TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes

Guodong Chen*
Northeastern University
Boston, MA, USA
chen.guod@northeastern.edu

Filip Hácha
University of West Bohemia
Plzeň, Czech Republic
hachaf@kiv.zcu.cz

Libor Váša
University of West Bohemia
Plzeň, Czech Republic
lvasa@kiv.zcu.cz

Mallesham Dasari
Northeastern University
Boston, MA, USA
m.dasari@northeastern.edu

Abstract

Time-varying meshes (TVMs), characterized by their varying connectivity and number of vertices, hold significant potential in AR/VR applications. However, their practical use is challenging due to their large file sizes and the complexity of time-varying topology. Many time-varying mesh compression methods attempted to exploit redundancy between consecutive meshes to compress TVMs more efficiently, however, most face difficulties in establishing stable vertex and surface correspondence between the frames of a TVM. We propose TVMC, a novel TVM compression method that leverages volume tracking and extracts high-quality reference meshes for inter-frame prediction. Specifically, we use as-rigid-as-possible volume tracking to align consecutive TVMs and track volume centers, followed by multidimensional scaling to refine reference centers. This allows us to precisely deform a group of frames to the reference space and extract the reference mesh which is then deformed to approximate each mesh in the group to get displacement fields for TVM compression. Extensive experiments show that TVMC outperforms state-of-the-art methods (e.g., Google Draco, V-DMC 4.0, etc.), with bitrates of 4-6 Mbps compared to 9-12 Mbps for Draco and 10-15 Mbps for V-DMC 4.0. It reduces the decoding time by 66.1% compared to Draco and enables an increased group of frames (up to 15) without significant distortion.

CCS Concepts

• **Computing methodologies → Computer graphics**.

Keywords

Time-varying mesh, volume tracking, mesh deformation, intersurface mapping, mesh compression

ACM Reference Format:
Guodong Chen, Filip Hácha, Libor Váša, and Mallesham Dasari. 2025. TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes. In *ACM Multimedia Systems Conference 2025 (MMSys '25), March 31-April 4, 2025, Stellenbosch, South Africa.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3712676.3714440

1 Introduction

With the rapid development of 3D sensing technologies [20], it has become increasingly feasible to produce complex 3D content with a high level of detail and quality. One of the most common and efficient ways of 3D content representation is triangle mesh sequences with changing topology and connectivity, also called time-varying meshes (TVMs). Differing from meshes in a 3D animation, time-varying meshes consist of a set of mesh frames that include time-varying topology, vertex numbers, vertex positions, and face connectivity, and each of them demands massive geometry and texture information to render high-quality visuals. Because of this, TVMs require a substantial amount of data for storage and transmission, necessitating an efficient compression method, as real-time streaming is impractical without using advanced networks [13], which are not available today.

Popular mesh compression tools, such as Google Draco [11], using methods such as TFAN [28] or EdgeBreaker[36], perform well in terms of compression ratio and latency by compressing each mesh *individually*. Each mesh in the sequence is treated as a separate entity during the compression process, so there is no need to manage dependencies or relationships between meshes. This feature provides Draco flexibility in handling different types of meshes; however, it does not exploit the inter-frame redundancy across the sequence of mesh frames and results in very high data rates for high-quality complex mesh sequences.

To improve mesh compression ratio, many other works have attempted to utilize spatial and temporal correlation to encode and decode meshes more efficiently. Some existing methods [1, 24, 26, 39] focus on dynamic mesh sequences with consistent topology across different frames, leveraging temporal correlation to improve compression performance. However, these approaches require the meshes to maintain a consistent topology and connectivity, similar to 3D animation, which limits their applicability to mesh sequence compression in real-world applications.

The Moving Picture Experts Group (MPEG) has developed several mesh compression standards, including IC (Interpolator Compression) [5], MESHGRID [37], and FAMC (Frame-based Animated Mesh Compression) [27]. Similarly, these standards are limited to processing dynamic mesh sequences with constant connectivity and cannot handle TVMs with changing topology. To address this issue, the MPEG 3D Graphics Coding (3DG) group recently issued a Call for Proposals (CfP) [30] for the Video-based Dynamic Mesh

Coding (V-DMC) standard. In response, Apple Inc. proposed video and subdivision-based mesh coding (VSMC) [26], which was selected as the foundation for the V-DMC standard. However, the temporally consistent re-meshing process required by VSMC is not always feasible, which imposes constraints on TVMs. Thus, efficient and effective compression of TVMs remains an open challenge.

Recently, embedded deformation has been used in 3D mesh compression [17, 21]. TVMs lack explicit vertex correspondence between frames and usually have varying connectivity and numbers of vertices over time, so establishing inter-frame correspondence and representing inter-frame differences is challenging. Embedded deformation addresses this problem by deforming a selected keyframe into the target frame of similar shape while preserving its topology. However, these approaches face fundamental limitations when self-contact regions (Section 3.3) exist in the keyframe, often resulting in severe visual artifacts and inaccuracies in decoded meshes.

To address the above limitations, we propose TVMC, a novel time-varying mesh compression method using volume-tracked reference meshes. TVMC not only exploits inter-frame redundancy to achieve a high compression ratio but also significantly reduces the deformation artifacts and distortions caused by self-contact regions. Specifically, we first adopt ARAP volume tracking to identify volume centers' positions within each mesh frame and the largest distances between each other during the whole sequence, along with a measurement of how tightly each pair of centers is bound together called *affinity* (Section 3.1). These affinities are then processed using Multidimensional Scaling (MDS) to derive a new set of centers called *reference centers* existing in reference space (Section 3.2). With the stable and accurate centers correspondence, we adopt center affinity deformation that can separate self-contact components, which is hard to achieve via traditional Radial Basis Functions (RBFs) mapping or key-vertices-based embedded deformation (Section 3.3). The affinity and Euclidean distance between centers is employed to calculate the *mesh self-contact degree* (Section 3.4). We deform meshes with a low self-contact degree to the reference space and use Poisson surface reconstruction to extract the volume-tracked reference mesh without self-contact regions (Section 3.5). Finally, we use center affinity deformation again to deform the reference mesh to approximate any mesh in the TVMs and compute displacements using subdivision surface fitting (Section 3.6). We show that our TVMC method outperforms Google Draco [11], V-DMC 4.0 [32], and recent embedded deformation-based methods such as [21] in terms of geometry compression ratio and decoding time. We highlight TVMC's performance from our evaluation as follows:

- TVMC achieves higher geometry compression rates compared to the state-of-the-art methods Draco and V-DMC 4.0, at a framerate of 30 fps and a D2-PSNR of around 80, with bitrates of 4 to 6 Mbps vs. 9 to 12 Mbps for Draco and 10 to 15 Mbps for V-DMC 4.0, respectively.
- TVMC supports real-time applications, reducing the decoding time by 66.1% compared to Draco.
- TVMC significantly increases the number of frames in a group of frames (GoF) (GoF = 15 vs. GoF = 5 in [21]) that can be compressed together without leading to rapid accumulated distortion.

2 Related Work and Limitations

There have been numerous works focused on 3D mesh compression [25, 34]. Based on the constraints of mesh sequence, we roughly divide them into static mesh compression [11, 28], dynamic mesh compression [1, 24, 26, 39], and time-varying mesh compression [7, 12, 15–19, 21, 29, 40]. Static mesh compression treats each mesh independently, it utilizes intra-redundancy to compress a single mesh. On the other hand, dynamic mesh compression compresses a sequence of meshes by storing a base mesh or key mesh along with a displacement field, which is feasible since dynamic meshes have a constant number of vertices and a constant connectivity. In contrast, time-varying mesh compression deals with mesh sequences where the geometry information changes and the topology and connectivity also vary, making it a more complex problem to solve.

