Embedded Deformation-based Compression for Human 3D Dynamic Meshes with Changing Topology

Huong Hoang[1], Kunyao Chen[1,2], Truong Nguyen[1], and Pamela Cosman[1]

[1]University of California, San Diego, CA, USA
[2]Qualcomm Institute, CA, USA
{hlhoang, kuc017, tqn001, pcosman}@ucsd.edu

Abstract

3D dynamic meshes offer significant potential in various applications, but their usage is still limited by their large file size. We present a novel method that can compress 3D human dynamic meshes effectively by using embedded deformation to extract the underlying transformations of consecutive frames. We target 3D dynamic meshes with changing connectivity which are more versatile compared to traditional mesh animation but also more challenging. To further reduce the transmission size, we propose a novel optimization-based technique to determine a sparse set of key nodes capable of transmitting the transformations efficiently.

1. Introduction

The Internet is undergoing a revolutionary change. While the current Internet carries text, audio, images, and video, the next generation of the Internet will focus on 3D-based scenes and models. Static 3D objects offer rich viewing experience while dynamic 3D models improve the immersive 3D experience with temporal information. The flexibility and potential of 3D dynamic meshes have spurred further exploration of their capabilities, making them a key area of interest in the field of computer graphics and related domains.

Recently, several approaches have been proposed to compress 3D mesh animations effectively [30], [39]. However, their methods focus on synthesized mesh sequences where the topology of different frames is fixed. Effective compression for 3D dynamic meshes with changing connectivity remains a challenging task. While it is possible to register them to an approximate mesh animation, the registration process can be time-consuming, making it unsuitable for real-time applications. Moreover, mesh registration is particularly challenging for models with intricate content such as loose clothing and hairstyles, as finding a matching template is often difficult and may require designing a special embodiment. This presents a significant obstacle to the effective use of scanned models in real-time scenarios, highlighting the need for new approaches to compress such sequences.

Although 3D dynamic meshes with changing connectivity are more general and versatile, the complexities of processing them pose significant challenges compared to traditional mesh animation. They do not have explicit vertex correspondence between frames, and usually have different numbers of vertices as well as varying connectivity over time. In this paper, we address this challenge by utilizing the deformable transformations of consecutive frames without requiring explicit vertex correspondence. The main contributions of this paper are:

- We present a novel compression method for 3D dynamic meshes with deformation transformations as the motion estimation. We show that the alignment information can effectively compress 3D dynamic meshes whose connectivity can be either fixed or varying over time.
- We propose an optimization-based approach to obtain an optimally sparse layout of key nodes for the underlying embedded deformation approach. The acquired key node set is both sparse and sufficient to transmit prediction information for a high quality predictive frame.
- We show that our optimization approach is beneficial not only to the compression, but also to the deformation in terms of convergence.

2. Related Work

This section reviews current research on Human 3D Dynamic Meshes Compression. We start by exploring existing research on 3D mesh animation compression for fixed connectivity. Then, we adapt to compression techniques for changing connectivity dynamic meshes. Lastly, we examine 3D human motion compression, which takes into account human perceptual characteristics.

2.1. 3D Dynamic Mesh Sequence Compression

Current compression techniques can be divided by their handling either fixed or varying connectivity, where fixed connectivity refers to having the same topology over frames, and is common for computer-designed animations.

Fixed connectivity. Proposed approaches can be separated into the following categories.

Prediction-based methods: Early works encoded differences in vertex position over frames [13], improved with second-order code prediction [7] or the use of ICP to match points and transmit motion parameters and error residues [21]. Various scalable schemes [46, 9, 3] involved predictively exploiting the already encoded spatio-temporal neighborhood using rotation-invariant coordinates, weighted spatial prediction, and iteratively decimating vertices. Observing that within a mesh sequence, the model's shape changes smoothly while the fine-scale details remain the same, [24, 11] represent the animated 3D model as the deformation of low frequencies while preserving high-frequency details. Recently, [25] used non-linear transformations to describe deformations of patches with similar motion patterns through the sequence, while [6] uses Graph Fourier Transforms to project the difference between a vertex and its neighbor vertices onto eigentrajectories.

PCA-based methods: PCA was proposed in [4] to represent dynamic mesh sequences, decomposing a matrix containing the geometry of all key frames in the original sequence into eigenvectors and eigenvalues. This allows progressive transmission because most of the information can be found in a linear combination of eigenvectors corresponding to small eigenvalues. This was improved with second-order prediction in [29] and a compression algorithm for the PCA basis in [48]. In [37], frames were aggregated into clusters with similar poses which were decomposed using PCA. The computational cost was addressed in [31], and a robust PCA method was presented in [30].

Segmentation-based methods: In [45], Lloyd's algorithm combined with PCA segmented a mesh into groups of vertex trajectories that move almost independently over frames. Complexity was reduced in [28], while [5] combined local coordinates and PCA. In [23], a segmentation scheme partitioned the first mesh of the sequence into sub meshes having independent deformations. The best affine transformations of submeshes were then computed and encoded. Spatial and temporal mesh clustering based on curvature and torsion was discussed in [51], while a greedy mesh segmentation was presented in [39] and [38].

Wavelet-based methods: Guskov et al. in [22] transformed an input mesh with an isotropic wavelet transform, and progressively encoded the resulting wavelet details using coefficient differences, while temporal wavelet transforms were used in [44, 10] to exploit the temporal coherence of the geometry component.

MPEG Framework: A skinning model was used in [41] in which vertices were partitioned into clusters and predicted from their individual weights combined with their cluster's transformation, and a set of flexible transform modules and layer predictors supported different compression functionalities.

Varying connectivity. 3D mesh sequences reconstructed from cameras will experience varying connectivity, but these have seen less research. In [52], an approximate global topology for the whole mesh sequence was proposed to deal with changing connectivity; the global topology is constructed by re-meshing the first frame and mapping it to the following frames using motion estimation. [15] introduced a similar idea but employs separate Groups of Frames (GoF), with each GoF having its own global topology. Extending 2D video block matching, [26] divided a mesh into cubic blocks which search for the best match block in the reference mesh frame, while a patch-based matching algorithm was used in [50]. Recently, [19] used the new Visual Volumetric Video-based Coding (V3C) standard to encode meshes by using orthogonal projections, followed by atlas packing and video coding. By sending the connectivity patch for every mesh frame, this method is able to deal with sequences with varying connectivity. A newly proposed method in [42] focuses on encoding the texture information in a dynamic mesh, as it encodes mesh geometry and topology for each frame separately. The paper presented a patch-based approach that reorganizes the intra-frame and inter-frame texture tiles to improve spatial and temporal correlations.

