

# Eigenspace compression: dynamic 3D mesh compression by restoring fine geometry to deformed coarse models

Mohammadali Hajizadeh<sup>1</sup> · Hossein Ebrahimnezhad<sup>1</sup>

Received: 24 March 2017 / Revised: 10 September 2017 / Accepted: 5 November 2017 /

Published online: 14 November 2017

© Springer Science+Business Media, LLC, part of Springer Nature 2017

**Abstract** Dynamic 3D mesh compression is of great practical important issues in computer graphics and multimedia applications. In this paper, an efficient compression algorithm is proposed to represent animated mesh sequences in a compact way, so that the storage and transmission of dynamic 3D meshes can be accomplished efficiently. The focus of this paper is on the animated mesh sequences with shared connectivity. The proposed method first computes coarse models (low frequency modes) of the animated sequence using the graph Laplacian matrix. Obtained coordinate weights are used at the decoder to reconstruct the coarse models of the sequence. Then, a novel approach is proposed to extract fixed details (high frequency modes or finer features) of the animated mesh. Finally, a details restoration process is applied at the decoder to add details back to the coarse models of the reconstructed sequence. The superiority of the proposed method to the current state of the arts is demonstrated in terms of low data rates for a given degree of perceived distortion.

**Keywords** Animated 3D mesh compression · Dynamic geometry · Graph laplacian · Fixed details · Coarse model · Details restoration

## 1 Introduction

Computer graphic is one of the most appealing areas for computer sciences and multimedia applications that several domains such as medicine, engineering, games, special effects in movies and animation can get benefits from its developments. Computer animation is one of the most important and attractive topics in computer graphics and multimedia entertainment

---

✉ Hossein Ebrahimnezhad  
ebrahimnezhad@sut.ac.ir

Mohammadali Hajizadeh  
m\_hajizadeh@sut.ac.ir

<sup>1</sup> Computer Vision Research Laboratory, Department of Electrical Engineering, Sahand University of Technology, Tabriz, Iran

applications with significant impact on animation, movies and video game industry, flight simulation, marketing, scientific visualization, etc. [3, 22, 30].

A 3D animation (also referred as dynamic 3D mesh/ animated 3D mesh/ 3D mesh sequence) is stored as consecutive frames distributed over time, each of which represents a static 3D mesh. The complexity and size of 3D animations have been increased due to technological advances in computer animation and growing demand for more realistic 3D models. Therefore, compression schemes are required for efficient storage, processing and transmission of dynamic 3D meshes.

Static 3D mesh compression has been widely explored over the last two decades which makes this field very mature. The reader is referred to [31, 34, 41] for detailed survey. Static 3D mesh compression techniques benefit from spatially high correlated adjacent vertices in order to find a compact representation of the static 3D mesh.

Dynamic 3D mesh compression is an important topic that has attracted significant attention in recent years. Most of the compression approaches focus on the animated mesh sequences with shared connectivity. Thus, the main objective of dynamic 3D mesh sequence compression is to remove the temporal correlation between consecutive frames in order to achieve a compact representation of the animated mesh sequence. It is theoretically possible to simply apply one of the prevalent static geometry compression methods (see [34, 41] for detailed survey) to compress the frames of an animated sequence, separately. While most of the static mesh compression techniques utilize only spatial correlation between the positions of neighboring vertices in a frame, the temporal correlation existing between adjacent frames of the sequence should be also taken into account in dynamic 3D animation compression. The previous works on dynamic 3D mesh compression can be roughly categorized into four groups [38]: *segmentation based techniques*, *PCA based techniques*, *wavelet based techniques* and *prediction based techniques*.

The first attempt at dynamic 3D mesh compression in the segmentation based approaches has been reported by Lengyel [26]. In this approach, the animated mesh sequence is initially segmented into sub-meshes and a rigid body transformation is obtained as a description for the motion of each sub-mesh. Then, a compact representation of the animated mesh sequence is achieved by encoding the parameters of the transformations, the base sub-meshes, and the approximation residuals. Ahn et al. [1] employed Discrete Cosine Transform (DCT) for encoding the residual errors. Zhang and Owen [62, 63] proposed an octree-based representation to analyze the motions between successive frames of the animated mesh sequence. This approach applies a trilinear interpolation on the generated set of motion vectors for each cell in order to approximate the motion of the vertices. Muller et al. [36, 37] introduced a rate-distortion (RD) optimization method to determine the best prediction mode among three pre-defined modes, namely trilinear interpolation, mean replacement and direct encoding as well as the clustering structure. Mamou et al. [32] utilized a skinning animation model for dynamic 3D mesh compression. The frame-wise motion of previously segmented patches is estimated using an affine motion model. Subsequently, the movements of adjacent patches are utilized in a weighted linear combination analysis to approximate the frame-wise motion of each vertex.

In their pioneering work, Alexa and Muller [2] introduced a compression algorithm based on the principal component analysis (PCA). In this work, each frame of the animated mesh sequence is stored in a column of the matrix  $A$ . A set of basis vectors is created by applying singular value decomposition (SVD) on the *covariance* matrix ( $AA^T$ ). PCA coefficients are defined for each frame as the projected values onto these basis vectors. Since the contribution of each basis vector to the reconstruction quality is defined by its corresponding PCA

coefficients, a subset of the most important basis vectors is selected to encode the animated mesh sequence in a compact way. Karni and Gotsman [24] extended this approach by encoding the PCA coefficients with a second-order linear prediction coding (LPC) which results in a reduced code size by exploiting the temporal coherence between neighboring frames. Sattler et al. [44] proposed to apply PCA coding to the meaningful clustered parts of the model which are achieved by analyzing the trajectories of all vertices. Amjou et al. [4, 7] expressed each vertex trajectory in a local coordinate system in order to analyze the model clusters. Moreover, required PCA coefficients for each cluster are encoded using a bit allocation procedure to achieve the desired precision. A signal-to-noise ratio (SNR) and temporal scalable PCA coding scheme is introduced by Heu et al. [19]. The compact representation of a mesh sequence is accomplished in a bit plane coder by encoding the basis vectors extracted via Singular value decomposition (SVD). The Bit planes are transmitted in a decreasing order of their contribution to the reconstruction quality. Vasa and Skala proposed to incorporate a PCA step into an Edge-Breaker-like [43] predictor. In their first method called the Coddyc algorithm [51], Vasa and Skala proposed to encode the PCA coefficients by using a parallelogram prediction which leads to better performance compared to the clustering-based methods. They introduced vertex decimation as a part of the compression process in their next work [52]. The encoder computes the accuracy of predictors in order to control the decimation process, making this approach well suited for interchanging predictors. In addition, the authors reported great improvements in [53] by introducing an efficient method to compress the PCA basis. Finally, the authors proposed two geometric predictors [54] which result in better performance and are particularly suitable for the PCA based compression approaches. More accurate predictions are achieved by the exploitation of the geometrical interpretation of the data.

The first compression approach using wavelet transform was introduced by Guskov and Khodakovskiy [16]. They applied an anisotropic wavelet transform on top of a progressive mesh hierarchy and encoded the differences between corresponding wavelet coefficients in order to exploit the temporal coherence present in the animated sequence. A temporal wavelet based coder was proposed by Payan and Antonini [39, 40] in which a bit allocation process is utilized to optimize the quantization of the obtained wavelet coefficients. Boulfani-Cuisinaud and Antonini [11] segmented the input mesh vertices into the groups sharing the same affine motion and then, filtered each group by using a scan-based temporal wavelet transform. Geometry video [12] is a new data structure to represent a 3D mesh as a geometry image. In this method, every frame is sampled into a geometry image. Geometry video is the sequence of geometry images in which the first image is intra-coded and the subsequent images are encoded by using a predictive coder with affine motion compensation. This algorithm was later extended by Mamou et al. [33] which resulted in significant improvements. In [45], the geometry image-based compression method was improved by exploiting the strong geometric correlation between geometry image and normal-map one. Dhibi et al. [14] proposed a 3D mesh compression algorithm for 3D deforming objects in which Multi Library Wavelet Neural Network (MLWNN) is utilized to align the mesh features and minimize the distortion.

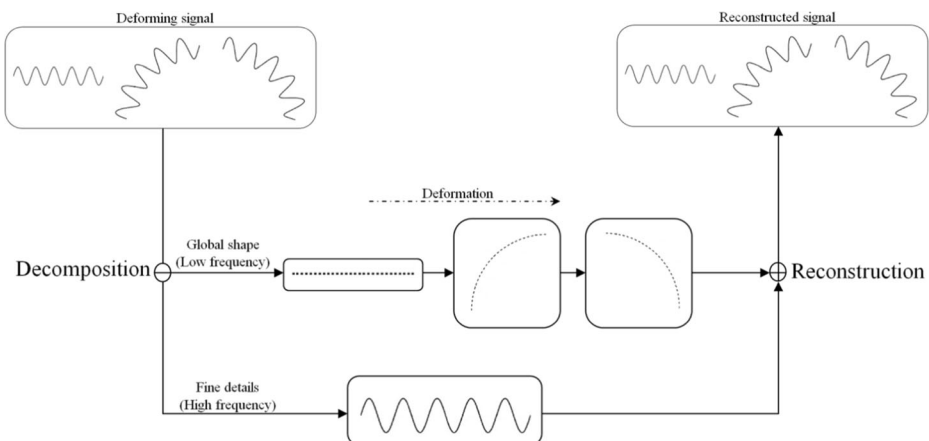
Yang et al. [61] presented a vertex-wise motion vector (MV) predictor to compress an animated mesh sequence in two steps. In the first step, the prediction of a vertex MV is accomplished by using the high spatial correlation between MVs of neighboring vertices. In the second step, the authors proposed a rate distortion (RD) optimization technique for spatial or temporal prediction of the error vectors. Ibarria and Rossignac [21] utilized a region growing based algorithm to determine the encoding order of the mesh vertices. Then, two

extrapolating space-time predictors (the Replica predictor and the ELP extension of the Lorenz predictor) were proposed to predict the location of each vertex using three of adjacent vertices in the current and previous frames. In [5], Amjoun and Strasser introduced a spatial predictive DCT coding to make their previous approach [4] appropriate for real time applications. A connectivity-guided predictive method [6] was proposed by the authors for single-rate compression of an animated mesh sequence. In this approach, the prediction errors are encoded in a local coordinate system which includes a normal and two tangential components. Another connectivity-guided predictive method was proposed by Stefanoski and Ostermann [47]. The main assumption in the presented angle-preserving predictor is that the dihedral angle between neighboring triangles does not change from frame to frame. Then, stefanoski et al. [50] established a linear predictor to incorporate spatial scalability in compressing the animated sequence. Animation frames are decomposed into spatial layers by applying the Patch-based mesh simplification algorithms. This method is enhanced in [48] by adding temporal scalability to the algorithm. The authors introduced SPC (Scalable Predictive Codec) as their final compression method [49]. In this method, animation frames are decomposed into spatial and temporal layers. In order to compensate local rigid motion, a space of rotation-invariant coordinates is utilized to perform the prediction. The authors benefited from fast binary arithmetic coding [35] to encode the quantized prediction errors. Bici and Akar [38] introduced three prediction structures to improve the SPC method. These prediction structures can be also utilized in other prediction based methods. Vasa et al. [56] proposed to compute an average mesh of the whole animated mesh sequence in an edge shape space. Then, the trajectories of the mesh vertices are encoded by applying a discrete geometric laplacian on the computed average mesh. A spatio-temporal predictor can be optionally used in order to improve the compression performance. In [17], a key-frame based algorithm is proposed to compress animated mesh sequence. Optimum blending weights of the key-frames are determined using a multi-objective optimization algorithm to predict the position of the vertices in non-key frames.

