MOTION-BASED MESH CLUSTERING FOR MCDWT COMPRESSION OF 3D ANIMATED MESHES

Yasmine Boulfani-Cuisinaud, Marc Antonini and Frédéric Payan

Laboratoire I3S - UMR 6070 CNRS , Université de Nice - Sophia Antipolis
2000, Route des Lucioles - 06903 Sophia Antipolis FRANCE
phone: +33 4 92 94 27 85, fax: +33 4 92 94 28 98, email: {boulfani,am,fpayan}@i3s.unice.fr

ABSTRACT

We present an efficient compression algorithm for sequences of triangular meshes with fixed connectivity. Our two main contributions in this paper are : a clustering technique which regroups the vertices following the same affine motion ; a geometry compensation technique based on the estimation of 3D motion parameters which are modeled by affine transform matrices. Then, a scan-based temporal discrete wavelet transform is applied on the compensated sequence, and the resulting wavelet coefficients are finally encoded by an efficient coding scheme which includes a bit allocation process. Simulation results show that our compression method provides good compression performances compared to some state of the art coders.

1. INTRODUCTION

3D animations are widely used in a variety of fields, like computer games, multimedia, medical imaging... They are often represented by a sequence of triangular meshes, each mesh being defined by the location of the vertices (geometry) and by a list of triangles (connectivity). In general, the meshes are irregular, and the connectivity of the meshes may also change with time. However, in this paper, we restrict our attention to the animations defined by a sequence of meshes sharing the same connectivity at any frame.

Most of the first compression methods proposed in 3D animations exploit the affine transformations of different segments of the meshes [1, 2, 3, 4]. Also, some approaches proposed to predict the vertex displacements along the sequence and then to encode the residual errors [5, 6]. Recently, several more complex prediction techniques have been proposed. In [7, 8, 9] for instance, the authors exploited the temporal coherence by clustering vertices with similar affine transforms between successive frames. In [9], Mamou and *al*. proposed a novel approach for 3D mesh compression based on a skinning animation technique. Their method is based on a piecewise affine predictor coupled with a skinning model and a DCT representation of the residuals errors. In [10], the authors used the mesh connectivity to determine the order of compression of vertex locations. In parallel, Alexa and Müller [11], proposed a coding scheme based on the Principal Component Analysis (*PCA*) to represent the mesh sequences with only a small number of basis functions. Karni and Gotsman improved this method by further exploiting the temporal coherence and finally encode the PCA coefficients with a second-order predictive coding called LPC [12]. The method of [11] has been also improved in [13], where the authors proposed to cluster vertices before using PCA. Briceno *et al.* presented an original approach in [14]. The technique is to project each frame onto a 2D image, and then to encode the resulting sequence of "2D images" with some well-known video techniques. Besides, several methods based on wavelets have been proposed. In [15], the authors proposed to exploit the temporal coherence of the geometry components by applying a B-spline wavelet transform on the successive vertex positions. Recently, a coder based on temporal wavelet filtering implemented in lifting scheme has been proposed [16, 17]. In [18], Guskov and Khodakovsky proposed to exploit the *parametric* coherence of some specific animations by combining a spatial multiresolution analysis and a predictive coding scheme based on a I-frames/P-frames approach (similar to some video compression methods). In parallel, J.H. Yang *et al.* also proposed a wavelet-based algorithm, but for sequences of irregular meshes with changing connectivity [19].

In this paper we propose an alternative way to compress 3D animations in the framework of MCDWT (motion compensation for discrete wavelet transform). In order to exploit the temporal coherence of the geometry of an animation, we propose a coding scheme combining a motion-based clustering for the estimation/compensation of the vertex displacements, and a temporal motion compensated wavelet transform (see figure 1). In our compression algorithm, the animation is processed in 4 steps: i) a clustering technique is used to regroup the neighbor vertices having the same affine motion; ii) the motion of each cluster is estimated in order to compensate the vertex displacements during the animation; iii) a scan-based temporal DWT is applied on the compensated animation. iv) the resulting wavelet coefficients are finally encoded with a coding scheme which includes a bit allocation process.

The rest of this paper is organized as follows. Section 2 presents the motion-based clustering approach and the scan-based processing. Section 3 presents the proposed motion estimation and the geometry compensation techniques. In section 4 we present thescan-based temporal DWT. Experimental results are given in section 5 and compared to results of some state of the art methods. We finally conclude and propose future works in section 6.

