MESHREPAIR: AUTOREGRESSIVE TOPOLOGY REPAIR FOR GENERATED MESHES

Anonymous authors
Paper under double-blind review

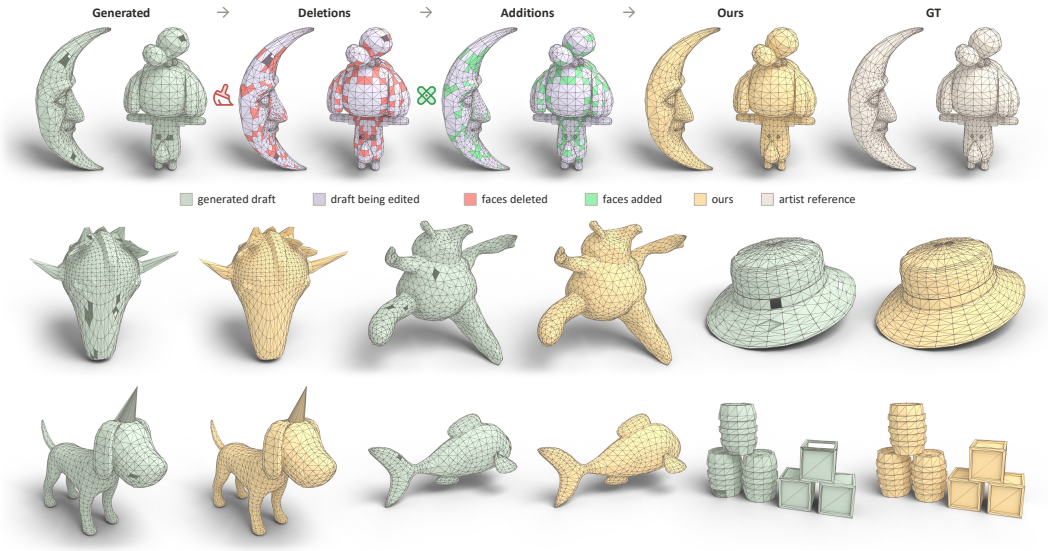


Figure 1: **MeshRepair repairs generated-mesh topology through face deletion and addition sequences.** Top: MeshRepair first predicts faces to delete (red) and then faces to add (green), producing the corrected meshes (ours), shown alongside the artist references (GT). Bottom: additional generated drafts and their corrected outputs.

ABSTRACT

Recent artisan mesh generation methods produce meshes whose topology comes close to the preferences of human artists. But gaps to artist quality remain because generated meshes commonly contain holes, stray faces, and duplicated faces. Regenerating a mesh from scratch is costly, and a new draw may change the preferred topology entirely or introduce errors elsewhere. In this work we study artisan topology correction. The task is to correct the topology of a generated mesh (the draft) in place, toward the layout an artist would make. We propose MeshRepair for this task. It is a face-level correction network based on a prefix-LM. It reads the mesh bidirectionally and emits an edit sequence autoregressively. The sequence consists of a deletion phase followed by an addition phase. We propose the novel formulation of the deletion head which picks faces to delete autoregressively and the addition head that selects vertices to form a face to add. The addition head also adds vertices when needed. Untouched faces pass through unchanged, so the corrected mesh keeps the vertices and the layout of the draft. The draft enters as a prefix and only the edits are generated. A correction therefore takes far fewer decoding steps than generating every face again with an autoregressive mesh generator. We benchmark MeshRepair against repair tools on scripted defects and on held-out drafts from two mesh generators. Across both scripted defects and generated drafts, MeshRepair improves the quad-equivalent face F1, removes non-manifold faces, and cleans up the topology.

