

TopoMesh: High-Fidelity Mesh Autoencoding via Topological Unification

Supplementary Material

A. Topo-VAE

In this section, we present the network architecture of Topo-VAE (Sec. A.1), then analyze the explicit mesh-level supervision paradigm (Sec. A.2). Finally, we provide comprehensive experimental details (Sec. A.3).

A.1. Network

Sparse Voxel-Point Encoder \mathcal{E}_P . A lightweight architecture comprising 12.1M parameters. Per-point coordinates are projected into 51-dimensional Fourier features and concatenated with surface normals to form 54-channel features. These features are then aggregated into sparse voxels via our Sparse Voxel-Point Attention. As outlined in Algorithm 1, the mechanism takes as input a shared learnable token \mathbf{q} , linear projections of point features \mathbf{k} and \mathbf{v} , a mapping index \mathcal{I} that assigns each point to its enclosing voxel, and the number of heads H .

Algorithm 1 Sparse Voxel-Point Attention

Require: $\mathbf{q} \in \mathbb{R}^{1 \times C}$, $\mathbf{k}, \mathbf{v} \in \mathbb{R}^{P \times C}$, $\mathcal{I} \in \{0, V-1\}^P$, H

- 1: $d_h = C/H$;
- 2: Reshape \mathbf{q} into $[H, 1, d_h]$;
- 3: Reshape \mathbf{k}, \mathbf{v} into $[H, P, d_h]$;
- 4: // 1. Similarity Score $\mathbf{S} \in [H, P]$
- 5: $\mathbf{S} = \text{Sum}(\mathbf{q} \odot \mathbf{k}, \text{dim} = -1) / \sqrt{d_h}$;
- 6: // 2. Attention Score $\mathbf{A} \in [H, P]$
- 7: $\mathbf{A} = \text{ScatterSoftmax}(\mathbf{S}, \text{index} = \mathcal{I}, \text{dim} = -1)$;
- 8: // 3. Weighted Value $\mathbf{W} \in [H, P, d_h]$
- 9: $\mathbf{W} = \mathbf{A}.\text{unsqueeze}(-1) \odot \mathbf{v}$;
- 10: // 4. Aggregated Output $\mathbf{O} \in [H, V, d_h]$
- 11: $\mathbf{O} = \text{ScatterAdd}(\mathbf{W}, \text{index} = \mathcal{I}, \text{dim} = 1, \text{size} = V)$;
- 12: Reshape \mathbf{O} into $[V, C]$;
- 13: **return** \mathbf{O} ;

This attention process efficiently maps variable numbers of unstructured points into structured sparse voxels.

Sparse Latent Encoder \mathcal{E}_S . A structured latent encoder with 81.8M parameters and the same architecture as Trelis [20]. The 1024-channel voxel features from the Sparse Voxel-Point Encoder are projected into a 768-dimensional hidden space and processed by 12 layers of 3D shifted window attention [13]. The resulting features are compressed into a 16-channel latent, which is split into an 8-channel mean μ and an 8-channel standard deviation σ for variational sampling. The overall encoder is represented by:

$$\mathcal{E} = \mathcal{E}_S \circ \mathcal{E}_P \quad (1)$$

Sparse Latent Decoder \mathcal{D}_S . A structured latent decoder with 97.2M parameters. The sampled 8-channel latent are projected into a 768-channel hidden space and processed by 12 layers of 3D shifted window attention blocks. To recover high-resolution geometry, the features are passed through three upsampling blocks, increasing the voxel resolution by a factor of $8 \times$ (e.g., $64^3 \rightarrow 512^3$, $128^3 \rightarrow 1024^3$). To maintain computational efficiency, we apply a pruning module after each upsampling block from SparseFlex [6], which removes redundant voxels based on a predicted occupancy.

Decoupled FlexiCubes Decoder \mathcal{D}_F . This parameter-free module is designed to deterministically extract mesh vertices and connectivity from the predicted sparse voxel features. Given the 61-channel feature \mathbf{f} for each active voxel, we decompose it into topological components (o, γ) and geometric components $(u, \delta, \alpha, \beta)$ via the following splitting and activation scheme:

$$\begin{aligned} \mathbf{o} &= \mathbb{I}(\mathbf{f}_{[0:8]} > 0), & \mathbf{u} &= \text{Softplus}(\mathbf{f}_{[8:16]}) \\ \delta &= \tanh(\mathbf{f}_{[16:40]}), & \beta &= \tanh(\mathbf{f}_{[40:52]}) \\ \alpha &= \tanh(\mathbf{f}_{[52:60]}), & \gamma &= \text{Sigmoid}(\mathbf{f}_{[60:61]}) \end{aligned}$$

