TOPOFLOWFORGE: PROGRESSIVE NATIVE MESH GENERATION VIA CASCADED FLOW MATCHING

Anonymous authors
Paper under double-blind review



Figure 1: Native meshes generated by TopoFlowForge across categories (furniture, characters, props), rendered with their predicted topology. The framed images show the UV unwrappings of several meshes in the figure, also revealing the quality of their topology. *Best viewed with zoom-in.*

ABSTRACT

3D content generation technology has significantly advanced the work of designers, as well as the 3D printing and gaming industries. However, it remains difficult to produce lightweight, editable, and topologically clean artistic content that is directly production-ready. To achieve this, we present **TopoFlowForge**, an artistic mesh foundation model that generates production-ready meshes. Specifically, TopoFlowForge decomposes the mesh generation process into vertices generation and their connectivity prediction, i.e., edges. We formulate vertices generation as a two-stage coarse-to-fine process and incorporate several effective loss functions to further enhance its performance. In the connectivity prediction stage, we propose a simple yet effective method for estimating the connectivity affinity between vertices and additionally predict per-vertex normals, which determines the correct orientation of faces. Besides, we construct a large-scale dataset combining hand-crafted 3D assets with public high-quality topology datasets. Based on this, a carefully designed data curation pipeline is employed to filter the raw dataset, retaining only high-quality topology data for model training. Our model is trained on the combined dataset and tested on both out-of-distribution hand-crafted set of 3D assets and public datasets. Under image conditioning, TopoFlowForge surpasses autoregressive methods in both generation quality and inference speed, achieving state-of-the-art results among open-source mesh topology generators. We will release all code and weights together with a portion of our test dataset.

1 INTRODUCTION

Triangle meshes are the fundamental basis of modern 3D production pipelines, including rigging, animation, and rendering engines. Image-conditioned 3D mesh generation models can greatly simplify the workflows for artists and designers. However, not all generated triangle meshes are ready

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.

for industrial deployment. Models based on SDF representations Zhang et al. (2023); Xiang et al. (2025b) produce meshes with excessively high face counts and topologically irregular structures, which poses three major challenges: (1) UV unwrapping on complex topologies bottlenecks the entire pipeline; (2) heavy storage overhead hinders real-time on-device rendering; and (3) irregular mesh topology complicates downstream rigging and animation. These challenges are particularly pronounced in consumer-device settings, including mobile platforms and emerging VR/AR hardware, where 3D assets must support interactive rendering and immersive user experiences.

Recent work has explored the direct generation of mesh topology, bypassing the intermediate SDF representation, in order to produce meshes with clean topology. For example, methods like DeepMesh Zhao et al. (2025a) and BPT Weng et al. (2025) directly generate the discrete vertex and face sets $(\mathcal{V}, \mathcal{F})$ in a sequential manner. These autoregressive methods typically rely on a tokenizer to map meshes into a one-dimensional token space, where generation is formulated as a next-token prediction process. The generated token sequence is then converted back into a mesh using a detokenizer. Such methods are often computationally inefficient at inference time. More importantly, encoding meshes as one-dimensional token sequences discards the intrinsic structural information present in native 3D representations, which frequently results in unstable generation performance. In contrast to these methods, Nexus Wang et al. (2026b), LATO Zhao et al. (2026) and LATO.2 Long et al. (2026) directly generate 3D primitives with diffusion models Peebles & Xie (2023), achieving higher-quality topology generation. However, these methods still suffer from unstable vertex generation quality or the lack of explicit orientation prediction for generated faces. Meanwhile, 3D datasets are inherently limited and scattered across diverse sources, while only a relatively small portion satisfies the quality requirements for clean topology. As a result, obtaining sufficient high-quality topological data for model training remains another major challenge.

To bridge this gap, we propose TopoFlowForge, a foundation model for native production-ready 3D content generation. Concretely, we decompose the pipeline into three stages. In the first stage, a latent-space DiT Peebles & Xie (2023) generates coarse vertices of an object in a 64^3 canonical space. Then, a mixture-of-experts(MOE) model in the original space refines and upsamples these primitives to high-fidelity vertex sets. This two-stage coarse-to-fine design(we called late-stage refinement) for vertex generation enables the vertices to maintain a globally correct structure while producing finer geometric details. Finally, in stage 3, a DiT is used to generate per-vertex features, from which a decoder predicts both the connectivity between vertices and the normal of each vertex, thereby reconstructing the final mesh with correct face normal directions.

Besides the generation architecture, we also construct a large-scale training dataset from a collection of in-house 3D assets and public datasets. A curation pipeline retains meshes with high-quality topology. Training on this diverse dataset enables TopoFlowForge to generalise across a broad range of object categories. Our evaluations show that TopoFlowForge achieves state-of-the-art performance among open-source methods and remains competitive with proprietary commercial methods.

In summary, our contributions are as follows:

- A novel framework TopoFlowForge that generates product-ready 3D meshes through a late-stage progressive refinement procedure, progressively upsampling only from 64^3 to 512^3 while maintaining topological consistency across resolutions.
- Topology-aware losses designed for vertex generation, including boundary-aware weighted binary cross-entropy to address class imbalance, degree-aware regularization to reinforce structurally important vertices and coplanarity constraints to preserve face structure.
- A joint edge-normal prediction algorithm that simultaneously predicts vertex connectivity and surface normals, enabling accurate face recovery with correct orientation and improving topology prediction accuracy.

