Implicit Surface Compression — with Good Old Discrete Cosine Transform and Motion Compensation

TAO JIN, Carnegie Mellon University, USA
SHENGXI WU, Carnegie Mellon University, USA
TIANSHU HUANG, Carnegie Mellon University, USA
MALLESHAM DASARI, Northeastern University, USA
SRINIVASAN SESHAN, Carnegie Mellon University, USA
ANTHONY ROWE, Carnegie Mellon University, USA and Bosch Research, USA

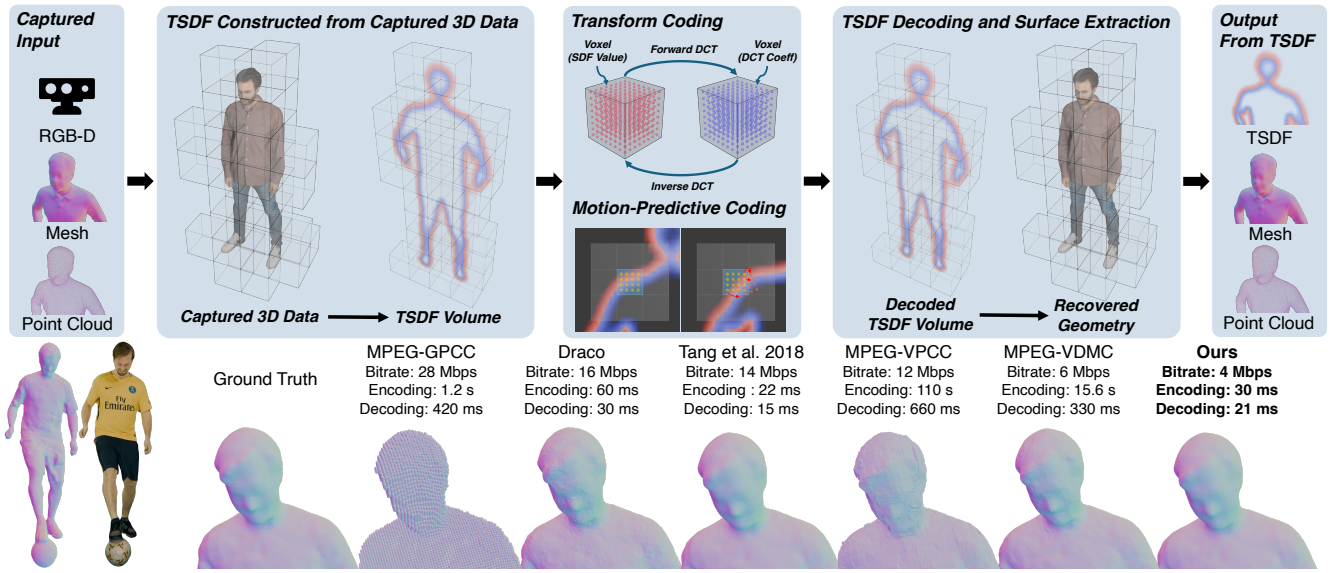


Fig. 1. Overview of our implicit-surface predictive codec for captured dynamic geometry. Our method targets 3D data produced by 3D capture/reconstruction pipelines, including RGB-D capture sources, captured meshes and point clouds. These inputs are represented as a sparse TSDF volume, which serves as the coding domain. We compress the dynamic TSDF sequence using block-wise transform coding and motion-predictive coding across frames to exploit spatial and temporal redundancy. The decoder reconstructs the TSDF volume and extracts meshes and point clouds through surface extraction algorithms.

The rapid adoption of volumetric capture technologies has created a pressing need for efficient storage and streaming of dynamic 3D content. Unfortunately, current compression standards often treat dynamic sequences as independent frames or rely on computationally expensive non-rigid registration, making them unsuitable for real-time applications or large-scale environments. In this paper, we present a novel end-to-end compression framework for dynamic Truncated Signed Distance Field volumes derived from captured 3D content, leveraging a representation that is temporally stable, easily parallelizable, and already widely used in scene reconstruction and volumetric fusion pipelines. We then adapt classic 2D video coding paradigms such as spatial coding via Discrete Cosine Transform and temporal coding using a real-time motion compensation pipeline to provide robust, real-time, and training-free encoding and decoding for 3D content. Extensive evaluations on human performance captures demonstrate that our codec achieves ~35% bitrate savings at equal distortion while operating in real time at 30 FPS, while stronger temporal coherence in large-scale synthetic environments yields up to 12× bitrate reduction at equal distortion.

CCS Concepts: • **Theory of computation → Data compression**; • **Computing methodologies → Volumetric models**; **Computer vision**; *Image compression*; *Reconstruction.*

ACM Reference Format:
Tao Jin, Shengxi Wu, Tianshu Huang, Mallesham Dasari, Srinivasan Seshan, and Anthony Rowe. 2026. Implicit Surface Compression — with Good Old Discrete Cosine Transform and Motion Compensation. *ACM Trans. Graph.* 45, 4, Article 109 (July 2026), 14 pages. https://doi.org/10.1145/3811298

Authors' Contact Information: Tao Jin, Carnegie Mellon University, Pittsburgh, PA, USA, taojin@andrew.cmu.edu; Shengxi Wu, Carnegie Mellon University, Pittsburgh, PA, USA, shengxiw@andrew.cmu.edu; Tianshu Huang, Carnegie Mellon University, Pittsburgh, PA, USA, tianshu2@andrew.cmu.edu; Mallesham Dasari, Northeastern University, Boston, MA, USA, m.dasari@northeastern.edu; Srinivasan Seshan, Carnegie Mellon University, Pittsburgh, PA, USA, srini@andrew.cmu.edu; Anthony Rowe, Carnegie Mellon University, Pittsburgh, PA, USA and Bosch Research, Pittsburgh, PA, USA, agr@andrew.cmu.edu.

ACM 1557-7368/2026/7-ART109
https://doi.org/10.1145/3811298

1 Introduction

The technology for capturing and playing back physical reality is maturing rapidly. Driven by the surge in consumer-grade depth cameras and depth estimation frameworks [Azure 2019; Wen et al. 2025; Yang et al. 2024], the proliferation of high-fidelity Virtual Reality headsets [Apple 2024; Meta 2026; Varjo 2026], and algorithmic breakthroughs in scene reconstruction and rendering [Kerbl et al. 2023; Mildenhall et al. 2021; Oechsle et al. 2021; Wang et al. 2021], large-scale dynamic 3D environments can be captured, reconstructed, and visualized with unprecedented realism. These advances are rapidly shifting the industry focus from static 3D models to dynamic volumetric video, enabling applications ranging from immersive telepresence to real-time digital twins.

Despite these advances in capture and display, the transport and storage of dynamic 4D content remains a critical bottleneck. Although static model compression is well-studied, 4D geometry compression is still underserved. For example, temporal data in 2D video has been a significant contributor to high compression ratios [Sze et al. 2014]. However, dynamic 3D sequences are often treated as a series of independent static frames; for instance, explicit surface coding tools like Google Draco [Galligan et al. 2018] typically compress each frame in isolation. Even though modern MPEG pipelines for dynamic geometry (e.g., V-PCC [Graziosi et al. 2020] and V-DMC [Zou et al. 2025]) provide temporal compression, their complexity and preprocessing overhead can make low-latency, real-time streaming challenging in practice. At the same time, many volumetric capture and dense RGB-D reconstruction systems [Millane et al. 2024; Newcombe et al. 2011] already maintain scene geometry internally as volumetric fields during fusion and mapping, yet existing codecs are primarily designed for explicit surface representations rather than this reconstruction-native representation.

To address these challenges, we focus on dynamic geometry represented as Truncated Signed Distance Fields (TSDFs) [Curless and Levoy 1996], a common representation in dense volumetric fusion [Jin et al. 2024; Millane et al. 2024; Newcombe et al. 2015, 2011], high-fidelity performance capture [Tang et al. 2018, 2020], and SLAM-based mapping pipelines [Dai et al. 2017; Prisacariu et al. 2017; Whelan et al. 2015a]. In these settings, TSDFs are already maintained as the working geometry representation, making them a natural target for compression and streaming. However, prior TSDF-based compression work has largely focused on small, bounded capture scenes and per-frame encoding without exploiting temporal redundancy, while large-scale TSDF reconstruction systems are designed for fusion and mapping rather than compact transmission. This leaves an important gap for real-time streaming of reconstructed 3D content.

Our core contribution is to treat the TSDF as a video stream, applying signal processing techniques akin to traditional 2D video codecs to 3D geometry. Unlike prior implicit codecs that learn *content-dependent* block transforms (e.g., Karhunen–Loève Transform (KLT) [Hotelling 1933; Tang et al. 2018] or neural-network-based transforms trained on a dataset [Tang et al. 2020]), we use the Discrete Cosine Transform (DCT), eliminating the overhead of data-driven training with only a small reduction in the quality of the reconstruction. Furthermore, we introduce a motion compensation pipeline tailored for voxelized geometry, including a simple "zero-motion" predictor and block-based rigid motion predictors that align local geometry across frames. Our ablations show that effective temporal compression *does not* require complex non-rigid tracking: zero-motion prediction already provides a strong fallback, 3-Degrees-of-Freedom (DoF) often achieves the best bitrate–distortion tradeoff, and full 6-DoF alignment can be limited by its higher motion-vector overhead.

