Vis Comput (2017) 33:811–821
DOI 10.1007/s00371-017-1395-4

ORIGINAL ARTICLE

Adaptive compression of animated meshes by exploiting orthogonal iterations

Aris S. Lalos[1] · **Andreas A. Vasilakis**[2] · **Anastasios Dimas**[1] · **Konstantinos Moustakas**[1]

Published online: 8 May 2017

Abstract We introduce a novel approach to support fast and efficient lossy compression of arbitrary animation sequences ideally suited for real-time scenarios, such as streaming and content creation applications, where input is not known a priori and is dynamically generated. The presented method exploits temporal coherence by altering the principal component analysis (PCA) procedure from a batch- to an adaptive-basis aiming to simultaneously support three important objectives: fast compression times, reduced memory requirements and high-quality reproduction results. A dynamic compression pipeline is presented that can efficiently approximate the *k*-largest PCA bases based on the previous iteration (frame block) at a significantly lower complexity than directly computing the singular value decomposition. To avoid errors when a fixed number of basis vectors are used for all frame blocks, a flexible solution that automatically identifies the optimal subspace size for each one is also offered. An extensive experimental study is finally offered, showing that the proposed methods are superior in terms of performance as compared to several direct PCA-based schemes while, at the same time, achieves plausible reconstruction output despite the constraints posed by arbitrarily complex animated scenarios.

Electronic supplementary material The online version of this article (doi:10.1007/s00371-017-1395-4) contains supplementary material, which is available to authorized users.

✉ Aris S. Lalos
aris.lalos@ece.upatras.gr

1 Electrical and Computer Engineering Department, University of Patras, Patras, Greece

2 Information Technologies Institute, Centre for Research and Technology Hellas, Athens, Greece

Keywords 3D animated meshes · Orthogonal iterations · Online compression

1 Introduction

With the rapid advances in high-performance computing, scanning operations and content creation tools, the output data are expanding rapidly generating massive datasets. While this can be mitigated by applying compression techniques to the data being archived or transferred, the tremendous computing resources required bring tough challenges to be solved. Recently, there has been increasing interest on acquiring, processing, storing and transmitting 3D animated meshes facilitating several real-time applications (e.g., *Microsoft Holoportation*, an immersive telepresence system).

Throughout the years, numerous approaches have been proposed improving more or less some of the key animation compression characteristics [16]: encoding–decoding requirements, reconstruction quality and compression rates. Without loss of generality, these methods can be classified either as *local*- or *global*-based, depending on the frame window analysis taken for compressing the animation sequence. The main benefit of the global approaches is an improved compression rate by analyzing the overall motion coherence, whereas the local ones focus on local frame-to-frame transitions allowing low-latency streaming.

Skinning as well as principal component analysis (PCA) can be considered as the most well-known global methods for providing efficient compact representations of rigid and highly deformable animations, respectively. Though a large variety of different strategies have been introduced in both skinning [8] and PCA [16], all of them suffer from excessive computational requirements, strongly dependent on the

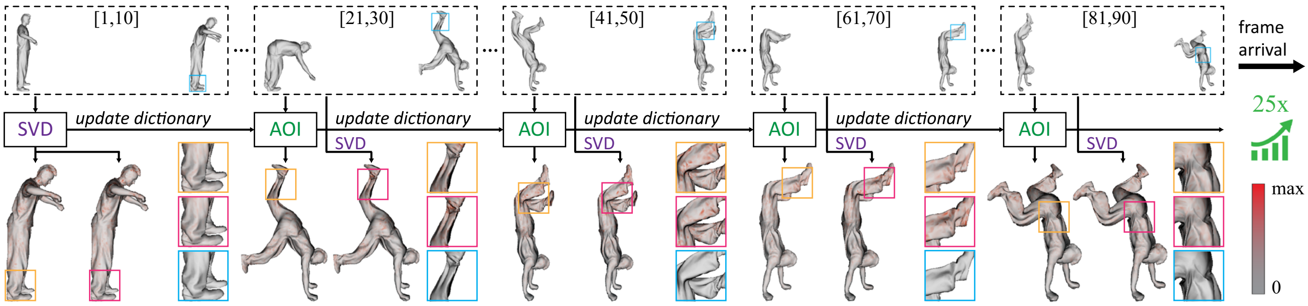

Fig. 1 Overview of our animation compression method. Assuming sequential arrival of the animation data (*top*), the dictionary is dynamically updated for each cluster (block of 10 frames) by employing adaptive orthogonal iterations (AOI). The reconstructed results of the Handstand animation confirm that AOI achieves similar quality in 25× faster times when compared to a direct SVD-based solution (*bottom*)

geometry size and the total animation length. While for non-interactive applications, where animation is known a priori, the compression choice can easily be determined solely based on the underlying motion, new challenges rise when attempting to support unknown arriving geometry (see Fig. 2).

In this work, we introduce a novel out-of-core adaptive approach capable of approximating efficiently the PCA-based dictionary at a significantly lower computational complexity than directly applying singular value decomposition (SVD) as illustrated in Fig. 1. Inspired by the observation that only small deformation variations will normally occur between consecutive poses [32], we split the animation sequence into uniform frame blocks and perform subspace tracking by processing each incoming block at a time. By exploiting temporal coherence, we are capable of efficiently estimating the dictionary of the current frame batch using as input the precomputed dictionary of the previous batch [4]. To this end, we present a general iterative method to perform orthonormalization by exploring several orthogonal iterations variants [24]. In cases of high motion change between frame blocks, the optimal subspace size k can be automatically and dynamically adjusted to consistently reconstruct each incoming block of animation data. Our method can be successfully applied in cases where the animated data are handled as separate geometry clusters [22] as well as for exploiting both spatial [1] and temporal correlations [27]. An extensive experimental evaluation study on a wide range of animations considering a large spectrum of settings and configurations is finally offered showing the notable improvements in processing times (up to 60×) as compared to prior art without sacrificing the final reconstruction quality.

2 Related work

Animation compression schemes aim to provide a compact representation of the original animation, without affecting the perceived amount of distortion during reconstruction. Assuming constant *connectivity* (which can be encoded in an

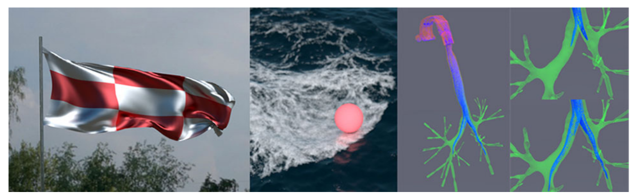

Fig. 2 Selected snapshots of (*left*) flag, (*middle*) ocean and (*right*) airflow dynamic simulations [12]

extremely efficient way [16], the sequence consists of a geometric evolution of the vertices of the initial mesh over time. A straightforward approach for compressing the animated geometry can be derived by applying one of the numerous available static mesh compression methods [3,13,14,18,25] to each one of the individual frames. Although such an approach would result in an efficient exploitation of the spatial coherence, it completely ignores the temporal one, missing a crucial factor to achieve higher compression ratios [9].

