MESHEXPAND: ORDER-AGNOSTIC ARTISTIC MESH GENERATION VIA LOCAL EXPANSION

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce MeshExpand, an order-agnostic and local framework for generating artistic meshes from input 3D shapes at high resolution. Recent learning-based methods formulate mesh generation as global sequence prediction, which couples generation to a predefined serialization order and can require long contexts as mesh resolution grows. Instead, we formulate mesh generation as learned local surface expansion. Starting from one or more seed triangles, MeshExpand iteratively selects an open boundary edge and predicts the opposite vertex from a bounded neighborhood of nearby faces and point samples, together with a global shape prior. This formulation maintains a constant-size local context per generation step that is independent of the output mesh resolution. Representing previously generated faces as an unordered set further removes dependence on a fixed generation order and enables the same model to support arbitrary growth directions, partial mesh completion, and local editing. MeshExpand also provides global and local resolution control by coupling triangle size with input point density, and uses online geometric validation and resampling to suppress invalid mesh configurations during generation. Experiments show that MeshExpand scales to meshes with up to 120K faces using less than 2 GB of GPU memory, transfers to out-of-distribution scanned and generated shapes with fewer degradation and improves geometric fidelity, orientation consistency, and manifoldness. Our code and model weights will be made publicly available upon paper acceptance.

1 INTRODUCTION

High-quality 3D assets are increasingly important for AR/VR/XR, gaming, and film production. A standard asset-creation pipeline often involves two stages. Artists first sculpt high-resolution geometry with dense polygons to capture fine surface details using tools such as ZBrush. These meshes are typically unsuitable for animation or real-time rendering because they contain excessive polygons and irregular topology. Artists therefore perform *retopology* to construct a compact and structured mesh that preserves the geometry and supports smooth shading and coherent deformation. Although this process produces high-quality assets, manual retopology remains time-consuming and requires substantial expertise.

Recent 3D generative frameworks broadly follow this two-stage paradigm. Latent diffusion models and related 3D generators (Wang et al., 2023; Xiang et al., 2025) first synthesize geometry as implicit fields or latent fields, from which surface meshes can be extracted. The resulting meshes often contain dense and irregular triangulations that are unsuitable for direct use by artists. A line of work (Chen et al., 2024b;c; Lionar et al., 2025; Liu et al., 2025) therefore tries to automate the second stage of retopology. They learn to reconstruct compact meshes from these geometric representations, using artist-created meshes as supervision. Following this line of work, we refer to such outputs as *artistic meshes*: meshes learned from artist-created data that are compact and regular, with clean edge flow, so that they remain usable for downstream editing. Most existing methods formulate artistic mesh generation as global sequence prediction: mesh vertices, edges, or faces are serialized according to a predefined traversal order and decoded autoregressively. This order-dependence also makes arbitrary partial completion and local modification difficult.

An alternative line of work (Shen et al., 2024; Kim et al., 2025; Wang et al., 2026) avoids autoregressive serialization by predicting the whole mesh using diffusion in parallel. These approaches

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.

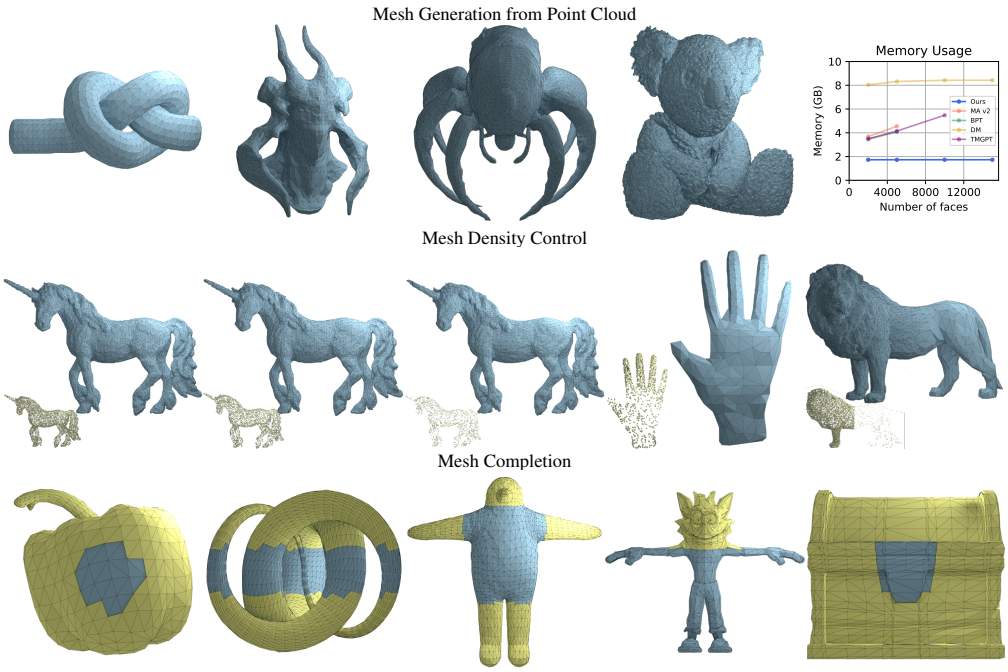

Figure 1: MeshExpand generates artistic meshes with up to 120K faces while using less than 2 GB of GPU memory. It supports global and local resolution control through the input point cloud (yellow), as well as mesh completion from partial inputs (yellow). Online geometric validation substantially reduces self-intersections and non-manifold configurations during generation. Additional results are provided in the Appendix A.

also substantially reduce generation latency, but they still reason over large sets of vertices and may require considerable computation and memory at high resolutions. More fundamentally, both paradigms treat mesh generation as a largely global prediction problem. This motivates us to reconsider mesh generation as a local geometric prediction problem rather than a global generation problem.

We introduce MeshExpand, which formulates artistic mesh generation as *learned local surface expansion*. Our approach revisits the classical seed-expansion paradigm (Bernardini et al., 1999; Cohen-Steiner & Da, 2004), but replaces hand-crafted geometric rules with a predictor of mesh connectivity learned from artist-created meshes. Unlike learned triangulation methods (Sharp & Ovsjanikov, 2020; Rakotosaona et al., 2021) that select triangles among candidates formed from input points, MeshExpand generates new vertices directly and can therefore place them away from the input samples. Starting from one or more seed triangles, MeshExpand repeatedly selects an open boundary edge and predicts the opposite vertex from a bounded neighborhood of nearby faces and point samples, together with a global shape prior. Each generation step therefore uses a bounded local context that is independent of the final mesh resolution.

This local formulation also removes the need for a fixed generation order. We represent previously generated faces as an unordered set, which allows the same model to expand the mesh from different boundary edges and under different traversal strategies. As a result, MeshExpand supports arbitrary growth directions, partial mesh completion, and local editing without changing the generation model.

