Computer Aided Geometric Design 73 (2019) 70–85

Contents lists available at ScienceDirect

Computer Aided Geometric Design

www.elsevier.com/locate/cagd

Adaptive representation of dynamic 3D meshes for low-latency applications

Gerasimos Arvanitis [a,*], Aris S. Lalos [b], Konstantinos Moustakas [a]

[a] Electrical and Computer Engineering, University of Patras, Greece
[b] Industrial Systems Institute, "ATHENA" Research Center, Greece

ARTICLE INFO

Article history:
Available online 5 August 2019

Keywords:
Robust PCA
Incremental SVD
Laplacian interpolation
Scalable coding

ABSTRACT

Recently, 3D visual representations of highly deformable 3D models, such as dynamic 3D meshes, are becoming popular due to their capability to represent realistically the motion of real-world objects/humans, paving the road for new and more advanced immersive virtual, augmented and mixed reality experiences. However, the real-time streaming of such models introduces increasing challenges related to low cost, low-latency and scalable coding of the acquired information. In view of this, this article proposes an efficient scalable coding mechanism, that decomposes a mesh sequence into spatial and temporal layers that remove a single vertex at each layer. The removed vertices are predicted by performing Laplacian interpolation of the motion vectors. The artifacts that are introduced in low-resolution representations are mitigated using a subspace based normal-vector denoising procedure, that is optimized to support low-latency streaming scenarios using incremental SVD. A novel initialization strategy offers robustness to outliers generated due to local deformations. An extensive evaluation study using several synthetic and scanned dynamic 3D meshes highlights the benefits of the proposed approach in terms of both execution time and reconstruction quality even in very low throughput scenarios of bit-per-vertex-per-frame (bpvf).

1. Introduction

3D meshes are widely used in various applications in different scientific fields from heritage science and education to health and robotics. Recently, interest in 3D mesh sequences has also increased because of the rapid growth of new 3D scanning technologies, 3D cinema/television, and immersive telepresence systems capable to provide XR[1] experiences. Streaming of 3D animated objects can be used in applications where the geometric data is live-captured and needs to be available in real-time. Nevertheless, these types of applications demand the storage and the transmission of a huge amount of 3D data. The real-time rendering of 3D models, representing either real-world or synthetic objects, generates massive datasets and it requires the use of efficient and fast algorithms for increasing the compression ratios without affecting noticeably the visual quality of the object. For this reason, various techniques on dynamic 3D mesh processing should be developed to address the growing demand and at the same time, to handle these important challenges. Geometry data are generally encoded in a lossy manner Peng et al. (2005). However, scalable coding seems to be the most promising approach

* Corresponding author.
E-mail addresses: arvanitis@ece.upatras.gr (G. Arvanitis), lalos@isi.gr (A.S. Lalos), moustakas@ece.upatras.gr (K. Moustakas).

[1] X refers to Virtual, Augmented or Mixed.

https://doi.org/10.1016/j.cagd.2019.07.005
0167-8396/

especially in cases where the network performance is unstable, allowing practical implementations for inferring the available bandwidth and adjusting rates for chunks while balancing metrics like quality, interruptions, number of rate switches (i.e., rate stability). Additionally, a stringent latency is a vital requirement for providing a pleasant immersive VR/AR experience Elbamby et al. (2018). This means that any real-transmitted dynamic 3D mesh must be easily perceivable, at any frame of its sequence, without affecting its visual quality regardless if a decrease in the transmission rate takes place at any moment. The main purpose of low-latency applications is avoiding to disturb the user's perception because this can negatively affect his/her quality of experience.

In this work, we take into account all the aforementioned problems and challenges in order to develop a scalable coding method for reliable adaptive representation of dynamic surface 3D meshes, with known connectivity, ideal for low-latency applications. Our research counts on the observation that a reduced frame of a mesh sequence can efficiently be reconstructed by taking advantage of the general spatiotemporal information of the entire animated mesh. The proposed method is scalable since a different number of points are transmitted in each frame, depending on the network capability. To summarize, the main contributions of this work are:

- We propose a mechanism for decomposing a mesh sequence into spatial and temporal layers that remove a single vertex at each layer. This decomposition is based on (i) topological characteristics of the mesh and (ii) the temporal behavior of each point separately as their position change through the time frames. In this way, we remove vertices that can be predicted accurately by their neighbors.
- Contrary to the conventional compression approaches for 3D mesh sequences, our method does not use a fixed compression rate for the whole dynamic 3D mesh but each mesh may have a different rate depending on the network's ability at the certain moment.
- We have created an online step for the online reconstruction of the removed frames by exploiting coherences on the subspaces corresponding to the different layered representations of the corresponding normals. This is achieved using incremental SVD (ISVD), ending up with a significant saving in complexity, allowing a close to real-time execution using current machines.
- Besides the fact that our method has a lot of parameters, we suggest the use of the ideal values so the user does not need to change or modify them for better results. In other words, all the used parameters are fixed and pre-defined, based on either the literature or the experimental analysis, so the users do not need to search for ideal values of parameters per model.

The rest of this paper is organized as follows: Section 2 reviews prior art in detail. Section 3 presents some preliminaries, necessary for the proposed method. It also highlights our contributions, emphasizing how the proposed pipeline is capable to improve both the reconstruction quality and the computational efficiency providing increased quality of experience to the user. Section 4 presents the qualitative, quantitative and comparative results of our method, highlighting its advantages, while Section 5 draws the conclusions.

2. Related work

3D tele-immersion allows the mixture of real and virtual content Mekuria et al. (2014), Desai et al. (2017). However, this type of applications requires the real-time transmission of triangular meshes which is a very challenging issue. Mueller et al. (2004) presented a complete system for efficient 3D animated object extraction, representation, coding, and interactive rendering. Zhang et al. (2010) proposed a method for generating progressive animated models based on local feature analysis and deformation area preservation. Matusik and Pfister (2004) presented a system for real-time acquisition, transmission, and high-resolution 3D display of dynamic multiview TV content. However, this system is applied to pixels and not to point clouds with connectivity. Stefanoski et al. (2007) presented a linear predictive compression approach for time-consistent 3D mesh sequences supporting and exploiting scalability. The algorithm decomposes each frame of a mesh sequence in layers employing patch-based mesh simplification techniques. This layered decomposition technique is consistent in time. Following the predictive coding paradigm, local temporal and spatial dependencies between layers and frames are exploited for compression. Isenburg et al. (2005) proposed a streaming compression scheme that allows encoding meshes on-the-fly by operating on a partial representation of the connectivity that is created and deleted as the mesh is fed in increments of single triangle and vertices to the compressor.

Yang et al. (2006), Yang et al. (2005) proposed different approaches that represent 3D dynamic objects with a semi-regular mesh sequence. Then, they compress the sequence using the spatiotemporal wavelet transform. Hachani et al. (2016) used 3D affine transforms to compute the prediction errors. Additionally, they achieved to improve the coding efficiency by optimizing the prediction error quantization, using a rate control mechanism. On the other hand, Zhang and Owen (2004), Zhang and Owen (2005) used an octree-based coder representing the motion between points of adjacent frames with an octree. The eight corners of each octree block are associated with eight corresponding motion vectors. The motion of each point in that block is approximated by the tri-linear interpolation of the corner motion vectors. Thanou et al. (2016) exploited both the spatial correlation inside each frame (through the graph) and the temporal correlation between the frames (through the motion estimation) for color and geometry compression. Subramanyam and Cesar (2018) used a point cloud codec that encodes additional information in an enhancement layer, then they propose to add inter prediction to the en-

hancement layer in order to gain further bit rate savings. Ibarria and Rossignac (2003) proposed an algorithm traversing each mesh surface in a depth-first order. The location of each vertex can be predicted using information of the spatial neighbor vertices as well as information of the corresponding vertices in the previous frame. Chen et al. (2017) proposed a method for cloth compression using local cylindrical coordinates which is suitable for cloth animation compression because the dihedral component of LCC describes the main feature of the animation sequence. One limitation is that this method cannot exploit the spatial coherence, which means that the compression ratio may be low if the triangles are much smaller than the granularity of wrinkles. Yang et al. (2018) proposed a method for compression and progressive transmission of dynamic mesh sequence exploiting both temporal and spatial redundancy to effectively reduce the data size of the dynamic sequence. Although the proposed compression algorithm has many advantages, there are still problems that remain to be solved, such as the adaptive progressive transmission of the mesh sequence and the perceptual metrics focus on local relations.

Anis et al. (2016) produced a scheme for the compression of dynamic 3D point clouds using sub-divisional meshes and graph wavelet transforms. Guskov and Khodakovsky (2004) proposed an algorithm that uses a wavelet coding method for mesh sequences. A progressive mesh hierarchy and an anisotropic wavelet are built for the first frame and maintained for subsequent frames. To exploit the temporal correlation, wavelet coefficients between adjacent frames are encoded differentially. A wavelet-based transform and coding scheme is also presented by El Sayeh Khalil et al. (2016), in which the proposed transform preserves geometric features at lower resolutions by adaptive vertex sampling and re-triangulation. Maglo et al. (2015) discussed in detail the problem of static and dynamic 3D mesh compression, summarizing a lot of novel approaches. They presented the main categories of the literature, give comparisons and evaluate the performance of the described algorithms.

Regarding the reconstruction of the dynamic 3D meshes, many works have also been presented Zhang and Xu (2018). Li et al. (2009) presented a framework for robust geometry and motion reconstruction of complex deforming shapes. However this method requires registration of each frame making it difficult to be applied in real-time of unregistered data. Li et al. (2012) presented a shape completion technique for creating temporally coherent watertight surfaces from real-time captured dynamic performances. A limitation of this method was that the unobserved regions in each frame have no geometric details in them. Süßmuth et al. (2008) described an approach for the reconstruction of animated meshes from a series of time deforming point clouds, given a set of unordered point clouds that have been captured by a fast 3D scanner. Due to the high memory consumption of the MPU approximation, this algorithm was currently limited to reconstruct point clouds containing a lot of points.