2. MOTION-BASED MESH CLUSTERING

The first step of our algorithm consists in the clustering of the meshes. Our clustering technique is based on the motion, *i.e.*, vertices with similar motion belong to the same cluster. The number of clusters obtained by this technique depends on the motion nature between the two frames.

2.1 Notations

Let us define $\mathscr{F}$ the sequence of T meshes:

$$\mathscr{F} = \{f_1, f_2, ..., f_t, ...f_T\}. \tag{1}$$

Let us also define $\mathscr{V}_t$ the set of V vertices at each frame t of the sequence[1] by :

$$\mathscr{V}_t = \{v_t^1, v_t^2, ..., v_t^i, ..., v_t^V\},\ \forall t, \tag{2}$$

and $\overline{\mathscr{V}}_i$ the set of neighbor vertices of v_t^i (neighbor vertices which form the same connex component). The set $\overline{\mathscr{V}}_i$ is identical for all frame t of the sequence (with fixed connectivity) and is written as:

$$\overline{\mathscr{V}}_i = \{(\overline{v}_t^1)_i, (\overline{v}_t^2)_i, ..., (\overline{v}_t^k)_i, ..., (\overline{v}_t^K)_i\},\ \forall t, \tag{3}$$

[1]Since the connectivity remains the same for all the frames of the sequence, the number V is constant whatever t.

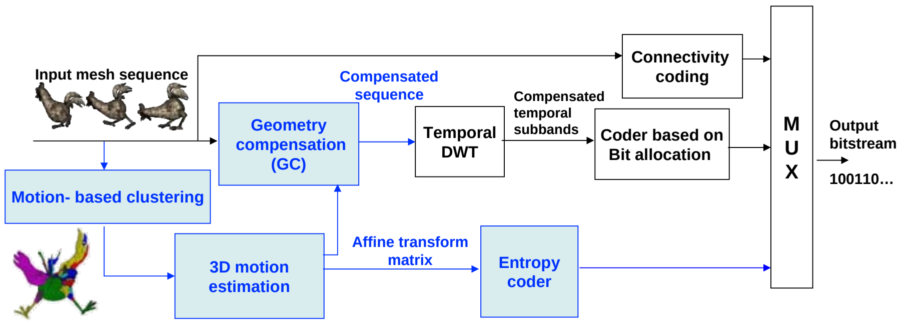


Figure 1: General structure of the proposed compression algorithm.

where K is the number of the neighbor vertices of v_t^i, such as $K \leq V$. Finally, the clustering for each frame t is denoted by:

$$\mathscr{C}_t = \{C_t^1, C_t^2, ..., C_t^n, ..., C_t^N\}, \tag{4}$$

with N the number of clusters at the frame t which depends on the motion between the frame t and the frame $t-1$.

2.2 Clustering Principle

The proposed technique creates a clustering of each connex component of the animation. The principle of the method is the following. We consider that a vertex at time t is related with a vertex at time $t-1$ by:

$$v^i{}_t = M_t^i * v^i{}_{t-1}, \; \forall v^i{}_t \in \mathscr{V}_t, \tag{5}$$

where the vertices are given in homogeneous coordinates[2]. M_t^i is the affine transform matrix. This matrix is of size (4×4) and contains 12 coefficients representing the motion (rotation, translation and scaling):

$$M_t^{v_i} = \begin{pmatrix} & [R,S]_{3\times3} & & [Tr]_{3\times1} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{6}$$

where the matrix $[R,S]_{3\times3}$ contains the rotation and scaling information, and $[Tr]_{3\times1}$ is the translation vector.
In order to solve the equation (5), we introduce the matrix $\mathscr{P}_t^i$ of size $(4 \times (Q+1))$ containing the coordinates of the vertex v_t^i and the coordinates of the Q vertices of its first order neighborhood ($Q \leq K \leq V$). So, $\mathscr{P}_t^i$ is given by:

$$\mathscr{P}_t^i = \begin{pmatrix} v_t^i & (\overline{v}_t^1)_i & \dots & (\overline{v}_t^q)_i & \dots & (\overline{v}_t^Q)_i \end{pmatrix} \tag{7}$$

where v_t^i and $(\overline{v}_t^q)_i$ are to be column vectors given in homogeneous coordinates.
The affine transform M_t^i is then computed as follows:

$$M_t^i = \mathscr{P}_t^i * (\mathscr{P}_{t-1}^i)^+ \tag{8}$$

where $(.)^+$ stands for the pseudo-inverse operator.