Beyond flexible generation, MeshExpand provides direct control over mesh resolution. We associate local point density with triangle scale during training, so that input point density acts as an explicit control signal at inference. This allows the generated mesh density to be adjusted globally or within selected regions through the input point cloud. We further combine learned vertex prediction with online geometric validation and resampling. Candidate triangles are checked for self-intersections, non-manifold configurations, degeneration, and inconsistent orientation before insertion. This hybrid design prevents geometric defects from accumulating during generation. Representative results are shown in Fig. 1.

In summary, our main contributions are:

- We formulate artistic mesh generation as *learned local surface expansion* rather than global sequence decoding. The resulting model operates with a bounded local context at each generation step and generates meshes with up to 120K faces using less than 2 GB of GPU memory.
- We introduce an order-agnostic generation mechanism based on unordered local face sets. A single model, trained once, performs under depth-first, breadth-first, top-down, and bottom-up growth orders and supports partial mesh completion and local editing without retraining.
- We develop density-guided resolution control and online geometric validation. The former provides global and spatially varying control over triangle density, and the latter reduces self-intersecting faces from 6.3 to 0.41 and open boundary edges from 639.7 to 19.7 per mesh.

2 Related Work

Mesh Generation from Point Clouds. Surface reconstruction from point clouds is a long-standing problem in geometry processing. Global methods such as Poisson reconstruction (Kazhdan et al., 2006) estimate an implicit field from oriented points and extract an isosurface using an adaptive variant of Marching Cubes (Lorensen & Cline, 1987). Classical methods like Ball Pivoting (Bernardini et al., 1999) and Advancing Front (Cohen-Steiner & Da, 2004) grow meshes via geometric criteria, backed by theoretical guarantees under strict sampling assumptions (Petitjean & Boyer, 2001; Boissonnat & Oudot, 2005). Learning-based counterparts replace these criteria with learned predictors—such as IER (Liu et al., 2020) for distance ratios, PointTriNet (Sharp & Ovsjanikov, 2020) for candidate classification, and DSE (Rakotosaona et al., 2021) for local Delaunay elements. While these methods construct triangles among input points, their objective targets surface reconstruction rather than artist-created topology. MeshExpand retains the local expansion paradigm but predicts new vertex positions directly, decoupling vertex placement from input samples and creating artistic topology.

Autoregressive Mesh Generation. Advances in 3D content creation (Li et al., 2024) have produced strong text- and image-conditioned shape generators (Hong et al., 2024; Xu et al., 2024; Chen et al., 2025; Xiang et al., 2025; Wang et al., 2023). Their outputs can require remeshing or retopology to meet downstream requirements for mesh complexity and connectivity. To generate artist-like topology, many methods serialize mesh geometry and connectivity into token sequences and model them autoregressively (Siddiqui et al., 2024; Weng et al., 2024a). Coordinate-based approaches such as MeshXL (Chen et al., 2024a) encode each triangle using nine coordinate tokens, producing long sequences as mesh resolution increases. Other methods (Chen et al., 2024c; Tang et al., 2024; Lionar et al., 2025; Wang et al., 2025; Song et al., 2025) reduce redundancy through connectivity-aware traversal or compact tokenization. BPT (Weng et al., 2024b)and DeepMesh (Zhao et al., 2025) compresses sequences through block-wise coordinate indexing and patch aggregation, while Meshtron (Hao et al., 2024) combines an hourglass architecture with sliding-window inference. These methods nevertheless generate meshes according to a prescribed serialization or traversal scheme. Furthermore, the sliding-window attention in such serialization is bounded by sequence position rather than surface proximity, so spatially adjacent faces generated much earlier may fall outside the available context. In contrast, MeshExpand predicts new triangles at open boundary edges using spatially selected local neighborhoods. Its unordered face representation avoids requiring a canonical ordering of the conditioning faces, allowing the same predictor to be applied under different growth strategies.

Diffusion Mesh Generation. To bypass the ordering dependency inherent in autoregressive model, SpaceMesh (Shen et al., 2024), FastMesh (Kim et al., 2025) and Nexus (Wang et al., 2026) use diffusion-based generation with continuous representations of mesh connectivity. These approaches reduce reliance on sequential face decoding, maximize parallelism to reduce latency, but operate on representations whose size grows with the number of vertices. MeshExpand instead formulates mesh construction as learned surface growth at the granularity of a single face. Each prediction uses a bounded set of neighboring faces and point samples together with a global shape prior, so the per-step predictor context is independent of the final mesh resolution. This bounds the local predictor's context requirements as the mesh grows, although storage of the generated mesh and geometric validation structures still scales with output size. Faces are added directly to the growing mesh without a separate patch assembly stage.

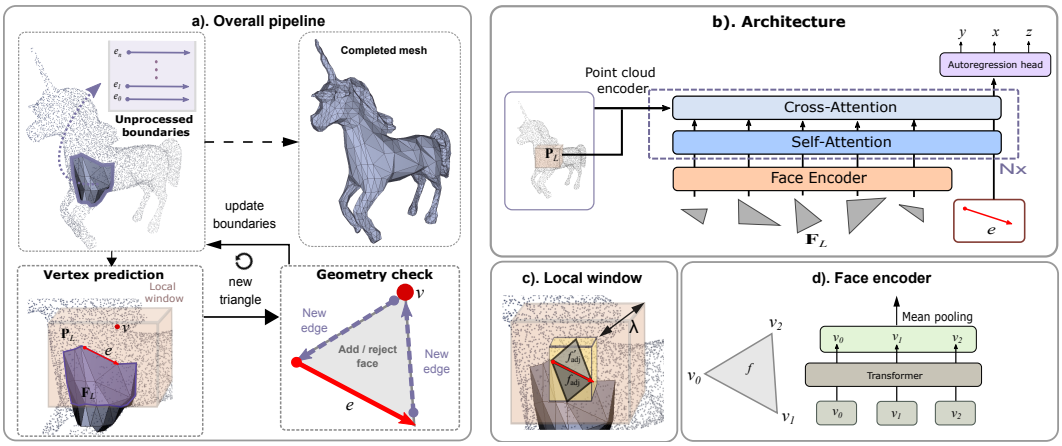

Figure 2: **(a) Overall pipeline:** MeshExpand grows the mesh by selecting an open boundary edge e, predicting its opposite vertex v, validating the candidate face, and updating the boundary set. **(b) Architecture:** A Transformer integrates local faces, local points, and a global shape prior through self- and cross-attention to predict the next vertex or a boundary token. **(c) Local window:** For each query edge, the adjacent face and its surrogate define an axis-aligned bounding box, which is dilated by λ_f for faces and λ_p for points to construct a bounded local context independent of the final mesh resolution. **(d) Face encoder:** A permutation-invariant encoder with pooling produces face features that are independent of vertex ordering.

