QUADLINK: AUTOREGRESSIVE QUAD-DOMINANT MESH GENERATION VIA POINT-RELATION LEARNING

Anonymous authors
Paper under double-blind review

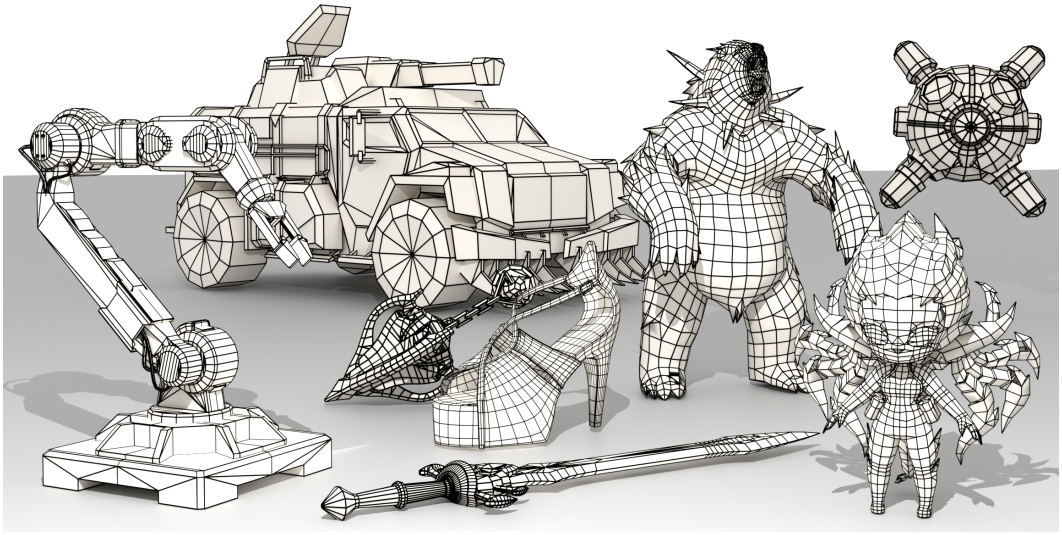

Figure 1: We propose **QuadLink**, an autoregressive framework for generating production-ready quad-dominant meshes from point clouds. The combination of a face-type-agnostic point-linking architecture and high-quality anisotropic training data curated by our Tri-to-Quad Operator endows **QuadLink** with strong generalization, enabling it to generate sparse, editable meshes with coherent edge flow across diverse 3D shapes.

ABSTRACT

The generation of production-ready quad-dominant meshes is a cornerstone of modern 3D content creation. Generating anisotropic quad-dominant meshes from point clouds is challenging, as existing methods are typically limited to producing either pure triangular meshes or pure quadrilateral meshes with isotropic densities. In this paper, we present **QuadLink**, a unified framework consisting of three stages for quad-dominant mesh generation by linking points into structured faces. QuadLink formulates polygonal mesh generation as a hybrid centroid-conditioned vertex linking model: it first predicts a unified set of anchors (vertices and face centroids), then learns centroid-conditioned links that associate vertices with face centroids, and finally assembles polygonal faces with a quad-first strategy guided by robust geometric verification strategies. This link-based formulation enables efficient generation of sparse and anisotropic quad-dominant meshes with coherent edge flow and meanwhile supporting hybrid polygonal topology. To construct training data for this model, we further introduce a *Tri-to-Quad Operator* that converts artistic triangle meshes into quad-dominant training data via global merge selection. Extensive experiments show that **QuadLink** produces production-ready quad-dominant meshes from point clouds and achieves improved geometric fidelity and topological quality compared to prior baselines. Our method natively supports hybrid polygonal topology, generalizing to arbitrary n-gon meshes without architectural changes.

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.

1 INTRODUCTION

Modern 3D generative models have made rapid progress in generating 3D geometry from text or images. However, most pipelines still prioritize surface reconstruction and typically rely on implicit or volumetric representations (e.g., SDFs (Zheng et al., 2022), voxels (Romanelis et al., 2025), or neural fields (Zhu et al., 2025)) followed by iso-surface extraction. While effective for capturing shapes, this workflow inevitably produces dense and topologically unstructured triangle meshes, leaving the crucial problem of production-ready artistic meshes to a separate remeshing stage. In practical content creation, topology is not merely a by-product of geometry: it directly determines whether an asset can be efficiently edited, animated, and integrated into modern production pipelines.

This gap motivates direct generation of editable mesh representations that better match production-ready topology rather than merely reconstructing surface geometry. Autoregressive mesh generation has recently emerged as a strong alternative by modeling meshes as discrete sequences (Hao et al., 2024; Zhao et al., 2025; Weng et al., 2025), demonstrating impressive capability in generating artistic triangle meshes. Yet, real production pipelines rarely stop at triangle-only outputs. Instead, they heavily rely on *quad-dominant* meshes as an editable intermediate representation. A quad-dominant mesh is composed predominantly of quadrilateral faces to provide a structured and editable surface layout, while retaining a small and purposeful number of triangles to function as localized topological relaxations for accommodating geometric and topological irregularities that cannot be resolved under strict quadrilateral constraints. This structure makes quad-dominant meshes not necessarily geometry-optimal, but often production-optimal which better supports practical texture workflows and geometry operations as illustrated in Fig. 3.

Despite their practical importance, generating production-ready quad-dominant meshes remains underexplored because of three challenges. **Hybrid Primitive Types.** Existing autoregressive methods (Hao et al., 2024; Weng et al., 2025; Zhao et al., 2025) serialize faces, so mixed triangles and quads require padding or type-specific tokens, increasing sequence ambiguity and complicating generation. **Anisotropic Primitive Density.** Artist-designed meshes allocate polygons non-uniformly, with large faces on simple regions and denser, directional edge flow near part boundaries and deformation axes. Such patterns are difficult to recover from dense triangulations (Lee & Schachter, 1980; Lorensen & Cline, 1998) and differ from the smooth, near-isotropic layouts favored by field-aligned remeshing methods (Dong et al., 2025; Tao et al., 2025; Bommes et al., 2009; Knöppel et al., 2013), as illustrated in Fig. 2. **Scarcity of Training Dataset.** Large-scale production-ready quad-dominant datasets remain limited, while common remeshing pipelines (Pietroni et al., 2021; Huang et al., 2018; Jakob et al., 2015; Exoside, 2019) tend to replace artistic anisotropy with geometry-driven tessellations, leaving a gap in training data.

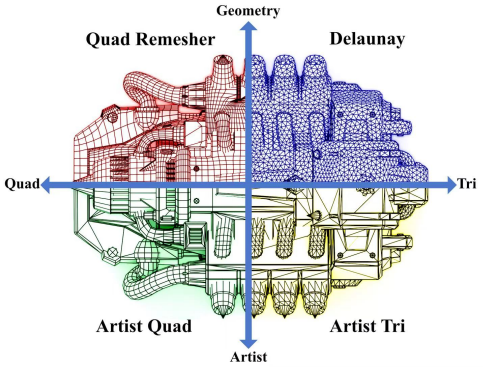

Figure 2: **Artist-designed meshes differ fundamentally from those produced by geometry processing pipelines.** We organize representative meshes along two axes: quadrilateral vs. triangular (horizontal) and artist vs. geometry-driven (vertical) within a single case for clear comparison.

We present **QuadLink**, a unified framework for quad-dominant mesh generation from point clouds. To address **hybrid primitive types**, we predict point-level anchors (vertices and face centroids) rather than serialize faces. To model **anisotropic primitive density**, we learn centroid-conditioned vertex groups in feature space, capturing long-range, directionally structured faces. To mitigate the **scarcity of training data**, we introduce a Tri-to-Quad Operator that turns artist-designed triangle meshes into anisotropic quad-dominant supervision. Together, these ideas support direct generation of editable meshes with mixed polygon types and coherent edge flow.

Our main contributions are summarized as follows:

- **QuadLink**: a unified framework consisting of three stages for quad-dominant mesh generation with a point-centric representation for ***hybrid*** polygonal layouts with substantially shorter token sequences (see Appendix C.5 for the token-length analysis) and scalability to arbitrary n-gons.

- A centroid-conditioned contrastive link modeling and robust face assembly scheme that learns vertex grouping and achieves SOTA performance for ***anisotropic*** production-ready topology.
- A SOTA tool to build high-quality quad-dominant ***dataset*** with semantic anisotropy and coherent edge flow via a *Tri-to-Quad Operator* using geometry prefiltering and global merge selection.

2 Related Works

Triangular Mesh Generation. Autoregressive mesh generation is a compelling paradigm for producing compact, artistic triangle meshes via causal discrete prediction. PolyGen (Nash et al., 2020) pioneered the use of autoregressive models for mesh generation. Afterwards, several works explore alternative autoregressive mesh representations, including neural fields, connectivity modeling, and LLM-based text formats (Shen et al., 2024; Chen et al., 2024; Wang et al., 2024), but with limited generation quality. MeshGPT (Siddiqui et al., 2024) established a great generative formulation that natively represents a mesh as a sequence of triangles, making autoregressive model the mainstream paradigm for artistic mesh generation. Most subsequent triangular mesh generation methods (Liu et al., 2025; Lionar et al., 2025; Chen et al., 2025b; Tang et al., 2024; Song et al., 2025a; He et al., 2025) build upon the autoregressive paradigm and achieve competitive performance. However, encoding each triangle requires nine coordinate tokens, causing sequence length to grow linearly with face count, which severely restricts scalability and makes these methods computationally expensive for high-resolution meshes. As a result, subsequent research has focused on improving *token compactness* to mitigate the scalability bottleneck of triangle-based autoregressive generation. Existing efforts mainly fall into several broad directions as follows.

Architectural approaches improve efficiency without changing the triangle representation, such as Meshtron (Hao et al., 2024) with hierarchical factorization and hourglass transformers, and iFlame (Wang et al., 2025), XSpecMesh (Chen et al., 2025a), and FlashMesh (Shen et al., 2025) with optimized attention or decoding strategies, but they still scale linearly with face count. Direct token compression methods reduce tokens per face via blocked or hierarchical patchification (Weng et al., 2025; Zhao et al., 2025), achieving substantial sequence shortening at the cost of enlarged vocabularies and degraded global connectivity. More radical representations depart from triangle sequences: FastMesh (Kim et al., 2025) generates vertex token sequence and vertex relations to greatly reduce token count, but only relies on quadratic adjacency prediction which results in chaotic connection. FACE (Wang et al., 2026) represents each triangle as a single face token for efficiency, but this abstraction reduces explicit control over vertex sharing and mesh connectivity.

Polygonal Mesh Generation. Directly generating quad-dominant meshes remains largely underexplored. In practice, most quad meshes are still obtained via post-processing rather than generation. A traditional line of approaches relies on *field-aligned parametrization*: it first computes a smooth cross field on the surface, then solves for a globally consistent parameterization, and extracts quadrilateral faces by tracing integer isolines or applying dedicated quad-extraction routines (e.g., Instant-Meshes (Jakob et al., 2015), Mixed-Integer Quadrangulation (Bommes et al., 2009), QuadriFlow (Huang et al., 2018), QuadWild (Pietroni et al., 2021), LibQEX (Ebke et al., 2013) and recent learning-accelerated field prediction such as NeurCross (Dong et al., 2024), CrossGen (Dong et al., 2025) and TopGen (Chen et al., 2026)). While effective for producing globally smooth and regular

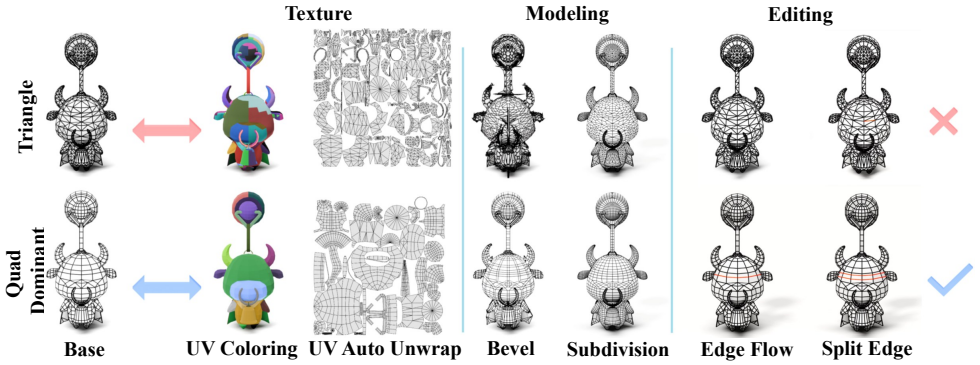

Figure 3: **Applications of Quad-Dominant Meshes.** Quad-dominant meshes enable cleaner semantic UV and auto-unwrapping, support common modeling operations like beveling and subdivision, and provide coherent edge flow for controllable edge-loop editing instead of edge-by-edge editing.