2.2. 3D Human Motion Compression

While a single 3D human model could be compressed by any static mesh compressor, many research groups actively exploit the temporal redundancies in 3D human sequences, using human skinning methods and human movement relationships. They can be separated into three main categories of representations: Skeleton, Markers and Mesh.

Skeleton representation: This direction simplifies human models to skeletons consisting of bones and joints. Time-varying joint trajectories were compressed in [2] by discrete wavelet transforms, combined in [35] with kinematics and key frames. Wavelet decompositions on joint rotation angles were approximated in [32], and a related

approach [8] optimized wavelet coefficient selection combined with inverse kinematics to adapt standard truncated wavelet compression techniques to the nature of skeletal animation data. Other schemes employing forward or inverse kinematics [34, 33, 32] sample joint trajectories into discrete patterns [18], fit cubic Bezier curves to bone trajectories [53], or use a tree hierarchy [27] to express the relationships of movements of body parts.

Marker representation: A 3D model can be represented as a set of 3D markers, with 40-50 markers at the minimum for a human. To compress marker movement, [36] segmented a motion sequence into subsequences such that poses within a subsequence lie near a low dimensional linear space; a segment is then compressed using PCA. Organizing 3D human markers into body parts, motion markers for one part were combined and represented by a host node in [20]. Each node is partitioned into motion segments which were represented by an entropy-coded index to the motion pattern database.

Mesh representation: Treating human models as meshes, [43] converted a mesh sequence into skinned rig models consisting of bones and a skin attached to the bones by weights. The bones undergo separate rigid transformations to animate the skin, whose geometry and color are compressed by Graph Wavelet Filter Banks. Using advances in human skeleton tracking combined with a frame-based compression method, skeleton motion information extracted from depth maps was used to represent the predictive frames in [17, 16], together with a connectivity compression algorithm.

3. Key frame-based dynamic mesh codec

In this section, we present our proposed codec specifically designed for time-varying dynamic mesh sequences. These sequences can have varying connectivity and changing numbers of vertices across frames, making them challenging to process efficiently. To overcome this challenge, our system divides the sequence into multiple GoFs and sends transformation information for predictive frames. This approach significantly reduces the amount of data that needs to be transmitted for a predictive frame, allowing for more efficient processing and storage of the sequence. In addition, our codec does not require the complete sequence to be available before processing, enabling real-time performance even for long sequences.

3.1. Encoder

3.1.1 Overview of the Embedded Deformation-based Encoding approach

Let $S = \{M_1, M_2, ..., M_F\}$ be the time-varying mesh sequence to be compressed, with F as the total number of frames in the sequence. Let M_t be a static mesh at time t containing V_t vertices, and static meshes in the sequence S might contain different topologies.

Our primary approach involves predicting mesh frames using transformation information from a set of key nodes that control the surface deformation of the mesh. These key nodes can be used to geometrically control the deformation of the entire mesh, allowing us to accurately predict future frames from a previous one.

Let $\mathbf{n} = \{n_j \in \mathbb{R}^3\} \in \mathbb{R}^{N\times 3}$ be the set of key nodes to control the deformation of the local area, and let $\mathbf{R} = \{R_j \in \mathbb{R}^3\} \in \mathbb{R}^{N\times 3}$ and $\mathbf{T} = \{t_j \in \mathbb{R}^3\} \in \mathbb{R}^{N\times 3}$ be the affine transformation information of each key node. Here, $j \in [1, N]$ where $N \ll V$ is the total number of key nodes controlling the deformation of the mesh. Let $\mathcal{GT}(.)$ be a geometric transformation operator which generates a geometric approximation M'_t of M_t so that M'_t has the same topology as M_{t-1}:

$$M'_t = \mathcal{GT}(M_{t-1}, \mathbf{n}, \mathbf{R}, \mathbf{T}) \tag{1}$$

Each vertex x_i in M_{t-1} , $i \in [0, V_{t-1} - 1]$ is deformed according to its Q neighboring key nodes:

$$x'_i = \mathcal{GT}(x_i) = \sum_{j=0}^{Q-1} w_{ij}(R_j(x_i - n_j) + t_j + n_j) \tag{2}$$

where n_j, R_j, t_j are the position, rotation and translation of key node j, respectively. w_{ij} is the influence weight of key node n_j for vertex x_i based on the Euclidean distance between them.

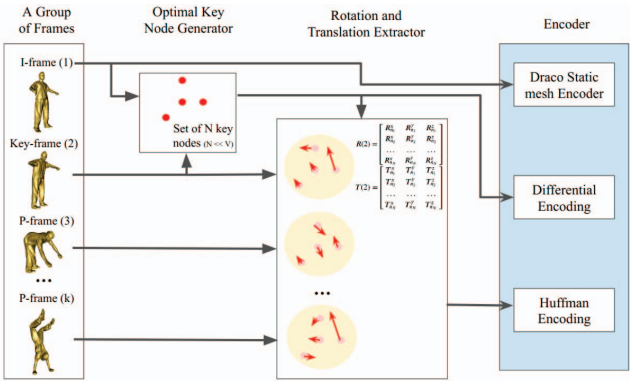

Figure 1. Encoder structure for a Group of Frames with ***k*** frames, where V is the average number of vertices in the sequence.

Figure 1 presents the block diagram of our proposed encoder. Encoding begins by selecting the first frame of a given GoF as the I-frame, typically encoded at high quality using a static mesh encoder. We use Draco [1] for I-frame coding. Then, we choose another frame in the GoF as the key frame, used to generate a set of N key nodes **n** along with the I-frame. It is worth noting that not every GoF requires a key frame, as the set of key nodes **n** can be reused

for other GoFs in the sequence if needed. The remaining frames in the GoF are referred to as P-frames, which are transmitted using prediction information **n**, **R** and **T**.

We predict our P-frames using the Embedded Deformation strategy [12] [47], leveraging the relations outlined in equations 1 and 2. This approach utilizes the set of key nodes **n**, as well as the transformation information **R** and **T**, to accurately predict the deformation of the mesh and generate high-quality P-frames.