2.1 Static Mesh Compression

Static mesh compression involves compressing vertices' positions, connectivity, textures, and other attributes. A prominent algorithm, Edgebreaker [36], developed by Rossignac in 1999, offers an efficient approach for manifold 3D mesh compression. In the Edgebreaker algorithm, each new triangle is adjacent to an already encoded one, enabling efficient compression of vertex positions and attributes such as normals. Using parallelogram prediction, Edgebreaker stores the difference between predicted and actual values, typically small, rather than absolute values. Using Edgebreaker, the state-of-the-art mesh compression tool, Draco [11], balances simplicity, efficiency, and performance well. However, Draco's limitation is that it compresses each mesh independently, which prevents it from efficiently compressing TVMs. This means it cannot utilize the temporal correlations that may be present across multiple meshes.

2.2 Dynamic Mesh Compression

Dynamic mesh is a sequence of static meshes with the same topology and connectivity that describes a temporal development of some physical surface, a common representation format in 3D animation. In 2021, the MPEG 3DG group issued a Call for Proposals on dynamic mesh coding [30], aiming to develop a new standard under the Visual Volumetric Video-Based Coding (V3C) standard, known as the V-DMC standard. Then the VSMC scheme proposed by Apple [26] was selected as the basis for the V-DMC standard. V-DMC uses inter-frame coding for dynamic meshes, representing a mesh with a decimated base mesh and a set of displacements generated from subdivision surface fitting. It estimates a motion field by tracking corresponding vertices between the current mesh and the reference mesh. The reference base mesh is then deformed to approximate the current base mesh, and ultimately subdivided and adjusted to fit the current mesh to get updated displacements for reconstruction.

However, to leverage temporal correlation, V-DMC requires the input mesh sequence to maintain a constant topology and connectivity. Although VSMC proposed a time-consistent re-meshing method to relax the constraint on mesh types, it is not always feasible, making the representation of inter-frame differences between the frames of a TVM still challenging.

2.3 Time-varying Mesh Compression

Modern 3D reconstruction techniques use a combination of RGB-D cameras, infrared (IR) cameras, and structured light sources to generate detailed 3D meshes. These meshes are typically time-varying, with different numbers of vertices, connectivity, and topology in each frame. So, how to process and utilize them efficiently is a major focus of research in this field.

Early work [41] used the first mesh frame in a given sequence as the 'reference mesh'. This first mesh is converted into a semi-regular normal mesh, followed by motion estimation to map it onto the following meshes. However, it accumulates errors frame by frame and brings significant artifacts and distortions.

In 2015, Collet et al. [6] estimated a feasibility score of each frame being a keyframe. They selected a relatively small set of meshes as keyframes and enforced that each keyframe covers a continuous frame range. In their keyframe choosing algorithm, they used larger surface area, lower-genus surface, and more connected components as the standard of identifying keyframe, which solved the self-contact problem to a certain extent. However, since they used a non-rigid ICP algorithm [23] to fit a keyframe to its neighboring frames, the fitting error is unavoidable with only vertex information. Furthermore, the algorithm repeats this process on the remaining frames until every frame is associated with a keyframe, which is computationally expensive.

Some other methods use embedded deformation [38] to deform selected reference mesh to each frame while maintaining topology. Based on the movements of strategically selected key points, a geometric transformation operator is applied to each vertex in the previous mesh to 'predict' the following meshes.

Hoang et al. [17] proposed an optimization-based technique to identify the optimal number and positions of key nodes for embedded mesh deformation. This approach minimizes the number of nodes required while ensuring relatively high-quality reconstruction. Similarly, Jin et al. from the KDDI Group were inspired by the embedded deformation proposed in [38] to construct an embedded graph representation for performing difference mapping between meshes [21]. However, these methods apply transformations to every vertex in the reference mesh based on merely geometry information, which can cause significant distortions when the reference mesh contains self-contact regions. Figure 1 illustrates this issue: if a mesh of a person placing their fingers on a basketball is deformed to a subsequent frame where the fingers and the basketball are separate, the vertices that were temporarily missing due to self-contact will also be missing in the deformed mesh, which leads to severe visual distortions where fingers still stick to the basketball (shown in Figure 1a).

3 TVMC Design

To address the challenges mentioned above, we propose a volume-tracked reference mesh-based time-varying mesh compression framework called TVMC. We use volume centers to build stable center correspondence rather than relying on traditional surface and vertices correspondence. We also adopt center affinity-based mesh deformation to deform a mesh by controlling its volume centers, which we demonstrate is more accurate than embedded deformation through key point control. TVMC creates a general, self-contact-free, volume-tracked reference mesh, significantly reducing deformation distortions caused by the self-contact regions. Extracted volume-tracked reference mesh can serve as the basis of many mesh compression methods that require a reference mesh, and center affinity-based mesh deformation can be a superior alternative to embedded deformation.

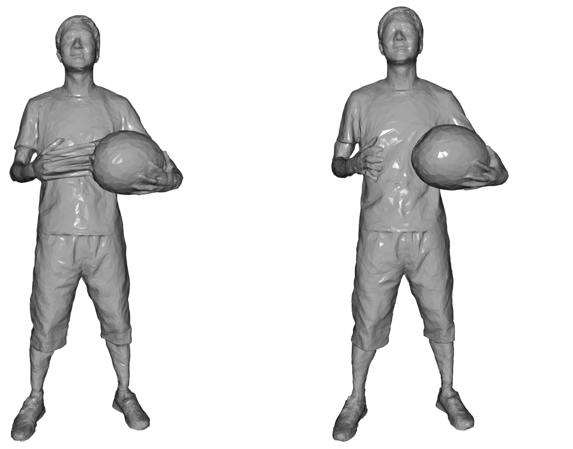

(a) With self-contact regions (b) Without self-contact regions

Figure 1: Impact of self-contact regions on deformation quality. Both meshes are deformed to the same target mesh. For Figure 1a, the fingers touch the basketball, leading to deformation distortions caused by self-contact regions. For Figure 1b, the fingers and basketball are separate, resulting in a more artifact-free deformation.

Figure 2 shows the simplified workflow of TVMC. The process begins with feeding a TVM sequence to As-Rigid-As-Possible (ARAP) volume tracking [9], which uniformly distributes a set of centers within the enclosed volume of each mesh frame. A Global Optimization step [8] is optional to improve the quality of the set of volume centers by eliminating several abnormal centers. The final tracked centers are then processed using Multidimensional Scaling (MDS) [4] to derive a set of reference centers that serve as the basis for subsequent operations. The input mesh sequence is then fed into Affinity Evaluation Module to calculate the self-contact degrees. We employ Mesh Editing [14] to deform meshes with a low self-contact degree to the reference space around reference centers by utilizing center affinity. Then the Reference Mesh Extraction Module adopts subdivision surface fitting to reduce the deformation deviation and generates a volume-tracked reference mesh through Poisson surface reconstruction [22]. This volume-tracked reference mesh is then deformed to approximate each mesh in the group while maintaining its topology and connectivity. Finally, we use the Subdivision Surface Fitting Module again to compute displacements between each reconstructed mesh and the volume-tracked reference mesh, enabling us to reconstruct each frame using the volume-tracked reference mesh and displacements.