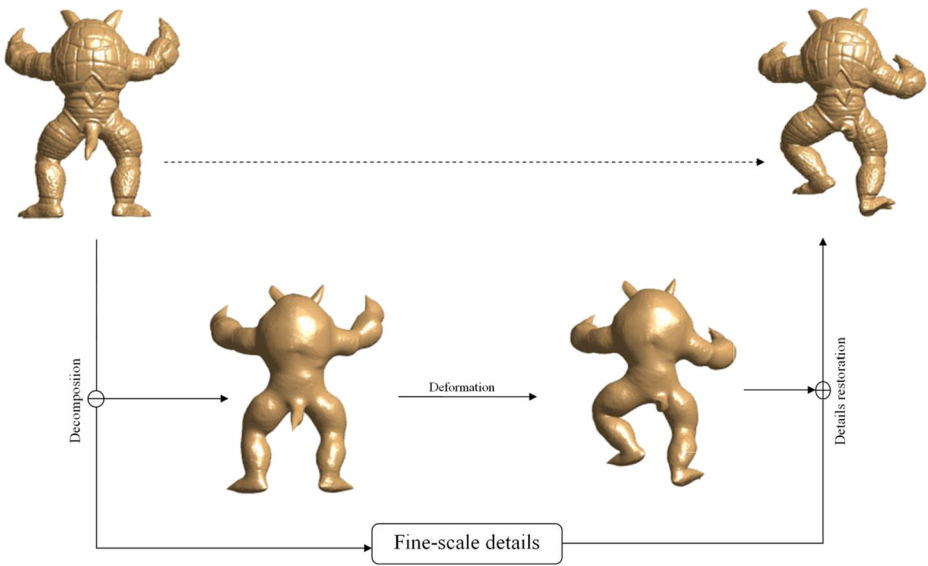
Recently, low-rank matrix recovery has found applications in 3D model processing. Matrix recovery is the problem of predicting missing entries in a partially observed matrix. It is also known as Robust PCA (robust principal component analysis) and was introduced in [59] for the first time. The main idea is to exploit the structural information of the matrix to recover the missing entries. Low-rank matrix recovery is a variant of the matrix completion problem which looks for the lowest rank matrix to recover the partially observed matrix from few linear measurements. Lu et al. [29] proposed to use a matrix recovery technique to restore the original shapes of a group of similar 3D objects. In [15], low-rank matrix completion is utilized to reconstruct 3D shapes of the objects in a video which have rigid and non-rigid motions. Yan et al. [60] proposed an efficient method to solve maximum margin matrix factorization (MMMF) for large scale data sets. The authors formulated MMMF as the Riemannian optimization problem and efficiently solved it by introducing a block-wise nonlinear Riemannian conjugate gradient algorithm. Wang et al. [58] introduced an efficient method to recover 3D motion by using a low-rank matrix recovery model. In [25], a low-rank matrix factorization based algorithm is proposed to recover the 3D structure of an object from a monocular video stream. Lin et al. [28] presented a factorization algorithm for the problem of structure from motion. They applied truncated nuclear norm to solve the factorization problem in a non-convex way.

Our observation is that most of the compression methods focus on animated mesh sequences with shared (fixed) connectivity. The main reason is that in animation techniques,

fixed connectivity allows for simple and efficient manipulation of the animated sequence. Moreover, most of the animated mesh sequences include 3D models which have fixed fine details throughout the animation. Thus, our aim is to exploit these features of animated mesh sequences to propose an efficient compression method. The main idea of this paper is to split the geometry of a 3D model into coarse and detail information. In an animated mesh sequence, a 3D model deforms over time. During the deformation, the global shape of the model changes smoothly, whereas the fine-scale details are generally fixed. Thus, the meaningful part of deformation data corresponds to the changes of the global shape. By removing fixed details from the animated 3D model, it is possible to significantly reduce the required data for representing the animated mesh sequence. The problem of dynamic 3D mesh compression has not been investigated from this point of view. With this motivation, we propose to decompose an animated 3D model into two frequency bands: low frequency band and high frequency band. Low frequencies correspond to global shape, while high frequencies represent fine details. Our goal is to represent the animated 3D model as the deformation of low frequencies (global shape or coarse model) while preserving high frequencies (fine details). This concept is illustrated in Fig. 1 with a simple 2D example. Decomposition of a deforming mesh into coarse and detail information is illustrated in Fig. 2. First, a smooth base surface (coarse model) is computed by removing the fine details. This coarse model is the low-frequency mode of the deforming mesh which can be represented by less data. The removed fine details represent the high-frequency modes which are extracted and stored as detail information. Thus, the animated mesh is reduced to the deforming coarse model and the extracted fine details. By restoring the fine details to the coarse model, we can reconstruct the original animated mesh. During the animation, only the coarse model deforms and the detail information is locally fixed. So, the deformation data of the animated sequence can be preserved and represented by the coarse model with less data. To achieve this end, we use the graph laplacian matrix and remove the detail information from a 3D animated model by truncating the number of eigenvectors in Eigen framework of the graph laplacian matrix. The graph laplacian matrix is a *combinatorial* laplacian and thus, depends only on the connectivity of the animated model which is fixed throughout the sequence and should be sent one time to



**Fig. 1** Decomposition of a deforming sine wave into two frequency bands. The frequency decomposition results in the dashed line as the global shape (low frequency component). By deforming the global shape and adding the fine details (high frequency component) onto it, we can represent the original signal in a simple way



**Fig. 2** Representation of an animated mesh as a deforming coarse model and fine details. By restoring the fine details to the coarse model, we can reconstruct the original animated mesh

the decoder. A simple and efficient algorithm is proposed to extract the removed fixed details. Since these details are fixed during the sequence, they are also sent to the decoder one time for total frames. In this way, we can significantly reduce the size of the required data to reconstruct the animated sequence. At the decoder, the coarse model for each frame is firstly computed by using the coefficients of the graph laplacian. Then, fixed details are restored to the coarse models in order to reconstruct the original animated mesh sequence.

The rest of the paper is organized as follows. Detailed description of the proposed method for dynamic 3D mesh compression is presented in section 2. We present and discuss the results of our experiments in section 3 and finally, we conclude the paper in section 4.

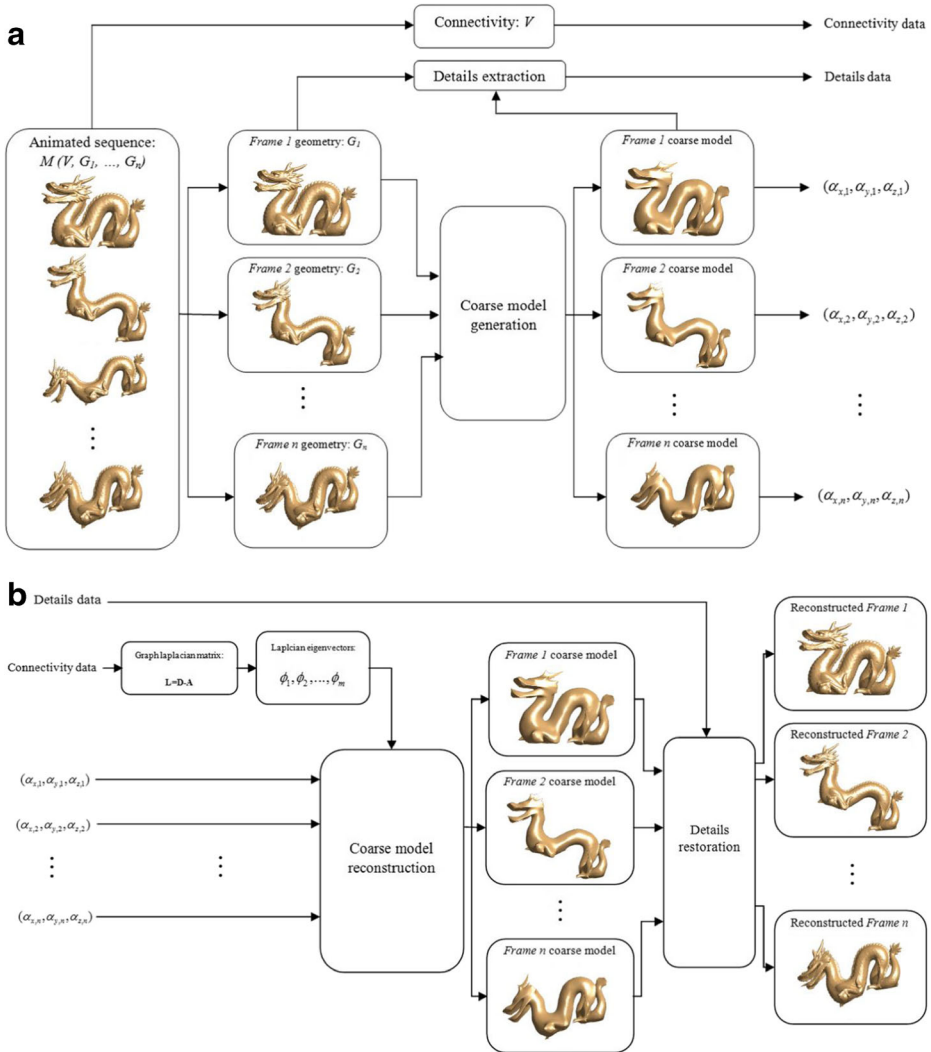
## 2 Eigenspace compression

The main idea behind the proposed method is described in this section. We assume that the input is an animated sequence of  $f$  triangular meshes  $M_1, \dots, M_f$  with same connectivity. It is assumed that the shared connectivity is encoded once, using any state-of-the-art algorithms.

The location of the  $i^{th}$  vertex in the  $j^{th}$  mesh is indicated by a vector of coordinates  $v_{i,j}$

$$= \left( v_{i,j}^x, v_{i,j}^y, v_{i,j}^z \right), i = 1, \dots, n, j = 1, \dots, f.$$

The proposed compression method includes several steps which are discussed in detail in the following subsections. A graphical abstract of the proposed method is shown in Fig. 3. First, the connectivity of the animated mesh sequence is utilized to compute the graph laplacian operator in the form of a matrix  $L \in \mathbb{R}^{n \times n}$ . Since the connectivity is encoded using one of the standard methods and sent to the decoder, this matrix can also be built at the decoder. The eigenvectors of the graph laplacian matrix are computed in order to form an orthonormal basis. The coarse model of each frame is generated by representing the geometry



**Fig. 3** Graphical abstract of the proposed method. (a) block diagram of the encoder (b): block diagram of the decoder

of each frame using a reduced number of the eigenvectors. In this way, the fixed fine details (also called high frequency modes) of the 3D model are filtered and removed. The next step is to extract the fixed details of the 3D model.

Detail extraction is accomplished by using a simple and efficient approach which is described in more detail later. Since these details are almost fixed through the animated sequence, they are encoded and sent to the decoder just once. In order to improve the reconstruction quality, residual errors for vertices with large prediction errors are also encoded and sent to the decoder. At decoder side, the coarse model of each frame is firstly computed using the eigenvectors of the graph laplacian matrix and the required data received from the encoder. Then, fixed details are added back to the coarse models. The final sequence is achieved by adding the residual errors to the reconstructed models.

## 2.1 Approximation-phase

### 2.1.1 Laplace-Beltrami operator

In differential geometry, the Laplace-Beltrami operator is the generalized form of the Laplace operator which is applied to the functions defined on surfaces in Euclidean space. Assume that  $N$  is a smooth, compact surface which is isometrically embedded in  $\mathbb{R}^3$ . For a twice differentiable function  $g : N \rightarrow \mathbb{R}$ , the Laplace-Beltrami operator  $\Delta_N$  applying to  $g$  is defined as the divergence of the gradient of  $g$ :

$$\nabla^2 g = \nabla \cdot \nabla g \quad (1)$$

The Laplace-Beltrami operator has been widely used as an effective tool in many geometric processing applications due to its useful properties [27, 64]. The Laplace-Beltrami operator is known as an intrinsic geometric quantity and the manifolds with isometric deformation share the same laplacian. Hence, the eigenfunctions and eigenvalues of the Laplace-Beltrami operator are invariant under isometric transformations. In other words, isometric manifolds have the same Laplacian, which makes the Laplace-Beltrami operator profitable to describe or capture isometric deformations. The eigenfunctions of the Laplace-Beltrami operator form an orthogonal basis for the space of square integrable functions defined on  $N$ . Similar to Fourier analysis for functions on a circle, Laplacian eigenfunctions with smaller eigenvalues are correlated to the low frequency modes (coarser features) of the manifold, while those with larger eigenvalues represent high frequency modes that characterize the details (finer features) of the input manifold  $M$ .