3.1.2 Rotation and Translation Extractor

Assuming that the set of key nodes **n** is known to the encoder, the *'Rotation and Translation Extractor'* is responsible for finding the optimal set of **R** and **T** (represented as red vectors in Figure 1) corresponding to the set of key nodes **n**, such that the predicted mesh is as close as possible to the source mesh. The rotation and translation vectors are each of length three.

Let $\mathcal{D}(.)$ be the solver adopted from the state-of-the-art mesh deformation method in [12] to obtain the optimal set of **R, T** defined through an objective function below.

$$\mathcal{D}(M_{t-1}, \mathbf{n}) = \underset{\mathbf{R},\,\mathbf{T}}{\operatorname{argmin}}(\mathcal{L}_{data} + \alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot}) \quad (3)$$

Here, $\mathcal{L}_{data}$ penalizes the geometry deviation between source and target; $\mathcal{L}_{rot}$ measures the deviation of Rotation R from the orthogonal matrix; and $\mathcal{L}_{reg}$ ensures the smoothness of the deformation.

Several embedded deformation approaches have been proposed, and most of them could be used in our proposed system to obtain **R** and **T** depending on the particular application. In our presented codec, we have chosen the method in [12] since it has the ability to deal with intricate human clothing by integrating isometric deformation into embedded deformation.

3.1.3 Optimal Key Node Generator

The *'Optimal Key Node Generator'* stage of the codec plays a critical role in determining the quality and bitrate for a predictive frame. Lowering the number of key nodes may reduce the bitrate as well as the reconstructed mesh quality. In Section 4, we propose an efficient solution to the optimization between compactness and quality. Our method leverages a set of sophisticated algorithms to find the optimal set of key nodes **n** that satisfies the target sparsity while minimizing the loss of information.

3.1.4 Encoding of the information

We have three main encoding blocks: a static mesh encoder, a differential encoder, and a Huffman encoder. We use the state-of-the-art static mesh encoder Draco [1] to encode I-frames. For any P-frame M_t, which is predicted using the global set of key nodes **n** and its own transformation set **R**, **T**, we map all the key nodes $n_j \in \mathbf{n}$ to the previous frame M'_{t-1} to identify the vertices' index corresponding to the key nodes, where M'_{t-1} is the reconstructed frame $t-1$ known by the decoder. Once the indices are obtained, the key node set **n** can be encoded efficiently by applying differential encoding on the list of indices.

The transformation values are calculated with an implicit quantization that comes from being stored as 4-byte floating point numbers. We do not apply any additional quantization, rather we focus on controlling the rate-distortion trade-off by adjusting the number of key nodes. We use Huffman coding for the sets of transformations **R**, **T** with separate Huffman codes for the R and T values. We use a two-pass technique in which the encoder goes through the entire sequence to collect statistics on the R and T values, creates two Huffman codes, and then encodes the sequence with those codes.

3.2. Decoder

The decoder decompresses the I-frames using the Draco decoder, and the predicted information **n**, **R**, **T** are extracted with Differential and Huffman decoding. Fig. 2 shows the decoder pipeline. After collecting **n**, **R**, **T** for a P-frame M_t, its distorted mesh M'_t can be reconstructed through deformation from M'_{t-1}, which could be either an I-frame, a key frame or a P-frame, using Eq. 2.

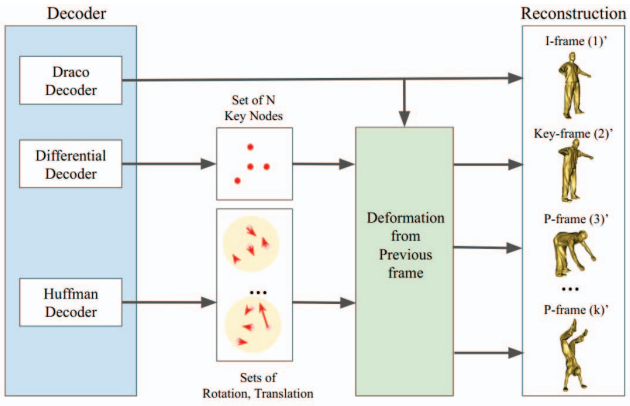

Figure 2. Decoder structure

4. Choosing Key Nodes: Alternating Direction Method of Multipliers (ADMM)

When compressing predictive frames, achieving a balance between compactness and quality is a major goal. The size of the compressed P-frame depends strongly on the number of key nodes used, while the frame's quality depends on both the number and positions of the key nodes.

One needs only a few key nodes in body parts that have limited movements (such as head and torso), whereas more nodes are needed to describe complex motions and intricate content (such as arms, legs and hands). In this section, we propose an efficient method to identify the optimal number of key nodes and their placement to minimize the number of nodes required while ensuring high-quality reconstruction.

4.1. Problem Formulation

Let $\mathbf{P} = \{\mathbf{R}, \mathbf{t}\}$ be the set of rotations and translations of corresponding key nodes in $\mathbf{n}$ for a certain P-frame. Typically, embedded mesh deformation approaches select the set of key nodes $\mathbf{n}$ such that the quality of M'_t is maximized. [12] gathered $\mathbf{n}$ by uniformly sampling over the surface of M_{t-1} for a very dense set of key nodes. However, from a compression perspective, it is desirable to minimize the number of key nodes while simultaneously enhancing the quality of deformation. To this end, we define an alternative objective function and its solver as:

$$\mathcal{D}^*(M_{t-1}) = \underset{\mathbf{n},\mathbf{P}}{\operatorname{argmin}}(\mathcal{L}_{de} + \lambda\|\mathbf{n}\|_0) \tag{4}$$

where $\mathcal{L}_{de} = \mathcal{L}_{data} + \alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot}$ and $\|\mathbf{n}\|_0$ is the l_0 *"pseudo-norm"* which is the number of non-zero elements in the vector, to emphasize sparsity. Note that zero values can be mistaken with the positions, so it is necessary to manipulate vertices' positions to be above zero for this objective function. Moreover, with the later proposed ADMM scheme, it is not required to do so.