Controllable Mesh Generation. Existing learned mesh generators provide several forms of control. EdgeRunner (Tang et al., 2024) and Meshtron (Hao et al., 2024) support global face-count conditioning, while SpaceMesh (Shen et al., 2024) demonstrates mesh repair. These capabilities motivate exploration of models that combine local completion with spatially varying resolution control. The local formulation of MeshExpand supports these operations within a single model. Growth can start from supplied open boundaries to enable partial completion, hole filling, and local editing without retraining. Density-aware training associates local point density with triangle scale, allowing resolution to be adjusted globally or within selected regions through the input point cloud. Online geometric validation and resampling complement these operations by checking candidate faces for self-intersections, non-manifold configurations, degeneracy, and inconsistent orientation, and rejecting candidates that fail these checks. Together, these mechanisms allow flexible mesh growth, mixed-resolution generation, and completion subject to geometric and topological validation.

3 Our Method

Overview. Our method generates a connected 2-manifold mesh as a set of triangles $\mathbf{F} = \{f \mid f = (v_0, v_1, v_2), v_i \in \mathbb{R}^3\}$, aligned with an input point cloud and associated normals $\mathbf{P} \in \mathbb{R}^{N\times 6}$, where N denotes the number of input points. As illustrated in Fig. 2a, MeshExpand grows the mesh through local geometric prediction rather than global mesh prediction. Starting from one or more seed triangles, MeshExpand initializes an unprocessed boundary set $\mathcal{E}$ with their edges. At each iteration, an edge $e \in \mathcal{E}$ is selected and removed from the set. If e is no longer an open boundary, it is discarded. Otherwise, MeshExpand predicts an opposite vertex: $v \sim P(v \mid e, \mathbf{F}_L, \mathbf{P}_L, \mathbf{z}_G) \in \mathbb{R}^3$, conditioned on the query edge e, the local faces $\mathbf{F}_L$, the local point samples $\mathbf{P}_L$, and a global shape prior $\mathbf{z}_G$. When the prediction is not a boundary indicator, we form a candidate face f' by pairing v with the two endpoints of e. The candidate is accepted only if it passes the geometric validity checks. The two new edges introduced by an accepted face are added to $\mathcal{E}$ only if they remain open boundaries. This local predict–validate–update process continues until no unprocessed boundary edge remains. The full generation procedure is summarized in the Appendix C.

3.1 Adaptive Local Window

To predict the next vertex v from a query boundary edge e, we restrict input to an adaptive local window that captures sufficient geometry without processing the entire growing mesh. Specifically, we construct a surrogate triangle f'_{adj} such that the adjacent triangle f_{adj} and f'_{adj} form a paral-

lelogram (Fig. 2c). Using the axis-aligned bounding box of this symmetric union rather than f'_{adj} alone reduces directional cues in local configuration (see Appendix B). We then dilate the bounding box by factors λ_f and λ_p to collect local faces $\mathbf{F}_L$ and points $\mathbf{P}_L$, respectively. Because spatial windows naturally adapt to local triangle size and point-triangle densities are coupled during training (Sec. 3.3), the number of collected faces and points remains stable across resolutions. Finally, we cap the local context at $N_{f\,\max}$ nearest faces and $N_{p\,\max}$ subsampled points for the worst case. Empirically, this hard cap is triggered in only 5% of cases.

3.2 Vertex Prediction

We formulate the next-vertex prediction as $v \sim P(v \mid e, \mathbf{F}_L, \mathbf{P}_L, \mathbf{z}_G)$, conditioned on the query edge e, local faces $\mathbf{F}_L$, local point samples $\mathbf{P}_L$, and global shape prior $\mathbf{z}_G$. The local inputs provide geometric and topological information around the current boundary, and the global prior provides shape-level context. We describe each representation and the prediction architecture below.

Query Edge (e). We represent the query edge $e := (v_0, v_1)$ as a token by concatenating the sinusoidal positional embeddings $\text{PosEmb}(\cdot)$ (Zhang et al., 2023) of its two vertices:

$$e = \text{MLP}(\text{PosEmb}(v_0) \oplus \text{PosEmb}(v_1)) \,. \tag{1}$$

Global Prior ($\mathbf{z}_G$). We use global point-cloud conditioning to provide shape-level context throughout generation. We uniformly sample N_G points and their associated normals to form $\mathbf{P}_G \in \mathbb{R}^{N_G \times 6}$. We encode $\mathbf{P}_G$ using the pretrained and frozen Michelangelo encoder (Zhao et al., 2023) to obtain $\mathbf{z}_G \in \mathbb{R}^{256 \times C}$, where C denotes the hidden feature dimension. We compute $\mathbf{z}_G$ once and keep it fixed throughout the generation process.

Local Point Cloud ($\mathbf{P}_L$). The local point cloud $\mathbf{P}_L \in \mathbb{R}^{N_p \times 6}$ contains N_p points and their normals collected within the adaptive local window. We first encode their spatial coordinates using sinusoidal positional embeddings and then process the resulting features with a shallow self-attention Transformer encoder. These features provide local geometric evidence for predicting the next vertex.

Local Faces ($\mathbf{F}_L$). The local face set $\mathbf{F}_L \in \mathbb{R}^{N_f \times 9}$ contains the N_f triangles within the local window, with each face represented by the coordinates of its three vertices. Rather than imposing a fixed vertex ordering, we encode each triangle using a permutation-invariant set function (Zaheer et al., 2017):

$$f := \rho(\{\text{PosEmb}(v_0), \text{PosEmb}(v_1), \text{PosEmb}(v_2)\}) \,, \tag{2}$$

where $\rho(\cdot)$ is implemented using a shallow Transformer encoder followed by mean pooling (Fig. 2d). This representation makes the face feature invariant to permutations of its vertices.

Architecture. Our prediction architecture is illustrated in Fig. 2b. We use a Transformer to integrate the query edge, local faces, local point samples, and global prior. We omit positional embeddings over the face tokens to avoid imposing an ordering on $\mathbf{F}_L$. Self-attention over the encoded faces captures local topological structure, and cross-attention to $\mathbf{P}_L$ and $\mathbf{z}_G$ incorporates local geometry and global shape context. The query edge e serves as the query token, from which the Transformer produces the vertex latent $\mathbf{h}_v$.

Vertex Decoding. Prior mesh generation methods often represent a vertex using separate tokens for its quantized x-, y-, and z-coordinates and decode them autoregressively (Chen et al., 2024a). Inspired by TreeMeshGPT (Lionar et al., 2025), we instead use a lightweight Transformer decoder conditioned on $\mathbf{h}_v$ to predict either the quantized vertex coordinates:

$$P(v \mid \mathbf{h}_v) = P(y \mid \mathbf{h}_v)\, P(x \mid y, \mathbf{h}_v)\, P(z \mid y, x, \mathbf{h}_v), \tag{3}$$

or a boundary label:

$$P(v \mid \mathbf{h}_v) = P(\perp \mid \mathbf{h}_v), \tag{4}$$

where $\perp$ indicates that no new face should be added across the query edge. This design separates contextual reasoning from coordinate decoding and keeps the coordinate autoregression lightweight, which also reduces the cost of subsequent resampling.

