A Preliminary Evaluation of 3D Mesh Animation Coding Techniques

Khaled Mamou Titus Zaharia Françoise Prêteux

Groupe des Ecoles des Télécommunications
Institut National des Télécommunications / ARTEMIS Project Unit
9 rue Charles Fourier 91011 EVRY, France

<Khaled Mamou, Titus.Zaharia, Françoise.Preteux>@int-evry.fr

ABSTRACT

This paper provides an overview of the state-of-the-art techniques recently developed within the emerging field of dynamic mesh compression. Static encoders, wavelet-based schemes, PCA-based approaches, differential temporal and spatio-temporal predictive techniques, and clustering-based representations are considered, presented, analyzed, and objectively compared in terms of compression efficiency, algorithmic and computational aspects and offered functionalities (such as progressive transmission, scalable rendering, computational and algorithmic aspects, field of applicability...).

The proposed comparative study reveals that: (1) clustering-based approaches offer the best compromise between compression performances and computational complexity; (2) PCA-based representations are highly efficient on long animated sequences (*i.e.* with number of mesh vertices much smaller than the number of frames) at the price of prohibitive computational complexity of the encoding process; (3) Spatio-temporal Dynapack predictors provides simple yet effective predictive schemes that outperforms simple predictors such as those considered within the interpolator compression node adopted by the MPEG-4 within the AFX standard; (4) Wavelet-based approaches, which provide the best compression performances for static meshes show here again good results, with the additional advantage of a fully progressive representation, but suffer from an applicability limited to large meshes with at least several thousands of vertices per connected component.

1. INTRODUCTION

Dynamic 3D content becomes a more and more present feature within nowadays multimedia applications, extensively exploited within the world of games, virtual and augmented reality systems, industrial simulation, and 3D CGI (*Computer Generated Imagery*) animation films that recently have known a world-wide success.

Within this context, the new economic challenges concern the elaboration and the seamless integration of efficient 3D representation technologies. Besides the traditional compression efficiency requirement, such dynamic 3D representations should enable new functionalities, such as real-time visualization on multiple terminals, progressive transmission over various networks, and scalable rendering with multiple Levels of Detail (LODs).

The progressive transmission functionality concerns the bitstream adaptation to different, fixed or mobile communication networks with various bandwidths. Here, the decoder can start displaying a coarse approximation of the animation sequence when some baseline, minimal information is received. A refinement process is then applied to this coarse representation in order to obtain progressively finer LODs and gradually improve the visual quality of the decoded animation sequence.

The scalable rendering functionality concerns the bitstream adaptation to terminals (*e.g.*, desktop computers, laptops, PDAs, mobilephones…) of various complexities with different memory and computational capacities, under real-time visualization constrains. Here, decimation-based approaches[15] are generally adopted in order to obtain simplified representation that are visually close to the original and which can be rendered at a low computation time.

Within this challenging applicative framework, the issue of elaborating efficient compression methodologies for memory consuming 3D animated meshes becomes of a crucial importance.

The bone-based animation (BBA) techniques adopted by the MPEG-4 AFX (*Animation Framework eXtension*) standard[28] provide an intuitive and compact representation, expressed in terms of animation parameters associated with a hierarchical structure of a skeleton. This highly efficient representation has the drawback of a limited applicability: complex and non rigid motions are difficult to represent with bones. In addition, the BBA technology imposes some strong constraints on the 3D object topology (*e.g.* seamless meshes made of a unique connected component) and requires a huge amount of human interaction for manually defining adequate animation parameters. In addition, content creators are reluctant to the BBA technology due to some intellectual property and content protection issues. In fact, the animation content can be easily re-produced and used for others meshes with identical skeletons, since by definition the animation parameters are mesh-independent.

In order to overcome the above-mentioned drawbacks, a key-frame representation is generally adopted in the animation industry. Here, the animation sequence is stored as a set of consecutive frames represented as a set of 3D meshes. The in-betweening frames are derived by using an interpolation procedure.

Key-frame representations can model generic animations and handle the content protection problem. However, such representations lead to a huge amount of dynamic 3D data, since even for short sequences of a few minutes, several thousands of key models are requested. In order to ensure the efficient storage, transmission and visualization of key-frame representations, efficient compression techniques need to be elaborated and developed.
This research issue has been for the first time considered in[22], where Lengyel introduced the concept of *time-dependent geometry compression*, detailed here below.

Let us consider a sequence of meshes $\left(M_i\right)_{i\in\{0,\ldots,F-1\}}$ (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 $G_i=(X_i,Y_i,Z_i)^t$, at instant t_i, is represented as a vector of size $3\times V$ (with V being the number of vertices) and includes the three *x*, *y* and *z* coordinates of all mesh vertices. The geometry of the object $G(t)$ at time t is obtained by linearly interpolating between the key-frames $(G_i)_i$ as described in Equation (1):

$$\forall t\in[t_i,t_{i+1}],\ G(t)=\frac{t_{i+1}-t}{t_{i+1}-t_i}\times G_i+\frac{t-t_i}{t_{i+1}-t_i}\times G_{i+1}. \tag{1}$$

The goal of time-dependent geometry compression is to elaborate adequate and compact representations of the $(G_i)_{i\in\{0,\ldots,F-1\}}$ geometry sequence, taking into account both spatial and temporal correlations. Let us note that the topology of the mesh being fixed all along the sequence, it can be coded only once for all meshes with a small overhead by applying standard static mesh compression encoders. The connectivity compression aspects will not be treated further in this paper.