Instead of sampling over the surface of M_{t-1} to obtain a set of key nodes, $\mathcal{D}^*$ serves as a solver to find the optimal set of both $\mathbf{n}$ and $\mathbf{P}$ for the compression task. We rewrite the objective function for $\mathcal{D}^*(M_{t-1})$ so that it can be solved with ADMM as:

$$\mathcal{D}^*(M_{t-1}) = \underset{\mathbf{n},\mathbf{P}}{\operatorname{argmin}}(\alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot} + \lambda\|\mathbf{n}\|_0) \tag{5}$$

subject to $M'_t = M_t$. The augmented Lagrangian, for a parameter $\rho > 0$, is defined as:

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{n}, \mathbf{P}, \mathbf{u}) = \alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot} + \lambda\|\mathbf{n}\|_0 + \\ \rho\mathbf{u}^T(M'_t - M_t) + \frac{\rho}{2}\|M'_t - M_t\|^2_{\mathcal{F}}\end{aligned} \tag{6}$$

Since M_t is a matrix with size $V_t \times 3$, computing $\mathbf{u}^T(M'_t - M_t)$ involves matrix computations for the whole mesh making it not suitable for a parallel computing scheme. Therefore, we rewrite the augmented Lagrangian in Eq. 6 into the scaled augmented Lagrangian as:

$$\begin{aligned}\mathcal{L}_\rho(\mathbf{n}, \mathbf{P}, \mathbf{u}) = \alpha_{reg}\mathcal{L}_{reg} + \alpha_{rot}\mathcal{L}_{rot} + \lambda\|\mathbf{n}\|_0 \\ + \frac{\rho}{2}\|M'_t - M_t + \mathbf{u}\|^2_{\mathcal{F}} - \frac{\rho}{2}\|\mathbf{u}\|^2_{\mathcal{F}}\end{aligned} \tag{7}$$

4.2. ADMM Steps

With the scaled augmented Lagrangian derived in Eq. 7, the ADMM repeats for $k = 1, 2, 3...$

$$\mathbf{P}^{(k)} = \underset{\mathbf{P}}{\operatorname{argmin}}(\mathcal{L}_\rho(\mathbf{n}^{(k-1)}, \mathbf{P}, \mathbf{u}^{(k-1)})) \tag{8}$$

$$\mathbf{n}^{(k)} = \underset{\mathbf{n}}{\operatorname{argmin}}(\mathcal{L}_\rho(\mathbf{n}, \mathbf{P}^{(k-1)}, \mathbf{u}^{(k-1)})) \tag{9}$$

$$\mathbf{u}^{(k)} = \mathbf{u}^{(k-1)} + \rho(M_t'^{(k)} - M_t) \tag{10}$$

For the initialization step ($k = 0$), we define the initial values for the parameters as:

- $\mathbf{n}^{(0)}$: uniformly sampled over the surface of M_{t-1} with a given number of key nodes
- $\mathbf{P}^{(0)} = \mathcal{D}(M_{t-1}, \mathbf{n}^{(0)})$
- $\mathbf{u}^{(0)} = \mathbf{0} \in \mathbb{R}^{(V_t \times 3)}$

This proposed scheme forces the ADMM to start with a large number of key nodes for the initial value of $\mathbf{n}^{(0)}$. The next iterations in ADMM repeatedly remove less important key nodes, to achieve a final $\mathbf{n}^{(K)}$ that is both sparse and capable of constructing a good approximation of M_t.

To solve the sub-problems in Eq. 8 and 9, we propose a three-step mechanism for each iteration of ADMM, summarized in the flowchart of Fig. 3. The normal mesh deformation solver $\mathcal{D}$ solves Eq. 8 while the combination of coordinate descent and gradient descent algorithms is targeted to solve Eq. 9. While the input of the deformation solver $\mathcal{D}$ and the coordinate descent algorithm comes from the previous iteration of ADMM, the input of the gradient descent block is the direct output of the coordinate descent block within the same iteration. All blocks will be described in detail in the subsections below.

4.2.1 $\mathcal{D}$ as the solver to obtain $\mathbf{P}^{(k)}$

Eq. 8 can be solved by the deformation operator $\mathcal{D}$ defined in Eq. 3 as follows:

$$\mathbf{P}^{(k)} = \mathcal{D}(M_{t-1}, \mathbf{n}^{(k-1)}) \tag{11}$$

where $\mathcal{L}_{data}$ needs to be modified to be consistent with the scaled augmented Lagrangian:

$$\mathcal{L}_{data} = \frac{\rho}{2}\|M'_t - M_t + \mathbf{u}^{(k-1)}\|^2_{\mathcal{F}} \tag{12}$$

where $\mathbf{u}^{(k-1)}$ is the running sum of the residuals, defined as:

$$\mathbf{u}^{(k-1)} = \mathbf{u}^{(0)} + \sum_{i=1}^{k-1}(M_t'^{(i)} - M_t). \tag{13}$$

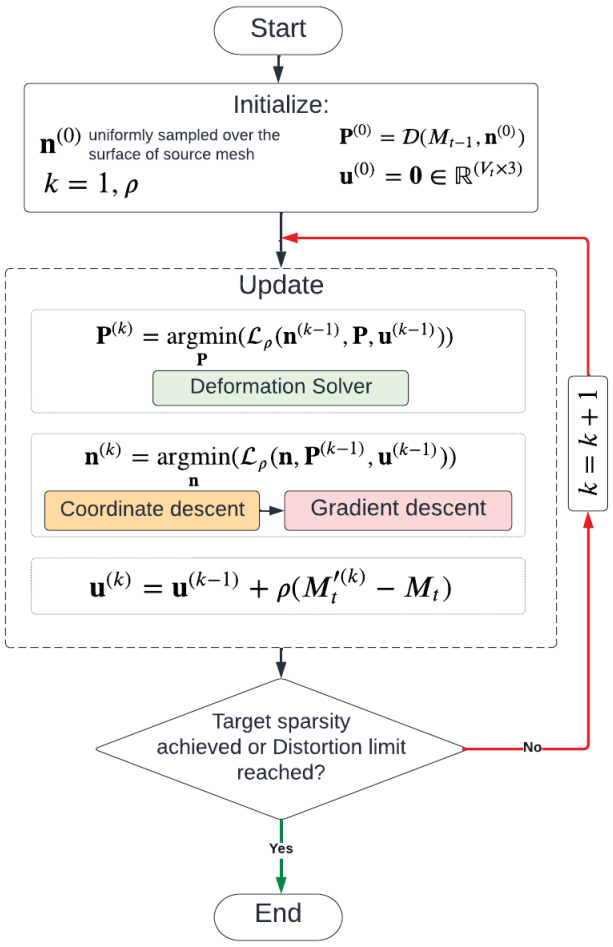

Figure 3. ADMM Flowchart

4.2.2 Key Node Sparsification by Coordinate Descent

As we want to emphasize sparsity, the l_0 norm problem can be solved by iteratively setting a coefficient of a key node to zero (i.e., removing that key node) one at a time while holding the other key nodes fixed using Coordinate Descent. For each candidate removal, the cost function based on Eq. 9 will be calculated; the key node with the lowest cost will be removed. The algorithm for this coordinate descent scheme is demonstrated in Algorithm 1.