A large variety of animation compression algorithms has been introduced the last few years in the literature [16]. One can classify them into two core groups according to whether the time coherence is locally or globally analyzed. Despite the computationally fast behavior of the local approaches (e.g., wavelets [19] and predictive coding methods [23]), for the remainder of this paper we will focus on global compression schemes (Fig. 2).

PCA-based approaches. Generally, these can be classified according to where PCA is used to exploit spatial and/or temporal coherence of the input data, the matrix representation of the sequence's *geometry*. Alexa and Müller [1] were the first to employ PCA to fit a low-dimension subspace to the animation dataset and express each frame as a linear combination of the largest eigenvectors (EigenShapes) that spans the selected subspace. By exploiting temporal coherence, Karni and Gotsman [9] advanced this method by performing linear *prediction coding* on the extracted spatial PCA coefficients. Luo et al. have further achieved better compression ratio and computation times by aggregating similar frames into temporal *clusters* and compress them separately [15]. Contrary to the streaming nature of our method, in this approach the frames belonging to the same cluster may not be contiguous.

Conversely, several authors [22,27] suggested to apply PCA on the temporal space (EigenTrajectories); the main benefit is that it involves the decomposition of a smaller covariance matrix, since the number of frames is usually less than the number of vertices. To capture more efficiently local similarity characteristics, the authors in [2,20,22] suggested segmenting the mesh into deformation-aware *clusters*, which are then compressed independently. An adaptive compression allocation procedure is also offered, assigning more PCA coefficients to clusters that undergo extreme deformations [2]. To achieve better compression ratio, several algorithms were introduced to code (i) the PCA *bases* via principal components [22] and non-least squares optimal linear/accelerated movement [28] as well as (ii) the PCA *coefficients* by local *predictor coding* schemes such as the parallelogram rule [27], neighborhood average and radial basis functions [29] and Laplacian coordinates [26]. These methods can enhance their final approximation by representing the residual motion compensation *errors* via temporal DCT-based (discrete cosine transform) [17] or PCA-based corrections [11]. Before storing or transmitting, *quantization* (uniformly [7] or not [28]) is finally performed to encode the floating point data. Figure 3 provides a generic diagram summarizing the stage flow which one may follow during a PCA-based compression pipeline.

Subspace tracking approaches. Unfortunately, computing SVD (the standard *direct* approach to estimate PCA), can be extremely memory-demanding and time-consuming for large-scale animation problems (specifically, $O(m^3)$ for a matrix of $m \times m$ size). On the other hand, *subspace tracking* algorithms are fast alternatives relying on the execution of iterative schemes for evaluating the desired eigenvectors per incoming block of floating point data [4]. These alternatives schemes can be classified into *low* ($O(mk^2)$) and *high* ($O(m^2k)$) complexity, where $k (\ll m)$ denotes the number of principal eigenvectors. The best performance among the high complexity class is achieved by the *Lanczos Iterations* (LI) [33], while the primary representative of the other class is based on the *Orthogonal Iterations* (OI) [24]. Although the LI method requires less iterations for evaluating the subspace of a symmetric matrix, the complexity of each one is $O(m^2k)$. On the other hand, the OI alternative can result in very fast solutions when the initial input subspace is close to the subspace of interest, as well as the size of the subspace remains at small levels [21]. The effectiveness of OI is attributed to the fact that both matrix multiplications and

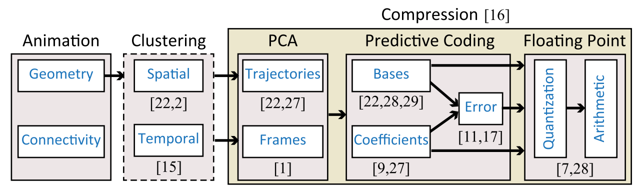

Fig. 3 High-level PCA-based compression framework. We revise accordingly the PCA computation stage providing notable speed benefits without altering the general pipeline

QR factorizations have been highly optimized for maximum efficiency on modern serial and parallel architectures. These properties make the OI approach more attractive for real-time applications. To the best of our knowledge, subspace tracking algorithms have never been applied to the animation compression problem, despite their wide success on a large range of filtering applications.

3 Overview of our method

Apart from the high-quality reconstruction output when compressing skeletal (skinning-based) and highly deformable objects (PCA-based), these methods assume that (i) all frames of the animation sequence are already known to the encoder, an unsuitable requirement for interactive scenarios; (ii) the maximum animation length lasts a few seconds (assuming real-time rendering at 30fps); as well as (iii) the geometry size is upper-bounded to a fixed number of triangles, avoiding memory overflows and performance delays.

We introduce a general, fast and lossy compression approach suited when the animated data are either dynamically produced or too large to fit into main memory at once. From a high-level point of view, the basic structure of our framework is very similar to that of the aforementioned methods, altering only the encoding stage from offline to online, leaving the rest unchanged including connectivity compression, geometry clustering, dictionary coding and quantization (see Fig. 3). In this work, the animation sequence is divided into an uniform block of frames corresponding to disjoint temporal intervals, which are then encoded independently one from another by performing adaptive (online) PCA-based compression (in either the spatial or the temporal domain). Assuming low dictionary variance between successive frame blocks, we successfully apply subspace tracking via adaptive orthogonal iterations (AOIs) significantly reducing the processing requirements needed if the conventional PCA schemes are alternatively utilized. When quality consistency of reconstructions is of utmost importance, the optimal subspace size k can dynamically be adjusted in an iterative manner. The entire pipeline is illustrated in Fig. 4 and discussed in further detail in the following sections.

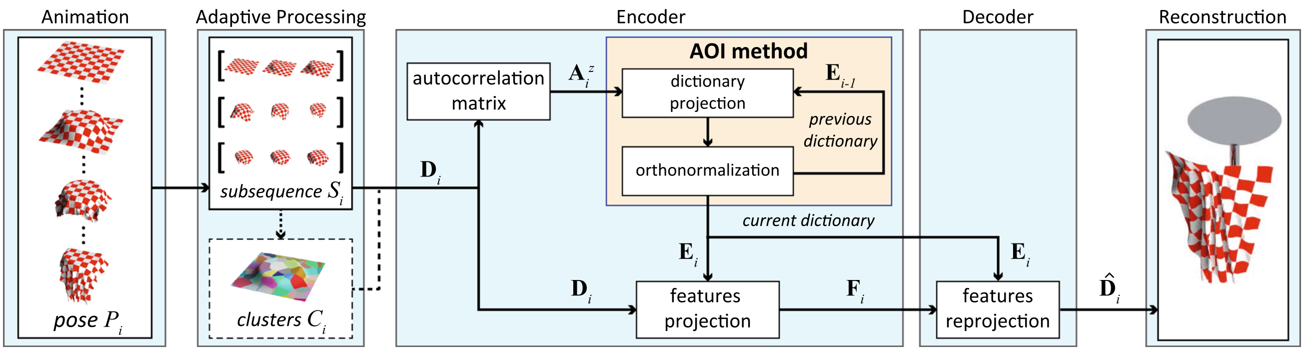

Fig. 4 Our adaptive animation compression pipeline. Temporal, and optionally spatial, clustering is initially employed to support the encoding of sequentially arriving animation frames (*left*). For each incoming block of frames, the dictionary is estimated by readjusting the subspaces computed at the previous block via adaptive orthogonal iterations (*middle*). High-quality reconstructions are finally computed by efficiently combining the estimated dictionary and feature vectors (*right*)

3.1 Preliminaries on PCA-based compression

The central idea of principal component analysis is to reduce the dimensionality of a dataset $\mathbf{D}$ consisting of a large number of interrelated variables without sacrificing the variation present in the dataset. This is achieved by transforming to a new set of variables, the *principal components*, which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables [5].