The issue of efficient compression of animated meshes has gained rapidly the interest of the scientific community as testifies the important number of recently developed works reported in the literature. This paper aims at giving an overview of the most recent animated mesh compression techniques. The paper is organized as follows. Section 2 presents the state of the art in the field of dynamic mesh compression. The most representative approaches, with underlying principles, computational and algorithmic-related aspects are here described and analyzed in details. Section 3 provides a comparative evaluation of the most promising approaches. The compression performances of the considered techniques are extensively discussed and objectively compared. Finally, Section 4 concludes the paper and opens perspectives of future work.

2. STATE OF THE ART

In this section, we review the most promising techniques for dynamic 3D mesh compression recently developed in the literature. The underlying principles of the different approaches are presented and their advantages and limitations are analyzed and discussed.

The first family of approaches concerns the differential predictive coding techniques.

2.1 Vertex-based predictive techniques

Here, the animation sequence is processed locally in space and time. At each stage of the compression process, only a limited number of previous frames (generally the last encoded frame) is involved when encoding the current frame, within the framework of a predictive scheme. A traversal order of the mesh vertices (*e.g.* such as the ones considered by mono-resolution connectivity encoders) is supposed to be available.

The Interpolation Compression (IC) scheme[27] recently adopted by the MPEG-4 AFX proposes an interpolation procedure aiming at exploiting redundancies between successive key-frames. The principle consists of reducing the amount of data by sub-sampling the initial sequence of key-frames that are subsequently differentially encoded.
Three simple prediction modes are currently supported, which involves spatial, temporal, and spatio-temporal predictors respectively described in (2), (3) and (4). The decoder restores the key-frames and generates the intermediate frames by applying a linear interpolation procedure. Different strategies can be applied for selecting the key-frames, starting from simple uniform sampling techniques, and up to more sophisticated procedure aiming to determine an optimal set of key-frames, minimizing an error criterion over the whole set of frames in the sequence. Such an approach is discussed in[27], where starting from the initial and final frames, a refinement procedure is iteratively applied while the error score becomes lower than a given threshold.

$$v_i^j = v_i^k + r_i^j , \tag{2}$$

$$v_i^j = v_{i-1}^j + r_i^j , \tag{3}$$

$$v_i^j = v_{i-1}^j + (v_i^k - v_{i-1}^k) + r_i^j . \tag{4}$$

Here, v_i^j is the position of vertex j at the instant i, r_i^j its prediction error, and k is the index of a previously decoded vertex. Such prediction rules are obviously too elementary, but offers the advantage of a low computational complexity.

The Dynapack approach[5] exploits a similar prediction with more elaborate local spatio-temporal predictors. Here, the traversal of the mesh vertices is guided by the connectivity compression scheme described in[20]. Authors propose two different predictors, so-called Extended Lorenzo Predictor (ELP) and Replica. The ELP generalizes the Lorenzo predictor introduced in[21] and extends the "parallelogram" prediction rule considered for static mesh compression[12] to the dynamic case. The ELP perfectly predicts (*i.e.*, with null residuals) translations and can be computed with low complexity (only additions and subtractions operations are needed). The Replica predictor further enhances this scheme and is able of perfectly predicting rigid body motions and uniform scaling transformations.
Predictive approaches offer the advantage of simplicity and low computational cost, which makes them adapted for real-time decoding applications. However, being based on a deterministic traversal of mesh vertices, such mono-resolution approaches do not support more advanced functionalities such as progressive transmission and scalable rendering. This drawback is overcome by the multi-resolution representations, based on the wavelets techniques described in the next section.

2.2 Wavelet-based approaches

Wavelets techniques have been extensively used for still image and video progressive compression applications with impressive compression performances. However, the extension of wavelet basis functions, usually defined on regular lattices, to 3D meshes of arbitrary topology is not straightforward.
For this reason, the first wavelet-based 3D mesh representations, described in the first section, modifies the original irregular topology in order to obtain regular or semi-regular connectivities adapted for wavelet construction.

2.2.1 Re-meshing based approaches

The re-meshing based techniques[14,16,17] lead to the best compression performances for static mesh encoding. Such approaches discard totally the original irregular connectivity and re-sample the mesh surface in order to obtain a semi-regular or regular topology well-suited to a wavelet construction. Beside the high compression performances, the wavelet techniques naturally support progressivity and scalability.
In[9], authors extend the regular re-meshing based technique[16] to the compression of animation sequences.
The approach presented in[16] includes the following steps: (1) construction of a mesh parameterization, defined on the unit square $[0,1]\times[0,1]$; (2) uniform re-sampling of the parametric domain and construction of so-called geometry

images which store the *x*, *y* and *z* vertex coordinates as the *R*, *G* and *B* planes of a color image; (3) encoding of the geometry image with a 2D traditional wavelet encoder .
The extension of this technique to animated meshes is straightforward: a geometry image is derived for each frame. A video sequence is thus constructed, and wavelet encoded. Two encoding modes *intra* and *predicted* are defined and applied for each frame. In the intra-mode, the frame (so called *I*-frame) is encoded independently, as a still image, without any reference to other frames. In the predicted mode, the prediction errors between current (so called *P*-frame) and previous frame are feeding the wavelet encoder.
The *I*-frame enables a random access to the compressed animation stream while the *P*-frames ensure the temporal decorrelation and thus the compression efficiency.
Let us note that defining a unit square parameterization is possible only for 0-genus meshes, homeomorphic to an open disc. In order to deal with meshes of arbitrary topologies, an optimized cutting procedure[16] is applied.
Minimizing the parameterization distortions is essential for guarantying the quality of the compressed meshes. Authors apply the algorithm described in [16] in order to minimize the geometric stretch metric[18] of the parameterization[19] over the set of all meshes in the sequence.
Re-meshing-based approaches are particularly well-suited for very low bitrate compression. However, their main drawbacks refer to:

(1) the high computational complexity of the encoding stage, determined by the cutting and the parameterization algorithms;
(2) the bad triangulation of the obtained re-sampled meshes in the case of high distortion parameterization; and
(3) the possible apparition of surface's tangent plane discontinuities at the level of the cut border.