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

Artisan mesh generation has advanced quickly. Autoregressive models serialize faces into discrete sequences and learn the layouts that artists prefer (Nash et al., 2020; Siddiqui et al., 2024; Chen et al., 2025a;b; Weng et al., 2025; Tang et al., 2025; Lionar et al., 2025; Zhao et al., 2025; Xu et al., 2026; Wang et al., 2026a). Latent models decode a whole mesh in parallel and reach more complex shapes (Shen et al., 2024; Zhao et al., 2026; Long et al., 2026; Li et al., 2026; Wang et al., 2026b; Anonymous, 2026). Structured-latent generators reach high geometric fidelity but output dense surfaces rather than an artist layout (Xiang et al., 2025; 2026; Lai et al., 2026). Their outputs still contain holes, stray faces, duplicated faces, and edges shared by more than two faces. Such defects are local, and the rest of the mesh is what the artist would have modeled. The usual remedy is a new draw from the generator but errors may reappear elsewhere. The topology also changes with every draw, so a layout the user already preferred is lost.

Existing repair tools are rule based. Hole filling in trimesh (Dawson-Haggerty et al., 2019), PyMeshLab (Muntoni & Cignoni, 2021), and Blender (Blender Online Community, 2025) closes every boundary loop, although many boundaries in a draft are intended openings. Non-manifold face removal deletes every face at an edge shared by more than two faces, although usually only one of them is wrong. MeshFix (Attene, 2010) rebuilds a watertight surface and discards the face layout. None of them can decide which faces the artist intended, because that decision needs the surrounding surface as context and a learned prior of artist layout. MeshPad (Li et al., 2025) is a learned editor, but requires a user-marked region and a target sketch, so it cannot run unattended.

We introduce the task of artisan topology correction and propose MeshRepair, the first model to perform it. The task is to correct the topology of a generated mesh in place toward the layout an artist would make. It edits the existing draft rather than regenerating the mesh from scratch. The input is the draft. The output is an edit sequence, i.e., the faces to delete and the faces to add. The target is the artist mesh, or reference, while treating the two triangulations of each near-planar quadrilateral as equivalent because the export-time diagonal is arbitrary. We scope the task to topology. MeshRepair keeps the draft vertices and adds a vertex only where one is missing.

MeshRepair is a face-level prefix-LM. The draft is canonicalized, i.e., its vertices are quantized to a voxel grid and its faces are sorted, and a small network turns the three vertices of each face into one token (Wang et al., 2026a). A prefix-LM fits the task because the draft is given in full and only the edits are unknown. The face tokens form a prefix that is read bidirectionally, so a missing face can be recognized from the surface around it. Only the edit sequence after the prefix is decoded autoregressively, first all deletions and then all additions, which is far cheaper than regenerating every face. At each deletion step the deletion head ranks the draft faces and removes the top one, until it emits the end-of-del token. At each addition step the addition head names the three vertices of a new face among the draft vertices. Selecting the NEW token instead predicts the coordinates of a vertex that the draft lacks. We train MeshRepair with teacher forcing. At inference the finished sequence is applied to the draft, and untouched faces pass through unchanged. By reading the mesh as a prefix, MeshRepair corrects topology errors automatically, without external inputs such as an editing region, a sketch, a point cloud, or other guidance.

We evaluate MeshRepair on scripted defects, which a program inserts into artist meshes, and on held-out drafts from two mesh generators. On both, MeshRepair raises the quad-equivalent face F1, removes nearly all non-manifold edges, and moves vertex valence toward the reference. Repair tools leave the face F1 flat or lower it because difficult cases require identifying which of several faces is wrong—a decision that requires learning a prior over artist-authored mesh topology, rather than indiscriminately closing holes or deleting all candidate faces.

Contributions.

- We introduce artisan topology correction, formulated as predicting topology-editing operations for a draft with defects such as holes, stray faces, and duplicated faces.
- We propose MeshRepair, an efficient prefix-LM with a one-face-one-token prefix and specialized deletion and addition heads, requiring only a fraction of the decoding cost of full-mesh regeneration.
- We benchmark MeshRepair on scripted defects and on generated drafts, and show that it recovers the artist layout in both cases where repair tools fail.