A cluster $C_t^n \in \mathscr{C}_t$ of the frame t, represented by the vertex v_t^i, will contain the vertex v_t^i and all of its neighbor vertices $(\overline{v}_t^k)_i$ among $\overline{\mathscr{V}}_i$ verifying:

$$\|M_t^i * (\overline{v}_{t-1}^k)_i - (\overline{v}_t^k)_i\|^2 < \varepsilon, \tag{9}$$

where ε is a threshold. The resulting partition verifies the following properties:

$$\cup C_t^n = f_t \;\text{ and }\; C_t^n \cap C_t^m = \emptyset, \; \forall m,n \tag{10}$$

[2]A vertex v given in homogeneous coordinates is written as $v^t = (x,y,z,1)$.

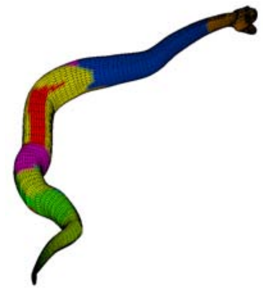


Figure 2: Clustering of the SNAKE sequence, $N = 23$ clusters.

In other way, the cluster C_t^n can be written as:

$$C_t^n = \{(\overline{v}_t^j)_n\}_{j=1,...,V'} \tag{11}$$

where the quantity V' ($V' \leq K \leq V$) is the number of vertices in the cluster C_t^n which verify the equation (9).

2.3 Scan-based processing

In the proposed approach, the sequence is processed on the fly (also called scan-based processing), meaning that the animation is processed by Groups Of Frames (denoted by GOF), treated and computed independently.
In the rest of the paper, we consider that a GOF is composed by T frames. Also, the number N of clusters per frame is supposed to be the same in each frame of the current GOF. To simplify, the clustering process is done for the first two frames of each GOF of the sequence and is maintained constant in the whole GOF.
As example, the clustering of the 7^{th} frame of the SNAKE sequence is shown in the figure 2, for $N = 23$ clusters.

3. MOTION ESTIMATION AND GEOMETRY COMPENSATION

3.1 Estimation of 3D Motion Parameters

Once the partition $\mathscr{C}_t$ is computed, we associate to each cluster $C_t^n \in \mathscr{C}_t$ an affine transform noted here M_t^n, defined as in section (2.2) by the formula (6). The estimation of the affine transform is done for each cluster between the frame t of the GOF ($t = 2, ..., T+1$) and the same cluster of the first frame (key frame) corresponding to $t = 1$ (see figure 3), such that:

$$(\overline{v}^j{}_t)_n = M_t^n * (\overline{v}^j{}_1)_n, \; \forall (\overline{v}^j{}_t)_n \in C_t^n, \tag{12}$$

The proposed motion estimation consists in computing a set of N affine transforms M_t^n per frame t, corresponding to the N clusters in

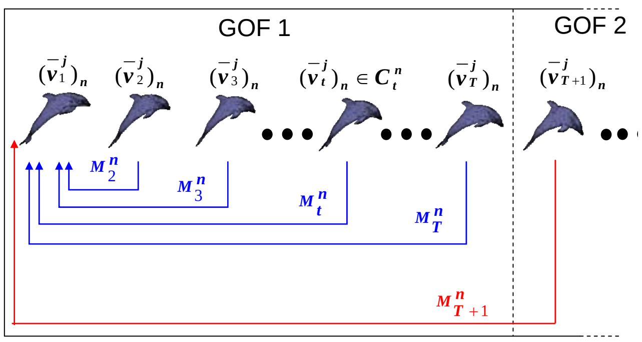


Figure 3: Estimation of the affine motion. To compute the scan-based wavelet transform (Section 4) and avoid boundary effects, we need to use the first frame of the next GOF, which is also estimated according to the first frame of the current GOF.