In order to overcome this limitation, irregular wavelet techniques, described in the next section are preserving the original mesh connectivity.

2.2.2 Irregular wavelet-based techniques

Irregular wavelets[10] defined for meshes of arbitrary topology have been poorly used since the set of parameters associated with such adaptive wavelets (values of the wavelet filter coefficient) need to be sent to the decoder as payload, which is expensive in term of bitrate. However, in the case of animated meshes, the parameterization information is shared by all the meshes in the sequence and can be transmitted only once for the first frame. In[4], Guskov and Khodakovsky exploit this idea in order to compress mesh sequences with irregular and anisotropic wavelet transform.
The *Wavelets Animation Compression* (AWC) approach[4] constructs first a Progressive Mesh (PM) hierarchy[15], defined on the first mesh M_0 of the sequence. The simplification process involves uniquely a half-edge collapse[15] decimation operator. Then, an anisotropic wavelet transform is constructed on the top of this PM structure. Starting from the basemesh, a refinement process is applied which consists of re-inserting the previously eliminated vertices until the initial connectivity is completely recovered. For each newly inserted vertex v at LOD j, an optimized, anisotropic wavelet filter Φ_j is computed. A relaxation procedure, which aims at minimizing a fairing functional defined over the vertex *v* star-like neighborhood[10], is here applied. The so-determined PM hierarchy and wavelets filters are finally used for computing the wavelets detail coefficients $\left(d_i^j\right)_{\substack{j\in\{1,\ldots,n\}\\ i\in\{1,\ldots,F-1\}}}$ for all the frames in the sequence (n stands for the number of LODs). Each detail d_i^j records the difference between the actual vertex position at frame i and its anisotropically predicted position from the LOD $j-1$.

The animation stream is encoded as two types of frames: I-frames and P-frames. The I-frames are encoded using only spatial wavelet transform and thus enable random access to the animation sequence. The P-frames linearly predict the wavelets details from the previous frames in order to capture the temporal correlations. The so-obtained coefficients are quantized and progressively sent to the decoder using a zero-tree-like algorithm.

The irregular wavelet approach provides low bit-rates (*cf.* Section 3). Since the encoding process is driven by the original connectivity, and no cutting procedure is requested, the technique is free of cracking problems and distorted re-triangulations. In addition, the algorithm computational complexity of both encoding and decoding processes is low (with real-time decoding capabilities).

The main challenge of such an approach concerns the wavelet filters coefficients which represent parametric information determined and optimized for the first frame. In order to ensure good compression results, such parameters should be adequate for all the frames in the sequence, which is not always the case in practice. The mesh sequences fulfilling this hypothesis are called *parametrically coherent*.
To fully exploit the multi-resolution character of the representation, the approach is specifically adapted for large meshes, for which a maximum number of levels of details can be obtained.

2.3 Clustering based approaches

The first clustering-based compression scheme has been introduced by Lengyel in [22]. Lengyel approach consists of splitting the mesh into sub-parts, so-called clusters, whose motion can be accurately described by rigid transform, defined with respect to a reference frame (*e.g.* the first frame in the sequence), and determined using an heuristic approach. Such a transform allows to apply a motion compensation procedure and to predict the position of vertices in the current frame from the reference one.
The object's motion is finally described by the set of rigid motion parameters associated with each cluster and the prediction residuals associated with each vertex.

A similar approach is presented in in[8] , where authors use some more sophisticated tools for improving the mesh segmentation and the motion estimation steps. Thus, an *Iterative Closest* Point (ICP) algorithm is used in order to determine the motion of each cluster. The mesh segmentation exploits a topology-based partitioning algorithm[25] for determining an initial partition which is further refined based on some motion coherency criteria.

The Lengyel's approach is further improved in[23], where animation is expressed only in term of rigid transforms (RT) (no residuals are here encoded). Authors introduce a new weighted least square mesh segmentation algorithm which minimizes the number of clusters under a distortion bound criterion.
However, such segmentation-based approaches lead to visually unpleasant motion discontinuities at the level of frontiers between clusters. In order to reduce this effect, a spatio-temporal smoothing scheme, based on a post-processing, low-pass filtering of vertex displacements is applied.

A different clustering-based representation, based on an octree decomposition of the object is proposed in[6]. Here, a cubic bounding box of the 3D object is first constructed. Eight motion vectors are associated with its corners. The motions of interior mesh vertices are computed using a cubic interpolation scheme. Then, the cubic bounding box is iteratively refined by applying a recursive cell subdivision scheme which results in an octree structure. For each cell of the obtained octree, new motions vectors, determined with a least squares approximation procedure, are associated with its corners. The octree structure and the associated motion vectors are finally quantized and entropy encoded.
An optimized version of this approach, so-called Dynamic 3D Mesh Coder (D3DMC), is introduced in[7]. Here, authors exploit a context-adaptive binary arithmetic coder[24], which makes it possible to efficiently take into account the signal statistics.