In our problem, the input is a triangular mesh approximating a surface of an object which can be regarded as a discrete function of vertex positions together with connectivity information. Therefore, we need a discrete form of the Laplace operator computed from the mesh. Different discretizations of the Laplacian are available in the literature [8, 20, 42, 64]. In this paper, we employ the graph Laplacian matrix [18, 64] as a combinatorial Laplacian whose coefficients are only based on connectivity information. The graph Laplacian  $\mathbf{L} = (l_{i,j})_{n \times n}$  is computed from the input mesh connectivity as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (2)$$

Where,  $\mathbf{A}$  is  $n \times n$  adjacency matrix obtained from the input mesh connectivity in which  $a_{ij}$  indicates whether or not the vertices  $v_i$  and  $v_j$  are connected by an edge, and  $\mathbf{D}$  represents  $n \times n$  matrix of vertex degrees (number of edges that originate from a vertex), such that:

$$(l_{i,j}) = \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise} \end{cases}$$

Since  $\mathbf{L}$  is a real and symmetric matrix, its eigenvectors constitute a complete set of unit norm *orthogonal* vectors with real, non-negative corresponding eigenvalues. Thus, we have an *orthonormal basis* for the input mesh. Similar to the work in [23], the computed eigenvalues are arranged in ascending order of their magnitude, and the respective eigenvectors are arranged in the same order. As discussed in [23], when the graph Laplacian is compared with the Discrete Fourier Transform (DFT), we see that the eigenvectors of the graph Laplacian and their respective eigenvalues are analogous to the basis functions and their frequencies used in



DFT, respectively. This means that the eigenvectors with smaller eigenvalues are related to the low-frequency basis vectors and the ones with larger eigenvalues effectively represent the high-frequency basis vectors. An important difference is that the actual Fourier basis functions are fixed whereas the basis vectors from graph Laplacian matrix change according to the input mesh connectivity.

The motivation for using the graph Laplacian matrix for discretization of the Laplacian is mostly attributed to the work by Karni and Gotsman [23], as well as to the work [9]. Karni and Gotsman [23] were the first to propose an efficient progressive compression method for static mesh geometry using the eigenvectors of a combinatorial mesh Laplacian matrix. Ben-Chen and Gotsman [9] subsequently proved that the eigenvectors of the graph Laplacian matrix construct an *optimal* basis for the spectral decomposition of the specified types of geometric mesh models. Optimal, here, means that most of the spectral energy of the geometric mesh model can be captured by a given number of leading eigenvectors which are arranged by their respective eigenvalues in ascending order. This was theoretically proved in [9] for 1D and 2D connected meshes and demonstrated empirically for meshes in three dimensions (i.e. embedded in  $\mathbb{R}^3$ ). The key part of the proof for 2D meshes states that for a given mesh connectivity and by assuming a normal distribution for all appropriate geometries that fit to this connectivity, the symmetric Laplacian matrix of the given connectivity and the inverse covariance matrix of the normal distribution of the appropriate geometries are equivalent up to a constant factor. In the other words, the eigenvectors of the Laplacian matrix are identical to the eigenvectors of the covariance matrix in reversed order.

Since, for random vectors from a zero-mean normal distribution, the Principal Component Analysis computed from the covariance matrix is the optimal orthogonal transform from the viewpoint of mean-squared error criterion [9], the Laplacian eigenvector basis leads to an optimal spectral decomposition for appropriate 2D mesh geometries with the same kind of distribution. The authors in [9] demonstrated that these assumptions are also acceptable for the true distribution of valid 3D mesh geometries and in practice, they are usually good enough. This demonstration is based on this simple observation that a vertex position of an actual 3D mesh is approximately close to the average of its neighbours, therefore we have a relatively smooth mesh in practice.

By considering the Laplace operator  $\Delta_N$  of an input manifold, we denote the eigenvectors of  $\Delta_N$  by  $\phi_1, \phi_2, \dots$ . These eigenvectors construct a basis for the family of square-integrable functions on  $N$  which is denoted by  $L^2(N)$ . Therefore, we can rewrite any function  $g \in L^2(N)$  in terms of  $\phi_i$  as  $g = \sum_{i=1}^{\infty} \alpha^i \phi_i$ , where  $\alpha^i$  is the inner product of  $g$  and  $\phi_i$  ( $\alpha^i = \langle g, \phi_i \rangle$ ) in the functional space  $L^2(N)$ . In this way, we can consider the function  $g$  as a vector  $\alpha = [\alpha^1, \alpha^2, \dots]$  in the infinite-dimensional space spanned by the Laplacian eigenvectors. Now, we consider the coordinate functions ( $g_x, g_y, g_z$ ) defined on  $N$  each of which simply contains the  $x, y$  and  $z$ -coordinate values of the vertices, respectively. By re-writing these coordinate functions, a surface can be represented in terms of three vectors ( $\alpha_x, \alpha_y, \alpha_z$ ) in the space spanned by the Laplacian eigenvectors. We denominate these vectors the coordinate weights of  $N$  as in [13]. We can entirely determine the embedding of a manifold by its coordinate weights once the eigenvectors are available. Finally, since higher eigenvectors have higher frequencies which are related to smaller details, we can truncate the number of the eigenvectors to keep only the top few coordinate weights. In this way, we can reconstruct the surface at varying levels of details.

### 2.1.2 Coarse model generation

Given an initial surface mesh with  $n$  vertices, we firstly compute the graph Laplacian using Eq. (2), and then calculate the  $n$  eigenvectors of the graph Laplacian (i.e.  $\phi_1, \phi_2, \dots, \phi_n$ ). These eigenvectors are arranged in ascending order of the eigenvalues. In order to generate a coarse model of the initial surface mesh, the fine details should be removed from the model. To this end, we restrict our consideration to the first  $m$  ( $m < n$ ) eigenvectors of the graph Laplacian. In this way, we develop a higher-level abstraction of the surface mesh that captures its low frequency modes (coarser features). Specifically, consider that  $G^{coarse} = \{v_1^{coarse}, v_2^{coarse}, \dots, v_n^{coarse}\}$  indicates the vertex set of the coarse model generated from the initial surface mesh vertices  $P^{org} = \{v_1^{org}, v_2^{org}, \dots, v_n^{org}\}$  using only the first  $m$  eigenvectors ( $\phi_1, \phi_2, \dots, \phi_m$ ) of the graph Laplacian. That is, if we define the coordinate functions  $(\hat{g}_x, \hat{g}_y, \hat{g}_z)$  as:

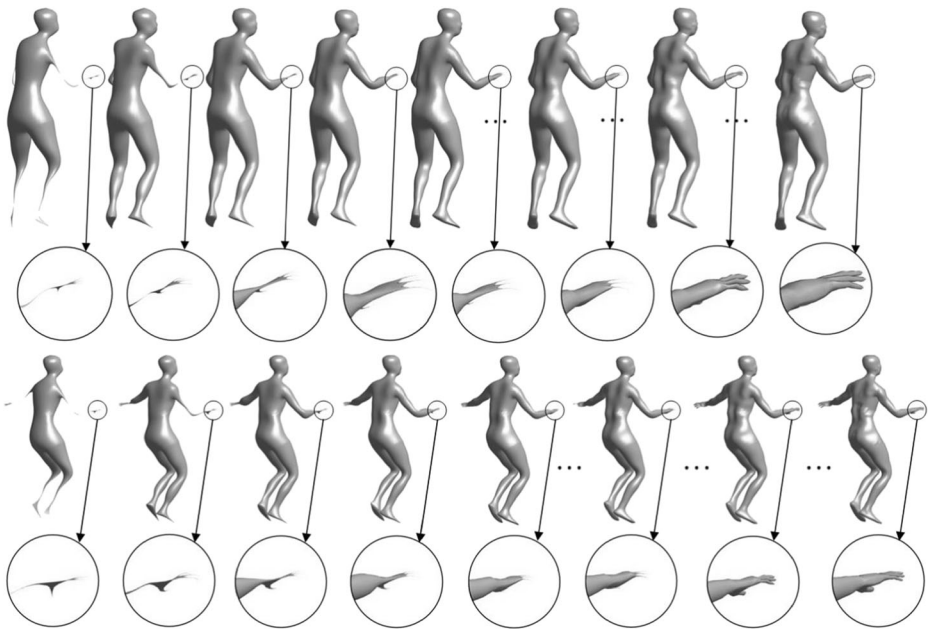
$$\hat{g}_x = \sum_{i=1}^m \alpha_x^i \phi_i, \hat{g}_y = \sum_{i=1}^m \alpha_y^i \phi_i, \hat{g}_z = \sum_{i=1}^m \alpha_z^i \phi_i \quad (3)$$

Then, we can calculate the vertices of the coarse model as  $v_i^{coarse} = \{\hat{g}_x(v_i), \hat{g}_y(v_i), \hat{g}_z(v_i)\}$  for  $i = 1, \dots, n$  with the same connectivity as the input mesh. The coarse models are obtained by calculating the coordinate functions using only the first few eigenvectors  $m$  ( $m < n$ ) of the Laplacian (Eq. (3)). The coarse models are the high-level abstractions of the original model. In other words, they represent the low frequency modes (global shape) of the original models. The level of the abstraction depends on the value of  $m$  i.e. the number of the eigenvectors which is utilized to calculate the coordinate functions. Thus, by using different values of  $m$ , we can create the coarse model at different levels of detail. Figure 4 illustrates the generated coarse models for sample frames (frames 1 and 100) of the dance test sequence. The coarse models are obtained for different values of  $m$ . The last model on the right shows the original model. It can be seen from this figure that by reducing the number of utilized eigenvectors in calculating the coordinate functions, the fine details are filtered more strongly and the generated model becomes coarser. Hand region is magnified to illustrate how the fine details are filtered and removed from fingers by representing the mesh models using reduced number of the eigenvectors of the graph laplacian matrix. The level of the filtration has an important effect on the extraction of fixed details and the quality of the reconstructed mesh sequence. Strong filtration of the fine details causes the coarse model to become very thin near the extremities, with a lot of fine details collapsing together. This results in poor normal estimation around sharp fine details. Thus, the optimal value of  $m$  is determined by considering the trade-off between compression ratio and reconstruction quality.

## 2.2 Details- phase

### 2.2.1 Details-extraction

The extraction of fine details (high frequency modes) is a necessary step in order to reconstruct the original mesh from its coarse model. In this paper, we consider that the animated mesh contains fixed details which are roughly constant during object deformation in the animated mesh sequence. As a consequence, the extraction of detail information is performed just for a reference frame. Here, we select the first frame of the sequence as the reference frame.