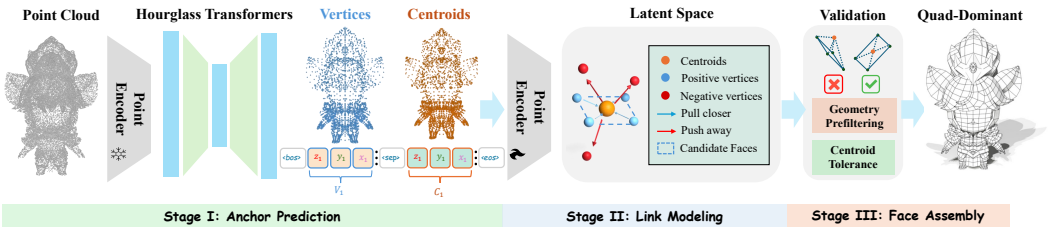

Figure 4: **Overview of QuadLink**. The pipeline consists of three stages: Stage I: Anchor Prediction, where the input point cloud is processed by a Point Cloud Encoder followed by Hourglass Transformers to generate vertex and centroid tokens. Stage II: Link Modeling, which uses contrastive learning to model the relationships between centroids and vertices. Stage III: Face Assembly, where candidate faces are progressively checked using validation criteria.

tessellations, these pipelines tend to over-optimize for surface smoothness and near-isotropic faces, which often deviates from modern anisotropic and semantically aligned artistic mesh layouts, and can be brittle on fine-grained details or complex topology. Some works (Jiang et al., 2015; Panozzo et al., 2014; Corman & Crane, 2025; Dielen et al., 2021) extend cross fields to anisotropic frame fields and integrable parameterizations by encoding direction-dependent scaling via metric or tensor formulations, enabling adaptive and feature-aligned quadrangulation. However, such anisotropy is fundamentally geometry-driven and locally defined, differing from the semantically structured, non-uniform anisotropy observed in production-ready artist-designed meshes.

These limitations motivate quad-dominant generative formulations that can directly learn from production-ready artist-designed meshes. Point2Quad (Li et al., 2025) generates quad candidates via k-NN grouping and filters them with an MLP classifier. However, it relies on post-processing heuristics, and the local k-NN formulation limits its scalability to complex geometries. QuadGPT (Liu et al., 2026) instead unifies triangles and quads by padding each face to a unified quadrilateral token length, which introduces additional padding tokens and makes face-type inference rely on sparse padding patterns, resulting in less efficient and less stable training.

In contrast, QuadLink introduces a generative formulation for quad-dominant meshes that aligns with modern production pipelines. Unlike prior autoregressive methods that rely on triangle-only representations or padding-based unification, QuadLink directly supports structured quad-dominant topology with shorter token sequences and less information loss (see Appendix C.5 for the token-length analysis). Together with a curated training dataset constructed via a self-developed Tri-to-Quad operator, it provides a scalable and efficient solution for quad-dominant mesh generation.

3 QuadLink

Given a surface point cloud of a 3D shape, QuadLink generates a corresponding quad-dominant mesh with high fidelity and anisotropic density. Instead of directly predicting face sequences, QuadLink adopts a centroid-conditioned vertex linking model that first generates anchors (vertices and face centroids) and then infers their connectivity to form polygonal faces.

An overview of our pipeline is shown in Fig. 4. Specifically, we first predict a compact set of anchors (Sec. 3.1), and then learn centroid-conditioned vertex links to capture face-level semantics (Sec. 3.2), and finally assemble these links into polygonal faces through a deterministic *quad-first, triangle-next* assembly strategy under both the features and geometric constraints (Sec. 3.3).

3.1 Stage I: Anchor Prediction

Given the input point cloud, the target of Stage I is to generate the vertices and face centroids, called anchors, of the underlying quad-dominant mesh. We follow an autoregressive architecture to generate these anchors as a token sequence. In the following, we first introduce how to convert these anchor points to a token sequence.

Token Representation. To represent anchor points as a token sequence, we adopt the following:

$$\mathcal{M} = \Big\{ \underbrace{z_1, y_1, x_1}_{\mathcal{T}(\mathbf{v}^{(1)})}, \ldots, \tau_{\text{sep}}, \underbrace{z'_1, y'_1, x'_1}_{\mathcal{T}(\mathbf{c}^{(1)})}, \ldots \Big\},$$

where each anchor point (z, y, x) is first quantized into three discrete coordinate tokens and ordered in $z - y - x$. $\mathcal{T}$ denotes the tokenization function, and τ_{sep} is a special discrete separating token to separate centroids from vertices.

Axis-aware Coordinate Tokenization. For the coordinate token, we construct a vocabulary of size 3072. To distinguish tokens from different axes, we assign disjoint index ranges to each coordinate: x takes values in $[0, 1024)$, y in $[1024, 2048)$, and z in $[2048, 3072)$. This separation scheme allows the generative model to explicitly identify the axis of a coordinate token based on its token ID. In contrast, a vanilla scheme sharing the same vocabulary $[0, 1024)$ for all axes forces the model to rely solely on the token's position in the sequence to determine the axis identity, which can lead to ambiguity. Further discussion of tokenization is provided in Appendix Sec. B.5 and B.6.

Model Architecture & Training. We generate the anchor point token sequence in an autoregressive manner using the cross-entropy loss. The input point cloud is encoded with an Adaptive Michelangelo Point Cloud Encoder and the generation process is conditioned on the encoded features with cross attentions. Our generation network follows Meshtron (Hao et al., 2024) to use a hierarchical transformer structure. More architecture details are provided in Appendix Sec. B.1 and B.2.

3.2 Stage II: Link Modeling

After getting the anchor points from Anchor Prediction *(Stage I)*, we proceed to link vertices into coherent polygons. To achieve this, our Link Modeling *(Stage II)* aims to learn a new feature space, which is a discriminative feature space which pulls valid (centroid, vertex) pairs closer together while pushing apart unpaired ones. This contrastive formulation encourages vertices belonging to the same face to group around their corresponding centroid beyond Euclidean space. Consequently, each vertex can be reliably assigned to its face by measuring the proximity in the learned feature space, which realizes the anisotropy of production-ready meshes. To achieve this, we finetune a Michelangelo point cloud encoder (Zhao et al., 2023), optimized by using a contrastive-learning objective based on triplet margin loss (Schroff et al., 2015), to learn such a feature space:

$$\mathcal{L}_{\text{triplet}} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{|\mathcal{N}^{(i)}|} \sum_{N \in \mathcal{N}^{(i)}} \Big[\|f(A^{(i)}) - f(P^{(i)})\|_2^2 - \|f(A^{(i)}) - f(N)\|_2^2 + m \Big]_+$$

where $[x]_+ = \max(0, x)$, $A^{(i)}$ denotes the anchor (the centroid token of face i), $P^{(i)}$ is a positive vertex belonging to this face, and each $N \in \mathcal{N}^{(i)}$ denotes a negative vertex sampled from other faces. The margin m enforces a separation between positive and negative pairs in the feature space. This formulation encourages the model to generate consistent features for vertices corresponding to the same centroids, which provides essential clues for us to connect these vertices.

Hard Negative Mining with Adaptive Top-K Selection. During training, we adopt a hard negative mining strategy with adaptive Top-K selection. Directly optimizing Eq. 1 over all negatives is often dominated by a large number of easy (near-zero) negatives, leading to weak and inefficient gradients. For each anchor–positive pair $(A^{(i)}, P^{(i)})$, we rank candidate negatives by their margin violation and select the Top-K hardest ones to compute the triplet loss. Inspired by curriculum learning, we employ a progressive hard-mining schedule that gradually increases K during training to further stabilize optimization. Further strategy details are provided in Appendix Section B.3.

3.3 Stage III: Face Assembly

After generating the anchor points in Anchor Prediction *(Stage I)* and the corresponding features in Link Modeling *(Stage II)*, our Face Assembly *(Stage III)* aims to connect these generated vertex points based on both the features and the geometric constraints. The key idea of Face Assembly *(Stage III)* is to first search the Top-K nearest vertices for each centroid in the latent feature space, and then validate whether the polygons formulated from these vertices conform to the geometric constraints. We summarize the complete procedure with pseudocode in Appendix Section B.4. Additional geometric verification criteria and thresholds are provided in Appendix Section A.3.

4 Quad-Dominant Data Curation

High-quality quad-dominant mesh datasets with production-ready topology remain extremely scarce. However, a vast amount of artist-designed triangle meshes already exists. These triangle meshes are typically generated by splitting the quadrilateral faces of production-ready quad-dominant models during the asset export or runtime stage. Therefore, if we can develop an effective reverse algorithm capable of merging adjacent triangles back into quadrilaterals while re-

covering artist-designed topology, we can unlock a large number of high-quality training data for quad-dominant mesh generation.

Appendix Sec. C.3 further verifies that local greedy edge merging can be suboptimal. Blossom-Quad formulates triangle pairing as a global minimum-cost perfect matching under geometric criteria, targeting pure quad meshes (Remacle et al., 2012).

Our goal is instead to recover semantically structured, anisotropic quad-dominant meshes with coherent edge flow. We therefore formulate the global merge selection below.

4.1 Global Merging Problem Formulation

Given a triangular mesh with face set $\mathcal{F}$ and candidate internal edges E_{int}, we assign $z_e \in \{0, 1\}$ to each edge e, where $z_e = 1$ selects the merge of its two incident triangles into q_e. Each candidate merge receives a weight w_e reflecting quad regularity and alignment with local surface flow. The score definitions and parameter settings are given in Appendix Section A.3.

We then solve a global maximum-weight matching problem:

$$\max_{z} \; w^\top z \quad \text{s.t.} \quad Az \leq \mathbf{1}, \;\; z_e \in \{0, 1\},$$

where A is the face–edge incidence matrix ensuring that each triangle participates in at most one merge. More details and parameters are provided in Appendix Section A.3.

Unlike perfect matching, our relaxed constraint $Az \leq \mathbf{1}$ allows some triangles to remain unmerged, producing quad-dominant meshes. Figure 5 compares our greedy and global variants with Blossom-Quad (Geuzaine & Remacle, 2009).

4.2 Solving with Geometry Prefiltering

Before matching, we discard candidates whose implied quads fail geometric constraints.

We use the same geometric checks as in Face Assembly (Stage III; Sec. 3.3) before solving the reduced maximum-weight matching, while enforcing deterministic normal consistency. Details and thresholds are provided in Appendix Section A.3.

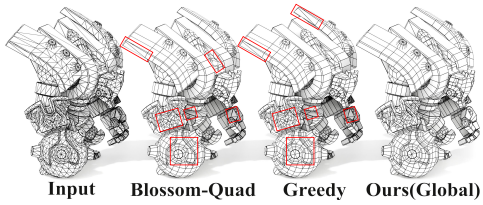

Figure 5: **Qualitative comparison with merge-edge methods.** We compare Blossom-Quad in Gmsh (Geuzaine & Remacle, 2009) with both greedy and global variants of our *Tri-to-Quad Operator*. Our global formulation yields higher-quality quad-dominant meshes for data curation.

5 Experiments

5.1 Dataset

Both QuadLink Anchor Generation *(Stage I)* and Link Modeling *(Stage II)* are trained on a curated dataset of 400k quad-dominant models, sourced from filtered internal assets licensed from 3D content providers. Each sample is first sanitized to remove mesh degeneracies, and then converted into a quad-dominant representation via our self-developed *Tri-to-Quad Operator*. The resulting dataset spans approximately 300 mesh categories, with each model containing up to 10k points and being normalized and discretized under a 1024-level coordinate resolution.

5.2 Experimental Protocols

Training and inference details are provided in Appendix Sections A.1 and A.2.

Baselines. We consider two types of baseline methods. The first category consists of recent triangle-based autoregressive models, including BPT (Weng et al., 2025), DeepMesh (Zhao et al., 2025), MeshAnythingV2 (Chen et al., 2025b), FastMesh (Kim et al., 2025), TreeMeshGPT (Lionar et al., 2025), MeshMosaic (Xu et al., 2025). Since these approaches are fundamentally designed to output triangular artist meshes, we apply the *same Tri-to-Quad* operator as a postprocessing step to obtain quad-dominant meshes, enabling a fair and consistent evaluation. The second category covers classical field-aligned quad remeshing pipelines that directly obtain quad meshes, for which we include three well-established methods Instant-Meshes (Jakob et al., 2015), QuadriFlow (Huang et al., 2018) and QuadWild (Pietroni et al., 2021). Note that QuadGPT (Liu et al., 2026) has no available code, making the comparison infeasible.

Metrics & Evaluation Dataset. We evaluate geometric fidelity using Chamfer Distance (CD), Hausdorff Distance (HD), and volumetric Intersection-over-Union (IoU) between generated meshes

and ground-truth meshes. To measure polygonal topology, we report the Quadrilateral Ratio (QR), defined as the proportion of quadrilateral faces among all faces. For comparisons with traditional remeshing methods, QR is often saturated since these methods typically produce pure-quad meshes. We therefore additionally report Edge Flow Ratio (EFR), which measures the proportion of edges following consistent edge flow patterns and provides a more discriminative indicator of structured quad layout quality. Details of **EFR** and discussions about **Singularities, Watertightness and Manifoldness** are provided in Appendix Sections B.7 and D.1.

For evaluation, we construct a test set of 100 models, excluding meshes used for training, from both public Objaverse (Deitke et al., 2023) and artist-created meshes to assess robustness across diverse shapes. We further conduct a user study with 26 professional 3D creators. Each participant evaluates 8 sampled models across nine methods under four criteria: neatness, artistry, shape fidelity and topology quality. For each criterion, the top three methods receive 3, 2, and 1 points, respectively, while the remaining receive 0 points.

5.3 COMPARISON

Methods	**Metrics (100 cases)**					**User Study (8 cases)**			
	CD↓	**HD↓**	**IoU↑**	**QR**	**EFR↑**	**Neatness↑**	**Artistry↑**	**Shape↑**	**Topology↑**
Mesh generation methods									
BPT	0.052	0.138	0.922	68.0%	0.328	0.772	0.737	0.330	0.578
DeepMesh	0.076	0.211	0.886	77.3%	0.381	0.808	<u>1.021</u>	<u>0.882</u>	<u>1.097</u>
FastMesh	0.036	0.087	0.954	5.4%	0.244	0.238	0.516	0.558	0.322
TreeMeshGPT	0.153	0.372	0.696	85.9%	0.201	0.080	0.063	0.089	0.065
MeshMosaic	0.062	0.233	0.924	47.2%	0.292	0.230	0.492	0.420	0.386
Traditional quad remeshing methods									
Instant-Meshes	0.038	0.120	0.905	100%	0.403	0.313	0.206	0.259	0.191
QuadriFlow	0.077	0.305	0.695	100%	0.289	0.108	0.087	0.137	0.186
QuadWild	<u>0.022</u>	<u>0.086</u>	<u>0.978</u>	100%	<u>0.547</u>	<u>1.136</u>	0.486	0.832	0.703
Ours	**0.021**	**0.056**	**0.995**	79.4%	**0.714**	**2.315**	**2.392**	**2.493**	**2.472**

Table 1: **Quantitative comparison with mesh generation and traditional quad remeshing methods.** The best scores are emphasized in **bold**, and the second are highlighted with <u>underline</u>.