3. Overview of our method

In this section, we introduce the necessary definitions and terminology related to the dynamic 3D meshes and additionally, we explain in detail every step of our method. In Fig. 1, the proposed framework is briefly presented, highlighting the most important procedures of our approach. In a nutshell, we start with the layer decomposition process taking into account the spatial and temporal information of the dynamic mesh. The output of this process corresponds to the active points at the end of the removal process.[2] After the transmission, each reduced frame is reconstructed. Firstly, we use a weighted Laplacian interpolation approach, as a coarse reconstruction process, in order to estimate the position of the removed vertices. Then, we perform a fine estimation step by tracking the normals subspace deviation between different layers using ISVD, significantly reducing the required complexity as compared to a conventional SVD based approach. The convergence of this approach is significantly accelerated using Robust PCA (RPCA) as an initialization procedure. Finally, each frame is fine-reconstructed penalizing displacement of the vertices over a tangent plane perpendicular to the local surface normal.

3.1. Basic definitions of static and dynamic 3D meshes

We start assuming the existence of a sequence of n static meshes $\mathcal{M}(t)\ \forall\ t = 1 \cdots n$, creating a 3D mesh sequence (dynamic mesh), such as $\mathbf{A} = [\mathcal{M}(1);\ \mathcal{M}(2);\ \ldots\ \mathcal{M}(n)]$. Each individual mesh $\mathcal{M}$, consisting of k vertices, can be represented by two different sets $\mathcal{M} = (P, \mathcal{F})$ corresponding to the vertices P and the indexed faces $\mathcal{F}$ of the mesh. Each vertex can be represented as a point $v_i = (x_i, y_i, z_i)\ \forall\ i = 1 \cdots k$. In this case, we create a vector of vertices $\mathbf{v} = [\mathbf{x},\ \mathbf{y},\ \mathbf{z}]$ in a 3D coordinate space such as $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^{k \times 1}$ and $\mathbf{v} \in \mathbb{R}^{k \times 3}$. This means that we have a set of k points such that $P = \{v_1, v_2, \ldots, v_k\}$. Additionally, each face is defined by a set of 3 indices to vertices $f_i = [v_{i1},\ v_{i2},\ v_{i3}]\ \forall\ i = 1 \cdots k_f$ where $k_f > k$ for the watertight 3D meshes that we use in our research, so we have k_f faces $\mathcal{F} = \left\{ f_1, f_2, \ldots, f_{k_f} \right\}$. Each face constitutes a triangle, the simplest surface unit, that can be represented by its centroid point $\mathbf{c}_i$ and its outward unit normal $\mathbf{n}_{\mathbf{c}i}\ \forall\ i = 1 \cdots k_f$. We define as $\mathbf{C} \in \mathbb{R}^{k \times k}$ the binary adjacency matrix with the following elements:

$$\mathbf{C}_{ij} = \begin{cases} 1 & if\ i, j \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

[2] At this point, it should be noted that the number of the removed vertices depends on the network's capability per frame.

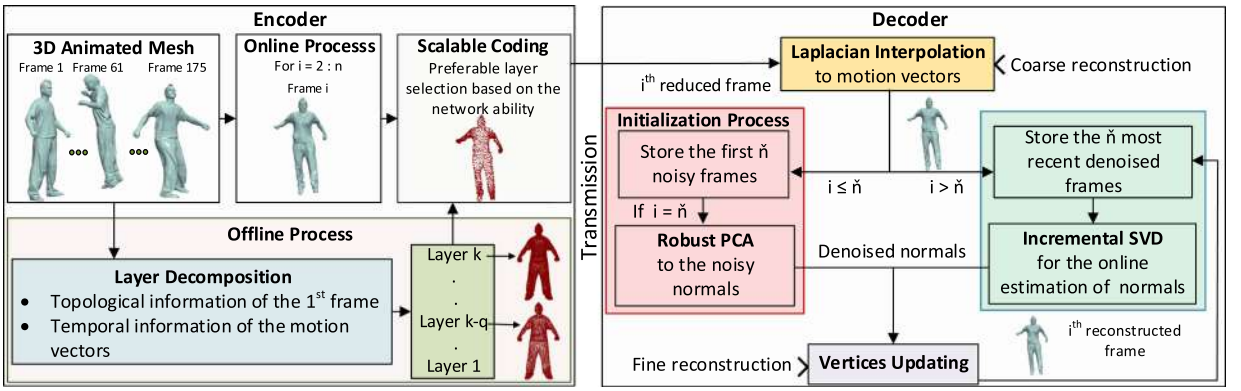

Fig. 1. The pipeline of the proposed method.

where $\mathcal{E}$ is the set of edges which can be directly derived from the sets P and $\mathcal{F}$. The connectivity is estimated once since it remains the same from frame to frame, in contrast with the position of vertices which changes. In this work, we assume that the topology is global, meaning that every static mesh of the same sequence has the same topology over time. Let $\mathbf{Z} \in \mathbb{R}^{m\times g}$ be a matrix which can be decomposed as $\mathbf{Z}_{m\times g} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$. We assume that its best κ-rank approximation could be described as $\hat{\mathbf{Z}}_{m\times g} = \mathbf{U}_\kappa \mathbf{\Lambda}_\kappa \mathbf{V}_\kappa^T$ where $\mathbf{U}_\kappa$ and $\mathbf{V}_\kappa$ are formed by the first κ columns of $\mathbf{U}$ and $\mathbf{V}$, respectively, and $\mathbf{\Lambda}_\kappa$ is the κ-th leading principal submatrix of $\mathbf{\Lambda}$.

3.2. Layer decomposition

In this paragraph, we propose a spatiotemporal layer decomposition algorithm for scalable coding of mesh sequences, similar to Ahn et al. (2013), which removes the vertices of a mesh taking into account both topological and temporal criteria. The vertex removal order is a vital process because it could affect the visual quality of each frame/mesh reconstruction influencing the prediction accuracy and hence the compression performance. We assume that only one vertex is removed at each layer so that k spatial layers are created. We notate as P_1 the set of points of the layer 1 which has only one vertex while in the highest layer k there is the set of points P_k consisting of k vertices. The relation between the set of points P_l and the exactly previous set P_{l-1} can be described as:

$$P_l = P_{l-1} \cup \{v\} \tag{2}$$

It is obvious that each layer has one more vertex in comparison to its exactly previous layer, and always $P_l \subseteq P \ \forall \ l = 1 \cdots k$. We need to mention here that the layer decomposition process is an offline procedure taking place before the start of the transmission.

3.2.1. Vertex removal using topology information

For the layer decomposition, we use an iterative process which removes one single vertex per iteration. Firstly, the removal cost for any candidate vertex is estimated and then the vertex with the lowest value is eliminated. The process is repeated k times until only one vertex will have remained in layer 1. The proposed removal cost function consists of two terms, namely the spatial C_s and the temporal C_t. The spatial factor is related to the geometry of the first frame and it is estimated as follows. We define as $R_j(i)$ the set of the j-ring neighbors of the i vertex and the $\hat{R}_j(i,l)$ as a subset of $R_j(i)$ with the remaining j-ring neighbors of i vertex in the l level. We also define as b the topological distance between any vertex v and its neighboring vertices. Topological distance shows the minimum number of edges with which two vertices are connected to each other, as shown in Fig. 2. The selected j value of $b_j \ \forall \quad j = 1 \cdots \infty$ is an important variable to determine the spatial prediction accuracy. We suggest the maximum value of j to be equal to 3 otherwise the process becomes time-consuming without providing any reconstruction benefit. At each layer, we remove this specific vertex which can be efficiently predicted by the reconstruction process. This is the reason why the vertex with the lowest value is selected. Finally, the removed vertex v_l at layer l is given by:

$$v_l = \underset{v\in M_l}{\arg\min}\, C(i,l) \tag{3}$$

where $C(i,l)$ is the removal cost of vertex i at layer l. We define the prediction inaccuracy $C_s(i,l)$ (spatial term) for vertex i at layer l as:

$$C_s(i,l) = \sum_{j=1}^{3} \frac{|R_j(i)| - |\hat{R}_j(i,l)|}{|R_j(i)|} \rho^j \tag{4}$$

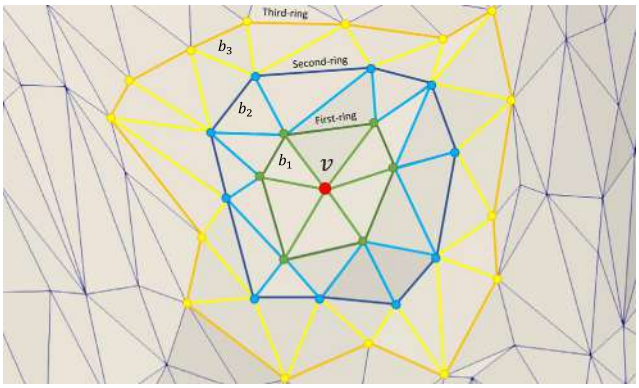

Fig. 2. Topological distance $b_j \ \forall \ \ j \in \{1, 2, 3\}$ between vertex v and other neighboring vertices. Vertices with topological distance equal to b_1 belong to first-ring area etc.

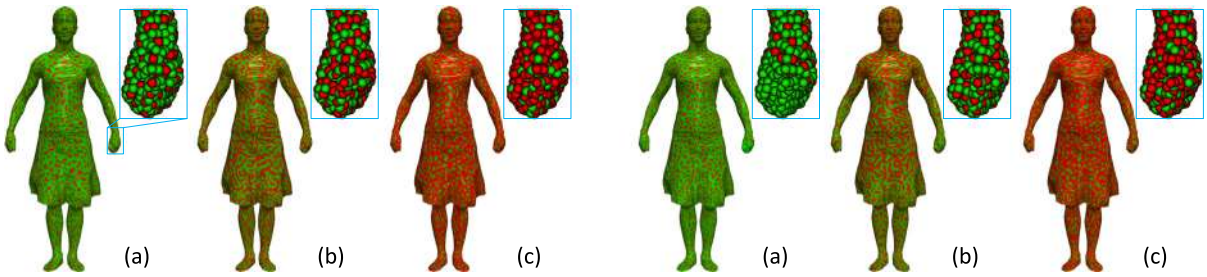

Fig. 3. [Left] Only topological information, [Right] Topological and temporal information (Samba of 9971 vertices). (a) 3000 vertices have been removed (layer P_{6971}), (b) 5000 vertices have been removed (layer P_{4971}), (c) 7000 vertices have been removed (layer P_{2971}). The green color represents the remaining vertices while the red color represents the removed vertices. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

where $|.|$ operator returns the number of elements in a set and ρ is a positive parameter. In the experiments, the fixed $\rho = 0.6$ is used for all the models. Note that $(|R_j(i)| - |\hat{R}_j(i,l)|)/|R_j(i)|$ becomes equal to 0 when $|\hat{R}_j(i,l)| = |R_j(i)|$, and it is equal to 1 when $|\hat{R}_j(i,l)| = 0$. In any case $|R_j(i)| \geq |\hat{R}_j(i,l)|$. Ahn et al. (2013) suggested the use of an extra factor which evaluates how each candidate vertex will affect its neighboring vertices if it is removed. However, its estimation is very time-consuming and the provided results do not appear any significant advantage. Instead of this factor, we propose the use of temporal information term C_t as described in the next paragraph.