where indices denote channel slicing. Here, \mathbf{o} determines the DMC [16] topology. The sdf value is reconstructed via $\mathbf{s} = \mathbf{u} \cdot (1 - 2\mathbf{o})$. Finally, $\{\mathbf{s}, \delta, \alpha, \beta\}$ parameterize the FlexiCubes [17] to compute vertex positions, while γ determines the triangulation strategy for the generated quadrilaterals. The overall decoder is represented by:

$$\mathcal{D} = \mathcal{D}_F \circ \mathcal{D}_S \quad (2)$$

A.2. Explicit Supervision

Difference from Prior Work. We underscore that our Explicit Supervision paradigm fundamentally differs from prior work by enabling an **end-to-end, fully differentiable optimization** of mesh vertex positions and connectivity.

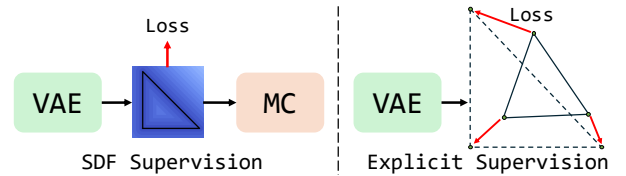


Figure S1. Comparison of different supervision signals.

As illustrated in Fig. S1, while SDF-based supervision [11, 12, 19, 21, 22] and Rendering-based supervision [6, 20] rely on indirect supervision, thus suffering from supervisory ambiguity and high-frequency detail loss, our method explicitly regresses (**drags**) predicted vertices to their exact ground-truth positions, providing strong, unambiguous gradients necessary for preserving sharp edges and corners.

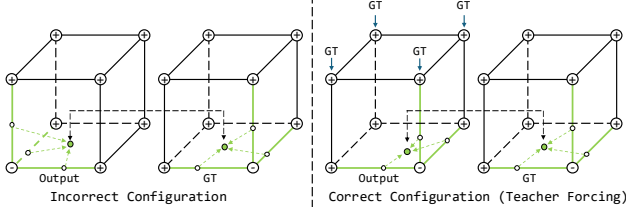


Figure S2. Stable training with Teacher Forcing.

Correspondence Construction. Explicit supervision necessitates a rigorous, element-wise correspondence between the predicted mesh and the ground truth. We construct this correspondence hierarchically:

- 1 **Voxel-Level.** A predicted voxel corresponds uniquely to the gt voxel sharing the same spatial coordinate $\mathbf{c} \in \mathbb{Z}^3$.
- 2 **Vertex-Level.** Vertices within corresponding voxels are matched. Our Teacher Forcing guarantees consistent topological structures (Fig. S2), ensuring identical vertex counts and valid gradient flow during training.
- 3 **Face-Level.** Quadrilaterals generated from identical grid edges are matched. Our Teacher Forcing guarantees the exact number of predicted faces during training.

Total Loss. Our final loss is a weighted combination of explicit mesh-level losses and several regularization terms to ensure stable training, formulated as follows:

$$L = \lambda_{\text{topo}} \mathcal{L}_{\text{topo}} + \lambda_{\text{vert}} \mathcal{L}_{\text{vert}} + \lambda_{\text{normal}} \mathcal{L}_{\text{normal}} + \lambda_{\text{occ}} \mathcal{L}_{\text{occ}} + \lambda_{\text{con}} \mathcal{L}_{\text{con}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}$$

Here, the weights λ_{topo} , λ_{vert} , λ_{normal} , λ_{occ} , λ_{con} , λ_{reg} , and λ_{KL} are set to constants 1.0, 0.4, 0.2, 0.1, 1.0, 0.01, and 10^{-6} , respectively.

A.3. Experiment

Topo-Bench. To rigorously evaluate reconstruction fidelity on complex geometries, we curate Topo-Bench, a challenging benchmark consisting of 1,000 high-quality meshes from Objaverse [5], Thingi10K [23], and Toys4K [18]. We visualize several samples in Fig. S3.

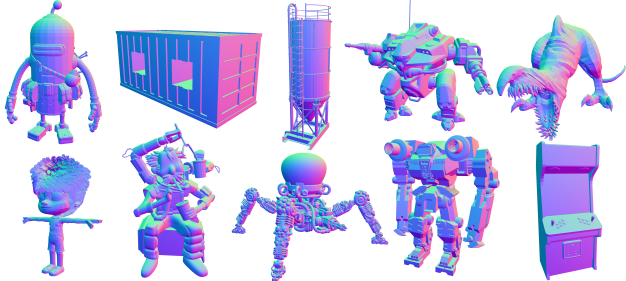


Figure S3. Samples from Topo-Bench

Baselines. The core objective of our work is to elevate the reconstruction fidelity of 3D VAEs. To ensure reproducibility and fair comparison, we restrict our evaluation to **open-sourced** methods, excluding Sparc3D [12], Hitem3D [15], Hunyuan3D-2.5 [10], Rodin-Gen-v2 [4], Meshy [1], etc.