Comparison with Mesh Generation Methods. As shown in Table 1, QuadLink surpasses all other baselines across objective geometric metrics and subjective user study scores. Qualitative comparisons in Fig. 7 further show that triangle-based generative models are difficult to convert into production-ready quad-dominant meshes via geometric postprocessing. This is because their outputs are produced from discrete autoregressive coordinate sequences, where discretization errors and probabilistic decoding noise can perturb vertices that should lie on planar patches or semantically aligned edge flows. Such local inconsistencies violate the assumptions of the

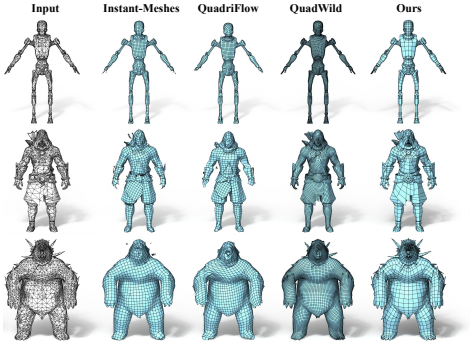

Figure 6: **Qualitative comparison with field-guided remeshing methods.** They tend to produce near-isotropic layouts and are brittle on fine-grained details or complex topology.

Comparison with Traditional Quad Remeshing Methods. As shown in Table 1, QuadLink outperforms traditional quad remeshing baselines across objective geometric metrics and subjective user study scores. Fig. 6 further shows that field-aligned remeshing tends to produce geometry-driven, near-uniform tessellations, while QuadLink better captures the semantically structured, non-uniform anisotropy existed in production-ready artist meshes. Appendix Sec. C.2 (*Field-aligned Remeshing Methods*) further shows how conventional quad-quality criteria can lead to unnecessarily dense faces and distort the target shape. Sec. C.3 (*Software-based Remeshing Baselines*) further verifies that local decisions can lead to suboptimal merge and degraded quad quality.

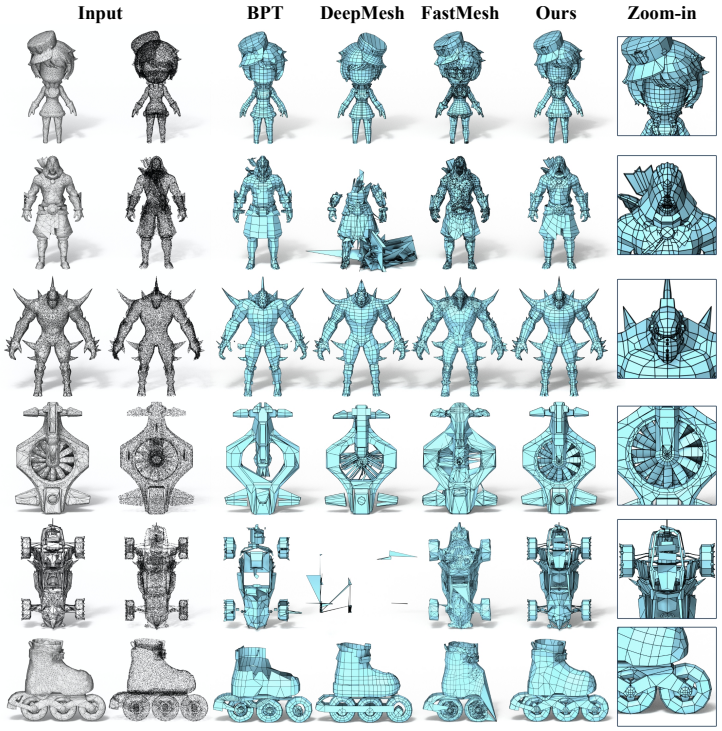

Figure 7: **Qualitative comparison with triangle-based generation methods postprocessed by our *Tri-to-Quad Operator*.**

rule-based *Tri-to-Quad operator*, making it difficult for a uniform geometric operator to robustly recover clean quad-dominant meshes. In contrast, QuadLink natively learns quad-dominant structures through centroid-conditioned vertex grouping, avoiding the fragility of triangle-first generation followed by postprocessing. More qualitative results are provided in Appendix Sec. C.1.

5.4 Ablation Studies

Ablation on our method. We ablate three key designs of our method: centroid guidance, Geometry Prefiltering, and Retrieval Space. As shown in Table 2, we compare Euclidean and latent-space retrieval, together with two validation modules: Geometry Prefiltering and Centroid Tolerance.

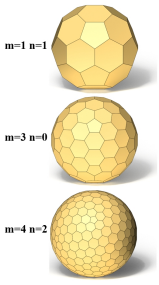

Figure 8: **Qualitative results on polygon generation with our method.**

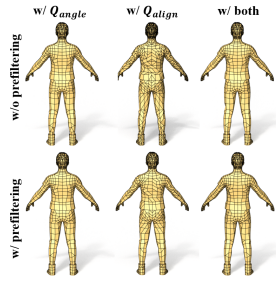

Figure 9: **Qualitative ablation on *Tri-to-Quad Operator*.**

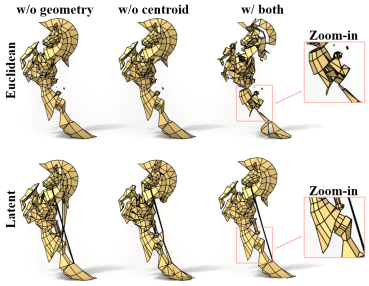

Figure 10: **Qualitative ablation on Face Assembly *(Stage III)* under different Geometric Verifications and Retrieval Spaces.**

(I) Centroid Guidance. Centroids provide explicit face-level anchors for grouping vertices into polygonal faces. Without centroid guidance, the model can only infer face grouping from pairwise vertex relations (Kim et al., 2025), making it much harder to recover stable and complicated topology, especially for anisotropic faces with long-range vertex connections observed in quad-dominant meshes.

(II) Geometry Prefiltering. Geometry Prefiltering removes degenerate or topologically unstable candidates before final assembly. Although relaxing validation can increase reconstruction rate and reduce runtime, it clearly degrades the quality of assembled faces.

Retrieval Space	Condition	recon_sec↓	recon_rate↑	CD↓	HD↓
Euclidean	w/o geometry	280.40	0.9907	0.0366	0.0746
	w/o centroid	<u>176.63</u>	0.9660	0.0392	0.0971
	w both	966.49	0.9549	0.0216	<u>0.0628</u>
Latent	w/o geometry	258.02	**0.9995**	<u>0.0211</u>	0.0658
	w/o centroid	**163.45**	0.9667	0.0314	0.1817
	w both (ours)	602.64	<u>0.9971</u>	**0.0109**	**0.0501**

Table 2: **Quantitative ablation on Stage III face assembly under different Geometric Verifications and Retrieval Spaces.** The best scores are emphasized in **bold**, while the second are highlighted with <u>underline</u>.

(III) Retrieval Space. Latent-space retrieval is crucial for anisotropic structures. As shown in Fig. 10, Euclidean retrieval fails on the seahorse reins, where nearby vertices are dominated by the dense body rather than the true incident vertices along the thin structure which requires aggressively expanding the search pool to eventually enumerate the correct face. In contrast, the learned latent space pulls incident vertices closer to their face centroid, producing compact and reliable candidate shortlists for long-range anisotropic faces.

In practice, candidate retrieval and verification are performed independently for each centroid, allowing the assembly process to be substantially accelerated through parallelized optimization.

Ablation on Tri-to-Quad Operator. With Geometry Prefiltering and both merging-quality terms enabled, QuadLink achieves the highest OEP and EFC scores (Table 3). Metric definitions and detailed component analysis are provided in Appendix Sec. A.3. Qualitative examples are shown in Fig. 9.

Metric	w/o Geometry Prefiltering			w/ Geometry Prefiltering		
	w_Q_{angle}	w_Q_{align}	w_both	w_Q_{angle}	w_Q_{align}	w_both
OEP ↑	0.9322	0.8248	0.9317	<u>0.9543</u>	0.9439	**0.9546**
EFC ↑	0.8641	0.7015	0.8642	<u>0.9005</u>	0.8763	**0.9014**

Table 3: **Quantitative ablation on *Tri-to-Quad Operator*'s components.** The best scores are emphasized in **bold**, while the second with <u>underline</u>.

6 Application

6.1 Production Pipeline

Quad-dominant meshes better support practical texture workflows and geometry operations than artistic triangle meshes, like UV textures, modeling and edge-flow editing, as illustrated in Fig. 3.

6.2 Polygonal Mesh Generation

We validate QuadLink's ability to generate n-gon meshes using Goldberg polyhedra, as shown in Fig. 8. Further details are provided in Appendix Section C.4.

7 Conclusion

We presented **QuadLink**, a unified framework consisting of three stages for natively generating production-ready quad-dominant meshes. By reformulating mesh generation as a compact and centroid-conditioned vertex linking autoregressive model along with face assembly under robust geometric verification, QuadLink enables scalable polygonal asset generation without fragile post-processing. To support learning at scale, we introduced a robust *Tri-to-Quad Operator* that provides high-quality quad-dominant supervision with structured anisotropy and coherent edge flow. Extensive experiments show that QuadLink achieves SOTA performance in all objective and subjective metrics, meanwhile making polygonal topology scalable and generating assets which are prepared for modern production pipeline. Limitations are discussed in Appendix Section D.3.

AI USE STATEMENT

We used generative AI tools to check grammar and improve the clarity of the manuscript. The research ideas originated from the authors, who also conducted the experiments and wrote the manuscript. We take responsibility for the final content of this work, including text, claims and experiment results.

REFERENCES

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48, 2009.

David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM transactions on graphics (TOG)*, 28(3):1–10, 2009.

Dian Chen, Yansong Qu, Xinyang Li, Ming Li, and Shengchuan Zhang. Xspecmesh: Quality-preserving auto-regressive mesh generation acceleration via multi-head speculative decoding. *arXiv preprint arXiv:2507.23777*, 2025a.

Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Zhibin Wang, Jingyi Yu, Gang Yu, et al. Meshxl: Neural coordinate field for generative 3d foundation models. *Advances in Neural Information Processing Systems*, 37:97141–97166, 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.

Yuguang Chen, Xinhai Liu, Xiangyu Zhu, Yiling Zhu, Zhuo Chen, Dongyu Zhang, and Chunchao Guo. Topgen: Learning structural layouts and cross-fields for quadrilateral mesh generation. *arXiv preprint arXiv:2603.10606*, 2026.

Etienne Corman and Keenan Crane. Rectangular surface parameterization. *ACM Transactions on Graphics (TOG)*, 44(4):1–21, 2025.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13142–13153, 2023.

Alexander Dielen, Isaak Lim, Max Lyon, and Leif Kobbelt. Learning direction fields for quad mesh generation. In *Computer Graphics Forum*, volume 40, pp. 181–191. Wiley Online Library, 2021.

Qiujie Dong, Huibiao Wen, Rui Xu, Xiaokang Yu, Jiaran Zhou, Shuangmin Chen, Shiqing Xin, Changhe Tu, and Wenping Wang. Neurcross: A self-supervised neural approach for representing cross fields in quad mesh generation. *arXiv e-prints*, pp. arXiv–2405, 2024.

Qiujie Dong, Jiepeng Wang, Rui Xu, Cheng Lin, Yuan Liu, Shiqing Xin, Zichun Zhong, Xin Li, Changhe Tu, Taku Komura, et al. Crossgen: Learning and generating cross fields for quad meshing. *ACM Transactions on Graphics (TOG)*, 44(6):1–15, 2025.

Hans-Christian Ebke, David Bommes, Marcel Campen, and Leif Kobbelt. Qex: Robust quad mesh extraction. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

Exoside. Quad remesher. `https://exoside.com/`, 2019.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

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.

George Hart. Goldberg polyhedra. In *Shaping space: Exploring polyhedra in nature, art, and the geometrical imagination*, pp. 125–138. Springer, 2012.

Yuze He, Yanning Zhou, Wang Zhao, Jingwen Ye, Yushi Bai, Kaiwen Xiao, Yong-Jin Liu, Zhongqian Sun, and Wei Yang. Charm: Control-point-based 3d anime hairstyle auto-regressive modeling. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas J Guibas. Quadriflow: A scalable and robust method for quadrangulation. In *Computer Graphics Forum*, volume 37, pp. 147–160. Wiley Online Library, 2018.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. Instant field-aligned meshes. *ACM transactions on graphics (TOG)*, 34(6):1–15, 2015.

Tengfei Jiang, Xianzhong Fang, Jin Huang, Hujun Bao, Yiying Tong, and Mathieu Desbrun. Frame field generation through metric customization. *ACM Transactions on Graphics (TOG)*, 34(4): 1–11, 2015.

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.

Felix Knöppel, Keenan Crane, Ulrich Pinkall, and Peter Schröder. Globally optimal direction fields. *ACM Transactions on Graphics (ToG)*, 32(4):1–10, 2013.

Ksami. Qremeshify: A blender extension for quad remeshing. `https://github.com/ksami/QRemeshify`, 2024.

Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

Zezeng Li, Zhihui Qi, Weimin Wang, Ziliang Wang, Junyi Duan, and Na Lei. Point2quad: Generating quad meshes from point clouds via face prediction. *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

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.