3.1 As-Rigid-As-Possible Volume Tracking

Consider a TVM sequence, $S = \{M_0, M_1, \ldots, M_{N-1}\}$, where N is the total number of frames in the sequence. Let $M(t) = (V_t, E_t, F_t)$ be a static mesh at time t. V_t, E_t, and F_t represent the sets of vertices,

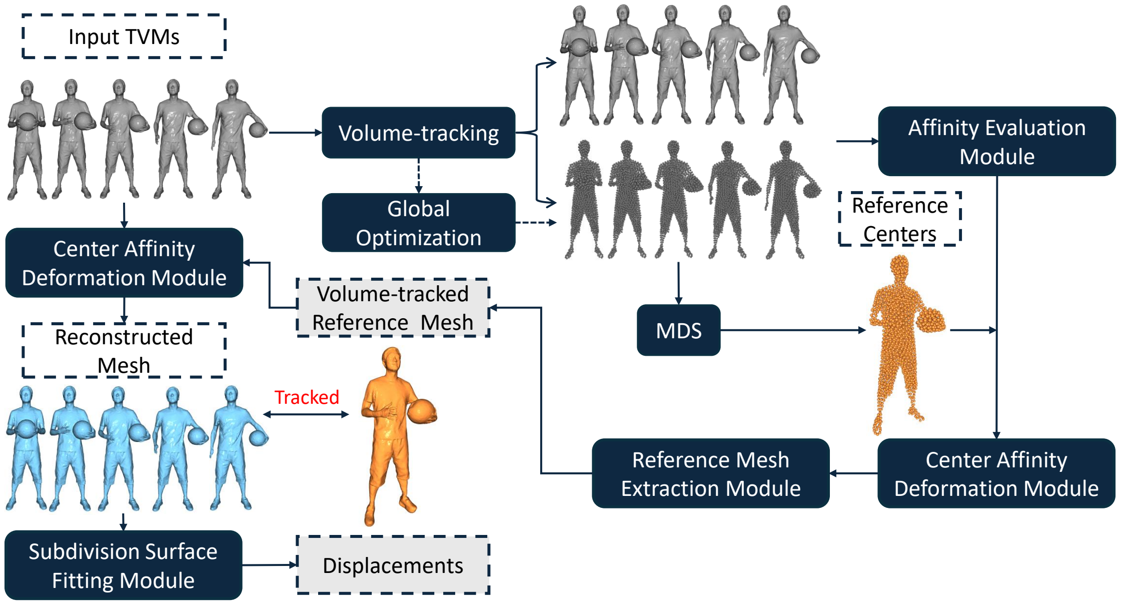

Input TVMs
Volume-tracking
Global Optimization
Affinity Evaluation Module
Reference Centers
Center Affinity Deformation Module
Volume-tracked Reference Mesh
MDS
Reconstructed Mesh
Tracked
Reference Mesh Extraction Module
Center Affinity Deformation Module
Subdivision Surface Fitting Module
Displacements

Figure 2: Simplified workflow of TVMC. TVMC consists of several key steps represented by colored boxes. The dotted boxes indicate the outputs of specific operations, while the grayed dotted boxes highlight the only two elements requiring compression. Gray meshes are the input TVMs, accompanied by their corresponding gray centers. The reference centers are shown in orange, along with the orange reference mesh, which represents the volume-tracked reference mesh. This reference mesh is deformed to the blue reconstructed meshes. The term "tracked" refers specifically to only changes in vertex positions, while maintaining the same connectivity and topology.

edges, and faces (triangles), respectively. We apply As-Rigid-As-Possible Volume Tracking and set the mode to *max affinity based tracking*, as proposed in [8], to the TVM sequence S. The method identifies a configurable number of K points set C, referred to as centers. Each center represents a small volume surrounding it with positions that vary over time. Each center follows a certain trajectory $c_i = [c_i(0), c_i(1), \dots, c_i(N-1)], c_i(f) \in \mathbb{R}^3$, where $c_i(f)$ is the position of the i-th center in the f-th frame. For any frame f, we define $C_f = \{c_1(f), c_2(f), \dots, c_K(f)\}$ as the set of all centers' positions in that frame. Briefly, the uniformly distributed centers are generated as follows. First, after setting a resolution for the longest axis, each mesh is converted to a tight regular square voxel grid. Then the fast winding number [2] is used to define the so-called indicator function (IF), identifying which voxels of the grid are enclosed by the mesh using the following equation:

$$IF(x) = \begin{cases} 1 & \omega_f(x) > \mu \\ 0 & \omega_f(x) \le \mu \end{cases}, \tag{1}$$

where $\omega_f(x)$ represents the fast winding number and μ is a threshold. In this paper, we set $\mu = 0.5$. The threshold of fast winding number provides some robustness to non-watertight meshes so TVMC does not strictly require fully watertight meshes. Enhancing the fast winding number algorithm and distinguishing open surfaces from relatively enclosed objects before volume tracking can improve TVMC's performance when handling inputs with open surfaces. Next, the algorithm samples K random voxels with $IF(x) = 1$ and denotes them centers. Each of the remaining voxels is then connected with its nearest center, and every center is moved to its surrounding neighbors' centroid iteratively. This approach ensures uniform distribution of the selected centers in the first frame. For each subsequent frame, the set of centers from the previous frame will first be linearly extrapolated, then an optimization energy function is minimized to ensure uniformity and smoothness of the next sets of centers.

We refer the reader to the original paper for a detailed description of the As-Rigid-As-Possible Volume Tracking method [8, 9].

3.2 Reference Centers Generation: Multidimensional Scaling

Next, we generate reference centers as the basis of the volume-tracked reference mesh. Reference centers are a specialized set of volume centers located in the reference space. Similar to C, they represent the area inside the mesh surface, but the reference mesh they represent is adjusted to a self-contact-free shape. In order to get relative positions of centers without self-contact, we first build a distance matrix from the largest distances of centers. Let d_{ij} be the largest distance, a_{ij} be the *affinity* between two centers c_i and c_j across the whole TVM. Here, affinity is a function of the maximum distance between the centers, representing how strongly two entities are bound together. The largest distance and *affinity* between two centers c_i and c_j is calculated using the Euclidean distance:

$$d_{ij} = \max_f \|c_i(f) - c_j(f)\|, \tag{2}$$

$$a_{ij} = \exp(-(\alpha d_{ij})^2), \tag{3}$$

where $f \in [0, N-1]$ and α is a weight decided by the global scale of the input TVM. The main idea behind maximum-distance-based affinities is that if two centers are far enough apart in any part of the sequence, they probably are not related to each other in the rest of the sequence, no matter how close they may get. Such centers should then not affect the deformation of the same part of the surface.

The largest distance matrix D is then defined as $D = [d_{ij}]_{K \times K}$. Once the distance matrix D is computed, it is used as the input of Multidimensional Scaling (MDS). MDS seeks to find a set of points $r_1, r_2, \dots, r_K$ in a 3-dimensional space that minimizes the stress function given by:

$$\min \sigma(R) = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - \|r_i - r_j\|\right)^2}{\sum_{i<j} d_{ij}^2}}. \tag{4}$$

Here, $R = [r_1, r_2, \dots, r_K]$ represents the reference centers we want in the 3-dimensional space and $\|r_i - r_j\|$ is the Euclidean distance between points r_i and r_j in the set of reference centers R. In R, connected centers should remain contiguous, while those that are, in fact, disconnected should be apart since we enforce the maximum distance over the length of the sequence. Finally, we perform a rigid registration of R that aligns R with C_f so that the volume-tracked reference mesh retains as many static regions as possible from the original meshes in the group. We use Singular Value Decomposition (SVD) to do this and call the space where the reference centers are located the reference space.

3.3 Self-contact Degree Evaluation

As mentioned in Section 2.3, deformations in self-contact regions can lead to severe distortions. To create a high-quality volume-tracked reference mesh, it is essential to gather as comprehensive surface information as possible and minimize the potential impact caused by self-contact regions. To this end, we exclude a certain number of the meshes in the group during the creation of the volume-tracked reference mesh, which allows us to increase the group size and improve the overall performance of TVMC.

Simply relying on the variation of the number of vertices, triangles, or vertex positions in TVMs to identify self-contact regions is not feasible. By utilizing center trajectories and affinity, we can estimate the likelihood of the presence of self-contact regions. Centers that are tightly bound together have a large affinity between them, while the further they move apart, the lower the affinity between them will be. Therefore, a pair of centers with low affinity but also small Euclidean distance suggests that these two centers are only temporarily close, which indicates an increased likelihood of self-contact region appearance.

Motivated by this observation, we calculate the mesh self-contact degree $SD = [sd_0, sd_1, \dots, sd(N-1)]$ using the following formula that adds up the product of affinity between all centers and their current Euclidean distance:

$$sd_f = -\sum_{i<j} (a_{ij} \cdot \|c_i(f) - c_j(f)\|). \tag{5}$$

The smaller the self-contact degree is, the more likely it is that the self-contact regions exist.