The input is a series of f consecutive static, point or triangular, meshes $\mathcal{P}_1, \ldots, \mathcal{P}_f$, namely *poses*. Each pose $\mathcal{P}_i$ with n vertices can be represented by two different sets $\mathcal{P}_i = (\mathcal{V}_i, \mathcal{F}_i)$ corresponding to the vertices $\mathcal{V}_i$ and the indexed faces $\mathcal{F}_i$. In our case, all poses have the same connectivity $\mathcal{F}_i = \mathcal{F}$, yet different geometry $\mathcal{V}_i$. So, the *animation data matrix* $\mathbf{D}$ can be defined as:

$$\mathbf{D} = \begin{bmatrix} \mathbf{p}_{11}, & \mathbf{p}_{12}, & \ldots, & \mathbf{p}_{1j}, & \ldots, & \mathbf{p}_{1n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{p}_{f1}, & \mathbf{p}_{f2}, & \ldots, & \mathbf{p}_{fj}, & \ldots, & \mathbf{p}_{fn} \end{bmatrix} \in \Re^{3f \times n}, \tag{1}$$

where $\mathbf{p}_{ij} = \left[x_{ij}, y_{ij}, z_{ij}\right]^{\mathrm{T}}$ are the x, y, z Euclidean coordinates of the vertex position $\mathbf{p}_j$ in pose $\mathcal{P}_i$.

The classic PCA-based approaches provide efficient representations in either the spatial [1] or the temporal space [22] by suggesting to evaluate the *autocorrelation matrix* $\mathbf{A}$ in the domain with the smaller size:

$$\mathbf{A} = \begin{cases} \frac{1}{e} \cdot \mathbf{D}\mathbf{D}^{\mathrm{T}} \in \Re^{e \times e} & \textbf{if } f \ll n \quad \text{(temporal)} \\ \frac{1}{n} \cdot \mathbf{D}^{\mathrm{T}}\mathbf{D} \in \Re^{n \times n} & \textbf{otherwise} \text{ (spatial)} \end{cases}, \tag{2}$$

where $e = 3f$. For the sake of simplicity, we will focus on the temporal case. After constructing the autocorrelation matrix (Eq. (2)), the direct SVD is applied to $\mathbf{A}$ leading to a factorization of the form:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^{\mathrm{T}} \tag{3}$$

where $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_e]$, $\mathbf{u}_i \in \Re^{e \times 1}$ and $\mathbf{U}^{\mathrm{T}}$ are called the *left-* and *right-singular* vectors, respectively. The matrix $\boldsymbol{\Sigma} = \mathrm{diag}(\lambda_i)$, $\lambda_i \in \Re$ contains the singular values of $\mathbf{A}$: $\lambda_i \geq \lambda_j > 0$, $\forall i > j \geq r = \mathrm{rank}(\mathbf{A})$. The encoding is performed by projecting the trajectories of each vertex to the subspace *dictionary* defined by the k-eigenvectors $\mathbf{E} = [\mathbf{u}_1, \ldots, \mathbf{u}_k] \in \Re^{e \times k}$ that correspond to the k-largest eigenvalues $[\lambda_1, \ldots, \lambda_k]$, creating a *feature vectors matrix* $\mathbf{F}$:

$$\mathbf{F} = \mathbf{E}^{\mathrm{T}}\mathbf{D} \in \Re^{k \times n}, \text{ where } k \leq r. \tag{4}$$

The compressed frames are finally decoded by multiplying the feature vectors with the dictionary:

$$\hat{\mathbf{D}} = \mathbf{E}\mathbf{F} \in \Re^{e \times n} \tag{5}$$

3.2 Adaptive processing via direct SVD

The most computational and memory-demanding operation of the previously described compression method is the SVD-based dictionary estimation. However, assuming that the dynamic data are processed in sequential manner (because it is either generated or streamed), a naive encoding solution is to temporally cluster the animation [15] into consecutive subsequences $\{\mathcal{S}_1, \mathcal{S}_2, \ldots\}$ of frame block sizes $\{d_1, d_2, \ldots\}$ and then directly perform SVD on each subsequence $\mathcal{S}_i = \{\mathcal{P}_{y+1}, \ldots, \mathcal{P}_{y+d_i}\}$, $y = \sum_{j=1}^{i-1} d_j$. For simplicity reasons, we subdivide the time space into uniform batches: $d_i = d, \forall i > 0$. In mathematical terms, the animation matrix $\mathbf{D}$ can be rewritten as a concatenation of sub-matrices $\mathbf{D}_i$ which we assume that are available sequentially:

$$\mathbf{D} = [\ \mathbf{D}_1^{\mathrm{T}}\ \mathbf{D}_2^{\mathrm{T}}\ \ldots\ ]^{\mathrm{T}},\ \text{where}$$
$$\mathbf{D}_i = \begin{bmatrix} \mathbf{p}_{(i-1)d+1,1}, & \ldots, & \mathbf{p}_{(i-1)d+1,n} \\ \vdots & \ddots & \vdots \\ \mathbf{p}_{id,1}, & \ldots, & \mathbf{p}_{id,n} \end{bmatrix} \in \Re^{3d\times n} \tag{6}$$

3.3 Adaptive processing via orthogonal iterations

By exploiting subspace tracking, we introduce an efficient way to adaptively estimate the dictionary at a lower cost as compared to the direct evaluation of SVD for each incoming sequence $\mathcal{S}_i$. In this section, we initially present the orthogonal iteration method (Sect. 3.3.1) followed by the analytic description of our adaptive scheme to the animation compression problem, capable of readjusting the dictionary of previous subsequence $\mathcal{S}_{i-1}$ in an iterative way. More specifically, we offer two alternative adaptive orthogonal iteration (AOI) schemes: the *Bandwidth-consistent* (BAOI) that allows the user to determine the compression efficiency by keeping fixed the dictionary size for the entire animation (Sect. 3.3.2) as well as the *Quality-consistent* (QAOI) that identifies the optimal subspace size required for achieving an user-defined acceptable reconstruction quality goal (Sect. 3.3.3).

