SIEVE: TOPOLOGY-FLEXIBLE MESH COMPRESSION VIA SURFACE-INTERSECTION CODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing implicit representation-based 3D mesh compression methods typically adopt signed distance fields representation, which can struggle to preserve open boundaries and thin structures. In this paper, we present **SIEVE**, **S**urface **I**ntersection **E**ncoding via **V**oxel **E**dges, a 3D mesh compression framework that encodes local surface intersections to accommodate diverse surface geometries and topologies. Its compact Dual Vertex–Edge (DV-Edge) representation stores grid-edge intersection flags and precomputed dual vertex positions on a sparse regular grid, encoding local geometry and the information needed to construct mesh connectivity. A hierarchical autoencoder compresses DV-Edge into an entropy-coded bitstream for mesh reconstruction. Extensive experiments on challenging scanned meshes and geometrically irregular surfaces demonstrate improved rate–distortion performance over the evaluated baselines, with better preservation of fine surface structures. The code will be publicly available.

1 INTRODUCTION

3D meshes are a primary representation for 3D assets in computer graphics, robotics, telepresence, and embodied perception. As modern reconstruction and generation systems produce increasingly high-resolution meshes, efficient transmission and storage become essential. A common approach to mesh compression explicitly encodes vertex positions and mesh connectivity (Maglo et al., 2015), as exemplified by Draco (Google, 2025) and Video-based Dynamic Mesh Coding (V-DMC) (MPEG, 2025; Zou et al., 2025). However, such methods can exhibit limited rate–distortion performance, particularly at low bitrates. Point cloud compression (PCC) methods (Schwarz et al., 2019; MPEG, 2025; Wang et al., 2023) can also be applied to meshes by sampling their surfaces into point clouds. However, point samples do not explicitly encode mesh connectivity, and reconstructing meshes from the decoded points can introduce additional distortion (Wang et al., 2026). Another approach compresses mesh geometry through implicit signed distance fields (SDFs) (Tang et al., 2020; Ren et al., 2025; Wang et al., 2026). By organizing distance samples into structured grids, these methods exploit spatial correlations for efficient coding.

However, converting meshes to signed distance fields and extracting surfaces from them can alter open boundaries and thin structures. To better preserve these features while retaining the benefits of structured coding, we propose SIEVE, a surface-aligned and topology-flexible mesh compression framework based on local surface intersections. Building on the line segment field (LSF) introduced in Shape as Line Segments (SALS) (Ren & Hou, 2025), we restrict its line-segment queries to the edges of a regular grid and derive a compact DV-Edge representation for mesh compression. DV-Edge stores grid-edge intersection flags and precomputed dual vertex positions only at active grid locations. Replacing intersection positions and normals with dual vertices reduces the attributes required for coding. These attributes describe local surface geometry and enable direct mesh reconstruction via dual contouring (Ju et al., 2002), as shown in Fig. 1.

We develop a neural codec tailored to the DV-Edge representation. The encoder maps DV-Edge attributes to a lower-resolution sparse latent feature. Quantized latent features are entropy-coded using a learned entropy model, while their grid coordinates are encoded separately. Then a decoder predicts active grid locations and reconstructs grid-edge intersection flags and dual vertex positions for direct mesh extraction. Experiments on real-world scans and synthetic meshes with diverse

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

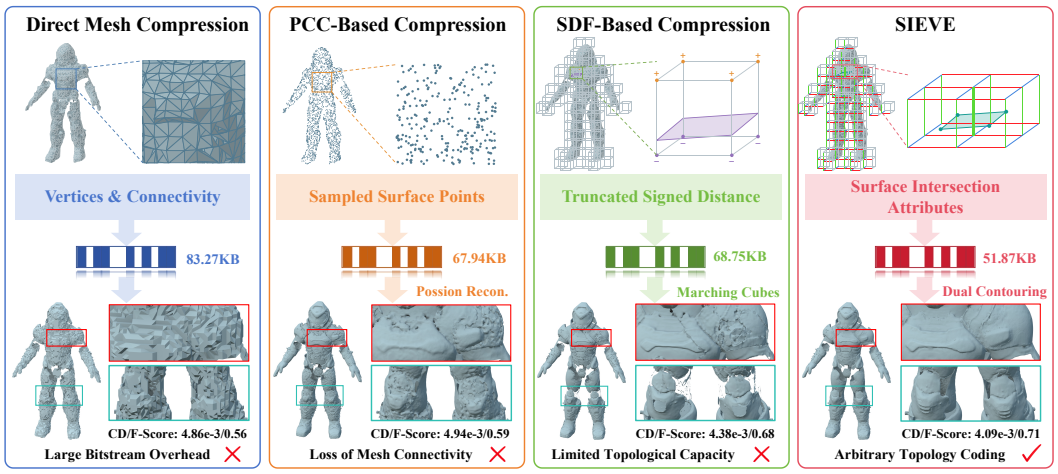

Figure 1: Comparison of SIEVE with other mesh compression paradigms. The methods shown in **(a)**, **(b)**, **(c)**, and **(d)** are **Draco** (Google, 2025), **G-PCC** (MPEG, 2025), **SIR-SNC** (Wang et al., 2026), and our proposed **SIEVE**, respectively. *Zoom in for details.*

topologies show that SIEVE achieves better rate–distortion (RD) performance than the evaluated baselines while maintaining efficient inference.

We summarize our contributions as follows.

- We derive DV-Edge, a compact representation for mesh compression that stores grid-edge intersection flags and precomputed dual vertex positions on a sparse grid.
- We develop a sparse neural codec that compresses DV-Edge through latent features and coordinates and reconstructs surface attributes for direct mesh extraction.
- Experiments on real-world scans and meshes with diverse topologies demonstrate that SIEVE achieves better rate–distortion performance than the evaluated baselines while maintaining efficient inference.

2 Related Work

Surface Representations. Surface representations differ in how they encode geometry and connectivity. Triangle meshes explicitly store vertices and faces, supporting open boundaries and complex geometric structures. Point clouds represent surfaces as discrete samples without explicitly encoding connectivity (Besl & McKay, 1992; Qi et al., 2017a;b). Geometry images (Gu et al., 2002; Zhang et al., 2023) organize surface coordinates on a 2D grid through parameterization, enabling the use of established image processing techniques. Many implicit representations encode occupancy or distance at spatial query points, including binary occupancy fields (BOFs) (Kazhdan & Hoppe, 2013; Mescheder et al., 2019), signed distance fields (SDFs) (Liu et al., 2021; Park et al., 2019; Atzmon & Lipman, 2020), and unsigned distance fields (UDFs) (Chibane et al., 2020; Ren et al., 2023). BOFs and SDFs can struggle to preserve open boundaries and thin structures, while extracting meshes from UDFs requires specialized procedures. By contrast, SALS (Ren & Hou, 2025) introduces the line segment field (LSF), which associates query segments with intersection flags and ratios to support flexible surface reconstruction.