**Fig. 4** Generated coarse models for *Frame 1* (top) and *Frame 100* (bellow) of *dance* sequence. (From left to right: generated coarse models for  $k = 1, 2, 3, 4, 5, 8$  and  $15\%$  ( $m = kn/100$ ). The last model shows the original frame). It is clear how details filtered from the model by reducing the number of utilized eigenvectors

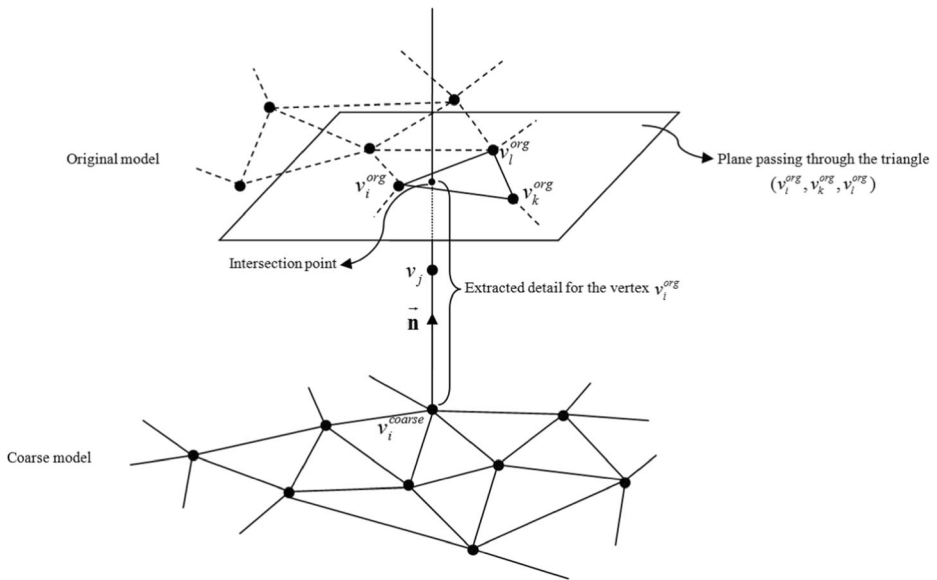
Extracted details will be added back to the reconstructed coarse model of each frame to reconstruct the whole animated sequence.

Details are calculated and extracted at each vertex of the reference frame. To explain the extraction process, assume that  $v_i^{coarse}$  is the corresponding vertex of  $v_i^{org}$  on the coarse model. There is a displacement between  $v_i^{coarse}$  and  $v_i^{org}$  due to removing some Eigenvectors of the graph laplacian matrix. Definition of the details as the displacement between the positions of the vertex on the original and the coarse model makes the details restoration a challenging and time-consuming process. Since the model is deforming through time, the displacement vectors should be represented in a rotation-invariant local coordinate frames around the vertices. In order to avoid the time-consuming process of computing the local coordinate frames like the method presented in [50], we present a simple and efficient approach to extract the details. In this approach, the details are extracted as the required displacement along the normal direction of the vertex on the coarse model through which the vertex is relocated on the surface of the original model. Figure 5 illustrates the proposed approach.

In this figure, to define the required displacement at each vertex, we find the intersection point between the normal line passing through the vertex and the surface of the original model. The normal line at the vertex  $v_i^{coarse}$  can be described as:

$$v_i^{coarse} + (v_j - v_i^{coarse})t, t \in \mathbb{R} \tag{4}$$

Where  $v_i^{coarse} = (x_i^{coarse}, y_i^{coarse}, z_i^{coarse})$  and  $v_j = (x_j, y_j, z_j)$  is another point along the line which can be obtained by adding the normal vector  $\vec{n}$  to  $v_i^{coarse}$  as follows:



**Fig. 5** Detail extraction for a typical vertex

$$v_j = v_i^{coarse} + \vec{n} = (x_i^{coarse} + n_x, y_i^{coarse} + n_y, z_i^{coarse} + n_z) \tag{5}$$

Since we assume triangular meshes in this paper, the normal line will intersect with a triangle on the surface of the original model. Experiments reveal that most of the vertices have a displacement along the near normal direction. This fact implies that the intersection point will be inside the neighboring faces of the vertex. Therefore, we restrict our consideration to the one-ring neighborhood of the vertex. If we consider the triangle with vertices  $(v_i^{org}, v_k^{org}, v_l^{org})$  in the one-ring neighborhood of the vertex  $v_i^{org}$  then, the general representation of the plane passing through these three vertices is:

$$v_i^{org} + (v_k^{org} - v_i^{org})u + (v_l^{org} - v_i^{org})w, u, w \in \mathbb{R} \tag{6}$$

Where,  $v_c^{org} = (x_c^{org}, y_c^{org}, z_c^{org})$  for  $c = i, k, l$ .

The intersection point of the normal line and this plane is therefore described by setting the line Eq. (4) equal to the plane Eq. (6):

$$v_i^{coarse} + (v_j - v_i^{coarse})t = v_i^{org} + (v_k^{org} - v_i^{org})u + (v_l^{org} - v_i^{org})w \tag{7}$$

Equation (7) can be rewritten as:

$$v_i^{coarse} - v_i^{org} = (v_i^{coarse} - v_j)t + (v_k^{org} - v_i^{org})u + (v_l^{org} - v_i^{org})w \tag{8}$$

which can be expressed in a matrix form as follows:

$$\begin{bmatrix} x_i^{coarse} - x_i^{org} \\ y_i^{coarse} - y_i^{org} \\ z_i^{coarse} - z_i^{org} \end{bmatrix} = \begin{bmatrix} x_i^{coarse} - x_j & x_k^{org} - x_i^{org} & x_l^{org} - x_i^{org} \\ y_i^{coarse} - y_j & y_k^{org} - y_i^{org} & y_l^{org} - y_i^{org} \\ z_i^{coarse} - z_j & z_k^{org} - z_i^{org} & z_l^{org} - z_i^{org} \end{bmatrix} \begin{bmatrix} t \\ u \\ w \end{bmatrix} \tag{9}$$

This problem can be solved by inverting the matrix in order to find the values of the variables  $t$ ,  $u$  and  $w$ :

$$\begin{bmatrix} t \\ u \\ w \end{bmatrix} = \begin{bmatrix} x_i^{coarse}-x_j & x_k^{org}-x_i^{org} & x_l^{org}-x_i^{org} \\ y_i^{coarse}-y_j & y_k^{org}-y_i^{org} & y_l^{org}-y_i^{org} \\ z_i^{coarse}-z_j & z_k^{org}-z_i^{org} & z_l^{org}-z_i^{org} \end{bmatrix}^{-1} \begin{bmatrix} x_i^{coarse}-x_i^{org} \\ y_i^{coarse}-y_i^{org} \\ z_i^{coarse}-z_i^{org} \end{bmatrix} \tag{10}$$

If the obtained solution satisfies the following condition:

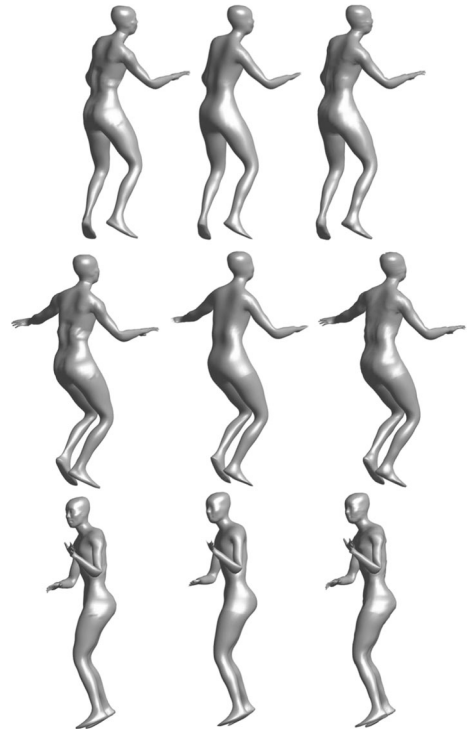
$$u, w \in [0, 1], (u + w) \leq 1 \tag{11}$$

then the intersection point will be on the plane inside the triangle spanned by the vertices  $v_i^{org}$ ,  $v_k^{org}$  and  $v_l^{org}$ . In this case, the normal line intersects the surface of the original mesh at the obtained intersection point. Otherwise, this intersection point is not on the surface of the original mesh and we should investigate the next triangle in the one-ring neighborhood of the vertex. Once the intersection point between normal line and the surface of the original mesh is obtained, the distance from the vertex to the intersection point is stored as the detail at that vertex. In fact, this distance is actually the required displacement to relocate the vertex on the surface of the original mesh.

### 2.2.2 Details-restoration

Details restoration is a process to reconstruct the mesh sequence from compressed data which is sent to the decoder. These data include the fixed connectivity of the animated 3D model ( $V$ ), coordinate weights vectors  $(\alpha_x, \alpha_y, \alpha_z)$  for each frame which contain only the weights related to the first  $m < n$  eigenvectors and the extracted details from the reference frame. Since we assume that the animated sequence has a fixed connectivity, the information of the mesh connectivity is sent to the decoder only once. Using this information, we can compute the graph Laplacian matrix (Eq. (2)) in order to calculate the eigenvectors  $\phi_1, \phi_2, \dots, \phi_m$ . Then, we use Eq. (3) to find the coordinate functions  $(\hat{g}_x, \hat{g}_y, \hat{g}_z)$  for each frame. In this way, the coarse model of each frame can be acquired. Finally, details are added back to the coarse models to reconstruct the mesh sequence. As mentioned in the previous section, extracted details are added along the normal direction at each vertex. Figures 6 and 7 show the reconstruction process for sample frames of test sequences. It can be seen from these figures that some parts of the models have fairly low reconstruction quality. Simulation results reveal that there are two main reasons for this problem: large deviation from the normal direction between the original model and the coarse one and the presence of varying details due to non-rigid deformations. The first situation causes the extracted details to be inaccurate at some vertices. In the second situation, some details of the model vary due to non-rigid deformations throughout the sequence. Thus, adding the extracted details to the reconstructed coarse models results in large errors at some vertices. However, experiments show that the number of vertices which have large deviation from the normal direction is very small compared to the total number of vertices (approximately 7 to 10% of the total number of vertices). Thus, we can improve the reconstruction quality by sending the residual errors for these vertices. To this end, we consider the largest residual error for each frame and send the residual errors for vertices at which the residual errors are bigger than 0.7 of the largest residual error. This value is defined considering the reconstruction quality and human perception. In

**Fig. 6** Reconstruction results for some sample frames of *dance* sequence (from top to bottom: *frame 3*, *frame 101* and *frame 200*). Left: original model, middle: extracted coarse model, right: reconstructed model after details restoration (adding back the details to the coarse model)



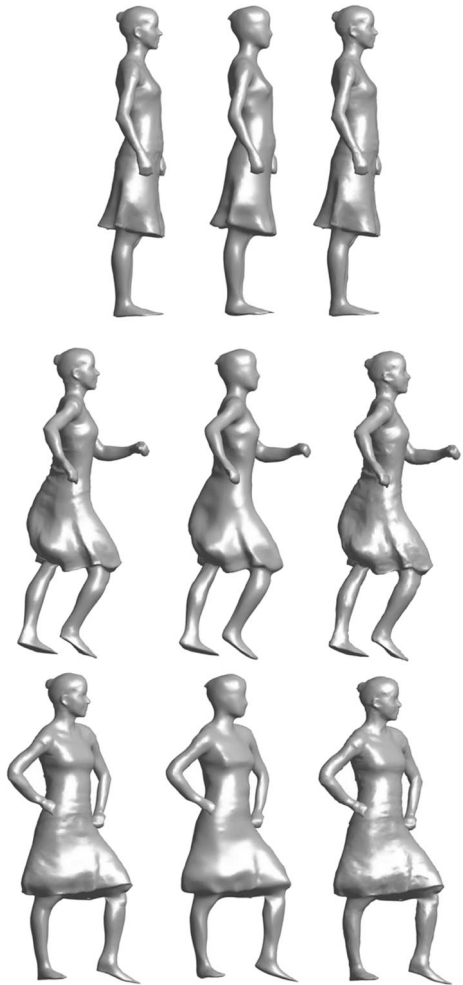
the presence of varying details, it is necessary to send the residual errors for the parts of the model with non-rigid deformations.

Figures 8 and 9 show the reconstructed models in Figs. 6 and 7 after adding the residual errors in order to improve the reconstruction quality. In Figs. 10 and 11, we compare the distribution of the reconstruction errors before and after adding the residual errors. Reconstruction errors are expressed as the percentage of the model's bounding box diagonal (i.e.,  $\text{error}/\text{diagonal} * 100$ ) whose values are shown with different colors. The smallest error is encoded by blue color while yellow color indicates the largest error. As seen from these figures, the quality of reconstruction is improved by adding the residual errors. Dance sequence contains a model with articular motions and thus, the residual errors are sent for a small portion of the mesh vertices which has a small effect on the compression performance. However, samba model has varying details due to the non-rigid deformation of clothing. Therefore, sending large residual errors for these parts negatively impacts the compression performance, although it improves the reconstruction quality.