the frame t. Each affine transform M^n_t can be computed as previously (formula (8)) by:

$$M^n_t = \mathscr{Q}^n_t * (\mathscr{Q}^n_1)^+ \qquad (13)$$

with $(.)^+$ the pseudo-inverse operator. $\mathscr{Q}^n_t$ is now a matrix of size $(4 \times V')$ including the set of vertices $(\bar{v}^j{}_t)_n \in C^n_t$, it is given by:

$$\mathscr{Q}^n_t = \begin{pmatrix} (\bar{v}^1_t)_n & (\bar{v}^2_t)_n & \dots & (\bar{v}^j_t)_n & \dots & (\bar{v}^{V'}_t)_n \end{pmatrix} \qquad (14)$$

with $(\bar{v}^j_t)_n$ a column vector.
Once the affine transform M^n_t is computed for each frame t ($t = 2,\dots,T+1$) and for each cluster C^n_t ($n = 1,\dots,N$), the whole affine transforms are entropy coded without loss with an arithmetic coder [20].

3.2 Geometry Compensation (GC)

The geometry compensation, denoted by GC in the rest of the paper, consists to displace all the frames of a given GOF on the first frame (key frame) of this GOF, while keeping this key frame unchanged. In other words, once the affine motion M^n_t is computed for each frame t of the GOF and for each cluster C^n_t according to the key frame, we compensate this motion and we reconstruct a new GOF which contains all the frames compensated on the first one.
Let us denote the new compensated cluster by:

$$\hat{C}^n_t = \{(\hat{\bar{v}}^j_t)_n\}_{j=1,\dots,V'} \qquad (15)$$

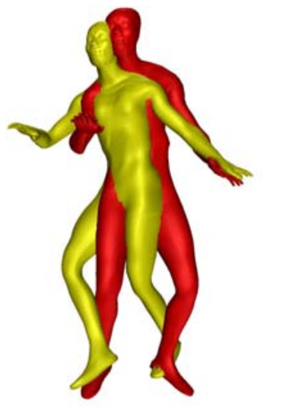
Each compensated vertex $(\hat{\bar{v}}^j_t)_n \in \hat{C}^n_t$ is computed as follows:

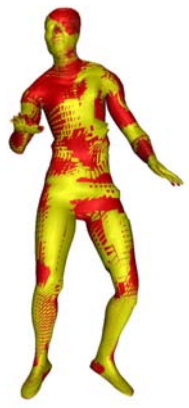
$$(\hat{\bar{v}}^j_t)_n = (M^n_t)^- * (\bar{v}^j_t)_n, \qquad (16)$$

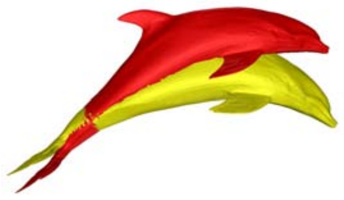
for all $t = 2,\dots T+1$, $n = 1,\dots,N$ and $j = 1,\dots,V'$, where $(.)^-$ is the inverse operator.
To show the efficiency of our motion estimation/compensation technique, the figure 4 compares the position of the last frame (in red) of a GOF (size of the GOF = 8) in function of the first frame (in yellow), with compensation or without compensation . We can observe the motion is well compensated for the three animations.
The temporal wavelet transform is then done on the compensated sequence as explained in the following section 4.

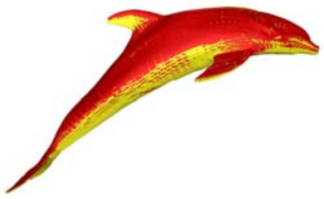
4. GEOMETRY COMPENSATED DWT

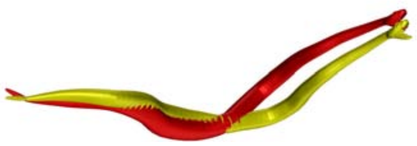
A practical problem in temporal wavelet transform implementation is related to memory requirements of time filters. Usually, wavelet transform is implemented by loading all the data in memory and then performing filtering. In the case of temporal filtering of mesh


(a) Without compensation.


(b) With compensation.


(c) Without compensation.


(d) With compensation.


(e) Without compensation.

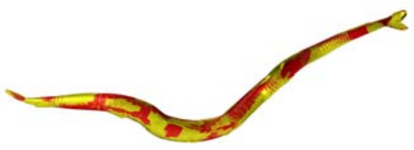

(f) With compensation.