Contributions. By effectively modeling temporal coherence, we achieve a significant leap in efficiency, enabling efficient, real-time streaming at 30 FPS:

- We design a novel, real-time encoder-decoder pipeline for implicit surfaces which exploits spatial and temporal redundancies using a DCT-based temporal compression pipeline and motion compensation.
- Through extensive evaluations and ablations, we show our approach offers ~35% bitrate savings at equal distortion compared to baselines, and is two orders of magnitude faster than temporal geometry codecs such as MPEG V-PCC and V-DMC.
- We provide a highly-optimized, open-source, GPU-accelerated C++/CUDA implementation of our compression pipeline, including both encoder and decoder on our project website.[1]

Limitations. Our primary target is dynamic captured or reconstructed geometry that is already represented, or naturally maintained, as TSDFs in volumetric fusion and scene reconstruction pipelines. More general geometry such as meshes or point clouds can also be compressed by first converting them to a TSDF, but this conversion is not lossless: it does not preserve mesh connectivity, texture coordinates, rigging, or other surface attributes, and it may smooth fine detail, or merge nearby contact surfaces. As a result, our method is less suitable for authored assets whose topology, parameterization, or exact surface structure must be preserved. In addition, this paper focuses on geometry compression only. We do not compress texture or other appearance features; when appropriate, appearance can be transmitted separately using conventional video streams and projectively textured in capture settings. Prior TSDF-based work has explored texture compression in offline settings [Tang et al. 2020], and incorporating similar ideas into a real-time system remains an important direction for future work.

2 Related work

Dynamic geometry compression standards. Recent surveys provide broad coverage of dynamic 3D mesh coding [Marvie et al. 2023] and inter-prediction methods for time-varying mesh compression [Dvořák et al. 2025]. They highlight a key distinction between constant-connectivity animated meshes and time-varying meshes, where connectivity, topology, texture coordinates, and vertex counts may change across frames, making temporal prediction substantially more challenging. MPEG has developed multiple pipelines for dynamic 3D content, including video-based point cloud compression (V-PCC) [Graziosi et al. 2020] and video-based dynamic mesh coding (V-DMC) [Zou et al. 2025]. These standards achieve strong rate-distortion performance, but often rely on substantial

[1] Project website: https://implicit-surface-codec.github.io/

preprocessing, such as patch/atlas generation in V-PCC or tracking/basemesh fitting in V-DMC, which can complicate low-latency real-time streaming. Our work targets a complementary point in the design space: a training-free, GPU-parallelizable codec operating on a stable TSDF domain with fast inter-frame prediction.

Dynamic mesh sequence compression and deformation tracking. Building on static mesh compression methods such as TFAN [Mamou et al. 2009], EdgeBreaker [Rossignac 2002], and Draco [Galligan et al. 2018], dynamic mesh compression exploits temporal redundancy through inter-frame correspondence, reference meshes, or non-rigid registration. Recent methods such as TVMC [Chen et al. 2025] use volume-tracked reference meshes for time-varying mesh sequences. Because many dynamic mesh codecs rely on temporal correspondence, related deformation tracking methods are also relevant. Template-based methods [Collet et al. 2015; Li et al. 2009] can achieve high quality but depend on accurate priors and may require expensive per-frame processing. Template-free reconstruction methods such as DynamicFusion [Newcombe et al. 2015] and ElasticFusion [Whelan et al. 2015b] maintain an evolving canonical model using warp fields or deformation graphs [Sumner et al. 2007], often with as-rigid-as-possible regularization [Igarashi et al. 2005; Sorkine et al. 2007]. Although some operate in real time, incremental tracking can accumulate errors under fast or large deformations, and deformation-graph assumptions are not naturally suited to topological or contact-state changes, such as surfaces that touch and later separate. Neural methods improve correspondence or deformation modeling [Bozic et al. 2021], but often require sequence-level optimization and target reconstruction/tracking rather than real-time compression. In contrast, our method avoids explicit non-rigid surface correspondence and performs temporal prediction directly in the TSDF block domain, while sharing the volumetric limitation that nearby contact surfaces may be smoothed or merged within a TSDF.

Implicit and volumetric geometry compression. Temporal redundancy has also been exploited for dynamic point clouds through motion compensation in voxelized domains [De Queiroz and Chou 2017] and 3D motion prediction for video-based point cloud compression [Li et al. 2019]. Implicit 3D representations, including TSDFs and occupancy fields, have been compressed with volumetric transforms and multiresolution encodings, such as wavelet-based methods [Schelkens et al. 2003], JPEG-2000 extensions [Schelkens et al. 2006], and transform decompositions of voxelized distance fields [Krivokuća et al. 2019]. For TSDF-centric capture, Tang et al. [2018] introduced real-time compression by training a blockwise KLT on localized TSDF blocks, later extended with learned neural codecs [Tang et al. 2020]. More recently, Ren et al. [2025] represented diverse mesh models as TSDF tensors and learned a shared auto-decoder with per-shape embeddings, demonstrating the potential of TSDF-like implicit tensors for compact geometry coding. These methods rely on content-specific transforms, learned neural priors, cross-shape redundancy, or largely per-frame coding. In contrast, our method is training-free and targets online 4D geometry streams by explicitly exploiting temporal redundancy through motion-compensated TSDF block alignment and transform-domain residual coding.

Neural and point-based scene representations. Recent scene representations, including neural implicit surfaces [Oechsle et al. 2021; Wang et al. 2021], Neural Radiance Fields [Barron et al. 2021; Mildenhall et al. 2021], and 3D Gaussian Splatting [Kerbl et al. 2023], have enabled high-quality reconstruction and novel-view synthesis from captured imagery. A growing line of feed-forward methods [Chen et al. 2024; Jiang et al. 2025] further aims to reduce reconstruction latency by learning priors across scenes. For example, MVSplat [Chen et al. 2024] predicts 3D Gaussian primitives from sparse posed images by constructing multi-view cost volumes, estimating depth, and unprojecting the predicted depths into Gaussian centers. These methods indicate an important trend toward efficient 3D scene representations, but address a different problem from geometry compression: they rely on trained networks or per-scene optimization, optimize view synthesis quality from images, and do not produce a codec bitstream or explicitly exploit inter-frame redundancy in dynamic geometry streams. In contrast, our work targets low-latency compression of time-varying geometry itself, using a training-free TSDF coding domain with block-based transform coding and motion-compensated temporal prediction.

3 Technical design

3.1 Overview

Explicit surface representations, such as meshes or point clouds, are challenging to compress in real-time using intra-frame coding and inter-frame prediction. As sampling densities vary, and mesh connectivity may change, expensive preprocessing such as correspondence estimation, basemesh fitting, or atlas/parameterization construction is often required to effectively generate inter-frame predictions. In addition to added latency, these steps are also frequently global in nature, making them difficult to scale to room-scale scenes and incremental online updates.

For our codec (Fig. 2), we instead operate in an implicit surface domain using TSDFs, where the surface is defined as the zero level set. This provides three key advantages:

- A TSDF defines a *fixed spatial domain* in which temporal prediction can be expressed as updates to the same spatial locations over time, making the predictor robust to connectivity and sampling changes.
- Within a truncation band, the TSDF is a locally smooth scalar signal, which yields stable quantization behavior and strong local correlation, well suited to block-based transform coding.
- The regular grid structure of the TSDF decomposes naturally into fixed-size voxel blocks which can be encoded, predicted, and updated in parallel.

Furthermore, only *active* surface-band blocks need to be encoded, allowing runtime to scale with surface area like explicit surfaces instead of the full scene volume. The next subsection details how we build the TSDF from the inputs our codec accepts (Sec. 3.2).

3.2 Geometry-to-TSDF construction

Volumetric capture pipelines typically provide geometry in one of two forms: a set of calibrated depth or RGB-D images from a multi-camera rig [Guo et al. 2019; Millane et al. 2024; Newcombe et al. 2011; Tang et al. 2018], or a pre-reconstructed explicit surface such

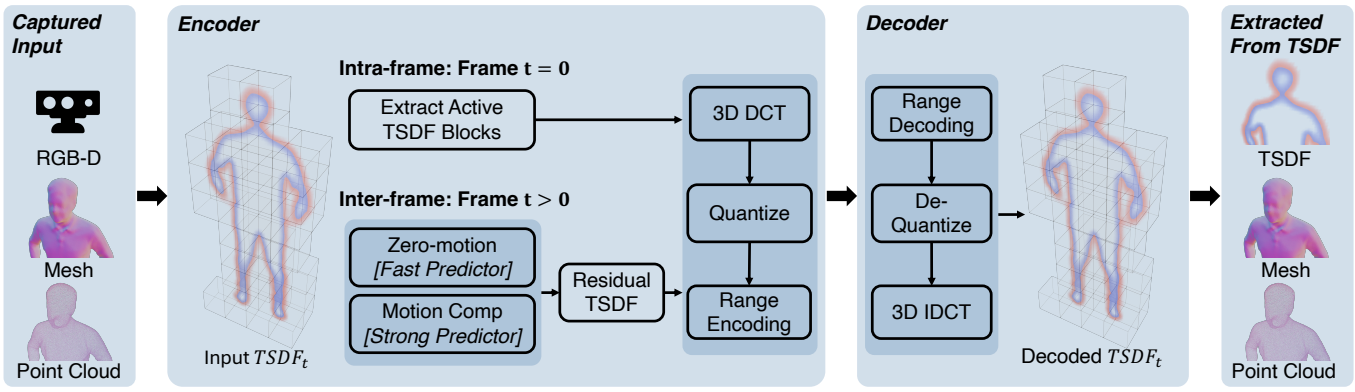


Fig. 2. System architecture of our implicit-surface temporal codec. Each frame is represented as a sparse TSDF volume of fixed-size voxel blocks, and we encode only the active blocks near the surface. The first frame (t=0) is intra-coded with per-block transform coding and entropy coding (Sec. 3.3), while subsequent frames are encoded relative to the previous frame using entropy-coded transform-domain residual coefficients (Sec. 3.4) and motion-compensated TSDF block alignment (Sec. 3.5). The decoder mirrors these steps to update its reference TSDF state over time, and an explicit surface is recovered via Marching Cubes. The illustrated blocks denote voxel blocks: each containing 8×8×8 voxels in our implementation. The figure shows a coarsened block layout for readability.

as a mesh or oriented point cloud. As illustrated on the left of Fig. 2, our codec accepts both types of input by mapping them onto the same sparse TSDF voxel grid used by the compression pipeline. Both conversion paths share the same high-level formulation: evaluate a signed distance at each voxel center, truncate it to a fixed band, and mark voxels inside the band as valid. They differ only in how the signed distance value is obtained. This section describes the two conversion paths: projective TSDF fusion from depth images and direct TSDF construction from explicit surfaces.

