TSMC: Time-varying 4D Scene Mesh Compression

GUODONG CHEN, Northeastern University, USA
LIBOR VÁŠA, University of West Bohemia, Czech Republic
AMRITA MAZUMDAR, NVIDIA, USA
MALLESHAM DASARI, Northeastern University, USA

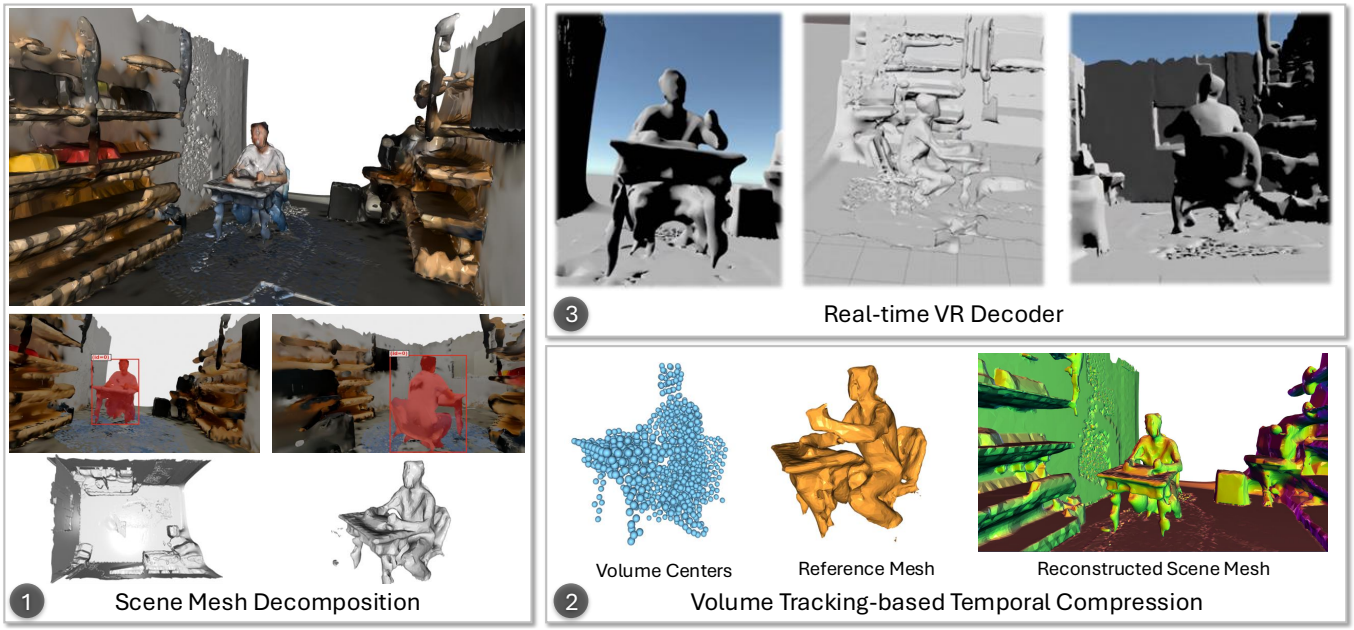


Fig. 1. We present TSMC, a novel compression method for time-varying 4D unbounded scene meshes. TSMC separates the scene into dynamic objects and static backgrounds, then efficiently compresses dynamic parts using reference meshes with displacement fields by generating a volume-tracked reference mesh for dynamic parts and encoding displacements. TSMC decoders can run on VR headsets such as Meta Quest 3 and achieve real-time decoding at 24 FPS.

Time-varying scene meshes are a widely used representation for volumetric video, offering immersive six degrees of freedom (6DoF) interaction in virtual environments. However, their significantly larger size presents major challenges for efficient storage, Internet transmission, and real-time streaming. Existing mesh compression standards are not well-suited for complex scene meshes with dynamic content, and no standardized encoding techniques have been widely adopted for this class of data. To address this gap, we propose TSMC, a novel compression framework for 3D time-varying scene mesh sequences. TSMC uses voxel-based analysis to establish temporally stable correspondences, segment static and dynamic regions, and extract volume-tracked reference meshes for dynamic content to support inter-frame prediction. Static backgrounds and reference meshes are compressed using Google Draco, while dynamic displacements are encoded using a combination of the Karhunen-Loève Transform (KLT) and Laplacian coordinates. Extensive experiments demonstrate that at 30 fps and an SSIM of 0.9, TSMC reduces bandwidth usage and decoding time by up to 56.54% and 85.42%, respectively, compared to state-of-the-art mesh compression methods. Our code and dataset are available at https://github.com/SINRG-Lab/TSMC.

CCS Concepts: • **Computing methodologies** → **Mesh geometry models**.

Additional Key Words and Phrases: Time-varying Mesh, Scene Mesh, Volumetric Video, Volume Tracking, Mesh Compression.

ACM Reference Format:
Guodong Chen, Libor Váša, Amrita Mazumdar, and Mallesham Dasari. 2026. TSMC: Time-varying 4D Scene Mesh Compression. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '26), July 19–23, 2026, Los Angeles, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3799902.3811079

Authors' Contact Information: Guodong Chen, Northeastern University, Boston, MA, USA, chen.guod@northeastern.edu; Libor Váša, University of West Bohemia, Plzeň, Czech Republic, lvasa@kiv.zcu.cz; Amrita Mazumdar, NVIDIA, New York City, New York, USA, amritam@nvidia.com; Mallesham Dasari, Northeastern University, Boston, MA, USA, m.dasari@northeastern.edu.

SIGGRAPH Conference Papers '26, Los Angeles, CA, USA

ACM ISBN 979-8-4007-2554-8/2026/07
https://doi.org/10.1145/3799902.3811079

1 INTRODUCTION

Free-viewpoint 4D mesh scene representation, generated from depth sensors, is emerging as a foundational media format, which offers highly realistic and immersive AR/VR experiences with six degree-of-freedom (6-DoF) interaction [Jin et al. 2024a; Tang et al. 2018].

Compared to alternative representations such as Gaussian splatting [Kerbl et al. 2023] or NeRF [Mildenhall et al. 2021], meshes provide an explicit and structured representation of scenes, offering accurate geometry, well-defined topology, and clear object boundaries. These properties make mesh-based representations particularly well-suited for applications such as autonomous driving [Abbasi et al. 2022], surgical environments [Özsoy et al. 2022], and industrial automation [Agapaki and Brilakis 2021]. Moreover, this structural advantage has recently motivated hybrid 3DGS approaches that integrate mesh priors into Gaussian splatting frameworks [Guédon and Lepetit 2024; Sun et al. 2026; Waczyńska et al. 2024; Zheng et al. 2025], further highlighting the importance of meshes as a core representation for dynamic 3D scenes.

Unlike traditional static or animated 3D meshes, 4D scene meshes reconstructed from depth measurements consist of temporally evolving mesh frames with changing topology, varying vertex counts, and dynamic face connectivity [Jin et al. 2024a; Tang et al. 2018]. These sequences often represent *unbounded* environments rather than isolated objects, combining largely static background geometry with multiple independently moving foreground objects. While the explicit mesh structure enables robust collision detection and object-level reasoning, it also introduces substantial computational complexity. Sensor noise and frame-to-frame reconstruction variability further exacerbate temporal inconsistency, making effective compression and redundancy removal in large-scale time-varying 4D scene meshes a nontrivial challenge.

Existing time-varying mesh compression methods, such as [Ren et al. 2025; Tang et al. 2018] employ an implicit Truncated Signed Distance Function (TSDF) representation; [Collet et al. 2015] uses nonrigid transformations; and [Hoang et al. 2023] applies embedded deformation. However, these methods fundamentally fail to compress large-scale, *unbounded scenes* effectively. They typically rely on deforming a selected keyframe to match subsequent frames with similar shapes while maintaining topology and connectivity. This approach proves inadequate for complex scenes, as it struggles with self-contact regions, leading to severe visual artifacts and inaccuracies in the decoded meshes. Furthermore, deformation-based strategies introduce additional challenges, where the movement of a single object can propagate distortions for the entire scene by affecting all vertices in the mesh.

Neural progressive meshes [Chen et al. 2023] learn a shared generative space of mesh detail and enable progressive transmission for compact shape encoding. However, this method is also designed for individual 3D shapes or bounded deforming objects and relies on learned subdivision refinements, which fundamentally limit their applicability to large-scale, unbounded 4D scene meshes, where topology, connectivity, and object composition vary over time.