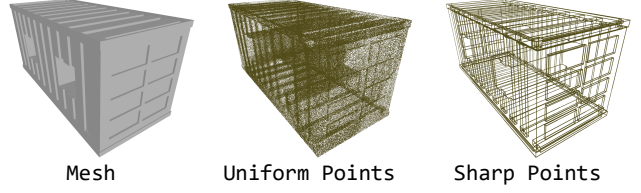


Figure S4. Visualization of uniform sampling and sharp edge sampling.

Evaluation Metrics. To ensure consistent evaluation, we normalize all meshes to $[-1, 1]^3$ and then compute 2D and 3D metrics. We evaluate visual fidelity using Absolute Normal Consistency (ANC) [12], computed over 150 views rendered at 512×512 resolution (camera radius 1.7, FoV 30°) following the camera sampling strategy in Trellis [20]. For geometric accuracy, we report Chamfer Distance (CD) and F-Score (F1) based on 500,000 uniformly sampled points. Furthermore, we introduce F1-Sharp to rigorously assess the preservation of sharp features. As shown in Fig. S4, this metric calculates the F-Score ($\tau = 0.005$) on 500,000 points sampled exclusively from high-curvature surface regions with dihedral angles sharper than 30° .

Image-to-3D Generation. To validate Topo-VAE as a powerful backbone for image-to-3D generation, we integrate it into a two-stage pipeline following Ultra3D [3]. For quantitative evaluation, we sample 200 objects from the Toys4K [18] dataset. For each object, we render 4 views, resulting in a total of 800 images. We report Fréchet Inception Distance (FID) [7] and Kernel Inception Distance (KID) [2] to assess the perceptual quality and diversity of the generated assets.

B. Topo-Remesh

In this section, we provide comprehensive details for Topo-Remesh. We first elaborate the definition and geometric properties of the L_∞ metric (Sec. B.1), then present the detailed algorithmic pipeline (Sec. B.2) and demonstrate its ability to extend to higher resolution (Sec. B.3).

B.1. L_∞ Metric

Definition. As defined in the main paper (Eq. ??), the L_∞ distance $D_\infty(P, Q)$ between a spatial query point P and its nearest surface point Q is the maximum Euclidean distance from P to the planes of all triangles $T(Q)$ incident to Q . Formally, let Π_i be the plane containing triangle T_i . The metric is given by:

$$D_\infty(P, Q) = \max_{T_i \in \mathcal{T}(Q)} d(P, \Pi_i) \quad (3)$$

A critical observation is that the surface point $Q_{L_2}^*$ minimizing the standard Euclidean distance is sometimes distinct from the point $Q_{L_\infty}^*$ that minimizes the L_∞ distance. Finding the globally optimal $Q_{L_\infty}^*$ is non-trivial due to the

non-smooth and anisotropic nature of the L_∞ landscape, which prevents standard acceleration structures (e.g., BVH, KD-Tree) from performing efficient pruning, rendering direct optimization computationally prohibitive.

To address this, we employ an approximation strategy based on local sampling. Instead of relying solely on the L_2 -nearest point, we generate a set of candidates $Q = \{Q_k\}$ within the regional neighborhood of P that are visible and locally convex relative to P . We then evaluate the L_∞ distance for all candidates and define the final SDF value as the minimum:

$$\text{SDF}_\infty(P) = \min_{Q_k \in Q} D_\infty(P, Q_k) \quad (4)$$

This approach ensures that we capture the correct ‘box-like’ distance field required to preserve sharp corners.

Implementation. Theoretically, calculating $D_\infty(P, Q_k)$ requires iterating over incident triangles defined by mesh topology. However, explicit connectivity is unreliable for raw meshes, which suffer from topological defects such as self-intersections and duplicate vertices.

To ensure robustness, we define the incident triangle sets $T(Q_k)$ via spatial proximity rather than explicit connectivity. Specifically, a triangle T_j is considered incident to Q_k if $d(Q_k, T_j) < \epsilon$. The L_∞ distance is computed by maximizing over this geometrically associated triangle set:

$$D_\infty(P, Q_k) \approx \max(d(P, \Pi_j) \mid d(Q_k, T_j) < \epsilon) \quad (5)$$

where ϵ is a small tolerance threshold 10^{-7} . This strategy effectively “stitches” disconnected components on the fly, ensuring resilience against diverse mesh artifacts. We summarize this calculation procedure in Algorithm 2.