We first define the common TSDF representation used by both input paths. Let μ denote the truncation band and let $\mathbf{p}$ be a voxel center on the global voxel grid. Given a signed distance value $s(\mathbf{p})$, we store the truncated signed distance

$$\tilde{s}(\mathbf{p}) = \mathrm{clamp}(s(\mathbf{p}), -\mu, +\mu), \tag{1}$$

together with the observation weight as a binary validity flag

$$W(\mathbf{p}) = \begin{cases} 1, & s(\mathbf{p}) \text{ is observed and } |s(\mathbf{p})| \le \mu, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Only voxels with $W(\mathbf{p}) >= 1$ are considered active and passed to the compression pipeline. The two input paths differ only in how the signed distance measurement $s(\mathbf{p})$ is computed.

Projective TSDF fusion. For calibrated depth-camera captures, we use the standard projective TSDF fusion formulation introduced by volumetric fusion systems such as KinectFusion [Newcombe et al. 2011] and voxel hashing systems [Nießner et al. 2013]. Given a depth image $D(u, v)$ with pinhole intrinsics $K = \{f_x, f_y, c_x, c_y\}$ and camera pose $(R, \mathbf{t})$, a voxel center $\mathbf{p}$ in the global frame is transformed into the camera frame as

$$\mathbf{p}_c = R^\top(\mathbf{p} - \mathbf{t}) = (x, y, z)^\top. \tag{3}$$

It is then projected into the depth image using the pinhole camera model:

$$u = f_x \frac{x}{z} + c_x, \qquad v = f_y \frac{y}{z} + c_y. \tag{4}$$

If $z > 0$ and (u, v) lies inside the image, we read the observed depth $\hat{z} = D(u, v)$ and compute the projective signed distance

$$s(\mathbf{p}) = \hat{z} - z. \tag{5}$$

The measurement is valid only when the projected pixel has a valid depth and $|s(\mathbf{p})| \le \mu$. With multiple calibrated views, each view produces a per-voxel TSDF measurement, and valid measurements are fused by weighted averaging into the global TSDF volume [Curless and Levoy 1996].

Explicit surface conversion. In some settings, the input to the codec is not depth images but explicit geometry produced by an upstream reconstruction pipeline, such as a fused mesh or an oriented point cloud. To support these inputs, we convert the explicit surface into the same TSDF grid used by the codec. We describe the mesh case below; the oriented-point case follows the same principle by replacing the closest-triangle query with a closest oriented-point query.

Let the input mesh be a set of triangles $\mathcal{T} = \{\tau_j\}$ in world coordinates. Each triangle $\tau \in \mathcal{T}$ is defined by vertices $(\mathbf{a}, \mathbf{b}, \mathbf{c})$, and a point on the triangle can be written in barycentric form as

$$\mathbf{q}(v, w) = \mathbf{a} + v(\mathbf{b} - \mathbf{a}) + w(\mathbf{c} - \mathbf{a}), \qquad v \ge 0,\ w \ge 0,\ v + w \le 1. \tag{6}$$

For each voxel center $\mathbf{p}$, we compute the closest-surface distance

$$d(\mathbf{p}) = \min_{\tau \in \mathcal{T}} \|\mathbf{p} - \Pi_\tau(\mathbf{p})\|_2, \tag{7}$$

where $\Pi_\tau(\mathbf{p})$ denotes the closest point to $\mathbf{p}$ on triangle τ. If the orthogonal projection of $\mathbf{p}$ lies inside the triangle, this point gives $\Pi_\tau(\mathbf{p})$; otherwise, we project onto the three edge segments and take the nearest point.

For watertight meshes with consistent orientation, we assign the sign using the normal of the closest triangle. Let $\tau^\star$ be the closest triangle and $\mathbf{n}_{\tau^\star}$ its unit normal. We define

$$s(\mathbf{p}) = \mathrm{sign}\big((\mathbf{p} - \Pi_{\tau^\star}(\mathbf{p}))^\top \mathbf{n}_{\tau^\star}\big)\, d(\mathbf{p}). \tag{8}$$

This local point-to-plane side test is efficient and works well when the input surface has a coherent inside/outside convention.

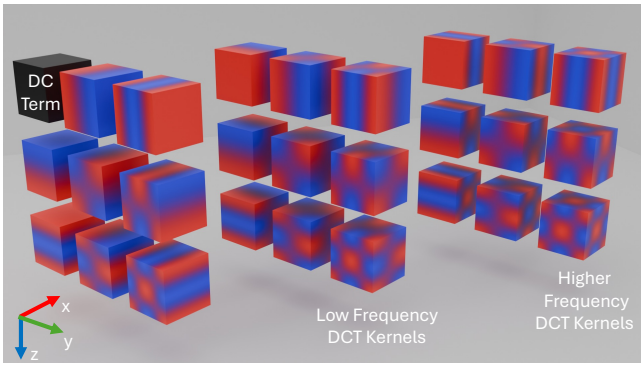


Fig. 3. Representative 3D DCT basis kernels, analogous to the 2D DCT basis table commonly used in JPEG image coding [Wallace 1991]. For readability, we show a $3 \times 3 \times 3$ subset from the full $8 \times 8 \times 8$ basis used by our codec. The DC kernel captures the block average; moving away from DC yields kernels with increasing spatial frequency along the x, y, and z axes. Our codec retains the first K low-frequency coefficients under a fixed scan order.

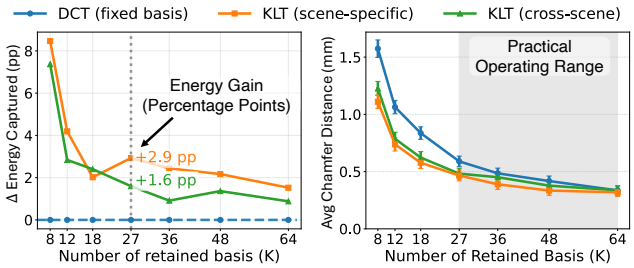


Fig. 4. DCT vs. KLT for TSDF blocks. Left: energy gain (percentage points) over the fixed 3D DCT baseline. Right: average Chamfer distance (millimeter) as a function of retained coefficients K. In the practical operating range $K \in [27, 64]$, cross-scene KLT closely tracks DCT, while scene-specific KLT provides only modest additional gain. Cross-scene KLT is evaluated on similar scene types and may degrade under out-of-distribution content. Error bars show std. over frames.

Non-watertight surfaces. Meshes reconstructed from real-world scans are often non-watertight, contain thin sheets, or include overlapping components. In these cases, a global inside/outside sign is ill-defined. Xu and Barbič [2020] address this problem for polygon-soup meshes by first computing an unsigned distance field and then constructing a manifold offset surface from which a signed distance field can be defined. Following this offset-surface intuition, we use a simplified unsigned-distance offset fallback:

$$s(\mathbf{p}) = d(\mathbf{p}) - \delta, \qquad \delta \approx 1\text{–}2 \text{ voxel size}. \tag{9}$$

This produces a stable zero level set corresponding to a thin offset shell around the input geometry, without requiring watertightness or consistent orientation.

For oriented point clouds, we compute the distance to the closest point and use its normal to define an analogous local point-to-plane signed distance. If the normals are unoriented or unreliable, we use the same unsigned-distance offset fallback. The signed distance values from either branch are truncated to the TSDF band and marked valid using the common rule above.

3.3 Intra-frame coding

TSDF blocks as coding units. To enable real-time encoding with predictable compute, we organize each frame's TSDF volume as a sparse set of fixed-size voxel blocks. Each voxel block contains $N \times N \times N$ voxels. These blocks form the basic coding units and can be processed independently in parallel, as shown in Fig. 2. A frame is therefore represented by the set of active voxel blocks, where active blocks are those containing voxels within the truncation band of the observed surface.

TSDF block transform coding. For each active TSDF block, we apply a 3D DCT to map TSDF values from the spatial domain to the frequency domain. Similar to the 2D DCT basis table commonly used in JPEG image coding [Wallace 1991], Fig. 3 visualizes representative 3D DCT kernels from our voxel-block transform. Because the TSDF within the truncation band is locally smooth, most energy concentrates in low spatial frequencies. We therefore retain the K lowest-frequency coefficients under a fixed 3D ordering (sorted by increasing index sum, i.e., Manhattan distance from DC) and set the remaining coefficients to zero. The decoder then reconstructs the block via the inverse 3D DCT transform.

DCT vs. learned transforms. Although data-driven transforms for TSDF blocks have been explored in prior work [Tang et al. 2018, 2020], we target real-time streaming across arbitrary scenes, and therefore avoid learned transforms that require computing and transmitting learned transform parameters in advance. The DCT also often approaches the compaction of the optimal data-dependent KLT for correlated signals [Ahmed et al. 2006]. In our experiments (Fig. 4), while a scene-specific KLT helps at small K, a cross-scene KLT closely tracks the fixed DCT in practical operating ranges required for good reconstruction performance (i.e., $K \in [27, 64]$), further motivating our choice of a data-independent 3D DCT.

Quantization and range coding. After truncating each block to K DCT coefficients, we quantize the coefficients with a fixed step for the DC term and a tunable step Δ_{AC} for AC coefficients. As a rate-distortion tradeoff, for a fixed K, smaller Δ_{AC} reduces rounding error (distortion) but increases rate (Fig. 5) by producing fewer zeros and a wider integer dynamic range.