2 Related Work

Mesh generation. Autoregressive mesh models serialize connectivity into discrete sequences: PolyGen uses pointer factorizations (Nash et al., 2020), MeshGPT and MeshAnything generate learned face tokens (Siddiqui et al., 2024; Chen et al., 2025a;b), BPT, TreeMeshGPT, and MeshMosaic scale to larger meshes (Weng et al., 2025; Lionar et al., 2025; Xu et al., 2026), and EdgeRunner, DeepMesh, and FACE improve scalability and tokenization (Tang et al., 2025; Zhao et al., 2025; Wang et al., 2026a). They represent connectivity exactly but generate a whole mesh sequentially. Latent alternatives such as SpaceMesh, LATO, LATO2, MeshFlow, and MeshCarve encode topology for parallel decoding (Shen et al., 2024; Zhao et al., 2026; Long et al., 2026; Li et al., 2026; Anonymous, 2026), and Nexus diffuses vertices and faces directly (Wang et al., 2026b). Structured-latent TRELLIS and LATTICE achieve high fidelity but produce dense, non-artist surfaces (Xiang et al., 2025; 2026; Lai et al., 2026). Using the draft as a prefix and decoding only edits, we retain efficiency and explicit connectivity without voxel aggregation (Long et al., 2026).

Repair and editing. MeshFix enforces watertightness by triangulating every open boundary loop, including openings intentionally left by the artist (Attene, 2010). Besides MeshFix, common 3D modeling libraries also offer standard tools to patch holes and remove stray edges but only target simple defects without any prior of artisan mesh topology. Specifically, trimesh (Dawson-Haggerty et al., 2019) fills a hole only when its boundary is a triangle or a quadrilateral and it does not patch holes that are larger. PyMeshLab (Muntoni & Cignoni, 2021) closes holes of any size, but it refuses non-manifold input, so the non-manifold faces must be removed first. Additionally, PyMeshLab can also wrongly detect holes that are in fact proper structures. Non-manifold face removal deletes faces until every edge borders at most two faces but never adds faces, and so opens new holes. Blender (Blender Online Community, 2025) merges duplicate vertices, removes loose and degenerate geometry, and fills holes. They close every boundary loop, including openings the artist left on purpose, and triangulate it without regard to the artist layout. To this end, several learned methods were proposed. MeshPad (Li et al., 2025) uses separate sketch-conditioned networks for deletion and addition. It is an interactive editor and requires user input. SpaceMesh (Shen et al., 2024) regenerates the connectivity of a user-marked region, conditioned on a point cloud. In contrast, MeshRepair requires no user-specified region or sketch and operates fully automatically: a single prefix-LM predicts face deletions and additions from the draft while preserving its unedited connectivity, thereby reducing user effort for the many small, scattered artifacts common in generated meshes.

3 Method

3.1 Problem Formulation

We treat topology correction as editing a canonical mesh. Let $M = \{f_1, \dots, f_F\}$ be the canonical draft mesh. Its vertices are quantized to a 512^3 grid, and each triangular face f_i is represented by the nine integer coordinates of its three vertices, sorted within the face. Duplicate faces are removed, and the remaining faces are sorted. Let $M^\star$ be the canonical reference mesh. MeshRepair predicts an *edit sequence* $E = (D, A)$, where D indexes draft faces to delete and A holds faces to add. The teacher edit sequence $(D^\star, A^\star)$ is the difference between draft and reference. $D^\star$ indexes the draft faces absent from the reference, and $A^\star$ holds the reference faces absent from the draft. Removing the faces indexed by D from M and adding the faces in A gives the repaired mesh $\mathrm{apply}(M, E)$:

$$D^\star = \{i : f_i \notin M^\star\}, \qquad A^\star = \{f \in M^\star : f \notin M\}, \qquad \mathrm{apply}(M, E) = \{f_i : i \notin D\} \cup A.$$

Diagonal flips in coplanar quads are excluded from teacher edits (Section 3.6; Figure 2).

3.2 One-Face-One-Token Prefix

Inspired by (Wang et al., 2026a), each face becomes one *face token*: each of its nine coordinates is embedded together with its position among the nine, and a two-layer MLP pools the nine embeddings. The face tokens of the draft form the prefix, and the edit sequence follows it as

$$\underbrace{f_1 \cdots f_F}_{\text{prefix}} \langle \text{BOS} \rangle \underbrace{f_{\delta_1} \cdots f_{\delta_{|D|}}}_{\text{deletion steps}} \langle \text{EOD} \rangle \underbrace{a_1 \cdots a_{|A|}}_{\text{addition steps}},$$