### 3 Experimental results

In order to compare the performance of the proposed compression algorithm with the state of the arts, various experiments are performed on five test sequences: *samba model* [57], *squat2* [57], *cowheavy*, *dance* and *horse gallop*. These sequences have different number of frames and vertices. The *dance*, *cowheavy* and *horse gallop* sequences contain articular deformations

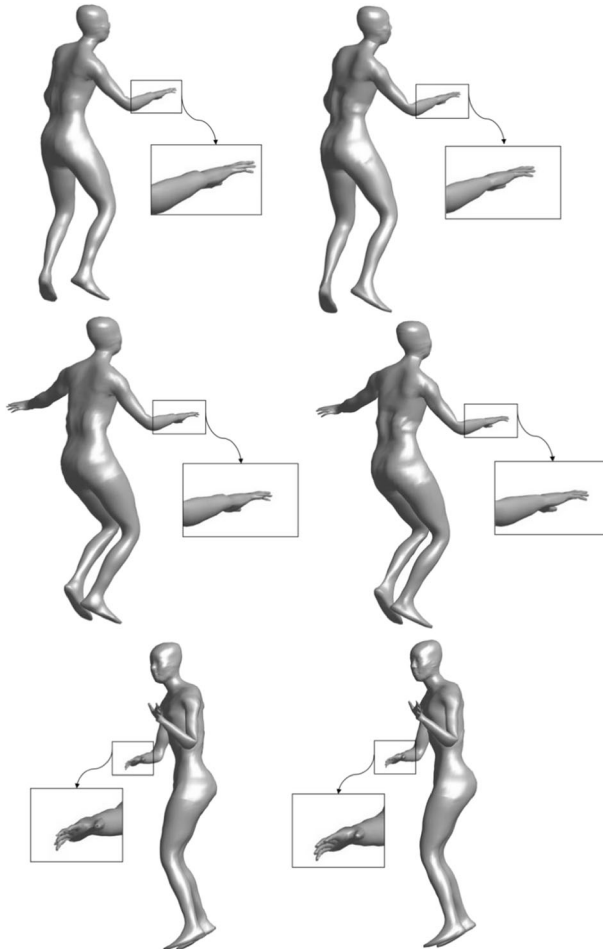
**Fig. 7** Reconstruction results for some sample frames of *samba* model (from top to bottom: *frame 3*, *frame 90* and *frame 174*). Left: original model, middle: extracted coarse model, right: reconstructed model after details restoration (adding back the details to the coarse model)



whereas the *samba* and *squat2* sequences have non-rigid deformations due to the clothing. Table 1 shows the properties of the test sequences.

The efficiency of a dynamic 3D mesh compression method depends on how much the method exploits spatial and temporal correlation in an animated mesh sequence to remove redundant information. In order to assess the performance of a compression method, we should consider two important factors: data rate and the distortion introduced by the compression. Efficient compression method is expected to yield lower distortion at the same data rate in comparison to other methods. The performance of the proposed algorithm is assessed by evaluating rate-distortion curves. We measure bitrates in terms of bits per vertex per frame (bpvf). The overall distortion between reconstructed and original sequences is measured using two error metrics:

- The KG error [24] which is a vertex-based error measure and is traditionally used in the literature to evaluate the overall distortion caused by compression. Therefore, it is utilized



**Fig. 8** Reconstruction results for sample frames of *dance* sequence (from top to bottom: *frame 3*, *frame 101* and *frame 200*). Left: before adding residual errors, right: after adding residual errors

in the experiments to compare the proposed method with previous studies. This error is defined as follows:

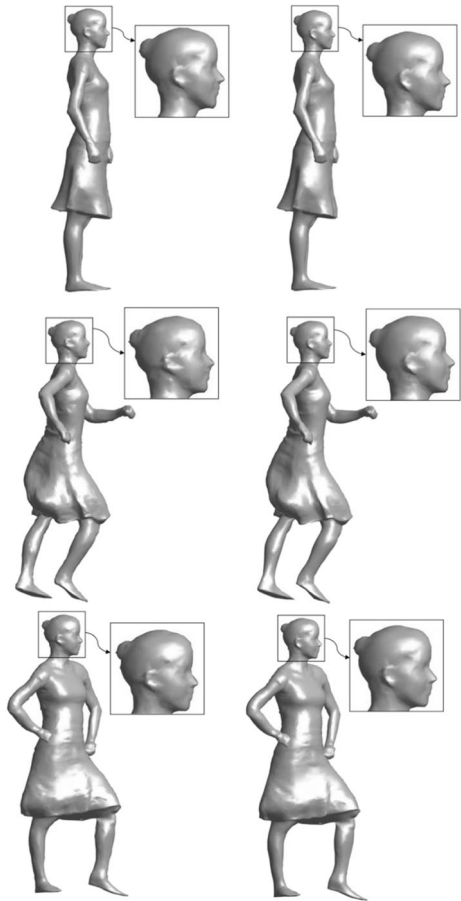
$$\text{KG error} = 100 \frac{\|\mathbf{M} - \tilde{\mathbf{M}}\|}{\|\mathbf{M} - E(\mathbf{M})\|} \quad (12)$$

Where  $\mathbf{M}$  is a  $3n \times f$  matrix and each frame of the original sequence is stored in a separate column of  $\mathbf{M}$ ;  $n$  and  $f$  are the number of vertices and frames, respectively.  $\tilde{\mathbf{M}}$  is a matrix containing reconstructed sequence.  $E(\mathbf{M})$  is an average matrix of the same dimensions as  $\mathbf{M}$ , in which the average vertex positions of all frames over time are stored in the columns. The distortion per frame is computed as the  $L^2$  norm of the original vertex positions relative to the reconstructed ones.

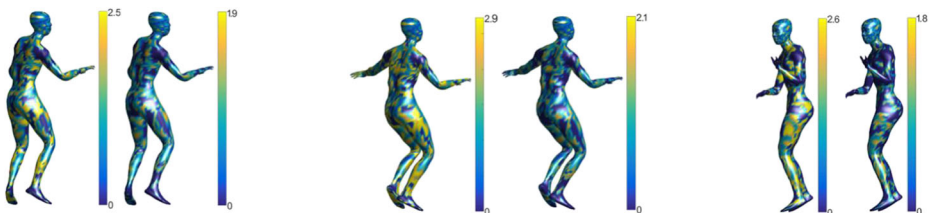
- Recently, however, researchers suggest using perceptual metrics to measure the amount of perceived distortion better than the conventional metrics. STED (spatiotemporal edge



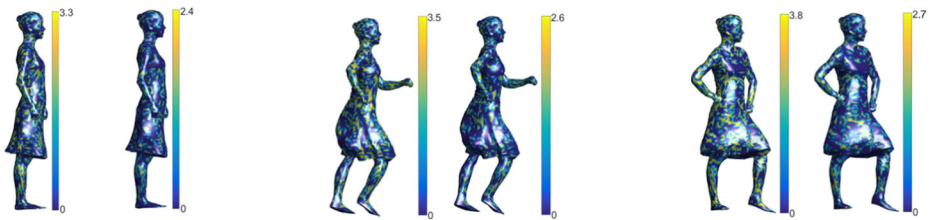
**Fig. 9** Reconstruction results for sample frames of *samba model* (from top to bottom: *frame 3*, *frame 90* and *frame 174*). Left: before adding residual errors, right: after adding residual errors



differences) is a new perceptual metric which is proposed in [55] for dynamic 3D mesh compression. In this metric, the changes of edge lengths between the reconstructed and original models are evaluated and it has a good correlation with the perceived distortion of dynamic 3D meshes. The overall error is defined as a hypotenuse of two errors: spatial error  $E_s$  and temporal error  $E_t$  which are related to each other using a weighting constant  $c$ :



**Fig. 10** Distribution of reconstruction error as the percentage of the model’s bounding box diagonal (i.e.,  $\text{error}/\text{diagonal} \times 100$ ) for sample frames of the *dance* sequence before and after adding residual errors. From left to right: frame 3 before and after adding the residual errors, frame 101 before and after adding the residual errors, frame 200 before and after adding the residual errors. The smallest error is encoded by blue color while yellow color indicates the largest error. It is clear that the quality of reconstruction is improved by adding residual errors for a small portion of the mesh vertices which has a small effect on the compression performance



**Fig. 11** Distribution of reconstruction error percentage for sample frames of *samba model* before and after adding residual errors. From left to right: frame 3 before and after adding the residual errors, frame 90 before and after adding the residual errors, frame 174 before and after adding the residual errors

$$\text{STED error} = \sqrt{E_s^2 + c^2 E_t^2} \quad (13)$$

$E_s$  represents the local standard deviation of the edges which is averaged over all the vertices and all the frames.  $E_t$  is defined as the average value of the relative temporal edge differences over all the vertices and all the frames. The reader can find more details about the STED error in [55]. The parameters of the STED error are defined according to [55].

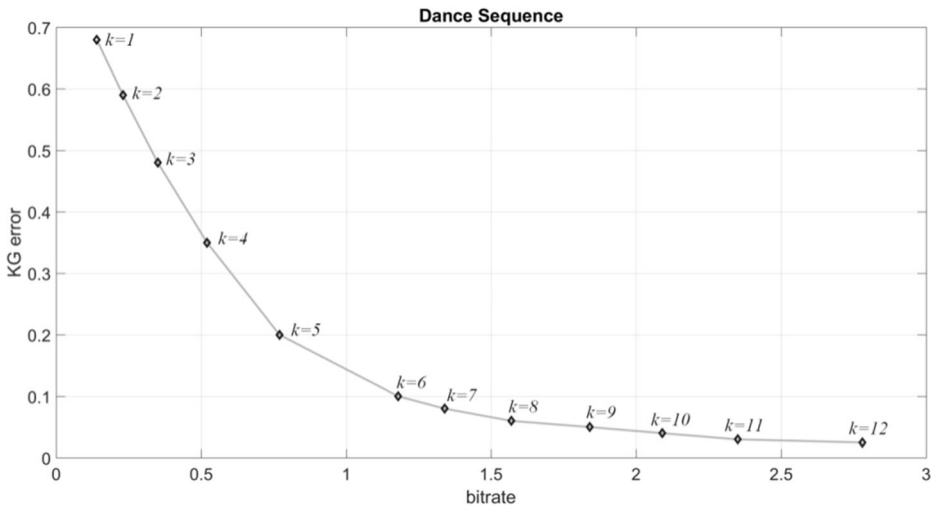
The value of  $m$  (the number of utilized eigenvectors of the graph laplacian matrix to create coarse model) is the only parameter to be tuned in the proposed method. As mentioned previously, there is a trade-off between compression ratio and reconstruction quality, depending on the selected value for  $m$ . It is necessary to increase the value of  $m$  to achieve more accurate reconstructed coarse models. However, the efficiency of compression will deteriorate by increasing  $m$  due to the need for sending more coordinate weights to the decoder side. Thus, the optimal value of  $m$  is determined by evaluating its impact on the reconstruction quality. To this end, the quality of reconstructed sequence is evaluated by the KG error for different values of  $m$ . Figure 12 shows the KG errors of the *dance* sequence for different values of  $k$  as the percentage of the utilized eigenvectors ( $m = k \cdot n/100$ ).

The errors are measured before adding residual errors to the reconstructed sequence in order to investigate the effect of  $m$  on the reconstruction quality. This figure clearly illustrates that the quality of reconstruction is improved by increasing  $k$  (or  $m$ ). However, the reduction in reconstruction error is significantly low after a particular value of  $k$  (for example  $k = 6$  in this case). Therefore, we select  $k = 6$  as the optimal value for the *dance* sequence. We can define the optimal values of  $k$  for other test sequences in a similar way. These values are tabulated in Table 2.

