mesh sequences is defined as the mean value of the frame to frame RMSE, computed over the set of all the frames in the sequence.

4.3. Compression results

In order to validate the proposed MCGV approach, we have selected for comparaison six compression techniques: D3DMC, AFX-IC, RT, PCA, Dynapack and GV. The compression results of the retained techniques are those reported in [13], [5], [7] and [4]. Table 2 summarizes the availability of the compression results with respect to the test sequences, and provides the references considered as source.

Animation	D3DMC	AFX-IC	RT	PCA	Dynapack	GV
"Cow"	-	-	-	-	-	[4]
"Chicken"	[13]	[13]	[5]	-	[7]	-
"Dolphin"	-	-	-	-	-	-
"Dance"	-	-	-	-	-	[4]
"Humanoid"	[13]	[13]	-	-	-	-
"Snake"	-	-	[5]	[5]	-	[4]

Table 2. Available compression results and associated source references.

Let us note that the "Chicken" sequence cannot be handled by the GV approach since it is composed of multiple CCs. In the case of the "Cow" animation, only the first 81 frames were considered by the GV encoder.

Concerning the parameters involved in the proposed MCGV approach, the number of clusters K was set to 20 for all the sequences. The real coefficients describing the affine transforms were quantized on 17 bits. The first frame of the sequence was encoded by using the Touma and Gotsman approach [18], with 12 bits of quantization for the geometry. The parametric coordinates associated with each vertex were also quantized on 8 bits. In order to be able to use JPEG/MPEG encoders, the predictions residuals were quantized on 8 bits.

The bitrates reported in this paper correspond to the whole binary stream necessary for decoding the animation sequences including: the compressed first frame, the parametric coordinates of the mesh vertices, the partition, the affine transform coefficients, and the error images.

The MCGIMs were compressed by using both a JPEG[1] and a DIVX 3.11[2] encoders.

In the case of the JPEG compression, the individual frames are encoded independently (intra-coding mode). This approach enables a random access to the individual frames, without the need of decoding all the sequence. In addition, the progressivity of the JPEG encoder makes it possible to achieve scalable transmission.

When applying the DIVX encoder, such functionalities are no longer supported, but we expect to improve the compression rates by taking into account the temporal correlations between consecutive frames in the sequence.

Let us note that for the GV approach two different encoding modes have also been considered: I-mode (all frames encoded independently) and P-mode (the first frame encoded independently and the rest in prediction mode).

Figure 6 plots the RMSEs obtained for each frame of the "Dolphin" animation sequence at different bitrates. Here, the DIVX encoding offers a gain of 60% in terms of bitrates when compared to the JPEG independent frame encoding mode. This shows that the MCGIMs sequence presents important temporal correlations that can be efficiently exploited by a predictive compression mode.

Figure 7 illustrates the performances of MCGV, AFX-IC and D3DMC techniques for the "Humanoid" dynamic mesh. Here, at 2.2 bpfv, MCGV with DIVX encoding provides respectively 80% and 94% lower compression distortion than the D3DMC and AFX-IC approaches. The low compression performances of the AFX-IC technique are explained by the simplicity of the spatio-temporal predictor involved which is unable to fully eliminate the existing redundancies.

Figure 8 compares the performances of MCGV to those of D3DMC, AFX-IC, RT, and Dynapack for the "Chicken" sequence. Here, D3DMC and MCGV outperform all the other encoders for low bitrates (less than 4 bpfv). At 2.2 bpfv, MCGV with DIVX encoding achieves 30% lower compression distortion than D3DMC. This can be explained

[1] http://www.jpeg.org/
[2] http://www.divx.com/

by the non-optimality of the octree-based motion decomposition considered.

Figure 9 presents a comparison between the proposed MCGV approach and the GV encoder. At 8 bpfv and in intra-coding mode (JPEG compression for MCGV and exclusively I-frames for the GV approach), MCGV shows respectively 50%, 60%, and 10% lower compression distortions than GV for the animation sequences "Cow", "Dance" and "Snake". When predictive encoding of the animation is considered, MCGV achieves, for the "Dance" sequence, an average gain of 12% in term of bitrate over GV for the same RMSE distortion.