3.2.2. Vertex removal using temporal information

Generally, a stationary or slow-motioned point is more likely to be accurately predicted. On the other hand, highly deformable surface patches are less accurately predictable. According to this observation, we propose a new factor that exploits the temporal information from frame to frame, taking into account the mean motion vector of each point, as shown below:

$$C_t(i) = \frac{\sum_{t=2}^{n} \|\mathbf{v}_i(t) - \mathbf{v}_i(t-1)\|_2}{n} \ \forall \ i = 1 \cdots k \tag{5}$$

where (t) represents the current frame while the (t-1) represents the previous frame. Then, the final removal cost is estimated as:

$$C(i,l) = C_s(i,l) + \lambda C_t(i) \tag{6}$$

where $\lambda = 0.1$. In Fig. 3, a frame of the animated model Samba is presented under different removal layers and using different removal cost functions. The difference between the two removal cost functions is more apparent in case Fig. 3-(a), wherein the second case more vertices of the hand are selected as active since they experience large deviations within frames as compared to other slowly varying vertices and consequently are expected to be predicted less accurately.

One of the strongest benefits of our approach is the fact that the layer decomposition algorithm sorts vertices based on their contribution to the accurate prediction of their real position, using both spatial and temporal criteria. In each layer decomposition step, the easiest predicted vertex is removed (i.e., this one that takes the lowest value of the cost function). This means that if time-variant wrinkles are apparent somewhere, then our algorithm will keep more vertices from this area and will remove vertices for other “more static” areas.

3.3. Laplacian interpolation to the motion vectors for coarse reconstruction

In this section, we describe the process for the online coarse reconstruction. For the reconstruction of each frame, the topological and geometrical information of the previously reconstructed frame is used. We follow the main idea of the method, proposed by Arvanitis et al. (2017b), but we take into account that the number of vertices, for any appeared incomplete frame, is variable and depends on the network's transmission ability.

3.3.1. Weighted graph Laplacian matrix

Generally, a binary Laplacian matrix provides information regarding the connectivity of vertices. However, weighted Laplacian matrices are able to provide additional information which can efficiently be utilized by a variety of other processes. In order to set the preferable constraints, we construct a modified weighted Laplacian matrix, similar to Arvanitis et al. (2018b), that takes into account the two following factors $\mathbf{H} \in \mathbb{R}^{k\times k}$ and $\mathbf{O} \in \mathbb{R}^{k\times k}$. Parameter $\mathbf{H}$ is related to the distance between connected vertices. The value of this parameter represents the inverse L^2 norm between two vertices v_i, v_j provided that v_i and v_j are connected to each other. It is estimated according to:

$$\mathbf{H}_{ij} = \begin{cases} \frac{1}{\|\mathbf{v}_i(t-1)-\mathbf{v}_j(t-1)\|_2+\epsilon} & \text{if } j \in R_1(i) \\ 0 & \text{otherwise} \end{cases} \quad \forall\, i = 1\cdots k \tag{7}$$

where ϵ is a very small positive number. For the estimation of this parameter, the values of the vertices from the previously reconstructed frame $(t-1)$ are used. Parameter $\mathbf{O}$ is related to the connecting proximity b (degree or topological distance) of an unknown vertex with an already known vertex. The initial known vertices have a value equal to 4 (reinforcing the contribution of the known values), while the value of the unknown vertices depends on their connectivity degree b, as shown in the following equation:

$$\mathbf{O}_{ij} = \begin{cases} 4\mathbf{C}_{ij} & \text{if } \mathbf{v}_i \text{ is known} \\ \frac{\mathbf{C}_{ij}}{b+1} & \text{otherwise} \end{cases} \quad \forall\, i, j = 1\cdots k \tag{8}$$

where $\mathbf{C}$ is the adjacency matrix as described in Eq. (1). Finally, the weighted adjacency matrix $\mathbf{C}_w$ is created using the estimated parameters $\mathbf{H}$ of Eq. (7) and $\mathbf{O}$ Eq. (8) according to:

$$\mathbf{C}_w = \mathbf{H} \circ \mathbf{O} \circ \mathbf{C} \tag{9}$$

where $\circ$ denotes the Hadamard product. Then, the weighted Laplacian matrix is estimated according to the following equation:

$$\mathbf{L}_w = \mathbf{D} - \mathbf{C}_w \tag{10}$$

where $\mathbf{D} = diag\{D_1, \ldots, D_k\}$ is a diagonal matrix with $D_i = \sum_{j=1}^{k} \mathbf{C}_{w_{ij}}$.

3.3.2. Weighted Laplacian interpolation

Oostendorp et al. (1989) first proposed that a triangulated 3D model can be interpolated with a curved surface by putting constraints on the Laplacian matrix $\mathbf{L}$. We follow the same line but applying it on the motion vectors $\boldsymbol{\delta}$ of the vertices instead of the vertices directly.

$$\boldsymbol{\delta}_i = [\delta_{xi},\ \delta_{yi},\ \delta_{zi}]^{\mathsf{T}} \begin{cases} \delta_{xi} = |v_{xi}(t) - v_{xi}(t-1)| \\ \delta_{yi} = |v_{yi}(t) - v_{yi}(t-1)| \\ \delta_{zi} = |v_{zi}(t) - v_{zi}(t-1)| \end{cases} \quad \forall\, i = 1\cdots k \tag{11}$$

Additionally, we use the estimated weighted Laplacian matrix $\mathbf{L}_w$ of Eq. (10) which encloses all the necessary constraints for an efficient weighted Laplacian interpolation. We also define $\mathbf{d} = [\boldsymbol{\delta}_1\ \boldsymbol{\delta}_2\ \cdots\ \boldsymbol{\delta}_k] \in \mathbb{R}^{k\times 3}$ the matrix which represents the motion vectors of each vertex of the mesh. The Laplacian of $\mathbf{d}$ is written as: $\Delta\boldsymbol{\delta} = \mathbf{L}_w\mathbf{d}$. Next, we split the $\mathbf{d}$ into two parts: $\mathbf{d_k} \in \mathbb{R}^{k'\times 3}$ containing the motion vectors of known vertices and $\mathbf{d_u} \in \mathbb{R}^{k-k'\times 3}$ containing zeros because of the unspecified values of the unknown vertices. Please note that $k-k'$ is equal to the decomposition layer. Correspondingly, the $\mathbf{L}_w$ can be partitioned into four parts: $\mathbf{L}_w = \begin{pmatrix} \mathbf{L}_{w_{11}} & \mathbf{L}_{w_{12}} \\ \mathbf{L}_{w_{21}} & \mathbf{L}_{w_{22}} \end{pmatrix}$, $\mathbf{L}_{w_{11}} \in \mathbb{R}^{k'\times k'}$, $\mathbf{L}_{w_{12}} \in \mathbb{R}^{k'\times k-k'}$, $\mathbf{L}_{w_{21}} \in \mathbb{R}^{k-k'\times k'}$, $\mathbf{L}_{w_{22}} \in \mathbb{R}^{k-k'\times k-k'}$. The Euclidean norm $|\Delta\boldsymbol{\delta}|$ is minimized according to:

$$\left| \begin{pmatrix} \mathbf{L}_{w_{11}} & \mathbf{L}_{w_{12}} \\ \mathbf{L}_{w_{21}} & \mathbf{L}_{w_{22}} \end{pmatrix} \begin{pmatrix} \mathbf{d_k} \\ \mathbf{d_u} \end{pmatrix} \right| = \left| \begin{pmatrix} \mathbf{L}_{w_{11}} \\ \mathbf{L}_{w_{21}} \end{pmatrix} \mathbf{d_k} + \begin{pmatrix} \mathbf{L}_{w_{12}} \\ \mathbf{L}_{w_{22}} \end{pmatrix} \mathbf{d_u} \right| \tag{12}$$

Then, we solve the following system of k equations with $k-k'$ variables:

$$\begin{pmatrix}\mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{22}}\end{pmatrix}\mathbf{d_u} = -\begin{pmatrix}\mathbf{L}_{w_{11}}\\ \mathbf{L}_{w_{21}}\end{pmatrix}\mathbf{d_k} \iff \mathbf{d_u} = -\left(\left(\begin{pmatrix}\mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{22}}\end{pmatrix}^{\mathsf{T}}\begin{pmatrix}\mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{22}}\end{pmatrix}\right)^{-1}\left(\begin{pmatrix}\mathbf{L}_{w_{12}}\\ \mathbf{L}_{w_{22}}\end{pmatrix}^{\mathsf{T}}\begin{pmatrix}\mathbf{L}_{w_{11}}\\ \mathbf{L}_{w_{21}}\end{pmatrix}\right)\mathbf{d_k} \tag{13}$$

For the fast solving of the Eq. (13), we follow the same line of thought as presented by Arvanitis et al. (2017a). We start by simplifying the representation of the parameters $\mathbf{H}$ and $\mathbf{O}$. We can say that the correspondingly matrices could be written as a set of k vectors of parameters with variance size depending on the first-ring area size: $\mathbf{H} = \{\mathbf{h}_1; \mathbf{h}_2; \cdots; \mathbf{h}_k\}$, $\mathbf{O} = \{\mathbf{o}_1; \mathbf{o}_2; \cdots; \mathbf{o}_k\}$ where $\mathbf{h}_i, \mathbf{o}_i \in \mathbb{R}^{|\hat{R}_1(i)|\times 1}\ \ \forall\ i = 1\cdots k$ and $\mathbf{d_{k}}_i$ consist of the known motion vectors of the neighbor vertices belonging on $\hat{R}_1(i)$. The solution is estimated based on an iterative process which is executed until $|\hat{R}_1(i)| = |R_1(i)|\ \ \forall\ i = 1\cdots k$.

$$\mathbf{d_u}_i = \frac{\sum_{\forall\ j\ \in \hat{R}_1(i)} \mathbf{h}_{ij}\mathbf{o}_{ij}\mathbf{d_k}_i}{\sum_{\forall\ j\ \in \hat{R}_1(i)} |\mathbf{h}_{ij}\mathbf{o}_{ij}|}\ \ \forall\ i = 1\cdots k \tag{14}$$

The closer the value of $|\hat{R}_1(i)|$ to $|R_1(i)|$ the faster the reconstruction process. For this reason the process is faster when the used decomposition layer is higher. Despite the fact that the geometric approach solves the same problem, described of our mathematical background in Eqs. (7)-(13), the execution is much faster. The coordinates of the missing vertices are estimated by updating their position of the previous frame using the motion vectors of Eq. (13).

$$\mathbf{v_u}(t) = \mathbf{v_u}(t-1) + \mathbf{d}_u,\ \ \forall\ t = 2\cdots n \tag{15}$$

Finally, all vertices of the incomplete frame (t) are known $\mathbf{v}(t) = \mathbf{v_k}(t) \cup \mathbf{v_u}(t)$ where $\mathbf{v_k} = [\mathbf{v_{k}}_1 \cdots \mathbf{v_{k}}_{k'}]$ and $\mathbf{v_u} = [\mathbf{v_u}_{k'+1} \cdots \mathbf{v_u}_{k-k'}]$. Please note that for the estimation of the motion vectors of the second frame, the first frame has to be known. Despite the extremely good results that this step provides, especially when the decomposition layer is high, there are some misaligned points affecting the visual quality of the final results. We refine these abnormalities following the procedure described below.