2 Related Work

Field-based mesh generation 3D content generation has evolved from object-specific optimization with pretrained 2D diffusion priors Poole et al. (2022); Wang et al. (2023) to feed-forward reconstruction models Hong et al. (2023); Tang et al. (2024a); Yang et al. (2024). Field-based methods decode implicit shapes from compact latents, with advances in vector sets Zhang et al. (2023),

sharp-edge-aware autoencoding Chen et al. (2025a), triplane diffusion Wu et al. (2024), and hybrid representations Guo et al. (2025); Zhang et al. (2024); Zhao et al. (2025b); Li et al. (2025a). TRELLIS Xiang et al. (2025b) introduces sparse spatial structures that concentrate computation near object surfaces, followed by related sparse and structured approaches He et al. (2025b); Li et al. (2025b); Wu et al. (2025); Xiang et al. (2025a). However, the extracted meshes are often dense and topologically unstructured, motivating native-mesh methods that directly generate vertices and faces using autoregressive or diffusion models.

Autoregressive native mesh generation Autoregressive methods directly generate the discrete vertex and face sets in a sequential manner. Nash et al. (2020) generates vertices followed by faces expressed as vertex-index sequences. Siddiqui et al. (2024) generates indices into a learned vocabulary embeddings. Chen et al. (2024) uses pointcloud features to condition artist-mesh generation on a target shape. Zhao et al. (2025a) further explores preference alignment for autoregressive mesh generation through Direct Preference Optimization (DPO). To reduce sequence redundancy, Chen et al. (2025b) reuses shared edges between consecutive adjacent triangles. Tang et al. (2024b), Weng et al. (2025) further reduces the coordinate-token costs by improving the tokenizer. Complementary to token compression, Hao et al. (2024) combines an hourglass architecture, truncate-sequence training, and sliding-window inference to support up to 64K faces. Wang et al. (2026a) assigns one transformer token per face, retaining autoregressive coordinate prediction within each face through a CausalMLP head. Kim et al. (2025) restricts autoregressive generation to vertices, then predicts vertex-pair connectivity in parallel and recovers faces from mutually connected triplets. These autoregressive methods share two fundamental limitations. Firstly, their sequential decoding makes inference latency scale with sequence length. Secondly, serializing mesh topology into one-dimensional token sequences discards the intrinsic spatial information in 3D assets, which frequently leads to unstable generation as geometric complexity increases.

Diffusion based native mesh generation Diffusion and flow-matching methods update mesh representations within each sampling step. Alliegro et al. (2023) applies discrete diffusion to quantized triangle soups, recovering coordinates from categorical noise. He et al. (2025a) encodes meshes into continuous face-level VAE tokens and generates them with a diffusion transformer conditioned on the target face count. Both represent meshes through triangle faces rather than a separately generated vertex set and its connectivity. Li et al. (2026) jointly encodes vertex positions, normals, and adjacency into continuous MeshVAE latents modeled by a diffusion transformer. Zhao et al. (2026) uses a two-stage pipeline to generate coarse structure voxels and predicts edges between vertex pairs using a connection head. Wang et al. (2026b) generates vertices through a diffusion model conditioned on preceding octree levels, followed by per-vertex topology latents prediction that recovers edges and faces. These works mark a significant advance over autoregressive methods by generating mesh directly in 3D space, thereby preserving the native spatial structure of meshes and achieving better generation performance. However, it still remains challenges in generating a stable mesh structure and consistent vertex connectivity. LATO.2 lacks explicit face orientation prediction, resulting in inconsistent face normals that require post-correction. Nexus relies on hierarchical octree diffusion for vertex generation, where errors accumulate across octree levels, leading to an unstable mesh structure. We try to address these limitations through our method.

3 METHOD

As a preliminary, a triangle mesh is conventionally represented as a set of vertices $\mathcal{V}$ and faces $\mathcal{F}$. In our implementation, each face is defined by three edges in the set of edges $\mathcal{E}$ together with a normal orientation. For topology generation, the vertices $\mathcal{V}$ are quantized to grid points at a given resolution, which reduces the search space and improves generation quality.

Based on the $(\mathcal{V}, \mathcal{E}, \mathcal{F})$ data structure, TopoFlowForge generates product-ready 3D content through three cascaded image-conditioned stages (Figure. 2). Specifically, stage 1 adapts a diffusion generation model to produce a coarse 64^3 occupancy voxel from 16^3 noise as the coarse-quantized vertex positions. Then, stage 2 operates directly in the generated 64^3 occupancy voxel, applying flow matching (FM) in normalized target spaces, where three level-specific experts are used to progressively refine the field to 512^3 resolution, yielding the final vertex set $\hat{\mathcal{V}}$. We refer to the combination of stages 1 and 2 as **late-stage refinement**. The model first predicts a coarse mesh shape, after which

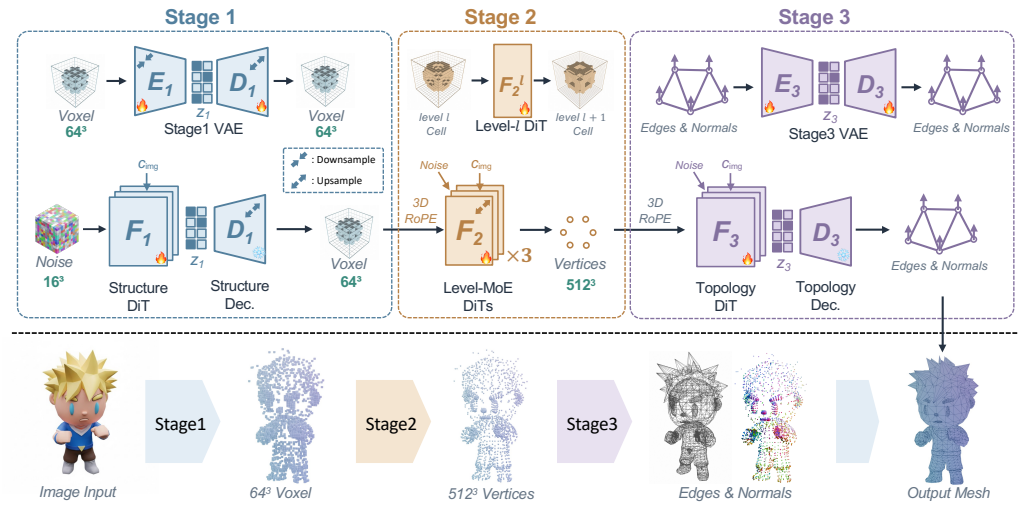

Figure 2: Overview of our proposed TopoFlowForge: Stage 1: Coarse-quantized vertices are generated at 64^3 resolution. Stage 2: Vertex positions are progressively refined from 64^3 to 512^3. Stage 3: Based on the generated vertices, we predict the final vertex connectivity to form mesh faces.