We then apply lossless range coding [Martin 1979] to the quantized coefficients. As different frequencies exhibit distinct statistics (low-frequency terms have larger dynamic range, whereas high-frequency terms concentrate near zero), per-index streams yield tighter probability models for the range coder, reducing bitrate. Thus, rather than encoding each block as a mixed K-vector, we group coefficients by DCT index across all active blocks, producing K coefficient streams.

3.4 Inter-frame temporal coding

Dynamic geometry streams exhibit strong temporal redundancy, where consecutive frames often remain highly correlated even as the surface changes over time. Exploiting this correlation through residual coding reduces the entropy of the transmitted signal and

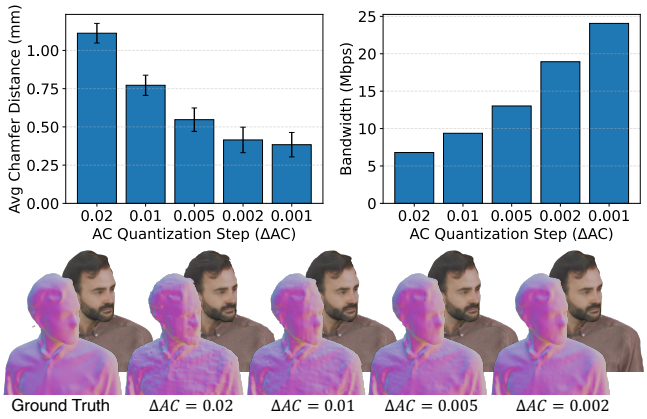


Fig. 5. Measuring the impact of AC quantization step Δ_{AC} on reconstruction quality (top-left), bandwidth (top-right), and qualitative results. Note we keep $K = 48$ fixed in this experiment, along with a fixed DC coefficient step size Δ_{DC}=0.001. Error bars indicate two standard deviations.

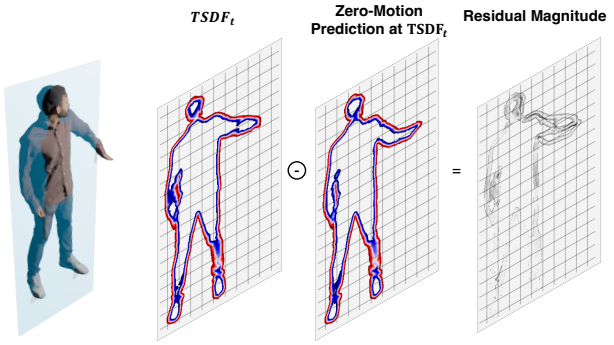


Fig. 6. Zero-motion prediction for TSDF temporal coding (visualized on 2D slice). We predict the current TSDF by reusing the previous reconstructed TSDF on the same voxel block grid. The residual is small and concentrated near surface change, which improves transform and range coding efficiency.

can substantially improve compression efficiency. This mirrors the classic transition in 2D media from intra-only image coding (JPEG-like) to inter-frame predictive video coding (MPEG-like), where much of the bitrate savings comes from sending changes rather than complete frames.

In our codec, temporal prediction operates directly on the block-wise TSDF field, using the same voxel-block coding unit as in intra-frame compression. Each frame is represented as a sparse set of active TSDF voxel blocks indexed on a common voxel block grid, so temporal updates can be expressed as blockwise residuals. This regular, indexable structure makes temporal coding simple and computationally efficient, and does not require explicit correspondence tracking often required by explicit geometry codecs.

Zero-motion block prediction. We begin by sending one intra-coded frame to initialize the decoder reference. For each subsequent frame, we reuse the previous reconstructed TSDF on the same voxel-block grid as a "zero-motion predictor" and encode only blockwise residual updates. Subtracting this predictor concentrates residual energy near regions of surface change, improving compressibility for subsequent transform and range encoding (Fig. 6).

Residual updates in the transform domain. We implement temporal updates in the same DCT space as intra coding. Let $\mathbf{s}_t^{(b)} \in \mathbb{R}^{N^3}$ denote the TSDF samples of block coordinate b at frame t, and let $\mathbf{c}_t^{(b)} \in \mathbb{R}^K$ denote the retained K coefficients of its 3D DCT representation under our fixed ordering (Sec. 3.3). The encoder maintains a reference coefficient buffer $\tilde{\mathbf{c}}_{t-1}^{(b)}$ from the previously reconstructed frame, keyed by block coordinates; for each active block b, we form a transform-domain residual

$$\Delta\mathbf{c}_t^{(b)} = \mathbf{c}_t^{(b)} - \tilde{\mathbf{c}}_{t-1}^{(b)}, \tag{10}$$

with $\tilde{\mathbf{c}}_{t-1}^{(b)} = \mathbf{0}$ if block b is absent (i.e., b is newly activated at time t). Conversely, if b existed in the reference but is inactive in the current frame, we treat it as a removal event by setting $\mathbf{c}_t^{(b)} = \mathbf{0}$. We quantize $\Delta\mathbf{c}_t^{(b)}$ using the same steps $(\Delta_{\mathrm{DC}}, \Delta_{\mathrm{AC}})$ as intra coding and range-encode the resulting integers using the same per-coefficient entropy coding strategy.

To prevent encoder-decoder drift, the encoder also performs a closed-loop reference update using the reconstructed residual:

$$\tilde{\mathbf{c}}_t^{(b)} = \tilde{\mathbf{c}}_{t-1}^{(b)} + Q^{-1}\big(Q(\Delta\mathbf{c}_t^{(b)})\big), \tag{11}$$

where $Q(\cdot)$ and $Q^{-1}(\cdot)$ denote quantization and dequantization.

Adaptive probability models. Range coding requires that the encoder and decoder share the same probability model (i.e., cumulative symbol frequencies) for each coefficient stream. A common *explicit-model* approach is to first build a histogram (or probability table) from the symbols and transmit this table as side information. The decoder reads the table and then decodes the payload using the corresponding model. Although effective in offline or per-frame settings, explicit tables introduce per-stream metadata overhead and complicate streaming when many short streams are produced.

For temporal coding, we instead use an *adaptive* range encoder. The encoder and decoder start from the same fixed initial model and update symbol frequencies online as the stream is encoded/decoded, so the probability model is reconstructed implicitly at the decoder without transmitting explicit histograms.

3.5 Motion compensation

Although zero-motion prediction (Sec. 3.4) already removes substantial temporal redundancy by reusing the previous reconstructed TSDF $s_{t-1}^{(b)}$ at the same block coordinates b, its performance degrades when the scene moves. Even modest object motion shifts the zero level set relative to the voxel grid, leaving residuals that are expensive to code. In real captures, motion is often non-rigid, but our codec targets online, real-time streaming, where we cannot afford global correspondence estimation or non-rigid tracking. Instead, we improve prediction with a lightweight *local rigid* motion model estimated independently per block, which captures much of the dominant displacement while keeping computation predictable.

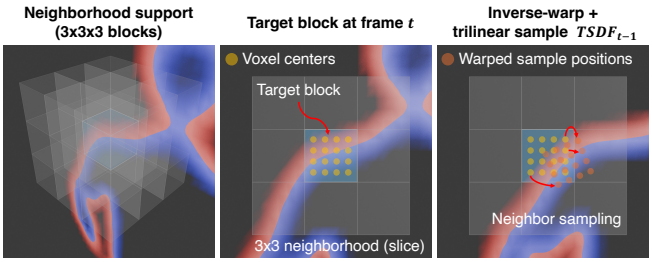


Fig. 7. TSDF block warping for motion-compensated prediction. We inverse-warp voxel-center samples of a target block at frame t into $TSDF_{t-1}$ and trilinearly interpolate over a stitched $3 \times 3 \times 3$ neighborhood. The larger support ensures valid interpolation even when warped samples land between voxel centers or in neighboring blocks.

Motion-compensated block warping. Given a target block $s_t^{(b)}$ at frame t, we form a motion-compensated predictor by inverse-warping its voxel-center sample positions into the previous reconstructed TSDF and resampling $s_{t-1}^{(b)}$ at the warped locations. First, as rigid motion is continuous, these warped locations typically fall at *fractional* positions between voxel centers; as such, we evaluate them via trilinear interpolation.

Rotation and translation can also map sample positions outside the target block, which means valid interpolation may require TSDF values from adjacent blocks. As illustrated in Fig. 7, we thus assemble a stitched 3×3×3 neighborhood around the corresponding block in $s_{t-1}^{(b)}$ to provide sampling support across block boundaries. This warped-and-resampled block serves as the predictor for residual coding, after which we encode transform-domain residual coefficients as in Sec. 3.4. We encode the per-block rigid transform (6-DoF) as motion side information; the decoder applies the same inverse warp to form an identical predictor before decoding residual coefficients.

Per-block motion estimation. We estimate per-block motion by aligning TSDF samples under a local 6-DoF rigid transform (rotation and translation). For the trilinear interpolate $\hat{s}_{t-1}^{(b)}(\cdot)$ over the $3 \times 3 \times 3$ neighborhood around each TSDF block b, we parameterize its estimated motion as a rigid transform $T(\cdot\,; \mathbf{r}, \mathbf{t})$ with translation $\mathbf{t} \in \mathbb{R}^3$ and Euler-angle rotation $\mathbf{r} \in \mathbb{R}^3$. Using the residuals

$$\varepsilon_i(\mathbf{r}, \mathbf{t}) = \hat{s}_{t-1}^{(b)}\left(T(\mathbf{p}_i; \mathbf{r}, \mathbf{t})\right) - s_t^{(b)}(\mathbf{p}_i), \tag{12}$$

we then minimize the squared residuals

$$\mathbf{t}, \mathbf{r} := \arg\min_{\mathbf{t},\mathbf{r}} \sum_{i \in \Omega} \varepsilon_i(\mathbf{r}, \mathbf{t})^2 \tag{13}$$

using a damped Gauss-Newton procedure [Levenberg 1944] to estimate $\mathbf{t}, \mathbf{r}$. In this algorithm, we iteratively estimate the Jacobian $J \approx \partial\boldsymbol{\varepsilon}/\partial[\mathbf{r}, \mathbf{t}]$ by finite difference and solve for the update step

$$\Delta\mathbf{p} = -\left(J^\top J + \lambda I\right)^{-1} J^\top \boldsymbol{\varepsilon}, \tag{14}$$

with a small damping λI for numerical stability. We also clamp the per-iteration translation and rotation update magnitudes to 1 voxel and 5° per step to prevent divergence and terminate early when the objective decrease falls below a threshold. Finally, if too few reliable samples are available or the $3 \times 3 \times 3$ neighborhood is incomplete, we fall back to zero-motion prediction $\mathbf{t}, \mathbf{r} = \mathbf{0}$ for that block.