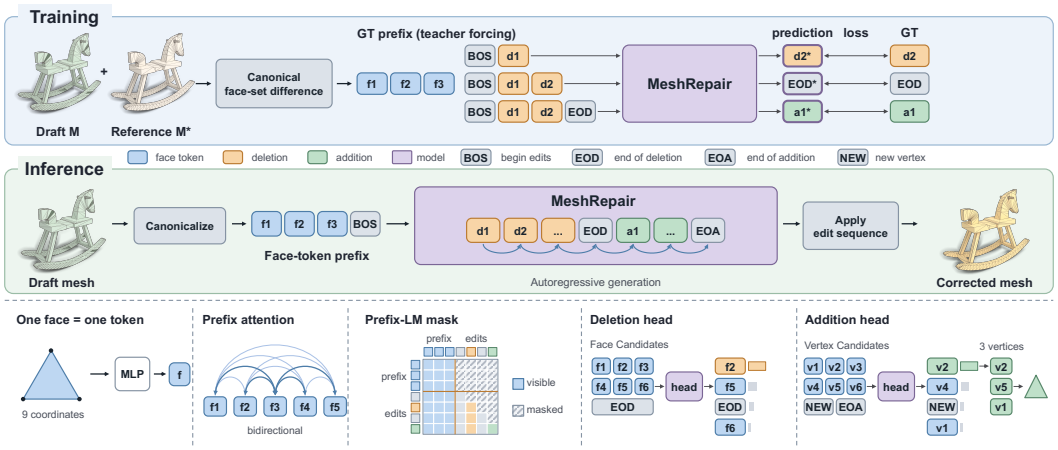


Figure 2: **Overview of MeshRepair.** Training (top): MeshRepair receives the draft prefix and the shifted ground-truth edit sequence under a causal mask, so predictions for all edit positions are computed in parallel. The loss compares each prediction (marked $*$) with its ground-truth next token (teacher forcing). Inference (middle): MeshRepair reads the canonicalized draft as a prefix, generates the deletion phase and then the addition phase autoregressively, and applies the result to the draft. Bottom: face tokenization, bidirectional prefix attention, the prefix-LM mask, and candidate ranking by the two heads (detailed for addition in Algorithm 1).

where $\delta_1 < \cdots < \delta_{|D|}$ are the deleted face indices and the added faces follow the canonical order. The begin-of-edits token $\langle \text{BOS} \rangle$ and the deletion steps predict the next deleted face or the end-of-del token $\langle \text{EOD} \rangle$. The emitted $\langle \text{EOD} \rangle$ is fed back, and with the addition steps predicts the next added face or the end-of-add token $\langle \text{EOA} \rangle$. Both deletion and addition steps receive the corresponding face-token and face-index embeddings. Segment embeddings mark the prefix and the two phases, and one RoPE counter indexes all positions, so a prefix position is a face index.

3.3 Prefix-LM Attention

The prefix is encoded *bidirectionally*, so every prefix token attends to every other prefix token. A hole is a *missing* face and is evident only against the surrounding surface. A causal encoding of the prefix could not attend to the faces that ought to bracket a defect. The edit tokens, i.e., all tokens after the prefix, attend to the entire prefix and causally to earlier edit tokens. Training uses a per-sample block-diagonal prefix-LM mask via FlexAttention, which packs variable-length samples with little padding. Inference uses a KV cache to prefill the prefix bidirectionally and decode edits causally, matching the attention pattern used during training. The backbone is a decoder-only transformer (Vaswani et al., 2017) (RMSNorm, RoPE, SwiGLU) with GPT-NeoX residual-projection initialization.

3.4 Deletion Head

The deletion head is a pointer over the prefix faces (Vinyals et al., 2015). For prefix face i we form a key from its final-layer hidden state h_i, its face token e_i, and a face-index embedding $P(i)$. The query is the hidden state g_t of the current deletion step:

$$k_i = W_k h_i + W_e e_i + P(i), \quad q_t = W_q g_t, \quad \ell_{t,i} = \frac{\langle q_t,\, k_i \rangle}{\sqrt{d_p}},$$

where d_p is the key dimension. The softmax runs over the F faces plus a learned key s for $\langle \text{EOD} \rangle$ with logit $\ell_{t,\varnothing} = \langle q_t, s \rangle / \sqrt{d_p}$. Selecting it ends the deletion phase, and a draft without deletions selects it first. Deletions are emitted in ascending index order, so we apply a *monotonic mask*. At each step only faces indexed above the previously deleted face and $\langle \text{EOD} \rangle$ are allowed. Training and decoding use the same mask. Feeding the face token of each deleted face rather than only its index lets the addition phase see which faces were removed.