By exploiting a more elaborate and semi-global representation, based on higher order, parametric motion models (rigid transforms, cubic polynomials…), the clustering-based approaches can compactly describe a large category of motions that can be modeled as piecewise rigid transforms. The encoding computational complexity is determined by the motion estimation and segmentation steps. However, the decoding process can be achieved with a low complexity, since only simple prediction models are involved.
The main limitation of the clustering-based techniques is related to the segmentation procedure which is computationally complex and may cause disgraceful motions discontinuities at low bit rates.

A final family of approaches concern the global, PCA-based representations, which treats the animation sequence as whole.

2.4 PCA- based approaches

The first PCA-based approach for 3D animation compression has been introduced by Alexa and Müller in[1].
First, a global motion compensation procedure is applied in order to de-couple elastic and rigid body motions. For each frame G_i, the rigid part of the motion is modeled globally as rigid 3D transform, denoted by R_i. The transform

parameters (rotation and translation components) are determined by minimizing the mean square error between the motion compensated frame $\tilde{G}_i = R_i(G_i)$, and the first frame G_0, considered as a reference frame. The differences between the $\tilde{G}_i$ vectors and the reference geometry G_0 can be considered as a non-rigid deformation field that can be efficiently describes by a principal component analysis (PCA). Let us denote by $\tilde{G}$ the matrix whose columns represent the $\tilde{G}_i$ vectors. The PCA is determined by using the singular value decomposition (SVD) of $\tilde{G}$ and is expressed as:

$$\tilde{G} = E \cdot S \cdot U^t, \tag{5}$$

where E is a column-orthogonal rectangular matrix, S is a diagonal matrix containing the singular values $(\lambda_j)_{j\in\{0,\ldots,F-1\}}$, and U an orthogonal square matrix. Let us denote by $(e_j)_{j\in\{0,\ldots,F-1\}}$ the orthogonal column vectors of E, considered to be sorted by decreasing order of corresponding singular values.

For an arbitrary integer *k*, with $1 \le k \le F$, the decomposition of $\tilde{G}_i$ within the set of orthogonal vectors $\{e_0, \ldots, e_{k-1}\}$, denoted by $\hat{\tilde{G}}_i^k$ is defined as:

$$\forall\, k \le F, \quad \forall\, i \in \{0, 1, \ldots, F-1\}, \quad \hat{\tilde{G}}_i^k = \sum_{j=0}^{k-1} \alpha_i^j e_j, \tag{6}$$

with coefficients α_i^j given by:

$$\alpha_i^j = \langle e_j, \tilde{G}_i \rangle, \tag{7}$$

where $\langle \cdot, \cdot \rangle$ denotes the dot product operator.

Since the $(e_j)_{j\in\{0,\ldots,k\le F-1\}}$ vectors are sorted by decreasing order of corresponding singular values, the (α_i^j) coefficients are capturing the most important part of the geometry signal energy, which ensures the efficiency and the progressivity of the representation. Moreover, The $\hat{\tilde{G}}_i^k$ provides the best approximation, in the mean square error sense, of geometries $\tilde{G}_i$ (defined in $\mathbb{R}^{3V}$) within a *k*-dimensional subspace.

The encoded representation includes first the $(e_j)_{j\in\{0,\ldots,k-1\}}$ basis vectors within the bitsream as a payload information. For each frame *i,* the global transform R_i and the decomposition coefficients (α_i^j) are then transmitted.

As the basis vectors need to be included within the bitstream as overhead, the PCA approach is efficient only for long animation sequences of small meshes with number of frames F much bigger (at least one order of magnitude) than the number of vertices V. When this condition is not satisfied, the PCA approach leads to poor compression results (*cf.* section 3). On the other hand, the SVD decomposition has a computational complexity cubic with the number of mesh vertices. This makes the encoding time prohibitive for large meshes with thousands of vertices.

In[2], Karni and Gotsman enhance the PCA approach by introducing a second order, temporal linear prediction (LP) scheme, that is applied to the sequence of (α_i^j) coefficients. Combining PCA with LP makes it possible to better exploit the temporal redundancies between successive frames. Authors claim a gain in compression efficiency with respect to the PCA approach of about 1.7.

A final refinement is introduced in[3], where authors propose to partition the mesh vertices into a set of clusters that are optimally adapted to a PCA representation. The segmentation is achieved by applying the modified Lloyd algorithm described in[11] to the trajectories of the mesh vertices. The obtained clusters are compressed independently by applying the PCA technique. This clustering-based PCA (CPCA) approach provides better compression performances especially for short animation sequences where the condition $F << V$ does not hold (*cf.* section 3).

Table 1 summarizes the principles, properties, and offered functionalities of the different families of approaches.

Approach		Principle	Encoding computation complexity	Progressive transmission	Scalable rendering	Applicability
Vertex-based predictive approaches	**MPEG-4 IC**	Local spatio-temporal prediction	*	no	no	all meshes
	Dynapack	Local spatio-temporal prediction	*	no	no	manifold meshes
Clustering-based approaches		Spatial segmentation, parametric motion models and temporal prediction	**	no	no	all meshes
Wavelet-based approaches	**Re-meshing**	Regular wavelet transform	***	yes	yes	manifold meshes admitting low-distortion parameterizations
	Irregular wavelets	Irregular and anisotropic wavelets transform on the top of a progressive mesh hierarchy	*	yes	yes	parametrically coherent manifold mesh sequences
PCA-based approaches		SVD decomposition performed on the set of all the frames	***	yes	no	all meshes

Table 1. A summary of the principal families of approaches and related principles, properties and functionalities.

Here, the computation complexities are evaluated only qualitatively (* - low, ** - medium, *** - high) since no theoretical bounds are available for the methods and since several pre-processing and optimization procedures may be involved. Our evaluation is based on the computation times reported in the literature.