Triangle Generation. We sample a batch of S candidate vertices from $P(v \mid \mathbf{h}_v)$ and evaluate them in decreasing order of probability. For each candidate vertex v, we form a triangle f using v and the two endpoints of e. We accept the candidate only if it passes the following geometric checks:

- **Self-intersection check**: rejects f if it intersects any existing triangle.
- **Non-manifold check**: rejects f if it creates a non-manifold configuration, such as three faces sharing an edge.
- **Degeneration check**: rejects f if any two of its vertices are identical.
- **Normal flipping check**: rejects f if $n_f^\top n_{f_{\text{adj}}} < -0.9$, where n_f and $n_{f_{\text{adj}}}$ denote the normals of f and its adjacent face.

We add the first candidate that passes all checks to the mesh. If no candidate is accepted, the query edge is treated as a boundary. An accepted triangle may close existing boundaries or introduce new open edges, and we update the unprocessed boundary set $\mathcal{E}$ accordingly.

3.3 Training

We design our training procedure to support the local and order-agnostic formulation of MeshExpand. Our training has three objectives. First, we expose the model to different local mesh configurations so that generation does not depend on a fixed traversal order. Second, we reduce reliance on incidental cues from previously generated faces and encourage the model to use the input geometry and normals. Third, we couple local point density with triangle scale so that mesh resolution can be controlled through the input point cloud.

Order-Agnostic Sequential Training. Given a 2-manifold triangular mesh $\mathbf{F}$ with N faces, we first select a seed face f_1 uniformly at random and perform a Depth-First Search (DFS) traversal over the mesh connectivity:

$$(f_1, f_2, \ldots, f_N) := \text{DFS}(\mathbf{F}, f_1).$$

For each face f_i, we record the traversal edge $e_{i\to j}$ that leads to an adjacent face f_j, or $e_{i\to\perp}$ if the edge lies on the mesh boundary. During training, we randomly sample one recorded edge as the query edge e and construct the corresponding local context and prediction target. To simulate partially generated meshes, we partition the faces according to the DFS order:

$$\mathcal{P}^-(f_i) = \{f_j \mid j < i\}, \tag{5}$$

$$\mathcal{P}^+(f_i) \sim \text{Dropout}(\{f_j \mid j > i\}, p_{\text{drop}}), \tag{6}$$

where $\mathcal{P}^-(f_i)$ contains previously visited faces and $\mathcal{P}^+(f_i)$ contains a randomly retained subset of future faces with $p_{\text{drop}} = 0.9$. We construct the local face set $\mathbf{F}_L$ by restricting $\mathcal{P}^-(f_i) \cup \mathcal{P}^+(f_i)$ to the adaptive local window around e. The prediction target is the vertex of f_j opposite to $e_{i\to j}$ or the boundary label $\perp$. DFS ordering with random seed masks roughly half the local triangles around e, but do not constrain the inference order. Our experiments show that the trained model can generate meshes under DFS, Breadth-First Search (BFS), top-down, and bottom-up growth orders, which also enables partial completion and local editing.

MAE Pretraining. Sequential training mimics inference but risks error propagation by biasing predicted orientation toward previous generated faces rather than input geometry. We address this issue with an MAE-style pretraining stage (He et al., 2022). We uniformly sample a query edge e and randomly drop faces from its local neighborhood with probability $p_{\text{drop}} = 0.5$. Reducing any ordering context encourages the model to infer the orientation of next triangle from the point cloud normals. Further analysis is provided in the Appendix B.

Density-Guided Resolution Control. We train MeshExpand to associate local point density with triangle scale. Instead of sampling a fixed number of points for each shape, we sample approximately four points per triangle, which preserves a consistent relationship between input point density and local mesh resolution. At inference, we control the generated triangle size by adjusting the point density globally or within selected regions. This provides both global and spatially varying resolution control without changing the generation model. We demonstrate this behavior in Fig. 1 and Fig. 9 in appendix.

4 Experiments

Implementation Details. Our main Transformer consists of 12 encoder layers with cross-attention and a hidden dimension of 512. The model contains approximately 90M trainable parameters. We

Table 1: Quantitative results on Objaverse, GSO and Trellis dataset show that MeshExpand provides the best overall generation metrics and consistency across datasets.

Method	Objaverse					GSO + Trellis				
	CD↓ $\times 10^{-2}$	HD↓ $\times 10^{-2}$	NC↑	|NC|↑	AD↓ %	CD↓ $\times 10^{-2}$	HD↓ $\times 10^{-2}$	NC↑	|NC|↑	AD↓ %
Traditional Methods										
Advancing Front	1.66	15.6	-0.01	0.83	11.0	1.46	15.6	-0.04	0.71	7.05
Scale-Space	3.12	26.0	0.68	0.87	29.9	3.52	24.1	0.61	0.81	24.9
Learning-based Methods										
MeshAnything V2	2.46	12.8	-0.06	0.90	8.3	6.13	42.6	-0.09	0.66	146
BPT	4.25	10.6	0.76	0.87	12.5	7.80	58.5	0.57	0.73	37.5
TreeMeshGPT	1.68	**9.28**	0.68	0.92	**3.63**	5.89	44.9	0.49	0.72	31.3
DeepMesh	3.01	18.8	0.67	0.92	26.6	14.9	90.1	0.41	0.69	83.9
FastMesh	1.57	10.2	0.02	0.88	204	4.96	36.3	0.00	0.70	341
Ours	**1.55**	12.2	**0.92**	**0.94**	5.4	**1.25**	**11.3**	**0.91**	**0.93**	**3.6**

train on 8×A100 GPUs for approximately five days with a batch size of 128 per GPU. Each training sample corresponds to one query edge and its associated local context. We use MAE pretraining for the first 30% of the training steps, followed by sequential training for the remaining 70%. Additional implementation details are provided in the Appendix C.

Training Dataset. We train MeshExpand on the TreeMeshGPT training set, a high-quality subset of Objaverse (Deitke et al., 2023). We apply 8-bit coordinate discretization. After preprocessing and filtering, we retain 30,000 meshes, with 1% reserved for validation and the remainder used for training. The Appendix C provides details of the preprocessing criteria and data augmentation. For each mesh, we sample points and their associated normals to construct the input point cloud, together with one triangle as the seed for training and generation.

Metrics. We evaluate the geometric and structural quality using the following metrics:

- **Chamfer Distance (CD) and Hausdorff Distance (HD).** Lower CD indicates higher geometric fidelity, and lower HD indicates better surface coverage.
- **Normal Consistency (NC) and Absolute Normal Consistency (|NC|).** Higher NC indicates better orientation consistency and fewer normal-flipping artifacts. |NC| ignores the normal sign and therefore measures geometric detail.
- **Area Difference (AD).** AD measures the relative surface-area discrepancy between the generated and reference meshes. It captures structural artifacts such as redundant or clustered triangles that may not be reflected by CD or NC.