Meshes can be reconstructed from sampled points using ball pivoting (Bernardini et al., 1999) or screened Poisson reconstruction (Kazhdan & Hoppe, 2013). The former directly connects samples into triangles; the latter reconstructs an implicit field before extracting a surface. Geometry images also allow mesh reconstruction through their 2D grid connectivity (Gu et al., 2002). Distance fields are constructed by evaluating point-to-triangle distances, with signed fields additionally requiring inside–outside classification. Conversely, Marching Cubes (Lorensen & Cline, 1998) extracts isosurfaces from field values, while Dual Contouring (Ju et al., 2002) computes cell vertices from intersection positions and normals. Among learning-based methods, Deep Marching Tetrahedra (Shen et al.,

2021) optimizes a deformable tetrahedral grid, while Neural Dual Contouring (Chen et al., 2022) predicts vertex positions and connectivity. MeshUDF (Guillard et al., 2022) and GeoUDF (Ren et al., 2023) extract meshes from UDFs. Beyond distance fields, SALS (Ren & Hou, 2025) extracts meshes from the LSF using edge-based dual contouring without global inside–outside labels.

Mesh Compression. Conventional mesh codecs compress vertex attributes and connectivity (Maglo et al., 2015). Edgebreaker (Rossignac, 1999) encodes mesh connectivity through triangle traversal. Draco (Google, 2025) combines connectivity coding with vertex attribute prediction and quantization. V-DMC (Zou et al., 2025) decomposes dynamic meshes into base meshes and subdivision displacement data, supporting video coding of displacement data. Geometry-image approaches (Gu et al., 2002) enable image coding of parameterized surfaces, although resampling and chart boundaries can affect reconstruction quality. Point cloud codecs can also be applied to samples from mesh surfaces. G-PCC (Schwarz et al., 2019) exploits hierarchical geometry coding, and SparsePCGC (Wang et al., 2023) learns multiscale occupancy distributions with sparse convolutions. These codecs do not encode the original mesh connectivity, so meshes must be reconstructed from the decoded points, potentially altering fine structures and boundaries.

SDF-based methods compress distance fields sampled on structured grids. Tang et al. (2020) compress truncated signed distance field (TSDF) volumes using an autoencoder and encode their signs losslessly to bound reconstruction error. NeCGS (Ren et al., 2025) augments TSDF volumes with deformation fields and jointly compresses a geometry set through quantized latent features and shared decoder parameters. SIR-SNC (Wang et al., 2026) stores SDF samples near the surface and compresses them using a sparse convolutional autoencoder, reducing the memory cost of dense volumes. In contrast, SIEVE compresses DV-Edge, a compact representation derived from the LSF that stores grid-edge intersection flags and dual vertex positions for direct mesh reconstruction.

3 Proposed Method

Given an input mesh, SIEVE organizes surface intersections with grid edges on a sparse grid and derives a compact DV-Edge parameterization. A sparse neural codec compresses DV-Edge, and the decoded attributes support direct mesh extraction. We review the line segment field (LSF) in Sec. 3.1, describe its sparse grid organization and DV-Edge parameterization in Sec. 3.2, and present the neural codec and training objective in Sec. 3.3.

3.1 Preliminaries: Line Segment Field (LSF)

Unlike SDFs, which encode signed distances at spatial query points, the line segment field (LSF) (Ren & Hou, 2025), describes a surface through its intersections with spatial line segments. Specifically, given a 3D surface $\mathcal{S}$ and a query line segment $\overline{\mathbf{uv}}$ in 3D space with endpoints $\mathbf{u}, \mathbf{v} \in \mathbb{R}^3$, their spatial relationship is characterized by two attributes:

- **Intersection flag** $o(\overline{\mathbf{uv}}) \in \{0, 1\}$**:** This binary indicator specifies whether the query line segment intersects the surface $\mathcal{S}$, taking the value 1 when an intersection exists and 0 otherwise.
- **Intersection ratio** $s(\overline{\mathbf{uv}}) \in [0, 1]$**:** When $o(\overline{\mathbf{uv}}) = 1$, this ratio specifies the location of the intersection point $\mathbf{p} \in \mathbb{R}^3$ along the segment. The point is obtained by linear interpolation between the endpoints:

$$\mathbf{p} = s(\overline{\mathbf{uv}})\mathbf{u} + (1 - s(\overline{\mathbf{uv}}))\mathbf{v}. \tag{1}$$

Together, the intersection flags and ratios define an implicit line segment field representation of the surface:

$$\mathcal{S} = \{s(\overline{\mathbf{uv}})\mathbf{u} + (1 - s(\overline{\mathbf{uv}}))\mathbf{v} \mid o(\overline{\mathbf{uv}}) = 1,\ \overline{\mathbf{uv}} \in \Omega\}, \tag{2}$$

where Ω denotes the set of all spatial query line segments.

Mesh Extraction. An explicit triangle mesh can be extracted from the LSF using edge-based dual contouring (E-DC). Evaluating the field along grid edges provides the intersection flags and positions. The unoriented surface normal at an intersection point is parallel to the gradient of the intersection ratio with respect to a segment endpoint, i.e., $\mathbf{n} \parallel \nabla_{\mathbf{u}} s$. Together, the intersection positions and normals serve as Hermite data for computing a dual vertex for each grid cell containing

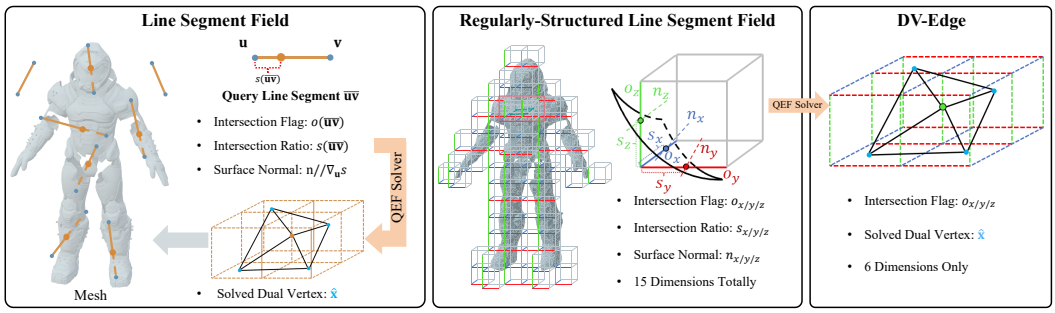

Figure 2: Illustration of the representation used in SIEVE. Starting from the original LSF, we derive regularly-structured LSF that encodes surfaces through their intersections with grid edges. To further facilitate compression, we combine dual vertices with edge intersection flags to obtain a more compact DV-Edge formulation, reducing the dimensionality from 15 to 6 while avoiding error accumulation from QEF solving.

intersected edges. Specifically, let $\mathcal{E} = \{\overline{\mathbf{u}_i\mathbf{v}_i}\}_{i=1}^{12}$ denote the edges of a grid cell, and let o_i indicate whether the i-th edge intersects the surface. For each intersected edge, $\mathbf{p}_i$ and $\mathbf{n}_i$ denote the intersection position and its associated unit normal, respectively. The dual vertex $\hat{\mathbf{x}}$ is computed by minimizing a quadratic error function (QEF):

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \sum_{\overline{\mathbf{u}_i\mathbf{v}_i}\in\mathcal{E}} \left(o_i(\mathbf{x}-\mathbf{p}_i)^\top \mathbf{n}_i\right)^2, \tag{3}$$

where o_i ensures that only intersected edges contribute to the objective. Each contributing term measures the squared distance from $\mathbf{x}$ to the tangent plane defined by $\mathbf{p}_i$ and $\mathbf{n}_i$, so the resulting dual vertex fits the local surface geometry.