Recent advances in volume tracking [Dvořák et al. 2023, 2022; Hácha et al. 2024] have demonstrated the potential to take advantage of the volume enclosed by meshes to establish stable correspondences between frames, offering a promising direction for addressing these challenges, and have been successfully used to compress time-varying *object* meshes in [Chen et al. 2025].

Building on this insight and the inherent properties of spatial and temporal correlations in large-scale scenes, we propose TSMC, a novel time-varying mesh compression method designed for 4D unbounded scenes. TSMC employs a region identification algorithm based on SAM3 [Carion et al. 2025] to effectively differentiate dynamic regions from static backgrounds. For dynamic regions, it constructs a volume-tracked reference mesh to resolve self-contact issues, which is then deformed to approximate each dynamic part of the frame and compute displacement fields. It then computes displacement fields by tracking vertex movements between the reference mesh and subsequent frames, ensuring accurate motion representation while preserving temporal coherence. TSMC processes the static background only once for a group of frames and encodes dynamic parts into one reference mesh and displacement fields. The displacement fields are compressed using Karhunen-Loève Transform and Laplacian coordinates [Váša et al. 2014; Váša and Petřík 2011], significantly reducing the data size while retaining high visual quality.

With extensive experiments, we show that TSMC outperforms the state-of-the-art scene compression methods, including Draco [Draco 2018], a scene mesh extended version of TVMC [Chen et al. 2025] denoted as TVMC*, a TSDF-based compression method [Tang et al. 2018] denoted as KLT, and the neural-based compression method NeCGS [Ren et al. 2025], in terms of scene compression ratio and decoding time. We highlight TSMC's performance from our evaluation as follows:

- TSMC achieves a compression ratio of around 200× to 240× at 30 FPS, with an SSIM of around 0.85 to 0.95, depending on the complexity of the scene mesh sequence. Overall, this is a 67.63%, 34.43%, 75.03%, 60.39% reduction in data compared to Draco [Draco 2018], TVMC* [Chen et al. 2025], KLT [Tang et al. 2018], and NeCGS [Ren et al. 2025], respectively.
- TSMC supports real-time applications, and can achieve an overall fps around 30 for scene mesh sequences.
- TSMC is the first method to enable compression specifically for full scene meshes, a capability not supported by prior approaches such as V-DMC 4.0 [MPEG 2023] and recent embedded deformation-based techniques [Chen et al. 2025; Jin et al. 2024b].

2 BACKGROUND AND RELATED WORK

2.1 Static Mesh Compression

Static mesh compression utilizes intra-frame redundancy to compress each mesh independently. Draco [Draco 2018], a state-of-the-art mesh compression tool, uses Edgebreaker [Rossignac 1999] algorithm that takes an explicit model (i.e., a triangle mesh) and compresses vertices' positions, connectivity, and other attributes. Another notable work, the TFAN algorithm [Mamou et al. 2009], organizes triangles into fans around shared vertices to reduce connectivity redundancy, particularly for manifold meshes. However, like Draco, TFAN operates purely on individual frames and lacks inter-frame prediction capabilities. A recent implicit approach [Tang et al. 2018] uses Karhunen-Loève Transform (KLT) to learn a global mean and principal components offline. However, the learned principal components do not generalize well across different objects in scenes. Deep learning methods like NeCGS [Ren et al. 2025] converts meshes into an improved version of TSDF, called TSDF-Def, to ensure the uniformity and low resolution of TSDF tensors and

Fig. 2. Post-deformation artifacts in self-contact regions, such as when arms rest on a desk or when drinking from a bottle. In these scenarios, the areas of self-contact may remain unnaturally stuck together after deformation.

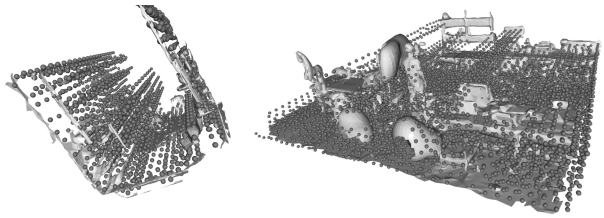


Fig. 3. Volume tracking results on the entire "Arena" scene mesh. Gray dots represent the estimated volume centers, and the gray surface represents the scene mesh. Directly applying volume tracking to the scene mesh fails, as the reference centers cannot correctly represent the region enclosed by the mesh surface due to the winding number issue.

use neural networks to compress them. However, NeCGS is also designed for static object compression.

2.2 Dynamic Mesh Compression

Dynamic mesh compression encodes a sequence of meshes by storing a base mesh and a displacement field, which is feasible because dynamic meshes have a constant topology.

In 2021, the MPEG 3DG group issued a Call for Proposals on dynamic mesh coding [MPEG 2021], which led to the development of the V-DMC standard. Apple's VSMC [Mammou et al. 2022] was then selected as the foundation for V-DMC. It represents dynamic meshes through a decimated base mesh and displacement vectors derived from subdivision surface fitting. 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. However, V-DMC depends on a consistent topology across the input mesh sequence. While VSMC [Mammou et al. 2022] introduces a time-consistent re-meshing method to remove these constraints, its instability makes the representation of inter-frame differences between time-varying meshes challenging.

2.3 Time-varying Mesh Compression

Time-varying mesh compression, on the other hand, deals with mesh sequences in which both the geometry and topology vary over time (typically captured by the real-world depth sensors), making it a more complex problem to solve. A common method in the literature to reduce complexity is to transform or relax time-varying meshes into dynamic meshes through deformation.

Early works [Yang et al. 2005] used the first mesh frame in a sequence as a reference mesh to estimate motion and map onto subsequent meshes. However, these methods often accumulate errors over time, leading to significant artifacts and distortions. To mitigate error accumulation, [Collet et al. 2015] proposed selecting keyframes based on factors such as surface area, genus, and connected components to better represent the sequence while addressing self-contact challenges. Despite improvements, they used the non-rigid ICP [Li et al. 2009] for mesh registration, which relies only on vertex information, leading to high fitting errors and high computational costs due to iterative optimizations.

In contrast to methods that directly apply ICP registration [Collet et al. 2015; Doumanoglou et al. 2014], many works proposed custom dynamic registration algorithms along with deformation methods such as embedded deformation [Sumner et al. 2007] to deform selected reference meshes while preserving topology. [Marvie et al. 2022] first decimated all time-varying meshes and then used Dijkstra's algorithm [Dijkstra 2022] to generate approximately uniformly distributed seeds for better tracking. Hoang et al. [Hoang et al. 2023] first used the intrinsic shape signatures (ISS) keypoints method [Zhong 2009], then introduced an optimization-based surface mapping technique to determine the correspondence between meshes for deformation, while Jin et al. [Jin et al. 2024b] developed an embedded graph representation to capture differences between meshes. However, these methods rely solely on geometric information, which can lead to significant distortions when self-contact regions are present in the reference mesh, as illustrated in Figure 2. Additionally, occlusion issues, such as parts of the mesh becoming invisible in certain frames due to contact (e.g., feet touching the floor), introduce further inaccuracies and distortions.

To address these problems, Chen et al. proposed TVMC [Chen et al. 2025], which employs volume tracking and center affinity deformation to compress a group of time-varying meshes into a single volume-tracked reference mesh along with corresponding displacement fields. However, TVMC is not directly applicable to time-varying scene meshes due to limitations in the in-out test based on winding number [Barill et al. 2018], which is unreliable for complex or open geometries often found in large scenes, as shown in Figure 3. Furthermore, TVMC encodes the displacement fields directly using Draco, thereby ignoring temporal redundancy and structural similarities across displacements induced by smooth and continuous motion.