Results on Objaverse. Table 1 (left) reports quantitative results on the in-distribution Objaverse dataset. MeshExpand achieves the lowest CD among the evaluated methods, with performance comparable to Advancing Front and FastMesh, and substantially outperforms the learning-based baselines in normal consistency. In particular, MeshExpand achieves the highest NC and |NC|, indicating better surface orientation and detail preservation. The small gap between NC and |NC| further shows that MeshExpand produces substantially fewer normal-flipping artifacts.

Results on GSO and Trellis. To evaluate generalization beyond the training distribution, we further test MeshExpand on GSO (Downs et al., 2022) captured using 3D scanners, and Trellis (Xiang et al., 2025) generated from user prompts. These inputs present different geometry and surface characteristics from the artist-created meshes used for training. Table 1 (right) reports the quantitative results, with representative examples shown in Fig. 3. MeshExpand maintains performance comparable to its in-distribution results across these datasets. In contrast, the learning-based baselines exhibit substantial degradation. These results demonstrate the strong out-of-distribution generalization of our local mesh generation formulation.

Qualitative Results. Fig. 3 compares qualitative results across datasets. For a clean Objaverse object, Advancing Front yields irregular topology while learning-based baselines succeed (row 1). Under 0.004 noise, MeshExpand preserves thin geometry better than baselines (row 2). On a scanned GSO toy, MeshExpand is the only method to correctly recover the structure (row 3), and it faithfully reconstructs a complex generated church from Trellis (row 4). These examples align with Table 1, highlighting MeshExpand's robustness and out-of-distribution generalization. More comparisons are provided in Fig. 8 of the appendix.

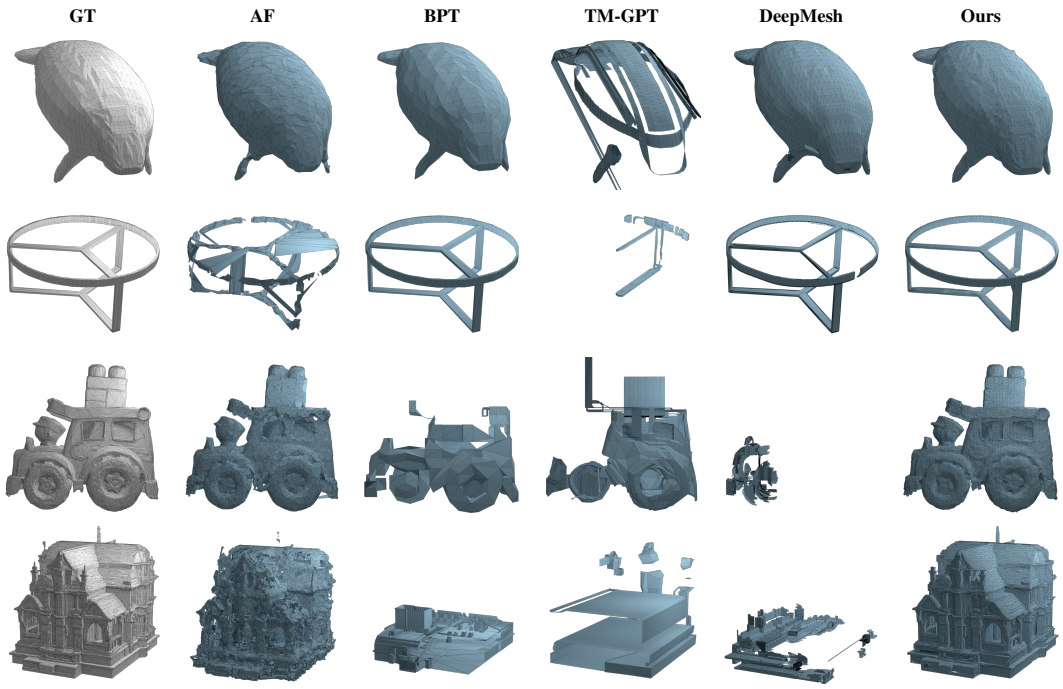

Figure 3: **Qualitative comparison** on Objaverse (rows 1–2), GSO (row 3), and Trellis (row 4), showing the strong generalization of MeshExpand.

Table 2: Number of self-intersecting faces and open boundary show geometric validation with resampling reduces defects.

Method	#intersect↓	#boundary↓
DeepMesh	19.28	358.6
w/o check	6.3	639.7
Ours	**0.41**	**19.7**

Memory Usage. Fig. 4 shows that while methods like TreeMeshGPT, BPT, and MeshAnything V2 suffer from memory growth as meshes scale, MeshExpand and DeepMesh maintain constant footprints. Notably, MeshExpand achieves this with significantly lower memory usage (within 2 GB up to 120K faces)

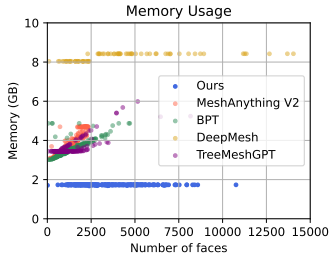

Figure 4: **Graphic memory footprint** shows that MeshExpand requires less than 2 GB of GPU memory regardless of the number of faces, making it practical on consumer GPUs.

Noise Robustness. MeshExpand also demonstrates strong robustness to input noise compared to existing methods (see Appendix E).

Ablation Study. The first row of Table 3 shows that MAE pretraining is the main factor that reduces normal-flipping artifacts. Table 2 further evaluates the effect of geometric validation and resampling on a watertight, self-intersection-free subset, for which the ground-truth meshes contain no self-intersections or open boundaries. The results show that geometric validation and resampling substantially reduce mesh defects while maintaining mesh integrity.

Order-Agnostic Generation and Mesh Completion. We first evaluate MeshExpand under different generation orders. Table 3 shows consistent performance across depth-first (DFS), breadth-first

Table 3: Once trained, MeshExpand generalizes to various generation orders without performance loss. Ablation in the first row shows that MAE pretraining enforces learning of face orientation.