Once the dual vertices have been computed, E-DC connects the vertices of the four grid cells incident to each intersected grid edge to form a quadrilateral. In SALS, the intersection point on that edge is then used to subdivide the quadrilateral into four triangles. Repeating this construction over the intersected grid edges produces the reconstructed triangle mesh.

3.2 Structured and Compact Surface Intersection Representation

In this section, we organize local surface-intersection attributes on a sparse grid and derive the compact DV-Edge representation for mesh compression, as shown in Fig. 2.

Regularly-Structured LSF. While the continuous LSF supports queries along arbitrary spatial line segments, it does not explicitly specify a regular and compact data structure for storing the intersection attributes. We therefore introduce two key modifications: **1**) we restrict LSF queries to grid edges aligned with the three coordinate axes; and **2**) we retain only active grid locations, defined as those with at least one associated grid edge intersecting the input surface.

Specifically, given a 3D asset, we represent it as a collection of feature tuples associated with sparse grids on a regular 3D grid of resolution $N \times N \times N$, as

$$\mathcal{F} = \{(f_i^{\text{int}}, p_i)\}_{i=1}^{L}, \tag{4}$$

where $p_i \in \{0, 1, \ldots, N-1\}^3$ denotes the coordinate of the i-th active grid. Here, L denotes the number of active grid locations. The geometric feature $f_i^{\text{int}} = (o_i, s_i, n_i)$ encodes the local intersection information along the three predefined grid edges, where:

- $o_i \in \{0, 1\}^3$ denotes the intersection flags for the X, Y, and Z edges, determining the topological connectivity.
- $s_i \in [0, 1]^3$ denotes the intersection ratios, precisely localizing the intersection points on the three edges.
- $n_i \in \mathbb{R}^{3\times 3}$ denotes the surface normals at the intersection points, which are critical for solving the QEF during mesh extraction.

To reconstruct the mesh surface from these intersection attributes, we employ E-DC from SALS. For each grid containing intersected edges, we solve a QEF to determine the optimal dual vertex position x_i and extract the mesh as in SALS.

The resulting representation supports conversion to and from triangle meshes. Its regular grid structure provides consistent spatial neighborhoods for processing local intersection attributes, while sparse storage concentrates geometric information on the mesh surface, yielding a compact representation for compression.

Compact DV-Edge Parameterization. In the reconstruction described above, intersection ratios and surface normals are used to compute dual vertex positions. To avoid reconstructing these intermediate attributes during decompression, we instead solve the QEF during representation construction and retain the resulting dual vertex positions together with the intersection flags. This yields the compact DV-Edge parameterization, in which the geometric feature associated with the i-th grid changes from f_i^{int} to $f_i^{\text{dv-edge}}$:

$$f_i^{\text{dv-edge}} = (o_i, x_i). \tag{5}$$

This conversion not only reduces the number of parameters per active grid from fifteen to six but also enables direct regression of the dual-vertex coordinate, bypassing the intermediate QEF solve during decompression and reducing numerical error accumulation. The entire SIEVE compression framework is built around this representation, with each component described in the following section.

3.3 Neural Compressor for DV-Edge

To compress this compact DV-Edge representation into a bitstream, we develop a sparse neural codec tailored to its structure, as shown in Fig. 3.

Encoder. Given $\{(f_i^{\text{dv-edge}}, p_i)\}_{i=1}^{L}$, sparse convolutional residual blocks extract spatial features. Two downsampling blocks combine sparse convolutions with spatial-to-channel rearrangement, each halving the resolution. The encoder $\mathcal{E}_\varphi$ produces

$$\mathcal{Y} = \mathcal{E}_\varphi\big(\{(f_i^{\text{dv-edge}}, p_i)\}_{i=1}^{L}\big) = \{(y_j, c_j)\}_{j=1}^{M}, \tag{6}$$

where $y_j \in \mathbb{R}^C$ and $c_j \in \mathbb{Z}^3$ denote latent feature vectors and their integer coordinates on the coarser grid, respectively, and M is the number of active latent sites.

Entropy model. Encoded latent features are quantized as $\widehat{y}_j = \text{round}(y_j)$. We adopt the same learned factorized entropy model as in SIR-SNC (Wang et al., 2026), which factorizes the likelihood of each latent vector into per-channel unit-bin likelihoods. At inference, the learned model arithmetic-codes the features $\widehat{y}_j$ and G-PCC geometry coding (Schwarz et al., 2019) transmits the coordinates c_j losslessly. Reported rates include both bitstreams.

During training, quantization is approximated by $\widetilde{y}_j = y_j + u_j$, where each component of u_j follows the uniform distribution $\mathcal{U}(-\frac{1}{2}, \frac{1}{2})$. The latent rate loss under the entropy model p_ψ is

$$\mathcal{L}_{\text{rate}} = -\sum_{j=1}^{M} \log_2 p_\psi(\widetilde{y}_j). \tag{7}$$

Decoder. The decoder receives the quantized latent features and their coordinates recovered from the two bitstreams. To reconstruct the DV-Edge representation, two sparse convolutional upsampling blocks expand each parent latent grid into eight child grids through channel-to-spatial rearrangement and each upsampling block doubles the resolution. Residual blocks refine the child features, and a multilayer perceptron (MLP) predicts occupancy probabilities $\widehat{t}_j^{(l)}$. The final decoder layer predicts edge probabilities $\widehat{o}_i$ and local dual vertex coordinates $\widehat{x}_i$.

During inference, we select upsampled active grids and edge intersections by thresholding the predicted probabilities $\widehat{t}_j^{(l)}$ and $\widehat{o}_i$ at τ_u and τ_e, respectively, with both thresholds set to 0.5. Mesh reconstruction connects the decoded dual vertices in neighboring grid cells according to the predicted edge-intersection flags.

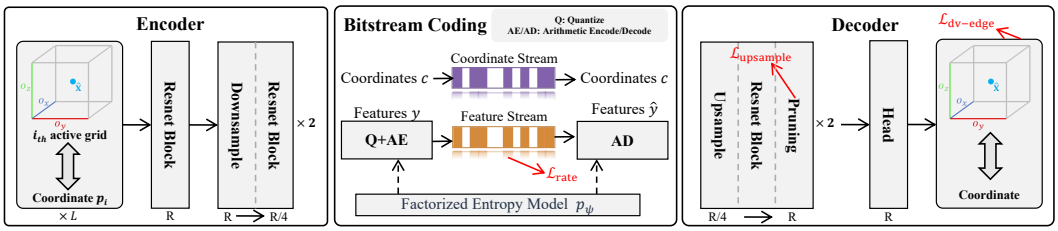

Figure 3: The pipeline of SIEVE. The encoder downsamples the DV-Edge representation of a 3D mesh to obtain sparse latent features and their coordinates. The latent features are encoded into a bitstream using an entropy model, while their coordinates are losslessly compressed via arithmetic coding. The decoder receives both bitstreams and upsamples the decoded latents to reconstruct DV-Edge at the same resolution as the input.