3.4 Center Affinity Deformation

For any frame f, we have C_f that defines the set of all centers' positions in f. With the set of reference centers R, it is possible to construct a transformation $\hat{T_{i,f}}$ for each center c_i from the space of frame f to the reference space. Firstly, we introduce the concept of embedded deformation [38] introduced in [17, 21], which allows mesh deformation while preserving the natural properties of the original meshes to a certain extent. We then provide a detailed description of applying embedded deformation with center information and incorporate center affinity to perform center affinity-based deformation.

In [17], a set of key points $KP = [p_0, p_1, \dots, p_{P-1}] \in \mathbb{R}^3$ is generated to control the areas around each key point, where P is the number of key points. Then a key point matching or registration method is applied to get key point correspondence between frame $f-1$ and frame f. For each vertex v_i in frame $f-1$, its position is updated according to the movements of its K nearest key points using the following equation:

$$v'_i = \sum_{j=0}^{K-1} w_{ij} (R_j (v_i - p_j) + T_j + p_j), \tag{6}$$

where w_{ij} is a distance-dependent influence weight. R_j and T_j are determined by solving:

$$\underset{R_j, T_j}{\arg\min} (E_{geometry} + \alpha E_{rotation} + \beta E_{smoothness}), \tag{7}$$

where $E_{geometry}$, $E_{rotation}$, and $E_{smoothness}$ penalize geometric error, rotation error, and smoothness error, respectively, and α and β are two weights. In [21], a mesh is first decimated into a base mesh. The vertices of the base mesh take place of the key points and the remaining steps follow a similar approach.

However, these embedded deformation methods update a vertex's position entirely depending on spatial Euclidean distance. The process of choosing K nearest key points would unavoidably choose key points that are close in terms of Euclidean distance but belong to different parts of the mesh. To address this issue, we adopt an improved deformation method called center affinity deformation. Center affinity deformation respects the topology by relying on center affinity instead of Euclidean distance to select the most suitable centers. The mesh deformation will significantly align with the transformation of its centers, while maintaining its topology.

In terms of transformation $\hat{T_{i,f}}$, we use dual quaternions to represent rigid transformations. We can represent a dual quaternion as $A + \epsilon B$, where A and B are ordinary quaternions and ϵ is the dual unit, which satisfies $\epsilon^2 = 0$. An ordinary quaternion can be represented as $a + b\mathbf{i} + c\mathbf{j} + d\mathbf{k}$, which satisfies $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. Leveraging the properties of dual quaternions, we can describe a transformation with a rotation quaternion R and a transformation vector $T = (x, y, z)$ in the format of $A = R + (x/2, y/2, z/2, 0)\epsilon$. We use dual quaternions because they are easy to blend and inverse, which makes them convenient and efficient for center-affine deformations.

Specifically, for each vertex v_i in frame f, its nearest center $c_{(1),f}$ is determined as:

$$c_{(1),f} = \underset{c_{(1),f}}{\arg\min} \|c_{(1),f} - v_i\|. \tag{8}$$

We assume that the nearest center $c_{(1),f}$ can best reflect the motion trajectory of the vertex v_i. Then we find the next $K-1$ 'nearest' centers with the highest affinity to the center $c_{(1),f}$. The blended transformation for updating vertices can be expressed as:

$$T_i = \frac{\sum_{j=1}^{k} w_j \cdot \hat{T}_{j,f}}{\sum_{j=1}^{k} w_j}. \tag{9}$$

With the normalization of dual quaternion, the vertex v_i can be updated to:

$$v_i' = \frac{T_i}{\|T_i\|}(v_i). \tag{10}$$

Based on a smooth minimum function called Boltzmann operator [3], we get w_j from the following equations:

$$d_j = \|c_{(j),f} - v_i\|, j \in 1, 2, \ldots, K, \tag{11}$$

$$w_j' \approx \frac{\exp(\frac{-d_j}{\sigma d_1 + \epsilon})}{\sum_{k=1}^{K} \exp(\frac{-d_k}{\sigma d_1 + \epsilon})}, \tag{12}$$

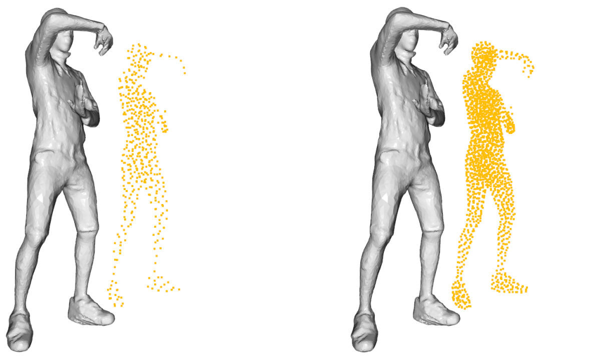
$$w_j = \frac{w_j' - \min_x(w_x')}{\sum_{k=1}^{K}(w_k' - \min_x(w_x'))}. \tag{13}$$

Distances d_j represent the distance between the j-th affine center to v_i, a small value ϵ is in the denominator to replace the calculation of a limit, thus simplifying the evaluation. σ controls the smoothness of the weights. After the normalization step in Equation 13, we obtain a set of weights that sum to one, which keeps all the centers inside the mesh surface during the deformation if the surface is sufficiently finely sampled.

In contrast to the original embedded deformation weights, the weights calculated in this way satisfy the interpolation condition, i.e., if the position of the deformed point v is the same as the position of the center c_i, then its weight $w_i = 1$ and the weights of the other centers $w_j = 0 \forall j \neq i$. For a more detailed description and other applications of center affinity deformation, we refer the reader to [14].

3.5 Volume-Tracked Reference Mesh Extraction

To improve the overall performance, TVMC divides the frames of the input TVM into several groups of frames and each group contains *GoF* meshes. As mentioned in Section 2.3, deformation in self-contact regions often leads to noticeable distortions, which can negatively impact the results of center affinity deformation and, consequently, the quality of the volume-tracked reference mesh. So, for each group, we drop $GoF/2$ meshes with a high possibility of containing self-contact regions by calculating their self-contact degrees introduced in Section 3.3, and adopt center affinity deformation to deform all the remaining frames to the reference space. Although our center affinity deformation method can deform meshes with different shapes to the same reference space with high accuracy, slight surface offsets exist between these deformed meshes. So, we employ subdivision surface fitting to improve the quality of our extracted volume reference mesh.



(a) ARAP volume tracking with 500 centers

(b) ARAP volume tracking with 2000 centers

Figure 3: Comparison of different numbers of centers. The orange points represent the volume centers of the gray mesh. In Figure 3b, 2000 centers capture the human body more accurately compared to the 500 centers shown in Figure 3a.

The subdivision surface fitting module first simplifies a mesh using mesh decimation methods, in this paper we use Quadric Error Metric (QEM) decimation [10]. Selecting the deformed meshes with the least self-contact degree as the key mesh, we feed the key mesh and all the decimated meshes to the subdivision surface fitting module. These decimated meshes are subdivided using one of the subdivision schemes in [35]. Here, we adopt the mid-point subdivision scheme for the entire process as it is computationally efficient and preserves the original mesh's topology.

Then, each vertex in these subdivided meshes will be updated to its nearest vertex in the key mesh. We use KDTrees to implement this in this paper. Subsequently, we only keep all the vertices and use this dense vertex set and Poisson surface reconstruction [22] implemented by Open3D [43] to extract a single surface mesh, with a setting of $depth = 9$. Based on the average numbers of vertices and triangles, we use QEM decimation again to simplify this to a certain degree and call this simplified mesh volume-tracked reference mesh. For the group of *GoF* meshes, this volume-tracked reference mesh contains complete components and surfaces to approximate each mesh with the help of center affinity deformation described in Section 3.4.