3.3.1 Orthogonal iterations (OIs)

The orthogonal iterations [24] is an iterative procedure that can be applied to compute the singular vectors corresponding to the k dominant singular values of a symmetric, nonnegative definite matrix and can be summarized in the following lemma [5]:

Lemma 1 *Consider a symmetric, positive definite matrix* $\mathbf{M} \in \Re^{n\times n}$ *with* $\lambda_j > 0$ *and* $\mathbf{u}_j,\ 1 \le j \le n$ *be its nonzero singular values and corresponding singular vectors, respectively. Consider the sequence of matrices* $\{\mathbf{E}(t)\} \in \Re^{n\times k}$, *defined in the iteration t as*

$$\mathbf{E}(t) = \boldsymbol{o\_norm}(\mathbf{ME}(t-1)\ ),\ t = 1, 2, \ldots \tag{7}$$

where $\boldsymbol{o\_norm}$ *function stands for the orthonormalization procedure. Then, provided that the initial matrix* $\mathbf{E}^{\mathrm{T}}(0)$ *is not singular results at* $\lim_{t\to\infty} \mathbf{E}(t) = [\mathbf{u}_1, \ldots, \mathbf{u}_k]$.*Note that the convergence rate of this function depends on the* $\sigma_k = |\lambda_k/\lambda_{k+1}|$ *factor.*

Algorithm 1: BAOI update process

$\mathbf{E}(0) \leftarrow \mathbf{E}_{i-1}$;
for $t \leftarrow 1$ **to** $t_{\max}$ **do**
  $\mathbf{E}(t) \leftarrow \mathbf{o\_norm}(\mathbf{A}_i^z\mathbf{E}(t-1))$;
end
$\mathbf{E}_i \leftarrow \mathbf{E}(t)$;

3.3.2 Bandwidth-consistent AOI (BAOI)

In order to derive a low complexity compression scheme, we exploit the two attractive properties of the OI: (i) computational efficiency as well as (ii) fast convergence when the initialization is close to the subspace of interest. To this end, we suggest executing a single orthogonal iteration per incoming subsequence $\mathcal{S}_i$ using as initial input the subspace $\mathbf{E}_{i-1}$ that corresponds to the previous subsequence $\mathcal{S}_{i-1}$. More specifically, when we acquire the data block $\mathbf{D}_i$, we can replace $\mathbf{M}$ in Eq. (7) with the symmetric data autocorrelation matrix $\mathbf{A}_i$, leading to the following method for estimating the subspace of interest $\mathbf{E}_i = \mathbf{o\_norm}(\mathbf{A}_i\mathbf{E}_{i-1})$. Depending on the choice of $\mathbf{A}_i$, we can obtain alternative subspace tracking methods. The simplest and most efficient selection is the block estimate of the autocorrelation matrix,

$$\mathbf{A}_i = \mathbf{D}_i\mathbf{D}_i^{\mathrm{T}} + \delta \cdot \mathbf{I}_{3d} \tag{8}$$

where δ is a small scalar value that is used to ensure that $\mathbf{A}_i$ is nonnegative definite and $\mathbf{I}_s$ is the identity matrix of size s.

A common way to increase the estimation accuracy of the presented scheme is by executing more than one orthogonal iterations $t_{\max}$ per incoming block of animated data $\mathbf{D}_i$, using a fixed matrix $\mathbf{A}_i$. Contrary to the direct SVD approach, our method can benefit from the convergence properties of OI by substituting the matrix $\mathbf{A}_i$ with $\mathbf{A}_i^z$, where $z > 1$. This action increases the convergence speed of the method from σ_k to σ_k^z. Taking into account these observations, we can conclude in Algorithm 1.

To preserve orthonormality and initialize the subspace to the direct SVD subspace, it is important that the initial estimate $\mathbf{E}_1$ is orthonormal. For that reason, the initial subspace is estimated by applying direct SVD via Eq. (3), while the following subspaces $i = 2, \ldots$ are adjusted using Algorithm 1. A summary of the proposed technique is highlighted with pink in Algorithm 3.

3.3.3 Quality-consistent AOI (QAOI)

So far we have been assumed that a fixed, either insufficient or redundant, number k of components per frame block is used, neglecting the fact that the motion behavior may significantly vary during the whole animation sequence. To this end, we present a dynamic scheme that automatically identifies the optimal subspace size k_i for "error consistently" compressing each incoming frame block $\mathcal{S}_i$ based on the previous subspace size k_{i-1} and two user-defined low- and high-quality thresholds $\epsilon_{low}, \epsilon_{high}$ (instead of adjusting $t_{\max}$), thus allowing the user to easily trade reconstruction quality with speedup. The idea is to repeatedly increase or decrease k_{i-1} factor with the number of iterations t, i.e., $k_i = k_{i-1} \pm t$, with $k_0 = k$, until the value of an error metric $e(t)$ lies within the goal quality range $\left(\epsilon_{low}, \epsilon_{high}\right)$. Note that if we, instead, split the animation into non-uniform frame blocks (while keeping the subspace size fixed), it will potentially introduce unnecessary delays attributed to the packetization of a large number of frames.

In practical scenarios, it is reasonable to assume that the feature vectors $\mathbf{F}_i$ of each block $\mathbf{D}_i$ reside in subspaces $\mathbf{E}_i$ of different sizes. The subspace size k_i of the incoming data block $\mathbf{D}_i = \left[\mathbf{d}_{i_1}, \ldots, \mathbf{d}_{i_n}\right]$, $\mathbf{d}_{i_j} \in \Re^{3d \times 1}$, should be carefully selected so that the relevant animation data block characteristics are identified with the minimum loss of information. To quantify the loss of information at each iteration t, we suggest using the l_2-norm of the following mean residual error $\mathbf{r}(t) = \sum_{j=1}^{n} \left(\mathbf{d}_{i_j} - \mathbf{E}(t)\mathbf{E}^{\mathrm{T}}(t)\mathbf{d}_{i_j}\right)$. When moving from a high to low motion complexity, the incoming frame block $\mathcal{S}_i$ requires less than k_{i-1} principal eigenvectors to be sufficiently reconstructed. In this case, the subspace size k_i is iteratively decreased by 1 by simply selecting the first $k_{i-1} - 1$ columns of $\mathbf{E}(t)$ until the goal quality is achieved $\|\mathbf{r}(t)\|_2 > \epsilon_{low}$. On the other hand, inspired by the incremental PCA, we can simply add one normalized column in the estimated subspace $\mathbf{E}(t) = \left[\mathbf{E}(t) \cdot (\mathbf{r}(t) / \|\mathbf{r}(t)\|_2)\right]$ and then perform the orthonormalization stage. The proposed dynamic scheme can be easily integrated in Algorithm 3 (see blue color) by replacing Algorithm 1 with 2.

Orthonormalization. There are a number of different choices that can be used for the orthonormalization of the estimated subspace. The most widely adopted are the *Householder Reflections* (HR), *Gram-Schmidt* (GS) and *Modified Gram-Schmidt* (MGS) methods [6]. While all variants exhibit different properties related to the numerical stability and computational complexity, without loss of generality, we used HR for the orthonormalization stage.