During training, we apply early pruning using ground-truth masks to retain features only at valid ground-truth locations. The occupancy predictions are supervised using the following loss:

$$\mathcal{L}_{\text{upsample}} = \sum_{l=1}^{2} \frac{1}{|\mathcal{C}_l|} \sum_{j \in \mathcal{C}_l} \text{BCE}(\hat{t}_j^{(l)}, t_j^{(l)}), \tag{8}$$

where $\mathcal{C}_l$ is the candidate set for the l-th upsampling level and $t_j^{(l)}$ its occupancy target.

The predicted edge probabilities and local dual vertex coordinates are supervised by o_i and x_i, respectively, using the following loss:

$$\mathcal{L}_{\text{dv-edge}} = \frac{1}{L} \sum_{i=1}^{L} (0.1\,\text{BCE}(\hat{o}_i, o_i) + 0.3\,\text{SmoothL1}(\hat{x}_i, x_i)). \tag{9}$$

Total Training Objective. The overall training objective combines the three losses:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{upsample}} + \lambda_2 \mathcal{L}_{\text{dv-edge}} + \lambda_3 \mathcal{L}_{\text{rate}}, \tag{10}$$

where $\lambda_1 = 0.1$, $\lambda_2 = 1.0$ and $\lambda_3 = 0.005$ weight each term to balance reconstruction quality and rate.

4 Experiments

4.1 Experimental Setup

Implementation Details. We use 32-dimensional latent features and set the output feature dimension at each downsampling and upsampling stage to 32. Following SIR-SNC (Wang et al., 2026), we train our model at a spatial resolution of 256. We use FlexGEMM (Xiang et al., 2026) as the backend for sparse 3D convolutions. We train the model for 50 epochs using Adam (Kingma & Ba, 2015) with a constant learning rate of 10^{-4}. At inference, we use the same trained model at input grid resolutions ranging from 192 to 768 to obtain different RD operating points.

Datasets. We train SIEVE on 3,000 meshes, with 1,000 from each of AMA (Vlasic et al., 2008), DT4D (Li et al., 2021), and Thingi10K (Zhou & Jacobson, 2016). We evaluate SIEVE on five datasets: ShapeNet (Chang et al., 2015), ScanNet (Dai et al., 2017), AOM (VVM Working Group, 2023), RWTT (Maggiordomo et al., 2020), and Objaverse (Deitke et al., 2023). ShapeNet contains synthetic CAD models, while ScanNet provides real-world scene scans. AOM contains textured meshes from multiple sources, covering human subjects, objects, buildings, and indoor scenes. RWTT comprises real-world objects and scenes reconstructed through various photogrammetry pipelines. Objaverse spans diverse categories and styles, including real-world scans and artist-created assets. We randomly select 30 meshes from each of the five evaluation datasets for evaluation. For brevity, we report combined AOM and RWTT results as *AOM + RWTT*, with other datasets reported separately.

Evaluation Metrics. We evaluate reconstruction quality using Chamfer distance (CD) and F-score, and additionally report normal consistency (NC) in the ablation studies. We compare RD curves to assess compression performance and report parameter counts and inference times to evaluate model size and computational efficiency.

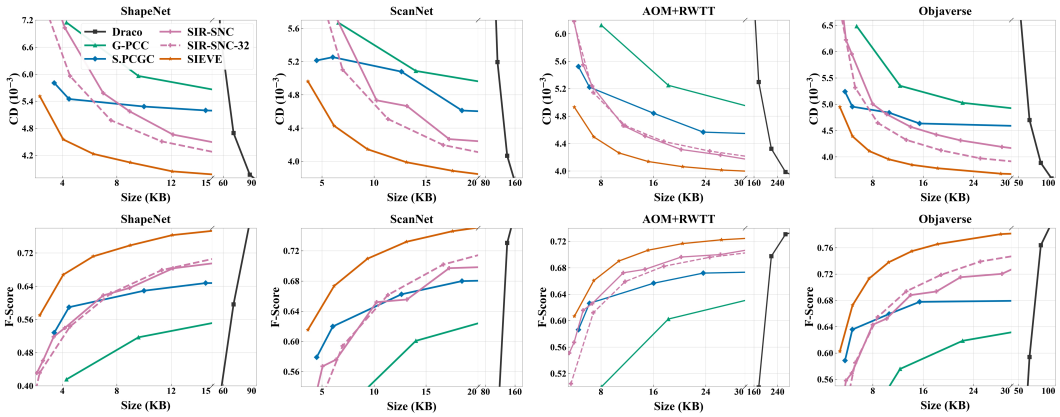

Figure 4: Rate–distortion comparisons of different compression methods. S.PCGC denotes SparsePCGC. Breaks in the horizontal axes accommodate the larger bitstreams produced by Draco.

Baselines. We compare SIEVE with Draco (Google, 2025) for direct mesh compression, G-PCC (MPEG, 2025) and SparsePCGC (Wang et al., 2023) for point-cloud-based compression, and SIR-SNC (Wang et al., 2026) for implicit-field-based compression. For G-PCC and SparsePCGC, we sample one million surface points per mesh and quantize their coordinates to 10 bits before compression. After decoding, we estimate normals and apply Poisson surface reconstruction (Kazhdan & Hoppe, 2013). To suppress unsupported surface regions, we remove reconstructed vertices farther than 0.02 from the decoded point cloud in normalized coordinates. To assess whether SIEVE's advantage over SIR-SNC stems from a larger model, we increase the channel width of SIR-SNC from 16 to 32. The resulting variant, denoted SIR-SNC-32, has approximately the same number of parameters as SIEVE (1.73M) and is trained on the same data with the same optimization settings.

4.2 Results and Comparisons

Compression Performance Comparison. As shown in Fig. 4, SIEVE achieves lower CD and higher F-score at comparable compressed sizes. Its advantage is particularly evident in the low-to-medium bitrate range, where Draco requires substantially larger bitstreams to achieve similar reconstruction quality. Over the shared compressed-size ranges, the point-cloud-based methods G-PCC and SparsePCGC also yield higher CD and lower F-score than SIEVE. SIR-SNC relies on an SDF-based representation, whose limitations in preserving open boundaries and thin structures can constrain reconstruction fidelity. SIEVE achieves better rate–distortion performance than SIR-SNC across the evaluated datasets. Table 1 further quantifies these gains, with SIEVE achieving BD-BR savings of 53–74% over SIR-SNC for CD and 35–62% for F-score. SIEVE also retains substantial BD-BR savings relative to the parameter-matched SIR-SNC-32.