Figure 4: We compare the position of the last frame (in red) of a GOF (size of the GOF = 8) in function of the first frame (in yellow), with compensation or without compensation. We can observe the motion is well compensated for these three animations.

sequences, this would require a huge memory size and moreover could imply an encoding delay as long as the sequence duration itself. As said in Section 2.3, a simple solution to the temporal filtering problem is to crop the input sequence in several short subsequences called GOF (Groups Of Frames) and then compute filtering for each GOF independently.

Once each GOF is geometry compensated, we can apply the scan-based wavelet transform on the compensated sequence. For this, we use the lifting scheme which is an efficient way to implement wavelet transforms. An interesting property of the lifting implementation is that for each transversal implementation of wavelet filter, it exists an equivalent lifting implementation. So, it is reversible, scalable and still assure perfect reconstruction.

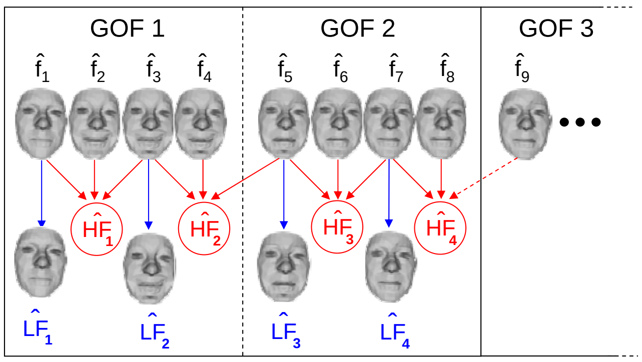

GOF 1
GOF 2
GOF 3
$\hat{f}_1$ $\hat{f}_2$ $\hat{f}_3$ $\hat{f}_4$ $\hat{f}_5$ $\hat{f}_6$ $\hat{f}_7$ $\hat{f}_8$ $\hat{f}_9$
$\hat{HF}_1$ $\hat{HF}_2$ $\hat{HF}_3$ $\hat{HF}_4$
$\hat{LF}_1$ $\hat{LF}_2$ $\hat{LF}_3$ $\hat{LF}_4$

Figure 5: Example of a scan-based wavelet transform decomposition (lifting scheme $(2,0)$ - 1 level decomposition)

Figure 5 shows the principle of the scan-based filtering on the FACE sequence, for a GOF of 4 frames, by using the lifting scheme $(2,0)$ proposed in [17]. Each GOF is independently filtered and in order to solve the boundary problem, we need to consider the first frame of next GOF to compute the filtering of the last frame of previous GOF. The scan-based approach allows to overcome this problem, by computing the temporal transform as it would be by considering all the sequence as a whole, but by keeping in memory only data necessary to compute the wavelet transform on one GOF. Also, a special treatment is done for the boundary effects of the last GOF of the sequence.

The high frequency (HF) details and the low frequency (LF) sequence of each GOF are encoded with an efficient bit allocation-based coder [17].
As the connectivity remains the same for all frames of the mesh sequence, we simply encode the connectivity of the first frame with the efficient coder of Touma and Gotsman [21].

5. SIMULATION RESULTS

5.1 Distortion Error Criterion

To evaluate the quality between the original sequences and the reconstructed ones, we use the metric error called *KG error*, introduced by Karni and Gotsman in [12]. This metric corresponds to the relative discrete L_2-norm both in time and space and is expressed in percent. It is given by:

$$KG\ error = 100\frac{||G-\hat{G}||}{||G-E(G)||} \qquad (17)$$

where G is a matrix of dimension $(3V,T)$ containing the geometry of the original sequence, $\hat{G}$ the quantized version of the geometry, and $E(G)$ an average matrix in which the t^{th} column is defined by:

$$\left(\bar{X}_t\,(1\,...\,1)\,,\bar{Y}_t\,(1\,...\,1)\,,\bar{Z}_t\,(1\,...\,1)\right)^T$$

with $\bar{X}_t$, $\bar{Y}_t$, and $\bar{Z}_t$ the mean values of the coordinate sets of each frame t.

5.2 Experimental Results

We have tested the efficiency of our coder on different sequences. Here, we present three animation sequences: DOLPHIN, SNAKE and DANCE, with different features presented in Table 1. Where T the number of frames, V the number of vertices per frame, K the number of the connex components of the mesh, and N is the average number of the clusters obtained for the whole GOFs of the sequence.