Variant	CD↓ $\times 10^{-2}$	HD↓ $\times 10^{-2}$	NC↑	|NC|↑	AD↓ %
w/o MAE	1.69	18.0	0.59	0.94	6.9
Bottom-up	1.55	10.0	0.94	0.96	3.9
Top-down	1.61	10.6	0.94	0.96	4.0
BFS	1.42	14.6	0.91	0.93	6.2
DFS (Default)	1.55	12.2	0.92	0.94	5.4

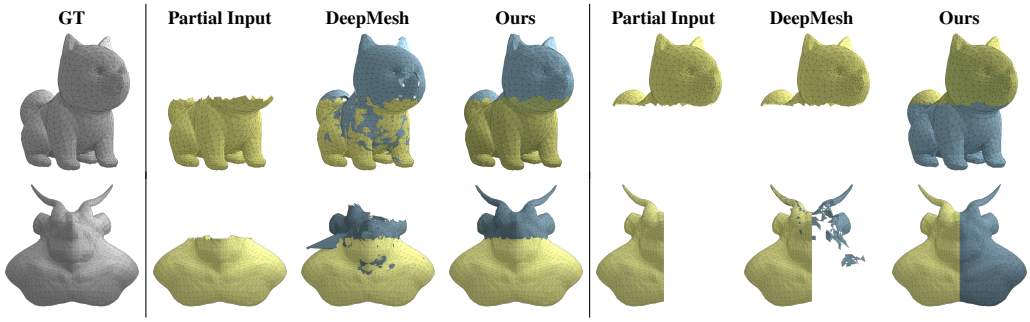

Figure 5: **Mesh completion.** DeepMesh exhibits directional bias and generates invalid geometry, whereas MeshExpand completes the mesh from any direction without topological artifacts.

Table 4: Mesh completion results under different partial inputs.

Partial Type	Method	CD↓ $\times 10^{-2}$	HD↓ $\times 10^{-2}$	NC↑	|NC|↑	AD↓ %
Bottom	DeepMesh	12.5	84.0	0.597	0.835	360
	Ours	**0.97**	**4.79**	**0.994**	**0.995**	**1.2**
Random	DeepMesh	25.5	120	0.538	0.782	150
	Ours	**1.05**	**5.36**	**0.986**	**0.991**	**1.2**

(BFS), bottom-up, and top-down traversal orders. These results confirm that MeshExpand is not tied to a particular generation sequence and can operate reliably under different mesh growth strategies.

We further evaluate mesh completion from partial inputs. Unlike order-dependent baselines, our order-agnostic formulation grows meshes from arbitrary open boundaries. As shown in Fig. 5, DeepMesh struggles with holes, self-intersections, or total failure when initialized from non-preferred directions (e.g., upper sections), whereas MeshExpand handles diverse boundaries seamlessly and outperforms it quantitatively as shown in Table 4. Fig. 1 provides more examples, including interpolation and extrapolation.

Artist Case Study. To evaluate the practical value of our completion workflow, we conducted a case study with an artist with eight years of experience. The artist first created the topology for a key region and then completed the remaining mesh using different strategies. Manual completion required 100 minutes. And MeshExpand completion followed by manual refinement required 55 minutes, which corresponds to a 45% reduction in manual effort. This single-artist case study is illustrative rather than a controlled user study.

5 Limitation

Our local generation formulation does not explicitly enforce global symmetry, which can lead to minor topological asymmetries for symmetric objects, as shown in Fig. 6. The current framework also requires at least one seed triangle for each connected component. Extending MeshExpand to fully automatic generation could therefore incorporate learned seed prediction, for example through a special start-of-surface token, or adopt classical seeding strategies from Advancing Front methods.

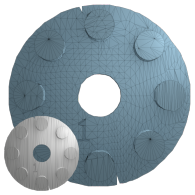

Figure 6: **Failure cases.** Limited local context causes asymmetric topology on the top of each cylinders.

6 Conclusion

We presented MeshExpand, an order-agnostic framework for artistic mesh generation from dense point clouds via learned local surface expansion. By growing meshes from seed triangles and predicting new vertices from local neighborhoods and a global shape prior, MeshExpand scales effortlessly to high-resolution meshes. Its unordered formulation further enables flexible growth strategies, mesh completion, and local editing, while density-guided training and online geometric validation ensure fine-grained resolution control and artifact-free outputs. We hope this local expansion paradigm opens new directions toward scalable, controllable, and artist-oriented mesh generation.

AI USE STATEMENT

In this work, we used generative AI tools to edit, refine, and polish the writing and improve readability. We have not used generative AI tools for other tasks that require disclosure. LLM-generated text and equations were verified and tested for correctness by the authors. We take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Implementation details, data preprocessing procedures, and the corresponding pseudocode are provided in the appendix. Code, weights, and training scripts will be made publicly available upon publication.

REFERENCES

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*, 1999.

Jean-Daniel Boissonnat and Steve Oudot. Provably good sampling and meshing of surfaces. *Graphical Models*, 67(5):405–451, 2005.

Jinnan Chen, Lingting Zhu, Zeyu Hu, Shengju Qian, Yugang Chen, Xin Wang, and Gim Hee Lee. Mar-3d: Progressive masked auto-regressor for high-resolution 3d generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11083–11092, 2025.

Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Yanru Wang, Zhibin Wang, Chi Zhang, et al. Meshxl: Neural coordinate field for generative 3d foundation models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

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*, 2024b.

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. *arXiv preprint arXiv:2408.02555*, 2024c.

David Cohen-Steiner and Frank Da. A greedy delaunay-based surface reconstruction algorithm. *The visual computer*, 20(1):4–16, 2004.

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. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *Proc. IEEE International Conf. on Robotics and Automation (ICRA)*, 2022.

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.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

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. In *International Conference on Learning Representations (ICLR)*, 2024.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing*, 2006.

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.

Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*, 2024.

Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26608–26617, 2025.

Jian Liu, Jing Xu, Song Guo, Jing Li, Jingfeng Guo, Jiaao Yu, Haohan Weng, Biwen Lei, Xianghui Yang, Zhuo Chen, et al. Mesh-rft: Enhancing mesh generation via fine-grained reinforcement fine-tuning. *arXiv preprint arXiv:2505.16761*, 2025.

Minghua Liu, Xiaoshuai Zhang, and Hao Su. Meshing point clouds with predicted intrinsic-extrinsic ratio guidance. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 68–84. Springer, 2020.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 1987.

Sylvain Petitjean and Edmond Boyer. Regular and non-regular point sets: Properties and reconstruction. *Computational Geometry*, 19(2-3):101–126, 2001.

Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J. Mitra, and Maks Ovsjanikov. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22–31, June 2021.

Nicholas Sharp and Maks Ovsjanikov. Pointtrinet: Learned triangulation of 3d point sets. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII 16*, pp. 762–778. Springer, 2020.

Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. Spacemesh: A continuous representation for learning manifold surface meshes. In *SIGGRAPH Asia 2024 conference papers*, pp. 1–11, 2024.

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 *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Gaochao Song, Zibo Zhao, Haohan Weng, Jingbo Zeng, Rongfei Jia, and Shenghua Gao. Mesh silksong: Auto-regressive mesh generation as weaving silk. *arXiv preprint arXiv:2507.02477*, 2025.

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*, 2024.

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. *ACM Transactions on Graphics (TOG)*, 45(4):1–14, 2026.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4563–4573, 2023.

Yuxuan Wang, Xuanyu Yi, Haohan Weng, Qingshan Xu, Xiaokang Wei, Xianghui Yang, Chunchao Guo, Long Chen, and Hanwang Zhang. Nautilus: Locality-aware autoencoder for scalable mesh generation. *arXiv preprint arXiv:2501.14317*, 2025.