Table 1: Bjøntegaard delta bitrate (BD-BR, %) of SIEVE relative to each baseline, reported as CD/F-score. Negative values indicate bitrate savings.

Methods	ShapeNet	ScanNet	AOM + RWTT	Objaverse
Draco	−92.8/−92.3	−93.7/−93.4	−95.8/−95.7	−90.9/−90.5
G-PCC	−93.2/−92.1	−92.4/−88.1	−91.8/−86.8	−94.1/−90.5
SparsePCGC	−88.6/−78.3	−79.2/−62.5	−77.5/−53.8	−80.1/−53.9
SIR-SNC	−70.3/−60.5	−59.1/−56.8	−53.9/−35.4	−73.8/−61.1
SIR-SNC-32	−60.7/−60.0	−50.2/−50.8	−55.9/−44.7	−60.2/−55.2

The visual comparisons in Fig. 5 further illustrate these advantages. SIEVE better preserves shape fidelity (e.g., the lamp in ScanNet and the sculpture in RWTT), the topology of slender structures (e.g., the fins in AOM), and complex local geometry (e.g., the hinge in ShapeNet). Draco produces severe quantization artifacts, while G-PCC and SparsePCGC tend to smooth, thicken, or distort fine structures during mesh reconstruction. SIR-SNC exhibits missing geometry in some of these regions, particularly around thin structures. SIEVE retains clearer boundaries and more coherent local geometry while using the smallest bitstreams among the illustrated examples. These observations are consistent with the design of DV-Edge, which explicitly encodes local surface-intersection attributes.

Efficiency Comparison. Table 2 compares parameter counts and encoding and decoding times across methods. All timing measurements are performed on an Intel Xeon Gold 6530 CPU, with neural codecs additionally using a single NVIDIA RTX PRO 6000 GPU. SIEVE achieves the short-

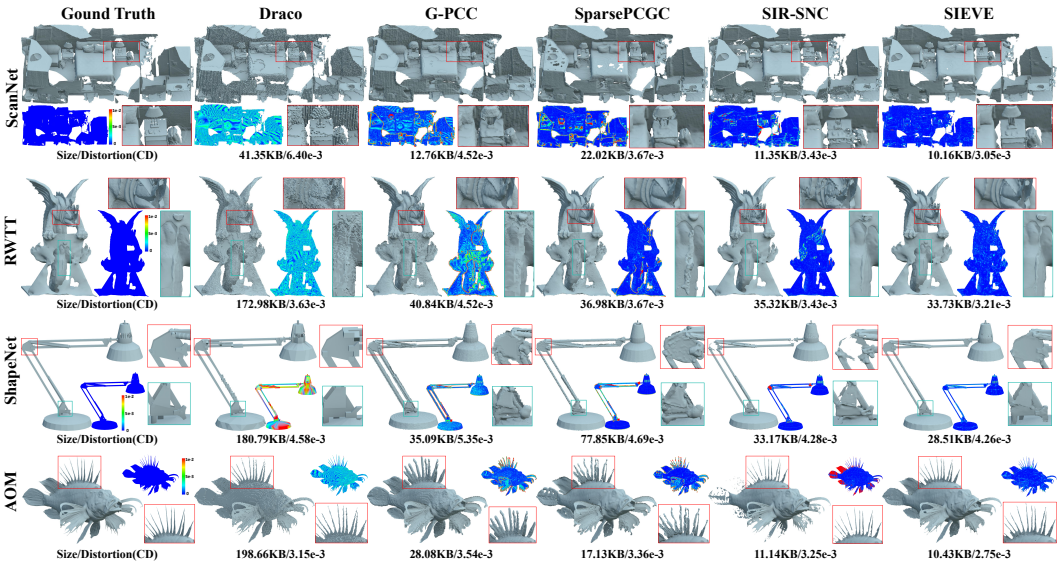

Figure 5: Visual comparisons of different compression methods. The annotations report compressed mesh size and Chamfer Distance. In the error heatmaps, blue indicates the lowest surface error, while red indicates the highest surface error (0.01) or missing structures. *Zoom in for details.*

Table 2: Efficiency comparisons of different compression methods. SIEVE* uses MinkowskiEngine in place of FlexGEMM for sparse convolutions. "–" denotes methods without learned parameters.

Metrics	Draco	SparsePCGC	G-PCC	SIR-SNC	SIR-SNC-32	SIEVE	SIEVE*
Parameters (M)	–	2.96	–	**0.43**	1.73	1.73	1.73
Encoding (s)	0.52	2.79	1.92	2.51	2.64	**0.39**	0.42
Decoding (s)	**0.10**	48.08	11.24	0.17	0.29	0.15	0.20

est end-to-end encoding time among the compared methods, at 0.39 s, covering all encoding stages: mesh-to-DV-Edge conversion, the encoder forward pass, and bitstream coding. Its decoding time, including mesh extraction, is 0.15 s, making it second only to Draco in decoding speed. Table 5 in the appendix provides a detailed runtime breakdown. However, compared to SIEVE, Draco's reconstruction quality degrades more sharply at low bitrates in the RD evaluation.

For a fairer comparison, we replace SIEVE's sparse convolution backend (i.e., FlexGEMM) with MinkowskiEngine (Choy et al., 2019), which is the same backend used by SIR-SNC. This variant, denoted as SIEVE*, retains the same architecture and model weights. Although its encoding and decoding times increase to 0.42 s and 0.20 s, respectively, SIEVE* still remains highly efficient, encoding faster than all compared baselines while maintaining competitive decoding speed.

Representation Fidelity Comparison. To isolate representation-induced distortion, we convert each input mesh into the intermediate representation used by each compression method (i.e., point clouds for G-PCC and SparsePCGC, TSDF for SIR-SNC, and DV-Edge for SIEVE) and reconstruct the mesh directly from that representation, bypassing the learned codec and entropy coding. For point clouds, we use the sampling and reconstruction settings of the point cloud compression baselines. For TSDF in SIR-SNC and DV-Edge in SIEVE, we use their respective mesh extraction procedures at resolutions 256 and 512, and compare their active-grid counts at each resolution. Draco is excluded because it does not use an intermediate surface representation. We also report surface CD (Surf. CD), defined as the symmetric mean distance from points sampled on each mesh to the triangle surface of the other mesh. This metric reduces the influence of finite target-point sampling and provides a more direct measure of representation-induced geometric distortion.

Table 3: Reconstruction fidelity of different representations. CD and Surf. CD are reported in units of 10^{-3}. Sites refer to sampled points for point clouds and active grids for TSDF and DV-Edge.