Algorithm 2: QAOI update process

$\mathbf{E}(0) \leftarrow \mathbf{E}_{i-1}$; $k_i \leftarrow (i > 0)\ ?\ k_{i-1} : k$;
for $t \leftarrow 1$ **to** ∞ **do**
  $\mathbf{E}(t) \leftarrow$ **o_norm**$(\mathbf{A}_i^z \mathbf{E}(t-1))$;
  $\mathbf{r}(t) \leftarrow \sum_{j=1}^{n} \left(\mathbf{d}_{i_j} - \mathbf{E}(t)\mathbf{E}^{\mathrm{T}}(t)\mathbf{d}_{i_j}\right)$; **if** $|\mathbf{r}(t)| < \epsilon_{low}$ **then**
    $\mathbf{E}(t) \leftarrow \left[\mathbf{E}(t)\ \frac{\mathbf{r}(t)}{|\mathbf{r}(t)|}\right]$; $k_i \leftarrow k_i + 1$;
  else if $|\mathbf{r}(t)| > \epsilon_{high}$ **then**
    $\mathbf{E}(t) \leftarrow \left[\mathbf{e}_1 \ldots \mathbf{e}_{k_i}\right]$; $k_i \leftarrow k_i - 1$;
  else
    break;
  end
end
$\mathbf{E}_i \leftarrow \mathbf{E}(t)$

Algorithm 3: BAOI/QAOI encoding process

Function: Encoder $(\mathbf{D}_i, \mathbf{E}_{i-1}, k, z, t_{\max}/\epsilon_{low}, \epsilon_{high})$
Input : Current block of data $\mathbf{D}_i$, dictionary of previous block $\mathbf{E}_{i-1}$, power number z, orthogonal iterations $t_{\max}$/errors $\epsilon_{low}, \epsilon_{high}$
Output : Dictionary $\mathbf{E}_i$ and feature vector matrix $\mathbf{F}_i$
Estimate autocorrelation matrix $\mathbf{A}_i$ via Eq. (8);
if $i == 1$ **then**
  Estimate initial dictionary $\mathbf{E}_1$ via Eq. (3);
else
  Update $\mathbf{E}_i$ using $\mathbf{E}_{i-1}$ and k_{i-1} via Alg. 1/2;
  Orthonormalization via HR, GS or MGS.
end
Estimate feature vectors matrix $\mathbf{F}_i$ via Eq. (4);
return $\{\mathbf{E}_i, \mathbf{F}_i\}$;
// where 'text' and 'text' defines the bandwidth- and quality-consistent AOI implementations, respectively.

4 Results and discussion

We present an experimental analysis of our subspace tracking approach as compared to the direct application of PCA in an adaptive setup focusing on encoding performance and decoding robustness (Sect. 4.2) of different animation sequences under a broad set of configuration parameters (Sect. 4.1). A short discussion is finally offered describing on how to select a compression variant from the given repertoire (Sect. 4.3).

4.1 Simulation setup

To provide an objective comparison between the tested solutions, we follow the general pipeline stages described in Section 3 and illustrated in Fig. 4 using a series of 3D dynamic point clouds and triangular meshes with fixed connectivity, that represent a wide range of rigid and highly deformable motions (see Table 1 and video).We assume that the animation poses are not known ahead in time and are generated and processed sequentially. To that end, we divide the animation into uniform blocks of consecutive frames or namely subsequences $\{\mathcal{S}_i\}$ (Sect. 3.2) that are processed individually. In addition, our framework supports also the segmentation and processing of spatial clusters allowing a parallel adaptive implementation of very dense animated meshes. **METIS** [10], an efficient topological partitioning approach was used that *uniformly* partitions a mesh based on its connectivity. For each data block, compact representations can be computed by approximating the k-largest PCA bases in *spatial* or *temporal space* via

SVD: Jacobi singular value decomposition (Sect. 3.2). Note that experiments

Table 1 Extensive comparison in terms of quality and performance of the SVD and BAOI for a variety of compression configuration setups in a highly motion-divergent collection of dynamic sequences

Animation	Vertices	Frames	Block	Bases	Rate	Error (STED) $\times 10^{-2}$			Performance (s)		
	n	f	d	k	bpvf	SVD	BAOI	BAOI(2)	SVD	BAOI	BAOI(2)
Name	[Temporal space]										
Tablecloth	4225	240	30	15	6.35	13.608	13.721	13.652	0.152	0.05 (**2.71**×)	0.06 (2.45×)
			60	30	6.39	13.142	13.321	13.211	0.464	0.09 (**4.73**×)	0.12 (3.93×)
Flag	2704	1000	125	50	5.62	26.211	26.935	26.438	13.26	0.69 (**19.22**×)	1.08 (12.28×)
			250	50	3.20	27.450	28.069	27.641	74.89	1.47 (**50.98**×)	2.62 (28.58×)
Tsunami	4225	1250	125	35	3.75	9.6370	9.6381	9.6376	27.11	1.28 (**21.18**×)	1.81 (14.98×)
			250	70	4.10	8.6660	8.6736	8.6663	140.10	2.58 (**54.30**×)	3.88 (36.11×)
$\text{Name}_{\text{clusters}}$	[Spatial space]										
Handstand$_{40}$	10,002	160	40	4	4.96	18.404	19.269	18.966	196.5	3.24 (**60.60**×)	6.38 (30.80×)
100			40	4	5.79	16.154	16.656	16.359	23.84	0.84 (**28.48**×)	1.71 (13.92×)
Camel$_{100}$	21,887	50	10	2	9.33	13.665	14.824	13.849	325.0	10.9 (**29.76**×)	20.9 (15.57×)
200			10	2	9.65	13.649	14.762	14.037	65.90	4.36 (**15.11**×)	7.44 (8.86×)
Elephant$_{200}$	42,321	48	24	4	7.93	10.983	10.907	10.900	157.2	5.08 (**30.93**×)	9.57 (16.43×)
400			24	4	8.61	9.7720	9.7860	9.7810	33.11	1.76 (**18.81**×)	3.39 (9.76×)

The minor reconstruction differences between the testing methods are significantly reduced (with a small speedup reduction cost) by performing an extra orthogonal iteration

The best speed up gains are highlighted in bold

were also tested with the truncated SVD resulting to similar observations.

BAOI$^z(t_{\max})$: bandwidth-consistent adaptive orthogonal iterations on the $\mathbf{A}^z$ matrix (Sect. 3.3.2). Note that the default values for BAOI parameters are $z = 1$, $t_{\max} = 1$ unless specified otherwise, meaning that BAOI corresponds to BAOI$^1(1)$.

QAOI$(\epsilon_{low}, \epsilon_{high})$: quality-consistent adaptive orthogonal iterations (Sect. 3.3.3) until goal reconstruction quality resides between $(\epsilon_{low} \epsilon_{high})$.

IPCA: Incremental PCA. It identifies the best subspace size for achieving a user-defined quality.

For simplicity reasons, *uniform quantization* [7] is performed on the generated dictionary ($q_d = 14bits$) and feature vectors ($q_f = 12bits$).

Encoding performance. We measure the performance in terms of milliseconds (**ms**) for executing only the dictionary estimation process (without including the initialization SVD step), since the rest stages remain the same for all variants under study (see Fig. 4). The experiments were performed on an Intel Core i7 4790 @3.6GHz CPU with 8GB RAM.