multiple refinement levels progressively upsample and improve its details. Compared with directly upsampling to the final resolution in a single continuous process, this late-stage design provides a better balance between structural stability and high-resolution detail synthesis. Finally, stage 3 generates the per-vertex features through a diffusion model in the latent space. A topology decoder then predicts both the edge set $\hat{\mathcal{E}}$ and face normals, yielding the final faces $\hat{\mathcal{F}}$ with correct orientations. All three stages are conditioned on image features c_{img} extracted by DINOv3 Siméoni et al. (2025).

3.1 Stage 1: Prior Adaptation for Coarse Structure

Given an input image, TopoFlowForge first generates coarse-quantized vertex positions for the target. Concretely, let $\mathbf{O}_{64} \in \{0,1\}^{1\times64\times64\times64}$ denote the coarse occupancy voxel grid at 64^3 resolution, which is then compressed by a VAE encoder E_1 to $\mathbf{z}_1 \in \mathbb{R}^{d_1\times16\times16\times16}$. Firstly, the sparse-structure encoder E_1 predicts the mean $\boldsymbol{\mu}_1$ and standard deviation $\boldsymbol{\sigma}_1$ of a diagonal Gaussian, defining the approximate posterior $q_1(\mathbf{z_1} \mid \mathbf{O}_{64})$ over the latent $\mathbf{z_1}$. Correspondingly, the mirrored decoder D_1 maps $\mathbf{z_1}$ back to a 64^3 field of occupancy logits. The compress process in encoder and mapping process in decoder correspond to the downsampling and upsampling process in Figure. 2, respectively.

$$(\boldsymbol{\mu}_1, \boldsymbol{\sigma}_1) = E_1(\mathbf{O}_{64}), \qquad q_1(\mathbf{z_1} \mid \mathbf{O}_{64}) = \mathcal{N}\big(\boldsymbol{\mu}_1, \mathrm{diag}(\boldsymbol{\sigma}_1^2)\big)\,. \tag{1}$$

Based on the latent $\mathbf{z_1}$ compressed by the VAE, a flow matching(FM) model is trained to accomplish the generation of coarse-quantized vertices from c_{img}. The DiT is trained to predict the deterministic posterior mean $\boldsymbol{\mu}_1(\mathbf{O}_{64})$ from the encoder E_1, standardized channel-wise. With the sampled noise $\mathbf{z}_1^0 \sim \mathcal{N}(0, I)$ and timestep $t \in (0,1)$, we use the linear interpolation path $\mathbf{z}_1^t = (1-t)\mathbf{z}_1^0 + t\mathbf{z}_1$. Its conditional velocity is constant along the path, giving the objective.

$$\mathcal{L}_{\text{S1_DiT}} = \mathbb{E}\Big[\big\|v_\theta(\mathbf{z}_1^t, t, c_{\text{img}}) - (\mathbf{z}_1 - \mathbf{z}_1^0)\big\|_2^2\Big]\,. \tag{2}$$

At inference, the learned velocity field is initialized from Gaussian noise. The resulting latent is then denormalized and decoded by the decoder D_1. Thresholding the decoder logits yields the coarse 64^3 occupancy voxel, serving as the coarse-quantized vertex positions. Since our stage 1 adopts the same architecture as the first stage of Trellis.2Xiang et al. (2025a), we are able to initialize it from the corresponding pretrained checkpoint and fine-tune it directly. This enables us to effectively leverage its learned prior for predicting the coarse object structure from images, while also substantially accelerating our training process.

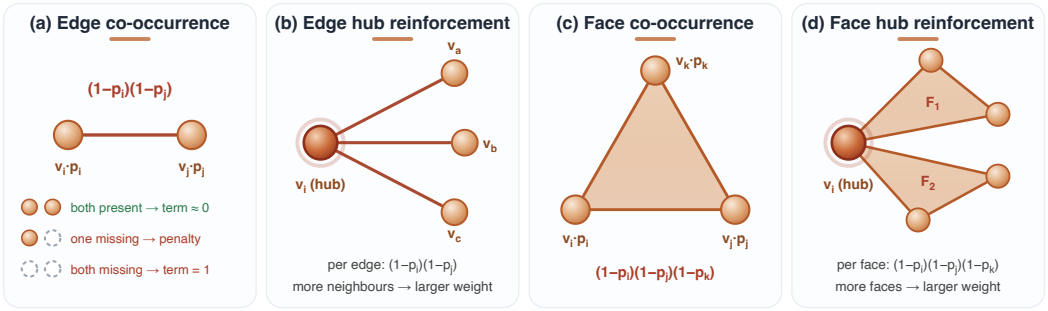

Figure 3: **Topology-aware co-occurrence priors of stage 2.** Two soft-OR co-occupancy penalties complement per-cell cross-entropy with mesh connectivity constraints. Edge and face priors activate only when all connected vertices are jointly predicted empty, naturally reinforcing structurally important high-degree vertices through accumulated constraints.