Haohan Weng, Yikai Wang, Tong Zhang, CL Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. *arXiv preprint arXiv:2405.16890*, 2024a.

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. *arXiv preprint arXiv:2411.07025*, 2024b.

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, 2025.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

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)*, 2023.

Ruowen Zhao, Junliang Ye, Zhengyi Wang, Guangce Liu, Yiwen Chen, Yikai Wang, and Jun Zhu. Deepmesh: Auto-regressive artist-mesh creation with reinforcement learning. *arXiv preprint arXiv:2503.15265*, 2025.

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in neural information processing systems*, 36:73969–73982, 2023.



Figure 7: **More artistic meshes** generated by MeshExpand within 2 GB of GPU memory.

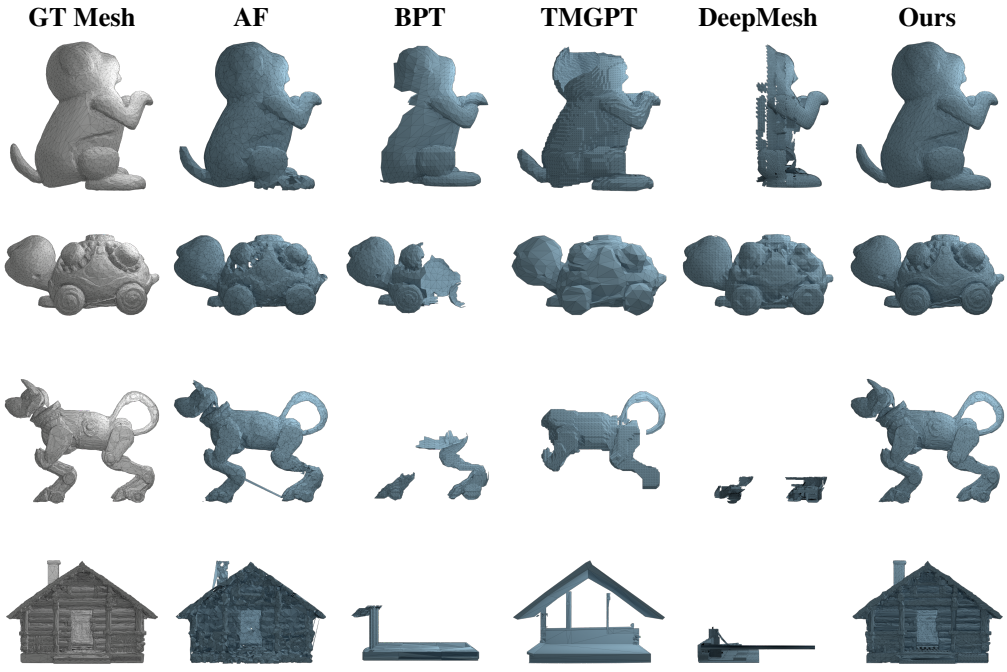

Figure 8: **More qualitative comparisons** on GSO (rows 1–2) and on Trellis (rows 3–4) demonstrate the generalizability of MeshExpand.

A Qualitative Result

More qualitative results are provided in Fig. 7. Symmetry is well preserved in most cases, and thin structures that are often missed are successfully recovered. Additional comparison results on the challenging GSO and Trellis datasets are also provided in Fig. 8. Furthermore, density control experiments are provided in Fig. 9, where we downsample the input point clouds into different levels, and MeshExpand generates meshes with corresponding face counts.

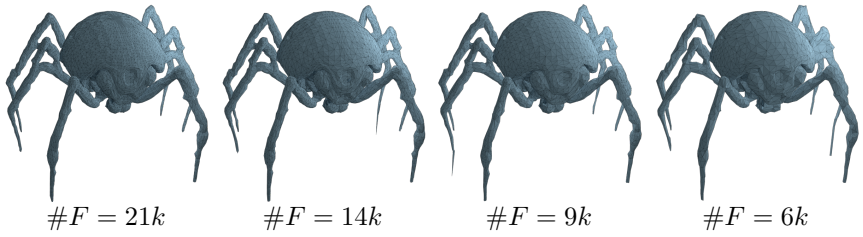

Figure 9: **Density-aware Mesh Generation.** MeshExpand produces meshes with varying face counts in response to different input point cloud densities.

B Triangle Orientation

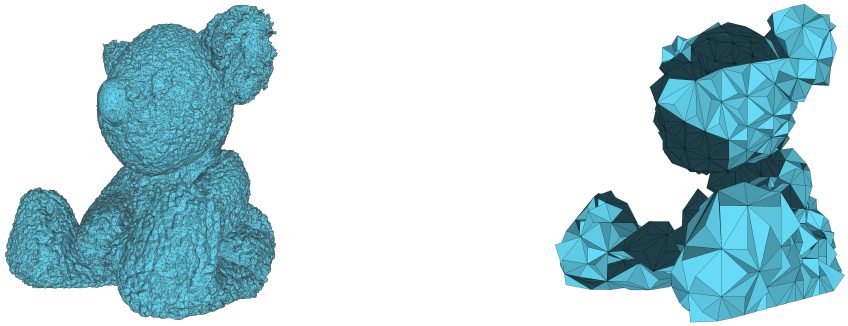

Figure 10: **Normal flipping.** While the Ground Truth shows consistent orientation, the mesh generated by MeshAnything V2 displays severe flipped normal defects. MeshLab Flat Shading is used to emphasize this phenomenon.

We observe that many baselines, as well as our MeshExpand without MAE pre-training, often generate triangles that lie on the surface but have flipped normals relative to the point-cloud normal (cf Fig.10). This issue is difficult to notice when using double-sided rendering, but becomes unacceptable when back-face culling or other shaders are applied. Flipped triangles not only cause implausible visual artifacts but also create topological defects such as Möbius loops and non-manifold structures, which hinder downstream processes like UV unwrapping. Trivial reorientation is impossible once Möbius loops form, so it is crucial to address this problem at generation. To understand its cause, we analyze a simple case within our framework. Given a directed edge lying on the surface defined by point cloud, the model can place the next vertex on either side v_r or v_l (cf Fig. 11a), forming two possible triangles f_r and f_l. By the right-hand rule of directed triangle, the normal of f_r points into the page, while the normal of f_l points out of the page. The model should choose the triangle whose normal matches the point-cloud normal. In practice, however, we find that the model often relies on unstable cues instead of the normal: the biased local bounding box (cf Fig. 11b), the locations of previous triangles (cf Fig.11c), or the existing triangles normal. While these signals are correct during training, they accumulate noise and errors at inference time, making them far less reliable than the input normals. To mitigate these failure modes, we enlarge the local bbox cover both f_{adj} and f'_{adj}; (2) use MAE pre-training to reduce dependency on previous triangles, and (3) encode triangles as undirected to avoid relying on existing normal. Together, these modifications significantly improve MeshExpand's NC over baselines.