Reconstruction quality. The evaluation of the distortion amount between the original and reconstructed animation is traditionally performed by vertex-based error metrics such as the well-established **KG** error metric [9] which is defined as $\mathbf{KG} = 100 \cdot \frac{||\mathbf{D}-\hat{\mathbf{D}}||_F}{||\mathbf{D}-E(\mathbf{D})||_F}$, where $||\cdot||_F$ denotes the Frobenius norm and $E(\mathbf{D})$ is a matrix whose columns consist of the average vertex positions for all frames. On the other hand, **STED** error metric has been shown to correlate well with perceived distortion by measuring *spatiotemporal edge differences* [30]. The overall error is evaluated as a hypotenuse of two parts: spatial $\mathbf{E_S}$ and temporal $\mathbf{E_T}$ errors using a weighting constant w to relate them **STED** $= \sqrt{\mathbf{E_S}^2 + w^2 \cdot \mathbf{E_T}^2}$. In this work, we have used both metrics to measure distortion related to absolute via KG (Figs. 1, 6, 7, 8, 10) as well as relative changes via STED (Table 1 and video) of the vertex positions.

Compression efficiency. Data rate is measured in *bits per vertex per frame* (**bpvf**) encapsulating the mesh connectivity, the feature vectors matrix and the dictionary coefficients required for reconstructing the original dataset at the decoder.

4.2 Experimental study

Impact of spatiotemporal correlations. Figure 5 illustrates a reconstructed frame using two state-of-the-art static mesh compression approaches, **MBL** [13] and **OD3GC** [18], as compared to the SVD and BAOI approaches. Observe that the exploitation of PCA significantly outperforms the application of static compression approaches on a frame-to-frame basis in terms of both reconstruction accuracy and execution time.

Impact of frame block size (d). Figure 6 illustrates how altering the time subdivision length $d = \{50, 250\}$ affects the reconstruction accuracy as well as the total processing time

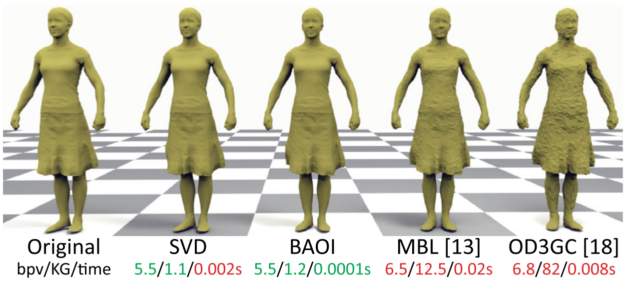

Fig. 5 Comparison of PCA-based approaches with static mesh compression methods at Samba animation sequence

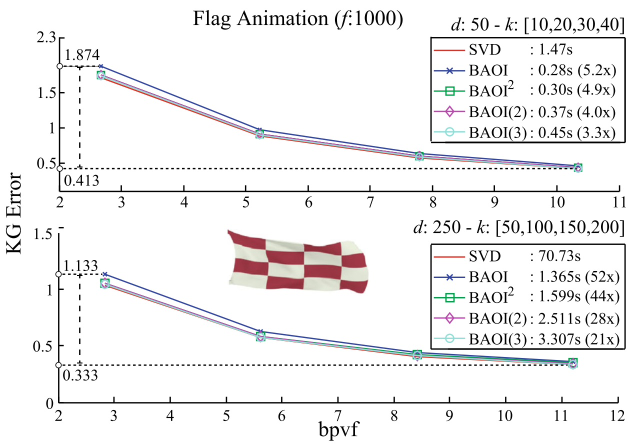

Fig. 6 Approximation and performance evaluation for the Flag animation sequence for different compression ratios using various frame block sizes

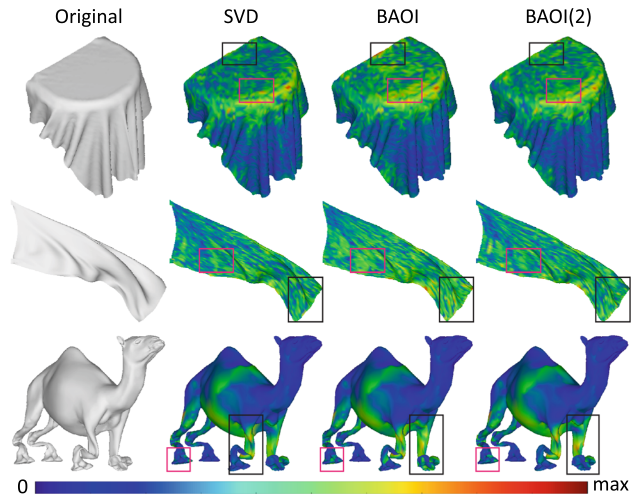

Fig. 7 Heatmap visualization differences of KG between SVD and BAOI for Tablecloth (*top*), Flag (*middle*) and Camel Collapse poses (*bottom*). *Insets* highlight how the severe approximation artifacts are mitigated using one extra OI

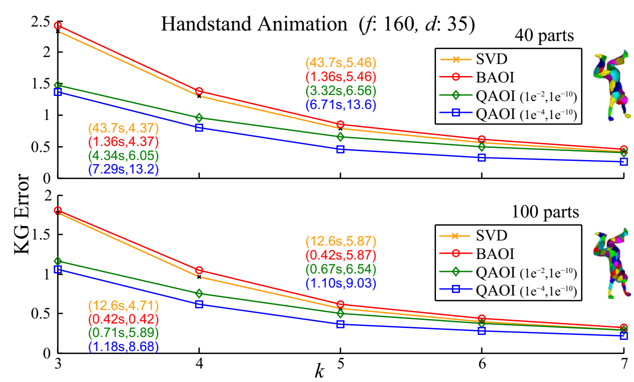

Fig. 8 Approximation as well as time and compression ratio (in *brackets*) for various k using different spatial clusterings on the Handstand animation

when encoding, under a wide compression ratio range (bpvf $= 2, \ldots, 12$), the Flag animation sequence that consists of 1000 frames. First of all, it is highly evident how the reconstruction quality of BAOI converges to the one of the SVD when the number of OI is slightly increased. Furthermore, the block size d should be carefully selected when focusing on the final reconstruction quality since updating the dictionaries less ($d = 250$) or more ($d = 50$) frequently does not guarantee better approximation output. In terms of performance, we observe the superiority of BAOI for all sizes when compared to the SVD method even when more than two OIs are employed. Note that while updating the dictionaries less frequently ($d = 250$) results to a higher performance gain ($21\times$–$52\times$), the total encoding time in each case is proportionally higher (following a linear behavior) than the one performed with a smaller block size ($d = 50$). Similar conclusions can also be derived from Table 1 (Temporal Space).

Impact of iterations/multiplications ($t_{\max}, z$). Figure 6 as well as Table 1 shows how the BAOI closes the gap and finally reaches, in high precision, the levels of reconstruction quality (in both KG and STED metrics) derived by the SVD when increasing the number of OI for a wide range of compression configurations and animations. We further observe that the speedup of BAOI is exponentially decreased when moving to higher OI. As we expected, BAOI^z is faster than BAOI($t_{\max}$) when $t_{\max} = z > 1$, since a matrix multiplication requires less computations time than performing one OI. Heatmap visualizations are also offered showing the distortion alleviation when one extra OI is used (Fig. 7).