3.2 Stage 2: Normalized Raw-Space Refinement with Level-MoE

Based on the coarse-quantized vertices, stage 2 aims to refine vertex positions through a multi-resolution refiner, as shwon in Figure. 2. Unlike stage 1, it operates directly in the uncompressed occupancy space without using a VAE. For each active parent cell g, the model incorporates its spatial location through Rotary Positional Encoding (RoPE) and predicts a binary vector $\mathbf{y}^{(g)} \in \{0,1\}^8$ to indicate which of the eight child cells are occupied.

Normalized Bernoulli raw-space prediction To further refine the vertex positions, we need a neural network that can understand the target more directly. We therefore adopt an x-prediction flow-matching model instead of v-prediction. Specifically, we normalize the ground-truth Bernoulli occupancy distribution $\{0,1\}$ to a zero-mean unit-variance distribution based on its statistics. Under this prediction paradigm, various techniques can be applied to further enhance the refinement quality.

Level-indexed mixture of experts. Within the hierarchical DiT framework, each refinement level increases the vertex resolution by a factor of two. As the resolution increases, the occupied fraction ρ_R decreases monotonically, yielding statistically distinct occupancy distributions across levels. We therefore assign a dedicated expert to each resolution, allowing it to specialize in its own occupancy statistics. The experts operate in cascade order, with occupied cells output from one level becoming the input for the next. The cells surviving on the final 512^3 grid provide the refined vertex set $\hat{\mathcal{V}}$.

Training objectives. Every expert is supervised by comparing its predicted child-occupancy probabilities against the binary target $\mathbf{y}^R$. The primary term is a weighted binary cross-entropy that addresses the strong imbalance quantified above. We define $p_c = \sigma(a_c)$ as the predicted occupancy of child cell c, with $y_c \in \{0,1\}$ as its ground-truth label,

$$\mathcal{L}_{\text{bce}} = \frac{\sum_c w_c \, \text{ce}_c}{\sum_c w_c}, \qquad \text{ce}_c = -\big[\alpha \, y_c \log p_c + (1 - y_c) \log(1 - p_c)\big], \tag{3}$$

where the positive-class weight $\alpha > 1$ raises recall on the rare occupied cells, and the spatial weight w_c reshapes the negatives: $w_c = 1$ on occupied cells, $w_c = w_{\text{near}}$ on empty cells lying within a radius r of some occupied cell, which is the thin surface band where false positives concentrate, and $w_c = w_{\text{far}}$ on the remaining bulk-empty cells, with $w_{\text{near}} > w_{\text{far}}$. Because this term scores every cell independently, it constrains neither vertex connectivity nor agreement across a shared edge or face, so we add two topology-aware priors on the same probabilities (Fig. 3). Let $\mathcal{E}$ collect the child-cell pairs containing the two endpoints of each ground-truth edge, and $\mathcal{F}$ collect the child-cell triples containing the three vertices of each ground-truth face. The edge endpoint prior

$$\mathcal{L}_{\text{edge}} = \frac{1}{|\mathcal{E}|} \sum_{(i,j)\in\mathcal{E}} (1 - p_i)(1 - p_j) \tag{4}$$

is a soft-OR co-occupancy penalty: it vanishes as soon as either endpoint is confidently occupied and grows only when both endpoints of a true edge are jointly predicted empty, injecting mesh

connectivity as an occupancy prior over the per-cell independent cross-entropy, as shown in the first panel of Figure 3. Furthermore, the objective can be rewritten as:

$$\mathcal{L}_{\text{edge}} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{V}} \Big(\sum_{j \in N(i)} (1 - p_j) \Big) (1 - p_i) \tag{5}$$

where $N(i)$ represents the neighborhood of vertex i. In this formulation, vertices with more neighbors are assigned larger weights, enhancing the supervision of key points as shown in the second panel of Figure 3. The face prior carries the same soft-OR logic to the three corners of a ground-truth face, as shown in the last two panel of Figure 3,

$$\mathcal{L}_{\text{face}} = \frac{1}{|\mathcal{F}|} \sum_{(i,j,k) \in \mathcal{F}} (1 - p_i)(1 - p_j)(1 - p_k)$$

In summary, the full stage 2 objective, in expectation over resolution R and flow time t, combines the weighted cross-entropy with the three priors,

$$\mathcal{L}_{\text{S2}} = \mathbb{E}_{R,t}[\mathcal{L}_{\text{bce}} + \lambda_{\text{e}} \mathcal{L}_{\text{edge}} + \lambda_{\text{f}} \mathcal{L}_{\text{face}}], \tag{6}$$

whose per-term weights $\lambda_{\text{e}}, \lambda_{\text{f}} \geq 0$ are set to balance the priors against the cross-entropy.

3.3 Stage 3: Per-Vertex Topology Generation

Given the vertex set $\hat{\mathcal{V}}$ from stage 2, stage 3 directly predicts vertex connectivity (i.e. edges $\mathcal{E}$) and per-vertex normal orientations. It combines a topology VAE with a latent flow matching model. The VAE learns a continuous feature for each vertex, while the flow matching models the distribution of these features conditioned on images.

Topology VAE. Unlike the occupancy VAE inherited in stage 1, this VAE performs no spatial compression: it attaches exactly one latent code to each vertex, so the number of latents always equals the vertex count $V = |\hat{\mathcal{V}}|$. It encodes two key pieces of information: how the vertex connects to others, as well as the surface normal at that vertex, as shown in Figure. 2. The vertex coordinates, already determined by stage 2, stay unchanged and are not re-encoded.