Algorithm 2 L_∞ Distance between P and Q_k

Require: $P, Q_k, \epsilon, \{T_j\}$

- 1: $D = 0$;
- 2: **for** each triangle $T_j \in \{T_j\}$ **do**
- 3: // 1. Point to Triangle Distance
- 4: **if** $d(Q_k, T_j) < \epsilon$ **then**
- 5: $\Pi_j = \text{Plane}(T_j)$;
- 6: // 2. Point to Plane Distance
- 7: $d_{\text{plane}} = d(P, \Pi_j)$;
- 8: // 3. L_∞ is the maximum over incident triangles.
- 9: $D = \max(D, d_{\text{plane}})$;
- 10: **end if**
- 11: **end for**
- 12: **return** D ;

Finally, we leverage a Bounding Volume Hierarchy (BVH) to accelerate the aforementioned operations, including nearest surface point search, local triangle retrieval for incident checks, and visibility verification. The complete

workflow for computing SDF_∞ at a query point P is summarized in Algorithm 3:

Algorithm 3 L_∞ SDF Calculation

Require: $P, \text{BVH}, \epsilon, \eta$

- 1: // 1. L_2 -nearest point as one candidate.
- 2: $Q = \text{BVH.ClosestPoint}(\text{src} = P)$;
- 3: // 2. Compute L_∞ distance between P and Q .
- 4: $S = \text{BVH.RobustLinf}(P, Q, \epsilon)$;
- 5: // 3. Heuristic search for another candidate.
- 6: $P' = P + \eta \cdot (P - Q)$;
- 7: $Q' = \text{BVH.ClosestPoint}(\text{src} = P')$;
- 8: // 4. Check visible.
- 9: **if** $Q' \neq Q$ and $\text{BVH.IsVisible}(P, Q')$ **then**
- 10: // 5. Compute L_∞ distance between P and Q' .
- 11: $S' = \text{BVH.RobustLinf}(P, Q', \epsilon)$;
- 12: // 6. SDF is the minimum L_∞ among candidates.
- 13: $S = \min(S, S')$;
- 14: **end if**
- 15: **return** S ;

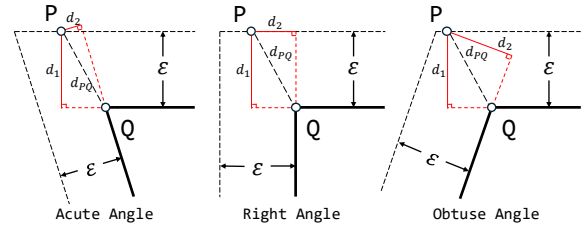


Figure S5. Robust to different local structures.

Robustness. The proposed L_∞ SDF formulation is inherently robust to common mesh artifacts. Specifically, by relying on point-to-plane distances within local neighborhoods, it remains resilient to topological defects, such as flipped normals, self-intersections, or non-manifold edges. Also, the maximum operation naturally filters out spurious inner structures, as shown in ??b (main). Finally, our formulation is agnostic to diverse local geometries, effectively handling acute, right, and obtuse angles, as demonstrated in Fig. S5.

B.2. Pipeline

Our Topo-Remesh framework operates in five sequential stages, fully accelerated on the GPU.

Voxelization and Flood-fill. We first discretize the input mesh into a sparse voxel grid. Subsequently, a flood-fill operation propagates from the exterior to identify surface-adjacent voxels. To ensure full coverage of the underlying geometry, we apply a 5-voxel dilation to this boundary, establishing a narrow band for the final surface extraction.

SDF Calculation. We evaluate the signed distance at grid corners within the narrow band to precisely identify surface-intersecting voxels. To enable surface dilation described in

the main paper (??), the final SDF is obtained by offsetting the L_∞ distance:

$$\text{SDF}(P) = \text{SDF}_\infty(P) - \varepsilon \quad (6)$$

where $\text{SDF}_\infty(P)$ is defined and computed by Eq. 4 and Algorithm 3, and ε denotes the dilation radius, which we set to $1/1024$.

Mesh Extraction. Given the non-smooth nature of the L_∞ distance field, we employ Occupancy-based Dual Contouring (ODC) [8] for surface extraction. Unlike methods dependent on continuous SDF magnitudes [14, 16] or gradients [9], ODC robustly reconstructs geometry relying solely on binary occupancy information. It resolves optimal vertex placements by iteratively querying occupancy across grid edges and faces. Please refer to the original paper for more details. Formally, the occupancy at a spatial point P is defined as:

$$\text{Occ}(P) = \mathbb{I}(\text{SDF}_\infty(P) \leq \varepsilon) \quad (7)$$

where \mathbb{I} is the indicator function and ε represents the dilation radius.