Algorithm 1 Addition phase of the addition head.

$\mathcal{C} \leftarrow$ vertices of the draft
loop
  $k_v \leftarrow \mathrm{Key}(v, E_{\mathrm{bnd}}[b(v)])$ for $v \in \mathcal{C}$ *# coordinate and boundary embeddings*
  for $j = 1, 2, 3$ **do**
    $\mathcal{A}_j \leftarrow (\mathcal{C} \setminus \{c_{<j}\}) \cup \{\text{NEW}\}$; add EOA if $j = 1$ *# allowed choices*
    $c_j \leftarrow \arg\max_{v \in \mathcal{A}_j} s_j(v)$ *# s_j selects vertex j*
    if $c_j =$ EOA **then return**
    if $c_j =$ NEW **then**
      for $a = z, y, x$ **do**
        $u_a \leftarrow \arg\max_i f_a(i)$ *# f_a scores the 512 values of coordinate a*
      end for
      $c_j \leftarrow (u_z, u_y, u_x)$
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{c_j\}$ *# make the new vertex selectable*
    end if
  end for
  emit the face (c_1, c_2, c_3) in canonical order
end loop

3.5 Addition Head

For each face addition at step t, the addition head constructs a triangular face by selecting its three vertices autoregressively. Let $\mathcal{C}$ denote the current vertex pool, containing draft vertices and vertices generated by earlier additions. For vertex j of the face, the candidates are the vertices in $\mathcal{C}$ and NEW; selecting NEW triggers the prediction of a new vertex. A query computed from the hidden state g_t at face-addition step t and the previously selected vertices $c_{<j} = (c_1, \ldots, c_{j-1})$ is matched against each candidate key to produce the selection score $s_j(v)$. Each candidate vertex key combines embeddings of its quantized coordinates with a learned embedding $E_{\mathrm{bnd}}[b(v)]$ of the binary flag $b(v)$ indicating whether the vertex lies on an open mesh boundary; NEW and EOA use learned keys. We use boundary flags to indicate whether a vertex is on boundaries, providing extra information to the network. For a new vertex, $f_a(i)$ scores coordinate value i conditioned on g_t, $c_{<j}$, and the previously predicted coordinates $u_{<a}$. Algorithm 1 summarizes the addition phase. During both training and decoding, vertices already chosen for the current face are masked out to prevent duplicate vertices. New vertices enter $\mathcal{C}$ and remain selectable for later faces.

3.6 Training and Inference

Training. Artist meshes often contain many quadrilaterals, whose export-time triangulation uses an arbitrary diagonal. We therefore define the target topology up to that choice. When two deleted faces and two added faces split the same four vertices along opposite diagonals, and the four vertices are coplanar within one voxel, the four faces are dropped from the teacher edit sequence. Both triangulations describe the same surface, so MeshRepair is taught neither diagonal, and the evaluation scores face sets under the same equivalence (Section 4.1). Training mixes draft/reference pairs, pairs with part of their edits already applied, and a few identity pairs with an empty edit sequence, which teach MeshRepair not to edit a correct mesh. Patch hole punching removes further faces in connected patches, which supplies missing faces and new vertices for the addition head. The loss $\mathcal{L} = \mathcal{L}_{\mathrm{del}} + \mathcal{L}_{\mathrm{add}}$ sums the teacher-forced cross-entropies of the two heads, each averaged over its own decisions. MeshRepair has about 2.08 billion parameters. The base model is trained for $227{,}000$ steps, and the main model fine-tunes it for $14{,}000$ steps on paired generated drafts (Section 4.2). Appendix C gives implementation details.