In TVMC, the quality of the volume-tracked reference mesh and overall compression performance is influenced by the number of centers we generated from ARAP volume tracking. Figure 3 shows the difference in visual results of a mesh from "Levi" sequence with different numbers of centers. Denser centers can better represent finer parts, such as the hands and lower legs, as compared between Figure 3a and Figure 3b, allowing ARAP volume tracking to be more effective. A smaller number of centers typically results in greater distortions when representing narrow regions, reducing the quality of the reference mesh and negatively impacting compression RD performance. However, using a larger number of centers increases computational costs and poses challenges to MDS in generating a high-quality set of reference centers. In this paper, the number of centers is set from 1000 to 4000 for all sequences.

3.6 Time-varying Meshes Compression

TVMC deforms the volume-tracked reference mesh to a geometric approximation f' of the frame f using the positions of reference centers and the positions of the centers C_f of the frame f, and ensures that f' has the same topology as the volume-tracked reference mesh. Therefore it is feasible to use simple displacement coordinates $dis_i \in \mathbb{R}^{n\times3}, i \in 1, 2, \ldots, GoF$ to represent each frame in the group, where n is the vertex number of the reconstructed meshes.

The volume-tracked reference mesh is compressed using the state-of-the-art mesh compression tool Draco [11] in relatively high quality with settings of $qp = 14, cl = 7$. Displacements are stored in *PointClouds* and we use Draco to compress them with the setting of *pointcloud*. Note that the compression of displacements and the volume reference mesh are separate, any other compression method such as MPEG TMC13 version 20.0 [31] or video coding method can be adopted.

On the decoder side, the volume-tracked reference mesh and displacements are decoded and they are fed into the Reconstruct Module to simply update the vertices positions of the volume-tracked reference mesh while maintaining its connectivity.

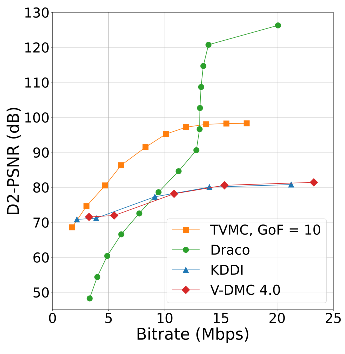
4 Experiments and Analysis

We conducted our experiments on a Lambda Vector equipped with an AMD Ryzen Threadripper 7960X 24-core 4.20 GHz CPU.

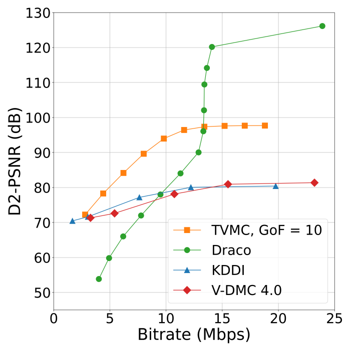
4.1 Datasets and Metrics

In this section, we evaluate TVMC on four complex TVM sequences that are defined and used by MPEG V-DMC [30]: "Dancer", "Basketball player", "Mitch", and "Thomas". These 4 TVM sequences, sourced from [42], represent human 3D objects with geometric and texture information. As our focus in this paper is on the compression of geometry information, we exclude the texture information and conduct experiments on 30 frames of each TVM. Specifically, we select the first 30 frames for "Dancer", "Basketball player", and "Mitch", for "Thomas" we select 120-th to 149-th frames because these 30 frames exhibit significant self-contact regions variation. We selected "Basketball Player" and "Thomas" specifically because they feature more self-contact regions, which present additional challenges for compression and can demonstrate TVMC's ability to efficiently deal with self-contact problems.

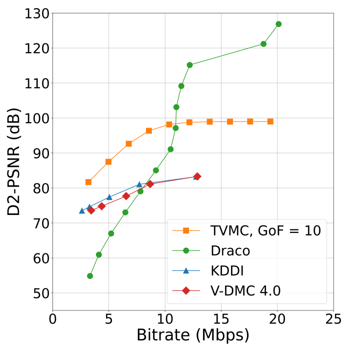
To evaluate the quality of decoded meshes, we use the state-of-the-art PSNR metrics as defined by the MPEG group [33] along with the root mean square error (RMSE) to quantify the degree of similarity between two sets of geometric data. PSNR metrics are well-established and are widely used in evaluating the performance of mesh compression techniques. In contrast, the structural similarity index measure (SSIM) is designed for 2D image data and is less suitable for evaluating 3D mesh geometry, so we do not consider SSIM in this work. The PSNR metrics include point-to-point error (D1) and point-to-plane error (D2). For 3D meshes, D2-PSNR is generally regarded as a more accurate evaluation metric, as it takes into account the normal vector of the underlying surface when calculating errors. We also evaluate the encoding and decoding time of TVMC and compare it with Draco [11], because our goal is to meet the real-time application requirement of at least 30 fps for decoding. Moreover, we include subjective evaluations to show TVMC's performance.



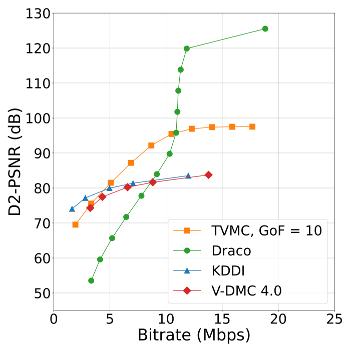
(a) RD performance and comparisons of "Dancer"



(b) RD performance and comparisons of "Basketball player"



(c) RD performance and comparisons of "Mitch"



(d) RD performance and comparisons of "Thomas"

Figure 4: Objective RD performance evaluation on 4 TVMs: "Dancer", "Basketball player", "Mitch", and "Thomas". We vary quantization parameter qp between 7 to 16 to generate the curves for TVMC while varying qp between 7 to 19 for Draco to reach a similar range of bitrates. The KDDI and V-DMC 4.0 curves are sourced from [21] for the same dataset.

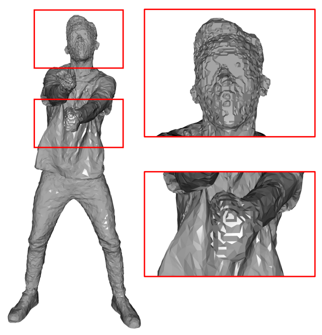
4.2 Experiments Details

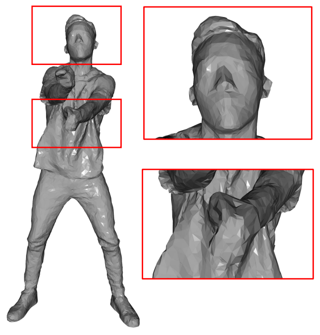
When applying ARAP volume tracking for all sequences, we set the mode to *max affinity based tracking* and volume grid resolution to 512. Additionally, we use $eps = 1e-10$ for MDS to generate reliable sets of reference centers. For center affinity deformation, we employ dual quaternions as a representation of rigid transformations. The parameter α mentioned in Section 3.2 for calculating center affinity is in the range from 10^{-2} to 10^{1}. The value of parameter σ for calculating distance-dependent influence weight when adopting center affinity deformation in Section 3.3 is set to 2.

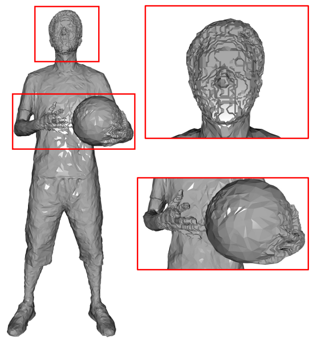
In TVMC, we use Draco to compress our generated volume-tracked reference mesh and displacements. Draco uses the parameter qp to control the level of geometric information quantization. To balance the trade-off between compression efficiency and quality, we set $qp = 14$ for the reference mesh and vary qp from 7 to 16 for displacements to achieve the target bitrates. We avoid $qp \leq 6$ as it introduces unacceptable artifacts.