The connectivity head determines whether an edge exists between each pair of vertices. A naive design would compare the two endpoint features $(\mathbf{f}_i, \mathbf{f}_j)$ by a symmetric similarity, such as the inner product $\langle W\mathbf{f}_i, W\mathbf{f}_j \rangle$ under a single shared projection. However, this inner product inside one embedding space favours *transitive* connectivity: once A is similar to B and B to C, the score of (A, C) is pushed up as well, which can produce numerous false connections in mesh adjacency prediction. We instead project each vertex into two *distinct* subspaces before matching. Writing $\mathbf{h}_i \in \mathbb{R}^d$ for the feature of vertex i, a shared edge MLP maps it to $\mathbf{f}_i = g_e(\mathbf{h}_i) \in \mathbb{R}^{d_e}$, and two independent bias-free projections form source W_s and destination W_d embeddings,

$$\mathbf{s}_i = W_s \mathbf{f}_i, \qquad \mathbf{d}_i = W_d \mathbf{f}_i, \qquad W_s, W_d \in \mathbb{R}^{d_e \times d_e}. \tag{7}$$

Because mesh edges are undirected, we symmetrise by:

$$a_{ij} = \frac{Q_{ij} + Q_{ji}}{2} = \frac{\mathbf{s}_i^\top \mathbf{d}_j + \mathbf{s}_j^\top \mathbf{d}_i}{2} + b. \tag{8}$$

$a_{ij} = a_{ji}$ is the logit judging whether there is an edge between vertex i and j. The complete $V \times V$ logit matrix is produced in parallel by batched matrix multiplication. At trining, the connectivity term $\mathcal{L}_{\text{edge}}$ is a class-balanced binary cross-entropy over the valid upper-triangular logits. At inference, thresholding these probabilities defines the edge set $\hat{\mathcal{E}}$. Triangular faces are then recovered as the closed three-cycles of this edge graph,

$$\hat{\mathcal{F}} = \{(i, j, k) : i < j < k,\ (i, j), (j, k), (k, i) \in \hat{\mathcal{E}}\}. \tag{9}$$

The recovered edge graph fixes *which* triangles exist but leaves their orientation undefined. Unlike some methods such as LATO.2 Long et al. (2026) adapting post-process to recover face orientation, we instead let the network emit orientation directly. Specifically, we use a normal head in decoder which runs in parallel and regresses a unit normal $\hat{\mathbf{n}}_i$ for each vertex from the same feature $\mathbf{f}_i$,

supervised by a cosine term $\mathcal{L}_{\text{normal}}$ against the ground-truth vertex normals. For a recovered face (i, j, k) with positions $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ and geometric normal $\mathbf{g}_{ijk} = (\mathbf{v}_j - \mathbf{v}_i) \times (\mathbf{v}_k - \mathbf{v}_i)$, the winding is kept when it agrees with the predicted vertex normals and two vertices are swapped otherwise,

$$(i, j, k) \;\mapsto\; \begin{cases} (i, j, k), & \mathbf{g}_{ijk} \cdot (\hat{\mathbf{n}}_i + \hat{\mathbf{n}}_j + \hat{\mathbf{n}}_k) \geq 0, \\ (i, k, j), & \text{otherwise.} \end{cases} \tag{10}$$

The two heads share the decoder and are optimised together with a Gaussian regulariser $\mathcal{L}_{\text{KL}}$ on the per-vertex posteriors, giving the topology VAE objective

$$\mathcal{L}_{\text{S3_VAE}} = \mathcal{L}_{\text{edge}} + \lambda_n \mathcal{L}_{\text{normal}} + \lambda_r \mathcal{L}_{\text{KL}}, \tag{11}$$

in which per-term weights $\lambda_n, \lambda_r \geq 0$ are set to balance the losses.

4 Experiments

4.1 Implementation Details

Datasets and conditioning. We train all stages of TopoFlowForge on the curated dataset, which retains roughly $800{,}000$ meshes after filtering. For evaluation we use the public Toys4K Stojanov et al. (2021) test split together with TopoEval-388(TE-388), a dataset of 388 hand-crafted examples with high quality ground-truth meshes spanning indoor, outdoor, and character categories, which we will release alongside the paper. For more details about trining dataset and network paramenters, please refer to the appendix A.1 and A.3.

4.2 Comparison with other methods

Quantitative comparison. Table 1 reports all six metrics defined in A.3 on Toys4K and our proposed TE-388: Chamfer (CD) and Hausdorff (HD) distances for geometric accuracy, ULIP-I and Uni3D-I for image-to-shape semantic alignment, and FD-Inception and FD-DINOv2 for distribution-level rendering fidelity. We compare with the two publicly available open-source methods: LATO.2 Long et al. (2026) and EdgeRunner Tang et al. (2024b) with single image conditioning. Our method consistently outperforms the baselines across all metrics.

Table 1: Quantitative comparison on Toys4K and TE-388 under single-image conditioning.

Method	CD ↓	HD ↓	ULIP-I ↑	Uni3D-I ↑	FD-Incep. ↓	FD-DINOv2 ↓
Toys4K						
LATO.2 Long et al. (2026)	0.1553	0.3663	0.1674	0.3330	78.9649	616.4222
EdgeRunner Tang et al. (2024b)	0.3264	0.6145	0.1514	0.2347	30.9466	339.7754
TopoFlowForge (Ours)	**0.0655**	**0.2451**	**0.1923**	**0.3509**	**18.1812**	**197.7314**
TE-388						
LATO.2 Long et al. (2026)	0.0991	0.2990	0.1509	0.2978	119.9100	931.6500
EdgeRunner Tang et al. (2024b)	0.3990	0.7102	0.1261	0.1878	67.0844	651.5899
TopoFlowForge (Ours)	**0.0415**	**0.2075**	**0.1829**	**0.3106**	**43.2544**	**363.7326**

Qualitative comparison. The two baselines exhibit characteristic failure modes. EdgeRunner, an autoregressive mesh generation approach that directly produces meshes with reasonably regular topological structure. However, its generation is often unstable, and premature prediction of the end token frequently leads to incomplete mesh outputs. LATO.2, a diffusion-based generator, can recover globally stable shapes, but its local geometry is often under-resolved: vertices are insufficiently dense in detailed regions, resulting in less refined structures and connectivity errors. Benefiting from our late-stage vertex generation strategy, TopoFlowForge achieves strong global shape stability while significantly outperforming LATO.2 in character-level detail, topological regularity, and edge connectivity accuracy.