Unfortunately, a public implementation of the previous works is not available, and therefore we could only compare the proposed method with the results which are reported in the corresponding papers. Figure 13 shows the rate-distortion (R-D) curves for the proposed

**Table 1** Utilized test sequences for performance evaluation

Test sequence	# vertices	# frames
Samba model	9971	175
Squat2	10,002	250
Cowheavy	2904	204
Dance	7061	240
Horse gallop	8431	48

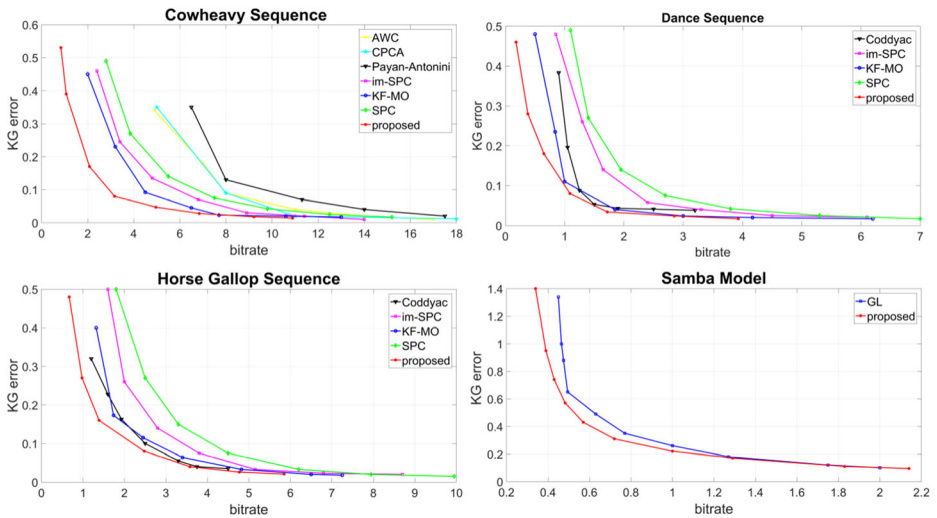


**Fig. 12** Rate distortion (R-D) curve of reconstructed dance sequence before adding residual errors for different values of  $k$  ( $m = k \cdot n/100$ )

algorithm as well as the previous methods, namely SPC (scalable predictive coding) [49], im-SPC (improved SPC) [38], KF-MO (key frame multi objective) [17], AWC (Anisotropic Wavelet Compression) [16], CPCA (Clustered PCA) [44], Payan and Antonini [40], Coddyc [51] and GL (geometric laplacian) [56]. R-D curves are measured in KG error in this figure. From this figure, we observe that the proposed method has superior performance for *dance*, *cowheavy* and *horse gallop* sequences. For the *cowheavy* sequence, the proposed method shows up to 52% and 21% lower bitrates than im-SPC and KF-MO techniques at the same distortion level. The SPC performance is far from the proposed algorithm. For *dance* and *horse gallop* sequences, the proposed method results in reduced data rates by 18% and 37% on average, respectively. The outstanding feature of these sequences is the articular body deformation of 3D model which results in highly fixed details throughout the sequence. Thus, the proposed method can effectively reconstruct the animated sequences with lower bitrates. However, *samba* model has somewhat varying details for particular parts of the model in cloth region because of non-rigid deformation in these parts. In the presence of non-rigid deformations, the details of 3D model are not fixed and will change throughout the sequence. Therefore, the quality of reconstructed sequence will be reduced at vertices with varying details. In order to enhance the reconstruction quality, it is necessary to send the residual errors for more parts of the model which negatively impacts on the

**Table 2** Selected optimal values of  $m$  for different test sequences

Test sequence	Total vertices $n$	Percentage of utilized eigenvectors $k$	Number of utilized eigenvectors $m$
Dance	7061	6	424
Cowheavy	2904	7	204
Horse gallop	8431	8	675
Samba	9971	9	898
Squat2	10,002	10	1000



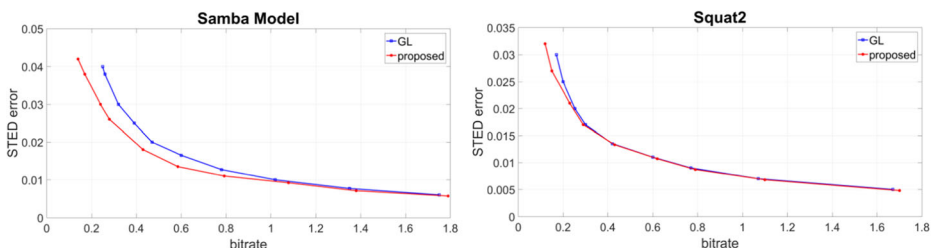
**Fig. 13** R-D curves (measured in KG error) comparison of the proposed method with previous methods

compression performance. Nevertheless, the proposed method has still competitive performance compared to the GL method.

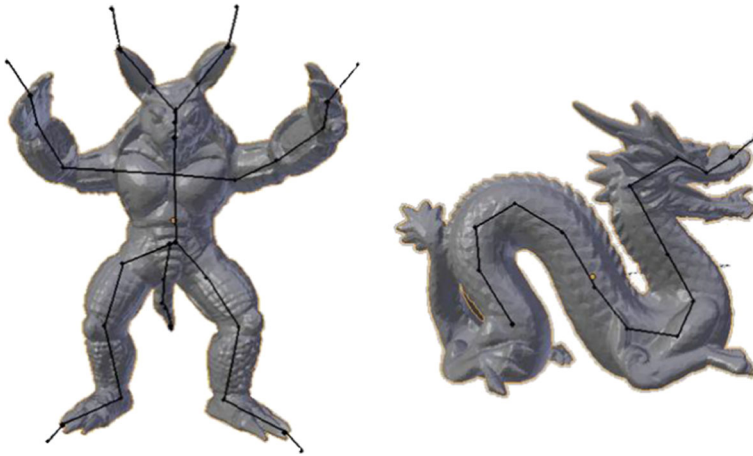
Figure 14 shows R-D curves for samba model and *squat2* sequence which are measured in STED error. The same problem of varying details is observed for *squat2* sequence. Although we need to send the residual errors for more vertices, the performance of the proposed method is highly competitive with the GL method.

The utilized test sequences for comparing the performance of the proposed method with the state of the arts have very fine details which may not give a deep insight into the proposed method. Therefore, we create two animated mesh sequences to provide a better understanding of the proposed compression algorithm. To this end, we choose the armadillo and dragon models from the Stanford 3D Scanning Repository [46]. The animated mesh sequences are generated by rigging the models in Blender [10] as the open source 3D modeling software (see Fig. 15). Table 3 shows the properties of the generated sequences. Sample frames of these sequences are illustrated in Fig. 16. The generated animated models have fixed details throughout the sequence.

The first step is to create a coarse model of the animated mesh using the graph Laplacian matrix. Figure 17 shows the created coarse models for the armadillo and dragon sequences. We have utilized different number of eigenvectors to remove fine details. As it can be seen from this figure, by reducing the value of  $m$  (the number of utilized eigenvectors to create the coarse



**Fig. 14** R-D curves (measured in STED error) comparison of the proposed method with previous works



**Fig. 15** Rigging the armadillo and dragon models in the Blender software to make animated sequences

model), the filtration of the fine details becomes stronger and we can generate high-level abstractions of the original model. The value of  $k$  ( $m = kn/100$ ) is set at 2 for these sequences.

The second step is to extract fine details which is carried out according to the procedure explained in section 2.2.1.

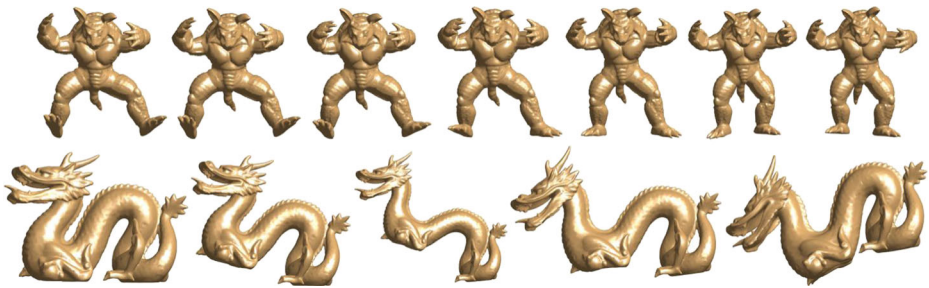
At the decoder side, the coarse models are reconstructed for different frames by using the graph Laplacian matrix which is obtained from the connectivity data. Then, extracted fine details at the first frame are restored to the reconstructed coarse models. Figures 18 and 19 show the reconstructed models by adding the fine details back to the generated coarse models at the decoder. Since the armadillo and dragon models have fixed details throughout the sequence, the reconstructed models have high quality. The possible reconstruction errors occur at vertices which have fairly large deviations from the normal direction between the original model and the coarse one. As mentioned previously, the number of these vertices are very small compared to the total number of the vertices and sending the residual errors has a little impact on the compression ratio. Figures 20 and 21 depict the reconstructed models after adding the residual error. We also compare the distribution of the reconstruction errors before and after adding the residual errors. We observe that the reconstruction quality is improved by sending the residual errors for a small part of the vertices.

The R-D curves of the proposed method for the armadillo and dragon sequences are presented in Fig. 22. We observe from this figure that the proposed method has excellent performance for animated meshes with highly fixed details throughout the sequence (i.e. for articular deformations) and it is possible to compress a sequence with lower bitrates and reconstruct it with lower distortion.

In order to evaluate the time complexity of the eigenspace compression algorithm, we split the total run time into two parts: the compression run time and the decompression run time.

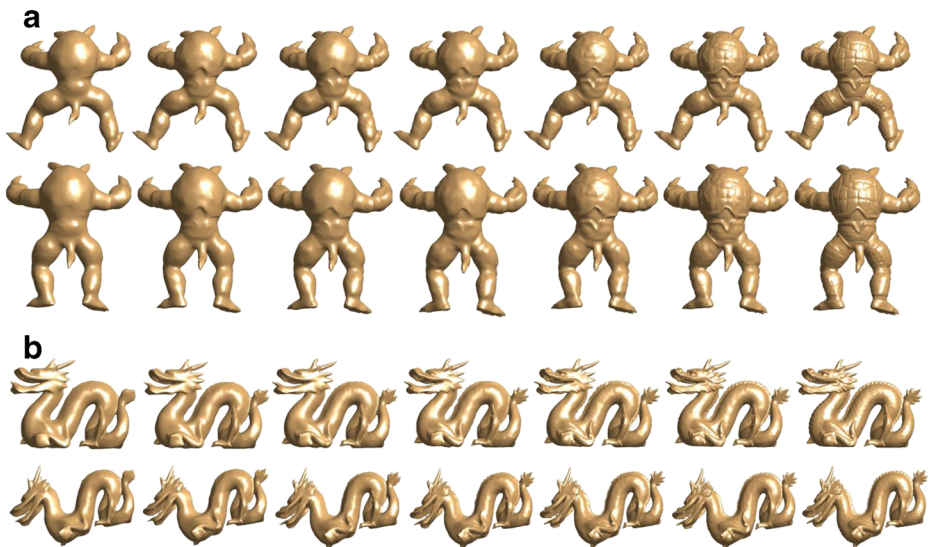
**Table 3** Properties of the created sequences

Sequence	#vertices	#faces	#frames
Armadillo	20,001	39,995	1–150
Dragon	19,878	39,774	1–150