Jian Liu, Chunshi Wang, Song Guo, Haohan Weng, Zhen Zhou, Zhiqi Li, Jiaao Yu, Yiling Zhu, Jing Xu, Biwen Lei, et al. Quadgpt: Native quadrilateral mesh generation with autoregressive models. In *International Conference on Learning Representations*, volume 2026, pp. 22114–22139, 2026.

Yuanpeng Liu, Ting-Uei Lee, Anooshe Rezaee Javan, and Yi Min Xie. Extending goldberg's method to parametrize and control the geometry of goldberg polyhedra. *Royal Society Open Science*, 9 (8), 2022.

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

Alessandro Muntoni and Paolo Cignoni. PyMeshLab, January 2021.

Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pp. 7220–7229. PMLR, 2020.

Daniele Panozzo, Enrico Puppo, Marco Tarini, and Olga Sorkine-Hornung. Frame fields: Anisotropic and non-orthogonal cross fields additional material. *Proceedings of the ACM TRANSACTIONS ON GRAPHICS (PROCEEDINGS OF ACM SIGGRAPH*, 3, 2014.

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, Marco Tarini, et al. Reliable feature-line driven quad-remeshing. *ACM Transactions on Graphics*, 40(4):1–17, 2021.

J-F Remacle, Jonathan Lambrechts, Bruno Seny, Emilie Marchandise, Amaury Johnen, and C Geuzainet. Blossom-quad: A non-uniform quadrilateral mesh generator using a minimum-cost perfect-matching algorithm. *International journal for numerical methods in engineering*, 89 (9):1102–1119, 2012.

Ioannis Romanelis, Vlassis Fotis, Athanasios Kalogeras, Christos Alexakos, Adrian Munteanu, and Konstantinos Moustakas. Efficient and scalable point cloud generation with sparse point-voxel diffusion models. *IEEE Transactions on Neural Networks and Learning Systems*, 2025.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

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.

Tingrui Shen, Yiheng Zhang, Chen Tang, Chuan Ping, Zixing Zhao, Le Wan, Yuwang Wang, Ronggang Wang, and Shengfeng He. Flashmesh: Faster and better autoregressive mesh synthesis via structured speculation. *arXiv preprint arXiv:2511.15618*, 2025.

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.

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*, 2025a.

Kaiyu Song, Hanjiang Lai, Yaqing Zhang, Chuangjian Cai, Yan Pan Kun Yue, and Jian Yin. Topology sculptor, shape refiner: Discrete diffusion model for high-fidelity 3d meshes generation. *arXiv preprint arXiv:2510.21264*, 2025b.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

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.

Jiong Tao, Yong-Liang Yang, and Bailin Deng. Learning conjugate direction fields for planar quadrilateral mesh generation. *arXiv preprint arXiv:2511.11865*, 2025.

Hanxiao Wang, Biao Zhang, Weize Quan, Dong-Ming Yan, and Peter Wonka. iflame: Interleaving full and linear attention for efficient mesh generation. *arXiv preprint arXiv:2503.16653*, 2025.

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: A face-based autoregressive representation for high-fidelity and efficient mesh generation. *arXiv preprint arXiv:2603.01515*, 2026.

Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. Llama-mesh: Unifying 3d mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024.

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.

Rui Xu, Tianyang Xue, Qiujie Dong, Le Wan, Zhe Zhu, Peng Li, Zhiyang Dou, Cheng Lin, Shiqing Xin, Yuan Liu, et al. Meshmosaic: Scaling artist mesh generation via local-to-global assembly. *arXiv preprint arXiv:2509.19995*, 2025.

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, 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.

Xinyang Zheng, Yang Liu, Pengshuai Wang, and Xin Tong. Sdf-stylegan: implicit sdf-based stylegan for 3d shape generation. In *Computer Graphics Forum*, volume 41, pp. 52–63. Wiley Online Library, 2022.

Chenliang Zhou, Fangcheng Zhong, Weihao Xia, Albert Miao, Canberk Baykal, and Cengiz Oztireli. Quartet of diffusions: Structure-aware point cloud generation through part and symmetry guidance. *arXiv preprint arXiv:2601.20425*, 2026.

Xiaoqiang Zhu, Yanhua Zhao, and Lihua You. Neuronal mesh reconstruction from image stacks using implicit neural representations. *Mathematics*, 13(8), 2025.

APPENDIX

A IMPLEMENTATION DETAILS

A.1 TRAINING.

For *stage I Anchor Prediction*, we first freeze our adaptive Michelangelo (Zhao et al., 2023) VAE as the point cloud encoder, which encodes 49,152 surface points into 4,096 latent tokens of dimension 768. The complete VAE pipeline is applied to extract rich geometric features, which serves as cross-attention context with the context interval as 1 for the mesh decoder. The decoder is a 1B-parameter autoregressive Transformer built with a two-stage hourglass architecture (depths of 8–16–8 layers), using 1,536 hidden dimensions, 16 attention heads, and a 4,096-dimension feed-forward network. To efficiently process long sequences, the model applies a linear downsampling layer with a shortening factor of 3, and adopts Rotary Position Embeddings (RoPE) with $\theta = 1\text{e}6$ for enhanced positional encoding. We train our mesh generation model using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, a base learning rate of $1\text{e}{-4}$, and a weight decay of $1\text{e}{-4}$. We adopt a linear warmup followed by a cyclic cosine annealing schedule to facilitate stable and efficient convergence.

For *stage II Link Modeling*, we finetune a pretrained Michelangelo VAE encoder with triplet margin loss using the AdamW optimizer with a base learning rate of $1\text{e}{-5}$ and a weight decay of $1\text{e}{-4}$. We also employ a linear warmup followed by a cyclic cosine annealing schedule. The margin m of triplet margin loss is also scheduled using a cosine warmup, starting from 0.2 and gradually increasing to 0.3. We adopt hard negative mining with a dynamic k value that transitions from 20 to 50 as the epoch grows, combined with Farthest Point Sampling (FPS) sampling of 3000 negative candidates per face to balance training efficiency and memory consumption.

A.2 INFERENCE.

For Anchor Prediction *(Stage I)*, we use temperature $T = 0.5$ with top-k = 20 and top-p = 0.95 for sampling to balance output diversity and stability.

For Link Modeling *(Stage II)*, we assemble faces using the Progressive Candidate Face Selection (PCFS) strategy with a maximum pool size of $m_{max} = 20$ under the latent-space ranking, and adopt a Hard-Negative Mining Schedule with $k_{start} = 20$ and $k_{end} = 50$.

For Face Assembly *(Stage III)*, we use the geometry prefiltering and centroid tolerance checks described in Appendix Section A.3.

A.3 DATA CURATION.

For Data Curation, we convert artist-created triangle meshes into quad-dominant meshes via our self-developed *Tri-to-Quad Operator* with a three-phase pipeline: geometry prefiltering, global merging and normal consistency. The geometric prefilters below are also used for candidate quadrilaterals during Face Assembly *(Stage III)*; centroid consistency is evaluated in Euclidean coordinates for both quadrilateral and triangle candidates.

Global Merge Scoring and Selection. For each internal edge e shared by two triangles, let q_e be the resulting candidate quad, with interior angles $\{\theta_i\}_{i=1}^4$. We denote by d_e the unit direction of e and by f_e the estimated local principal-curvature direction at its midpoint, computed by one-ring PCA on the tangent plane. We combine quad regularity and edge-flow alignment as

$$\begin{aligned} Q_{\text{angle}}(q_e) &= \frac{1}{360^\circ} \sum_{i=1}^{4} \max\big(0, 90^\circ - |\theta_i - 90^\circ|\big), \\ Q_{\text{align}}(e) &= \sqrt{1 - (d_e \cdot f_e)^2}, \\ w_e &= \alpha_1 Q_{\text{angle}}(q_e) + \alpha_2 Q_{\text{align}}(e). \end{aligned} \tag{1}$$

The angle score favors near-right-angle quads. The alignment score favors removing candidate edges orthogonal to the local principal direction, so the retained edges better follow the surface flow. Both scores lie in $[0, 1]$; we use $\alpha_1 = 0.8$ and $\alpha_2 = 0.2$ in all experiments.

With a binary variable z_e indicating whether candidate edge e is merged, we select merges by solving

$$\max_{z} \; w^\top z \quad \text{s.t.} \quad Az \le \mathbf{1}, \qquad z_e \in \{0, 1\}, \quad e \in E_{\text{int}}, \tag{2}$$

where A is the face–edge incidence matrix, so each triangle can participate in at most one merge. Unlike perfect matching ($Az = \mathbf{1}$), this relaxed constraint leaves some triangles unmerged and

therefore supports quad-dominant output. Perfect matching instead requires every triangle to be paired, which requires an even number of triangles and a feasible matching; Blossom-Quad may otherwise rely on auxiliary edges and postprocessing heuristics.

Geometry Prefiltering. Candidate quad merges are enumerated from all internal edges shared by exactly two adjacent triangles. For each candidate quadrilateral Q, we apply the following geometry prefiltering constraints:

1. **Interior Angle constraint.** Let $\{\theta_j\}_{j=1}^4$ denote the four interior angles of Q. To avoid extreme or degenerate quadrilaterals, we require all angles to lie within a prescribed range:
$$\theta_{\min} \le \theta_j \le \theta_{\max}, \quad \forall j \in \{1,2,3,4\}, \tag{3}$$
where we set $[\theta_{\min}, \theta_{\max}] = [30^\circ, 140^\circ]$ in practice.
2. **Convexity constraint.** We reject self-intersecting or concave quads by checking orientation consistency under a canonical vertex ordering.
3. **Dihedral constraint.** To preserve sharp feature edges and avoid highly twisted quads, we split Q along its two diagonals and require the resulting internal dihedral angles ϕ_1 and ϕ_2 to satisfy
$$\phi_1 \le \phi_{\text{thresh}}, \quad \phi_2 \le \phi_{\text{thresh}}, \tag{4}$$
where $\phi_{\text{thresh}} = 45^\circ$.
4. **Centroid constraint.** Let $S = \{v_j\}_{j=1}^k$ denote the candidate-face vertices, where $k = 3$ for a triangle and $k = 4$ for a quadrilateral. We compute the geometric centroid in Euclidean coordinates as $\hat{c}(S) = \frac{1}{k}\sum_{j=1}^k v_j$, and let c_{gen} be the generated centroid. We accept a quadrilateral candidate only if
$$\|\hat{c} - c_{\text{gen}}\|_2 \le \tau_{\text{quad}}, \tag{5}$$
where $\tau_{\text{quad}} = 2 \times 10^{-3}$. For triangle candidates, we apply the same centroid validation with a separate threshold $\tau_{\text{tri}} = 5 \times 10^{-3}$.
5. **UV-seam constraint.**
We represent the input mesh in loop-level form, where each triangle corner has its own position $p_i \in \mathbb{R}^3$ and UV coordinate $t_i \in [0,1]^2$. Corners are welded only when both their positions and UV coordinates agree within tolerance:
$$\text{weld}(i,j) = 1 \iff \|p_i - p_j\|_\infty \le \varepsilon_{\text{pos}} \ \wedge \ \|t_i - t_j\|_\infty \le \varepsilon_{\text{uv}}, \tag{6}$$
where $\varepsilon_{\text{pos}} = \varepsilon_{\text{uv}} = 10^{-6}$. Thus, corners at the same 3D position but on opposite sides of a UV seam retain different vertex indices. The resulting face set partitions into vertex-adjacency connected components $\{I_k\}_{k=1}^K$, each corresponding to one UV island. A seam edge e is a boundary edge of every island it borders:
$$|\{f \in I_k : e \subset f\}| = 1, \quad \text{for each island } I_k \text{ adjacent to } e. \tag{7}$$
Since a merge candidate must be shared by exactly two triangles, we generate candidates independently within each island. Consequently,
$$E_{\text{seam}} \cap E_{\text{cand}} = \varnothing, \tag{8}$$
so seam edges are never removed and the UV parametrization is preserved. Output UVs are stored per vertex in the welded index space. Fig. 11 illustrates this seam-preserving behavior.
6. **Symmetry-plane constraint.**
We estimate a mirror-symmetry plane from the vertex centroid $\bar{c}$ and the eigenvectors $\{n_i\}_{i=1}^3$ of the vertex covariance matrix. Each principal axis defines a candidate plane $\Pi_i = \{x : n_i^\top x + d_i = 0\}$ through $\bar{c}$, with $d_i = -n_i^\top \bar{c}$. We score a candidate by its reflection coverage:
$$C(\Pi) = \frac{1}{|V|}\sum_{v \in V} \mathbf{1}[\text{dist}(r_\Pi(v), V) < \varepsilon_{\text{sym}}], \qquad r_\Pi(v) = v - 2(n^\top v + d)n, \tag{9}$$
where the nearest-vertex distance is queried using a uniform grid hash and $\varepsilon_{\text{sym}} = 10^{-2}\,\text{diag}(B)$, with $\text{diag}(B)$ denoting the bounding-box diagonal. We refine the offset d with a one-dimensional search and accept the plane only when $C(\Pi) \ge \tau_{\text{cov}}$, where $\tau_{\text{cov}} = 0.9$. To