Compression. We decompose mesh attributes into compact primitives to achieve efficient storage. Vertex positions (3×32 bits) are decoupled into integer voxel coordinates (3×10 bits) and intra-voxel offsets (3×10 bits), allowing both components to be quantized and tightly bit-packed into two 32-bit integers. Vertex connectivity is decoupled into occupancy (8 bits) at voxel corners and a triangulation decision (1 bit) for each generated face (Fig. S6). By anchoring these decisions to the voxel with the minimal coordinate, we pack the flags for three axes into 3 bits per voxel. **Crucially, these primitives serve directly as the training targets for Topo-VAE, without the need for intermediate mesh reconstruction.**

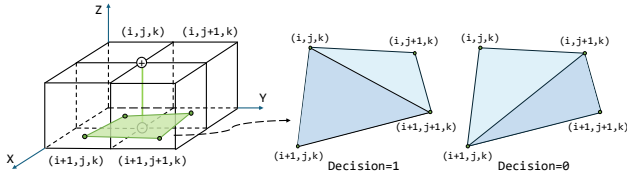


Figure S6. Triangulation decision.

During decoding, vertex positions are recovered via dequantization and aggregation of voxel coordinates with intra-voxel offsets, while connectivity is efficiently assembled following the indexing protocol of FlexiCubes [17]. Theoretically, the mesh extracted from N active voxels typically contains $V \approx N$ vertices and $F \approx 2N$ faces. Standard storage (using 32 bits for both) will consume $12V + 12F \approx 36N$ bytes. In contrast, our representation requires only packed voxel coordinates ($4N$ bytes), packed intra-voxel offsets ($4N$ bytes), packed occupancy

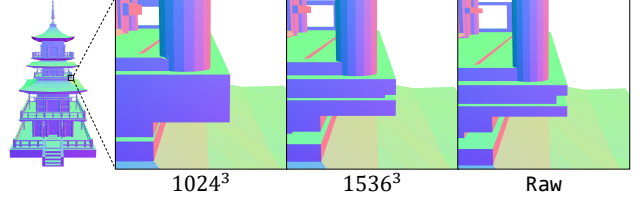


Figure S7. Remeshing under different resolution.

($1N$ bytes), and packed triangulation flags ($3N$ bits), totaling approximately $9.375N$ bytes, yielding a compression rate of 74%.

B.3. Scaling

Our Topo-Remesh algorithm is inherently adaptive to varying resolutions. Scaling the voxel grid to higher resolutions (e.g. 1536^3) enhances the preservation of fine geometric details (Fig. S7), albeit at an increased computational cost. (35s at 1536^3 compared to 14s at 1024^3).

C. Future Work

TopoMesh achieves high-fidelity mesh reconstruction at 1024^3 . While further scaling to higher resolution enables the capture of finer details, it incurs cubic computational costs and remains bounded by voxel discretization. We think a promising path toward "infinity resolution" is continuous local representation, where intra-voxel geometry is parameterized as continuous fields or point distributions. This approach enables mesh extraction at arbitrary precision while maintaining computational efficiency.

D. Visualization

We provide more visualizations of remeshing in Fig. S8, VAE reconstruction in Fig. S9, and Fig. S10. and image-to-3D generation in Fig. S11.