Table 1 shows that only the wavelet-based techniques provide both progressive transmission and scalable rendering. In particular, the irregular wavelet transform has a very low computational complexity, which makes it very attractive for mobile and real-time applications. However, both regular and irregular wavelet representations have a limited applicability, since they rely on strong hypotheses related to the mesh sequences.
On the contrary, clustering-based, and predictive approaches correspond to mono-resolution representations that do not support such advanced functionalities. However, such approaches offer the advantage of the lowest encoding computational complexity because of their simplicity.
An intermediate level in terms of functionalities is offered by the PCA-based approaches which supports progressivity but no scalable rendering, at the price of a prohibitive computational complexity.

For our comparison, we retained the most promising and representative approaches of each family:

- the irregular wavelet-based encoding scheme AWC[4],
- the PCA-based approaches PCA[1], LPCA[2], and CPCA[3],
- the vertex prediction-based techniques AFX-IC[27] and Dynapack[5], and
- the clustering-based encoders RT[23] and D3DMC[7].

In addition, we have considered two keyframe-independent encoders based on the static MPEG-4[13] and TG[12], approaches. Here, the different key-frames of the animation sequence are considered independently and encoded using static mesh compression techniques.

Let us now analyze the performances of the considered approaches in term of compression efficiency.

3. COMPARATIVE STUDY

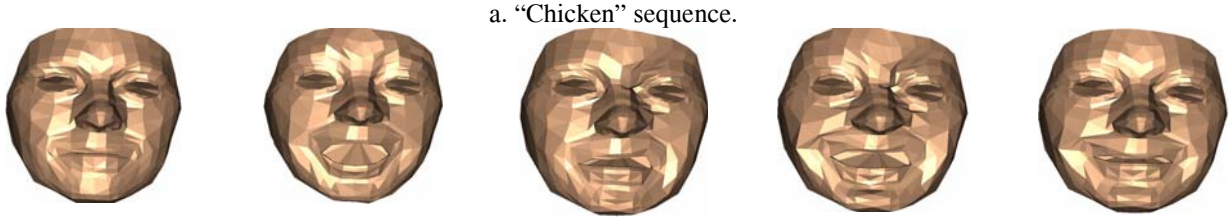
In this section we report, compare and discuss the compression performances of the various 3D animation compression approaches.

Let us first describe the test data set considered for evaluation.

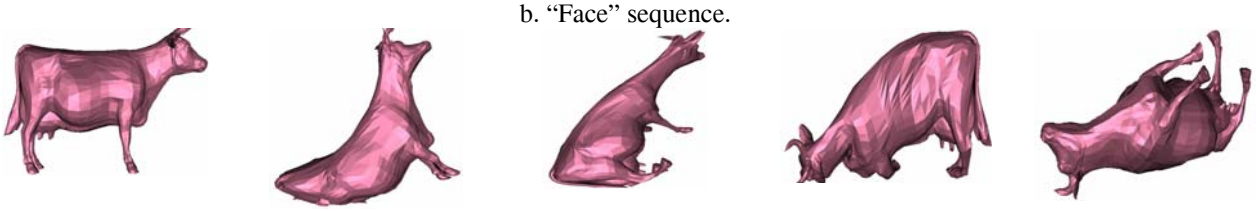
3.1 Evaluation corpus

In order to be able to perform objective comparisons, we have considered as data set the three animation sequences (Figure 1) used by the majority of the works developed in the literature, so-called "Chicken", "Face" and "Cow". Table 2 summarizes their properties, expressed in terms of numbers of vertices, triangles, Connected Components (CC) and frames.

a. "Chicken" sequence.



b. "Face" sequence.



c. "Cow" sequence.

Figure 1. Test data set: Frames from the "Chicken", "Face", and "Cow" sequences.

Animation sequence	Number of vertices	Number of triangles	Number of CCs	Number of frames	Original size (KB)
"Chicken"	3030	5664	41	400	14,544
"Cow"	2904	5804	1	204	7,108
"Face"	539	1042	1	10001	64,686

Table 2. Properties of the animated mesh sequences considered in the evaluation.

In Table 2, the original sizes, denoted by S_O are reported in Kilo Bytes (KB) and are measured as follow:

$$S_O = 2^{-10} \times F \times V \times 3 \times n_{bytes}, \quad (8)$$

where F, v and n_{bytes} stand for the numbers of frames, vertices and bytes to encode a vertex coordinate. For all the sequences $n_{bytes} = 4$, which corresponds to the floating point representation.

The considered models offer a good variability in terms of spatial and temporal sizes. The "Face" and "Cow" mesh models are composed of an unique CC, with a significant difference in size, the "Cow" model being approximatively 6 times more complex. On the contrary, the chicken model is composed of 41 CCs, which yields an average of 73 vertices per CC.

From the motion point of view, the sequences can be characterized as low, repetitive motion for "Face", where only the mouth and the eyes of the virtual character are moving, and no global motion is present. The "Cow" animated mesh describes important elastic motions with global stretching and shrinkage of the cow body. The "Chicken" sequence exhibits a both articulated motion (*i.e.*, rigid transforms of body subparts) and local elastic deformations.

Evaluating compression performances requires to consider a set of objective criteria, characterizing both compression rates and distortions of the reconstructed models.

3.2 Objective evaluation criteria

The compression gains are evaluated in terms of both bitrates and compression ratios.

The *compression rates* are expressed in terms of bits per vertex per frame (bpvf). Here, the total number of bits needed to represent the sequence is divided by the number of frames and the number of vertices. This approach makes it possible to compare mesh sequences with different number of vertices and frames.