3 TSMC

Figure 4 provides an overview of the TSMC pipeline. The core idea behind TSMC is to exploit temporal redundancy across adjacent mesh frames by identifying and leveraging stable reference structures called volume centers (§ 3.1). To manage the complexity of large, unbounded scenes, TSMC first separates static backgrounds from dynamic foregrounds using a region differentiation algorithm based on SAM3 [Carion et al. 2025] (§ 3.2). For dynamic regions, TSMC constructs a volume-tracked reference mesh that avoids self-contact artifacts and serves as a deformation anchor. Displacement fields are then computed by tracking vertex-level deviations between this reference and each frame, and are compactly encoded

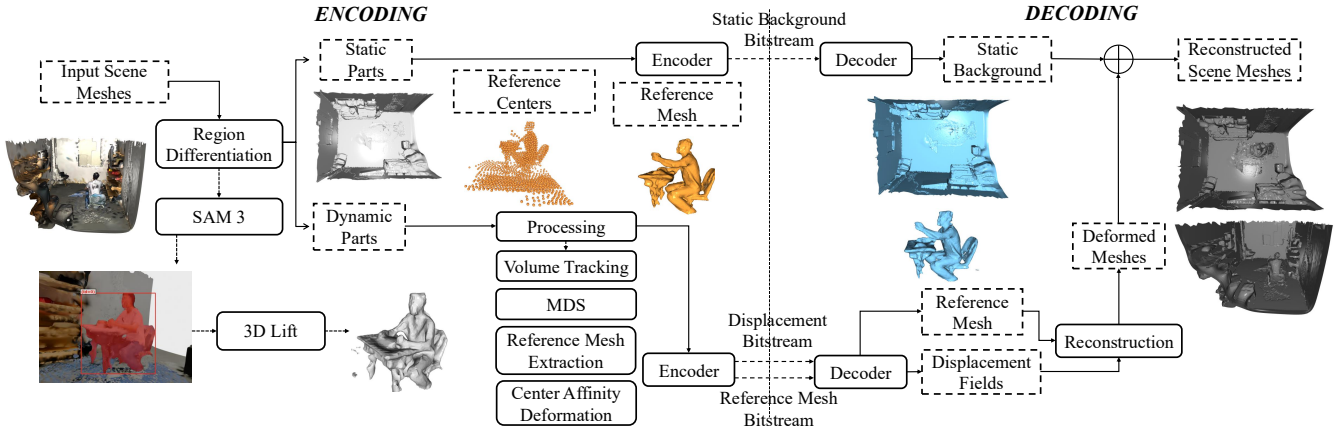


Fig. 4. Simplified workflow of TSMC. The process consists of several key stages, with major operations enclosed in solid boxes. Dotted boxes represent the inputs and outputs of specific processing steps. The workflow begins with the input scene meshes, shown in their original colors, which are then divided into static and dynamic parts through region differentiation. The dynamic parts are utilized to generate a set of reference centers (orange) and a reference mesh (orange). The Static parts are encoded into the Static Background Bitstream, while the reference centers and reference mesh undergo processing to generate displacement fields, which are then encoded into the Displacement Bitstream using KLT and Laplacian coordinates. During decoding, the static background and reference mesh are first decoded. The reference mesh is then deformed into various shapes using displacement fields and subsequently combined with static parts to produce the final reconstructed meshes.

using a hybrid of the Karhunen-Loève Transform and Laplacian coordinates. This results in high compression ratios while preserving geometric fidelity and temporal coherence (§ 3.3). The following subsections detail each of these components in depth.

3.1 As-Rigid-As-Possible Volume Tracking

Consider a time-varying mesh 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, edges, and faces (triangles), respectively. To obtain stable correspondence between adjacent meshes, we apply volume tracking [Dvořák et al. 2023, 2022] to S to extract representative volume centers $C = \{\mathbf{C}_k\}_{k=1}^{K}$, where each trajectory $\mathbf{C}_k = \{c_k^t\}_{t=0}^{N-1}$, with $c_k^t \in \mathbb{R}^3$, traces the spatial trajectory of a localized volume across all N frames. Here, K denotes the total number of tracked centers. Regarding how volume tracking works, we refer readers to the supplementary material.

3.2 Static and Dynamic Scene Decomposition

Applying the above volume tracking directly to full scene meshes is computationally inefficient and often inaccurate, as shown in Figure 3. To address this, TSMC first performs a robust decomposition into static and dynamic regions, targeting only the dynamic components for temporal modeling. Static parts, such as walls, floors, and other background structures, can be encoded once or periodically for a group of frames, while dynamic parts, corresponding to moving objects, are encoded with frame-specific motion compensation. This separation not only reduces redundancy but also improves the effectiveness of temporal compression strategies.

Given a time-varying mesh sequence S, our goal is to decompose each mesh M_t into static and dynamic parts in a temporally consistent manner. For each mesh M_t, we render the surface from a fixed set of X viewpoints $\mathcal{V} = \{v_1, \ldots, v_X\}$, shared across all frames. This produces a set of rendered images $I_{t,x}$.

We identify dynamic regions using a two-stage SAM3-based tracking strategy. First, we select a reference viewpoint v_r and apply SAM3 [Carion et al. 2025] dense tracking to the image sequence $\{I_{t,r}\}_{t=0}^{N-1}$ to initialize dynamic object hypotheses based on temporal mask variations. Next, for each frame t, we treat the X rendered views $\{I_{t,x}\}_{x=1}^{X}$ as a multi-view video and apply SAM3 tracking again to obtain per-view masks: $\mathcal{M}_t = \{m_{t,1}, \ldots, m_{t,X}\}$, where $m_{t,x} \in \{0,1\}^{H\times W}$ indicates dynamic pixels under viewpoint v_x.

To lift 2D dynamic masks to the mesh surface, we compute per-pixel triangle visibility via ray casting using the Open3D [Zhou et al. 2018]. For each viewpoint v_x, we generate a primitive-ID image:

$$P_{t,x} \in \{-1, 0, \ldots, |F_t| - 1\}^{H\times W}, \tag{1}$$

where each pixel stores the index of the visible triangle or -1 if the ray misses the marked surface.

For each face $f \in F_t$, we accumulate dynamic votes and visibility counts across all views as follows:

$$V_t(f) = \sum_{x=1}^{X} \sum_{p\in\Omega} \mathbf{1}\left[P_{t,x}(p) = f \ \wedge \ m_{t,x}(p) = 1\right], \tag{2}$$

$$U_t(f) = \sum_{x=1}^{X} \sum_{p\in\Omega} \mathbf{1}\left[P_{t,x}(p) = f\right], \tag{3}$$

where Ω denotes the image domain. We then define a normalized dynamic score for each face as:

$$R_t(f) = \frac{V_t(f)}{\max(U_t(f), 1)}. \tag{4}$$

A face is classified as dynamic if $R_t(f) \geq \tau$, where τ is a predefined threshold. Finally, we construct two sub-meshes by grouping faces according to this partition. Through this SAM3-based scene

decomposition process, TSMC achieves robust and precise decomposition of each mesh into static and dynamic regions. Static parts are encoded once and reused across frames, while dynamic parts are compressed using the techniques described in § 3.3.

3.3 Dynamic Parts Compression

Once static and dynamic regions have been identified, TSMC treats the dynamic parts as a new time-varying mesh sequence. As we mentioned in § 2, a key challenge in compressing dynamic meshes is handling self-contact regions where different parts of a mesh come into close proximity or collide, such as limbs touching the body or feet contacting the ground. Such cases often degrade the quality of embedded deformation methods. A recent method, TVMC [Chen et al. 2025] addressed this at the object level by constructing a self-contact-free reference mesh, but it does not scale to complex scene meshes. Building on this, TSMC first generates a self-contact-free reference mesh tailored to the dynamic regions.

3.3.1 Reference centers generation. To construct a self-contact-free reference mesh for dynamic parts, TSMC leverages the volume centers obtained from volume tracking (§ 3.1) as implicit guidance. These centers represent small tracked volumes whose trajectories span the mesh sequence, providing a robust basis for geometric correspondence over time. To extract a stable reference surface, we aim to reposition these volume centers into a canonical 3D configuration that best reflects their long-term spatial relationships. To do that, we first build a distance matrix from the largest distances between centers. Let d_{ij} be the largest Euclidean distance between two centers c_i and c_j over the entire sequence. The largest distance matrix D is 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 $r_1, r_2, \ldots, r_K$ in a 3-dimensional space that minimizes the stress function given by:

$$\min \sigma(R) = \sqrt{\frac{\sum_{i<j} \left(d_{ij} - \|r_i - r_j\|\right)^2}{\sum_{i<j} (d_{ij})^2}}. \tag{5}$$

Here, $R = [r_1, r_2, \ldots, r_K]$ 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.