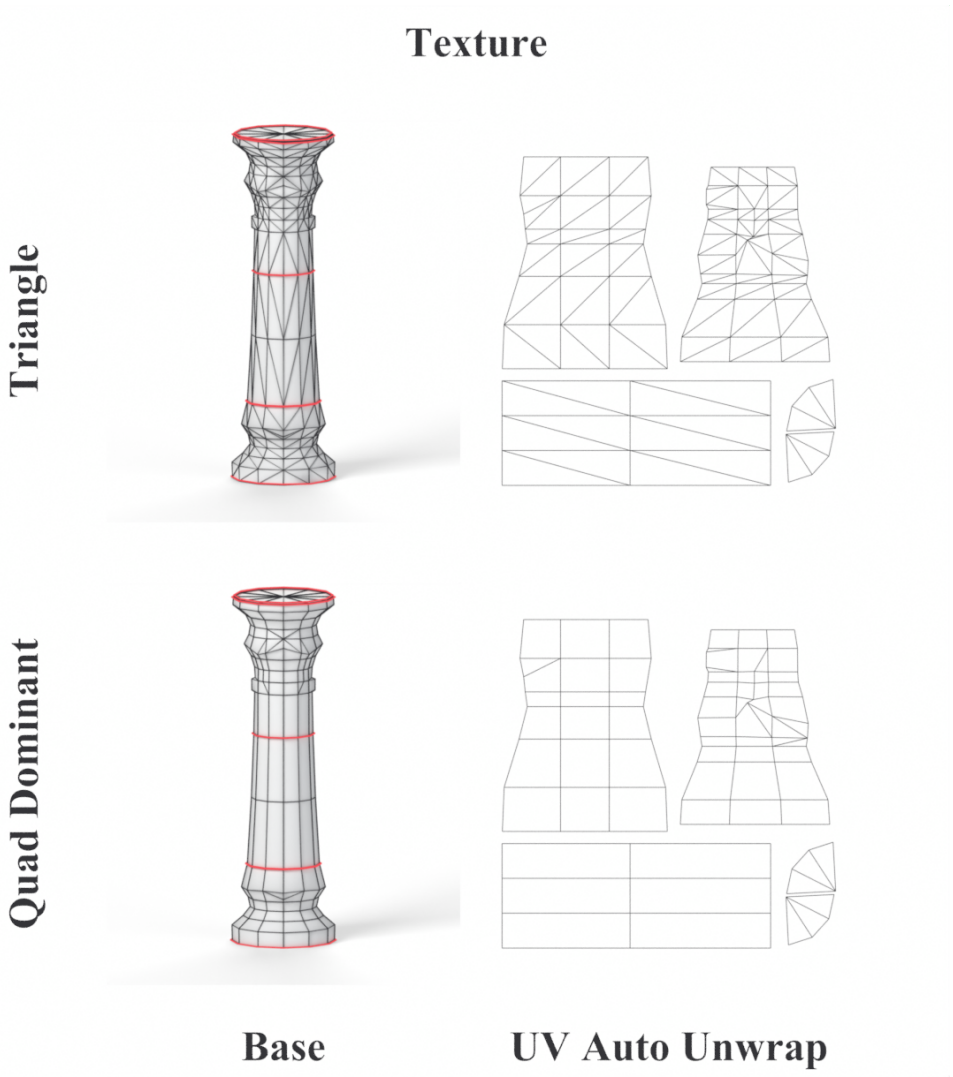

Figure 11: UV-seam preservation in the Tri-to-Quad Operator.

reject degenerate mid-surface planes on flat or rod-like objects, we also discard a candidate if its band contains too many vertices:

$$\frac{\left|\{v \in V : |n^\top v + d| < \delta\}\right|}{|V|} > \tau_{\text{band}}, \qquad \tau_{\text{band}} = 0.2. \tag{10}$$

Given an accepted plane, an internal edge $e = (u, v)$ is protected from merging only when both endpoints lie close to that plane:

$$|n^\top u + d| < \delta_e \ \wedge \ |n^\top v + d| < \delta_e, \qquad \delta_e = \min(\delta,\, \beta\|u - v\|_2)\,, \tag{11}$$

with $\beta = 0.2$. The global tolerance is selected adaptively, starting from a cap $\delta_{\text{cap}} = 5 \times 10^{-3}\ \text{diag}(B)$. If the sorted vertex-to-plane distances contain a decisive multiplicative gap of at least $30\times$, indicating an exact on-plane vertex loop, we place δ within that gap; otherwise, we use the numerical floor $\delta = 10^{-3}\ \text{diag}(B)$. The edge-length-dependent term restricts protection to edges that closely follow the plane, avoiding excessive protection of thin structures pierced by it. Fig. 12 shows an example of the detected plane and protected edges.

Let E_{sym} denote the set of protected symmetry-plane edges. The final candidate pool is

$$E_{\text{cand}} = E_{\text{int}} \setminus (E_{\text{seam}} \cup E_{\text{sym}})\,, \tag{12}$$

and all remaining candidates proceed through constraints 1–4 and the quality-scored matching unchanged.

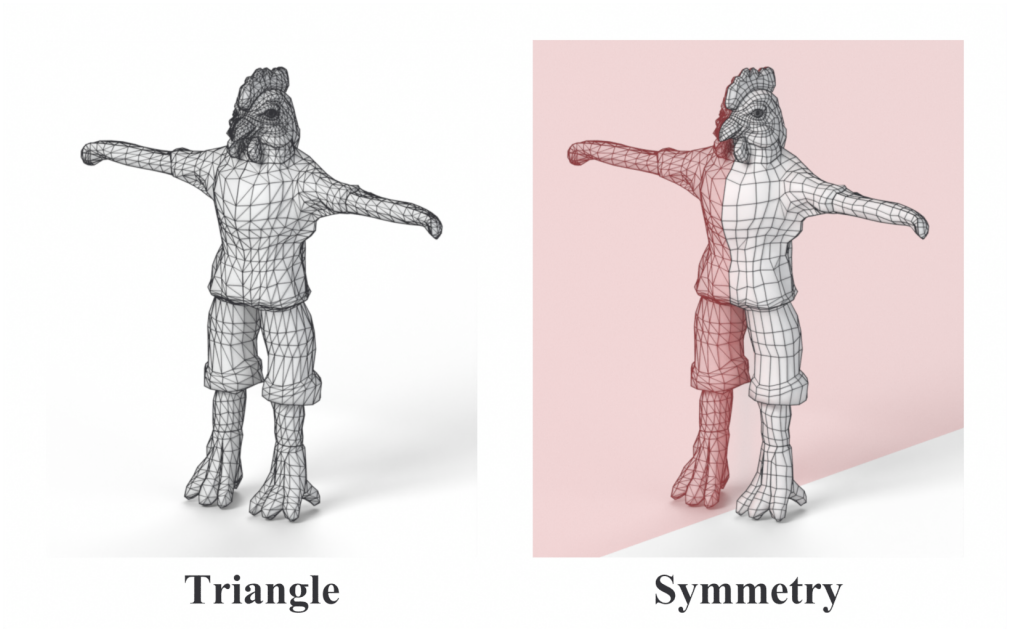

Figure 12: Symmetry-plane detection and edge protection in the Tri-to-Quad Operator.

Global Merging. Details are provided in the main paper.

Tri-to-Quad Operator Ablation.

We ablate two main components in our *Tri-to-Quad Operator*: Geometry Prefiltering and Merging Quality Equation Q_{angle} and Q_{align}. As shown in Table 3, we apply all variants of our operator to artistic triangle meshes.

Since our merge-edge operator will preserve the original geometry with similar geometric metrics, we therefore introduce two topological metrics: *Opposite Edge Parallelism* (OEP) and *Edge Flow Continuity* (EFC). OEP measures the average parallelism of opposite edge pairs within each quad, with $|\cos(e_0, e_2)|$ and $|\cos(e_1, e_3)|$, reflecting local quad regularity. EFC measures the directional consistency of opposite edges across adjacent quads, reflecting edge flow smoothness. The results show that Geometry Prefiltering and Q_{angle} substantially improve topology by removing invalid candidates and favoring quads with regular angles. Although Q_{align} yields smaller quantitative gains, it is crucial in ambiguous regions with similar angle quality to produce smoother edge flow around structures such as heels, shoulders, and crotch junctions, as shown in Fig. 9.

Normal Consistency. We enforce deterministic normal consistency during merging as shown in Fig. 13 : triangles are merged only if $\mathbf{n}_A \cdot \mathbf{n}_B > 0$, and the resulting quad normal (computed via Newell's method) is oriented consistently by flipping vertex order if necessary, yielding globally outward-facing normals.

B Architecture Details

B.1 Hourglass Transformer for Stage I Anchor Prediction

Rather than treating mesh token generation as a generic sequence modeling problem, we leverage the hierarchical structure of our two-level (point–coordinate) token representation and adopt the Hourglass Transformers design from Meshtron (Hao et al., 2024). This architecture consists of a causality-preserving shortening stage followed by a symmetric upsampling stage, forming a three-block hourglass structure. Compared to face-level autoregressive serialization, this design improves computational efficiency while retaining the essential global structure needed for Anchor Prediction *Stage I* point-level generation.

Let the input token embeddings be $\mathbf{E}^{(0)} \in \mathbb{R}^{L \times D}$, where L is the token length and D is the embedding dimension. The first Transformer block operates at full resolution and compresses the sequence with a shortening module using downsampling factor $r = 3$, producing a compact bottleneck representation. A second Transformer block then models dependencies at the bottleneck scale. Finally,

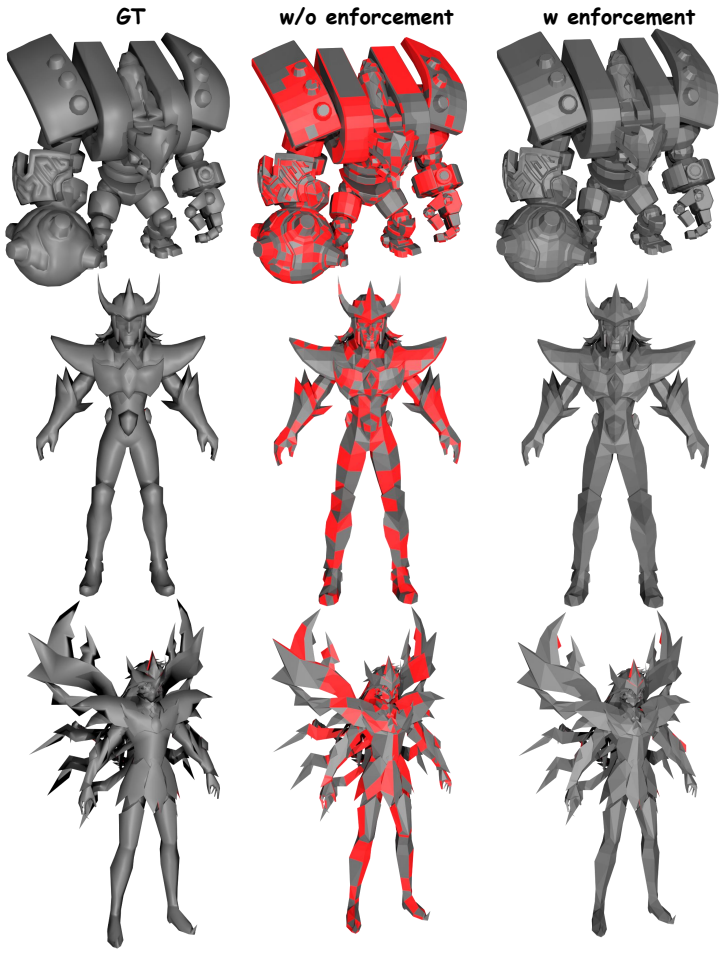

Figure 13: **Qualitative visualizations of normal consistency enforcement during triangle merging.** w/o enforcement leads to faces with inconsistent normal directions, highlighted in red (inward-facing faces). w enforcement shows results with consistent gray (outward-facing faces).

we restore the original temporal resolution through an upsampling stage followed by a third Transformer block:

$$\begin{aligned}\mathbf{E}^{(1)} &= \mathrm{Shorten}_3\Big(\mathrm{Block}_1\Big(\mathbf{E}^{(0)}\Big)\Big) \in \mathbb{R}^{\frac{L}{3}\times D},\\ \mathbf{E}^{(2)} &= \mathrm{Block}_2\Big(\mathbf{E}^{(1)}\Big) \in \mathbb{R}^{\frac{L}{3}\times D},\\ \mathbf{E}^{(3)} &= \mathrm{Block}_3\Big(\mathrm{Upsample}_3\Big(\mathbf{E}^{(2)}\Big)\Big) \in \mathbb{R}^{L\times D}.\end{aligned} \tag{13}$$

We follow Meshtron (Hao et al., 2024) to implement $\mathrm{Shorten}_3(\cdot)$ and $\mathrm{Upsample}_3(\cdot)$ in a causality-preserving manner. In our setting, a single shortening stage with $r = 3$ is sufficient for Stage-I point-level generation, offering a favorable trade-off between efficiency and information preservation compared to padding-heavy face-level baselines.

B.2 Adaptive Michelangelo Point Cloud Encoder

Unlike the original Michelangelo encoder (Zhao et al., 2023), which uses a fixed number of learnable query tokens, we additionally develop an *adaptive* variant trained on multi-scale surface point clouds. Given N surface points (the union of mesh vertices and face centroids), the adaptive encoder produces a variable-length context sequence with N embedding tokens, where N varies across samples. This design is mainly introduced to support Link Modeling *(Stage II)*, where the number of surface points naturally changes per asset. In Anchor Prediction *(Stage I)*, we follow the canonical Michelangelo setting and retain a fixed set of learnable query tokens to obtain a compact global context for generation.

At each Transformer block, we inject the encoded point-cloud context through a residual cross-attention layer:

$$\mathbf{E}' = \mathbf{E} + \mathrm{CrossAttn}(\mathbf{E}, \mathbf{C}), \tag{14}$$

where $\mathbf{E} \in \mathbb{R}^{L\times D}$ denotes the current token embeddings in the Hourglass Transformer, and $\mathbf{C}$ denotes the encoded point-cloud context from the Michelangelo encoder.