3.6 Implementation

We implement the codec in C++ with CUDA acceleration [Nickolls et al. 2008]. The codebase is around 30K lines of code. All experiments are run on a Linux workstation with an Intel i9-13900K CPU, an NVIDIA RTX 4090 GPU, and 64 GB RAM.

Sparse TSDF representation and conversion. Each frame is represented as a sparse TSDF volume decomposed into voxel blocks. Active blocks are stored in a GPU-side spatial hash table using voxel hashing [Nießner et al. 2013], enabling fast random access and parallel block processing for encoding and decoding. Our implementation ingests mesh, point-cloud, and depth inputs via the conversion pipeline of Sec. 3.2 on the same voxel grid used by the codec. Meshes are first normalized to metric scale; watertight inputs use the closest-triangle signed distance (Eq. (8)), while non-watertight scan inputs fall back to the offset field (Eq. (9)).

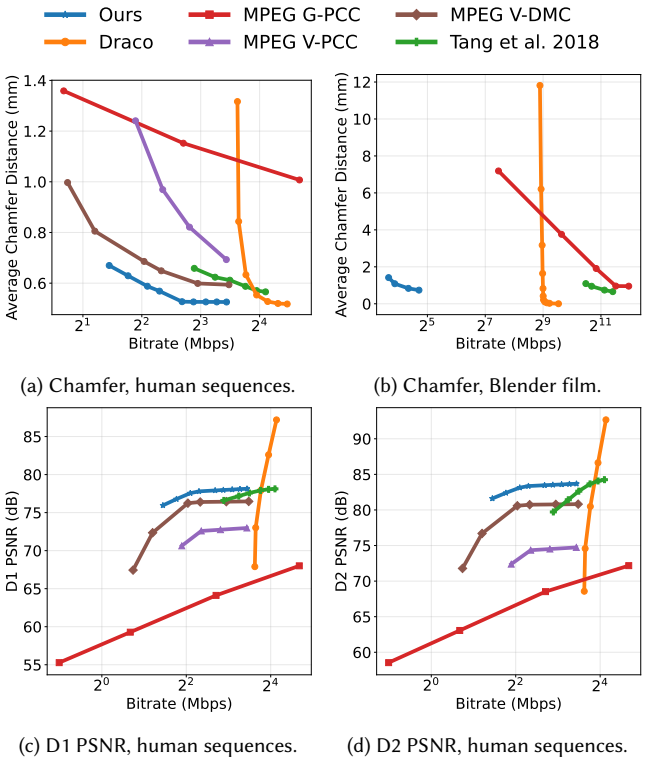
Projective texture mapping. For qualitative renderings on the V-SENSE and Owlii human performance dataset [Xu et al. 2017; Zerman et al. 2020], we simulate a multi-RGBD capture setup by placing 8 calibrated cameras around the subject in Blender and rendering per-frame RGB-D images. Geometry is reconstructed via the projective TSDF fusion of Sec. 3.2, and we apply standard projective texturing on GPU, similar to the setup used by Tang et al. [2018]. This pipeline assumes a static, surround-camera configuration and is therefore limited to object-scale sequences. It is not directly applicable to mobile-camera capture or room-scale scenes. We do not contribute new texture compression in this paper and leave it for future work.

User-tunable parameters. Our codec exposes a set of user-tunable parameters: voxel resolution, voxel block size, TSDF truncation band, DC/AC quantization steps, and the number of retained DCT coefficients. In our experiments, the voxel block size is fixed to 8×8×8, and the TSDF truncation band is set to one block width, i.e., 8 voxels. We use a voxel resolution of 5 mm for the human sequences and 1 cm for the Blender Film scenes.

4 Evaluation

4.1 Datasets and metrics

We evaluate on two dataset categories that cover both human-centric deforming surfaces and larger-scale dynamic scenes: 1) we use six dynamic human sequences, including four from the public V-SENSE dataset [Zerman et al. 2020] and two from the Owlii dataset [Xu et al. 2017]; V-SENSE is a common benchmark in human-centric volumetric media, while Owlii is used as a test case for the MPEG compression standard. Each frame in the V-SENSE and Owlii sequences contains approximately 400K and 40K faces, respectively; 2) to evaluate effectiveness and scalability on larger scenes (spanning roughly 10–20 meters), we use two publicly available Blender films (*Charge* and *Wing It* [Blender Studio 2022, 2023]). We select three scenes in total, each with 300 frames, and extract only prominent scene objects as mesh/point cloud sequence at 2 million faces/vertices.



(a) Chamfer, human sequences.

(b) Chamfer, Blender film.

(c) D1 PSNR, human sequences.

(d) D2 PSNR, human sequences.

Fig. 8. Rate–distortion curves (geometry-only bitrate, Mbps in log2 scale). (a,b) Average Chamfer distance in millimeters, lower is better. (c,d) Point-to-point (D1) and point-to-plane (D2) PSNR in dB on the human sequences, higher is better. MPEG V-PCC and V-DMC are omitted in (b) due to invalid reconstructions.

Metrics. We report ***geometry-only*** bitrate and measure geometric distortion using the symmetric Chamfer distance [Tang et al. 2020] between a reference point set P and a reconstructed point set Q:

$$\mathrm{CD}(P,Q) = \frac{1}{2|P|}\sum_{\mathbf{p}\in P}\min_{\mathbf{q}\in Q}\|\mathbf{p}-\mathbf{q}\|_2 + \frac{1}{2|Q|}\sum_{\mathbf{q}\in Q}\min_{\mathbf{p}\in P}\|\mathbf{q}-\mathbf{p}\|_2. \quad (15)$$

Chamfer distance is correspondence-free and applies uniformly to meshes and point clouds by evaluating both as point sets. We evaluate point-cloud methods directly. For mesh methods, we convert each mesh to a point set via area-weighted surface sampling. We sample points at a uniform resolution of 5 mm for the human sequence dataset and 1 cm for the Blender Film scenes, using a fixed random seed across methods, and report the mean over all frames. For the human sequences, we additionally report the point-to-point (D1) and point-to-plane (D2) Peak Signal-to-Noise Ratio (PSNR) following the standard MPEG evaluation metric [Graziosi et al. 2020].

4.2 Comparison to baselines

We compare against state-of-the-art geometry codecs across representations: TSDF coding (Tang et al. [2018])[2], mesh compression (Draco [Galligan et al. 2018] and MPEG V-DMC [Zou et al. 2025]), and point-cloud compression (MPEG G-PCC/V-PCC [Graziosi et al. 2020]). For all methods, we report *geometry-only bitrate*, excluding color attributes and texture data.

Fig. 8 summarizes rate–distortion performance using geometry-only bitrate and symmetric Chamfer distance, with MPEG-style point-to-point (D1) and point-to-plane (D2) PSNR additionally reported on the human sequences (Fig. 8c, 8d). On the human sequences (Fig. 8a), our TSDF codec consistently achieves lower bitrate at comparable distortion across the evaluated operating range. This gain holds against a TSDF-based intra-only baseline [Tang et al. 2018], indicating that our block transform design, combined with temporal prediction and motion-compensated alignment, produces a significantly more compressible residual. Compared with established mesh and point-cloud codecs (Draco, MPEG G-PCC/V-PCC/V-DMC), our method is particularly efficient at low bitrates while remaining competitive at higher rates, suggesting that directly coding the locally smooth TSDF signal is well matched to deforming human geometry.

The D1/D2 PSNR curves on the human sequences (Fig. 8c, 8d) show a consistent trend. Our method achieves comparable or higher point-to-point and point-to-plane PSNR than the competing codecs at substantially lower bitrates across the operating range. Tang et al. [2018] reaches similar PSNR only at higher rates, while Draco attains high PSNR primarily at much larger bitrates. MPEG G-PCC, V-PCC, and V-DMC remain less efficient in this regime. These results confirm that the gains observed under Chamfer distance are not specific to a single distortion metric, and also hold under the geometry quality measures used in the standard MPEG evaluation.

On the large-scale Blender Film scenes spanning approximately 10–20 meters (Fig. 8b), our codec remains rate-efficient, whereas conventional geometry codecs require substantially higher bitrates for similar distortion. We omit MPEG V-PCC and MPEG V-DMC in this setting, as their reconstructions are invalid due to inherent design assumptions: V-PCC relies on object-scale 2D atlas parameterization that degrades with scene extent, while V-DMC depends on deforming a tracked template mesh and becomes brittle in large environments. Our blockwise transform coding avoids atlas parameterization and template tracking altogether, enabling stable and robust compression in large-scale scenes.

Qualitative results. In addition to substantially lower bitrate, our method also produces qualitatively better reconstructions at equivalent Chamfer distances due to the intrinsic smoothing effect imposed by TSDFs. Unlike non-TSDF-based baselines, our method does not produce visually unpleasant blocky or jagged artifacts (Fig. 9, 12, 13). This contrast is especially apparent in large scenes, where baseline methods can achieve an equivalent Chamfer distance, but at the cost of much higher bitrate along with jagged artifacts (Fig. 11).