The *compression ratio C* is defined as:

$$C = \frac{S_O}{S_C}, \tag{9}$$

where S_C and S_O stand respectively for the sizes of the compressed stream and of the original sequence.

Concerning the evaluation of the reconstructed sequence's quality, we have retained the d_a measure introduced in[2] and the Root Mean Square Error[26] (RMSE), which are the most frequently used objective distortion measures in the literature. Let us recall their basic definitions.

The d_a *error* is defined as:

$$d_a = \frac{\| \Gamma - \breve{\Gamma} \|}{\| \Gamma - E(\Gamma) \|}, \tag{10}$$

where Γ is a $3V \times F$ matrix with the key-frames geometries $G_i = (X_i, Y_i, Z_i)^t$ as columns, $\breve{\Gamma}$ the compressed version of Γ and $E(\Gamma)$ the average matrix in which each column consists of the average vertex positions for all frames. Here, the denominator stands for a normalization factor which makes it possible to compare meshes defined at different scales.

The *RMSE error*[26] between two surfaces S_1 and S_2 is defined as:

$$RMSE(S_1, S_2) = \frac{1}{D} \times \max\left(\sqrt{\frac{1}{|S_1|} \iint_{p \in S_1} d(p, S_2)^2 \, dS_1}, \sqrt{\frac{1}{|S_2|} \iint_{p \in S_2} d(p, S_1)^2 \, dS_2} \right). \tag{11}$$

where $d(p, S) = \min_{q \in S} \| p - q \|_2$. Here, the normalization factor D is the length of the diagonal of the mesh's bounding box.

The RMSE error between two mesh sequences is defined as the mean value of the frame to frame *RMSE*, calculated over all the frames in the sequence.

3.3 Compression results

Tables 3, 4 and 5, summarize the performances of the retained compression approaches on the three considered sequences. Not all compression performances are available for the three sequences. Figure 2 presents the rate/distortion curves for some of the encoders on the three considered sequences.

As expected, the static MPEG-4[13]and TG[12] approaches lead to poor compression performances since they do not exploit the temporal correlations.

Concerning the vertex predictive approaches, the MPEG-4 AFX-IC[27] encoder (here, the keyframes were obtained by applying uniform subsampling with a factor of 2) provides better results than the MPEG-4 static compression approach. However its performances are poor when compared to other spatio-temporel predictive schemes such those considered by Dynapack (which offer a 50% gain for the "Chicken" sequence).

Concerning the PCA-based approaches[1,2,3] , they provide the best rate/distortion curves on the "Face" sequence. Here, the global representation, the small number of vertices ($V << F$) and the repetitive character of the animation sequence make such approaches optimal. The LPCA compression scheme[2] improves the performance of the PCA approach[2] by a factor of 1.7, while the CPCA technique[3] provides compression results comparable to LPCA.

Approach	Rate (bpvf)	Size (KB)	Compression ratio	Distortion
MPEG-4 (static)	15-27	2278-4162	3,5-6,3	RMSE: 0,0004-0,007
TG (static)	14-26	2123-4038	3,6-6,9	RMSE: 0,0004-0,007
PCA	5,9-23,6	895-3577	4-16	d_a: 0,03-0,677
CPCA	2,8-8,7	424-1318	11-34	d_a: 0,002-0,139
Dynapack (ELP)	4,1-9	622-1368	10,5-23	RMSE: 0,0002-0,0136
Dynapack (Replica)	4,1-8,7	622-1322	11-23	RMSE: 0,0002-0,0136
RT	0,24-7,2	37,2-1085,8	13,5-390	RMSE: 0,0195-0,08
D3DMC	0,36-3,3	55-500	29-264	RMSE: 0,005-0,048
AFX-IC	5,2-10,3	800-1560	8-18	RMSE: 0,007-0,04

Table 3. Compression performances for the "Chicken" sequence using various encoders.

Approach	Rate (bpvf)	Size (KB)	Compression ratio	Distortion
PCA	0,9-5,3	600-3485	18.1-106.6	d_a: 0,05-0,149
CPCA	1,5-10	1010-6738	9.6-63	d_a: 0,013-0,58
LPCA	0,5-3,1	322-2046	31-337	d_a: 0,05-0,149
Dynapack	4,7-10,2	3090-6873	14-20.5	d_a: 0,521-0,001
AWC	3,9-10,3	2628-6940	9.3-24.6	d_a:0,05-0,52

Table 4. Compression performances for the "Face" sequence using various encoders.

Approach	Rate (bpvf)	Size (KB)	Compression ratio	Distortion
CPCA	4-18	296-1332	5,3-24	d_a: 0,01-0,35
LPCA	11-40	814-2962	2,4-8,7	d_a: 0,49 - 0,01
AWC	5-17	370-1258	5,6-19,2	d_a:0,33-0,01

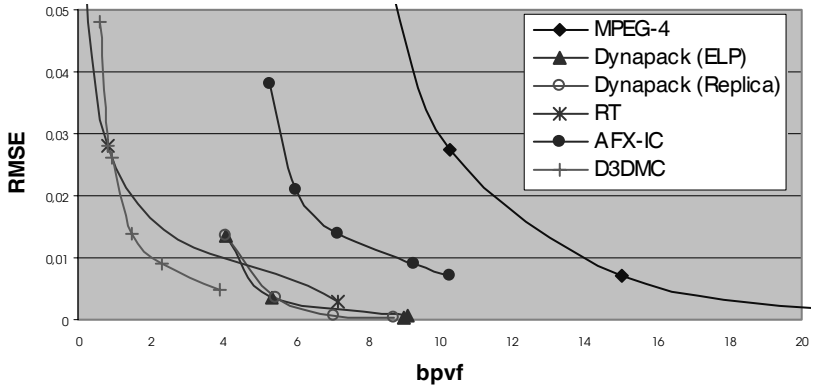
Table 5. Compression performances for the "Cow" sequence using various encoders.