Finally, Figure 10 illustrates the rate/distortion performances for MCGV, RT and PCA approaches for the "Snake" sequence. The PCA approach leads here to the worst performances. This result is explained by the fact that the PCA technique is optimized for animation sequences with a number of frames much greater than the number of vertices ($F \gg V$). Such an assumption is obviously not satisfied by the "Snake" sequence ($F = 134, V = 9179$). Here again, for all bitrates, MCGV provides better compression performances than RT (*e.g.* 50% lower compression distortion at 4 bpfv). This proves that the DCT-based compression of the non-affine part of the motion is more efficient than the multiple rigid transform representation proposed by the RT approach.

Experiments establish that the proposed MCGV technique is particularly efficient when applied to articulated dynamic meshes such as "Dance" (60% lower distortions than GV) and "Humanoid" (94% and 80% lower distortions than AFX-IC and D3DMC) where the piecewise affine prediction stage is fully exploited.

5. Summary and Conclusion

This paper introduced a new compact representation for 3D dynamic meshes, based on a piecewise affine prediction scheme coupled with a MCGIM representation of the prediction residual errors.

Experimental results show that the proposed compression scheme outperforms the GV, D3DMC, RT, PCA, Dynapack, and AFX-IC approaches. The proposed MCGV technique proves to be particularly well-adapted for animated meshes with articulated motion.

Future work, will concern the optimization of the motion segmentation stage. In particular, we shall investigate how to optimally determine the number of clusters by tacking into account both anatomical and error bound constraints.

References

[1] M. Alexa and W. Müller. Representing animations by principal components. *Computer Graphics Forum*, 19:411–418, 2000.

[2] N. Aspert, D. Santa-Cruz, and T. Ebrahimi. Mesh: Measuring errors between surfaces using the hausdorff distance. *In Proc. of the IEEE International Conference in Multimedia and Expo (ICME)*, 1:705–708, 2002.

[3] M. Bourges-Sevenier and E. Jang. An introduction to the MPEG-4 animation framework extension. *In IEEE transactions on Circuits and Systems for Video Technology*, 14:928–936, 2004.

[4] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos: a new representation for 3D animations. *ACM Siggraph Symposium on Computer Animation*, pages 136–146, 2003.

[5] G. Collins and A. Hilton. A rigid transform basis for animation compression and level of detail. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 21–29, 2005.

[6] I. Guskov and A. Khodakovsky. Wavelet compression of parametrically coherent mesh sequences. *ACM Siggraph Symposium on Computer Animation*, pages 183–192, 2004.

[7] L. Ibarria and J. Rossignac. Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. *ACM Siggraph Symposium on Computer Animation*, pages 126–133, 2003.

[8] E. S. Jang, J. D. K. Kim, S. Y. Jung, M. J. Han, S. O. Woo, and S. J. Lee. Interpolator data compression for MPEG-4 animation. *In IEEE transactions on Circuits and Systems for Video Technology*, 14:989–1008, 2004.

[9] T. Kanungo, D. M. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 24:881–892, 2002.

[10] Z. Karni and C. Gotsman. Compression of soft-body animation sequences. *Computer Graphics*, 28:25–34, 2004.

[11] J. Lengyel. Compression of time-dependent geometry. *Proc. of ACM Symposium on Interactive 3D Graphics*, pages 89–96, 1999.

[12] K. Mamou, T. Zaharia, and F. Preteux. A preliminary evaluation of 3D mesh animation coding techniques. *Proc. SPIE Conference on Mathematical Methods in Pattern and Image Analysis*, 5916:44–55, 2005.

[13] K. Müller, A. Smolic, M. Kautzner, P. Eisert, and T. Wiegand. Predictive compression of dynamic 3d meshes. *Proc. of IEEE International Conference on Image Processing*, 1:621–624, 2005.

[14] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge Press, NY, USA, 2^{nd} edition, 1992.

[15] P. Sander, P. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parameterization. *Microsoft Research, MSR-TR-2002-27*, 2002.

[16] P. Sander, Z. Wood, S. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. *Eurographics Symposium on Geometry Processing*, pages 146–155, 2003.

[17] M. Sattler, R. Sarlette, and R. Klein. Simple and efficient compression of animation sequences. *ACM Siggraph Symposium on Computer Animation*, pages 209–217, 2005.

[18] C. Touma and C. Gotsman. Triangle mesh compression. *Proc. of Graphics Interface*, pages 26–34, 1998.

[19] J. Zhang and C. Owen. Octree-based animated geometry compression. *Proc. of IEEE Data Compression Conference*, pages 508–517, 2004.