B.3 Hard Negative Mining Details

In Link Modeling *(Stage II)*, we adopt Hard Negative Mining with Adaptive Top-K Selection Strategy to let the model first learn coarse separation with fewer hard negatives and later incorporate more challenging negatives for improved discriminability.

We define the squared embedding distance as

$$d(u, v) \triangleq \|f(u) - f(v)\|_2^2. \tag{15}$$

For each anchor–positive pair $(A^{(i)}, P^{(i)})$, we define the margin violation of a candidate negative N as

$$\mathrm{violation}(N) = d\Big(A^{(i)}, P^{(i)}\Big) - d\Big(A^{(i)}, N\Big). \tag{16}$$

We then select the Top-K negatives with the largest violation values:

$$\mathcal{N}_{\mathrm{hard}}^{(i)}(k) = \mathrm{TopK}_{N\in\mathcal{N}^{(i)}}\big(\mathrm{violation}(N),\, k\big). \tag{17}$$

The triplet margin loss is computed only over these hard negatives:

$$\mathcal{L}_{\mathrm{triplet}}^{\mathrm{hard}} = \frac{1}{M}\sum_{i=1}^{M}\frac{1}{|\mathcal{N}_{\mathrm{hard}}^{(i)}(k)|}\sum_{N\in\mathcal{N}_{\mathrm{hard}}^{(i)}(k)}\max\Big(0,\; d(A^{(i)}, P^{(i)}) - d(A^{(i)}, N) + m\Big). \tag{18}$$

Inspired by curriculum learning (Bengio et al., 2009), we progressively increase k during training:

$$k(t) = \min\Big(k_{\max},\; k_{\min} + \big\lfloor \alpha t \big\rfloor\Big), \tag{19}$$

where t denotes the training epoch, and $k_{\min}$ and $k_{\max}$ are the initial and final Top-K values.

B.4 Stage III Face Assembly Details

Contrastive Learning Trained Model (CLTM) Given a face centroid c and a vertex set $\mathcal{V}$, $\mathtt{CLTM}(c, \mathcal{V})$ ranks vertices by their distances to the centroid-conditioned embedding, and returns a shortlist of candidate vertices that are most likely to belong to the same face. This retrieval step significantly reduces the combinatorial search space for face reconstruction while preserving anisotropic and production-style vertex grouping learned in Link Modeling *(Stage II)*.

Progressive Candidate Face Selection (PCFS) We reconstruct the mesh topology using a retrieve-and-verify strategy. First, for each centroid, we retrieve the Top-K nearest vertices in the latent feature space as candidates. We then generate potential quadrilaterals by combining these vertices and rank them in ascending order based on the mean latent feature distance between the constituent vertices and the target centroid. We iterate through this ranked list and perform geometric verification to check for validity. We employ a *quad-first, triangle-next* strategy: The first quadrilateral that satisfies these geometric constraints is selected. If no valid quadrilateral is found among the candidates, we then naturally attempt to form a valid triangle from the candidate vertices, applying the streamlined verification process to ensure topological correctness. Final centroid consistency is evaluated in Euclidean coordinates using the geometric centroid of the candidate vertices, as specified in Appendix Section A.3.

We denote by $\mathtt{PCFS}(c_i, P_m, k)$ a progressive enumerator that returns candidate k-vertex sets $S \subseteq P_m$. During inference, we adopt a deterministic *quad-first, triangle-next* assembly strategy: we first enumerate $k = 4$ candidates returned by $\mathtt{PCFS}(\cdot, k = 4)$ and apply the validation rules; if no valid quad is found, we fall back to enumerate $k = 3$ candidates via $\mathtt{PCFS}(\cdot, k = 3)$. To avoid redundant computation across expanding candidate pools, combinations tested in smaller pools are cached and skipped when evaluating larger pools.

Algorithm 1: Stage III Face Assembly with Geometry Prefiltering and Centroid Tolerance

Require: Centroid set $\mathcal{C}$, vertex set $\mathcal{V}$
Require: Contrastive Learning Trained Model $\texttt{CLTM}(\cdot)$ in B.4
Require: Progressive Candidate Face Selection $\texttt{PCFS}(\cdot)$ in B.4
Require: Geometry prefiltering rules $\texttt{GeoFilter}(\cdot)$
Require: Centroid tolerance τ_{quad}, τ_{tri}
Ensure: Constructed face set $\mathcal{F}$

$\mathcal{F} \leftarrow \emptyset$
for each centroid $c \in \mathcal{C}$ **do**
  $\mathcal{V}_{\text{cand}} \leftarrow \texttt{TopK}\big(\texttt{CLTM}(c, \mathcal{V}), K = 20\big)$
  $\texttt{found} \leftarrow$ **false**
  ▷ Quad-first
  for each 4-vertex set $S \leftarrow \texttt{PCFS}(c, \mathcal{V}_{\text{cand}}, k = 4)$ **do**
    if $\texttt{GeoFilter}(S)$ **and** $\texttt{QuadCentroidTol}(S, c) \le \tau_{\text{quad}}$ **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{\texttt{MakeQuad}(S)\}$
      $\texttt{found} \leftarrow$ **true**
      break
    end if
  end for
  ▷ Tri-Fallback
  if not $\texttt{found}$ **then**
    for each 3-vertex set $S \leftarrow \texttt{PCFS}(c, \mathcal{V}_{\text{cand}}, k = 3)$ **do**
      if $\texttt{GeoFilter}(S)$ **and** $\texttt{TriCentroidTol}(S, c) \le \tau_{\text{tri}}$ **then**
        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\texttt{MakeTri}(S)\}$
        break
      end if
    end for
  end if
end for
return $\mathcal{F}$

We provide the detailed pseudocode of Algorithm 1, which specifies the complete Face Assembly *(Stage III)* face assembly pipeline used in our experiments. This procedure operationalizes the centroid–vertex links predicted in Link Modeling *(Stage II)* by retrieving a compact candidate vertex set and progressively constructing valid polygonal faces via PCFS. The algorithm incorporates both geometric prefiltering and centroid consistency checks, enabling efficient and reliable face generation while avoiding combinatorial explosion.

B.5 Mesh Tokenization and Vocabulary Strategies

Anchor Prediction *(Stage I)* predicts a unified set of anchors, including both mesh vertices and face centroids. This introduces two practical tokenization challenges for autoregressive sequence modeling. First, the sequence must distinguish different point types, i.e., vertices and centroids. Second, each 3D coordinate token must encode its axis identity (z, y, or x). We therefore study both sequence organization strategies and vocabulary designs.

Sequence Organization. To analyze how point-type organization affects convergence and training stability, we evaluate four representative tokenization modes.

Single. This mode serializes only vertices sorted by the z–y–x lexicographic order. It serves as a simplified baseline that isolates the optimization difficulty without mixing vertices and centroids:

$$\mathcal{M} = \Big\{ \mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \dots \Big\}.$$

Dual Codebook (Mixed). This mode mixes vertices and centroids into a single sequence and jointly sorts them by the z–y–x lexicographic order. To distinguish the two point types, vertex coordinates and centroid coordinates are encoded using disjoint vocabularies:

$$\mathcal{M} = \Big\{ \mathbf{v}^{(1)}, \mathbf{c}^{(1)}, \mathbf{v}^{(2)}, \dots \Big\}.$$

Dual Codebook (Separate). This mode separates vertices and centroids into two consecutive blocks, each sorted by the z–y–x lexicographic order. Similar to the mixed variant, it uses disjoint coordinate vocabularies for vertices and centroids:

$$\mathcal{M} = \Big\{ \underbrace{\mathbf{v}^{(1)}, \dots}_{\mathbf{V}}, \underbrace{\mathbf{c}^{(1)}, \dots}_{\mathbf{C}} \Big\}.$$

Single Separate. This mode also separates vertices and centroids into two blocks, but inserts an explicit <sep> token between them. The delimiter provides sequence-level type separation, allowing

vertices and centroids to share the same coordinate vocabulary:

$$\mathcal{M} = \Big\{ \underbrace{\mathbf{v}^{(1)}, \ldots}_{\mathbf{V}}, \texttt{<sep>}, \underbrace{\mathbf{c}^{(1)}, \ldots}_{\mathbf{C}} \Big\}.$$

Axis-aware Vocabulary. Besides point-type organization, we further study whether coordinate axes should share a vocabulary. A common strategy is to use the same coordinate vocabulary for z, y, and x, relying on positional embedding such as RoPE (Su et al., 2024) to infer axis identity. In contrast, our axis-aware vocabulary assigns disjoint index ranges to different axes, so that each coordinate token explicitly encodes whether it corresponds to z, y, or x.

As shown in Tab. 4, axis-aware factorization consistently improves convergence across tokenization modes. We attribute this to the reduced burden of implicit classification: the model no longer needs to infer both point type and coordinate axis solely from sequence context. Among all variants, *Single Separate* with axis-aware vocabulary achieves the best convergence efficiency. We therefore adopt this design in our final model.

Method	per_axis_vocab	Token Structure	Vocab Size	Converge Epoch
single	false	vertex [0–1023]	1024	55
	true	z[0–1023], y[1024–2047], x[2048–3071]	3072	36
dual	false	vertex [0–1023], centroid [1024–2047]	2048	74
	true	z_v[0–1023], y_v[1024–2047], x_v[2048–3071],z_c[3072–4095], y_c[4096–5119], x_c[5120–6143]	6144	66
dual_separate	false	vertex [0–1023], centroid [1024–2047]	2048	59
	true	z_v[0–1023], y_v[1024–2047], x_v[2048–3071],z_c[3072–4095], y_c[4096–5119], x_c[5120–6143]	6144	65
single_separate	false	vertex [0–1023] + ⟨SEP⟩ + centroid [0–1023]	1025	64
	true	z[0–1023], y[1024–2047], x[2048–3071] + ⟨SEP⟩	3073	52

Table 4: **Tokenization ablation for Stage I anchor prediction.** We evaluate different token structures and per-axis vocabulary factorization, reporting vocabulary size and convergence epoch (loss ≤ 0.1). Per-axis factorization improves convergence across all schemes, with single_separate achieving the best efficiency–stability trade-off. Results are obtained on a 2K-category-balanced subset and verified to hold on the full training set.

B.6 Ablation on Tokenization Selection

As shown in Fig. 15, per axis vocab has a relatively positive effect for all tokenization methods. As shown in Fig. 16, single separate tokenization method with per axis vocabulary surpasses both dual and dual separate tokenization methods on convergence efficiency. All results are obtained on a 2K-category-balanced subset for better visualization and verified to hold on the full training set.

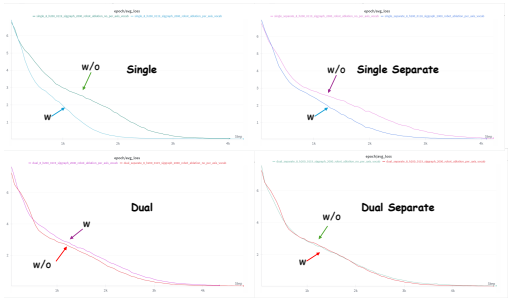

Figure 15: Comparisons for convergence rate of each tokenization method w or w/o per axis vocabulary.

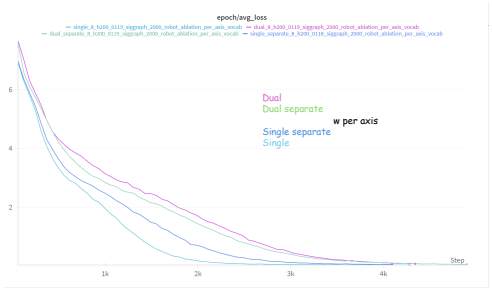

Figure 16: Comparisons for convergence rate among each tokenization method with per axis vocabulary.

B.7 Edge Flow Ratio (EFR)

While CD, HD, and IoU evaluate geometric fidelity, they do not capture whether a quad-dominant mesh preserves production-ready edge flow. We therefore introduce *Edge Flow Ratio* (EFR) to measure how well the edges of an output mesh align with salient feature lines of the ground-truth mesh, including sharp creases, boundary contours, and loop-like structures.

Feature Line Extraction. Given the ground-truth mesh $\mathcal{M}_{\text{gt}}$, we first detect hard edges, including boundary edges and sharp edges whose dihedral angles exceed a threshold. Connected hard-edge chains are then grouped into two types of feature lines: long polyline features, which capture creases and ridges, and closed loop features, which capture ring-like structures such as holes or cylindrical caps. Each feature line is represented as an ordered 3D polyline. Fig. 12 visualizes representative extracted feature lines.

Edge Chain Matching. For each ground-truth feature line $\mathbf{p} = (p_1, \ldots, p_K)$, we search for the best-matching edge chain on the output mesh $\mathcal{M}_{\text{out}}$. We first resample $\mathbf{p}$ into dense points $\{\hat{p}_i\}_{i=1}^{M}$ and estimate local unit tangents $\{\hat{t}_i\}_{i=1}^{M}$. Output vertices close to the feature line are collected as

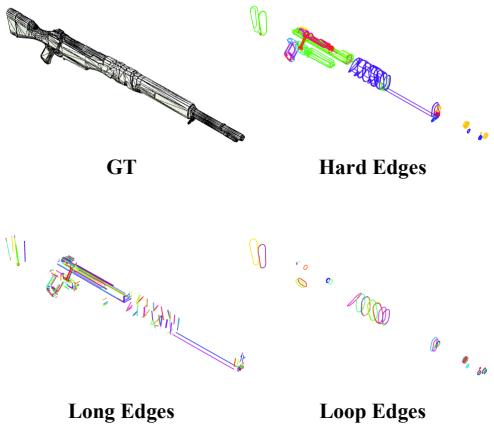

Figure 12: **Qualitative visualizations of feature line extraction for Edge Flow Ratio (EFR) calculation.**

$$\mathcal{V}_{\text{near}} = \left\{ v \in \mathcal{V}_{\text{out}} \;\middle|\; \min_i \|v - \hat{p}_i\| < \delta \right\},$$

where each nearby vertex is associated with its closest feature-line sample

$$\phi(v) = \arg\min_i \|v - \hat{p}_i\|.$$

Starting from each nearby vertex, we greedily trace an edge chain by following the adjacent vertex whose outgoing direction best aligns with the local feature-line tangent:

$$v_{k+1} = \arg\min_{u \in \mathcal{N}(v_k) \cap \mathcal{V}_{\text{near}}} \arccos\left(\frac{u - v_k}{\|u - v_k\|} \cdot \left(s\, \hat{t}_{\phi(u)} \right) \right),$$

where $\mathcal{N}(v_k)$ is the one-ring neighborhood of v_k, and $s \in \{-1, +1\}$ resolves the traversal direction. The traversal stops when no adjacent vertex satisfies the angular threshold. We perform this search bidirectionally and retain the chain $\mathbf{q}^*$ with the smallest distance to $\mathbf{p}$.