Runtime and memory comparison. Tab. 1 reports runtime and memory on the V-SENSE human sequence and the Blender Film dataset. Our codec delivers real-time encoding and decoding on V-SENSE (~400K polygons). In contrast, most other pipelines incur

[2]Neither Tang et al. [2018] nor Tang et al. [2020] provide reference code. We re-implemented Tang et al. [2018] for comparison, but do not include Tang et al. [2020] as it requires a substantially more complex end-to-end reproduction, including training and inference, beyond our evaluation scope.

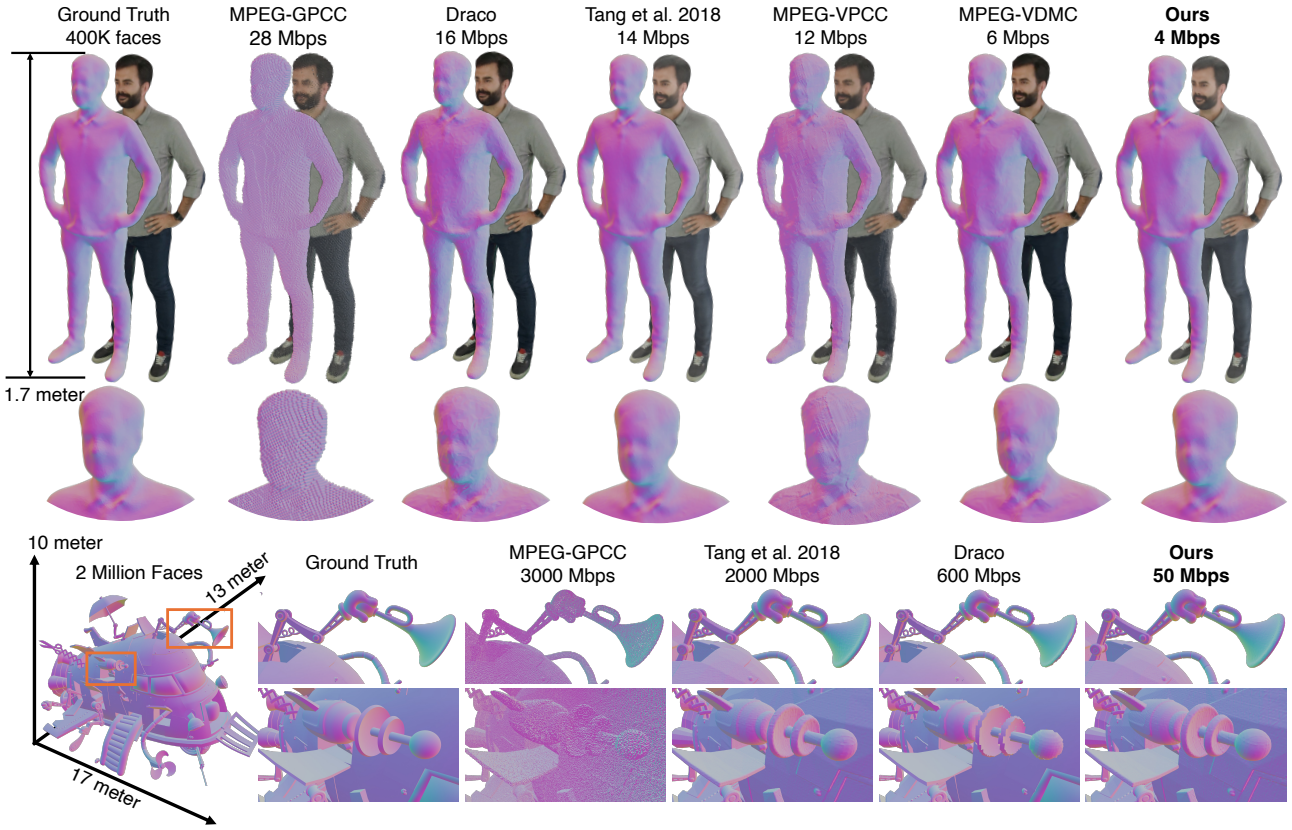


Fig. 9. Qualitative comparisons at similar average Chamfer distance. Top: V-SENSE human sequence (textured and normal-colored renderings). Bottom: Blender Film *Wing It* scene. Axis labels indicate the scene's real-world scale. Our codec preserves fine surface detail at substantially lower bitrates. TSDF-based pipelines (Tang et al. [2018] and ours) may appear smoother than the ground truth because converting an explicit mesh to a discretized TSDF and re-extracting the zero level set (via Marching Cubes) introduces implicit smoothing.

substantially higher latency due to heavier preprocessing and coding toolchains (e.g., patch/atlas construction and basemesh fitting). For Tang et al. [2018] and our method, the reported memory corresponds to GPU usage: both rely on sparse spatial hashing backends [Nießner et al. 2013] that trade a larger working set for higher throughput. On the Blender Film stress test (~2M polygons), our method requires more GPU memory to maintain the hashed TSDF working set, while still maintaining practical latency.

4.3 Impact of motion prediction

Zero-motion prediction (inter-frame residuals with predictor $\hat{s}_t = s_{t-1}$) yields large improvements over Intra-frame only compression, indicating substantial temporal redundancy even without motion estimation.

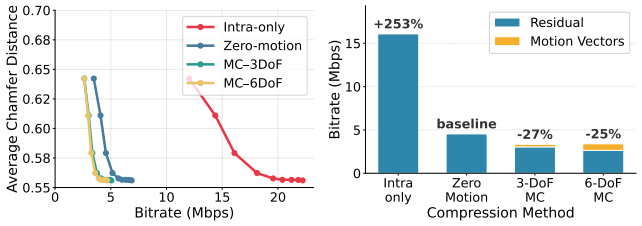
Applying motion compensation further improves rate-distortion performance by better aligning the previous TSDF frame s_{t-1} to predict the next frame $\hat{s}_t$, though zero-motion prediction already captures the majority of temporal redundancy (Fig. 10a).

However, although more powerful prediction by applying 6-DoF motion compensation instead of translation-only 3-DoF motion compensation can reduce residuals, it does not necessarily maximize net savings due to increased motion overhead. For example, with $K = 48$ DCT coefficients, 3-DoF motion compensation reduces total bitrate by **27%** with motion vectors taking **9%** of the total, whereas

Table 1. Average latency and memory footprint on the V-SENSE human sequence and the Blender Film (per frame, unless noted).

	Encode Latency		Decode Latency		Encode Memory	
Method	**V-SENSE**	**Blender**	**V-SENSE**	**Blender**	**V-SENSE**	**Blender**
Draco	60 ms	520 ms	30 ms	320 ms	14.3 MB	350 MB
Tang 2018	22 ms	1.6 s	15 ms	190 ms	6 GB	23 GB
V-PCC†	110 s	N/A	660 ms	N/A	600 MB	N/A
G-PCC	1.2 s	13 s	420 ms	1.3 s	100 MB	420 MB
V-DMC†	15.6 s	N/A	330 ms	N/A	0.9 GB	N/A
Ours	30 ms	2.1 s	21 ms	230 ms	6 GB	23 GB

†For MPEG V-PCC, V-DMC, memory is reported *per-frame*; peak memory increases over the Group-of-Frames due to retained reference state.



(a) Rate-Distortion curves. (b) Bit allocation at K=48.

Fig. 10. Impact of motion prediction. (a) RD curves for Intra-only, Zero-motion, and motion-compensated prediction (MC–3DoF/6DoF). (b) Residual vs. motion-vector bitrate at K=48, illustrating the tradeoff between improved prediction and motion side information.

6-DoF motion compensation reduces bitrate by **25%** but spends **22%** on motion vectors (Fig. 10b).

5 Conclusion

We presented a real-time 4D compression codec for implicit surfaces that effectively balances compression efficiency, generality, and speed. Our codec significantly reduces inter-frame redundancy without relying on content-specific training. We further compress the residuals using spatial DCT, enabling efficient encoding of dynamic scenes with varying topology. Our codec is lightweight, parallelizable, and suitable for streaming and real-time applications. Our experiments demonstrate strong compression ratios and reconstruction quality. Future work includes extending the codec to support hierarchical or adaptive block structures, integrating learned residual encoders, incorporating appearance compression, and exploring temporal transforms for longer-range dependencies in 4D sequences.

Acknowledgments

This work was supported by Bosch Corporate Research and in part by the National Science Foundation under Award IIS-2544317. We thank the anonymous reviewers for their constructive feedback. We are also grateful to Ioannis Gkioulekas for an insightful discussion, Yoshiki Takashima for suggesting the use of the Blender Film sequences to demonstrate the scalability of our codec, and Ankush Jain for saving the day with last-minute help in generating the `.bbl` file before the submission deadline.

References

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. 2006. Discrete cosine transform. *IEEE transactions on Computers* 100, 1 (2006), 90–93.

Apple. 2024. Apple Vision Pro. https://www.apple.com/apple-vision-pro/. Accessed: 2026-05-03.

Azure. 2019. Azure Kinect DK. https://azure.microsoft.com/en-us/services/kinect-dk/. Online; accessed May 2022.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. 2021. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5855–5864.

Blender Studio. 2022. *Charge (Project Page).* https://studio.blender.org/projects/charge/

Blender Studio. 2023. *Wing It! (Project Page).* https://studio.blender.org/projects/wing-it/

Aljaz Bozic, Pablo Palafox, Michael Zollhofer, Justus Thies, Angela Dai, and Matthias Nießner. 2021. Neural deformation graphs for globally-consistent non-rigid reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1450–1459.

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.

Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. 2024. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*. Springer, 370–386.

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.

Brian Curless and Marc Levoy. 1996. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 303–312.

Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. 2017. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics (ToG)* 36, 4 (2017), 1.

Ricardo L De Queiroz and Philip A Chou. 2017. Motion-compensated compression of dynamic voxelized point clouds. *IEEE Transactions on Image Processing* 26, 8 (2017), 3886–3895.