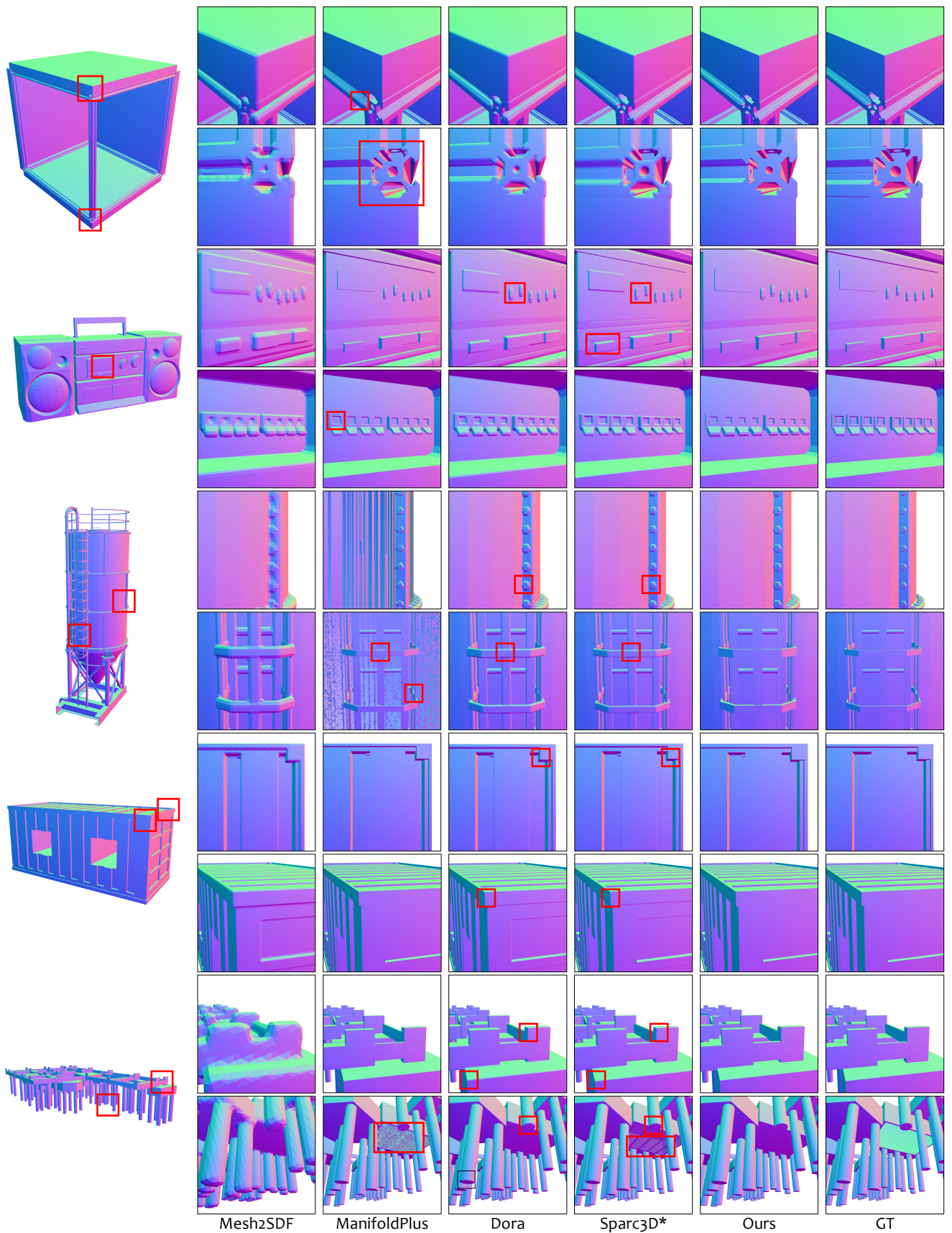


Figure S8. Visual comparisons of remeshing. Our method produces clean meshes with sharp features. * means a re-implementation version.