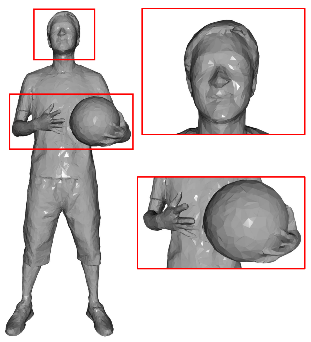
Table 1: Objective comparisons between TVMC and Draco for the qualitative results shown in Figure 5. D2-PSNR is calculated using the PSNR metrics defined by the MPEG group [33], where higher PSNR values indicate better quality. RMSE values are presented as the log10 of the raw RMSE, smaller number indicates a smaller average difference between the original and reconstructed mesh. Table 1 also includes average encoding and decoding times, measured with a *GoF* setting of 10. Additionally, we illustrate the bitrate proportion allocated to displacements (for *GoF* frames total).

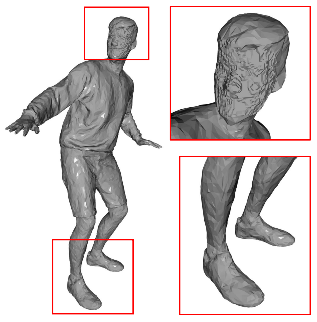
TVMs	TVMC					Draco			
	D2-PSNR	RMSE	Encoder	Decoder	Portion	D2-PSNR	RMSE	Encoder	Decoder
Dancer	84.10	-3.63	13.38 s	15.20 ms	93.08%	66.55	-2.76	181.70 ms	46.20 ms
Basketball player	84.14	-2.65	21.12 s	14.60 ms	93.14%	66.01	-1.74	229.27 ms	47.07 ms
Mitch	87.46	-3.78	10.65 s	13.40 ms	93.36%	66.99	-2.76	139.33 ms	35.40 ms
Thomas	81.50	-3.54	10.88 s	12.60 ms	93.52%	65.65	-2.75	137.93 ms	35.60 ms

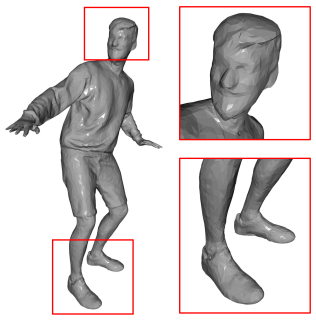

(a) Draco, "Dancer", 6.13 Mbps

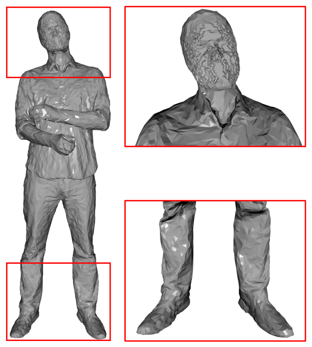

(b) TVMC, "Dancer", 6.12 Mbps


(c) Draco, "Basketball", 6.19 Mbps


(d) TVMC, "Basketball", 6.21 Mbps


(e) Draco, "Mitch", 5.20 Mbps


(f) TVMC, "Mitch", 4.97 Mbps


(g) Draco, "Thomas", 5.20 Mbps

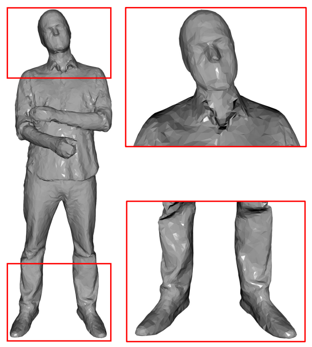

(h) TVMC, "Thomas", 5.09 Mbps

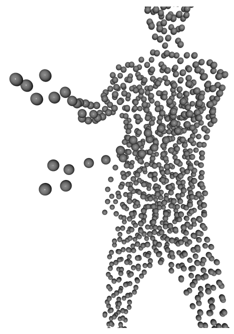
Figure 5: Subjective evaluation of "Dancer", "Basketball player", "Mitch", and "Thomas". Results using Draco and TVMC are compared at bitrates between about 5 and 6. Key differences in the face, hands, and feet are highlighted and zoomed within the red frames for closer comparison. Specific bitrates for each result are indicated in the captions of each subfigure.

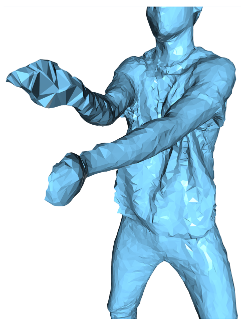
4.3 Experiments Results

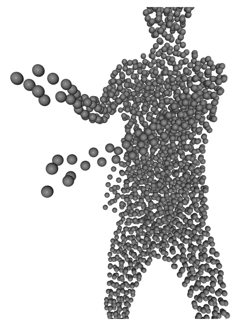
In TVMC, the number of centers (Section 4.3.2) and Group of Frames length (GoF) (Section 4.3.3) are two main parameters. We show full experiment results of RD performance for each TVM and compare them with Google Draco [11], V-DMC 4.0 [32], and an embedded deformation-based method proposed by KDDI group [21]. Note that we directly use the results of V-DMC 4.0 and the embedded deformation-based method presented in [21]. In the remainder of this section, we evaluate how different numbers of centers and GoF settings influence TVMC.

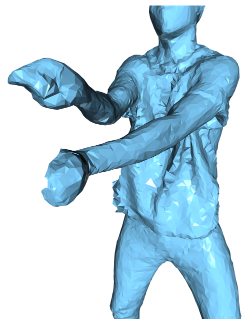
4.3.1 Comparisons. Figure 4 shows the objective evaluation of RD performance in D2-PSNR vs. bitrates. We compare TVMC with three baselines: Draco [11], the embedded deformation-based method proposed by KDDI group [21], and V-DMC 4.0 [32]. We set the number of centers to 1000, 2000, 3000, and 4000, and the GoF values to 5, 10, and 15 for all 4 TVM sequences. Due to page limits, we present only the results with 2000 volume centers and a GoF of 10.

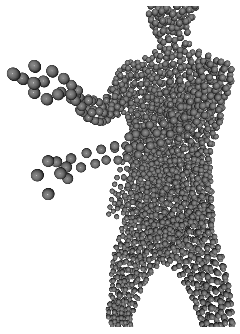
From Figure 4, we can see that TVMC outperforms Draco, KDDI's method, and V-DMC 4.0, with the bitrates varying from 5 Mbps to 10 Mbps. Take "Dancer" as an example, TVMC can achieve a D2-PSNR of 80 with a bitrate less than 5 Mbps, while Draco, KDDI's method, and V-DMC require about 10 Mbps, 14 Mbps, and 15 Mbps to get

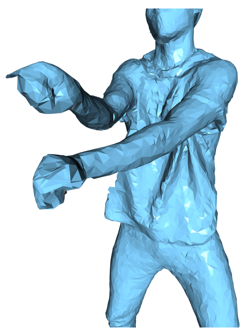

(a) Visualization of 1000 centers

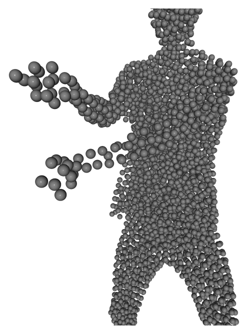

(b) Result with 1000 centers


(c) Visualization of 2000 centers


(d) Result with 2000 centers


(e) Visualization of 3000 centers


(f) Result with 3000 centers


(g) Visualization of 4000 centers

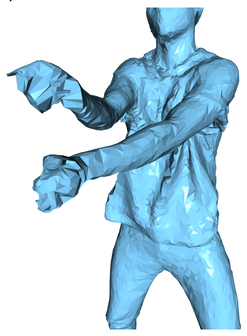

(h) Result with 4000 centers

Figure 6: Subjective evaluation with different numbers of centers. Volume centers are represented with black spheres, and their corresponding reconstructed meshes are drawn in blue.