3.4. Online scalable coding using fine reconstruction

The coarse reconstructed method, provided by the previously presented step, demonstrates impressive performance. However, in cases where the scalable coding is responsible for highly incomplete frames $>$ 60% then noise appears in specific areas (e.g., nonrigid areas, areas with high motion between consecutive frames). To remove these abnormalities, a fine reconstruction step is utilized. Firstly, we estimate the ideal (denoised) centroid normals of all faces and then we use them to update the position of any vertex of the mesh using an iterative process. Although a deformation method, as this one presented by Huang et al. (2006), could be used as the main method for the fine reconstruction step, providing very plausible visual results, it would be very time consuming process for the final reconstruction of the whole dynamic 3D mesh.

3.4.1. Initialization strategy by exploiting outliers via RPCA

For the denoising of the first $\bar{n}$ frames, we follow a batch approach in order to exploit more effectively their coherence using RPCA. As a result, we initially create a spatiotemporal matrix $\mathbf{E} \in \mathbb{R}^{k_f \times 3\bar{n}}$ according to:

$$\mathbf{E} = \begin{bmatrix} \mathbf{n_{c}}_1(1) & \mathbf{n_{c}}_1(2) & \dots & \mathbf{n_{c}}_1(\bar{n}) \\ \mathbf{n_{c}}_2(1) & \mathbf{n_{c}}_2(2) & \dots & \mathbf{n_{c}}_2(\bar{n}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{n_{c}}_{k_f}(1) & \mathbf{n_{c}}_{k_f}(2) & \dots & \mathbf{n_{c}}_{k_f}(\bar{n}) \end{bmatrix} \tag{16}$$

where $\bar{n} \ll n$ and $\mathbf{n_{c}}_i(\bar{n}) = [\mathbf{n_{cx}}_i(\bar{n});\ \mathbf{n_{cy}}_i(\bar{n});\ \mathbf{n_{cz}}_i(\bar{n})]$ represents the ith centroid normal of $\bar{n}^{th}$ frame. To note here that the experimental process has shown that a patch with size $\bar{n} = [5 - 10]$ is enough.

The motivation for using RPCA, as initialization strategy, counts on the observation that the noisy (misaligned) normals of the coarse reconstructed mesh are also affected by outliers. This impulsive noise structure is attributed to the fact that most of the vertices keep their original position or have been already well reconstructed due to the weighted Laplacian interpolation step. Because of this observation, we assume that the presented type of noise is more like sparse outliers than noise with normal distribution. Then, the matrix $\mathbf{E}$, consisting of the k_f centroid normals of the first $\bar{n}$ frames, may be decomposed as: $\mathbf{E} = \mathbf{S} + \mathbf{N}$ where $\mathbf{S}$ is a low-rank matrix representing the real data while $\mathbf{N}$ is a sparse matrix representing the space where the noise lies. The low-rank matrix $\mathbf{S}$ can be recovered by solving the following convex optimization problem:

$$\text{minimize}\ \ \|\mathbf{S}\|_* + \lambda\|\mathbf{N}\|_1,\ \ \text{subject to}\ \ \mathbf{S} + \mathbf{N} = \mathbf{E} \tag{17}$$

where $\|\mathbf{S}\|_*$ denotes the nuclear norm of the matrix which is the sum of the singular values of $\mathbf{S}$. This convex problem can be solved using an Augmented Lagrange Multiplier (ALM) algorithm, as described in Lin et al. (2009):

$$l(\mathbf{S},\mathbf{N},\mathbf{Y},\mu) \doteq \|\mathbf{S}\|_* + \lambda\|\mathbf{N}\|_1 + \langle \mathbf{Y}, \mathbf{E}-\mathbf{S}-\mathbf{N}\rangle + \frac{\mu}{2}\|\mathbf{E}-\mathbf{S}-\mathbf{N}\|_F^2 \tag{18}$$

The ALM method for solving the RPCA problem can be described as an iterative process, estimating: $\mathbf{S}^{(h+1)} = \arg\min_{\mathbf{S}} l(\mathbf{S},\mathbf{N}^{(h)},\mathbf{Y}^{(h)},\mu^{(h)})$, $\mathbf{N}^{(h+1)} = \arg\min_{\mathbf{N}} l(\mathbf{S}^{(h+1)},\mathbf{N}^{(h)},\mathbf{Y}^{(h)},\mu^{(h)})$, where (h) represents the number of the iteration. For the estimation of matrices $\mathbf{S},\mathbf{N},\mathbf{Y}$ we iteratively compute the following equations:

$$(\mathbf{U},\mathbf{\Sigma},\mathbf{V}) = \text{SVD}(\mathbf{E}-\mathbf{N}^{(h)} + (1/\mu^{(h)})\mathbf{Y}^{(h)}) \tag{19}$$

$$\mathbf{S}^{(h+1)} = \mathbf{U}\mathcal{Q}_{(1/\mu^{(h)})}[\mathbf{\Sigma}]\mathbf{V}^T \tag{20}$$

$$\mathbf{N}^{(h+1)} = \mathcal{Q}_{\lambda\mu_h^{-1}}[\mathbf{E}-\mathbf{S}^{(h+1)} + (1/\mu^{(h)})\mathbf{Y}^{(h)}] \tag{21}$$

$$\mathbf{Y}^{(h+1)} = \mathbf{Y}^{(h)} + \mu^{(h)}(\mathbf{E}-\mathbf{S}^{(h+1)}-\mathbf{N}^{(h+1)}),\ \mu^{(h+1)} = \xi\mu^{(h)} \tag{22}$$

where $\mathcal{Q}_\nu[\mathbf{\Theta}]$ is a shrinkage operator which subtracts the value ν of every element of matrix $\mathbf{\Theta}$, $\mu^{(0)} = 1.25/||\mathbf{E}||_2$ and the iterative process terminates when $||\mathbf{E}-\mathbf{S}-\mathbf{N}||_F/||\mathbf{E}||_F < \gamma$ where the tolerance for stopping criterion is $\gamma = 10^{-7}$. Once the convergence criterion has been satisfied, the iterative process stops and the matrices $\mathbf{S}^{(h)},\mathbf{N}^{(h)} \in \mathbb{R}^{k_f\times 3\bar{n}}$ are returned. Then we use the elements of the low-rank matrix $\mathbf{S}$ to refine the $\bar{n}$ meshes updating the positions of their vertices. The low-rank matrix $\mathbf{S}$, consisting of the denoised normals represented as $\dot{\mathbf{n}}$. However, the process does not return unit normals so we need to normalized them, as shown in Eq. (23).

$$\mathbf{S} = \begin{bmatrix} \dot{\mathbf{n}}_{\mathbf{c}1}(1) & \dot{\mathbf{n}}_{\mathbf{c}1}(2) & \dots & \dot{\mathbf{n}}_{\mathbf{c}1}(\bar{n}) \\ \dot{\mathbf{n}}_{\mathbf{c}2}(1) & \dot{\mathbf{n}}_{\mathbf{c}2}(2) & \dots & \dot{\mathbf{n}}_{\mathbf{c}2}(\bar{n}) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{\mathbf{n}}_{\mathbf{c}k_f}(1) & \dot{\mathbf{n}}_{\mathbf{c}k_f}(2) & \dots & \dot{\mathbf{n}}_{\mathbf{c}k_f}(\bar{n}) \end{bmatrix},\quad \dot{\mathbf{n}}_{\mathbf{c}i}(j) = [\frac{\dot{\mathbf{n}}_{\mathbf{c}\mathbf{x}_i}(j)}{\|\dot{\mathbf{n}}_{\mathbf{c}i}(j)\|_2};\ \frac{\dot{\mathbf{n}}_{\mathbf{c}\mathbf{y}_i}(j)}{\|\dot{\mathbf{n}}_{\mathbf{c}i}(j)\|_2};\ \frac{\dot{\mathbf{n}}_{\mathbf{c}\mathbf{z}_i}(j)}{\|\dot{\mathbf{n}}_{\mathbf{c}i}(j)\|_2}]\ \forall\ j = 1\cdots\bar{n} \tag{23}$$