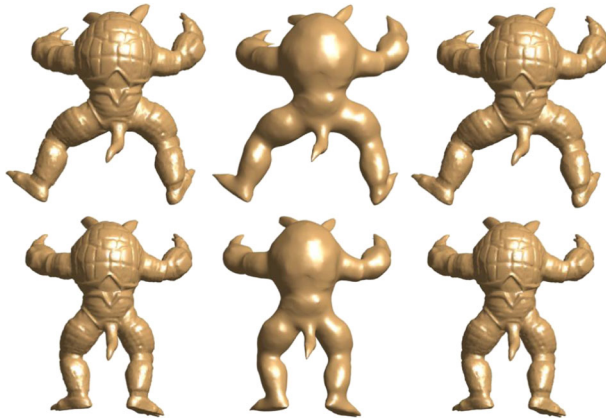


**Fig. 16** Sample frame from the armadillo (top) and dragon (bottom) sequences

The compression phase includes three main stages, namely: coarse model generation (CMG), details extraction (DE), and encoding the required data (ENC) including coordinate weights for each frame, extracted details and residual errors. The decompression phase includes three main stages, namely: reconstruction of coarse model for each frame (RCM), restoration of details to the reconstructed coarse models (RD) and adding residual errors to the reconstructed models (AR). In Table 4, the processing time of each stage is illustrated for the compression and decompression parts separately. The run times are expressed as the processing time per frame. MATLAB R2016a is used to implement the experiments on a computer with Intel® Core i7 CPU with 2.8 GHz, 8GB RAM and ATI GPU with 4GB video memory. We notice that the coarse model generation stage in the compression phase and the reconstruction of coarse models stage in the decompression phase take longer than the other stages. This can be explained by the fact that we require to compute the eigenvectors of the graph laplacian matrix to generate the coarse models. However, the eigenvectors are computed just once for



**Fig. 17** Generated coarse models for: (a) *Frame 1* (top) and *Frame 100* (bottom) of the armadillo sequence (from left to right: generated coarse models for  $k=2, 3, 4, 5, 8, 15$  and  $100\%$  ( $m = kn/100$ )). The last model shows the original frame ( $k = 100$ )), (b) *Frame 1* (top) and *Frame 120* (bottom) of the dragon sequence (from left to right: generated coarse models for  $k=2, 3, 4, 5, 8, 15$  and  $100\%$  ( $m = kn/100$ )). The last model shows the original frame ( $k = 100$ )). It is clear how details filtered from the model by reducing the number of utilized eigenvectors

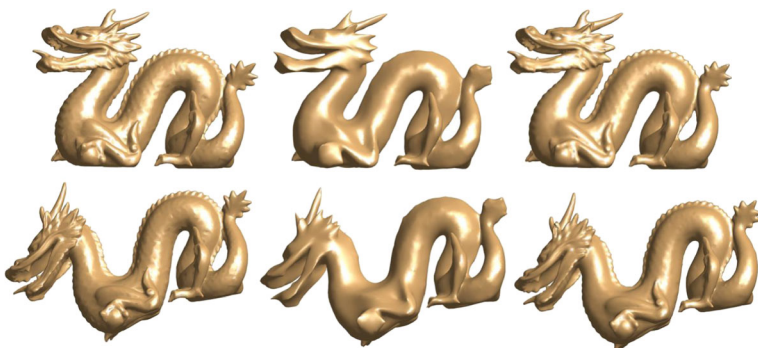


**Fig. 18** Reconstruction of sample frames from the amadillo sequence by restoring the extracted details to the coarse models: (top) *Frame 2*, (bottom) *Frame 100*. (from left to right: original model, coarse model, reconstructed model by restoring details)

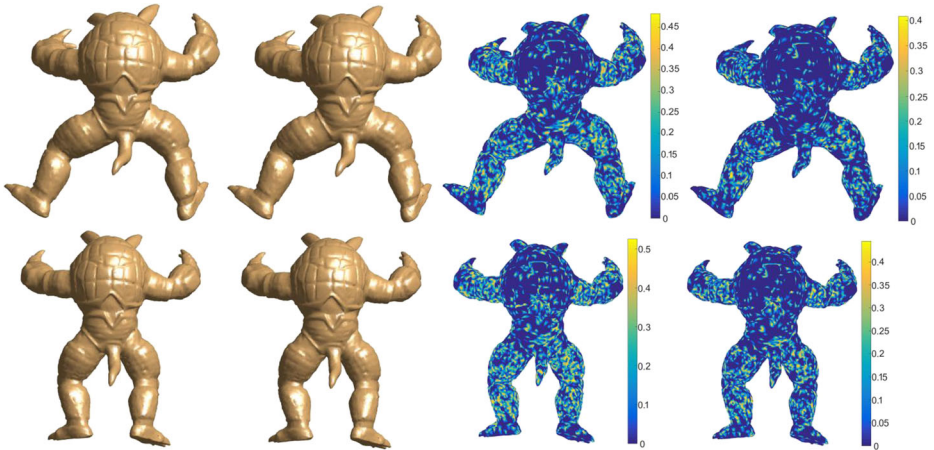
the whole sequence. As total run time is divided to the number of frames, the processing time of the proposed method is very low for long sequences. Thus, the run time depends on the number of frames in the animated mesh sequence and the number of vertices in the 3D mesh. The evaluation of the time complexity reveals that the run time of the proposed method is in the limits of practical applicability. On the other hand, MATLAB scripts require compilation and interpretation which increase the execution time. Thus, C++ implementation will decrease the total run time of the proposed method.

### 3.1 Discussion

Any deformation of an object can be considered in 3 categories: rigid motion, articular motion, and non-rigid motion. In rigid motion, total vertices of the object deform under fixed motion (i.e. fixed rotation and translation) matrix. In articular motion, the object consists of multiple parts in which every part deforms under its own motion matrix that is different from motion matrix of other parts. At the same time, the points belong to



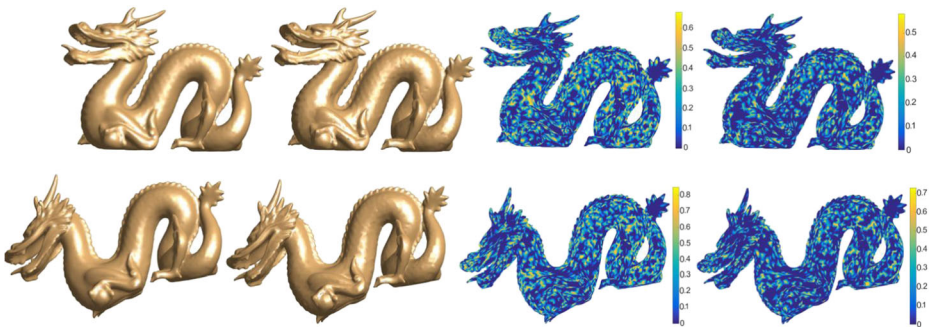
**Fig. 19** Reconstruction of sample frames from the dragon sequence by restoring the extracted details to the coarse models: (top) *Frame 2*, (bottom) *Frame 120*. (from left to right: original model, coarse model, reconstructed model by restoring details)



**Fig. 20** Reconstruction of sample frames from the armadillo sequence after adding residual errors for a small number of vertices and comparison with the reconstruction by adding back only the extracted details: (top) *Frame 2*, (bottom) *Frame 100*. (from left to right: reconstructed model before adding residual errors, reconstructed model after adding residual errors, distribution of the reconstruction error percentage before adding residual errors, distribution of the reconstruction error percentage after adding residual errors)

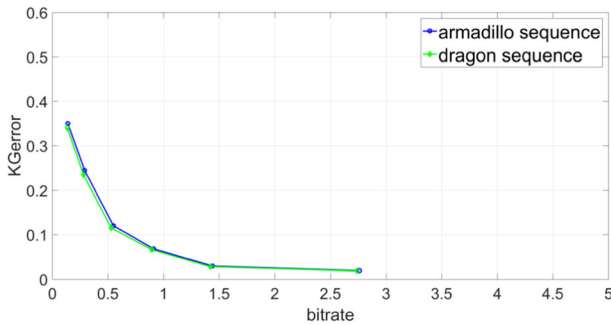
each part follow fix motion. In non-rigid motion, each point of object can move arbitrarily and there is no group of similar motions.

The proposed method is mainly based on this assumption that the fine details of an animated mesh model are highly fixed throughout sequence and thus it is possible to reconstruct the animated sequence by restoring extracted details to coarse models. This assumption is true for most of the practically utilized animated models which have articular deformations during time. Since we extract the fine details along the normal directions, reconstruction errors for this kind of animated models occur for a small number of vertices at which vertex displacements between original model and the coarse one have large deviations from the normal direction. Thus, it is necessary to send residual errors for these vertices in order to have a good reconstruction quality. However, the number of these vertices is very low and the proposed method shows excellent performance in these cases.



**Fig. 21** Reconstruction of sample frames from the dragon sequence after adding residual errors for a small number of vertices and comparison with the reconstruction by adding back only the extracted details: (top) *Frame 2*, (bottom) *Frame 120*. (from left to right: reconstructed model before adding residual errors, reconstructed model after adding residual errors, distribution of the reconstruction error percentage before adding residual errors, distribution of the reconstruction error percentage after adding residual errors)





**Fig. 22** R-D curves (measured in the KG error) of the proposed method for the armadillo and dragon sequences

In the presence of non-rigid deformations, the details of 3D model are not fixed and change throughout the sequence. Therefore, the quality of the reconstructed sequence will be reduced in regions with varying details. In order to enhance the reconstruction quality, it is necessary to send the residual errors for more points of the model which negatively impacts on the compression performance. However, if we have non-rigid deformations in small regions of the model (like the samba and squat2 models), it is possible to send the residual errors without significant effect on the compression performance. Experimental results demonstrate that the proposed method has competitive performance in these cases compared to the state of the art methods.

For animated mesh models with varying details in large regions, we need to send the residual errors for a large number of vertices to improve the reconstruction quality. Thus, the compression performance deteriorates for this kind of animated models and this is the main limitation of the proposed method. In previous works, some solutions are proposed to solve the problem of large residual errors. For example in [37], a rate-distortion mechanism is presented to optimally quantize the residual errors. A temporal discrete cosine transform (temporal-DCT) representation is used in [32] to compress the prediction errors. In [33], Mamou et al. propose to represent residual errors as the multi-chart geometry images (MCGIM) which are then compressed by using standard image encoders such as JPEG and MPEG-4. Of course, even for the varying details in large regions, our method can decrease the residual error by using the smoothed version plus fixed details as an initial guess of such vertices and in this way we can encode the reduced residual error with small number of bits and do a minor compression.

**Table 4** Average run times of the proposed method for the test sequences

Test sequence	Compression			Decompression			<i>k</i>
	CMG (sec)	DE (sec)	ENC (sec)	RCM (sec)	RD (sec)	AR (sec)	
Cowheavy	0.005	0.012	0.002	0.005	0.002	0.0003	7
Dance	0.074	0.041	0.005	0.071	0.005	0.0003	6
Horse gallop	0.53	0.95	0.004	0.52	0.006	0.0006	8
Samba	0.54	0.082	0.006	0.52	0.007	0.0004	9
Squat2	0.56	0.094	0.008	0.55	0.008	0.0005	10
Armadillo	0.25	0.28	0.007	0.24	0.015	0.0006	2
Dragon	0.25	0.3	0.007	0.24	0.014	0.0007	2

## 4 Conclusion

In this paper, we presented an efficient compression method for animated mesh sequences with shared connectivity. Dynamic 3D models usually have fine details (high frequency modes) which are usually fixed throughout the sequence. These fixed details result in high temporal correlation between frames of the animated sequence. The proposed method removes details by applying the graph laplacian operator and using lower number of eigenvectors. In this way, we can reconstruct a coarse model of each frame at the decoder side. Moreover, the graph laplacian operator is a combinatorial laplacian and only depends on the connectivity information which is sent to the decoder just once. A simple and efficient approach is proposed to extract the removed fixed details from a reference frame. Extracted details are added back to the reconstructed coarse models at the decoder.

Experimental results reveal that the proposed method has superior performance for animated sequences with articular body deformations and outperforms the previous works. For animated mesh sequences with varying details in small regions of 3D model, it is necessary to send more residual errors which negatively impact the compression performance. Nevertheless, the proposed method has still competitive performance for this kind of animated sequences compared to the state of the arts. The main limitation of the proposed method is related to animated meshes which have varying details in large regions of the model. In these cases, we require to send residual errors for most of the vertices which significantly decrease the compression ratio. Moreover, the need to calculate the eigenvectors of the graph laplacian matrix makes the proposed algorithm sensitive in the case of irregular meshes. Nevertheless, for regular and semi-regular meshes, the proposed method results in a satisfactory performance compared with the state of the art.