We also define a_{ij} as the *affinity* of these centers, which is a function of the maximum distance and reflects how strongly the two centers are connected. The affinity a_{ij} is given by:

$$a_{ij} = \exp(-(\sigma d_{ij})^2), \tag{6}$$

where σ is a scaling factor determined by the global scale of the input sequence, which is in the range from 10^{-2} to 10^{1}. This affinity encodes long-term structural proximity and is later used in the deformation process to preserve coherent geometry.

3.3.2 Displacement fields calculation. After getting the reference centers from MDS, TSMC then extracts the self-contact-free reference mesh specifically for dynamic parts based on that. The way to do that is to deform dynamic parts from multiple meshes in the group to the shape represented by reference centers, then employ Poisson surface reconstruction [Kazhdan et al. 2006] based solely on the dense set of vertices. Many deformation algorithms can be applied here, such as As-Rigid-As-Possible deformation [Sorkine and Alexa 2007] and embedded deformation [Hoang et al. 2023; Jin et al. 2024b]. While As-Rigid-As-Possible deformation and embedded deformation utilize some localized feature, they update a vertex's position entirely depending on the spatial Euclidean distance. To prevent the influence of neighbors that are close in terms of Euclidean distance but belong to different parts of the mesh, TSMC uses the center affinity deformation, drawing inspiration from [Chen et al. 2025; Hácha et al. 2024].

Center-affinity deformation uses dual quaternions to represent rigid transformations due to their efficient blending and inversion properties. For each vertex v_i^t at frame t, we first identify its nearest center in terms of Euclidean distance:

$$k_1 = \arg\min_k \|v_i^t - c_k^t\|, \tag{7}$$

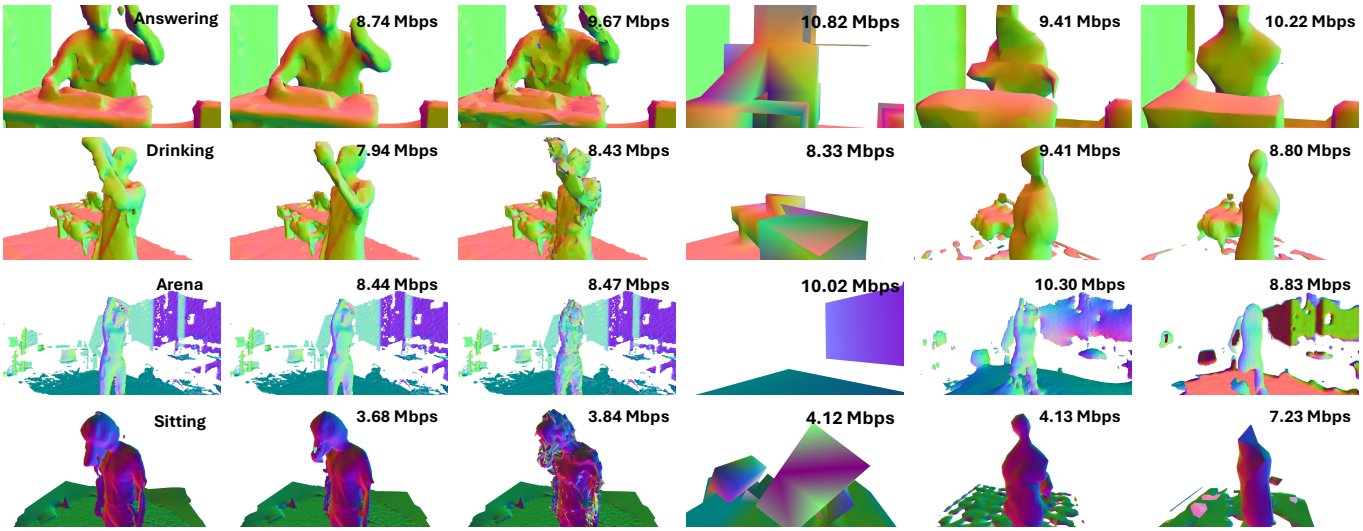
and denote the corresponding center as $c_{k_1}^t$.

We then select a set of neighboring centers $\mathcal{N}(k_1)$ consisting of the top-K_{nbr} centers with the highest affinity $a_{k_1 k}$ to $c_{k_1}^t$, rather than using Euclidean proximity. These centers are used to construct the transformation for vertex deformation. As a result, the deformed mesh will significantly align with the transformation of its centers, while maintaining its topology.

Although our center-affinity deformation method can deform meshes with different shapes to the same reference space with high accuracy, large motion differences can bring excessive distortions. Therefore, TSMC divides the sequence into groups of GoF meshes, where GoF denotes the group of frames size. Within each group, we apply center affinity deformation to obtain a dense vertex set. Quadric Error Metric (QEM) decimation [Garland and Heckbert 1997] and subdivision schemes in [Peters and Reif 2008] can be used for surface fitting to reduce distortion. Here, we adopt the mid-point subdivision scheme for the entire process as it is computationally efficient and preserves the original mesh's topology, which is vital for stable and high-speed decoding.

The dense vertex set is fed into the Poisson reconstruction algorithm [Kazhdan et al. 2006] implemented by Open3D [Zhou et al. 2018] (with a setting of $depth = 9$) to extract a single surface. After adopting QEM decimation again, we call this simplified mesh volume-tracked reference mesh, which contains complete components and surfaces to approximate each of the meshes with the help of center affinity deformation. Then TSMC can deform it to a geometric approximation M_t' of the mesh M_t using the positions of reference centers and the centers of the mesh M_t. Since the deformed meshes share a constant topology, each frame can be efficiently represented using a displacement field and the reference mesh.

3.3.3 Karhunen-Loève Transform and Laplacian coordinates compression. So far, we have extracted the redundancy between time-varying meshes and represented them as a single reference mesh along with displacement fields. Considering the fact that for time-varying meshes the displacements represent a series of continuous movements, we can do the second-order trajectory tracking for displacements to further increase the compression performance. To extract the similarity and geometry structure of displacement fields, we create the displacement vector as $\mathbf{d}_i = (\text{dis}_i^0, \text{dis}_i^1, \ldots, \text{dis}_i^{GoF-1}) \in \mathbb{R}^{3\cdot GoF}$, where dis_i^t represents the (x, y, z) displacement coordinates



(a) Ground Truth (b) TSMC (Ours) (c) TVMC* (d) Draco (e) KLT (f) NeCGS

Fig. 5. Overall compression efficiency of TSMC compared to several baselines across multiple scenes and operating bitrates, demonstrating consistently higher visual quality at equal or lower data rates. Colors correspond to normalized vertex normals mapped to $[0, 1]$, making geometric distortions and self-contact artifacts clearly visible on non-textured meshes. Compared to TVMC*, Draco, KLT, and NeCGS, TSMC preserves fine motion and surface details with fewer artifacts, especially in self-contact regions, while achieving comparable or lower bitrates.

for the i-th vertex at the time t, and use Karhunen-Loève Transform (KLT) [Jorgensen and Song 2007] to reduce their dimensionality.

Using the collection $\{\mathbf{d}_i \in \mathbb{R}^{3\cdot GoF}\}_{i=1}^{n}$ of n vertices as training signals, we learn a linear transformation matrix $\mathbf{P}$ and a mean vector $\boldsymbol{\mu}$ that define the forward and inverse KLT transforms for the displacement vectors. Let $\mathbf{D} \in \mathbb{R}^{n\times 3\cdot GoF}$ denote the matrix formed by stacking $\mathbf{d}_i$ row-wise. This PCA-based method allows us to retain only the first m principal components (eigenvectors) for lossy encoding and reconstruction. For each displacement vector $\mathbf{d}_i$, the forward KLT transform and its inverse are given by:

$$\hat{\mathbf{z}}_i = (\mathbf{P}\mathbf{I}_m)^T(\mathbf{d}_i - \boldsymbol{\mu}), \tag{8}$$

$$\hat{\mathbf{d}}_i = \mathbf{P}\mathbf{I}_m\hat{\mathbf{z}}_i + \boldsymbol{\mu}. \tag{9}$$

Here, $\mathbf{I}_m$ is a diagonal matrix that retains only the first m principal components. In this way, the collection can be compactly represented as a matrix $\mathbf{G} = (g_1, g_2, \ldots, g_n) \in \mathbb{R}^{n\times m}$, where each

$$g_i = (\mathbf{P}\mathbf{I}_m)^T(\mathbf{d}_i - \boldsymbol{\mu}) \in \mathbb{R}^m. \tag{10}$$