Representation	CD↓	Surf. CD↓	F-score↑	NC↑	# Sites↓
Points + Poisson	4.38	1.14	0.69	0.91	1000k
TSDF, $R = 256$	4.55	1.35	0.66	0.85	348k
DV-Edge, $R = 256$	3.68	0.29	**0.78**	0.92	**127k**
TSDF, $R = 512$	3.92	0.49	0.74	0.88	1461k
DV-Edge, $R = 512$	**3.62**	**0.10**	**0.78**	**0.93**	530k

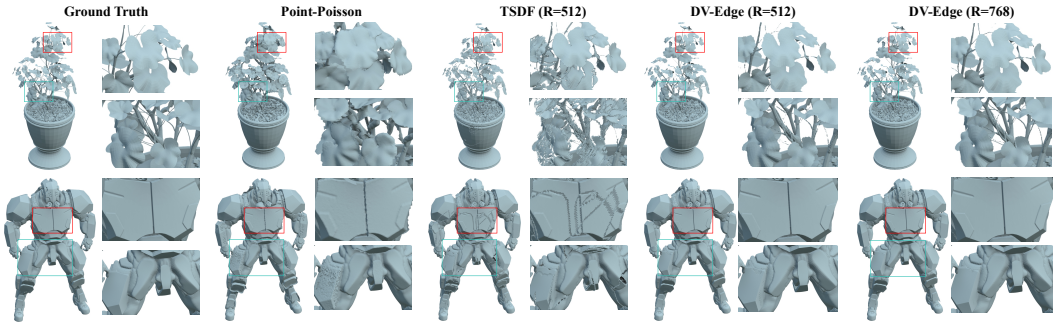

Figure 6: Visual comparison of meshes reconstructed from different representations. R denotes the grid resolution. TSDF and DV-Edge are compared at $R = 512$; an additional DV-Edge result at $R = 768$ illustrates the effect of increased resolution.

Table 3 shows that our DV-Edge representation yields lower reconstruction distortion than the evaluated point-cloud and TSDF representations. Compared with TSDF, it reduces surface CD by approximately 79% and 80% at resolutions 256 and 512, respectively, while using only about 36–37% as many active sites. This compact support follows from storing surface-intersecting cells rather than a band of distance samples around the surface, reducing the number of spatial locations processed by sparse convolutions. Fig. 6 further shows that reconstruction from DV-Edge better preserves open boundaries and fine structures than the evaluated Poisson- and TSDF-based pipelines. DV-Edge thus provides a compact, high-fidelity representation for subsequent compression.

4.3 Ablation Studies

Effect of DV-Edge Parameterization. We compare DV-Edge with the original intersection-based parameterization of regularly-structured LSF under the same training settings. As shown in Fig. 7, DV-Edge achieves better rate–distortion performance across the tested resolutions. Beyond reducing the per-grid feature dimensionality from 15 to 6, DV-Edge encodes precomputed dual vertices, avoiding error propagation from decoded intersection ratios and normals through the QEF solve.

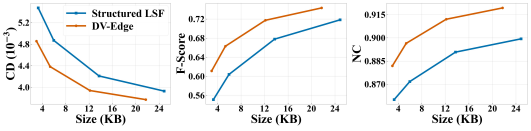

Figure 7: Rate–distortion comparison of DV-Edge and the original regularly-structured LSF.

Decoder Thresholds. We investigate the effects of different decoder thresholds τ_u and τ_e while keeping the compressed bitstream unchanged. Table 4 reports the quantitative results. The example in Fig. 10 in the appendix illustrates how lower thresholds can create spurious geometry, whereas higher thresholds can introduce surface holes. The default setting $\tau_u = \tau_e = 0.5$ performs best across all three metrics.

Table 4: Ablation study on the decoder threshold.

τ_u	τ_e	CD (10^{-3}) ↓	F-score ↑	NC ↑
0.1	0.1	5.06	0.5824	0.7365
0.3	0.3	4.62	0.6335	0.8347
0.5	0.5	**4.38**	**0.6630**	**0.8965**
0.7	0.7	4.98	0.6602	0.8774
0.9	0.9	5.90	0.6366	0.7935

5 Conclusion

We presented SIEVE, a mesh compression framework combining structured surface-intersection representation with sparse neural coding. Building on line segment field , the compact DV-Edge representation stores edge intersections and local dual vertices, enabling high-fidelity mesh reconstruction. A hierarchical sparse codec compresses this representation into an entropy-coded bitstream. Experiments on synthetic meshes and real-world scans demonstrate improved rate–distortion performance over the evaluated baselines and efficient inference, with better preservation of thin structures, open boundaries, and fine shape features.

AI USE STATEMENT

Generative AI tools, including large language models (LLMs), were used to assist with manuscript writing and language refinement, including sentence rephrasing, grammar checking, and improvements to readability and clarity, as well as the preparation of statistical summaries of experimental results. These tools were not used to develop the research ideas, design the methodology, or conduct the experiments. All AI-assisted text and statistical summaries were reviewed and verified by the authors, who take full responsibility for the accuracy, originality, and integrity of the manuscript, including the interpretation of results and the conclusions.

REFERENCES

Matan Atzmon and Yaron Lipman. Sal: Sign agnostic learning of shapes from raw data. In *Proc. CVPR*, pp. 2562–2571, 2020. 2

Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 2

Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pp. 586–606, 1992. 2

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 6

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. Neural dual contouring. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 3

Julian Chibane, Gerard Pons-Moll, et al. Neural unsigned distance fields for implicit function learning. *Proc. NeurIPS*, 33:21638–21652, 2020. 2

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proc. CVPR*, pp. 3070–3079, 2019. 8

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. CVPR*, pp. 2432–2443. IEEE, 2017. 6

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proc. CVPR*, pp. 13142–13153, 2023. 6

Google. Draco: 3D graphics compression. https://github.com/google/draco, 2025. 1, 2, 3, 7

Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pp. 355–361, 2002. 2, 3

Benoit Guillard, Federico Stella, and Pascal Fua. Meshudf: Fast and differentiable meshing of unsigned distance field networks. In *Proc. ECCV*, pp. 576–592, 2022. 3

Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 339–346, 2002. 1, 2

Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Transactions on Graphics (ToG)*, 32(3):1–13, 2013. 2, 7

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proc. ICLR*, 2015. 6

Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proc. ICCV*, pp. 12686–12696, 2021. 6

Shi-Lin Liu, Hao-Xiang Guo, Hao Pan, Peng-Shuai Wang, Xin Tong, and Yang Liu. Deep implicit moving least-squares functions for 3d reconstruction. In *Proc. CVPR*, pp. 1788–1797, 2021. 2

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pp. 347–353. 1998. 2

Andrea Maggiordomo, Federico Ponchio, Paolo Cignoni, and Marco Tarini. Real-world textured things: A repository of textured models generated with modern photo-reconstruction tools. *Computer Aided Geometric Design*, 83:101943, 2020. 6

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

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proc. CVPR*, pp. 4460–4470, 2019. 2

MPEG. Mpeg pcc reference software (tmc13). https://github.com/MPEGGroup/mpeg-pcc-tmc13, 2025. 1, 2, 7

MPEG. White Paper on Video-based Dynamic Mesh Coding. *ISO/IEC JTC1/SC29/AG03 N208*, 2025. 1

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proc. CVPR*, pp. 165–174, 2019. 2

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proc. CVPR*, pp. 652–660, 2017a. 2

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Proc. NeurIPS*, 30, 2017b. 2

Siyu Ren and Junhui Hou. Shape as line segments: Accurate and flexible implicit surface representation. In *Proc. ICLR*, volume 2025, pp. 82842–82862, 2025. 1, 2, 3