Jan Dvořák, Filip Hácha, Gerasimos Arvanitis, David Podgorelec, Konstantinos Moustakas, and Libor Váša. 2025. Survey of Inter-Prediction Methods for Time-Varying Mesh Compression. In *Computer Graphics Forum*, Vol. 44. Wiley Online Library, e15278.

Frank Galligan, Michael Hemmer, Ondrej Stava, Fan Zhang, and Jamieson Brettle. 2018. Google/draco: a library for compressing and decompressing 3d geometric meshes and point clouds.

Danillo Graziosi, Ohji Nakagami, Shinroku Kuma, Alexandre Zaghetto, Teruhiko Suzuki, and Ali Tabatabai. 2020. An overview of ongoing point cloud compression standardization activities: Video-based (V-PCC) and geometry-based (G-PCC). *APSIPA Transactions on Signal and Information Processing* 9 (2020), e13.

Kaiwen Guo, Peter Lincoln, Philip Davidson, Jay Busch, Xueming Yu, Matt Whalen, Geoff Harvey, Sergio Orts-Escolano, Rohit Pandey, Jason Dourgarian, et al. 2019. The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (ToG)* 38, 6 (2019), 1–19.

Harold Hotelling. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* 24, 6 (1933), 417.

Takeo Igarashi, Tomer Moscovich, and John F Hughes. 2005. As-rigid-as-possible shape manipulation. *ACM transactions on Graphics (TOG)* 24, 3 (2005), 1134–1141.

Lihan Jiang, Yucheng Mao, Linning Xu, Tao Lu, Kerui Ren, Yichen Jin, Xudong Xu, Mulin Yu, Jiangmiao Pang, Feng Zhao, et al. 2025. Anysplat: Feed-forward 3d gaussian splatting from unconstrained views. *ACM Transactions on Graphics (TOG)* 44, 6 (2025), 1–16.

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.

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

Maja Krivokuća, Philip A Chou, and Maxim Koroteev. 2019. A volumetric approach to point cloud compression–part ii: Geometry compression. *IEEE Transactions on Image Processing* 29 (2019), 2217–2229.

Kenneth Levenberg. 1944. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics* 2, 2 (1944), 164–168.

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.

Li Li, Zhu Li, Vladyslav Zakharchenko, Jianle Chen, and Houqiang Li. 2019. Advanced 3D motion prediction for video-based dynamic point cloud compression. *IEEE Transactions on Image Processing* 29 (2019), 289–302.

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. doi:10.1002/cav.319

G Nigel N Martin. 1979. Range encoding: an algorithm for removing redundancy from a digitised message. In *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, Vol. 2.

Jean-Eudes Marvie, Maja Krivokuća, and Danillo Graziosi. 2023. Coding of dynamic 3D meshes. In *Immersive Video Technologies*. Elsevier, 387–423.

Meta. 2026. Meta Quest 3. https://www.meta.com/quest/quest-3/. Accessed: 2026-05-03.

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.

Alexander Millane, Helen Oleynikova, Emilie Wirbel, Remo Steiner, Vikram Ramasamy, David Tingdahl, and Roland Siegwart. 2024. nvblox: Gpu-accelerated incremental signed distance field mapping. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2698–2705.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. 2015. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 343–352.

Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*. Ieee, 127–136.

John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. 2008. Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue* 6, 2 (2008), 40–53.

Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. 2013. Real-time 3D reconstruction at scale using voxel hashing. *ACM Transactions on Graphics (ToG)* 32, 6 (2013), 1–11.

Michael Oechsle, Songyou Peng, and Andreas Geiger. 2021. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *Proceedings of the IEEE/CVF international conference on computer vision*. 5589–5599.

Victor Adrian Prisacariu, Olaf Kähler, Stuart Golodetz, Michael Sapienza, Tommaso Cavallari, Philip HS Torr, and David W Murray. 2017. Infinitam v3: A framework for large-scale 3d reconstruction with loop closure. *arXiv preprint arXiv:1708.00783* (2017).

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. 2002. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics* 5, 1 (2002), 47–61.

Peter Schelkens, Adrian Munteanu, Joeri Barbarien, Mihnea Galca, Xavier Giro-Nieto, and Jan Cornelis. 2003. Wavelet coding of volumetric medical datasets. *IEEE Transactions on medical Imaging* 22, 3 (2003), 441–458.

Peter Schelkens, Adrian Munteanu, Alexis Tzannes, and Chris Brislawn. 2006. Jpeg2000. part 10. volumetric data encoding. In *2006 IEEE international symposium on circuits and systems*. IEEE, 4–pp.

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

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

Vivienne Sze, Madhukar Budagavi, and Gary J Sullivan. 2014. High efficiency video coding (HEVC). *Integrated circuit and systems, algorithms and architectures* 39 (2014), 40.

Danhang Tang, Mingsong Dou, Peter Lincoln, Philip Davidson, Kaiwen Guo, Jonathan Taylor, Sean Fanello, Cem Keskin, Adarsh Kowdle, Sofien Bouaziz, et al. 2018. Real-time compression and streaming of 4d performances. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–11.

Danhang Tang, Saurabh Singh, Philip A. Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G. Guleryuz, Yinda Zhang, Shahram Izadi, Andrea Tagliasacchi, Sofien Bouaziz, and Cem Keskin. 2020. Deep Implicit Volume Compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Varjo. 2026. Varjo XR-4 Series. https://varjo.com/products/xr-4. Accessed: 2026-05-03.

Gregory K Wallace. 1991. The JPEG still picture compression standard. *Commun. ACM* 34, 4 (1991), 30–44.

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689* (2021).

Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. 2025. Foundationstereo: Zero-shot stereo matching. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 5249–5260.

Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J Leonard, and John McDonald. 2015a. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *The International Journal of Robotics Research* 34, 4-5 (2015), 598–626.

Thomas Whelan, Stefan Leutenegger, Renato F Salas-Moreno, Ben Glocker, and Andrew J Davison. 2015b. ElasticFusion: Dense SLAM without a pose graph.. In *Robotics: science and systems*, Vol. 11. Rome, 3.

Hongyi Xu and Jernej Barbič. 2020. Signed distance fields for polygon soup meshes. In *Graphics interface 2014*. AK Peters/CRC Press, 35–41.

Yi Xu, Yao Lu, and Ziyu Wen. 2017. Owlii dynamic human mesh sequence dataset. In *ISO/IEC JTC1/SC29/WG11 m41658, 120th MPEG Meeting*, Vol. 1. 8.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. 2024. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10371–10381.

Emin Zerman, Cagri Ozcinar, Pan Gao, and Aljosa Smolic. 2020. Textured mesh vs coloured point cloud: A subjective study for volumetric video compression. In *Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*.

Wenjie Zou, Shizhuo Zhang, Fuzheng Yang, and Marius Preda. 2025. Standardization Status of MPEG Video-based Dynamic Mesh Coding (V-DMC). In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 1–5.

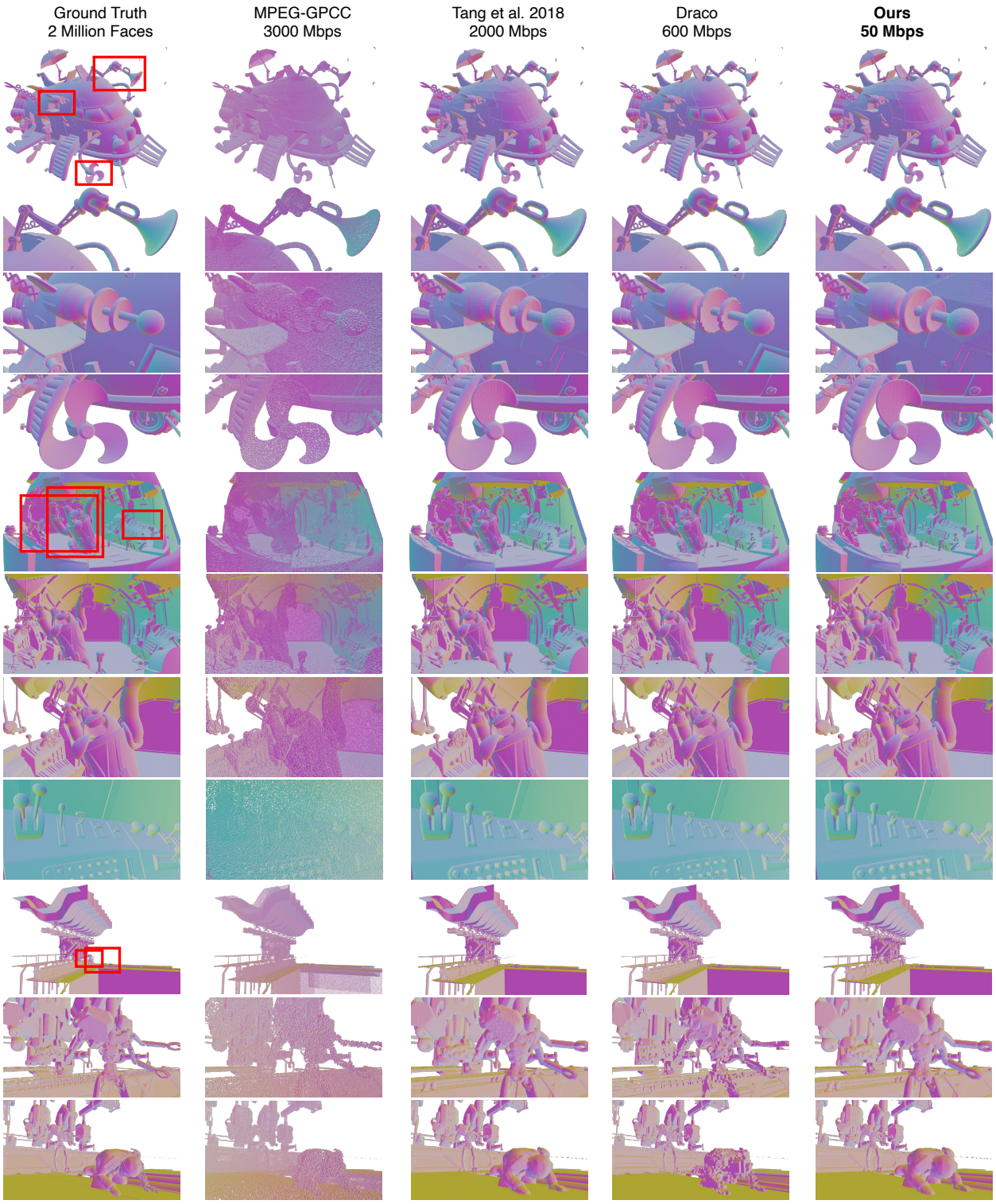


Fig. 11. Blender Film qualitative comparison (large-scale scenes). Normal-colored surface renderings for representative Blender Film scenes at comparable distortion, with zoomed-in crops highlighted in red. Bitrate is averaged across all scenes. For our method, we use a voxel resolution of 1 cm, retain $K = 48$ DCT coefficients, set $\Delta_{\mathrm{DC}} = 0.005$, and use an average AC quantization step of $\Delta_{\mathrm{AC}} = 0.01$.