3.4.2. Online refining using ISVD

The previously mentioned step (initialization strategy) is applied once, only for the patch of the first $\bar{n}$ noisy frames. After that, we use the knowledge of the reconstructed frames in order to estimate the denoised normals of any new presented frame. In the literature, many of the proposed methods are trying to estimate the ideal normals. However, none of them are fast enough for real-time or online applications. To overcome this limitation, we suggest the use of an incremental approach. The SVD updating algorithm, described in detail in Zha and Simon (1999), Kwok and Zhao (2003), provides an efficient way to carry out the SVD of a larger matrix $[\mathbf{S}_{k_f\times 3\bar{n}}, \mathbf{B}_{k_f\times 3r}]$, where $\mathbf{B}$ is an $k_f \times 3r$ matrix consisting of the k_f centroid normals of the r additional frames. At this point it should be noted that the matrix $\mathbf{S}$ is an already low-rank matrix consisting of the denoised normals of the $\bar{n}$ previous frames. Specifically, for the normal's estimation of the $\bar{n}+1$ frame, the matrix $\mathbf{S}$, as described in Eq. (23), is used, while for any frame $> \bar{n}+2$ the matrix $\mathbf{S}$ is updated as we will show later in Eq. (27). The $r \le \bar{n}$ represents the number of the observed noisy frames where we want to estimate the denoise normals. Typically, $r = 1$ because each new frame appears online, however, if a buffer could store a sequence of coarse reconstructed meshes then more frames (block of frames) could be used, increasing the benefits that ISVD method provides regarding the execution times. By exploiting the orthonormal properties and block structure, the SVD computation of $[\mathbf{S},\mathbf{B}]$ can be efficiently carried out by using the smaller matrices, $\mathbf{U}_q, \mathbf{V}_q$, and the SVD of the smaller matrix $\begin{bmatrix} \mathbf{\Lambda}_q & \mathbf{U}_q^T\mathbf{B} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}$. The computational complexity analysis and details of the SVD updating algorithm are described in Zha and Simon (1999). The steps of the ISVD method that we use are described in the next Eqs. (24) - (26). Firstly, we apply a qr(.) decomposition of the $(\mathbf{I}-\mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}$ in order to estimate the matrices $\mathbf{Q}$ and $\mathbf{R}$:

$$\mathbf{QR} = \text{qr}((\mathbf{I}-\mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}) \tag{24}$$

Next, we obtain the q-rank SVD of the $(q+r)\times(q+r)$ matrix:

$$\begin{bmatrix} \mathbf{\Lambda}_q & \mathbf{U}_q^T\mathbf{B} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} = \hat{\mathbf{U}}\hat{\Lambda}\hat{\mathbf{V}}^T \tag{25}$$

where r' is the rank of $(\mathbf{I}-\mathbf{U}_q\mathbf{U}_q^T)\mathbf{B}$. Then, the best q-rank approximation of $[\mathbf{S},\mathbf{B}]$ is:

$$[\mathbf{S},\mathbf{B}] = ([\mathbf{U}_q,\mathbf{Q}]\hat{\mathbf{U}})\hat{\mathbf{\Lambda}}(\begin{bmatrix} \mathbf{V}_q & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}\hat{\mathbf{V}})^T \tag{26}$$

Finally, the matrix $\mathbf{B}$ obtains the denoised normals of the new frame which will be used to update the vetrices. Matrix $\mathbf{S}$ will be updated, by a left shifting operation denoting as $\mapsto$, in order to obtain the most recent information for more efficient online estimation of the denoised normals of the next frame, according to:

$$\mathbf{S}: (\dot{\mathbf{n}}_{\mathbf{c}1},\ \dot{\mathbf{n}}_{\mathbf{c}2},\ \cdots,\ \dot{\mathbf{n}}_{\mathbf{c}(\bar{n}-1)},\ \dot{\mathbf{n}}_{\mathbf{c}\bar{n}}) \mapsto (\dot{\mathbf{n}}_{\mathbf{c}2},\ \dot{\mathbf{n}}_{\mathbf{c}3},\ \cdots,\ \dot{\mathbf{n}}_{\mathbf{c}\bar{n}},\ \mathbf{B}) \tag{27}$$

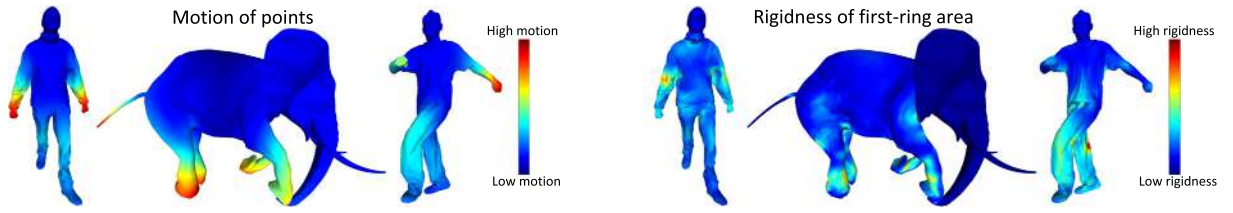

Fig. 4. Heat map visualization showing: [Left] the total distance of each centroid that travels through the frames, [Right] the total change of the first-ring area of each centroid. The heat maps have been applied in a random snapshot of each 3D animation models (Crane, Elephant gallop and March 2 respectively). The deep blue color represents low changes and the dark red represents high changes, as also shown in the provided colormaps.

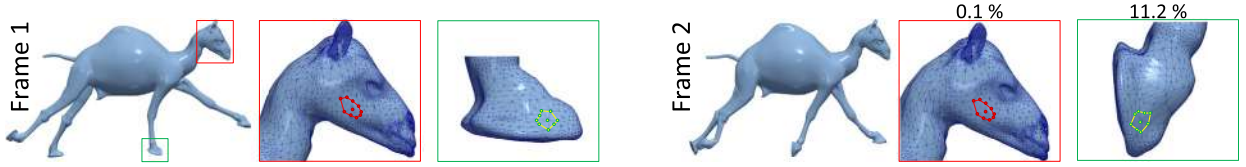

Fig. 5. Two consecutive frames of the animated mesh Camel gallop. Different areas have different behavior, with respect to the change of their shape from frame to frame, depending on the type of motion that each area manifests.

where $\dot{\mathbf{n}}_{\mathbf{c}i}$ represents the ith row of matrix $\mathbf{S}$. Please note that similar to the previous step using RPCA, the centroid normals must be normalized again so that any normal to be equal to the unit vector.

3.5. Ideal normals for the vertex updating

As we referred earlier, the apriori knowledge of the denoised normals of a mesh could be effectively used for fine denoising. The next step is to use the centroid normals, estimated in the previous steps, for achieving online fine reconstruction. In the past, a lot of researchers have adopted the same vertex updating algorithm, firstly described in Sun et al. (2007), mainly because of its robustness and the very good provided results. We follow the same line of thought but we give different reliability to different points. More specifically, we assume that some points are more reliable than others. Two parameters affect mostly the classification of a point as trustable or not. The first one is the rigidness of the area (first-ring) where a point lies (please see Fig. 4-[Right]) and the second is the total distance that a point covers through the frames (please see Fig. 4-[Left]).

More specifically, in regard to the first parameter, we suggest that only vertices with a significant change ($> 1\%$) of their first-ring area between two frames will be updated, otherwise, we assume that they have been correctly reconstructed under the Laplacian interpolation step. The criterion that is used for the characterization of a point as rigid or not depends on the percentage change of its first-ring area between two consecutive frames and it is described in the Eq. (28):

$$\Phi(i) = \begin{cases} 1 & \text{if } \frac{|\text{area}(i,l) - \text{area}(i,l-1)|}{\max(\text{area}(i,l), \text{area}(i,l-1))} > 1\% \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

where $\text{area}(i,l)$ is the first-ring area of point i as it appears in the l frame and $\Phi(i) = 1$ means that the i point needs to be updated for more accurate results. An example of two areas (rigid and no-rigid) is shown in Fig. 5. Rigid area (e.g., head of camel) means better reconstruction using Laplacian interpolation regardless of the value of the motion vectors.

Regarding to the second parameter, the temporal factor C_t is used, as it has been defined in Eq. (5). Then the updated weights are estimated according to Eq. (30), giving emphasis to the known and less moving points. Additionally, in order to make the process more time efficient, the known points included in the P_l set, are excluded from the updating process. The fine-tuned normals are then used to update the vertices according to Sun et al. (2007). The stability and the robustness of this approach has made it very popular and it has been used in a lot of other papers Zhang et al. (2015), Wang et al. (2016), Wang et al. (2015), Yadav et al. (2017), Arvanitis et al. (2018a).

$$\mathbf{v}_i^{(e+1)} = \mathbf{v}_i^{(e)} + \frac{\sum_{j \in \boldsymbol{\Psi}_i} \boldsymbol{\beta}_j \bar{\mathbf{n}}_{cj} (\langle \bar{\mathbf{n}}_{cj}, (\mathbf{c}_j^{(e)} - \mathbf{v}_i^{(e)}) \rangle)}{|\boldsymbol{\Psi}_i|} \; \forall \, i \notin P_l, \; \Phi(i) = 1 \tag{29}$$

$$\boldsymbol{\beta}_i = \begin{cases} 4C_t(i) & \text{if } \mathbf{v}_i \text{ is known} \\ C_t(i) & \text{otherwise} \end{cases} \tag{30}$$

$$\mathbf{c}_j^{(e+1)} = (\mathbf{v}_{j1}^{(e+1)} + \mathbf{v}_{j2}^{(e+1)} + \mathbf{v}_{j3}^{(e+1)})/3 \; \forall \, j \in \boldsymbol{\Psi}_i \tag{31}$$

where $\langle \mathbf{a}, \mathbf{b} \rangle$ represents the dot product of $\mathbf{a}$ and $\mathbf{b}$, (e) represents the number of iteration and matrix $\boldsymbol{\Psi}_i$ is the cell of vertices that are directly connected with the vertex $\mathbf{v}_i$. This iterative process can be considered as a gradient descent process

that is executed for minimizing the energy term $\sum_{j\in\Psi_i} \|\bar{\mathbf{n}}_{cj}(\mathbf{c}_j^{(e)} - \mathbf{v}_i^{(e)})\|_2$ across all faces. Finally, we need to mention here that for any of the used animated models, the ideal number of iterations e for an efficient fine reconstruction, was only 1-2. This is due to the fact that the coarse reconstructed frames are already denoised and need just a refinement through the aforementioned approach so as to remove outliers. Algorithm 1 briefly presents the steps of the proposed process.