Next, we create a discrete geometric Laplace operator [Váša et al. 2014] to further compress the matrix $\mathbf{G}$. The geometric Laplacian $\mathbf{L} \in \mathbb{R}^{n\times n}$ is defined as follows:

$$\mathbf{L}_{ii} = 1, \quad \mathbf{L}_{ij} = -\frac{w_{ij}}{\sum_{k\neq i} w_{ik}}, \quad i \neq j, \tag{11}$$

where w_{ij} are weights computed using the MV formula [Floater 2003]. Note that the geometric Laplacian $\mathbf{L}$ is computed using the *decoded* reference mesh, ensuring that both the encoder and decoder operate on the same set of values so that the correct Laplacian $\mathbf{L}$ can be computed on the client side without requiring additional streaming. We employ a GPU to accelerate its calculation and optimize the time within 10 to 20 ms per frame. l anchor points are added in $\mathbf{L}$ to eliminate ambiguities and improve stability for reconstruction. The resulting extended Laplacian $\mathbf{L}^* \in \mathbb{R}^{(n+l)\times n}$ has full column rank and thus can be reconstructed via least-squares. Using this extended Laplacian $\mathbf{L}^*$, we can convert $\mathbf{G}$ into matrix $\mathbf{Y} \in \mathbb{R}^{(n+l)\times m}$ with much smaller entropy:

$$\mathbf{Y} = \mathbf{L}^*\mathbf{G}. \tag{12}$$

The values in the matrix $\mathbf{Y}$ are then uniformly quantized and encoded for streaming. On the decoder side, the matrix $\mathbf{G}$ is decoded by solving the sparse least squares problem:

$$\bar{\mathbf{G}} = ((\mathbf{L}^*)^T\mathbf{L}^*)^{-1}(\mathbf{L}^*)^T\bar{\mathbf{Y}}, \tag{13}$$

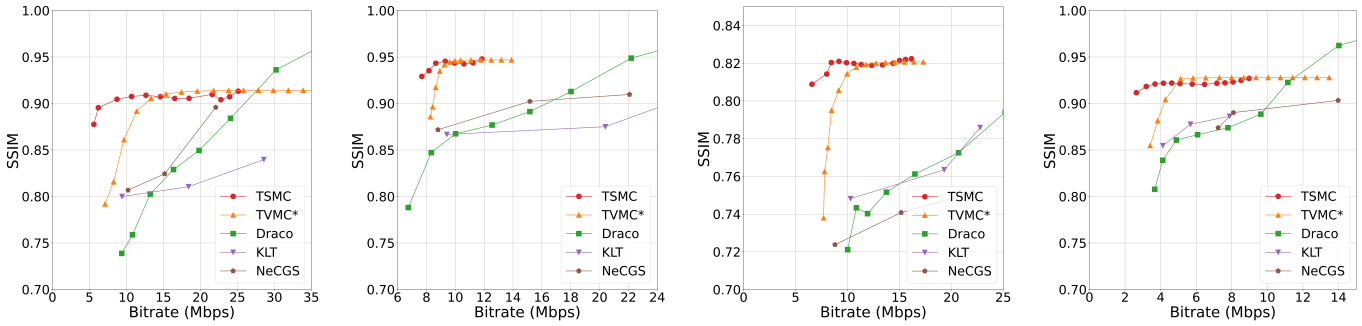
where $\bar{\mathbf{Y}}$ is the decoded $\mathbf{Y}$ at the client side.

For reference mesh compression, we observed that intra-frame compression using Draco [Draco 2018] consistently outperformed KLT-based methods in terms of decoding time, particularly for full, unbounded scene meshes. Consequently, we use Draco with high-quality settings ($qp = 14$, $cl = 7$) to encode the volume-tracked reference mesh. For temporal compression, we apply our custom KLT+Laplacian pipeline to encode the displacement fields.

4 EVALUATION

4.1 Dataset and metrics

To evaluate on a broader range of time-varying scene mesh sequences, we use high-resolution 4D scenes from [Kim et al. 2026]. This dataset is captured using a setup with four Azure Kinect cameras [Microsoft 2022], recording at 640x576 depth resolution and 30 FPS. We use Open3D [Zhou et al. 2018] to convert RGB-D images into TSDF volumes and then extract meshes via the Marching Cubes [Lorensen and Cline 1998]. We named these datasets as *Arena*



(a) RD performance for the *Answering* dataset

(b) RD performance for the *Drinking* dataset

(c) RD performance for the *Arena* dataset

(d) RD performance for the *Sitting* dataset

Fig. 6. Rate-distortion (RD) performance evaluation on *Answering*, *Drinking*, *Arena*, and *Sitting* scene sequences. We vary the number of basis vectors used for TSMC, *qp* level during encoding for TVMC* and Draco, and TSDF resolution and block size for KLT and NeCGS to reach a similar range of bitrates. Note that below 0.7 SSIM, the quality is not perceivable.

and *Sitting*, each containing 200 frames. Additionally, we conduct experiments on the FSVVD [Hu et al. 2023] dataset, a point cloud dataset for complex scenes. To convert point cloud to 4D meshes, we use MeshLab [Cignoni et al. 2008] and Open3D [Zhou et al. 2018]. Specifically, we selected two datasets from FSVVD, *Answering* with 339 frames and *Drinking* with 316 frames. We further construct a *Synthetic* dataset with a high dynamic ratio by synchronizing and combining 4 time-varying mesh sequences used by the MPEG V-DMC standard [MPEG 2021]: *Dancer*, *Basketball player*, *Mitch*, and *Thomas*, each with 300 frames [Yang et al. 2023]. This dataset is designed to evaluate the scalability and robustness of TSMC under highly dynamic and diverse scene conditions.

Metrics. D2-PSNR metric, as defined by the MPEG group [MPEG 3DG 2019] is primarily suited for dynamic meshes where vertex correspondences are available, as they measure the error based on the Euclidean distance between tracked vertex pairs. However, in time-varying meshes without such correspondence information, the nearest neighbor is typically used instead, which introduces limitations in accurately evaluating D2-PSNR. To conduct a more comprehensive evaluation, we also use the SSIM [Chen and Bovik 2011] metric from four viewpoints to evaluate the quality of decoded meshes, considering both static background and dynamic parts simultaneously. To increase the accuracy and perceptual relevance of SSIM, we disable lighting effects and convert vertex normals to RGB colors, then render the meshes with these normal-based colors.

4.2 Comparisons

We compare TSMC with several state-of-the-art methods:

- **Draco** [Draco 2018]: A widely adopted tool that performs intra-frame compression independently on each mesh frame.
- **TVMC***: A scene mesh-compatible extension of TVMC [Chen et al. 2025] method, adapted to handle full scenes by bypassing the original winding number test. Displacement fields are computed as in TVMC but compressed directly using Draco instead of the KLT + Laplacian pipeline, providing a strong baseline for reference-based compression with standard tools.
- **KLT** [Tang et al. 2018]: An implicit compression approach that represents scenes as time-varying TSDF volumes and applies KLT to reduce temporal redundancy.
- **NeCGS** [Ren et al. 2025]: A TSDF-based neural compression method that introduces TSDF-Def to regularize geometry and employs neural networks to learn efficient scene representations.

4.3 Qualitative Results

Figure 5 presents qualitative comparisons between TSMC and state-of-the-art compression methods across four datasets and operating bitrates. Vertex-normal-colored renderings are used to visualize geometric distortions on non-textured meshes, making artifacts, particularly in self-contact regions, clearly visible and less dependent on lighting conditions. Across all four datasets, TSMC consistently achieves the highest quality at the lowest data rate. In contrast, TVMC* and Draco require aggressive quantization settings ($qp < 3$ or 4) to remain under 10 Mbps or 5 Mbps, often introducing temporal jitter or severe geometric distortions. TSDF-based methods such as KLT and NeCGS degrade rapidly at low bitrates due to reduced TSDF resolution and limited basis vectors, resulting in over-smoothing and missing background structures.

We show more results and comparison on these 4 datasets in Figure 7, Figure 8, and the supplementary video. From the visual comparisons, we can clearly see that TSMC significantly eliminates the effects of self-contact regions and has a much better visual quality than the other 4 baselines with a relatively low bitrate. Similarly, limited bitrates lead to over quantization for TVMC* and Draco, and huge distortions for TSDF-based methods, KLT, and NeCGS.