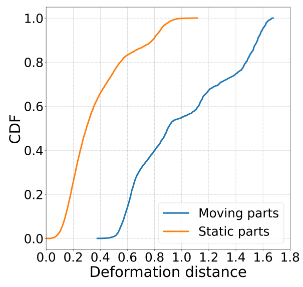
the same quality, respectively. For a bitrate larger than about 13 Mbps, Draco can achieve better D2-PSNR, however, reconstructed meshes with D2-PSNR larger than 80 are of high-enough quality. In terms of geometry compression, TVMC achieves a relatively high quality with much lower bitrate requirements, which is vital for 3D video streaming and storage. TVMC is a lossy compression method that improves compression ratio by using mesh decimation, which sacrifices some quality. Additionally, mesh deformation inevitably introduces some loss.

Note that distortions caused by self-contact regions cannot be directly reflected by evaluation metrics, since the errors are computed based on Euclidean distance between vertices. To assess these distortions, visual comparisons must be made subjectively. In Figure 5, we show the reconstructed meshes using Draco and TVMC at approximately equal bitrates. With a relatively low bitrate, about 5 Mbps to 6 Mbps, Draco causes obvious distortions on the parts of faces, hands, or feet, while TVMC performs much better. The detailed descriptions of corresponding objective evaluations and encoding and decoding time are presented in Table 1.

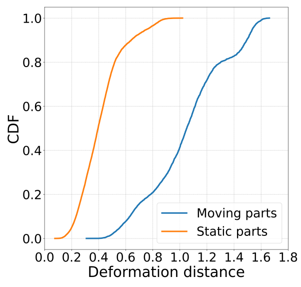
Table 1 also demonstrates that TVMC supports real-time applications by presenting its encoding and decoding time compared with Draco. TVMC requires, on average, 13.95 ms to decode a frame with precomputed and compressed displacements. In contrast, even under relatively ideal conditions, any compression method that decodes meshes through deformation may require over 100 ms per frame for decoding and reconstruction. From the experiments, we get results that TVMC can reduce decoding time by 66.1% compared to Draco. Although TVMC requires longer encoding times, these processes can be performed offline. Reducing the decoding time, on the other hand, is more critical to supporting real-time rendering and visualization, ultimately enhancing the user quality of the experience.

Figure 7 shows the Cumulative Distribution Functions (CDF) of the square root of the displacements' 2-norm, i.e., the deformation distance that each vertex of volume-tracked reference mesh moves. Ideally, only the vertices in the moving parts should shift, and the displacements of vertices in the static parts should be zero. However, due to a series of operations such as decimation, compression, and fitting, these values are not exactly zero but are very close. For "Thomas", in the static parts, over 85% of the deformation distances are less than or equal to 0.6, while in the moving parts, only about 10% of the deformation distances fall below 0.6. Thus, the information entropy is low, which benefits the compression process and helps TVMC achieve lower bitrates with relatively high quality.

4.3.2 Impact of the number of centers. We now investigate how the number of centers can affect TVMC's performance. As shown in Figure 6, increasing the number of centers from 1000 to 4000 results in more gray spheres (volume centers) representing the hands of the "Dancer". However, the quality of the reconstructed mesh does not always improve with a higher number of centers. In Figure 6a, only about five centers are used to represent each hand. This insufficient number of centers cannot adequately support stable center affinity deformation, leading to noticeable distortions after applying subdivision surface fitting. When the number of centers increases to 2000, the visual quality improves significantly



(a) CDF of "Basketball player"



(b) CDF of "Thomas"

Figure 7: Cumulative distribution function (CDF) of deformation distance for "Basketball player" and "Thomas". The blue lines represent moving parts, while the orange lines represent static parts. Moving parts refer to main regions that change when the volume-tracked reference mesh is deformed to one of the meshes in the group, whereas static parts remain relatively unchanged. The distances are adjusted for relative scaling, as different TVMs have varying scales.

compared to 1000 centers since the additional centers provide better support for representation and deformation of the hands.

From Figure 6f, we can see that the result with 3000 centers does not differ significantly from Figure 6d, however, the increase in the number of centers slightly reduces the quality of the left and right hands. This is because more centers may introduce more challenges for MDS in generating a stable set of reference centers. When the number of centers increases, MDS must consider a larger number of pairwise distances, resulting in higher computational cost and potential errors in the optimization process. Additionally, Time-varying meshes usually have complex or irregular geometry distributions, making it more challenging to find a solution that balances all pairwise distances without introducing artifacts. With an excessive number of centers, MDS-generated reference centers are more likely to contain abnormal outputs, significantly reducing TVMC's overall performance, as shown in Figure 6h. Since the center affinity deformation relies on the movements of the nearest centers and $K-1$ centers with the highest affinity, a large number of centers can, in fact, harm the quality of center affinity deformation.

It is important to note that the optimal number of centers is highly dependent on the particular characteristics of the TVM sequence. Fine-tuning experiments can help increase TVMC's performance.

4.3.3 Impact of Group of Frames (GoF). In Figure 8, we show the RD performance under different settings of GoF ranges in 5, 10, and 15. We demonstrate that the varying GoF will not negatively impact the performance of our method, there is no significant difference in D2-PSNR performance under different GoF settings, with only slight fluctuations. This highlights that our method avoids error accumulation when compressing a group of meshes, a common issue in many reference-based compression methods such as [17, 21]. In contrast, our method can reach a GoF of 15, while the methods above struggle to exceed a GoF of 5 without a rapid increase in distortion. This feature is provided by the volume-tracked reference mesh, created from reference space, representing a self-contact-free shape. We use volume-tracked reference mesh to replace traditional

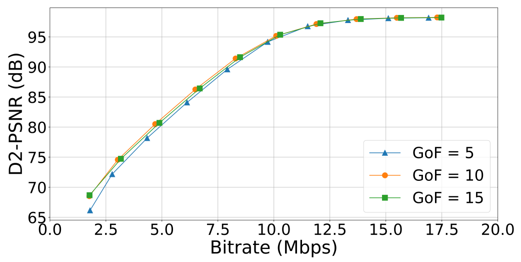


Figure 8: RD performance of "Dancer" under different GoFs of 5, 10, and 15.

reference mesh, cooperating with our center affinity deformation, avoiding the distortion accumulation problem efficiently.

Although TVMC efficiently addresses the issue of accumulated distortion, and can increase the GoF to 15 or more, using a GoF over 20 can lead to noticeable distortions. This is because deformations can cause deviations, and deformed frames for creating the volume-tracking reference mesh may not fully align as a cohesive surface, which degrades the quality of Poisson reconstruction, especially in narrow areas of the mesh, such as the wrists, lower legs and fingers. Since the subsequent displacement calculations and mesh decoding are based on the volume-tracked reference mesh, the overall performance of TVMC is decreased. Furthermore, as the GoF increase exceeds a certain limit, the meshes within the group exhibit greater shape variations. Deforming the reference mesh into a very different shape often results in noticeable distortions.

5 Conclusion

This paper proposes TVMC, a novel time-varying mesh compression approach that achieves high-efficiency mesh compression by utilizing volume centers and creating a self-contact-free volume-tracked reference mesh. With this self-contact-free volume-tracked reference mesh, TVMC enables low-bitrate mesh geometry compression while maintaining a high reconstruction quality. While TVMC utilizes the volume enclosed by the mesh surface, it does not strictly require fully watertight meshes. We evaluate TVMC on popular TVMs captured with real-world RGB-D sensors and demonstrate that it outperforms state-of-the-art methods such as Draco and V-DMC 4.0. Moreover, TVMC addresses the issue of accumulated distortions and increases the GoF size. Finally, TVMC enables real-time applications by achieving an average decoding time of less than 20 ms per frame. The main limitation of TVMC is that currently it only supports geometry information compression. Our future work includes upgrading TVMC to enable textured TVM compression.

Acknowledgments

This work was supported by Northeastern University, and the Czech Science Foundation under grants 23-04622L. We thank the anonymous reviewers and the shepherd Ketan Mayer-Patel for their thoughtful and constructive feedback.