Alignment Score and EFR. Given a ground-truth feature line $\mathbf{p}$ and its matched output chain $\mathbf{q}^*$, we uniformly resample both curves to N_s points and compute the bidirectional curve distance:

$$d(\mathbf{p}, \mathbf{q}^*) = \min\left(\frac{1}{N_s} \sum_{i=1}^{N_s} \|\tilde{p}_i - \tilde{q}_i\|, \frac{1}{N_s} \sum_{i=1}^{N_s} \|\tilde{p}_i - \tilde{q}_{N_s - i + 1}\| \right).$$

The corresponding alignment score is

$$s(\mathbf{p}, \mathbf{q}^*) = \exp\left(-\frac{d(\mathbf{p}, \mathbf{q}^*)}{\tau} \right),$$

where τ controls the sensitivity to misalignment. Finally, EFR is defined as the average alignment score over all extracted feature lines:

$$\text{EFR} = \frac{1}{N_L + N_C} \left(\sum_{i=1}^{N_L} s(\ell_i, \mathbf{q}^*_{\ell_i}) + \sum_{j=1}^{N_C} s(c_j, \mathbf{q}^*_{c_j}) \right),$$

where $\{\ell_i\}_{i=1}^{N_L}$ and $\{c_j\}_{j=1}^{N_C}$ denote long polyline features and loop features, respectively. Higher EFR indicates better edge-flow alignment with the ground-truth feature structure.

Implementation Details. We use proximity thresholds $\delta = 0.05$ for long features and $\delta = 0.01$ for loop features. The angular thresholds are set to 0.12 radians for long features and 0.78 radians for loops to accommodate higher curvature. We use $M = 100$ samples for tangent-guided search and $N_s = 500$ samples for final curve-distance computation. Feature extraction and matching are performed per connected component, with bounding-box prefiltering to skip spatially disjoint components.

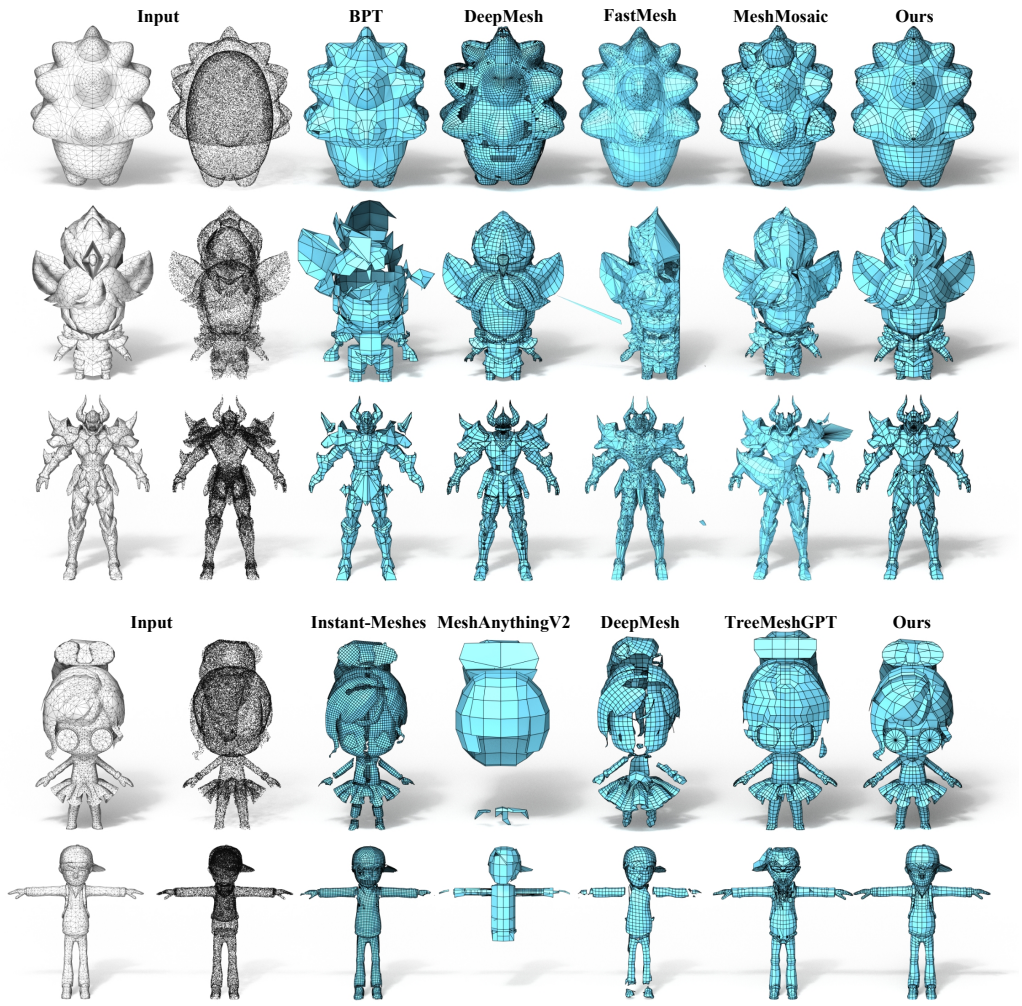

Figure 13: More qualitative comparison with triangle-based generation methods postprocessed by our *Tri-to-Quad Operator*. The upper three examples compare our method with BPT (Weng et al., 2025), DeepMesh (Zhao et al., 2025), FastMesh (Kim et al., 2025), and MeshMosaic (Xu et al., 2025). The lower two examples compare our method with Instant-Meshes (Jakob et al., 2015), MeshAnythingV2 (Chen et al., 2025b), DeepMesh (Zhao et al., 2025), and TreeMeshGPT (Lionar et al., 2025).

C More Qualitative Results

C.1 Qualitative Comparison with Mesh Generation Methods.

We provide more qualitative comparison with other triangle-based generation methods postprocessed by our *Tri-to-Quad Operator* in Fig. 13. Our method natively learns semantically anisotropy layout and coherent edge flow from quad-dominant mesh dataset through centroid-conditioned vertex grouping, avoiding the fragility of triangle-first generation followed by uniform geometric postprocessing.

C.2 Field-aligned Remeshing Methods

We present qualitative visualizations comparing with traditional quad-remeshing tools on representative meshes in Fig. 14 highlight common failure modes of field-aligned parametrization methods, including unnecessarily dense tessellation, isotropic layouts and shape infidelity introduced to satisfy traditional mathematical quad quality criteria.

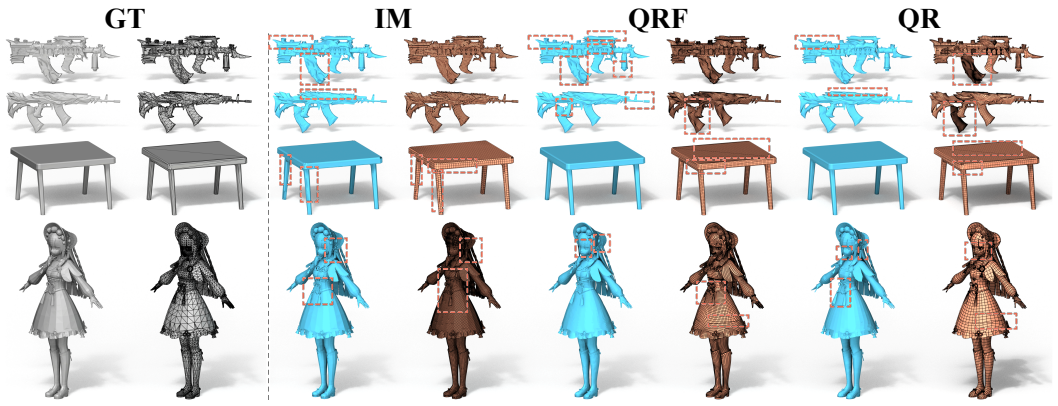

Figure 14: Qualitative visualizations of traditional quad-remeshing methods. The results show that these methods often generate unnecessarily dense face counts even on simple surfaces, make suboptimal topological decisions, and distort the underlying shape in pursuit of traditional mathematically regular quads. (GT = Ground-Truth triangular mesh; IM = Instant-Meshes (Jakob et al., 2015); QRF = QRemeshify (Ksami, 2024); QR = Quad Remesher (Exoside, 2019).)

C.3 Software-based Remeshing Baselines

We compare our global *Tri-to-Quad Operator* against two more baselines: a greedy-based variant of our operator and a built-in algorithm in PyMeshLab (Muntoni & Cignoni, 2021).

Greedy Baseline. This baseline uses the same optimization objectives as ours, but optimizes it via local heuristics. At each step, it selects the internal edge that maximizes the immediate quality gain, updates connectivity, and repeats until no valid merges remain. Due to its purely local decisions, it frequently converges to suboptimal merge configurations, resulting in degraded quad quality.

Pymeshlab Baseline. We use PyMeshLab's built-in MeshLab filter `meshing_tri_to_quad_dominant` with `level=2` (*Better quad shape*), which converts a triangular mesh into a quad-dominant mesh by pairing suitable adjacent triangles. This setting is the highest-quality option provided by the filter and is used as the PyMeshLab baseline in our qualitative comparison.

As illustrated in Fig. 15, our method produces quads that are more regular and exhibit fewer artifacts in both shape and topology under Global Merging Problem Formulation and Geometry Prefiltering.

C.4 Polygonal Mesh Generation

Stage I predicts only vertices and face centroids, rather than explicit face connectivity with a fixed valence, so the same architecture can in principle represent arbitrary n-gon meshes. Large-scale, high-quality n-gon datasets are extremely difficult to obtain; we therefore use Goldberg polyhedra (Liu et al., 2022; Hart, 2012) to curate data and test this capability. We train on a subset of Goldberg topologies and evaluate on unseen topology values without architectural changes. As shown in Fig. 8, QuadLink naturally generalizes to arbitrary n-gon meshes and generates valid Goldberg polyhedra with unseen topology values.

Dataset Curation. We procedurally construct *Goldberg polyhedra* using a convex-hull-based geodesic-dual construction. For each pair of non-negative integers (m, n), with $m \geq n$, we define the Goldberg index as

$$T = m^2 + mn + n^2. \tag{20}$$

We first enumerate integer lattice points (a, b) inside the fundamental Goldberg triangle,

$$\begin{aligned} (m+n)a + nb &\geq 0, \\ mb - na &\geq 0, \\ T - ma - (m+n)b &\geq 0. \end{aligned} \tag{21}$$

Each lattice point is mapped to the 20 faces of a unit icosahedron by barycentric interpolation and radial projection. To avoid class-dependent connectivity ambiguities for $n \neq 0$, we recover the geodesic triangulation by computing the convex hull of the projected spherical point set, and then take its topological dual to obtain the Goldberg polyhedron. The resulting dual mesh has

$$\begin{aligned} &V = 20T, \qquad E = 30T, \qquad F = 10T + 2, \\ &F = 12 \text{ pentagons} + (10T - 10) \text{ hexagons}. \end{aligned} \tag{22}$$

The integers (m, n) are frequency parameters that control the Goldberg subdivision pattern and thereby determine the number and relative size of polygons. By varying the integer pair (m, n), this procedure enables scalable generation of Goldberg meshes with controlled polygonal topology.

Topology Conditioning. As the Goldberg index T increases, normalized Goldberg polyhedra become increasingly close to the unit sphere, making their surface point clouds nearly indistinguishable across different topologies. This creates a *degenerate conditioning* scenario: the point cloud encoder produces nearly identical context for every Goldberg class, yet the model must generate meshes with vastly different face counts and connectivity. To resolve this ambiguity, we introduce a lightweight *topology encoder* that injects the Goldberg index T as an explicit conditioning signal. Concretely, a three-layer MLP maps the normalized scalar $\hat{T} = T/T_{\max}$ (with $T_{\max} = 500$) to a set of $k = 4$ context tokens of dimension d (matching the point cloud latent dimension):

$$\mathbf{c}_{\text{topo}} = \text{MLP}(\hat{T}) \in \mathbb{R}^{k \times d}. \tag{23}$$

These tokens are concatenated to the point cloud context $\mathbf{C}_{\text{pc}} \in \mathbb{R}^{N \times d}$ before cross-attention, yielding an augmented context $\mathbf{C} = [\mathbf{C}_{\text{pc}}; \mathbf{c}_{\text{topo}}] \in \mathbb{R}^{(N+k) \times d}$. During training, T is derived from the sample identifier; During inference, the user specifies the desired T to steer generation toward the target topology. The topology encoder adds only $\sim$0.5K learnable parameters and introduces no architectural change to the mesh transformer itself.

Experiments & results. We enumerate valid (m, n) pairs with unique T values and select 100 topologically distinct Goldberg classes, spanning $T \in [1, 324]$ and face counts from 12 to 3242. For each topology, we generate one canonical mesh and four randomly rotated variants, yielding 500 meshes in total. All meshes are normalized to $[-1, 1]^3$. We use 95 topologies for training and reserve 5 unseen topologies for testing.