As it requires only a single pass through the data at each iteration, coordinate descent can be very efficient for large datasets, especially meshes whose number of vertices can be in the thousands. It can also be parallelized as the removal of each key node can be evaluated independently.

To further accelerate the process, we design a reduced geometric transformation operator for this coordinate descent algorithm. While the original geometric transformation operator requires deformation for every vertex in the mesh M_{t-1}, we save the deformed results in a cache memory and perform re-deformation only on vertices affected by the candidate removal. The reduced operator is illustrated in Algorithm 2.

Algorithm 1 Key Node Set Sparsification

$\mathbf{n} \leftarrow$ Set of given key nodes
$M_{t-1} \leftarrow$ source mesh
$M_t \leftarrow$ target mesh
$e_{min} \leftarrow \infty$
for $n_j \in \mathbf{n}$ **do**
  $\mathcal{S} \leftarrow \mathbf{n} \backslash n_j$
  $\widetilde{\mathbf{R}}, \widetilde{\mathbf{T}} \leftarrow \mathcal{D}(M_{t-1}, \mathcal{S})$ ▷ $\mathcal{D}$ follows Eq.(3)
  $M_t' \leftarrow \mathcal{GT}(M_{t-1}, \mathcal{S}, \widetilde{\mathbf{R}}, \widetilde{\mathbf{T}})$ ▷ $\mathcal{GT}$ follows Eq.(1)
  $e_j \leftarrow \|M_t - M_t'\|_{\mathcal{F}}^2$
  if $e_j < e_{min}$ **then**
    $e_{min} \leftarrow e_j$
    $n^* \leftarrow n_j$
  end if
end for
$\mathbf{n} \leftarrow \mathbf{n} \backslash n^*$ ▷ Remove node n^*

Algorithm 2 Reduced Geometric Transformation Operator

$M_{t-1} \leftarrow$ source mesh
$\widetilde{M}_t \leftarrow$ Deformed mesh before removal
$n^* \leftarrow$ to-be-removed node
for $v_i \in M_{t-1}$ **do**
  if $v_j \in n^*$.control_area() **then**
    $v_i' \leftarrow \mathcal{GT}(v_i)$ ▷ $\mathcal{GT}$ follows Eq.(2)
  else
    $v_i' \leftarrow \tilde{v}_i$ ▷ $\tilde{v}_i \in \widetilde{M}_t$
  end if
end for
$M_t' \leftarrow \{v_1', v_2', ..., v_{V_{t-1}}'\}$

The Key Node Set Sparsification process is repeated until the cost function is less than the l_0 norm penalty or the maximum allowable distortion is reached.

4.2.3 Key node relocation through Gradient Descent

As node removal is expected to introduce some additional distortion on the deformed mesh M_t', the key node relocation algorithm aims to reduce the cost function by iteratively updating the positions of key nodes in the direction of the negative gradient of the objective function until a local minimum is reached. Let the cost function for this gradient descent be defined as $\mathcal{L}_{gd} = \|M_t - M_t'\|_{\mathcal{F}}^2$.

For simplicity, we do not include $\mathcal{L}_{reg}$ and $\mathcal{L}_{rot}$ in this loss function. The gradient of the cost function can be found by applying the chain rule to the derivatives of the cost function with respect to the set of key nodes $\mathbf{n}$:

$$\nabla_{\mathbf{n}}\mathcal{L}_{gd} = \frac{\partial \mathcal{L}_{gd}}{\partial \mathbf{n}} = \frac{\partial \mathcal{L}_{gd}}{\partial M'_t} \times \frac{\partial M'_t}{\partial \mathbf{n}} \tag{14}$$

$$\frac{\partial \mathcal{L}_{gd}}{\partial M'_t} = 2 \times (M'_t - M_t) \tag{15}$$

$$\frac{\partial M'_t}{\partial \mathbf{n}} = \{\frac{\partial M'_t}{\partial n_j}\} = \{\sum_{i=1}^{L} w_{ij}(\mathbf{1} - R_j)\} \tag{16}$$

where L is the number of vertices belonging to the region controlled by node n_j. In Eq. 16, $\frac{\partial M'_t}{\partial \mathbf{n}}$ is a list containing gradients of every key node in $\mathbf{n}$. The individual gradient $\frac{\partial M'_t}{\partial n_j}$ of key node n_j is calculated based on the derivative of the loss between x_i and x'_i if x_i falls inside the area controlled by n_j.

The key nodes then move along the negative gradient of the objective function: $\mathbf{n} = \mathbf{n} - \sigma \times \nabla_{\mathbf{n}}\mathcal{L}_{gd}$, where σ as the learning rate. As the geometric transformation operator involves determining the area controlled by the key nodes, the gradient descent algorithm should move key nodes within a limited region to ensure their lists of controlled vertices are not significantly altered. Therefore, a small learning rate σ is suitable for this purpose.

5. Experimental Results

5.1. Optimal Key Node Generator Performance

In our system, the compression rate is primarily controlled through the number of key nodes for P-frames. The *'Optimal Key Node Generator'* identifies the optimal number of nodes and their positions in 3D to achieve a given target quality. Naturally, a smaller number of key nodes reduces transmission size but also leads to a decrease in quality. In this section, we visually illustrate the trade-off between rate and distortion when using a limited number of key nodes.

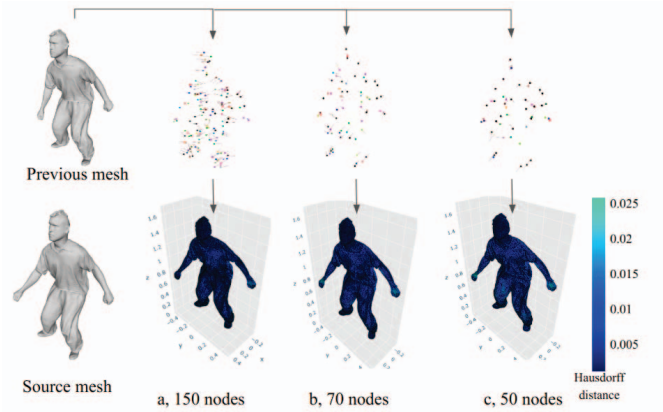

Figure 4. Frame 21 of 'bouncing', originally with 10,000 vertices, reconstructed from 150, 70, and 50 key nodes.