Figure 4: Qualitative comparison with open-source polygon-based 3D mesh generation models. Given an image as input, our model generates the finest and most complete 3D meshes.

Table 2: Ablation of stage 2 normalisation and topology-aware losses.

Stage 2 variant	CD ↓	HD ↓	ULIP-I ↑	Uni3D-I ↑	FD-Incep. ↓	FD-DINOv2 ↓
w/o. Geometry-aware Losses	0.0771	0.2734	0.1515	0.2501	52.6284	423.8286
w/o. Normalised Target	0.0576	0.2529	0.1647	0.2897	47.2389	390.2450
Ours	**0.0415**	**0.2075**	**0.1829**	**0.3106**	**43.2544**	**363.7326**

Qualitative comparison with commericial models We benchmark our method against two commercial mesh generation models through their online services. As shown in Figure 5, our model achieves a competitive performance in geometric details and image alignment, though the topology is slightly less regular than the commercial baselines.

4.3 Ablation Studies

We evaluate the two design questions specified by our vertex-generation formulation. The first study quantifies the effects of stage 2 level normalisation and geometry-aware losses on TE-388. The second study qualitatively examines whether combining stage 1 and stage 2 as late-stage progressive refinement improves structural stability.

Stage 2 normalisation and geometry-aware losses. We compare three matched Stage 2 variants while keeping Stage 1, Stage 3, training data, and inference settings fixed. The first uses level-wise Bernoulli normalization with only weighted BCE. The second predicts unnormalized raw occupancy with geometry-aware losses added. The full model combines normalization with all geometry-aware terms in Eq. equation 6. As shown in Tab. 2, model performance drops when the geometry-aware losses or the normalization strategy is removed.

Late-stage progressive refinement. We compare the full stage 1-to-stage 2 cascade with a matched single-stage variant that attempts to generate the final 512^3 vertex support directly. As



Figure 5: Qualitative comparison with commercial polygon-based 3D mesh generation models. Given an image, our model generates more detailed and stable meshes with better alignment.

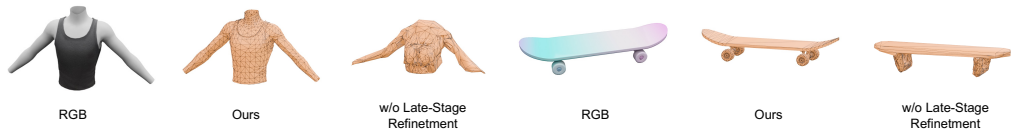

Figure 6: Qualitative ablation of late-stage progressive refinement, comparing continuous vertex generation with the late-stage refinement on structurally challenging examples.

illustrated in fig. 6, replacing the late-stage refinement strategy with direct continuous upsampling degrades both the overall skateboard geometry and the fidelity of the wheel details. The progressive model is expected to preserve the global layout established by stage 1 while stage 2 adds fine vertices within that support, yielding more stable structures than direct high-resolution generation.

5 Conclusion and Limitations

We present TopoFlowForge, a cascaded 3D mesh generation model that produces product-ready content by combining vertex generation with edge prediction. Several effective losses and normal priors are employed to enhance generation quality and image alignment. However, the generated topology remains less regular than manually crafted models. Future work will focus on improving data curation and model architecture to further enhance output quality.

6 AI Usage Statement

In this work, we used generative AI tools for assisting with translation. We have not used them for developing theoretical models or conceptual frameworks, generating synthetic datasets, formulating mathematical claims, or proposing or refining hypotheses. Additionally, we used generative AI tools for creating or edit software code. We have reviewed all AI-assisted work.

REFERENCES

Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023.

Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pp. 16251–16261, June 2025a.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. *arXiv preprint arXiv:2406.10163*, 2024.

Yiwen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13922–13931, 2025b.

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. *arXiv preprint arXiv:2212.08051*, 2022.

Jingyu Guo, Sensen Gao, Jia-Wang Bian, Wanhu Sun, Heliang Zheng, Rongfei Jia, and Mingming Gong. Hyper3d: Efficient 3d representation via hybrid triplane and octree feature for enhanced 3d shape variational auto-encoders. *arXiv preprint arXiv:2503.10403*, 2025.

Zekun Hao, David W. Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024.

Xianglong He, Junyi Chen, Di Huang, Zexiang Liu, Xiaoshui Huang, Wanli Ouyang, Chun Yuan, and Yangguang Li. Meshcraft: Exploring efficient and controllable mesh generation with flow-based dits. *arXiv preprint arXiv:2503.23022*, 2025a.

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. *arXiv preprint arXiv:2503.21732*, 2025b.

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.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

Jeonghwan Kim, Yushi Lan, Armando Fortes, Yongwei Chen, and Xingang Pan. Fastmesh: Efficient artistic mesh generation via component decoupling. *arXiv preprint arXiv:2508.19188*, 2025.

Weiyu Li, Antoine Toisoul, Tom Monnier, Roman Shapovalov, Rakesh Ranjan, Ping Tan, and Andrea Vedaldi. MeshFlow: Efficient artistic mesh generation via MeshVAE and flow-based diffusion transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5849–5858, 2026.

Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, et al. Triposg: High-fidelity 3d shape synthesis using large-scale rectified flow models. *arXiv preprint arXiv:2502.06608*, 2025a.