## References

1. Ahn J, Kim C, Kuo C, Ho Y (2001) Motion-compensated compression of 3D animation models. *Electron Lett* 37(24):1445–1446
2. Alexa M, Muller W (2000) Representing animations by principal components. *Computer Graphics Forum* 19(3):411–418
3. Amjoun R (2009) Compression of static and dynamic three-dimensional meshes. PhD thesis, Eberhard Karls University of Tübingen
4. Amjoun R, Straßer W (2007) Efficient compression of 3-D dynamic mesh sequences. *Journal of the WSCG* 15(1–3):32–46
5. Amjoun R, Straßer W (2008) Predictive-DCT coding for 3D mesh sequences compression. *J Virtual Real Broadcast* 5(6)
6. Amjoun R, Straßer W (2009) Single-rate near lossless compression of animated geometry. *Comput Aided Des* 41(10):711–718
7. Amjoun R, Sondershaus R, Straßer W (2006) Compression of Complex Animated Meshes. In: Nishita T, Peng Q, Seidel HP (eds) *Advances in Computer Graphics. Lecture Notes in Computer Science*, vol 4035. Springer, Berlin, Heidelberg, p 606–613
8. Belkin M, Sun J, Wang Y (2008) Discrete laplace operator on meshed surfaces. In: *Proceedings of the twenty-fourth annual symposium on Computational geometry* pp 278–287
9. Ben-Chen M, Gotsman C (2005) On the optimality of spectral compression of mesh data. *ACM Trans Graph* 24(1):60–80
10. Blender. Open source 3D graphics modeling, animation, creation software: <http://www.blender.org>
11. Boulfani-Cuisinaud Y, Antonini M (2007) Motion-based geometry compensation for dwt compression of 3D mesh sequence. In: *IEEE International Conference in Image Processing*, pp 217–220
12. Briceno H, Sander P, McMillan L, Gortler S, Hoppe H (2003) Geometry videos: a new representation for 3D animations. In: *Proceedings of ACM Symposium on Computer Animation* pp 136–146

13. Dey TK, Ranjan P, Wang Y (2012) Eigen Deformation of 3D Models. *Vis Comput* 28(6):585–595
14. Dhibi N, Elkefi A, Bellil W, Amar CB (2017) Multi-layer compression algorithm for 3D deformed mesh based on multi library wavelet neural network architecture. *Multimed Tools Appl* 76(20):20869–20887. <https://doi.org/10.1007/s11042-016-3996-8>
15. Fragkiadaki M, Salas M, Arbelaez P, Malik J (2014) Grouping-based low-rank trajectory completion and 3D reconstruction. Proceedings of the 27th International Conference on Neural Information Processing Systems (NIPS 2014) pp 55–63
16. Guskov I, Khodakovskiy A (2004) Wavelet compression of parametrically coherent mesh sequences. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, Eurographics Association, Aire-la-Ville, Switzerland, pp 183–192
17. Hajizadeh MA, Ebrahimezhad H (2016) Predictive compression of animated 3D models by optimized weighted blending of key-frames. *Computer Animation and Virtual Worlds* 27(6):556–576
18. He Y, Lu H, Xie H (2014) Semi-supervised non-negative matrix factorization for image clustering with graph laplacian. *Multimed Tools Appl* 72(2):1441–1463
19. Heu J, Kim C, Lee S (2009) SNR and temporal scalable coding of 3-D mesh sequences using singular value decomposition. *J Vis Commun Image Represent* 20(7):439–449
20. Hildebrandt K, Polthier K (2011) On approximation of the laplace-beltrami operator and the willmore energy of surfaces. *Comput Graphics Forum* 30(5):1513–1520
21. Ibarria L, Rossignac J (2003) Dynapack: space-time compression of the 3D animations of triangle meshes with fixed connectivity. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation pp 126–135
22. James DL, Twigg CD (2005) Skinning mesh animations. *ACM Trans Graph* 24(3):399–407
23. Karni Z, Gotsman C (2000) Spectral compression of mesh geometry. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000) pp 279–286
24. Karni Z, Gotsman C (2004) Compression of soft-body animation sequences. *Comput Graph* 28(1):25–34
25. Kennedy R, Balzano L, Wright SJ, Taylor CJ (2016) Online algorithms for factorization-based structure from motion. *Comput Vis Image Underst* 150:139–152
26. Lengyel JE (1999) Compression of time-dependent geometry. In: Proceedings of the 1999 Symposium on Interactive 3D Graphics pp 89–95
27. Levy B (2006) Laplace-beltrami eigenfunctions: towards an algorithm that understands geometry. In: IEEE International Conference on Shape Modeling, Invited Talk
28. Lin Y, Yang L, Lin Z, Lin T, Zha H (2017) Factorization for projective and metric reconstruction via truncated nuclear norm. International Joint Conference on Neural Networks (IJCNN 2017)
29. Lu M, Zheng B, Takamatsu J, Nishino K, Ikeuchi K (2010) 3D shape restoration via matrix recovery. Asian conference on computer vision (ACCV 2010) pp 306–315
30. Maglo A, Lavoue G, Dupont F, Hudelot C (2015) 3D mesh compression: survey, comparisons and emerging trends. *ACM Comput Surv* 47(3):1–40
31. Alliez P, Gotsman C (2003) Recent advances in compression of 3D meshes. In: Proceedings Symposium on Multiresolution in Geometric Modeling
32. Mamou K, Zaharia T, Prêteux F (2006) A skinning approach for dynamic 3D mesh compression. *Comput Anim Virtual Worlds* 17(3–4):337–346
33. Mamou K, Zaharia T, Prêteux F (2006) Multi-chart geometry video: a compact representation for 3D animations. In: Third International Symposium on 3D Data Processing, Visualization, and Transmission pp 711–718
34. Mansouri S, Ebrahimezhad H (2016) Efficient axial symmetry aware mesh approximation with application to 3D pottery models. *Multimed Tools Appl* 75(14):8347–8379
35. Marpe D, Schwarz H, Wiegand T (2003) Context-based adaptive binary arithmetic coding in the H. 264/AVC video compression standard. *IEEE Trans Circuits Syst Video Technol* 13(7):620–636
36. Muller K, Smolic A, Kautzner M, Eisert P, Wiegand T (2005) Predictive compression of dynamic 3D meshes. In: 2005 I.E. International Conference on Image Processing pp 621–624
37. Müller K, Smolic A, Kautzner M, Eisert P, Wiegand T (2006) Rate-distortion optimized predictive compression of dynamic 3D mesh sequences. *Signal Process Image Commun* 21(9):812–828
38. Oguz Bici M, Bozdagi AG (2011) Improved prediction methods for scalable predictive animated mesh compression. *J Vis Commun Image Represent* 22(7):577–589
39. Payan F, Antonini M (2005) Wavelet-based compression of 3D mesh sequences. In: Proceedings of IEEE ACIDCA-ICMI
40. Payan F, Antonini M (2007) Temporal wavelet-based compression for 3D animated models. *Comput Graph* 31(1):77–88
41. Peng J, Kim CS, Kuo CCJ (2005) Technologies for 3D mesh compression: a survey. *J Vis Commun Image Represent* 16:688–733

42. Reuter M, Wolter FE, Peinecke N (2006) Laplace-beltrami spectra as “shape-DNA” of surfaces and solids. *Comput Aided Des* 38(4):342–366
43. Rossignac J (1999) Edgebreaker: connectivity compression for triangle meshes. *IEEE Trans Vis Comput Graph* 5(1):47–61
44. Sattler M, Sarlette R, Klein R (2005) Simple and efficient compression of animation sequences. In: *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp 209–217
45. Shi Y, Ding W, Qi N, Yin B (2014) Prediction-based realistic 3D model compression. *Multimed Tools Appl* 70(3):2125–2137
46. Stanford 3D Scanning Repository. <http://graphics.stanford.edu/data/3Dscanrep/>
47. Stefanoski N, Ostermann J (2006) Connectivity-guided predictive compression of dynamic 3D meshes. In: *2006 I.E. International Conference on Image Processing* pp 2973–2976
48. Stefanoski N, Ostermann J (2008) Spatially and temporally scalable compression of animated 3D meshes with MPEG-4/FAMC. In: *Proceedings of the IEEE International Conference on Image Processing* pp 2696–2699
49. Stefanoski N, Ostermann J (2010) SPC: fast and efficient scalable predictive coding of animated meshes. *Comput Graphics Forum* 29:101–116
50. Stefanoski N, Liu X, Klie P, Ostermann J (2007) Scalable linear predictive coding of time-consistent 3D mesh sequences. *3DTV-Conference The True Vision – Capture, Transmission and Display of 3D Video* pp 1–4
51. Váša L, Coddyc SV (2007) Connectivity driven dynamic mesh compression. In: *3DTV-Conference The True Vision – Capture, Transmission and Display of 3D Video* pp 1–4
52. Váša L, Skala V (2009) Combined compression and simplification of dynamic 3D meshes. *Comput Anim Virtual Worlds* 20(4):447–456
53. Váša L, Skala V (2009) COBRA: compression of the basis for PCA represented animations. *Comput Graphics Forum* 28:1529–1540
54. Váša L, Skala V (2010) Geometry-driven local neighbourhood based predictors for dynamic mesh compression. *Comput Graphics Forum* 29:1921–1933
55. Váša L, Skala V (2011) A perception correlated comparison method for dynamic meshes. *IEEE Trans Vis Comput Graph* 17(2):220–230
56. Váša L, Marras S, Hormann K, Brunnett G (2014) Compressing dynamic meshes with geometric Laplacians. *Comput Graphics Forum* 33(2):145–154
57. Vlasic D, Baran I, Matusik W, Popovic J (2008) Articulated mesh animation from multi-view silhouettes. *ACM Trans Graph* 27(3):1–9
58. Wang M, Li K, Wu F, Lai U-K, Yang J (2016) 3D motion recovery via low rank matrix analysis. *IEEE International Conference on Visual Communications and Image Processing (VCIP 2016)*
59. Wright J, Ganesh A, Rao S, Peng Y, Ma Y (2009) Robust principal component analysis: exact recovery of corrupted low-rank matrices via convex optimization. In: *Advances in Neural Information Processing Systems 22*. MIT Press
60. Yan Y, Tan M, Tsang I, Yang Y, Zhang C, Shi Q (2015) Scalable maximum margin matrix factorization by active riemannian subspace search. *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI 2015)* pp 3988–3994
61. Yang J, Kim C, Lee S (2002) Compression of 3-D triangle mesh sequences based on vertex-wise motion vector prediction. *IEEE Trans Circuits Syst Video Technol* 12(12):1178–1184
62. Zhang J, Owen C (2004) Octree-based animated geometry compression. In: *Proceedings of the Data Compression Conference* pp 508–517
63. Zhang J, Owen C (2007) Octree-based animated geometry compression. *Comput Graph* 31(3):463–479
64. Zhang H, Kaick O, Dyer R (2010) Spectral mesh processing. *Comput Graphics Forum* 29(6):1865–1894



**Mohammadali Hajizadeh** : was born in Iran, on June 20, 1984. He received the B.Sc. and M.Sc. degrees in Electronic and Communication Engineering from Tabriz University and Sahand University of Technology in 2008 and 2011, respectively. Now he is a Ph.D. student at Sahand University of Technology. His major interests are image and multimedia processing, computer vision, computer graphics and animation.



**Hossein Ebrahimnezhad** : was born in Iran 1971. He received the B.Sc. and M.Sc. degrees in Electronic and Communication Engineering from Tabriz University and K.N.Toosi University of Technology in 1994 and 1996, respectively. In 2007, he received his Ph.D. degree from Tarbiat Modares University. His research interests include image and multimedia processing, computer vision, computer graphics, 3D model processing and soft computing. Currently, he is a full professor at Sahand University of Technology.