4.4 Objective Rate Distortion Performance

Figure 6 shows the rate-distortion performance of TSMC and the baselines. We evaluated the performance of TSMC by varying the number of volume centers to 1000, 2000, 3000, and 4000, and the group of frames (GoF) of 5, 10, and 15. In this section, we present the results with 2000 volume centers and a GoF of 10.

As shown in Figure 6, TSMC outperforms all baselines in maintaining a relatively high quality at bitrates below 15 Mbps. Notably,

Table 1. Objective comparisons between TSMC and baselines for the qualitative results shown in Figure 5 and Figure 8. The dynamic ratio indicates the proportion of vertices belonging to dynamic parts in the original scene. SSIM results are calculated after converting the vertex normal to RGB colors and assigning it to non-textured meshes, where higher SSIM values indicate better quality. Reported SSIM values are averaged over four viewpoints. Table 1 also includes average decoding times, showing that TSMC can decode scenes in real-time at ≈30 FPS. The first- and second-best values for D2-PSNR, SSIM, and decoding time in each scene are highlighted in orange and yellow, respectively.

Dataset	Dynamic Ratio	TSMC (Ours)			TVMC*			Draco			KLT			NeCGS		
		D2-PSNR ↑	SSIM ↑	Decoder ↓	D2-PSNR ↑	SSIM ↑	Decoder ↓	D2-PSNR ↑	SSIM ↑	Decoder ↓	D2-PSNR ↑	SSIM ↑	Decoder ↓	D2-PSNR ↑	SSIM ↑	Decoder ↓
Answering	24.46%	54.63	**0.9046**	40.41 ms	**55.90**	0.8610	**12.07 ms**	33.70	0.7588	14.00 ms	44.05	0.7999	47.17 ms	37.90	0.8068	42.48 ms
Sitting	16.31%	50.44	**0.9209**	29.10 ms	**50.60**	0.8815	**5.022 ms**	27.87	0.8389	7.00 ms	37.75	0.8546	65.35 ms	44.91	0.8739	21.09 ms
Arena	9.487%	**51.20**	**0.8204**	31.50 ms	51.10	0.7951	**14.03 ms**	15.36	0.7211	19.00 ms	25.80	0.7482	215.98 ms	25.17	0.7238	170.80 ms
Drinking	4.654%	**58.64**	**0.9335**	36.98 ms	57.74	0.8964	12.01 ms	28.79	0.8472	**12.00 ms**	41.49	0.8667	53.21 ms	47.41	0.8718	45.41 ms
Average	**13.73%**	**53.48**	**0.8949**	**34.50 ms**	**53.84**	**0.8585**	**10.78 ms**	**26.43**	**0.7915**	**13.25 ms**	**37.27**	**0.8174**	**95.43 ms**	**38.85**	**0.8191**	**69.95 ms**

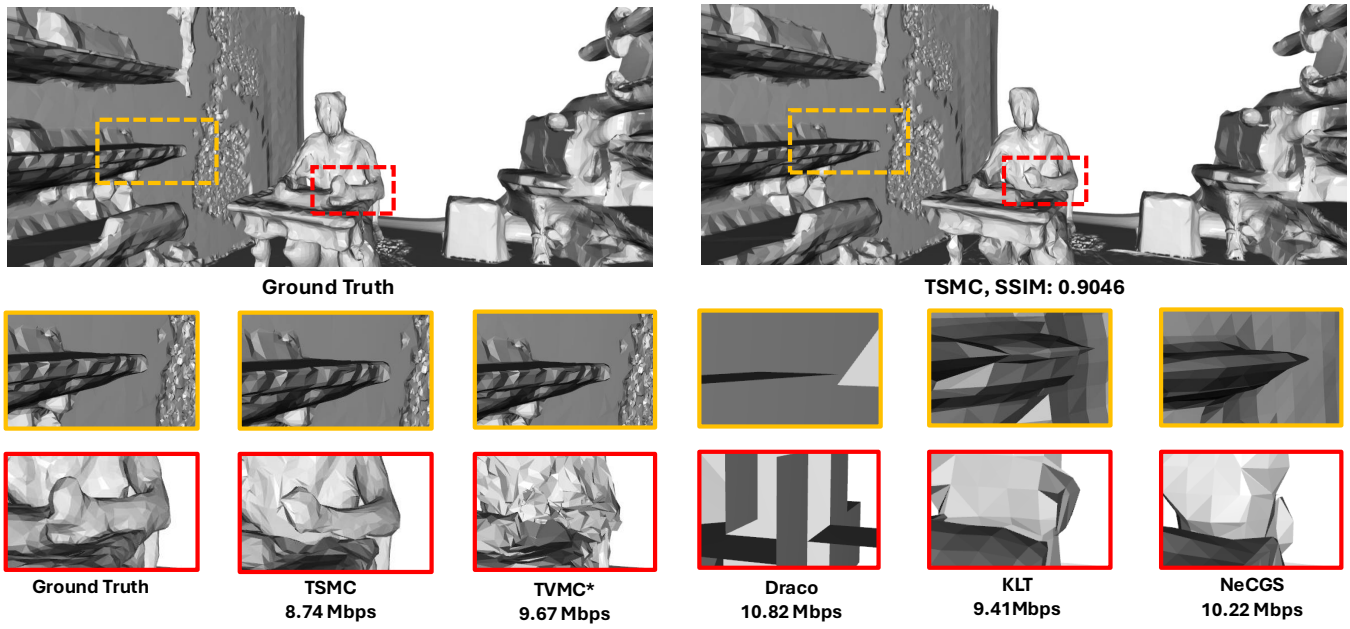
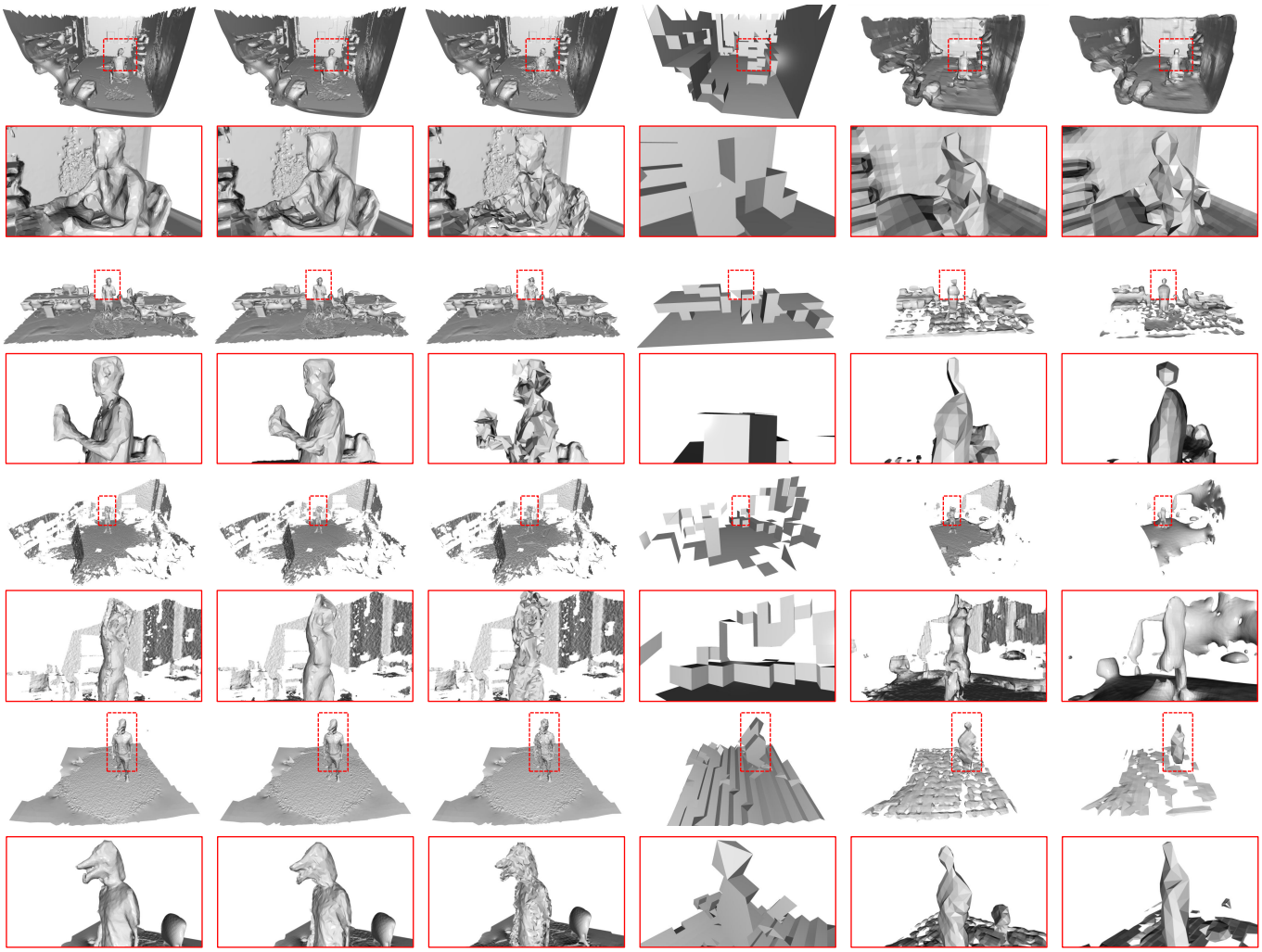