However, for the "Chicken" and "Cow" sequences, the PCA-based compression results dramatically degrade. Thus, for the some range of d_a error (Table 3), the compression rates of the PCA encoder on the sequence "Chicken" are 5 times greater when compared to those obtained on the sequence "Face" !
For the "Chicken" sequence, the number of vertices $V = 3030$ is 7.5 greater than the number of frames ($F = 400$). This makes the "payload" of the PCA basis vectors, that need to be integrated into the bitstream and transmitted to the decoder, expensive. For the "Chicken" sequence it is estimated[2] to 5-22 bpvf.
Finally, let us note The CPCA approach improves the PCA and LPCA performances respectively by a factor of 3 and 2. This nice behavior is explained by the fact CPCA makes profit from the mesh partitioning procedure involved which leads to a spatially more local motion representation.

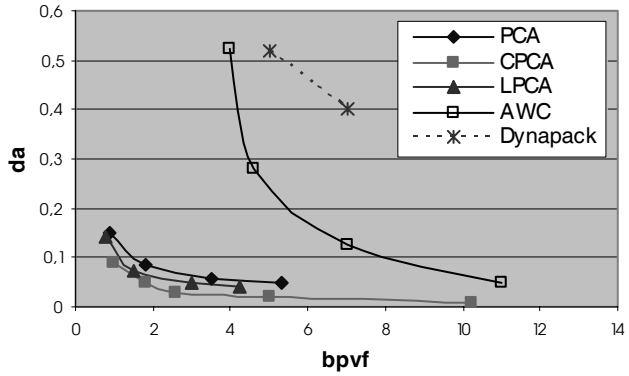
For the "Chicken" sequence, the clustering based approaches RT[23] and D3DMC[7] show the best compression performances at low bit rates (less than 4 bpvf). Table 3 demonstrates the superiority of these encoders over the PCA-based approaches. When compared to AFX-IC and Dynapack, the clustering-based techniques provide respectively up to 70% and 60% better compression performances. The D3DMC encoder outperforms RT approach for bitrates higher than 2 bpvf. For bitrates ranging from 5 to 9 bpvf, the Dynapack approach shows slightly better results than RT.

No compression results of AWC[4] were provided for the "Chicken" sequence. The multi-CC structure of this animated mesh makes this approach inadequate since the CCs are too tiny (in average 73 vertices per CC) to enable the construction of a progressive mesh hierarchy with a sufficient number of LODs .
Concerning the seamless mesh sequences, on the "Face" sequence the AWC approach is outperformed by the PCA-based approaches. Here again, the small size of the mesh reduces the efficiency of this encoder. Despite this limitation the AWC has better compression performances than Dynapack. For the "Cow" sequence, the AWC approach provides the better rate/distortion curve. Here, the initial mesh is sufficiently large (2904 vertices) to permit the construction of a

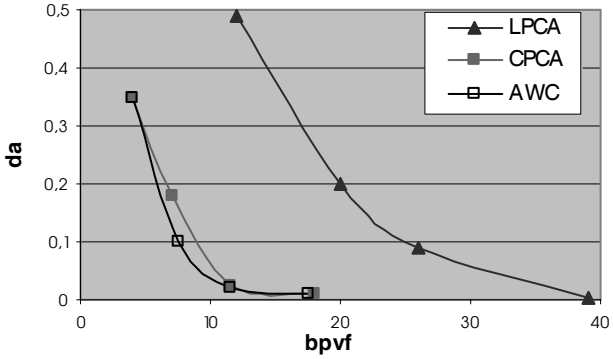
hierarchy with a large number of LODs, well-suited for zero-tree encoding. The compression performances of the LPCA are far from those of AWC since the condition $V << F$ does not hold. The clustering strategy adopted by CPCA[3] overcomes this limitation and offers comparable results to AWC.
These preliminary cross-evaluation shows that the clustering-based RT and D3DMC approaches provide the most interesting compression performances, and offers a good tradeoff between compression performances and computational complexity. Their main limitation relates to the limited functionalities offered, since they do not support neither progressive transmission, nor scalable rendering.



a. "Chicken" sequence.



b. "Face" sequence.



(c) "Cow" sequence.

Figure 2. Rate/distortion curves for: (a) "Chicken", (b) "Face", and (c) "Cow" sequences using various encoders.

4. CONCLUSION AND PERSPECTIVES

This paper presents an overview of the state of the art in animated meshes compression. It focuses on the new trends in this field.
The recently developed animation compression techniques are presented and evaluated in terms of compression performances and provided functionalities. Experimental results show that the clustering based techniques RT[23] and D3DMC[7] lead to the best compression performances at low bitrates. The PCA-base approaches[1,2,3] prove to be particularly efficient for long animated sequences of small meshes (hundreds of vertices). The Dynapack[5] prediction scheme outperforms the AFX-IC[27]. Only the PCA-based and wavelets-based[4,9] encoders provide progressive transmission functionalities. The wavelet-based approaches enable also scalable rendering.

Our future work will address the issue of adapting the clustering-based approaches to progressive compression. The problem of discontinuities of the displacements field at the clusters boundaries will be addressed from a theoretical and implementation point of view.

ACKNOWLGEMENT

The "Chicken" sequence is property of Microsoft Incorporation. The "Face" sequence, generated by Demetri Terzopoulos, was kindly provided by Zachi Karni from Max-Planck-Institut für Informatik, while the "Cow" sequence by Matthias Müller from NovodeX AG Zurich.