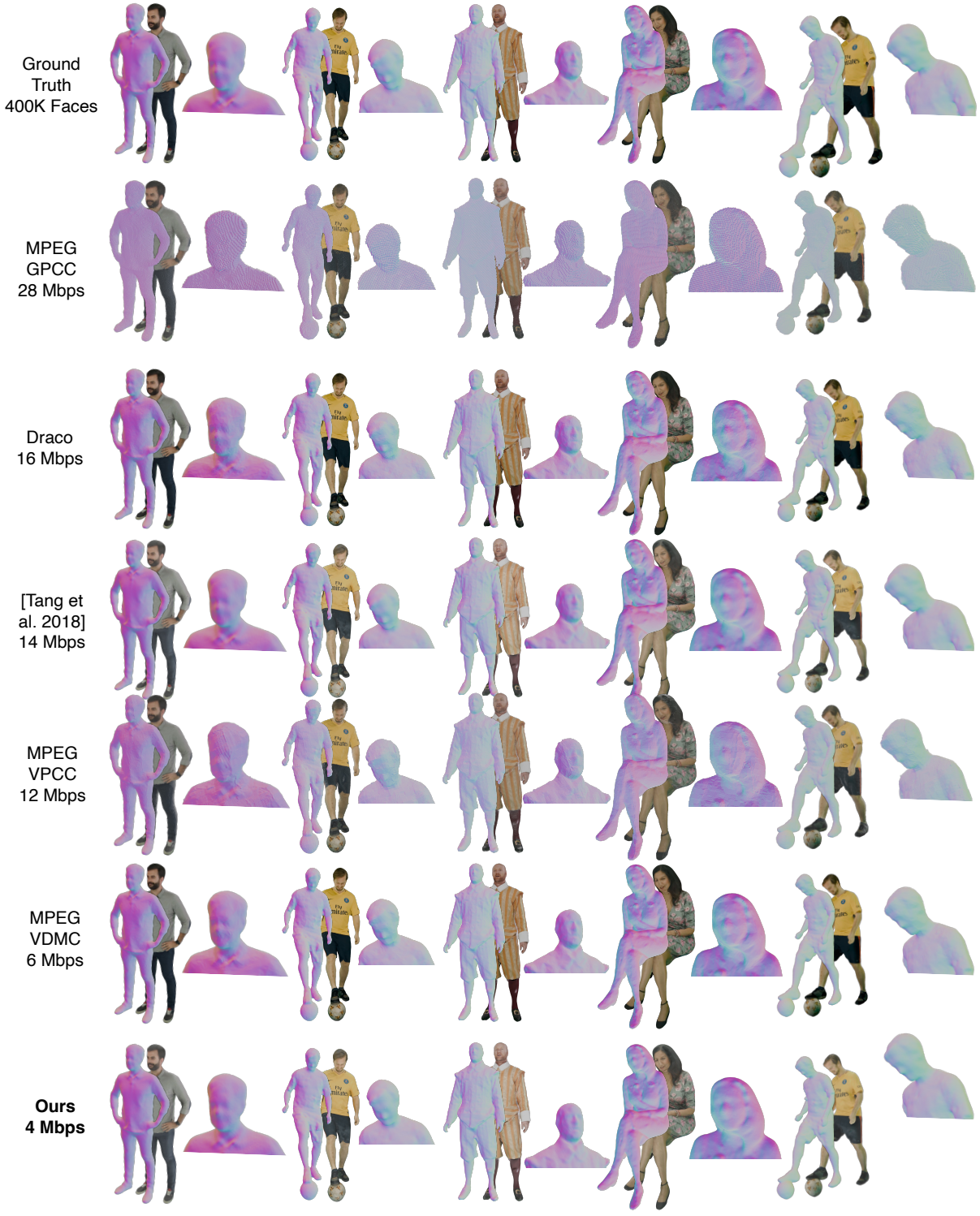


Fig. 12. V-SENSE Dataset qualitative comparison. Textured and normal-colored surface renderings for four V-SENSE human sequences (Rafa2, Matis, AxeGuy, LubnaFriends) at comparable distortion. Bitrate is averaged across all human sequences. For our method, we use a voxel resolution of 5 mm, retain $K = 64$ DCT coefficients, set $\Delta_{\mathrm{DC}} = 0.001$, and use an average AC quantization step of $\Delta_{\mathrm{AC}} = 0.01$.

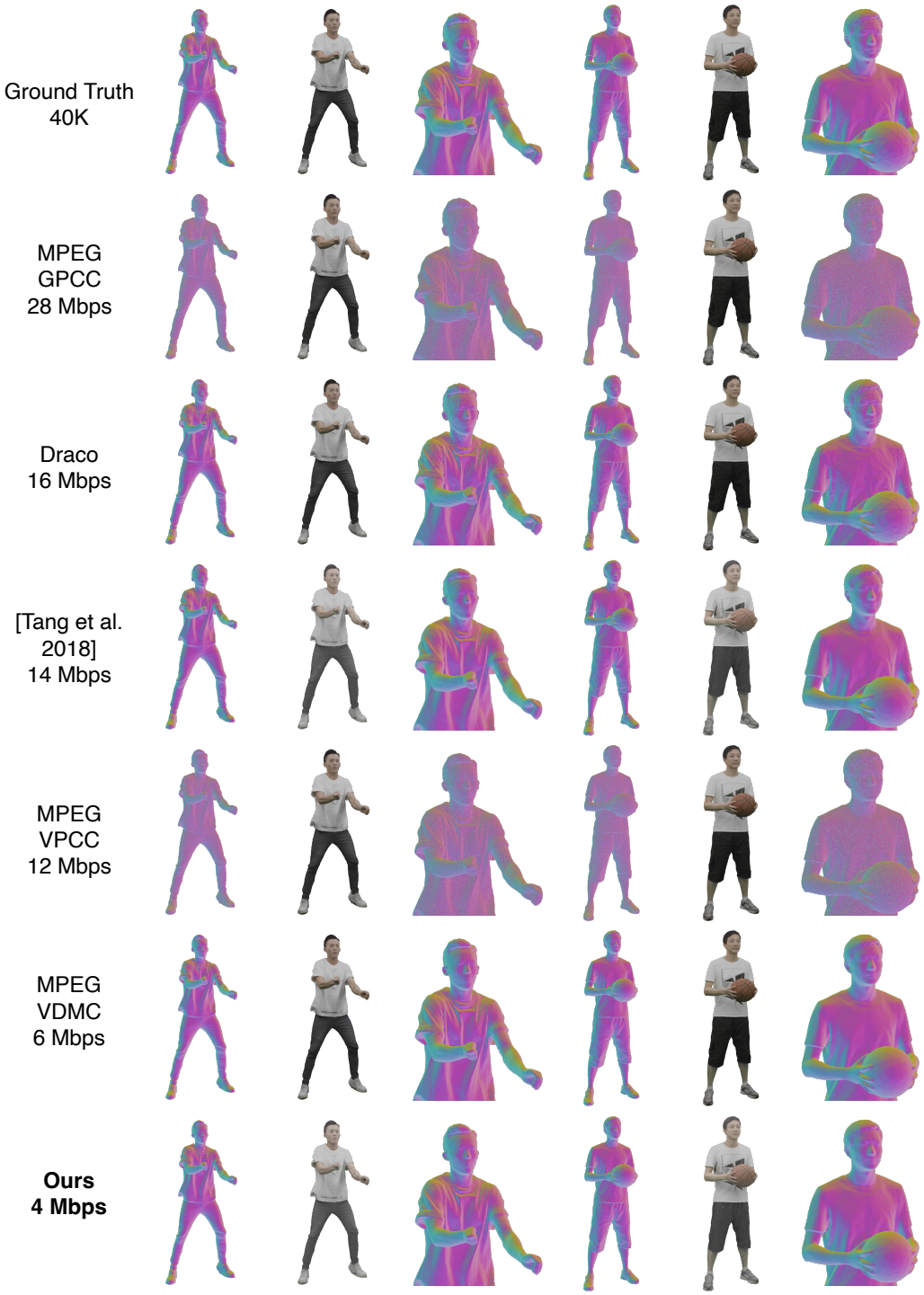


Fig. 13. Owlii Dataset qualitative comparison. Textured and normal-colored surface renderings for two Owlii human sequences (Basketball Player, Dancer) at comparable distortion. Bitrate is averaged across all human sequences. For our method, we use a voxel resolution of 5 mm, retain $K = 64$ DCT coefficients, set $\Delta_{DC} = 0.001$, and use an average AC quantization step of $\Delta_{AC} = 0.01$.