Fig. 4 shows the reconstructed meshes of frame 21 in the MIT dataset [49]'s 'bouncing' sequence using 150, 70 and 50 key nodes.

Providing the codec with more nodes leads to improved reconstruction outcomes; the deformation can be more flexible when there are more key points controlling the surface. However, from a compression standpoint, fewer nodes can lead to lower bitrate, with some sacrifice in precision.

5.2. Predictive Frame Compactness Evaluation

This section focuses on evaluating predictive frame compactness in our compression system by comparing the rate-distortion curves for different GoF sizes. This comparison will also offer a valuable guide to selecting the optimal GoF size that aligns with different preferences. The metric for evaluating the size of a sequence is: $r = \frac{B}{\sum_{t=1}^{F} V_t}$, where B is the total size in bits after compression, to represent the bit size per vertex per frame (bpvf).

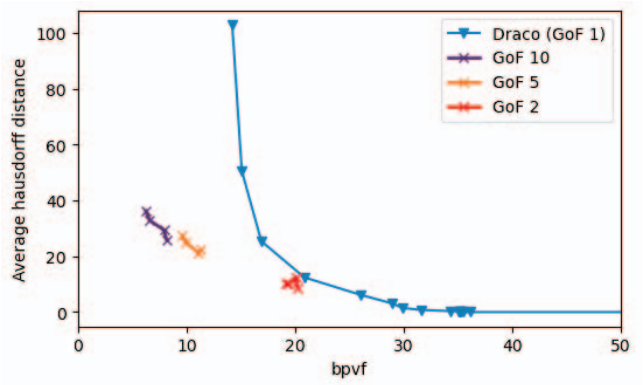

Figure 5. Longdress sequence frames #1071-#1091 encoded with different GoF sizes. For the lines indicating GoF sizes larger than 1, different markers represent varying numbers of key nodes used for compression.

Figure 5 shows a comparison among various sizes of GoF for the Longdress sequence from frame #1071 to #1941 with an average number of vertices of 22,307. This sequence is encoded using a single set of key nodes. The figure illustrates a clear trend that improves compression performance with larger GoF size. Specifically, as the GoF size increases, the overall bitrate substantially decreases for the same output distortion. These results emphasize the benefits of using our Embedded Deformation-based P-frames for efficient video compression under resource constraints.

5.3. Impact of generated key nodes on deformation

Embedded deformation approaches rely on the underlying set of key nodes to control the deformation. The choice of the layout of those key nodes determines the flexibility of the deformation and directly affects the final alignment re-

sults. Typically, key nodes are gathered by sampling on the source surface [12]. The authors of [47] stated that mesh simplification algorithms can also produce good alignment results. In this section, we compare the performance of mesh deformation using the optimal key nodes obtained from our ADMM mechanism with traditional methods: uniform sampling on the surface, and mesh simplification.

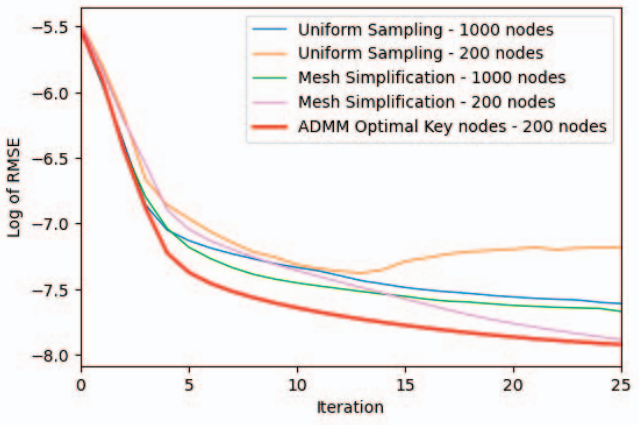

Figure 6. Average bi-directional RMSE of using different layouts for Mesh Deformation approach.

Figure 6 demonstrates the performance of the state-of-the-art mesh deformation method [12] given different layouts for the key nodes. It can be observed that with our optimal key nodes, the deformation method converges faster than other traditional methods of sampling key nodes, even with a sparser structure.

5.4. Accumulated Distortion of P-frames

Our codec employs a predictive approach to generate each P-frame from its previous frame. That previous frame might itself be a P-frame, when the size of a GoF is larger than two. Compression is typically applied to P-frames by reducing the number of key nodes, resulting in some degree of distortion. Figure 7 illustrates how the distortion in P-frames accumulates over time.

To address this, for each P-frame it will be important to encode the prediction errors that exceed some distortion threshold. We plan to add a mechanism for encoding prediction residuals in our future work.

5.5. Time-varying mesh sequence compression performance: Comparison with baselines

In this section, we compare our method with [17] (called Skeleton-based) and [40] (called TFAN), with the reference results collected directly from [17]. We evaluate our method on 1600 frames of the Dimitrios sequence with an average vertex count per frame of 35,100 [17]. There are four sets of key nodes, generated with different sparsity levels, influencing the compression of this sequence. We use the surface

Figure 7. Accumulated distortion on the consecutive P-frames of a GoF of 5. Samba sequence frame 21 to 25.

error metric RMSE used in [17]:

$$RMSE = \frac{1}{M} max\Big(\sum_{t=1}^{F} e(M_t, M_t'), \sum_{t=1}^{F} e(M_t', M_t)\Big) \quad (17)$$

where $e(M_t, M_t')$ denotes the directed root mean squared distance between M_t and M_t', obtained from METRO [14].

The comparison in Figure 8 shows that our proposed approach produces less distortion at low bitrates (below 8 bpvf) thanks to the ADMM scheme identifying sparse key node sets that effectively reconstruct high-quality predictive frames. However, our method does not demonstrate a significant reduction in distortion as the bitrate increases. This is primarily due to the absence of encoding the prediction residuals, which is crucial for accurate reconstruction.