Algorithm 1 Scalable Coding of Dynamic 3D Meshes.

```
   // Offline and Online Processes in the Encoder Side
   Input  : Dynamic 3D mesh
   Output: k different Layers (set of vertices) and reduced frames
   // Offline Process (Layer Decomposition)
 1 for l = 1 ··· k do
 2 |  Estimate the set of vertices P_l based on topological and temporal information via Eq. (6);
 3 end
   // Online Process (Scalable Coding)
 4 for i = 1 ··· n do
 5 |  Choose which layer P_l (set of vertices) you would send based on network ability
 6 end
   // Online Process in the Decoder Side (Reconstruction)
   Input  : Reduced frames of the dynamic 3D mesh
   Output: Reconstructed dynamic 3D mesh
 7 for i = 1 ··· n do
 8 |  Coarse reconstruction based on Laplacian interpolation to the motion vectors via Eqs. (7)-(15);
   |     // Denoised normals and ouliers removal
 9 |  Creation of spatiotemporal matrix E Eq. (16);
   |  if i < n then
10 |  |  Estimation of low-rank matrix based on Robust PCA via Eqs. (17)-(23);
11 |  else
12 |  |  Online refining using Incremental SVD via Eqs. (24)-(27);
13 |  end
14 |  Fine reconstruction using the vertices updating Eqs. (28)-(31);
15 end
```

4. Results

The effectiveness of our approach in terms of both reconstruction quality and computation complexity is highlighted through a thorough experimental study, presented in this section.

4.1. Experimental setup, datasets and metrics

In all the presented experiments, a PC Intel core i7-4790 CPU @ 3.60GHz, 16 GB RAM was used. The main core of the algorithms is written in C++ and Matlab.

We have used a wide range of 3D animated models. More specifically, the experiments and any other presented figures of this work are carried out using two different well-known 3D animation datasets such as: (i) A dataset consisting of artificial sequences of moving animal models James and Twigg (2005), (ii) A dataset consisting of motion capture animations representing humans in different kind of movement actions Vlasic et al. (2008).

The quality of the reconstructed results is evaluated using a variety of different metrics. (i) θ: represents the angle between the normal of the ground truth face and the resulting face normals, averaged over all faces. (ii) NMSVE (Normalized Mean Square Visual Error): has been shown to correlate well with perceived distortion by measuring the average error in the Laplacian and Cartesian domains Karni and Gotsman (2000). (iii) KG error metric: has been introduced by Karni and Gotsman and it has been designed for the evaluation of animated triangle meshes. The detailed description of this metric is presented in Karni and Gotsman (2004). (iv) STED (Spatiotemporal edge difference): focus on the local changes of the error rather than on the absolute value of it. More details about this metric are presented in Vasa and Skala (2011). The two first metrics (θ, NMSVE) are commonly used for the evaluation of each 3D mesh separately, while the last two (KG, STED) are used for the evaluation of the whole dynamic mesh as an object.

4.2. Scalable coding evaluation

In this section, we evaluate the benefits of the proposed pipeline in terms of both reconstruction quality and execution time. In particular, we explicitly show how this method could be efficiently used and why we chose each one of the selected steps.

	F/B	ISVD	SVD	Speed up
Jumping	1	**0.10** (0.100)	**0.98** (0.980)	9.8x
	10	**0.24** (0.024)	**4.15** (0.415)	17.2x
	50	**1.10** (0.022)	**9.45** (0.189)	8.59x
	75	**1.43** (0.019)	**11.52** (0.153)	8.05x
Camel	1	**0.28** (0.280)	**4.61** (4.610)	16.46x
	8	**0.48** (0.060)	**17.24** (2.155)	35.91x
	12	**0.61** (0.050)	**24.20** (2.016)	14.76x
	24	**1.30** (0.054)	**45.84** (1.910)	35.26x
March 2	1	**0.11** (0.110)	**0.88** (0.880)	8x
	25	**0.57** (0.022)	**10.11** (0.404)	17.73x
	50	**1.15** (0.023)	**12.12** (0.242)	10.53x
	125	**2.59** (0.020)	**16.51** (0.132)	6.37x
Elephant	1	**0.52** (0.520)	-	-
	8	**1.61** (0.201)	-	-
	12	**1.73** (0.144)	-	-
	24	**2.90** (0.120)	-	-

Fig. 6. Execution times by using ISVD Vs. SVD. The numerical results are expressed in seconds.

Table 1
Execution times per each frame for different decomposition layers and different models.

Name of model	Decomposition layer	Execution time per frame in (sec.) (linear approach)	Execution time per frame in (sec.) & speed up (geometric approach)
Bouncing	P_{8000}	1.392561	0.040240 (34.61x)
	P_{6000}	2.739982	0.066960 (40.92x)
	P_{4000}	3.691753	0.085024 (43.42x)
	P_{2000}	8.080768	0.111490 (72.48x)
Crane	P_{8000}	1.422852	0.041686 (34.13x)
	P_{6000}	2.665091	0.065021 (40.99x)
	P_{4000}	3.682691	0.082462 (44.66x)
	P_{2000}	8.132842	0.144083 (56.44x)
Handstand	P_{8000}	1.401952	0.039392 (35.59x)
	P_{6000}	2.604363	0.060651 (42.94x)
	P_{4000}	3.714263	0.078702 (47.19x)
	P_{2000}	8.293463	0.125693 (65.98x)
Jumping	P_{8000}	1.396755	0.041670 (33.52x)
	P_{6000}	2.798114	0.065335 (42.83x)
	P_{4000}	3.721518	0.083198 (44.73x)
	P_{2000}	8.378740	0.122232 (68.55x)
Swing	P_{8000}	1.171689	0.034310 (34.15x)
	P_{6000}	2.403907	0.059041 (40.72x)
	P_{4000}	3.481418	0.076251 (45.66x)
	P_{2000}	7.726545	0.122875 (62.88x)

4.2.1. Impact of using ISVD

ISVD is used not only because it allows an online solution but also because it is a very fast process especially when it is used in blocks of frames. Specifically, the larger the size of a block (e.g., frames per block (F/B)) the more apparent the benefits of the ISVD, regarding the execution time. ISVD is a vital step for this research because it allows the online reconstruction of frames in acceptable rates. Fig. 6 shows the execution times of incremental and traditional SVD, for different animated models and for different blocks of frames. Observing the table, it is obvious that ISVD is much faster than the traditional SVD (from 8 to 35.91 times faster). The bold values represent the execution times per block of frames while the values in the parenthesis represent the execution times per frame. To mention here that an extra benefit of ISVD is that it can be used for large matrices too, like in the case of Elephant model (last lines of Fig. 6). On the other hand, SVD cannot be directly applied without extra modification (e.g., separation in parts). Table 1 presents the execution times of the coarse reconstruction step using two different approaches, mentioned as “linear” and “geometric”. More specifically, the “linear” approach directly solves the linear system of Eq. (13) while the “geometric” approach uses the iterative process of Eq. (14) for the solution of the system.

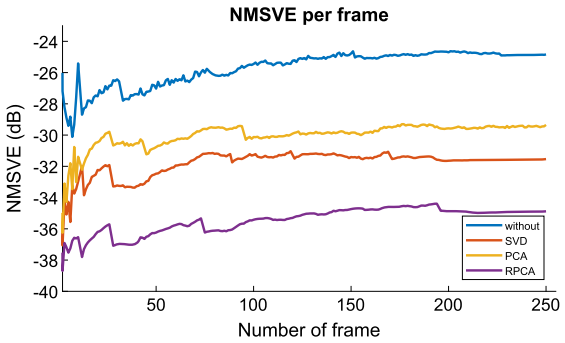

Fig. 7. NMSVE error per frame in (dB) for the dynamic 3D model March 2. Each line represents a different initialization strategy.

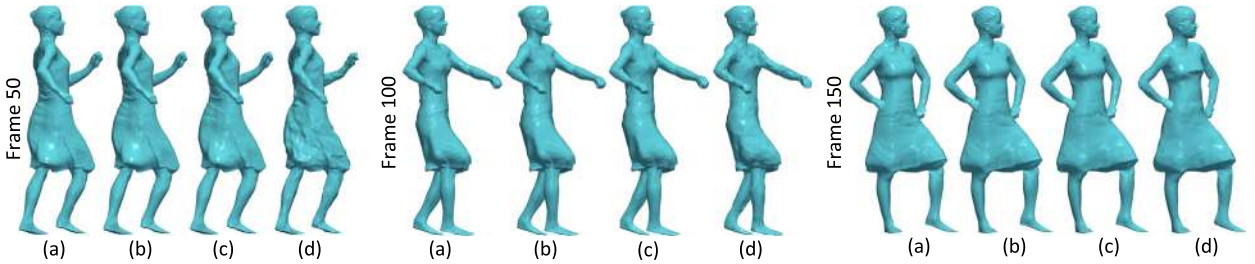

Fig. 8. Dynamic 3D mesh Samba (9971 points each frame). (a) layer P_{6971} (3000 points have been removed $\sim 30\%$), (b) layer P_{4971} (5000 points have been removed $\sim 50\%$), (c) layer P_{2971} (7000 points have been removed $\sim 70\%$), (d) layer P_{971} (9000 points have been removed $\sim 90\%$).

4.2.2. Impact of using RPCA as initialization strategy

As mentioned earlier in Paragraph 3.4.1, for efficiently applying the ISVD step, some initial knowledge is required. More specifically, the low-rank matrix of any new entrance is estimated using the updated matrix $\mathbf{S}$ which contains the values of the $\bar{n}$ previous frames, as described in Eq. (27). However, for the first $\bar{n}$ frames, this knowledge is missing. The proposed method fails if the initial matrix $\mathbf{S}$ is not already low-rank (e.g., using the noisy frames without further processing). Alternative approaches, returning a low-rank matrix of the first $\bar{n}$ frames, could also be used (e.g., SVD, PCA) as an initialization strategy. However, RPCA satisfies simultaneously both the reconstruction quality and the lower computational complexity, as shown in Fig. 7.

4.3. Experimental analysis

The effectiveness of our method is presented in the following figures. The results of our proposed method are compared with: (a) A layer decomposition approach using an Efficient Fine-granular Scalable Coding Algorithm (EFSCA) of 3D mesh sequences for low-latency streaming applications, as described in details in Ahn et al. (2013). (b) The Frame-based Animated Mesh Compression (FAMC) method which is promoted within the MPEG-4 standard Mamou et al. (2008). This method uses different types of transforms to encode the residual motion compensation error, namely (1) the Discrete Cosine Transform (DCT) and (2) the integer to integer lifting-based bi-orthogonal wavelet transform.