REFERENCES

1. M. Alexa, W. Müller, "Representing animations by principal components", *Computer Graphics Forum,* Vol. 19, pp. 411-8, August 2000.
2. Z. Karni, C. Gotsman, "Compression of soft-body animation sequences", *Computer Graphics*, Vol. 28, pp. 25-34, 2004.
3. M. Sattler, R. Sarlette, R. Klein, "Simple and efficient compression of animation sequences", *ACM Siggraph Symposium on Computer Animation*, July 2005.
4. I. Guskov, A. Khodakovsky, "Wavelet Compression of Parametrically Coherent Mesh Sequences", *ACM Siggraph Symposium on Computer Animation*, pp. 183-192, Grenoble, France, 2004.
5. L. Ibarria, J. Rossignac, "Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity", ACM *Siggraph Symposium on Computer Animation*, pp. 126-13, San Diego, California, 2003.
6. J. Zhang, C. B. Owen, "Octree-based animated geometry compression", *Proc. of IEEE Data Compression Conference*, pp. 508-517, Snowbird, UT, March 2004.
7. K. Müller, A. Smolic, M. Kautzner, P. Eisert, T. Wiegand, "Predictive compression of dynamic 3D meshes", *ISO/IEC JTC1/SC29/WG11*, *MPEG04/M11240*, Palma de Mallorca, Spain, October 2004.
8. S. Gupta, K. Sengupta, A. A. Kassim, "Compression of dynamic 3D geometry data using iterative closet point algorithm", *Computer Vision and Image Understanding*, pp. 116-130, 2002.
9. H. M. Briceňo, P. V. Sander, L. McMillan, S. Gortler, H. Hoppe, "Geometry videos: a new representation for 3D animations", *ACM Siggraph Symposium on Computer Animation*, pp. 136-146, 2003.
10. I. Daubechies, I. Guskov, P. Schröder, W. Sweldens, "Wavelets on irregular point sets", *Phil. Trans. R. Soc. Lond. A*, Vol. 357, pp. 2397-2413, 1999.
11. N. Kambhatla, T. K. Leen, "Dimension reduction by local PCA", *Neural Computation*, Vol. 9, pp. 1493-1516, 1997.
12. C. Touma, C. Gotsman, "Triangle mesh compression", *Proc. of Graphics Interface*, pp.26–34, 1998.
13. G. Taubin, J. Rossignac, "Geometric compression through topological surgery", *ACM Transactions on Graphics*, Vol. 17, pp. 84-115, 1998.
14. A. Khodakovsky, P. Schröder, W. Sweldens, "Progressive geometry compression", *ACM Siggraph*, pp. 271-278, 2000.
15. H. Hoppe. "Progressive meshes", *ACM Siggraph*, pp. 99–108, 1996.
16. X. Gu, S. Gortler, H. Hoppe, "Geometry images", *ACM Siggraph*, pp. 355-361, San Antonio, Texas, 2002.
17. H. Hoppe, E. Praun., N. Dodgson, M. Floater, M. Sabin, "Shape compression using spherical geometry images", *Advances in Multiresolution for Geometric Modelling*, pp. 27-46, 2005.
18. P. V. Sander, J. Synder, S. J. Gortler, H. Hoppe, "Texture mapping progressive meshes ", *Proc. Computer Graphics*, pp. 409-416, 2001.
19. M. S. Floater, "Parameterization and smooth approximation of surface triangulations", *Computed Aided Geometric Design*, Vol. 14, pp. 231-250, 1997.
20. J. Rossignac, A. Safonova, A. Szymczak, "3D compression made simple: Edgebreaker on a corner table", *Proc. of Shape Modeling International Conference*, pp. 278-283, 2001.
21. L. Ibarria, P. Lindstrom, J. Rossignac, A. Szymczak, "Out-of-core compression and decompression of large n-dimensional scalar fields", *Eurographics*, pp. 343-348, September 2003.
22. J. Lengyel, "Compression of time-dependent geometry", *Proc. of ACM Symposium on Interactive 3D Graphics*, pp. 89-96, 1999.
23. G. Collins, A. Hilton, "A rigid transform basis for animation compression and level of detail", *Submitted to Vision, Video and Graphics*, 2005.
24. D. Marpe, H. Schwarz, T. Wiegand, "Context-Based Adaptive Binary Arithmetic Coding in the H.264/MPEG4-AVC Video compression Standard", *IEEE transactions on Circuits and Systems for Video Technology*, Vol. 13, pp. 620-636, 2003.
25. B. Hendrickson, R. Leland, "A Multilevel Algorithm for Partitioning Graphs", *Technical Report SAND93-1301*.
26. N. Aspert, D. Santa-Cruz, T. Ebrahimi, "MESH: Measuring Errors between Surfaces using the Hausdorff distance", *In Proc. of the IEEE International Conference in Multimedia and Expo (ICME) 2002*, Vol. 1, pp. 705-708, Lausanne, Switzerland, 2002.
27. E. S. Jang, J.D.K. Kim, S. Y. Jung, M.J. Han, S. O. Woo, S.J. Lee, "Interpolator Data Compression for MPEG-4 Animation", *In IEEE transactions on Circuits and Systems for Video Technology*, Vol. 14, 2004.
28. M. Bourges-Sevenier, E.S. Jang, "An introduction to the MPEG-4 Animation Framework extension", *In IEEE transactions on Circuits and Systems for Video Technology*, Vol. 14, pp. 928-936, 2004.