Siyu Ren, Junhui Hou, Xiaodong Chen, Ying He, and Wenping Wang. Geoudf: Surface reconstruction from 3d point clouds via geometry-guided distance representation. In *Proc. ICCV*, pp. 14168–14178, 2023. 2, 3

Siyu Ren, Junhui Hou, Weiyao Lin, and Wenping Wang. Neural compression for 3d geometry sets. In *Proc. ICCV*, pp. 25294–25304. IEEE, 2025. 1, 3

Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics*, 5(1):47–61, 1999. 3

S. Schwarz, M. Preda, V. Baroncini, et al. Emerging MPEG standards for point cloud compression. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9:133–148, 2019. 1, 3, 5

Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Proc. NeurIPS*, 34:6087–6101, 2021. 2

Danhang Tang, Saurabh Singh, Philip A Chou, Christian Hane, Mingsong Dou, Sean Fanello, Jonathan Taylor, Philip Davidson, Onur G Guleryuz, Yinda Zhang, et al. Deep implicit volume compression. In *Proc. CVPR*, pp. 1293–1303, 2020. 1, 3

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. pp. 1–9, 2008. 6

VVM Working Group. Call for proposals on static polygonal mesh coding. *Alliance for Open Media, document: VVM-2023-004o-v2*, 2023. 6

Jianqiang Wang, Dandan Ding, Zhu Li, Xiaoxing Feng, Chuntong Cao, and Zhan Ma. Sparse tensor-based multiscale representation for point cloud geometry compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(7):9055–9071, 2023. 1, 3, 7

Jianqiang Wang, Siyu Ren, and Junhui Hou. Neural compression of 3d meshes using sparse implicit representation. In *Proc. ICLR*, volume 2026, pp. 66366–66380, 2026. 1, 2, 3, 5, 6, 7

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. In *Proc. CVPR*, pp. 14419–14429, 2026. 6

Qijian Zhang, Junhui Hou, Yue Qian, Yiming Zeng, Juyong Zhang, and Ying He. Flattening-net: Deep regular 2d representation for 3d point cloud analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):9726–9742, 2023. 2

Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 6

Wenjie Zou, Shizhuo Zhang, Fuzheng Yang, and Marius Preda. Standardization status of mpeg video-based dynamic mesh coding (v-dmc). In *Proc. ICASSP*, pp. 1–5, 2025. 1, 3

A Detailed Inference Time

Table 5 reports SIEVE's runtime breakdown at a resolution of 256. The total runtime covers encoding and decoding. Encoding consists of mesh-to-DV-Edge conversion, neural encoding into latent features, and bitstream coding of the latent features and coordinates. Decoding mirrors these stages in reverse order.

Table 5: SIEVE's detailed inference time by stage.

Stage	Time (s)
Mesh-to-DV-Edge conversion	0.18
Neural encoding	0.04
Bitstream encoding	0.17
Total encoding	**0.39**
Bitstream decoding	0.07
Neural decoding	0.03
DV-Edge-to-mesh conversion	0.05
Total decoding	**0.15**

B Comparison Results on AOM and RWTT

Fig. 8 presents separate RD curves for the AOM and RWTT test sets, with SIEVE achieving the best compression performance on both.

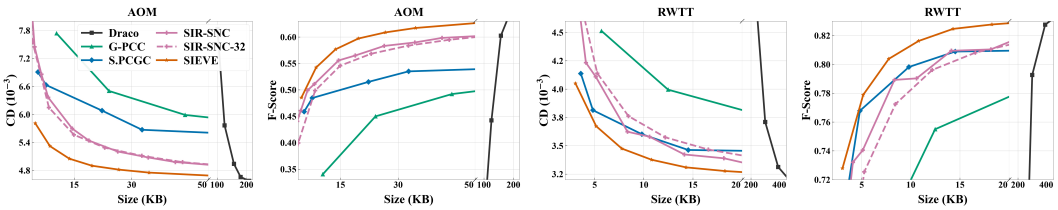

Figure 8: Rate-Distortion comparisons of different compression methods on AOM and RWTT.

C Robustness to Duplicate Faces

We compare the robustness of different methods when compressing meshes containing duplicate faces. Although these duplicate faces do not affect the visual appearance of the meshes, they may affect compression performance. Each method is evaluated on the same geometry without prior face deduplication or mesh repair to assess its robustness to such defective meshes, as shown in Fig. 9.

Draco directly encodes mesh vertices and connectivity, so duplicate faces introduce redundant connectivity and increase the bitstream size despite leaving the surface geometry unchanged. G-PCC and SparsePCGC exhibit only minor changes in bitrate and reconstruction quality, since their sampled point representations do not explicitly retain face multiplicity. However, their reconstruction fidelity remains lower than SIEVE's in both conditions. SIR-SNC shows a pronounced loss of reconstruction quality with duplicate faces because its ray-based SDF sign assignment is sensitive to coincident triangles. These can alter intersection-count parity, introducing incorrect inside–outside labels and disrupting subsequent surface extraction. SIEVE preserves the overall shape and local structures of the input meshes, with identical bitstream sizes and nearly unchanged reconstruction quality regardless of the presence of duplicate faces, demonstrating its advantages over other methods in robustness to duplicate faces and compression fidelity.

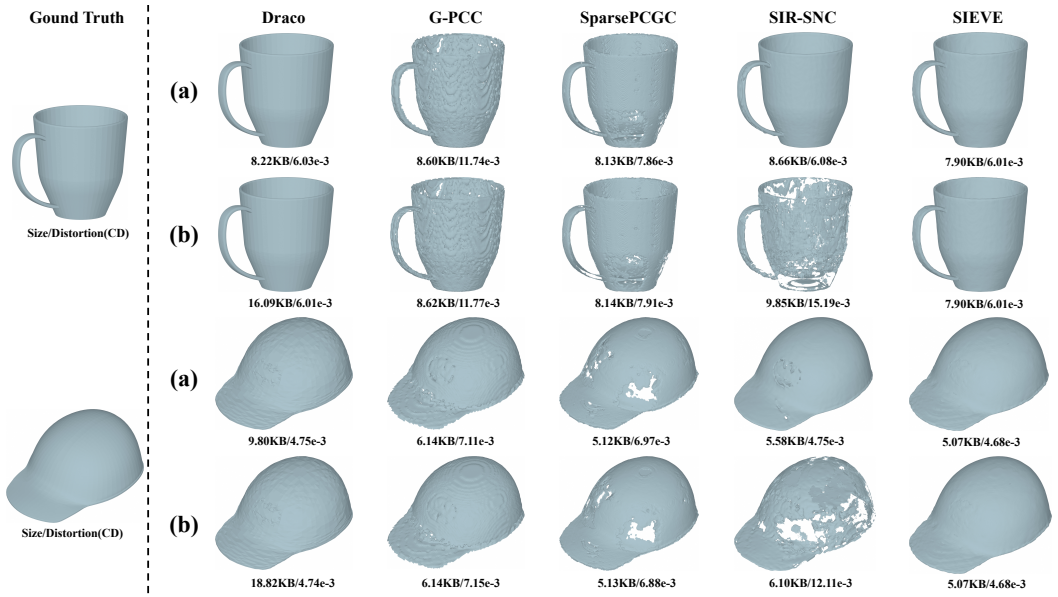

Figure 9: Robustness to duplicate faces. Rows (a) and (b) show compression reconstructions of meshes without and with duplicate faces, respectively.