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*, 2025b.

Hang Long, Tianhao Zhao, Junkai Lin, Youjia Zhang, Huipeng Guo, Rendong Liang, Jiale Xu, Jozef Hladký, Matthias Nießner, and Wei Yang. LATO.2: Factorized 3D mesh generation with vertex and topology flow. *arXiv preprint arXiv:2607.10623*, 2026.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pp. 7220–7229. PMLR, 2020.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4195–4205, 2023.

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19615–19625, 2024.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. *arXiv preprint arXiv:2101.07296*, 2021.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, 2016.

Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024a.

Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024b.

Hanxiao Wang, Yuan-Chen Guo, Ying-Tian Liu, Zi-Xin Zou, Biao Zhang, Weize Quan, Ding Liang, Yan-Pei Cao, and Dong-Ming Yan. FACE: Face-based autoregressive representation for high-fidelity and efficient mesh generation. *arXiv preprint arXiv:2603.01515*, 2026a.

Hanxiao Wang, Ying-Tian Liu, Yuan-Chen Guo, Qi-Yuan Feng, Zi-Xin Zou, Ding Liang, Biao Zhang, and Yan-Pei Cao. Nexus: Native mesh generation with diffusion. *arXiv preprint arXiv:2607.13563*, 2026b.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in neural information processing systems*, 36:8406–8441, 2023.

Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, et al. Scaling mesh generation via compressive tokenization. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11093–11103, 2025.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3D: Scalable image-to-3d generation via 3d latent diffusion transformer. In *Advances in Neural Information Processing Systems*, volume 37, pp. 121859–121881, 2024.

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip Torr, et al. Direct3d-s2: Gigascale 3d generation made easy with spatial sparse attention. *arXiv preprint arXiv:2505.17412*, 2025.

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. *arXiv preprint arXiv:2512.14692*, 2025a.

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 Computer Vision and Pattern Recognition Conference*, pp. 21469–21480, 2025b.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. ULIP: Learning a unified representation of language, images, and point clouds for 3D understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1189, 2023.

Xianghui Yang, Huiwen Shi, Bowen Zhang, Fan Yang, Jiacheng Wang, Hongxu Zhao, Xinhai Liu, Xinzhou Wang, Qingxiang Lin, Jiaao Yu, Lifu Wang, Jing Xu, Zebin He, Zhuo Chen, Sicong Liu, Junta Wu, Yihang Lian, Shaoxiong Yang, Yuhong Liu, Yong Yang, Di Wang, Jie Jiang, and Chunchao Guo. Hunyuan3D 1.0: A unified framework for text-to-3d and image-to-3d generation. *arXiv preprint arXiv:2411.02293*, 2024.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions On Graphics (TOG)*, 42(4):1–16, 2023.

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 Transactions on Graphics (TOG)*, 43(4):1–20, 2024.

Ruowen Zhao, Junliang Ye, Zhengyi Wang, Guangce Liu, Yiwen Chen, Yikai Wang, and Jun Zhu. Deepmesh: Auto-regressive artist-mesh creation with reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10612–10623, 2025a.

Tianhao Zhao, Youjia Zhang, Hang Long, Jinshen Zhang, Wenbing Li, Yang Yang, Gongbo Zhang, Jozef Hladký, Matthias Nießner, and Wei Yang. LATO: 3D mesh flow matching with structured topology-preserving latents. *arXiv preprint arXiv:2603.06357*, 2026.

Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv preprint arXiv:2501.12202*, 2025b.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3D: Exploring unified 3D representation at scale. In *International Conference on Learning Representations (ICLR)*, 2024.

A Appendix

A.1 Training Data

Data sources. TopoFlowForge is trained on a large and intentionally heterogeneous mesh dataset drawn from three complementary sources: 1. A large-scale in-house 3D dataset, dominated by everyday indoor commodities models, most of which are manually reconstructed to ensure high-quality geometry and topology. 2. Manually crafted in-house dataset, which densifies the categories our catalogue covers only sparsely. 3. Public open-source dataset, including Objaverse Deitke et al. (2022) and TexVerse. The scale and diversity of our training dataset are essential for topological generalization and explain why TopoFlowForge transfers well to in-the-wild product imagery. Every mesh in the dataset is decimated to at most $20{,}000$ faces and normalised into the $[-1, 1]^3$ cube before occupancy and topology targets are extracted. Image conditioning is generated by rendering each asset in Blender at 1024^2 resolution. The renders are encoded by a single frozen DINOv3 Siméoni et al. (2025) backbone that is shared across all three stages, so inference reduces to one image forward pass with no 3D encoder in the loop. For every asset, we render a full 360° ring of surrounding

views with a physically based renderer, providing the paired image-mesh supervision for training. Although most of these meshes already have clean, well-structured topology, the raw collection still contains many low-quality, noisy samples that must be filtered out before they are usable for topology supervision.

Data curation. A carefully designed multi-stage curation pipeline is adopted to filter the raw dataset. We first bucket all assets by face count and sample the buckets as evenly as possible during training, so that neither low-poly nor dense meshes dominate the gradient and the model observes the full spectrum of mesh complexity. We then screen the entire dataset with a suite of classical topology statistics, including the fraction of non-manifold faces, the distribution of vertex degrees (the number of edges incident to each vertex), and the spatial uniformity of the vertex layout, keeping only assets whose connectivity is regular enough to supply a learnable edge-flow signal. Beyond these hand-crafted criteria, we further build a VLM-based agent that inspects the rendered views of each surviving asset and prunes the residual low-quality cases that the numeric filters overlook, progressively refining the corpus down to the artist-grade topology on which TopoFlowForge is trained. We will open-source our data filtering code, along with our model weights and codebase.