Impact of partitioning (***c***): Figure 8 shows the impact on compression quality and performance when uniform geometry clustering the Handstand animation into $c = \{40, 100\}$ parts. First of all, it is clearly shown that the more components mesh is partitioned into, the less distortion artifacts arise (see also spatial space scenarios in Table 1). However, this comes at the cost of sacrificing compression efficiency, since we store the same number of features ($k = \{3, \ldots, 7\}$) for more sub-meshes. On the other hand, it should be noted that processing small independent parts reduces significantly the computational complexity of the reconstruction, allowing at the same time a parallel compression implementation (not

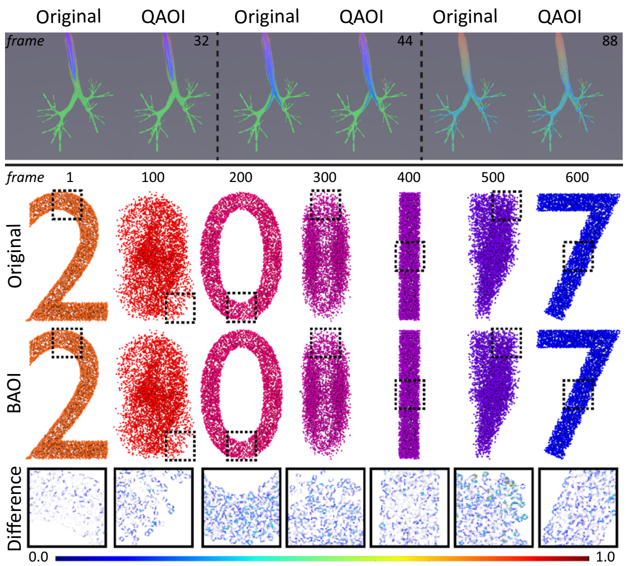

Fig. 9 QAOI (*top*) and BAOI (*bottom*) produce high-quality visual reproduction despite the extreme temporal incoherence between frame blocks in two particle simulations

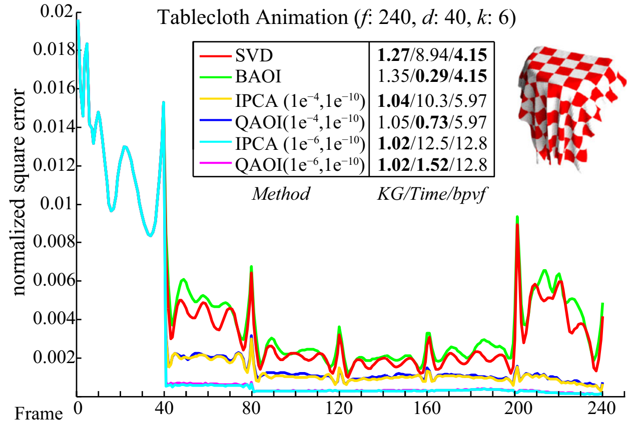

Fig. 10 Observe how fast QAOI reaches the levels of reconstruction quality of IPCA for the entire Tablecloth animation

performed here). Last but not least, the segmentation must not reach high levels in order to maintain the performance gain of BAOI.

Impact of frame incoherence: Figure 9 shows the rich reconstruction quality when QAOI and BAOI ($d = 200$) are employed, respectively, on highly inconsistent point cloud animation generated from (a) the transient analysis of the particles behavior in a patient-specific 3D lung model reconstructed from CT scans ($f = 215$) [12] and (b) morphing from one letter into another forming the word "2017" ($f = 600$). Despite the high motion variance of animated particles, performing dictionary updates with QAOI and BAOI do not impose any noticeable perceptual visual error, generating in the end plausible global illuminated image results with a huge performance speedup (up to 18.64×) as compared to the direct SVD approach.

Impact of error thresholds ($\epsilon_{low}, \epsilon_{high}$). Figure 8 shows that the QAOI scheme allows the user to determine the reconstruction quality (by altering the error bounds) of the Handstand animation, but by deteriorating the compression efficiency and slightly increasing the execution time as compared to the BAOI approach. However, it still achieves up to 10× lower execution times as compared to the SVD approach. Similar conclusions are also drawn from Fig. 10 by observing the per frame normalized square error of the reconstructed Tablecloth animation ($f = 240, d = 40, k = 6$). While BAOI scheme exhibits the fastest execution time and the best compression efficiency, it suffers from high variability in the reconstruction error. On the other hand, QAOI provides a stable reconstruction accuracy (almost identical when compared to the IPCA) that can easily be adjusted by the defined thresholds. However, this comes with a slight increase in the execution time (more OI) as well as a significant increase in the final compression rate (dynamic k).

4.3 Discussion and limitations

The thorough experimental study on a vast collection of 3D dynamic meshes that represent a wide range of animations (ranging from rigid motions to complex simulations, see Table 1 and video) showed that the subspace tracking approaches allow the robust estimation of dictionaries at significantly lower execution times compared to the direct SVD implementations. In general, the BAOI scheme focuses on fast-streaming scenarios, while QAOI approach aims at providing progressively high reconstruction accuracy. Despite the superiority of BAOI when compared to the direct SVD, the initial subspace size should be carefully selected in order to simultaneously achieve the highest reconstruction quality and fastest compression times (Fig. 6). Experiments showed that plausible reconstructed animations can be generated by employing either $t_{\max} = 1$ and $z = 2$ or $t_{\max} = 2$. The execution of more $t_{\max}, z$ increases the computational complexity without affecting noticeably the reconstruction quality (Fig. 6). Finally, the approximation artifacts that occur in a single frame block may slightly increased and propagated when moving to the subsequent ones (Fig. 9). To address the latter issue, we suggest to either re-initialize the subspace of interest using the SVD or execute an QAOI update, when the decoded meshes are detected to drift too far from the original ones.

5 Conclusions

We have introduced two novel approaches to support fast and efficient compression of fully dynamic scenarios with an undefined motion pattern and unknown topology modification behavior. In the heart of our pipeline lies an adaptive PCA-based dictionary estimation stage designed to simultaneously maintain three important criteria: *fast encoding times*, *plausible decoding results* and *out-of-core behavior*. While this stage is general and independent of the underlying compression framework, an extensive analysis has demonstrated the performance superiority of our online variant compared to prior compression solutions, while the reconstruction quality is maintained in high level of detail. Despite the tremendous progress on the landscape of the dynamic compression field, we believe that our approach provides a novel insight at a key area with renewed research interest, where high potential for novel improvements such as dynamic clustering [15,31] is feasible in the near future.

Acknowledgements This work has been supported by the H2020-PHC-2014 RIA project MyAirCoach (Grant No. 643607).