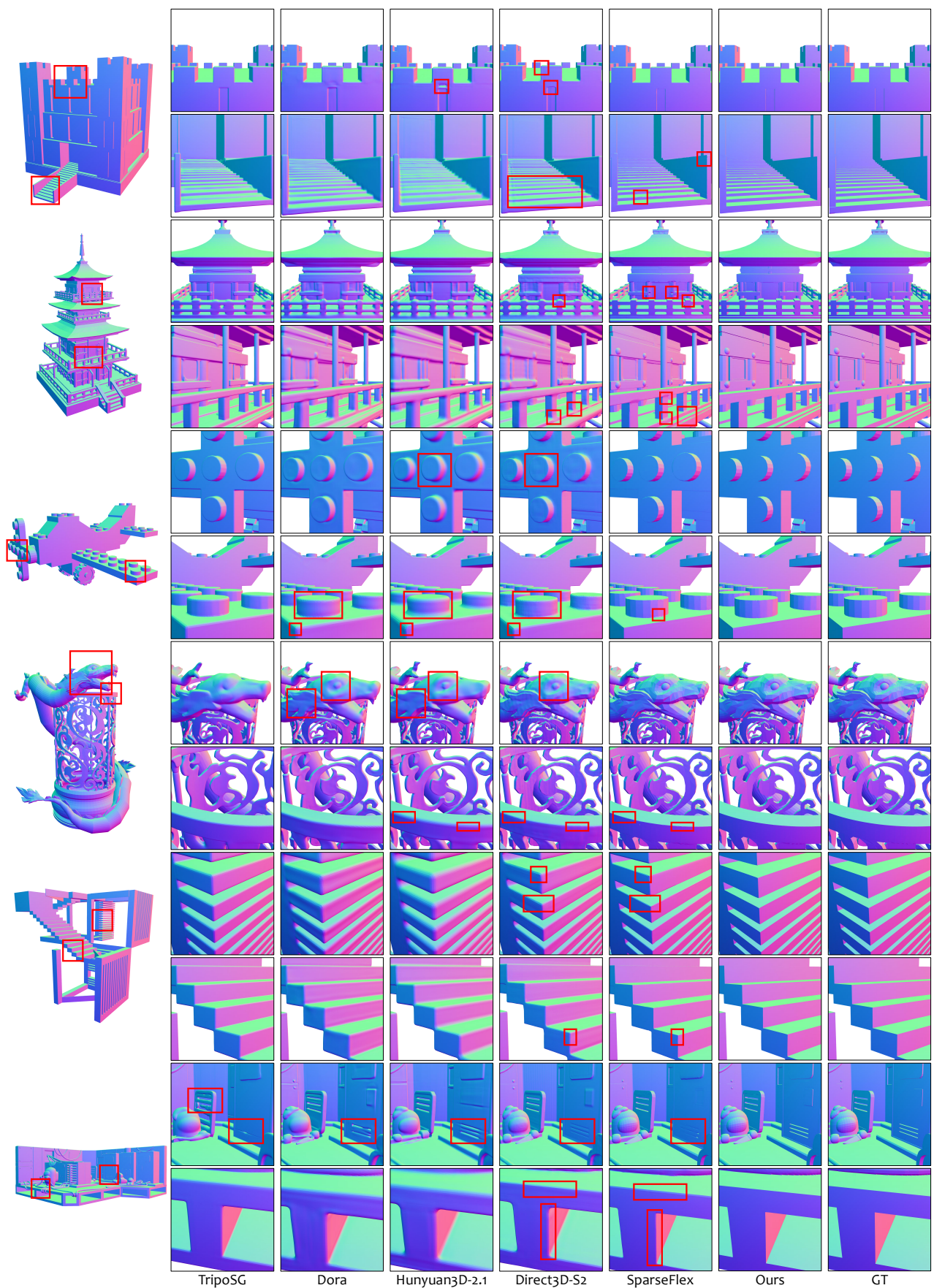


Figure S9. Visual comparisons of VAE reconstruction. Our method better preserves sharp features, as highlighted by red boxes.