Inference. MeshRepair prefills the draft, decodes the edit sequence greedily under the same masks as in training, and applies it to the draft once. Winding is restored for export. A few decoding safeguards with no training counterpart, such as caps on the number of edits, are listed in Appendix C.

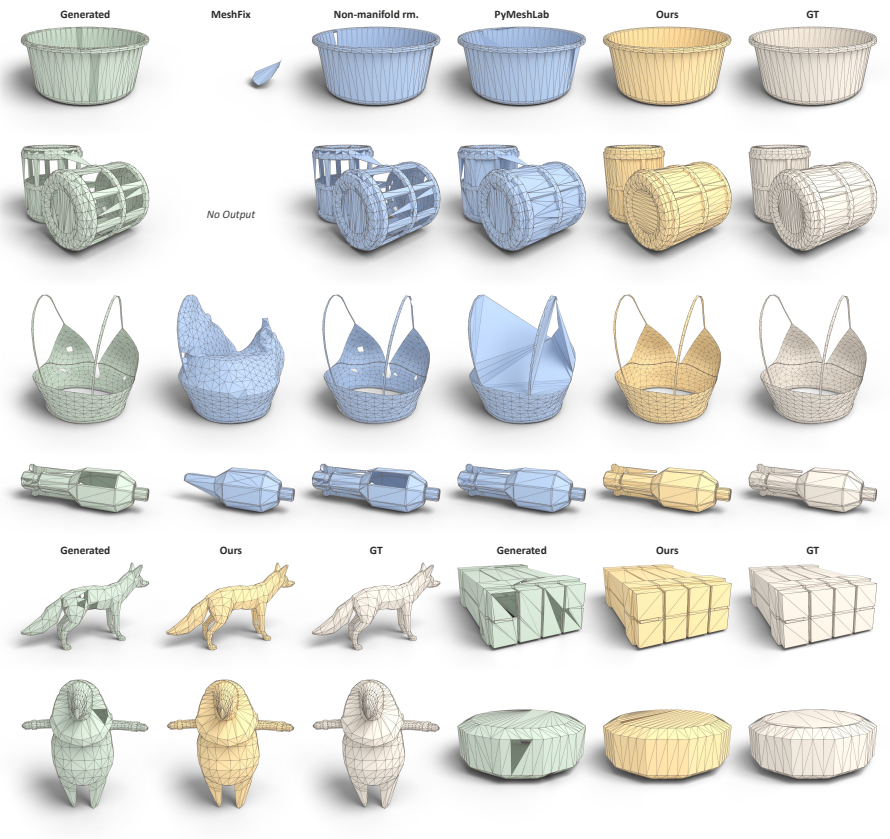


Figure 3: **Selected qualitative corrections.** Top four rows: each generated draft is processed by three repair tools (MeshFix, non-manifold face removal, and PyMeshLab hole closing) and by MeshRepair, next to the artist reference (GT). Bottom two rows: two drafts per row, each shown as generated draft, MeshRepair's correction, and GT. Views and scale are shared within each row. These selected examples illustrate local topology correction. Aggregate performance is reported in Table 1.

4 Experiments

4.1 Evaluation Protocol

We evaluate topology correction using face-level precision, recall, and F1 rather than relying solely on surface-distance metrics, since the target defects primarily alter face connectivity while leaving vertex positions largely unchanged. Every mesh is canonicalized as in Section 3.1 before scoring, so duplicate faces are removed first.

Quad-equivalent face set. Artist meshes often contain many quadrilaterals, whose export-time triangulation uses an arbitrary diagonal. Both diagonals describe the same surface and the same modeling intent, so requiring the exact triangles would penalize a correct edit for a choice the artist never made. We therefore identify the two triangulations of a near-planar quadrilateral, a rule we call the quad equivalence. Two faces sharing an edge match the opposite split of the same four vertices when the fourth vertex lies within $\epsilon = 1$ voxel of the plane through the other three. No other difference counts as a match, so a missing, extra, or displaced face remains an error. Face precision and recall are the fractions of output and reference faces matched under this rule, and the face F1 combines them. Δ is the gain in face F1 over the unedited draft. The matching is greedy and bounded as in Appendix C. On a draft where any compared method leaves more than $2{,}048$ unmatched output faces, every method is scored by exact matching instead.