Sequence	T	V	K	N
SNAKE	128	9179	1	35
DANCE	200	7061	1	36
DOLPHIN	64	6179	1	28

Table 1: Considered features of the used sequences.

Figures 6, 7 and 8 show the curves *KG Error/bitrate* for the three different sequences, using the proposed coder for different GOF sizes. The results are compared to the Wavelet-based coder proposed in [17] (without clustering and without GC). Also, we compare the coding performances of the proposed coder with the PCA-based coder of [11] , the CPCA- based coder of [13] and the Skinning-based coder of [9]. We choose different GOF sizes of 8 frames, 64 frames (DOLPHIN), 128 frames (SNAKE) and 200 frames (DANCE). The bitrate is given in bits per vertex per frame.

For the different figures 6, 7 and 8, we observe that the best results are given by the Skinning-based coder of [9]. However, the three cases of the proposed coder provides best results than Wavelet-based coder of [17] and PCA-based coder (for figure 6). Also, for the figure 6 the proposed coder using the lifting scheme $(4,2)$ is better than the CPCA-based coder of [13] (at high bitrates). Nevertheless, the proposed coder using scan-based processing for a GOF of size 8 frames is interesting when huge sequences must be processed. Indeed, a large sequence requires a huge memory and moreover could imply an encoding delay as long as the sequence duration itself.

6. CONCLUSIONS AND FUTURE WORKS

We have presented in this paper a new method to encode the 3D mesh sequences with fixed connectivity. We proposed an original approach to construct a partition of the geometry of the mesh sequence based on the motion of the sequence. Also, we proposed to compensate the geometry of the sequence using the estimation of its 3D motion. Our compression scheme uses a scan-based wavelet transform applied on the compensated sequence which requires low memory. Experimentally, we have shown that using motion estimation and GC before the lifting steps reduces the energy of the wavelet coefficients and improves the efficiency of the coder when small GOF size are used. First experimental results are promising.
For our future research, we are improving the clustering method as well as the proposed motion estimation and GC approach.

REFERENCES

[1] J. E. Lengyel, "Compression of time-dependent geometry," in *ACM Symposium on Interactive 3D Graphics*, 1999, pp. 89–96.

[2] A. Shamir and V. Pascucci, "Temporal and spatial level of details for dynamic meshes," in *Proceedings of Virtual Reality Systems and Techniques*, 2001, pp. 423–430.

[3] J.H. Ahn, C.S. Kim, C.C. Jay Kuo, and Y.S. Ho, "Motion compensated compression of 3D animation models," *IEEE Electronics Letters*, vol. 37, no. 24, pp. 1445–1446, Nov 2001.

[4] S.Gupta, K.Sengupta, and A. Kassim, "Registration and partitioning-based compression of 3d dynamic data," *IEEE Trans. on Circuits Syst. Video Techn.*, vol. 13, no. 11, pp. 1144–1155, 2003.

[5] J.H. Yang, C.S. Kim, and S.-U. Lee, "Compression of 3D triangle mesh sequences based on vertex-wise motion vector prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1178–1184, Dec. 2002.

[6] L. Ibarria and J. Rossignac, "Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity," *ACM Symp. Computer Animation*, pp. 126–135, 2003.

[7] Karsten Muller, Aljoscha Smolic, Matthias Kautzner, and Thomas Wiegand, "Rate-distortion optimization in dynamic mesh compression," in *ICIP*, 2006.

[8] J. Zhang and C.B. Owen, "Hybrid coding for animated polygonal meshes: Combining delta and octree," in *International Conference on Information Technology: Coding and Computing*, 2005, pp. 68–73.

[9] K. Mamou, T. Zaharia, and F. Prêteux, "A skinning approach for dynamic 3d mesh compression," *Computer Animation and Virtual Worlds*, vol. 17, no. 3-4, pp. 337–346, July 2006.

[10] Nikolce Stefanoski and Joern Ostermann, "Connectivity-guided predictive compression of dynamic 3d meshes," in *Proc. of ICIP*, october 2006.

[11] M. Alexa and W. Muller, "Representing animations by principal components," *Comput. Graph. Forum 19*, vol. 3, 2000.

[12] Z. Karni and C. Gotsman, "Compression of soft-body animation sequences," *Computers and Graphics*, vol. 28, pp. 25–34, 2004.