Fig. 7. Comparison between TSMC and baselines on the *Answering* dataset. We compare TSMC with all baselines and show the SSIM value and per-frame decoding time for TSMC, along with bitrates for all methods at 30 fps. Despite its high compression ratio and low bitrate, TSMC can reconstruct the scene mesh sequence with the least distortion.

TSMC achieves high-quality reconstruction at low bitrates due to the combined use of KLT and Laplacian coordinates compression. For example, in the *Answering* sequence, TSMC achieves an SSIM above 0.9 at around 8 Mbps, whereas TVMC* requires nearly 13 Mbps to reach comparable quality. TSMC manages to do this by noticing the structural similarity between displacements. For a series of continuous actions, the KLT can effectively capture over 95% of the displacement information using as few as least $m = 3$ basis vectors. By further converting only these m basis vectors into Laplacian coordinates, additional redundancy is removed by exploiting spatial coherence. This enhanced compression efficiency explains why TSMC stands out even better at low bitrates.

However, TSMC's SSIM fluctuates slightly with increasing bitrate and reaches a peak when all basis vectors are used. This behavior arises because TSMC can already reconstruct a reasonably accurate overall geometry using just the initial few basis vectors. As more basis vectors are included, the shape is further refined toward the ground truth. Unlike quantization-based methods, this refinement is not strictly linear, leading to fluctuations in quality before converging to its maximum. In contrast, Draco can become nearly lossless as the bitrate increases. TSMC and TVMC* are inherently lossy due to the volume-tracking and reference mesh extraction processes, which impose an upper limit on the achievable quality.

Furthermore, as shown in Table 1, under similar bitrates, TSMC achieves comparable D2-PSNR with TVMC*, while substantially outperforming all other baselines in terms of SSIM. TSMC also achieves an average decoding time of 34.50 ms, slower than TVMC* (10.78 ms) and Draco (13.25 ms), but up to 85.42% faster than KLT and NeCGS, which rely on the Marching Cubes algorithm [Lorensen and Cline 1998] for decoding. This efficiency is due to TSMC's use of simple subdivision and displacement deformation techniques during decoding. Moreover, TSMC enables GPU acceleration for the displacement



(a) Ground Truth (b) TSMC (c) TVMC* (d) Draco (e) KLT (f) NeCGS

Fig. 8. Qualitative evaluation on four scenes: *Answering*, *Drinking*, *Arena*, and *Sitting*. For each scene, we visualize both the ground truth meshes and the reconstructed meshes produced by different methods. The corresponding data rates and SSIM values are detailed in Figure 5 and Table 1. In each visualization, the region enclosed by the solid red box provides a magnified view of the area highlighted by the dotted red box in the original scene mesh.

decoding. Decoding large meshes typically consumes substantial computational resources, however, the reuse of static backgrounds in TSMC greatly reduces decoding times. While TSMC's encoding complexity is relatively high (e.g., a few seconds/frame), its fast decoding speed makes it well-suited for real-time playback style applications in which encoding can be done offline in advance.

4.5 Results on Synthetic Dataset

The *Synthetic* dataset represents an extreme scenario with a 100% dynamic ratio, where all regions in the scene exhibit continuous motion. Increasing the dynamic proportion to this level naturally reduces the relative gains obtained from our static–dynamic region differentiation algorithm, as the proportion of static regions becomes negligible.

Table 2. Objective results of TSMC and baselines on *Synthetic* dataset with 100% dynamic ratio. All methods are compared at comparable D2-PSNR levels to enable a fair evaluation of other metrics, including SSIM, decoding time, and bitrate.

Method	D2-PSNR ↑	SSIM ↑	Decoder (ms) ↓	Bitrate (Mbps) ↓
TSMC (Ours)	63.472	0.9416	66.04	8.72
TVMC*	60.385	0.8820	36.46	9.847
Draco	57.597	0.8990	9.10	14.487
KLT	61.450	0.9424	24.70	10.737
NeCGS	61.739	0.9340	6.89	6.931

Nevertheless, the dynamic-region compression component of TSMC remains effective under such conditions. Our results in Table 2

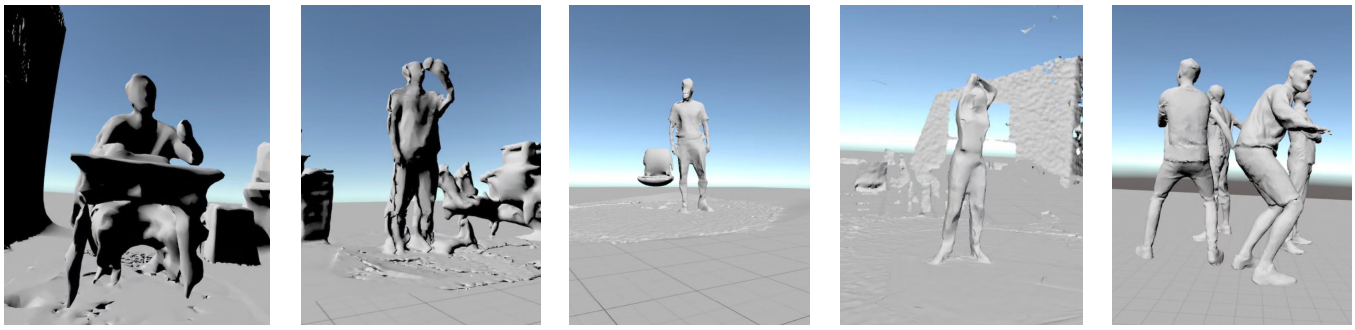


Fig. 9. Screenshots of TSMC real-time VR playback on a standalone Meta Quest 3 headset. All decoding and rendering are performed fully on-device and can achieve 24 FPS.

Table 3. Impact of TSMC on a hybrid Gaussian–mesh framework LMG [Sun et al. 2026] on the *Dancer* sequence. TSMC reduces the mesh bitrate by nearly 60× while maintaining comparable rendering quality in terms of SSIM and PSNR. The Gaussian splatting (GS) bitrate remains unchanged because of the same number of splats.

Method	SSIM ↑	PSNR ↑	GS Bitrate (Mbps)	Mesh Bitrate (Mbps)
LMG w/ TSMC	0.9462	25.35	719.8	6.13
LMG w/o TSMC	0.9594	26.46	719.8	357.6

show that TSMC still achieves competitive performance compared to baseline methods, and in many cases continues to outperform them in terms of rate–distortion efficiency. This demonstrates the robustness of TSMC across a wide spectrum of 4D scenes.

4.6 Real-time TSMC decoder on Meta Quest 3

We implement TSMC on Meta Quest 3 together with a full VR playback system that can achieve a FPS of 24. To the best of our knowledge, TSMC is the first 4D mesh decoder that runs entirely on a standalone VR headset, demonstrating its practicality for 4D decoders. Figure 9 shows screenshots of the VR playback system. Additional visual results of the real-time decoding are provided in the supplementary video. While not yet optimized for higher frame rates, this implementation highlights the practicality of TSMC for on-device 4D mesh playback under realistic hardware constraints.