[20] K. Zhou, J. Snyder, B. Guo, , and H. Y. Shum. Iso-charts: Stretch-driven mesh parameterization using spectral analysis. *Proc. Eurographics Symposium on Geometry Processing*, pages 45–54, 2004.

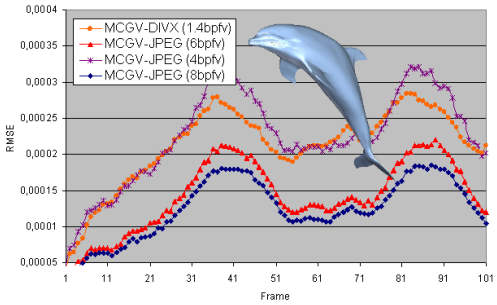


Figure 6. "Dolphin" sequence at different bitrates: DIVX predictive encoding leads to 60% lower bitrates than JPEG intra encoding mode.

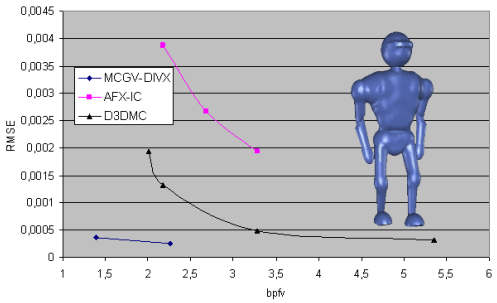


Figure 7. MCGV outperforms MPEG-4/AFX-IC and D3DMC for "Humanoid" sequence.

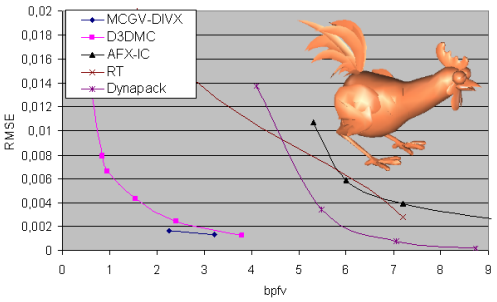
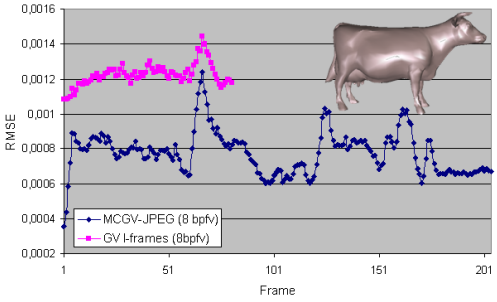
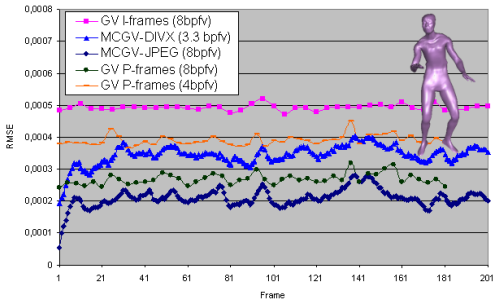


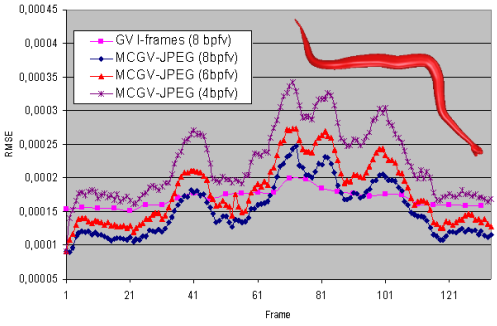
Figure 8. MCGV outperforms AFX-IC, D3DMC, RT and Dynapack for "Chicken" sequence.



(a) "Cow"



(b) "Dance"



(c) "Snake"

Figure 9. MCGV achieves up to 60% lower compression distorsions than GV.

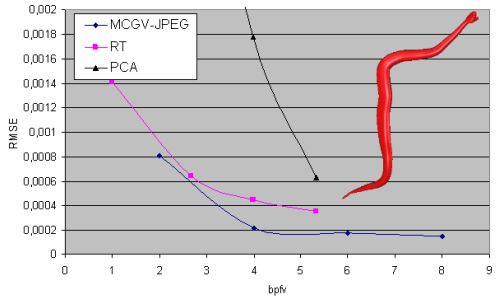


Figure 10. MCGV outperforms RT and PCA for the "Snake" sequence.