[13] Mirko Sattler, Ralf Sarlette, and Reinhard Klein, "Simple and efficient compression of animation sequences," in *Proc. of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2005, pp. 209–217, ACM Press.

[14] H.M. Briceno, P.V. Sander, L. McMillan, S. Gotler, and H. Hoppe, "Geometry videos: a new representation for 3D animations," *ACM Symp. Computer Animation*, pp. 136–146, 2003.

[15] A.C. Lopes and M.N. Gamito, "Wavelet compression and transmission of deformable surfaces over networks," in *Proc. of the 10th Portuguese Computer Graphics Meeting*, 2001, pp. 107–114.

[16] F. Payan, Y. Boulfani, and M. Antonini, "Temporal lifting scheme for the compression of animated sequences of meshes," in *Proceedings of IEEE International Workshop VLBV 2005*, Sardinia, Italy, september 2005.

[17] F. Payan and M. Antonini, "Temporal wavelet-based geometry coder for 3d animations," *Computer & Graphics, In Press, Available Online*, 2006.

[18] I. Guskov and A. Khodakovsky, "Wavelet compression of parametrically coherent mesh sequences," in *Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, August 2004.

[19] J-H. Yang, C-S. Kim, and S-U. Lee, "Progressive compression of 3D dynamic sequences," in *Proc. ICIP 2004*, Oct. 2004.

[20] Ian H. Witten, Radford M. Neal, and John G. Cleary, "Arithmetic coding for data compression," New York, NY, USA, 1987, vol. 30, pp. 520–540, ACM Press.

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

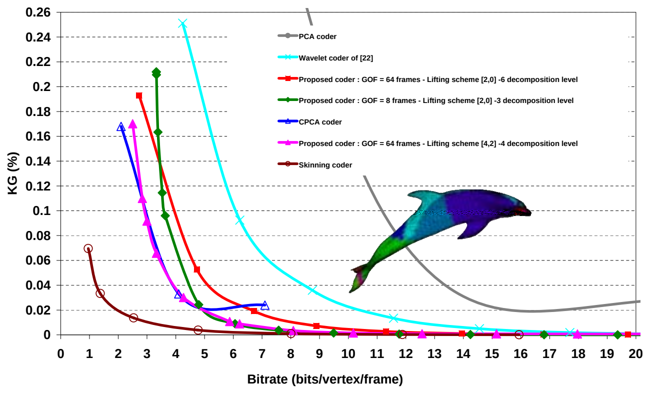


Figure 6: *KG Error/bitrate* for DOLPHIN relative to different compression methods.

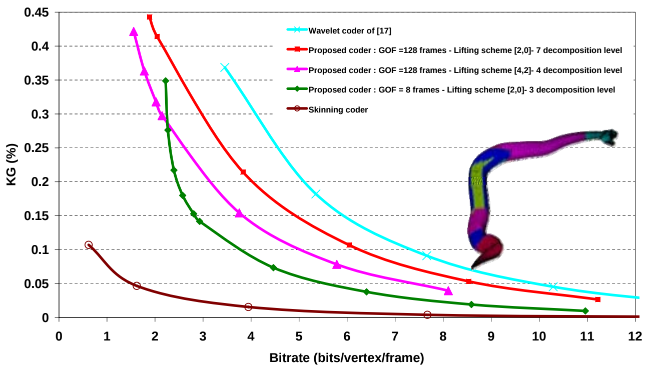


Figure 7: *KG Error/bitrate* for SNAKE relative to different compression methods.

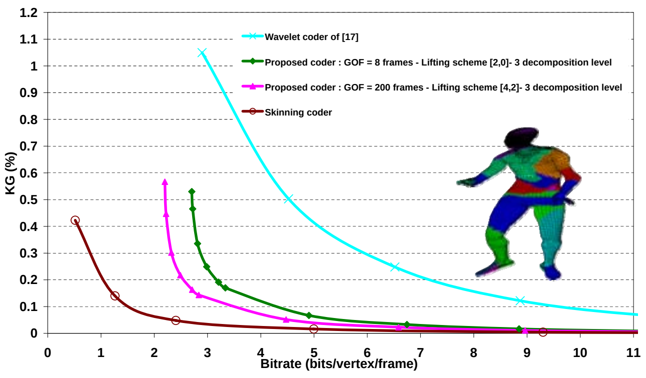


Figure 8: *KG Error/bitrate* for DANCE relative to different compression methods.