D Decoder Threshold Visualization

We keep the bitstream fixed and vary only the decoder threshold. Fig. 10 presents a visual comparison of the reconstructed meshes.

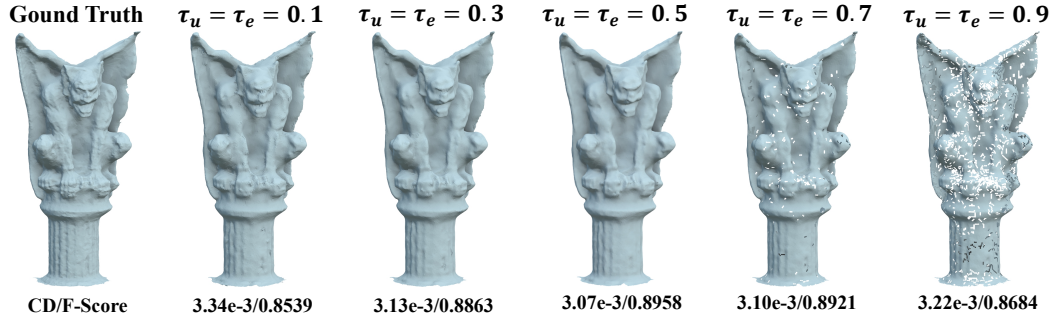

Figure 10: Visual comparison at different decoder thresholds.

E More Visual Results

We present additional mesh compression results in Fig. 11 and Fig. 12. Compared with existing methods, SIEVE achieves higher reconstruction fidelity for meshes with complex topology.

F Limitation

Due to the nature of query-edge-based representation, SIEVE cannot guarantee watertight decompressed meshes in the same way as SDF-based reconstruction with Marching Cubes. Small holes may arise during decompression for certain meshes, but they can be alleviated through hole-filling postprocessing , as shown in Fig. 13. Future work could explore integrating SDF representations into SIEVE to further improve the flexibility and generalization of compression and reconstruction.

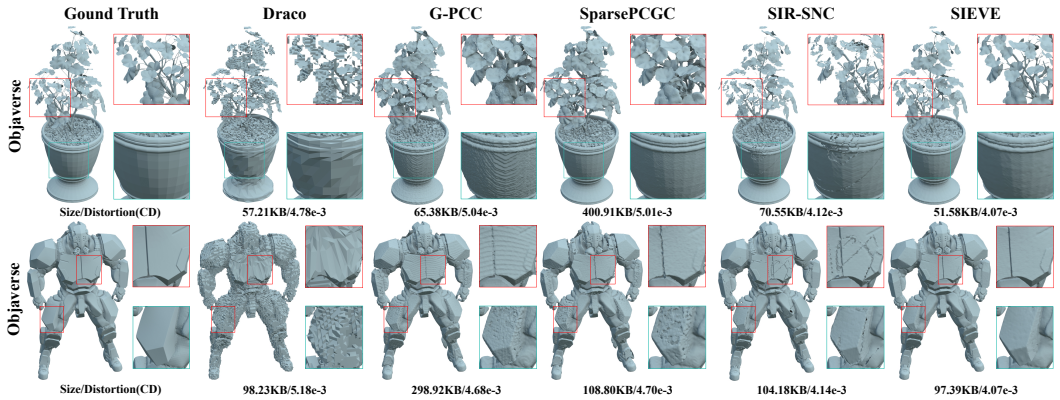

Figure 11: More visual comparisons of different compression methods.

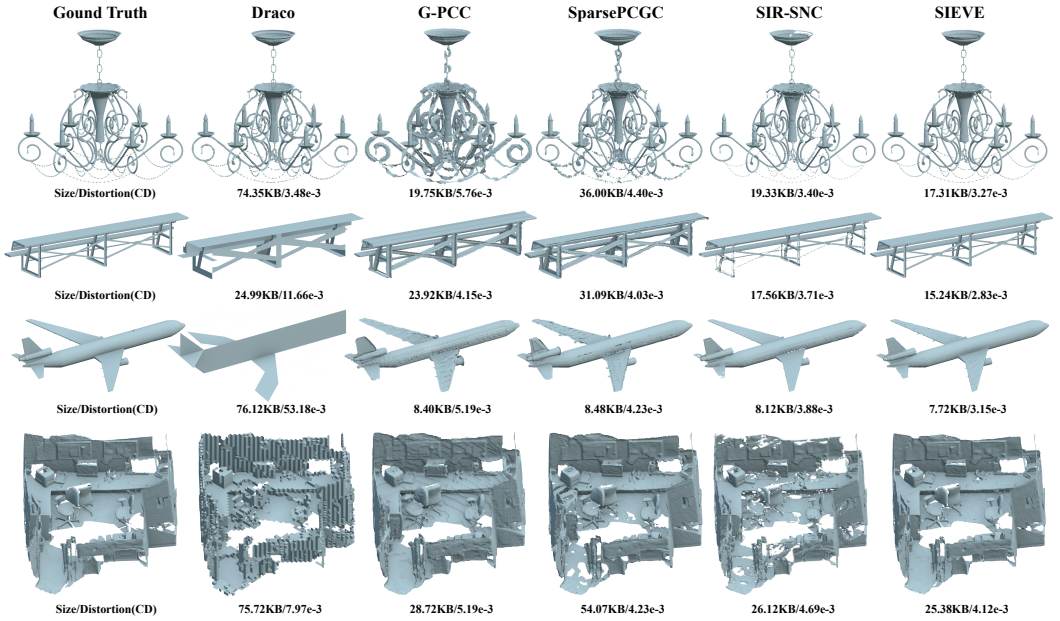

Figure 12: More visual comparisons of different compression methods.

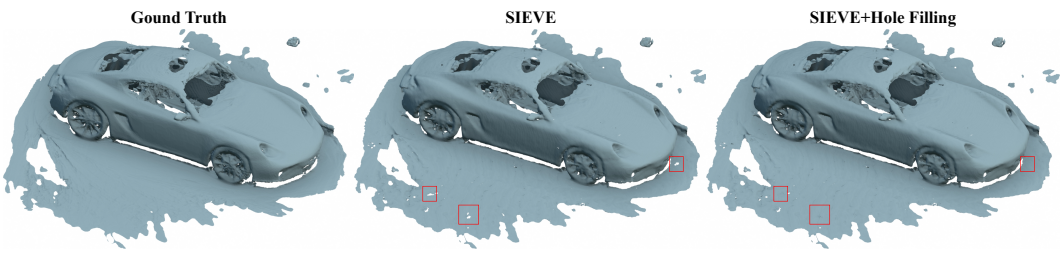

Figure 13: Mesh compression results with and without hole filling postprocessing.