We finetune our pretrained model without architectural changes. The only modification is in Face Assembly *(Stage III)*, where we replace the *quad-first, tri-next* strategy with a *hexagon-first, pentagon-next* strategy. Qualitative results are provided in the main paper.

C.5 Efficiency/Token Length Analysis.

Training and inference times in autoregressive mesh generation are influenced by generation completeness, model parameters, data distribution, sliding-window size and many other factors, making direct wall-clock comparisons unfair. Therefore, we follow common practice by using token length to reflect the representation-level efficiency gain, since dense self-attention scales as $\mathcal{O}(N^2)$ in sequence length.

For Meshtron's three-level face–vertex–coordinate tokenization (Hao et al., 2024) (Sec. 2), each triangular face contributes three vertices with three coordinates each, giving

$$N_{\text{Meshtron}} = 3 \times 3 \times F_{\triangle} = 9F_{\triangle}. \tag{24}$$

Because vertices shared by adjacent faces are repeatedly tokenized, this representation scales with the number of triangle faces. QuadLink instead predicts each unique vertex and face center once, giving

$$N_{\text{ours}} \approx 3\big(V + F_{\text{poly}}\big). \tag{25}$$

For a closed, genus-zero triangular mesh, Euler's formula $V - E + F_{\triangle} = 2$ and the face–edge incidence relation $3F_{\triangle} = 2E$ imply $F_{\triangle} = 2V - 4 \approx 2V$, so $N_{\text{Meshtron}} = 9F_{\triangle} \approx 18V$. For a closed genus-zero mesh containing only triangles and quads, define $q = F_4/F_{\text{poly}}$ as the fraction of quad faces, where $F_3 = (1 - q)F_{\text{poly}}$ and $F_4 = qF_{\text{poly}}$. Since every edge is incident to two faces, $2E = 3F_3 + 4F_4 = (3 + q)F_{\text{poly}}$. Substituting $E = (3 + q)F_{\text{poly}}/2$ into Euler's formula $V - E + F_{\text{poly}} = 2$ gives

$$F_{\text{poly}} = \frac{2(V - 2)}{1 + q} \approx \frac{2V}{1 + q}. \tag{26}$$

QuadGPT (Liu et al., 2026) assigns a fixed 12-token block to each face: 12 coordinate tokens for a quad and nine coordinate tokens plus three padding tokens for a triangle. With $N_{\text{ours}} \approx 3(V +$

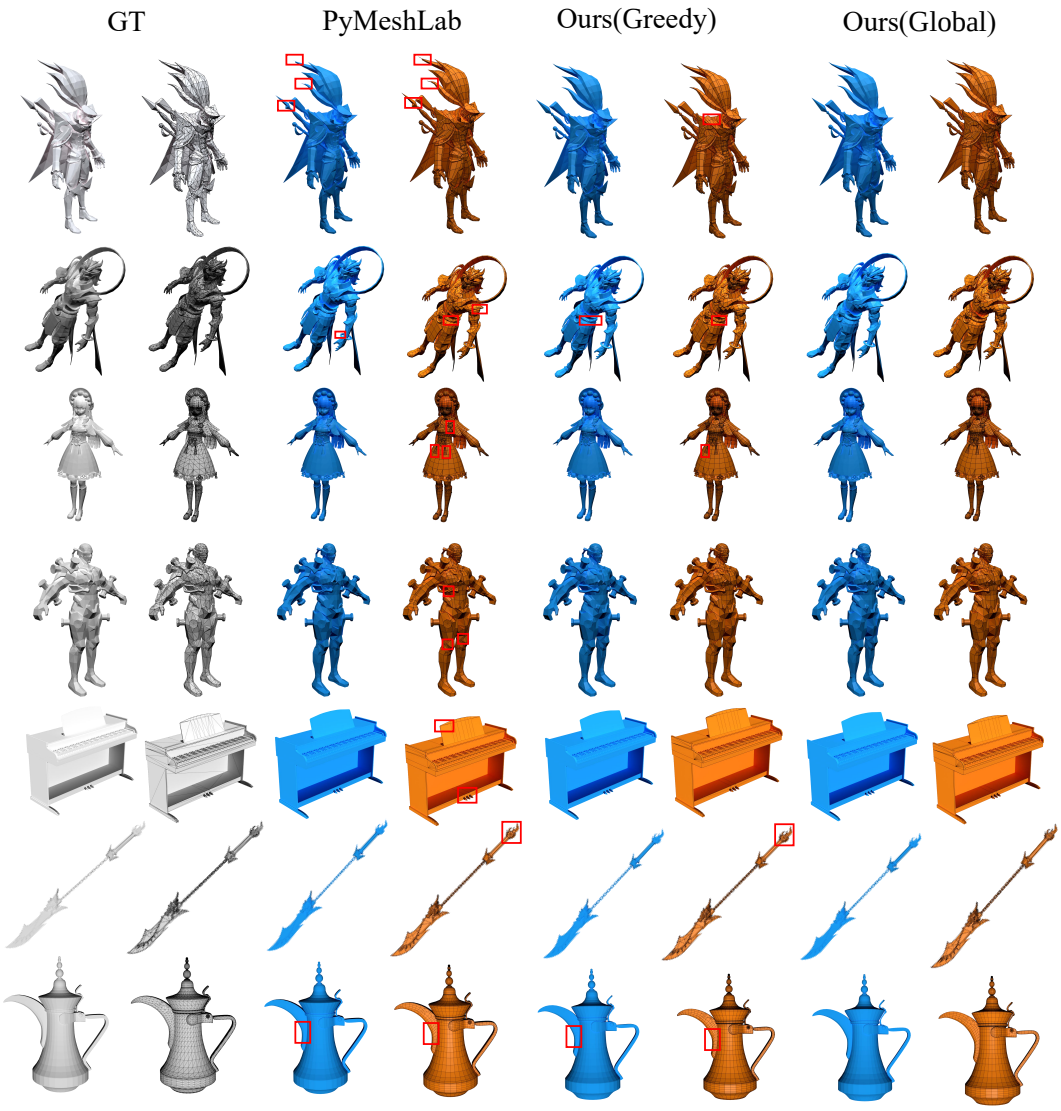

Figure 15: Qualitative comparison with Software-based quad remeshing methods. We compare our global *Tri-to-Quad Operator* against a greedy-based variant of our operator and a built-in algorithm in PyMeshLab (Muntoni & Cignoni, 2021). Extensive results show that our operator exhibits fewer artifacts in both shape and topology.

F_{poly}), the token-count ratios are

$$\begin{aligned} \frac{N_{\text{ours}}}{N_{\text{Meshtron}}} &\approx \frac{3(V+F_{\text{poly}})}{18V} \approx \frac{q+3}{6(q+1)}, \\ N_{\text{QuadGPT}} &= 12F_{\text{poly}}, \\ \frac{N_{\text{ours}}}{N_{\text{QuadGPT}}} &\approx \frac{3(V+F_{\text{poly}})}{12F_{\text{poly}}} \approx \frac{q+3}{8}. \end{aligned} \tag{27}$$

For a pure triangle mesh ($q = 0$), QuadLink uses about half as many tokens as Meshtron. For a pure quad mesh ($q = 1$), it uses about one-third as many. Compared with QuadGPT, it uses 37.5%–50% as many tokens, depending on the quad fraction q.

DeepMesh and BPT shorten mesh sequences with block-wise or patch-wise tokenization (Zhao et al., 2025; Weng et al., 2025). This compression trades sequence length against vocabulary size and structure, while making the face–vertex–coordinate hierarchy less explicit. If generation stops before a block or patch is complete, local face connectivity may remain unresolved, increasing the risk of holes.

D DISCUSSIONS

D.1 SINGULARITIES, WATERTIGHTNESS AND MANIFOLDNESS

There remains a substantial gap between the geometric criteria commonly emphasized in traditional remeshing pipelines and the practical requirements of production-ready mesh design or generation. Traditional geometry processing methods often prioritize mathematical regularity, such as minimizing singularities, enforcing watertightness, and preserving manifoldness. In contrast, production-ready assets are frequently optimized for editability and compatibility with downstream workflows, where such criteria are not always strictly enforced and may even be intentionally relaxed.

Singularities. In a pure quadrilateral mesh, a regular interior vertex is expected to have valence four, which indicates it is incident to four quad edges. Vertices with non-four valence, such as valence three, five, or higher, are commonly referred to as singularities. In traditional quad remeshing, singularities play a central role because they determine how the global edge flow branches, merges, or changes direction. Consequently, traditional field-aligned pipelines often aim to minimize the number of singularities, place them at geometrically meaningful locations, and maintain smooth cross field orientation with near-uniform quad sizes.

However, this notion is less directly aligned with production pipeline. Game and animation assets are rarely required to be mathematically clean pure-quad meshes throughout the entire pipeline. Instead, artists commonly work with quad-dominant meshes during modeling and editing, while final assets may be triangulated for rendering or engine deployment. Once triangles and occasional n-gons are introduced, strict valence-based singularities become abundant and less informative as a standalone quality measure. In such settings, production quality is more often determined by whether the mesh exhibits coherent and controllable edge flow. Well-structured edge flow supports common artist operations, including UV unwrapping, beveling, subdivision, and local editing, even when the mesh contains singularities or mixed face types. Therefore, rather than treating singularities as artifacts to be universally minimized, production-oriented topology often uses them as localized flow-control mechanisms that help redirect, terminate, or connect edge loops in a practical and editable manner.

Watertightness and Manifoldness. Watertightness and manifoldness are also central criteria in traditional geometry processing, especially for simulation, physical analysis and volumetric processing, where closed and well-defined surface topology is often required. A watertight mesh forms a closed surface without holes or boundary gaps, while a manifold mesh ensures that the local neighborhood of each point behaves like a disk or half-disk. These properties make the surface mathematically well behaved and simplify many downstream algorithms.

However, production-ready assets are not always optimized under these strict assumptions. In game, animation and content-creation pipelines, meshes are often composed of multiple disconnected components, overlapping parts, open surfaces, clothing or hair layers, accessories, thin shells for preserving well-designed shape details and topological editing flexibility. Such structures may be non-watertight or locally non-manifold, but they are still valid and useful production assets as long as they support operations such as modeling, editing, texturing, rigging, rendering, asset assembly and so on.

Therefore, while watertightness and manifoldness remain important for specific applications such as simulation or manufacturing, they are not uniform indicators for all production pipelines. For editable asset generation, overly enforcing these constraints may remove semantic structural separation, alter part boundaries, or introduce unnecessary geometric and topological repairs. In this work, we therefore focus on generating quad-dominant meshes that better supports production pipeline for games and animation, rather than enforcing watertight or manifold structure as strict objectives.

D.2 TOPOLOGY PRESERVATION AND LAYOUT DIVERSITY

Current autoregressive methods lack strict mathematical guarantees to preserve the macro-topology (e.g.,genus) implied by point cloud, although they empirically achieve high topological fidelity. For micro-topology(mesh connectivity), a single surface naturally admits multiple valid quad layouts. Our model embraces this one-to-many mapping by adjusting sampling parameters(T/top-k/top-p), enabling diverse yet plausible generated layouts for the same input. Incorporating explicit topological validation remains an important future direction for exploration.

D.3 LIMITATIONS

While QuadLink learns directly from production-ready quad-dominant meshes with semantically anisotropic layouts and coherent edge flow, its Anchor Prediction *(Stage I)* still follows an autoregressive generation paradigm. As a result, QuadLink shares several limitations commonly observed

in autoregressive 3D generative models, especially in terms of interactive and controllable remeshing. In traditional field-aligned parametrization pipelines such as Instant-Meshes (Jakob et al., 2015), users can often provide interactive guidance, such as symmetry constraints and target polygon counts. In contrast, integrating such controls into an autoregressive mesh generative model remains non-trivial, because these constraints are not naturally represented as the same type of coordinate tokens used for point prediction.

Symmetry Control. For symmetric shapes, users often expect the generated topology to preserve symmetry across corresponding parts. For example, if an input character has two approximately symmetric arms, the remeshing results should ideally maintain symmetric topology on both sides. However, small geometric asymmetries in the input point cloud, such as an accessory attached to only one arm, can perturb the autoregressive generation process. Since QuadLink orders anchor tokens according to a fixed spatial ordering and predicts them sequentially, such local asymmetries may propagate through next-token prediction and lead to asymmetric topology even on regions that are semantically expected to remain symmetric. One possible direction (Zhou et al., 2026) is to explicitly generate or condition on symmetry axes before generation. However, introducing such structural tokens increases sequence length and creates a mixed-token representation whose semantics differ from standard mesh coordinate tokens, potentially making autoregressive training less stable.

Target Polygon Count. Another limitation is explicit control over the target polygon count. In production pipelines, artists often require different levels of detail (LODs), where each asset must satisfy a target polygon count. Current autoregressive mesh generative models cannot reliably control vertices or faces count by simply conditioning on a scalar target count. This limits their direct use in workflows that require strict polygon budgets. Future work may require stronger control mechanisms or alternative generative formulations, such as discrete diffusion or refinement-based architectures (Song et al., 2025b), to support controllable polygon-count generation.

Overall, traditional field-aligned parametrization methods are easier to combine with interactive controls because such constraints can be directly written into rule-based optimization pipelines. However, their reliance on handcrafted rules also limits their robustness and generalization to complex production assets. In contrast, 3D generative models such as QuadLink provide stronger data-driven modeling capacity, but still lack a unified representation for interactive constraints. We believe that future modular designs or hybrid generative architectures that explicitly encode symmetry , polygon counts and other interative information will be important steps toward fully controllable production-ready mesh generation.