References

[1] Rachida Amjoun and Wolfgang Straßer. 2009. Single-rate near lossless compression of animated geometry. *Computer-Aided Design* 41, 10 (2009), 711–718.

[2] Gavin Barill, Neil G Dickson, Ryan Schmidt, David IW Levin, and Alec Jacobson. 2018. Fast winding numbers for soups and clouds. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

[3] L. Boltzmann. 1868. *Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten: vorgelegt in der Sitzung am 8. October 1868.* k. und k. Hof- und Staatsdr. https://books.google.com/books?id=vxWqmgEACAAJ

[4] Ingwer Borg and Patrick JF Groenen. 2007. *Modern multidimensional scaling: Theory and applications.* Springer Science & Business Media.

[5] Mikaël Bourges-Sévenier. 2002. An introduction to MPEG-4 animation framework extension (AFX). In *Proceedings. International Conference on Image Processing*, Vol. 3. IEEE, III–III.

[6] Alvaro Collet, Ming Chuang, Pat Sweeney, Don Gillett, Dennis Evseev, David Calabrese, Hugues Hoppe, Adam Kirk, and Steve Sullivan. 2015. High-quality streamable free-viewpoint video. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–13.

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

[8] Jan Dvořák, Filip Hácha, and Libor Váša. 2023. Global optimisation for improved volume tracking of time-varying meshes. In *International Conference on Computational Science.* Springer, 113–127.

[9] Jan Dvořák, Zuzana Káčereková, Petr Vaněček, Lukáš Hruda, and Libor Váša. 2022. As-rigid-as-possible volume tracking for time-varying surfaces. *Computers & Graphics* 102 (2022), 329–338.

[10] Michael Garland and Paul S. Heckbert. 1997. Surface simplification using quadric error metrics. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '97).* ACM Press/Addison-Wesley Publishing Co., USA, 209–216. https://doi.org/10.1145/258734.258849

[11] Google. 2018. Draco. https://google.github.io/draco/.

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

[13] Himanshu Gupta, Max Curran, Jon Longtin, Torin Rockwell, Kai Zheng, and Mallesham Dasari. 2022. Cyclops: An fso-based wireless link for vr headsets. In *Proceedings of the ACM SIGCOMM 2022 Conference.* 601–614.

[14] Filip Hácha, Jan Dvořák, Zuzana Káčereková, and Libor Váša. 2024. Editing mesh sequences with varying connectivity. *Computers & Graphics* 121 (2024), 103943.

[15] Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2007. Time-varying mesh compression using an extended block matching algorithm. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 11 (2007), 1506–1518.

[16] Seung-Ryong Han, Toshihiko Yamasaki, and Kiyoharu Aizawa. 2008. Geometry compression for time-varying meshes using coarse and fine levels of quantization and run-length encoding. In *2008 15th IEEE International Conference on Image Processing.* IEEE, 1045–1048.

[17] Huong Hoang, Kunyao Chen, Truong Nguyen, and Pamela Cosman. 2023. Embedded Deformation-based Compression for Human 3D Dynamic Meshes with Changing Topology. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 2252–2262.

[18] Junhui Hou, Lap-Pui Chau, Ying He, and Nadia Magnenat-Thalmann. 2014. A novel compression framework for 3D time-varying meshes. In *2014 IEEE International Symposium on Circuits and Systems (ISCAS).* IEEE, 2161–2164.

[19] Junhui Hou, Lap-Pui Chau, Minqi Zhang, Nadia Magnenat-Thalmann, and Ying He. 2014. A highly efficient compression framework for time-varying 3-D facial expressions. *IEEE Transactions on Circuits and Systems for Video Technology* 24, 9 (2014), 1541–1553.

[20] Tao Jin, Mallesham Dasa, Connor Smith, Kittipat Apicharttrisorn, Srinivasan Seshan, and Anthony Rowe. 2024. Meshreduce: Scalable and bandwidth efficient 3d scene capture. In *2024 IEEE Conference Virtual Reality and 3D User Interfaces (VR).* IEEE, 20–30.

[21] Xudong Jin, Jianfeng Xu, and Kei Kawamura. 2024. Embedded Graph Representation for Inter-Frame Coding of Dynamic Meshes. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 4070–4074.

[22] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, Vol. 7.

[23] Hao Li, Bart Adams, Leonidas J Guibas, and Mark Pauly. 2009. Robust single-view geometry and motion reconstruction. *ACM Transactions on Graphics (ToG)* 28, 5 (2009), 1–10.

[24] Guoliang Luo, Zhigang Deng, Xin Zhao, Xiaogang Jin, Wei Zeng, Wenqiang Xie, and Hyewon Seo. 2020. Spatio-temporal segmentation based adaptive compression of dynamic mesh sequences. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 16, 1 (2020), 1–24.

[25] Adrien Maglo, Guillaume Lavoué, Florent Dupont, and Céline Hudelot. 2015. 3d mesh compression: Survey, comparisons, and emerging trends. *ACM Computing Surveys (CSUR)* 47, 3 (2015), 1–41.

[26] Khaled Mammou, Jungsun Kim, Alexis M. Tourapis, Dimitri Podborski, and David Flynn. 2022. Video and Subdivision based Mesh Coding. In *2022 10th European Workshop on Visual Information Processing (EUVIP).* 1–6. https://doi.org/10.1109/EUVIP53989.2022.9922888

[27] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. 2006. A skinning approach for dynamic 3D mesh compression. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 337–346.

[28] Khaled Mamou, Titus Zaharia, and Françoise Prêteux. 2009. TFAN: A low complexity 3D mesh compression algorithm. *Computer Animation and Virtual Worlds* 20, 2-3 (2009), 343–354.

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

[30] MPEG 2021. *Cfp for dynamic mesh coding.* MPEG. Online, ISO/IEC JTC1/SC29/WG7/N00231.

[31] MPEG. 2022. Mpeg-pcc-tmc13. https://github.com/MPEGGroup/mpeg-pcc-tmc13.

[32] MPEG 2023. *V-DMC TMM 4.0.* MPEG. ISO/IEC JTC1/SC29/WG7/N00595.

[33] MPEG 3DG 2019. *Common test conditions for point cloud compression.* MPEG 3DG. Geneva, Switzerland, ISO/IEC JTC1/SC29/WG11 N18474.

[34] Jingliang Peng, Chang-Su Kim, and C-C Jay Kuo. 2005. Technologies for 3D mesh compression: A survey. *Journal of visual communication and image representation* 16, 6 (2005), 688–733.

[35] Jrg Peters and Ulrich Reif. 2008. *Subdivision Surfaces* (1st ed.). Springer Publishing Company, Incorporated.

[36] Jarek Rossignac. 1999. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics* 5, 1 (1999), 47–61.

[37] Ioan Alexandru Salomie, Adrian Munteanu, Augustin Gavrilescu, Gauthier Lafruit, Peter Schelkens, Rudi Deklerck, and Jan Cornelis. 2004. MESHGRID-A compact, multiscalable and animation-friendly surface representation. *IEEE transactions on circuits and systems for video technology* 14, 7 (2004), 950–966.

[38] Robert W Sumner, Johannes Schmid, and Mark Pauly. 2007. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers.* 80–es.

[39] Libor Vasa and Václav Skala. 2007. Coddyac: Connectivity driven dynamic mesh compression. In *2007 3DTV Conference.* IEEE, 1–4.

[40] Toshihiko Yamasaki and Kiyoharu Aizawa. 2010. Patch-based compression for time-varying meshes. In *2010 IEEE International conference on image processing.* IEEE, 3433–3436.

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

[42] Qi Yang, Joël Jung, Timon Deschamps, Xiaozhong Xu, and Shan Liu. 2023. Tdmd: A database for dynamic color mesh subjective and objective quality explorations. *arXiv preprint arXiv:2308.01499* (2023).

[43] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. 2018. Open3D: A Modern Library for 3D Data Processing. *arXiv:1801.09847* (2018).