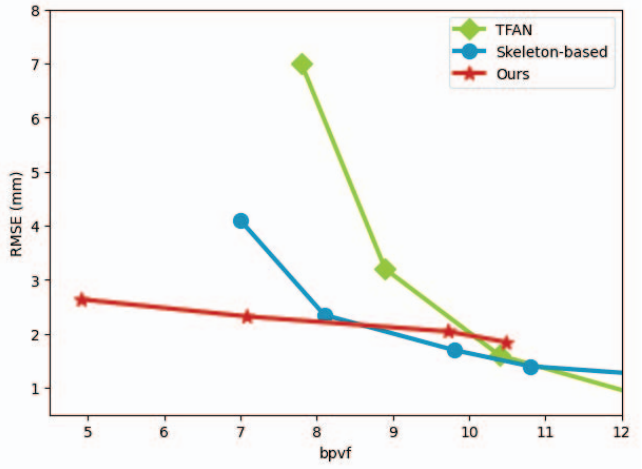

Figure 8. Surface error distortion RMSE versus bitrate for the Dimitrios sequence for frames #400 - #2000.

6. Conclusions and Future Work

In this article, we present a novel compression approach to compress time-varying human mesh sequences by exploiting transformations based on embedded deformations.

Our experimental results show that this type of information has the potential to effectively compress mesh sequences with changing connectivity. In addition, we proposed a mechanism to find sparse key node sets, allowing for manipulation of the compressed bitrate while reconstructing high-quality predictive frames.

The proposed system enables real-time processing, interactive virtual reality experiences, and efficient 3D content streaming, particularly for models with inconsistent topologies such as scanned models. Furthermore, the *'Optimal Key Node Generator'* may provide valuable insights into regions undergoing prominent changes between frames. These insights could be beneficial for gesture analysis, motion tracking, feature extraction, and other related tasks.

A limitation of our proposed codec is the absence of prediction error encoding. In our future work, we intend to design an efficient mechanism to encode prediction errors specific to embedded deformation-based transformations, allowing additional control over the distortion-rate trade-offs for the reconstructed frames.

References

[1] Draco: 3D Data Compression. https://github.com/google/draco.

[2] Amr Adel Hassan Ahmed, Adrian Hilton, and Farzin Mokhtarian. Adaptive Compression of Human Animation Data. In *Eurographics 2002 - Short Presentations*. Eurographics Association, 2002.

[3] Jae-Kyun Ahn, Yeong Jun Koh, and Chang-Su Kim. Efficient Fine-Granular Scalable Coding of 3D Mesh Sequences. *IEEE Transactions on Multimedia*, 15(3):485–497, 2013.

[4] Marc Alexa and Wolfgang Muller. Representing Animations by Principal Components. *Computer Graphics Forum*, 2000.

[5] Rachida Amjoun, Ralf Sondershaus, and Wolfgang Straßer. Compression of complex animated meshes. In *Advances in Computer Graphics*, pages 606–613. Springer Berlin Heidelberg, 2006.

[6] Gerasimos Arvanitis, Aris S. Lalos, and Konstantinos Moustakas. Fast Spatio-temporal Compression of Dynamic 3D Meshes, 2021.

[7] Chandrajit L Bajaj, Valerio Pascucci, and Guozhong Zhuang. Single resolution compression of arbitrary triangular meshes with properties. DCC '99, page 247, USA, 1999. IEEE Computer Society.

[8] Philippe Beaudoin, Pierre Poulin, and Michiel van de Panne. Adapting Wavelet Compression to Human Motion Capture Clips. In *Proceedings of Graphics Interface 2007*, GI '07, page 313–318, New York, NY, USA, 2007. Association for Computing Machinery.

[9] M. Oguz Bici and Gozde B. Akar. Improved prediction methods for scalable predictive animated mesh compression. *Journal of Visual Communication and Image Representation*, 22(7):577–589, 2011.

[10] Yasmine Boulfani-Cuisinaud and Marc Antonini. Motion-Based Geometry Compensation for DWT Compression of 3D mesh Sequences. In *2007 IEEE International Conference on Image Processing*, volume 1, pages I – 217–I – 220, 2007.

[11] Chengju Chen, Qing Xia, Shuai Li, Hong Qin, and Aimin Hao. Compressing animated meshes with fine details using local spectral analysis and deformation transfer. *The Visual Computer*, 36(5):1029–1042, June 2019.

[12] Kunyao Chen, Fei Yin, Bang Du, Baichuan Wu, and Truong Q. Nguyen. Efficient registration for human surfaces via isometric regularization on embedded deformation. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–13, 2022.

[13] M.M. Chow. Optimized geometry compression for real-time rendering. In *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, pages 347–354, 1997.

[14] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.

[15] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. High-quality streamable free-viewpoint video. *ACM Trans. Graph.*, 34(4), jul 2015.

[16] Alexandros Doumanoglou, Dimitrios Alexiadis, Stylianos Asteriadis, Dimitrios Zarpalas, and Petros Daras. On human Time-Varying Mesh compression exploiting activity-related characteristics. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6147–6151, 2014.

[17] Alexandros Doumanoglou, Dimitrios S. Alexiadis, Dimitrios Zarpalas, and Petros Daras. Toward real-time and efficient compression of human time-varying meshes. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(12):2099–2116, 2014.

[18] Sylvie Gibet and Pierre-François Marteau. Analysis of human motion, based on the reduction of multidimensional captured data – application to hand gesture compression, segmentation and synthesis. In *Articulated Motion and Deformable Objects*, pages 72–81. Springer Berlin Heidelberg.

[19] Danillo Bracco Graziosi. Video-based dynamic mesh coding. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3133–3137, 2021.

[20] Qin Gu, Jingliang Peng, and Zhigang Deng. Compression of human motion capture data using motion pattern indexing. *Computer Graphics Forum*, 28(1):1–12, Mar. 2009.

[21] Sumit Gupta, Kuntal Sengupta, and Ashraf A. Kassim. Compression of Dynamic 3D Geometry Data Using Iterative Closest Point Algorithm. 87(1–3):116–130, jul 2002.

[22] Igor Guskov and Andrei Khodakovsky. Wavelet Compression of Parametrically Coherent Mesh Sequences. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '04, page 183–192, Goslar, DEU, 2004. Eurographics Association.

[23] Meha Hachani, Azza Ouled Zaid, and William Puech. Segmentation-based compression scheme for 3D animated models. *Signal, Image and Video Processing*, 10(6):1065–1072, Jan. 2016.