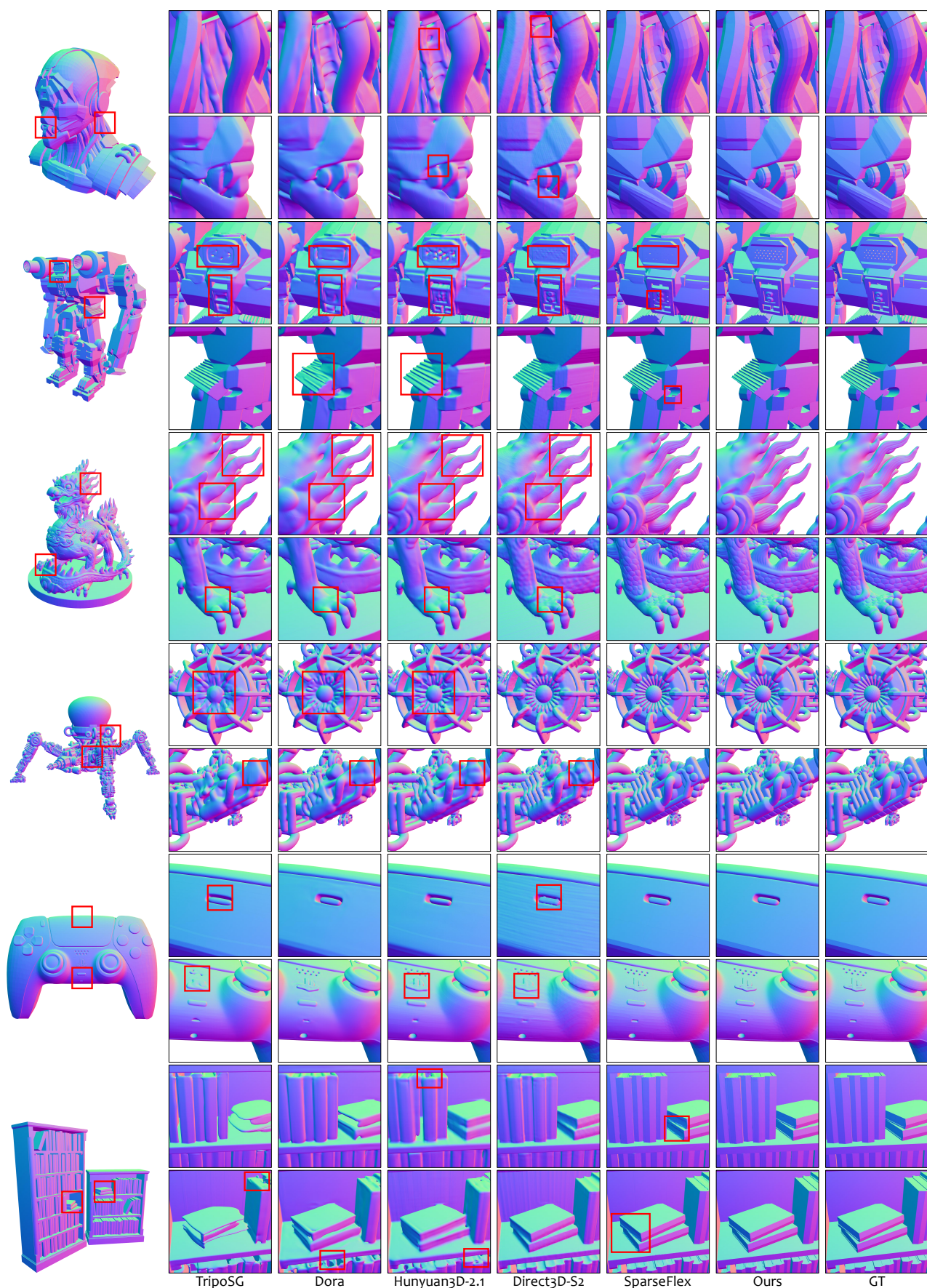


Figure S10. Visual comparisons of VAE reconstruction. Our method better preserves fine geometric details, as highlighted by red boxes.



Figure S11. Visual comparisons of image-to-3D Generation. Our method produces fine geometric details with better image alignment, as highlighted by red boxes.

References

- [1] Meshy AI. Meshy ai – meet the world’s most popular and intuitive free ai 3d model generator. <https://www.meshy.ai/>. 2
- [2] Mikolaj Binkowski, Danica J. Sutherland, Michael Arbel, and Arthur Gretton. Demystifying MMD gans. In *International Conference on Learning Representations (ICLR)*, 2018. 2
- [3] Yiwen Chen, Zhihao Li, Yikai Wang, Hu Zhang, Qin Li, Chi Zhang, and Guosheng Lin. Ultra3d: Efficient and high-fidelity 3d generation with part attention. *arXiv preprint arXiv:2507.17745*, 2025. 2
- [4] Deemos. Rodin-gen2. <https://hyper3d.ai/>. 2
- [5] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2
- [6] Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. Sparseflex: High-resolution and arbitrary-topology 3d shape modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 1
- [7] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 2
- [8] Jisung Hwang and Minhyuk Sung. Occupancy-based dual contouring. In *SIGGRAPH Asia*, 2024. 4
- [9] Tao Ju, Frank Losasso, Scott Schaefer, and Joe D. Warren. Dual contouring of hermite data. *ACM Trans. Graph.*, 2002. 4
- [10] Zeqiang Lai, Yunfei Zhao, Haolin Liu, Zibo Zhao, Qingxiang Lin, Huiwen Shi, Xianghui Yang, Mingxin Yang, Shuhui Yang, Yifei Feng, Sheng Zhang, Xin Huang, Di Luo, Fan Yang, Fang Yang, Lifu Wang, Sicong Liu, Yixuan Tang, Yulin Cai, Zebin He, Tian Liu, Yuhong Liu, Jie Jiang, Linus, Jingwei Huang, and Chunchao Guo. Hunyuan3d 2.5: Towards high-fidelity 3d assets generation with ultimate details. *arXiv preprint arXiv:2506.16504*, 2025. 2
- [11] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. 1
- [12] Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3d: Sparse representation and construction for high-resolution 3d shapes modeling. *arXiv preprint arXiv:2505.14521*, 2025. 1, 2
- [13] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1
- [14] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1987, Anaheim, California, USA, July 27-31, 1987*, 1987. 4
- [15] MathMagic. Hitem3d - ai-powered 3d model generator. <https://www.hitem3d.ai/>. 2
- [16] Gregory M. Nielson. Dual marching cubes. In *IEEE Visualization*, 2004. 1, 4
- [17] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM Trans. Graph.*, 2023. 1, 4
- [18] Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2
- [19] Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, and Yao Yao. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. 1
- [20] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 1, 2
- [21] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 2023. 1
- [22] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Trans. Graph.*, 2024. 1
- [23] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 2