References

1. Alexa, M., Müller, W.: Representing animations by principal components. Comput. Graph. Forum **19**(3), 411–418 (2000)
2. Amjoun, R., Straßer, W.: Efficient compression of 3D dynamic mesh sequences. WSCG **15**(1–3), 99–106 (2007)
3. Cheng, Z.Q., Liu, H.F., Jin, S.Y.: The progressive mesh compression based on meaningful segmentation. Vis. Comput. **23**(9), 651–660 (2007)
4. Comon, P., Golub, G.H.: Tracking a few extreme singular values and vectors in signal processing. Proc. IEEE **78**(8), 1327–1343 (1990)
5. Golub, G.H., Van Loan, C.F.: Matrix Computations. Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore (1996)
6. Hua, Y.: Asymptotical orthonormalization of subspace matrices without square root. Signal Process. Mag. **21**(4), 56–61 (2004)
7. Ibarria, L., Rossignac, J.: Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. In: Proceedings of the 2003 SIGGRAPH/EG Symposium on Computer Animation (SCA'03), pp. 126–135. Aire-la-Ville, Switzerland (2003)
8. Jacobson, A., Deng, Z., Kavan, L., Lewis, J.: Skinning: Real-time shape deformation. In: ACM SIGGRAPH 2014 Courses (2014)
9. Karni, Z., Gotsman, C.: Compression of soft-body animation sequences. Comput. Graph. **28**(1), 25–34 (2004)
10. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1), 359–392 (1998)
11. Kry, P.G., James, D.L., Pai, D.K.: Eigenskin: Real time large deformation character skinning in hardware. In: Proceedings of the 2002 SIGGRAPH/EG Symposium on Computer Animation (SCA'02), pp. 153–159 (2002)
12. Lalas, A., et al.: Numerical assessment of airflow and inhaled particles attributes in obstructed pulmonary system. In: 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 606–612 (2016)
13. Lalos, A., et al.: Compressed sensing for efficient encoding of dense 3D meshes using model-based Bayesian learning. IEEE Trans. Multimed. **19**(1), 41–53 (2017)
14. Lee, D.Y., Sull, S., Kim, C.S.: Progressive 3D mesh compression using MOG-based Bayesian entropy coding and gradual prediction. Vis. Comput. **30**(10), 1077–1091 (2014)
15. Luo, G., Cordier, F., Seo, H.: Compression of 3D mesh sequences by temporal segmentation. Comput. Anim. Virtual Worlds **24**(3–4), 365–375 (2013)
16. Maglo, A., Lavoué, G., Dupont, F., Hudelot, C.: 3D mesh compression: survey, comparisons, and emerging trends. ACM Comput. Surv. **47**(3), 44:1–44:41 (2015)
17. Mamou, K., Zaharia, T., Prêteux, F.: A skinning approach for dynamic 3D mesh compression. Comput. Anim. Virtual Worlds **17**(3–4), 337–346 (2006)
18. Mamou, K., Zaharia, T., Prêteux, F.: TFAN: a low complexity 3D mesh compression algorithm. Comput. Animat. Virtual Worlds **20**(2), 343–354 (2009)
19. Payan, F., Antonini, M.: Temporal wavelet-based compression for 3D animated models. Comput. Graph. **31**(1), 77–88 (2007)
20. Rus, J., Váša, L.: Analysing the influence of vertex clustering on PCA-based dynamic mesh compression. In: Proceedings of the 6th International Conference on Articulated Motion and Deformable Objects (AMDO'10), pp. 55–66. Springer, Berlin (2010)
21. Saad, Y.: Analysis of subspace iteration for eigenvalue problems with evolving matrices. SIAM J. Matrix Anal. Appl. **37**(1), 103–122 (2016)
22. Sattler, M., Sarlette, R., Klein, R.: Simple and efficient compression of animation sequences. In: Proceedings of the 2005 SIGGRAPH/EG Symposium on Computer Animation (SCA'05), pp. 209–217. ACM, NY, USA (2005)
23. Stefanoski, N., Ostermann, J.: SPC: fast and efficient scalable predictive coding of animated meshes. Comput. Graph. Forum **29**(1), 101–116 (2010)
24. Strobach, P.: Fast recursive orthogonal iteration subspace tracking algorithms & applications. Signal Process. **59**(1), 73–100 (1997)
25. Tian, J., et al.: Adaptive coding of generic 3D triangular meshes based on octree decomposition. Vis. Comput. **28**(6), 819–827 (2012)
26. Váša, L., Marras, S., Hormann, K., Brunnett, G.: Compressing dynamic meshes with geometric laplacians. Comput. Graph. Forum **33**(2), 145–154 (2014)
27. Váša, L., Skala, V.: CoDDyaC: Connectivity driven dynamic mesh compression. In: 3DTV Conference, pp. 1–4. IEEE (2007)
28. Váša, L., Skala, V.: COBRA: compression of the basis for PCA represented animations. Comput. Graph. Forum **28**(6), 1529–1540 (2009)
29. Váša, L., Skala, V.: Geometry-driven local neighbourhood based predictors for dynamic mesh compression. Comput. Graph. Forum **29**(6), 1921–1933 (2010)
30. Váša, L., Skala, V.: A perception correlated comparison method for dynamic meshes. IEEE Trans. Vis. Comput. Graph. **17**(2), 220–230 (2011)
31. Vasilakis, A.A., Fudos, I.: Pose partitioning for multi-resolution segmentation of arbitrary mesh animations. Comput. Graph. Forum **33**(2), 293–302 (2014)
32. Vasilakis, A.A., Fudos, I., Antonopoulos, G.: PPS: pose-to-pose skinning of animated meshes. In: Proceedings of the 33rd Computer Graphics International (CGI'16), pp. 53–56. ACM, New York, NY, USA (2016)
33. Xu, G., Kailath, T.: Fast estimation of principal eigenspace using Lanczos algorithm. SIAM J. Matrix Anal. Appl. **15**(3), 974–994 (1994)

Aris S. Lalos received his Diploma degree, his M.A.Sc. degree and his Ph.D. from the Computer Engineering and Informatics Department, University of Patras, Greece, in 2003, 2005 and 2010, respectively. He has been a research fellow at the Technical University of Catalonia (UPC), Barcelona, Spain, from October 2012 to December 2014. In period October 2011–October 2012 was a telecommunication research engineer at Analogies S.A, an early stage start up. He is an author of 51 research papers in international journals (21), conferences (27) and edited books (3). He has participated in several European projects related to the ICT and eHealth domain.

Andreas A. Vasilakis received his Ph.D. on the field of Computer Graphics at the same Department under the supervision of Prof. Ioannis Fudos. He has also received M.Sc. and B.Sc. degrees from the same institution in 2008 and 2006, respectively. After a successful two-year research collaboration with AUEB graphics group working on screen-space global illumination problems, he recently joined CERTH ITI as postdoc researcher. His research interests include character animation, geometry processing and interactive photorealistic rendering. He is also a member of ACM SIGGRAPH and Eurographics Association.

Anastasios Dimas was born in Boston, Massachusetts, in 1989. His undergraduate studies (5-year program) were done at the Computer Engineering & Informatics Department at the University of Patras, Greece, where he received his Diploma in 2013. That same year he started his master studies (2-year program) at the Technical University of Munich, Germany, where he earned his M.Sc. degree in October 2015. In parallel to his master studies, he worked as an intern at the Fraunhofer Institute for Embedded Systems and Communication Technologies for 9 months.

Konstantinos Moustakas is an Associate Professor in the Electrical and Computer Engineering Department, University of Patras, and Head of the Visualization and Virtual Reality Group. His main research interests include virtual, augmented and mixed reality, 3D geometry processing, haptics, virtual physiological human modeling, information visualization, and physics-based simulations. During the latest years, he has been the (co)author of more than 130 papers in refereed journals, edited books, and international conferences. He is the coordinator of the GameCar Horizon2020 project and scientific coordinator of the NoTremor FP7 project. He is a senior member of the IEEE, the IEEE Computer Society and Eurographics.