In Fig. 8, we present the reconstructed results using the pipeline of our proposed method for different frames (50, 100, 150) of the same dynamic mesh (Samba). In this example, we have assumed that any frame of the animation has been compressed with the same compression rate, through the same compression scenario ($\sim 30\%, 50\%, 70\%, 90\%$), showing how the reconstructed dynamic mesh is affected because of a consistent compression rate. Fig. 9 shows how the NMSVE metric changes during the frames of the dynamic mesh under of different experimental scenarios. Targeting a more realistic scenario, we assume that the change of layers happens in blocks of frames, each block consisting of 50 consecutive mesh instances. Specifically, the experimental scenarios that we studied are: (i) Stable coding: the points of any frame of the dynamic mesh is reduced based on the same layer (P_{2000}, P_{4000}, P_{6000} correspondingly). (ii) Random coding: the points of each block of frames is reduced based on a random layer in a range of (P_{500}-P_{9500}). (iii) Scalable coding: means that the points of each block of frames are reduced based on a layer which is lower of the layer of the previous block of frames using an incremental step of 1000. The initial layer of the first block of frames is P_{8000} (2002 points have been removed). We can observe that the NMSVE error is much lower in the first frames, where the motion of the object has not started yet (relatively stationary frames). However, after that, the error is slightly increased but without being affected by the type of the motion of the object.

As shown in this figure, the random coding, which typically is more close to a realistic scenario (network with variable transmission capability), has overall better results. It should be noticed that in this scenario it is possible to remove 9502

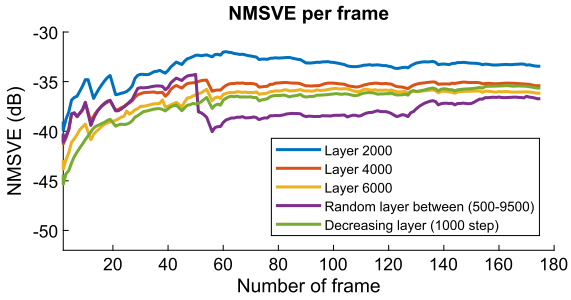

Fig. 9. NMSVE error expressed in (dB) per frame for the dynamic 3D model Bouncing. Each line of this plot represents a different experimental scenario.

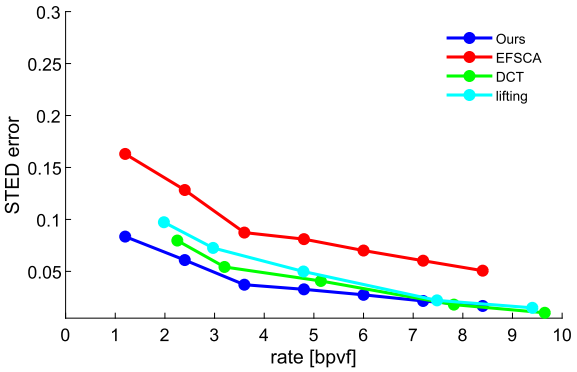

Fig. 10. STED error of the reconstructed results using different approaches and different rates of bpvf. (Jumping Model).

points (in layer P_{500}) from a frame which is equal to 95% of the dynamic mesh (for the Bouncing model consisting of 10,002 points).

Fig. 10 illustrates the STED error through the change of bpvf rate for different approaches. We can see that in high rates of bpvf, our proposed method has a very similar response to the two approaches of FAMC, however, when the bpvf is decreased our approach seems performs better. In Fig. 11, we present comparisons between our method and others using different rates of bpvf. For the evaluation we use the KG error metric. In Fig. 12, we present the heatmap visualization of θ metric for different reconstructed models. Additionally, we provide the mean θ of each frame for the different approaches, as well as enlarged detail of the reconstructed models for easier comparison. Besides the good results that the approaches of the FAMC method provide, a significant disadvantage of this method is the fact that it compresses the whole animation in a file using the same rate of bit-per-vertex for any frame. However, this approach is inappropriate for online applications where the bandwidth is unstable. To simulate variable bandwidth capabilities and adjust rates for chunks we separate the whole animation in blocks-of-frames (e.g., 10 frames per block) and at any block, a different bpvf is used. Note, that even if our method does not theoretically guarantee non-occurrence of tangled meshes, nevertheless we use centroid normals with always positive direction (outside of the 3D object) so as to practically never face this problem.

5. Conclusions and limitations

In this work, we presented an efficient approach for online scalable coding of dynamic 3D meshes. This method is totally parameter free and it can be used without further changes or extra parameterization. A significant advantage of the proposed method is its ability to transmit different bpv per each frame depending on the instant network's capability. Additionally, the selection of the transmitted vertices is optimized, taking into account both the spatial and temporal information. This gives an extra benefit to the reconstruction process to handle more efficiently the received vertices providing more accurate results. The main objective of our method is to provide the most efficient solution combining both the low execution time and the high perception quality of the reconstructed result (using a variety of evaluations metrics, namely θ, NMSVE, KG error). A very fast reconstruction process with bad results or high-quality results in a time-consuming process is not acceptable.

However, despite the effectiveness demonstrated by a variety of presented experiments and metrics, there are still some limitations. The transmitted dynamic mesh must be known in advance in order to create the decomposed layers. For this reason, applications, which require simultaneous scanning and transmission of the captured frames of a moving object, need an extra processing step so that each frame to have the same number of vertices and the same connectivity.

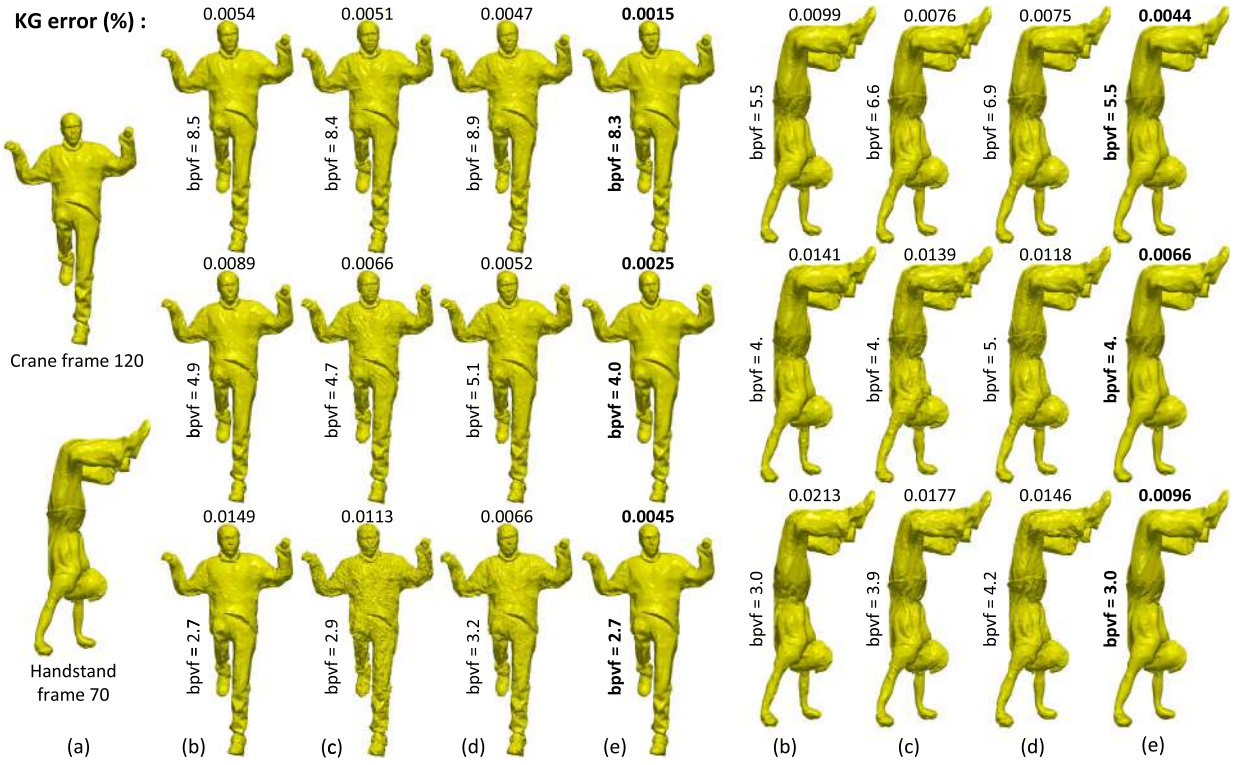

Fig. 11. KG error of the reconstructed results using different approaches and different rates of bpvf. (a) Original mesh and reconstructed using: (b) EFSCA Ahn et al. (2013), (c) the lifting approach of the FAMC method Mamou et al. (2008), (d) the DCT approach of the FAMC method Mamou et al. (2008), (e) our approach.

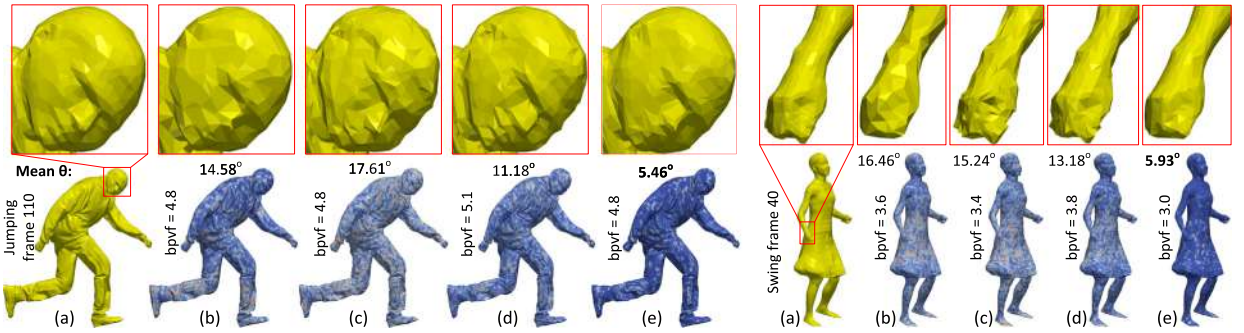

Fig. 12. Heatmap visualizing the θ metric per face in different colors. The mean θ of the reconstructed mesh is also provided. (a) Original mesh and reconstructed using: (b) EFSCA Ahn et al. (2013), (c) the lifting approach of the FAMC method Mamou et al. (2008), (d) the DCT approach of the FAMC method Mamou et al. (2008), (e) our approach.

Declaration of competing interest

No conflict of interest.

Appendix A. Supplementary material

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.cagd.2019.07.005.

References

Ahn, J.K., Koh, Y.J., Kim, C.S., 2013. Efficient fine-granular scalable coding of 3d mesh sequences. IEEE Trans. Multimed. 15, 485–497. https://doi.org/10.1109/TMM.2012.2235417.