4.7 Discussion with Gaussian Splatting and Other Volumetric Content

Although TSMC is designed for compressing meshes, we note that other 4D representations, such as point clouds [Liang et al. 2024] and Gaussian splats [Liang et al. 2026; Sun et al. 2026; Zheng et al. 2025], are also widely used. We do not consider point clouds in our evaluation due to their relatively lower visual fidelity for a given bitrate. Instead, we focus on Gaussian splatting approaches, which have recently demonstrated strong rendering quality and are increasingly adopted in high-fidelity volumetric pipelines. TSMC can also benefit hybrid 3DGS frameworks that integrate meshes with Gaussian splats. In these systems, meshes are often used as structural priors (e.g., for binding or regularizing Gaussian primitives), and their storage cost can still be significant. To demonstrate this, Table 3 reports results on the LMG framework [Sun et al. 2026], which binds Gaussian splats to mesh triangles, evaluated on the *Dancer* sequence from our *Synthetic* dataset. By applying TSMC to compress the mesh component, we reduce the mesh bitrate approximately by the factor of 60 (from 357.6 Mbps to 6.13 Mbps). Despite this significant reduction, the rendering quality remains comparable, with only minor differences in SSIM and PSNR. Gaussian splats here are encoded using traditional Entropy encoding. Note that the SSIM and PSNR metrics reported here are computed on rendered RGB images within the Gaussian splatting pipeline, and therefore differ from the metrics defined in Section 4.1.

5 CONCLUSION

In this paper, we propose TSMC, a novel compression method for time-varying 4D scene mesh sequences. To the best of our knowledge, TSMC is the first mesh compression method that can exploit temporal redundancy for inter-frame coding to compress *large and unbounded scene meshes*. Unlike prior methods that are limited to static or object-level dynamic meshes, TSMC supports compression of full scenes with complex spatial and temporal variations. Moreover, TSMC can also benefit hybrid 3DGS frameworks that integrate meshes with Gaussian splats. With extensive experiments, we demonstrate that TSMC outperforms several state-of-the-art baselines, including Draco [Draco 2018], a scene mesh extended version of TVMC [Chen et al. 2025] denoted as TVMC*, a TSDF-based compression method [Tang et al. 2018] denoted as KLT, and the neural-based compression method NeCGS [Ren et al. 2025], both in quantitative and qualitative evaluations.

Acknowledgments

This work was partially supported by the U.S. National Science Foundation (NSF) grant IIS-2544317 and the Czech Science Foundation grant 25-16495K "Coordinate-based representation of time-varying meshes".

References

Rashid Abbasi, Ali Kashif Bashir, Hasan J Alyamani, Farhan Amin, Jaehyeok Doh, and Jianwen Chen. 2022. Lidar point cloud compression, processing and learning for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems* 24, 1 (2022), 962–979.

Eva Agapaki and Ioannis Brilakis. 2021. Instance segmentation of industrial point cloud data. *Journal of Computing in Civil Engineering* 35, 6 (2021), 04021022.

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.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. 2025. SAM 3: Segment Anything with Concepts. arXiv:2511.16719 [cs.CV] https://arxiv.org/abs/2511.16719

Guodong Chen, Filip Hácha, Libor Váša, and Mallesham Dasari. 2025. TVMC: Time-Varying Mesh Compression Using Volume-Tracked Reference Meshes. In *Proceedings of the 16th ACM Multimedia Systems Conference*. 79–89.

Ming-Jun Chen and Alan C Bovik. 2011. Fast structural similarity index algorithm. *Journal of Real-Time Image Processing* 6, 4 (2011), 281–287.

Yun-Chun Chen, Vladimir Kim, Noam Aigerman, and Alec Jacobson. 2023. Neural progressive meshes. In *ACM SIGGRAPH 2023 Conference Proceedings*. 1–9.

Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. 2008. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*, Vittorio Scarano, Rosario De Chiara, and Ugo Erra (Eds.). The Eurographics Association. https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136

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.

Edsger W Dijkstra. 2022. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: his life, work, and legacy*. 287–290.

Alexandros Doumanoglou, Dimitrios Alexiadis, Stylianos Asteriadis, Dimitrios Zarpalas, and Petros Daras. 2014. On human time-varying mesh compression exploiting activity-related characteristics. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 6147–6151.

Google Draco. 2018. Draco. https://google.github.io/draco/.

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.

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.

Michael S Floater. 2003. Mean value coordinates. *Computer aided geometric design* 20, 1 (2003), 19–27.

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

Antoine Guédon and Vincent Lepetit. 2024. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5354–5363.

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

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.

Kaiyuan Hu, Yili Jin, Haowen Yang, Junhua Liu, and Fangxin Wang. 2023. FSVVD: A dataset of full scene volumetric video. In *Proceedings of the 14th Conference on ACM Multimedia Systems*. 410–415.

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

Xudong Jin, Jianfeng Xu, and Kei Kawamura. 2024b. 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.

Palle ET Jorgensen and Myung-Sin Song. 2007. Entropy encoding, Hilbert space, and Karhunen-Loève transforms. *Journal of mathematical physics* 48, 10 (2007).

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

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 2023. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.* 42, 4 (2023), 139–1.

Jaehong Kim, Tao Jin, Mallesham Dasari, Srinivasan Seshan, and Anthony Rowe. 2026. SceneHub4D: A Dataset and Evaluation Framework for 6-DoF 4D VR Scenes. *IEEE Transactions on Visualization and Computer Graphics* (2026).

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.

Zhicheng Liang, Junhua Liu, Mallesham Dasari, and Fangxin Wang. 2024. Fumos: Neural compression and progressive refinement for continuous point cloud video streaming. *IEEE Transactions on Visualization and Computer Graphics* 30, 5 (2024), 2849–2859.

Zhicheng Liang, Dayou Zhang, Linfeng Shen, Miao Zhang, Jian Zhang, Bin Ju, Mallesham Dasari, Fangxin Wang, and Jiangchuan Liu. 2026. 4DGStream: Variable bitrate dynamic Gaussian splatting streaming. *IEEE Transactions on Multimedia* (2026).

William E Lorensen and Harvey E Cline. 1998. Marching cubes: A high resolution 3D surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*. 347–353.

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

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.

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.

Microsoft. 2022. as. https://azure.microsoft.com/en-us/services/kinect-dk/. Online; accessed May 2022.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

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

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

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

Ege Özsoy, Evin Pınar Örnek, Ulrich Eck, Tobias Czempiel, Federico Tombari, and Nassir Navab. 2022. 4d-or: Semantic scene graphs for or domain modeling. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 475–485.

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

Siyu Ren, Junhui Hou, Weiyao Lin, and Wenping Wang. 2025. Neural Compression for 3D Geometry Sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 25294–25304.

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

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, Vol. 4. Citeseer, 109–116.

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

Yuan-Chun Sun, Guodong Chen, Sam Ziaie Kondori, Mallesham Dasari, and Cheng-Hsin Hsu. 2026. LMG: Efficient Streaming of Layered Mesh–Gaussian 3D Scenes. In *Proceedings of the 17th ACM Multimedia Systems Conference*.

Danhang Tang, Mingsong Dou, Peter Lincoln, Philip Davidson, Kaiwen Guo, Jonathan Taylor, Sean Fanello, Cem Keskin, Adarsh Kowdle, Sofien Bouaziz, Shahram Izadi, and Andrea Tagliasacchi. 2018. Real-time compression and streaming of 4D performances. *ACM Trans. Graph.* 37, 6, Article 256 (Dec. 2018), 11 pages. https://doi.org/10.1145/3272127.3275096

Libor Váša, Stefano Marras, Kai Hormann, and Guido Brunnett. 2014. Compressing dynamic meshes with geometric laplacians. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 145–154.

Libor Váša and Oldřich Petřík. 2011. Optimising perceived distortion in lossy encoding of dynamic meshes. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1439–1449.

Joanna Waczyńska, Piotr Borycki, Sławomir Tadeja, Jacek Tabor, and Przemysław Spurek. 2024. Games: Mesh-based adapting and modification of gaussian splatting. *arXiv preprint arXiv:2402.01459* (2024).

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.

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).

Chengwei Zheng, Lixin Xue, Juan Zarate, and Jie Song. 2025. Gaustar: Gaussian surface tracking and reconstruction. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 16543–16553.

Yu Zhong. 2009. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In *2009 IEEE 12th international conference on computer vision workshops, ICCV Workshops*. IEEE, 689–696.

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