A.2 Gallery for More Results

Figure. 7 presents additional image-conditioned generation results. It can be seen that our method yields meshes with strong global structural stability and fine geometric details, and generalizes effectively across a wide range of categories.

A.3 Implementation Details

Stage-1 coarse-occupancy model. Stage 1 reuses the TRELLIS sparse-structure modules of Sec. 3.1 without architectural change, coupling an occupancy VAE with a latent flow DiT. The VAE is a symmetric 3D convolutional encoder-decoder with channel widths $\{32, 128, 512\}$ and eight latent channels; it compresses the 64^3 binary occupancy grid into a $16^3 \times 8$ latent and mirrors it back to 64^3 logits, with occupancy read off at a zero threshold. The denoiser is a 30-block transformer with hidden size 1536, 12 attention heads, an FFN ratio of 5.33, and 3D rotary position embeddings, totalling roughly 1.3B parameters. It operates on the $16^3 \times 8$ latent, cross-attends to the frozen DINOv3 tokens c_{img}, and is finetuned with the velocity objective of Eq. equation 2 after the channel-wise latent standardisation. At inference, Euler integration yields a latent that is denormalised and decoded by the frozen VAE.

Stage-2 level-MoE refinement. The cascade of Sec. 3.2 runs directly in raw occupancy space without a VAE. It comprises three structurally identical but parameter-independent experts assigned one-to-one to levels 6, 7, and 8, i.e. the $64^3 \to 128^3$, $128^3 \to 256^3$, and $256^3 \to 512^3$ transitions. Each expert is a 28-block transformer with hidden size 1536, 12 heads, and an FFN ratio of 5.33: a linear layer reads the eight-dimensional per-cell child-occupancy state, 3D RoPE over cell centres and a level embedding are injected, cross-attention consumes the DINOv3 tokens c_{img}, and a linear head regresses the eight child logits. Targets are standardised per level, supervised under Eq. equation 6, and integrated in normalised space through the induced velocity.

Stage-3 topology model. Stage 3 pairs a per-vertex topology VAE with a latent flow DiT (Sec. 3.3). The VAE encoder is a VecSet Zhang et al. (2023)-style network of hidden width 1024 with four graph-convolution layers over the vertex neighbourhood followed by six cross-attention and four self-attention layers (8 heads) that pool the mesh into 2048 latent tokens, emitting one 128-dimensional Gaussian latent per vertex. The decoder (1024-wide, six self-attention and four query layers, 8 heads) scores pairwise connectivity with the bilinear head of Eq. equation 8 through a 512-dimensional edge branch and regresses unit vertex normals through a 256-dimensional normal branch, trained under Eq. equation 11. The latent DiT is a 32-layer transformer with hidden size 1536 and 24 heads that injects the Stage-2 vertex coordinates through voxel RoPE, cross-attends to the DINOv3 tokens, and predicts the per-vertex latent flow. Among the three stages, only the Stage-1 denoiser inherits an absolute parameter budget from TRELLIS ($\approx$ 1.3B); the Stage-2 and Stage-3 backbones are reported through their width, depth, and head counts.



Figure 7: Additional image-conditioned results. Our method generates meshes with both strong global structural stability and high-fidelity local details.*Best viewed with zoom-in.*

The three stages are trained on a shared dataset and can be optimized simultaneously. Stage 1 is initialized from the pretrained checkpoint of the Trellis Xiang et al. (2025b) structural voxel generation model and further fine-tuned on our own dataset. Its VAE is finetuned on 4 GPUs just for several hours, whereas its DiT is trained on 8 GPUs and converges in approximately 7 days. The

DiT in Stage 2 is trained on 32 GPUs for around 14 days. For Stage 3, the VAE is trained on 8 GPUs for about 3 days, and the DiT is trained on 16 GPUs for roughly 7 days.

Evaluation metrics. We assess geometry with two surface distances, semantics with two image-shape alignment scores, and rendering fidelity with two Fréchet feature distances.

- *Chamfer Distance* (CD) is computed between dense point samples of the generated and ground-truth surfaces as the average of the two directional mean nearest-neighbour distances, $\mathrm{CD} = \frac{1}{2}(\overline{d}_{p\to g} + \overline{d}_{g\to p})$, and reflects mean surface deviation; lower is better.
- *Hausdorff Distance* (HD) is taken on the same point samples as the maximum of the two directional worst-case distances, and is therefore sensitive to outlier faces and broken topology; lower is better.
- *ULIP-I* Xue et al. (2023) measures whether the generated shape matches the conditioning image in a joint embedding space: we sample $8{,}192$ surface points, normalise them to the unit sphere, encode them with the ULIP point encoder, and report the cosine similarity to the embedding of the input image produced by its paired image tower; higher is better.
- *Uni3D-I* Zhou et al. (2024) follows the same image-to-shape alignment protocol with $10{,}000$ points and the larger Uni3D point encoder aligned to an EVA-CLIP image tower; higher is better.
- *FD-Inception* is a distribution-level Fréchet distance over multi-view renders of the generated and reference meshes under a shared untextured shading protocol, embedding each render set with an InceptionV3 Szegedy et al. (2016) network (2048-dimensional pooled features) and reporting $\|\boldsymbol{\mu}_g - \boldsymbol{\mu}_r\|_2^2 + \mathrm{tr}\big(\boldsymbol{\Sigma}_g + \boldsymbol{\Sigma}_r - 2(\boldsymbol{\Sigma}_g\boldsymbol{\Sigma}_r)^{1/2}\big)$ between the two Gaussians fitted to these features Heusel et al. (2017); lower is better.
- *FD-DINOv2* applies the same Fréchet distance to features from a DINOv2 ViT-L/14 Oquab et al. (2023) encoder (1024-dimensional class token); lower is better.