[24] Mohammadali Hajizadeh and Hossein Ebrahimnezhad. Eigenspace Compression: Dynamic 3D Mesh Compression

by Restoring Fine Geometry to Deformed Coarse Models. *Multimedia Tools Appl.*, 77(15):19347–19375, aug 2018.

[25] Mohammadali Hajizadeh and Hossein Ebrahimnezhad. NLME: A Nonlinear Motion Estimation-Based Compression Method for Animated Mesh Sequence. *Vis. Comput.*, 36(3):649–665, mar 2020.

[26] Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. Time-Varying Mesh Compression Using an Extended Block Matching Algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(11):1506–1518, 2007.

[27] hou Kai, ian Feng, ao Guo, and en Zhong. A clustering compression method for 3D Human motion capture data. In *2014 9th International Conference on Computer Science & Education*, pages 781–784, 2014.

[28] Chi-Kang Kao, Bin-Shyan Jong, and Tsong-Wuu Lin. Representing Progressive Dynamic 3D Meshes and Applications. In *2010 18th Pacific Conference on Computer Graphics and Applications*, pages 5–13, 2010.

[29] Zachi Karni and Craig Gotsman. Compression of soft-body animation sequences. *Computers & Graphics*, 28(1):25–34, Feb. 2004.

[30] Aris S. Lalos, Gerasimos Arvanitis, Aristotelis Spathis-Papadiotis, and Konstantinos Moustakas. Feature Aware 3D Mesh Compression Using Robust Principal Component Analysis. In *2018 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2018.

[31] Aris S. Lalos, Andreas A. Vasilakis, Anastasios Dimas, and Konstantinos Moustakas. Adaptive compression of animated meshes by exploiting orthogonal iterations. *The Visual Computer*, 33(6-8):811–821, May 2017.

[32] Chao-Hua Lee and Joan Lasenby. 3D Human Motion Compression Using Wavelet Decomposition. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, page 104–es, New York, NY, USA, 2006. Association for Computing Machinery.

[33] Chao-Hua Lee and Joan Lasenby. An Efficient Wavelet-Based Framework for Articulated Human Motion Compression. In *Advances in Visual Computing*, pages 75–86. Springer Berlin Heidelberg, 2008.

[34] Shiyu Li, Masahiro Okuda, and Shin ichi Takahashi. Compression of Human Motion Animation Using the Reduction of Interjoint Correlation. *EURASIP Journal on Image and Video Processing*, 2008:1–15, 2008.

[35] Shiyu Li, M. Okuda, and S. Takahashi. Hierarchical human motion compression with constraints on frames. In *The 2004 47th Midwest Symposium on Circuits and Systems, 2004. MWSCAS '04.*, volume 1, pages I–253, 2004.

[36] Guodong Liu and Leonard McMillan. Segment-based human motion compression. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, page 127–135, Goslar, DEU, 2006. Eurographics Association.

[37] Guoliang Luo, Frederic Cordier, and Hyewon Seo. Compression of 3D mesh sequences by temporal segmentation. *Computer Animation and Virtual Worlds*, 24(3-4):365–375, May 2013.

[38] Guoliang Luo, Zhigang Deng, Xiaogag Jin, Xin Zhao, Wei Zeng, Wenqiang Xie, and Hyewon Seo. 3d mesh animation compression based on adaptive spatio-temporal segmentation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '19, New York, NY, USA, 2019. Association for Computing Machinery.

[39] Guoliang Luo, Zhigang Deng, Xin Zhao, Xiaogang Jin, Wei Zeng, Wenqiang Xie, and Hyewon Seo. Spatio-Temporal Segmentation Based Adaptive Compression of Dynamic Mesh Sequences. *ACM Trans. Multimedia Comput. Commun. Appl.*, 16(1), mar 2020.

[40] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. Tfan: A low complexity 3d mesh compression algorithm. *Comput. Animat. Virtual Worlds*, 20(2-3):343–354, jun 2009.

[41] K. Mamou, T. Zaharia, F. Preteux, N. Stefanoski, and J. Ostermann. Frame-based compression of animated meshes in MPEG-4. In *2008 IEEE International Conference on Multimedia and Expo*, pages 1121–1124, 2008.

[42] Jean-Eudes Marvie, Maja Krivokuća, Céline Guede, Julien Ricard, Olivier Mocquard, and François-Louis Tariolle. Compression of time-varying textured meshes using patch tiling and image-based tracking. In *2022 10th European Workshop on Visual Information Processing (EUVIP)*, pages 1–6, 2022.

[43] Ha Q. Nguyen, Philip A. Chou, and Yinpeng Chen. Compression of human body sequences using graph wavelet filter banks. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6152–6156, 2014.

[44] Frédéric Payan and Marc Antonini. Temporal wavelet-based compression for 3D animated models. *Computers & Graphics*, 31(1):77–88, Jan. 2007.

[45] Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Simple and efficient compression of animation sequences. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, page 209–217, New York, NY, USA, 2005. Association for Computing Machinery.

[46] Nikolče Stefanoski and Jörn Ostermann. SPC: Fast and Efficient Scalable Predictive Coding of Animated Meshes. *Computer Graphics Forum*, 29(1):101–116, 2010.

[47] Robert W. Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. *ACM Transactions on Graphics*, 26(3):80, July 2007.

[48] L. Váša and V. Skala. COBRA: Compression of the basis for PCA represented animations. *Computer Graphics Forum*, 28(6):1529–1540, Sept. 2009.

[49] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008 papers*, pages 1–9. 2008.

[50] Toshihiko Yamasaki and Kiyoharu Aizawa. Patch-based compression for Time-Varying Meshes. In *2010 IEEE International Conference on Image Processing*, pages 3433–3436, 2010.

[51] Bailin Yang, Luhong Zhang, Frederick W. B. Li, Xiaoheng Jiang, Zhigang Deng, Meng Wang, and Mingliang Xu.

Motion-aware compression and transmission of mesh animation sequences. *ACM Trans. Intell. Syst. Technol.*, 10(3), apr 2019.

[52] Jeong-Hyu Yang, Chang-Su Kim, and Sang-Uk Lee. Progressive coding of 3D dynamic mesh sequences using spatiotemporal decomposition. In *2005 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 944–947 Vol. 2, 2005.

[53] Kai Zhou. An effective method for human animation compression. In *2013 International Conference on Wavelet Analysis and Pattern Recognition*, pages 85–89, 2013.