Anis, A., Chou, P.A., Ortega, A., 2016. Compression of dynamic 3d point clouds using subdivisional meshes and graph wavelet transforms. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 6360–6364.

Arvanitis, G., Lalos, A., Moustakas, K., Fakotakis, N., 2018a. Feature preserving mesh denoising based on graph spectral processing. IEEE Trans. Vis. Comput. Graph., 1. https://doi.org/10.1109/TVCG.2018.2802926.

Arvanitis, G., Lalos, A.S., Moustakas, K., Fakotakis, N., 2017a. Fast and effective dynamic mesh completion. In: 25 International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision WSCG 2017, pp. 125–132.

Arvanitis, G., Lalos, A.S., Moustakas, K., Fakotakis, N., 2017b. Weighted regularized laplacian interpolation for consolidation of highly-incomplete time varying point clouds. In: 2017 3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), pp. 1–4.

Arvanitis, G., Spathis-Papadiotis, A., Lalos, A.S., Moustakas, K., Fakotakis, N., 2018b. Outliers removal and consolidation of dynamic point cloud. In: 2018 25th IEEE International Conference on Image Processing (ICIP), pp. 3888–3892.

Chen, J., Zheng, Y., Song, Y., Sun, H., Bao, H., Huang, J., 2017. Cloth compression using local cylindrical coordinates. Vis. Comput. 33, 801–810. https://doi.org/10.1007/s00371-017-1389-2.

Desai, K., Raghuraman, S., Jin, R., Prabhakaran, B., 2017. Qoe studies on interactive 3d tele-immersion. In: 2017 IEEE International Symposium on Multimedia (ISM), pp. 130–137.

El Sayeh Khalil, J., Munteanu, A., Denis, L., Lambert, P., Walle, R., 2016. Scalable feature-preserving irregular mesh coding. Comput. Graph. Forum 36, 275–290. https://doi.org/10.1111/cgf.12938. https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12938.

Elbamby, M.S., Perfecto, C., Bennis, M., Doppler, K., 2018. Towards low-latency and ultra-reliable virtual reality. CoRR.abs arXiv:1801.07587.

Guskov, I., Khodakovsky, A., 2004. Wavelet compression of parametrically coherent mesh sequences. In: Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. Eurographics Association, Goslar Germany, Germany, pp. 183–192.

Hachani, M., Zaid, A.O., Puech, W., 2016. Rate-distortion optimized compression algorithm for 3d triangular mesh sequences. In: 2016 Data Compression Conference (DCC), p. 600.

Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y., 2006. Subspace gradient domain mesh deformation. In: ACM SIGGRAPH 2006 Papers. ACM, New York, NY, USA, pp. 1126–1134.

Ibarria, L., Rossignac, J., 2003. Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In: Symposium on Computer Animation.

Isenburg, M., Lindstrom, P., Snoeyink, J., 2005. Streaming Compression of Triangle Meshes. In: Desbrun, M., Pottmann, H. (Eds.), Eurographics Symposium on Geometry Processing 2005, The Eurographics Association.

James, D.L., Twigg, C.D., 2005. Skinning mesh animations. In: ACM SIGGRAPH 2005 Papers. ACM, New York, NY, USA, pp. 399–407.

Karni, Z., Gotsman, C., 2000. Spectral compression of mesh geometry. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 279–286.

Karni, Z., Gotsman, C., 2004. Compression of soft-body animation sequences. Comput. Graph. 28, 25–34. https://doi.org/10.1016/j.cag.2003.10.002.

Kwok, J.T., Zhao, H., 2003. Incremental eigen decomposition. In: PROC. ICANN, pp. 270–273.

Li, H., Adams, B., Guibas, L.J., Pauly, M., 2009. Robust single-view geometry and motion reconstruction. In: ACM SIGGRAPH Asia 2009 Papers. ACM, New York, NY, USA, pp. 175:1–175:10. http://doi.acm.org/10.1145/1661412.1618521.

Li, H., Luo, L., Vlasic, D., Peers, P., Popović, J., Pauly, M., Rusinkiewicz, S., 2012. Temporally coherent completion of dynamic shapes. ACM Trans. Graph. 31, 2:1–2:11. https://doi.org/10.1145/2077341.2077343. http://doi.acm.org/10.1145/2077341.2077343.

Lin, Z., Chen, M., Ma, Y., 2009. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. CoRR abs/1009.5055.

Maglo, A., Lavoué, G., Dupont, F., Hudelot, C., 2015. 3d mesh compression: Survey, comparisons, and emerging trends. ACM Comput. Surv. 47, 44:1–44:41. https://doi.org/10.1145/2693443. http://doi.acm.org/10.1145/2693443.

Mamou, K., Zaharia, T., Preteux, F., 2008. Famc: The mpeg-4 standard for animated mesh compression. In: 2008 15th IEEE International Conference on Image Processing, pp. 2676–2679.

Matusik, W., Pfister, H., 2004. 3d tv: A scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. ACM Trans. Graph. 23, 814–824. https://doi.org/10.1145/1015706.1015805. http://doi.acm.org/10.1145/1015706.1015805.

Mekuria, R., Sanna, M., Izquierdo, E., Bulterman, D.C.A., Cesar, P., 2014. Enabling geometry-based 3-d tele-immersion with fast mesh compression and linear rateless coding. IEEE Trans. Multimed. 16, 1809–1820. https://doi.org/10.1109/TMM.2014.2331919.

Mueller, K., Smolic, A., Merkle, P., Kautzner, M., Wiegand, T., 2004. Coding of 3 d meshes and video textures for 3 d video objects.

Oostendorp, T.F., van Oosterom, A., Huiskamp, G., 1989. Interpolation on a triangulated 3d surface. J. Comput. Phys. 80, 331–343. https://doi.org/10.1016/0021-9991(89)90103-4.

Peng, J., Kim, C.S., Kuo, C.C.J., 2005. Technologies for 3d mesh compression: a survey. J. Vis. Commun. Image Represent. 16, 688–733. https://doi.org/10.1016/j.jvcir.2005.03.001.

Stefanoski, N., Liu, X., Klie, P., Ostermann, J., 2007. Scalable linear predictive coding of time-consistent 3d mesh sequences. In: 2007 3DTV Conference, pp. 1–4.

Subramanyam, S., Cesar, P., 2018. Enhancement layer inter frame coding for 3d dynamic point clouds. In: 2018 IEEE Games, Entertainment, Media Conference (GEM), pp. 332–337.

Sun, X., Rosin, P., Martin, R., Langbein, F., 2007. Fast and effective feature-preserving mesh denoising. IEEE Trans. Vis. Comput. Graph. 13, 925–938. https://doi.org/10.1109/TVCG.2007.1065.

Süßmuth, J., Winter, M., Greiner, G., 2008. Reconstructing animated meshes from time-varying point clouds. In: Proceedings of the Symposium on Geometry Processing. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, pp. 1469–1476. http://dl.acm.org/citation.cfm?id=1731309.1731332.

Thanou, D., Chou, P.A., Frossard, P., 2016. Graph-based compression of dynamic 3d point cloud sequences. IEEE Trans. Image Process. 25, 1765–1778. https://doi.org/10.1109/TIP.2016.2529506.

Vasa, L., Skala, V., 2011. A perception correlated comparison method for dynamic meshes. IEEE Trans. Vis. Comput. Graph. 17, 220–230. https://doi.org/10.1109/TVCG.2010.38.

Vlasic, D., Baran, I., Matusik, W., Popović, J., 2008. Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph. 27, 97.

Wang, P.S., Fu, X.M., Liu, Y., Tong, X., Liu, S.L., Guo, B., 2015. Rolling guidance normal filter for geometric processing. ACM Trans. Graph. 34, 173:1–173:9. https://doi.org/10.1145/2816795.2818068.

Wang, P.S., Liu, Y., Tong, X., 2016. Mesh denoising via cascaded normal regression. ACM Trans. Graph. 35, 232:1–232:12. https://doi.org/10.1145/2980179.2980232.

Yadav, S.K., Reitebuch, U., Polthier, K., 2017. Mesh denoising based on normal voting tensor and binary optimization. IEEE Trans. Vis. Comput. Graph., 1. https://doi.org/10.1109/TVCG.2017.2740384.

Yang, B., Jiang, Z., Shangguan, J., Li, F.W.B., Song, C., Guo, Y., Xu, M., 2018. Compressed dynamic mesh sequence for progressive streaming. Comput. Animat. Virtual Worlds, e1847. https://onlinelibrary.wiley.com/doi/abs/10.1002/cav.1847. https://onlinelibrary.wiley.com/doi/pdf/10.1002/cav.1847. e1847 cav.1847.

Yang, J.H., Kim, C.S., Lee, S.U., 2005. Progressive coding of 3d dynamic mesh sequences using spatiotemporal decomposition. In: 2005 IEEE International Symposium on Circuits and Systems, vol. 2, pp. 944–947.

Yang, J.H., Kim, C.S., Lee, S.U., 2006. Semi-regular representation and progressive compression of 3-d dynamic mesh sequences. IEEE Trans. Image Process. 15, 2531–2544. https://doi.org/10.1109/TIP.2006.877413.

Zha, H., Simon, H., 1999. On updating problems in latent semantic indexing. SIAM J. Sci. Comput. 21, 782–791. https://doi.org/10.1137/S1064827597329266.
Zhang, H., Xu, F., 2018. Mixedfusion: Real-time reconstruction of an indoor scene with dynamic objects. IEEE Trans. Vis. Comput. Graph. 24, 3137–3146. https://doi.org/10.1109/TVCG.2017.2786233.
Zhang, J., Owen, C.B., 2004. Octree-based animated geometry compression. In: Data Compression Conference, 2004. Proceedings. DCC 2004, pp. 508–517.
Zhang, J., Owen, C.B., 2005. Hybrid coding for animated polygonal meshes: combining delta and octree. In: International Conference on Information Technology: Coding and Computing (ITCC'05) - Volume II, vol. 1, pp. 68–73.
Zhang, S., Zhao, J., Wang, B., 2010. A local feature based simplification method for animated mesh sequence. In: 2010 2nd International Conference on Computer Engineering and Technology. V1–681–V1–685.
Zhang, W., Deng, B., Zhang, J., Bouaziz, S., Liu, L., 2015. Guided mesh normal filtering. Pacif. Graph. 34.