C Implementation Details

The pseudocode is presented in Algorithm 1. We use a local window expansion factor of $\lambda_f = 1.2$ for faces, and a factor with half triangle bigger $\lambda_p = \lambda_f + 0.5 = 1.7$ for points. The box is extend

Algorithm 1: MeshExpand generation process

Input: $\mathbf{P}_{\mathrm{G}}$ // Global point cloud + normals
$\mathbf{P}$ // Input point cloud + normals
$\mathbf{F}_{\text{seed}}$ // Seed triangles
Output: $\mathbf{F}$ // Output faces
/* Initialization */
$\mathbf{z}_{\mathrm{G}} \leftarrow \mathrm{encode}(\mathbf{P}_{\mathrm{G}})$ // Encode global prior
$\mathbf{F} \leftarrow \mathbf{F}_{\text{seed}}$ // Init mesh
$\mathcal{E} \leftarrow \mathrm{open_boundary}(\mathbf{F})$ // Init boundary set
/* Iterate until no open boundaries */
while $\mathcal{E} \neq \varnothing$ **do**
  /* Select query edge */
  $e \leftarrow \mathrm{pick}(\mathcal{E})$;
  /* Skip closed edge */
  if *not* $\mathrm{is_boundary}(e)$ **then**
    continue;
  /* Find adjacent face */
  $f_{\text{adj}} \leftarrow \mathrm{adjacent_face}(e)$;
  /* Set adaptive windows */
  $\mathbf{B}_f, \mathbf{B}_p \leftarrow \mathrm{estimate_bbox}(f_{\text{adj}}, e)$;
  /* Collect local faces */
  $\mathbf{F}_{\text{local}} \leftarrow \mathrm{collect}(\mathbf{F}, \mathbf{B}_f)$;
  /* Collect local points */
  $\mathbf{P}_{\text{local}} \leftarrow \mathrm{collect}(\mathbf{P}, \mathbf{B}_p)$;
  while *True* **do**
    /* Sample opposite vertex */
    $v \sim P(v \mid e, \mathbf{F}_{\text{local}}, \mathbf{P}_{\text{local}}, \mathbf{z}_G)$;
    /* No face added */
    if $v =$ `boundary_token` **then**
      /* Next boundary */
      break;
    /* Form candidate face */
    $f' \leftarrow \mathrm{new_triangle}(v, e)$;
    /* Pass geometry checks */
    if $\mathrm{geo_test}(f')$ **then**
      /* Add face */
      $\mathbf{F} \leftarrow \mathbf{F} \cup f'$;
      /* Update boundaries */
      $\mathcal{E} \leftarrow \mathcal{E} \cup \{e_i \in \{e_1, e_2\} \mid \mathrm{is_boundary}(e_i)\}$;
      /* Next boundary */
      break;

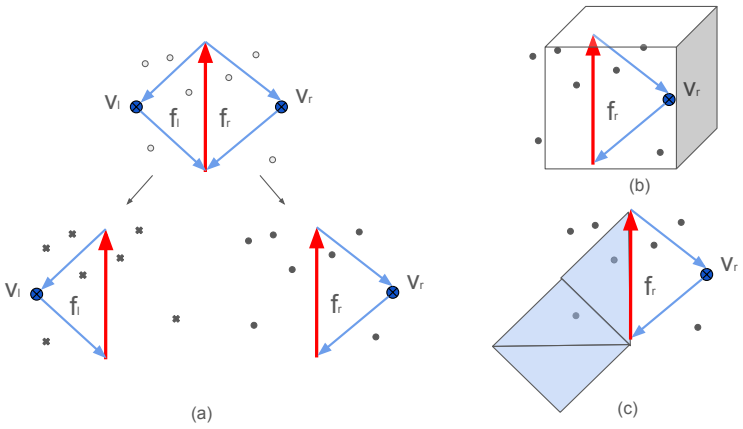

Figure 11: (a) Given a directed edge, there are two candidate triangles: f_l, when the point cloud normals facing inside, and f_r, when the point cloud normals facing outside. In practice, however, the model often ignores the point cloud normal and overfits to other cues, like the local bbox (b) or the previous triangles (c), leads to accumulated error.

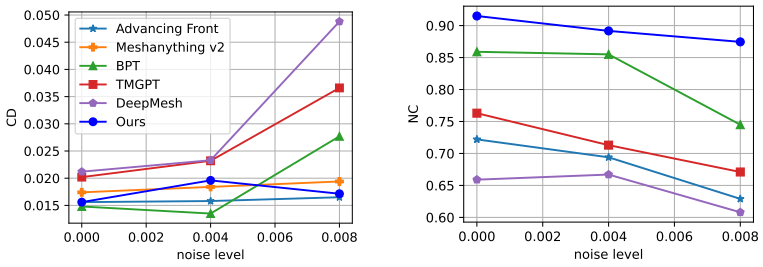

Figure 12: **Noise resistance.** CD increases notably for BPT, DeepMesh, and TreeMeshGPT under higher noise, but remains low for Advancing Front, MeshAnything V2 and MeshExpand. Only MeshExpand maintains a consistently high NC across all noise levels. (MeshAnything V2's NC is omitted as it falls below the y-axis range.)

by $\lambda_{f/p} * d$ along all axes, where d is the original box diagonal length, ensuring the final box is non-degenerate. The maximum samples are $N_{f\,\max} = 192$ and $N_{p\,\max} = N_{f\,\max} * 6 = 1152$. Under this configuration, 94% of points within the local window are included.

D Data Preprocessing and Augementation

The triangle encoder is a 4-layer Transformer encoder with a hidden size of 256. The main Transformer is a 12-layer Transformer encoder with cross-attention and a hidden size of 512. The vertex decoder is a 2-layer Transformer decoder with a hidden size of 512. All meshes are centered and normalized within $[-1, 1]$. We apply 8-bit discretization, merge identical vertices, remove degenerate triangles, split connected components, and select the components with more than 1000 faces. After filtering, we retain a total of 30,000 meshes with 1% reserved for validation and use the remainder for training. To increase data diversity, we apply the following augmentations:

- **Scaling:** Each axis is scaled independently by a factor randomly chosen from the range $[0.7, 1]$.
- **Rotation:** The mesh is rotated around the y-axis (gravity axis) by a uniformly sampled angle in $[-180, 180]$ with 70% probability.

E EXPERIMENTS ON NOISE ROBUSTNESS

We evaluate robustness to noisy inputs by adding Gaussian noise with variances of 0.004 and 0.008 to the input point positions and normals. Fig. 12 reports the resulting Chamfer Distance (CD). As the noise level increases, BPT, DeepMesh, and TreeMeshGPT exhibit substantial degradation. MeshAnything V2 remains relatively robust due to its Marching Cubes preprocessing, although this additional step may smooth geometric details and increase preprocessing cost. In contrast, MeshExpand maintains comparatively low CD across both noise levels without additional preprocessing and remains robust to perturbations in the input geometry.