Edge F1. Edge F1 compares undirected edges after snapping each output vertex to a reference vertex in the same voxel or in one of its six face-adjacent voxels. Edges that touch a vertex without a reference counterpart are left out, so the score measures connectivity among shared vertices.

Edit accuracy. Deletion and addition precision and recall ask whether a method made the right edits. The required edits are the exact set differences between draft and reference without the quad equivalence. The edits made are the set differences between draft and output, so the same definition applies to methods that emit no edit sequence. A method that makes no edit of a kind on a pair scores zero precision and recall for it on that pair.

Aggregation. All scores are computed per draft/reference pair and macro-averaged over the stated set of pairs. A reported F1 is therefore the mean of per-pair F1 values rather than the harmonic mean of the averaged precision and recall. Surface distances are reported in Appendix A. Because MeshRepair keeps the draft vertices, surface distances change little and mainly show whether a method degrades the geometry.

4.2 Setup

Training data. Training has two stages. The base model learns from reconstruction pairs, in which a reference mesh is encoded and decoded by a mesh VAE at a continuous range of noise levels. The decoded mesh is the draft, and patch hole punching adds missing faces on top (Section 3.6). The main model then fine-tunes on generated drafts mixed with the reconstruction pairs. MeshCarve (Anonymous, 2026) and LATO2 (Long et al., 2026) generate these drafts as connectivity on the vertices of training-split reference meshes. In the generated pairs the draft and reference share one vertex set, so every label is a pure connectivity edit.

Evaluation data. We evaluate on two kinds of drafts. The scripted-defect set corrupts held-out artist meshes with two kinds of scripted damage, holes and over-connections: connected patches of faces are removed, and spurious faces are inserted between existing vertices. The generated set uses held-out Objaverse shapes. For each shape, MeshCarve and LATO2 generate connectivity on the reference vertices, and the reference is the quantized artist mesh. Every method is scored on the same drafts. Because the vertices are given, these drafts isolate connectivity errors from vertex placement. Both generators also supply the fine-tuning pairs, so this set tests held-out shapes from generators seen in training.

Baselines. We compare against repair tools that need no reference, applied to the same drafts. MeshFix (Attene, 2010) rebuilds a manifold surface. Non-manifold face removal deletes faces until no edge is shared by more than two faces, and never adds any. Hole filling in trimesh (Dawson-Haggerty et al., 2019) and hole closing in PyMeshLab (Muntoni & Cignoni, 2021) add faces across boundary loops. PyMeshLab first receives the non-manifold removal because it refuses non-manifold input. Blender cleanup (Blender Online Community, 2025) merges duplicate vertices, removes loose and degenerate geometry, and then fills holes.

4.3 Main Results

Table 1 shows that the repair tools do not recover the artist layout. Hole filling in trimesh, PyMeshLab, and Blender adds faces across boundary loops. Most of the added faces are absent from the reference, so face F1 stays flat or drops on both generators. A boundary in a generated draft is often an intended opening or the side effect of a wrong face elsewhere. A filler that closes every loop cannot tell the two apart. Non-manifold face removal raises precision by deleting faces at edges shared by more than two faces. It cannot tell which of those faces is wrong, so it loses about as much recall as it gains precision. MeshFix rebuilds the surface altogether, which discards the draft connectivity and fails on part of the drafts.

MeshRepair is the only method that raises face F1 on both generators and on the scripted defects. The gain is led by precision, since MeshRepair deletes faces the reference does not contain. Its deletion precision is the highest in every block. Unlike non-manifold removal, these deletions are chosen from the whole draft, so face recall is preserved. The gain is larger on LATO2 than on MeshCarve. LATO2 drafts start further from the reference, while MeshCarve drafts are already close and leave less to correct. Edge F1 follows the same ordering, with a clear margin on LATO2 and a small one

Table 1: **Topology and editing results.** Macro averages under the protocol in Section 4.1. †MeshFix rebuilds the surface, so it deletes nearly every draft face and attains high deletion recall by over-deleting. MeshFix rows average only pairs for which it returns a mesh. Every reported value is shaded on a light blue sequential scale, normalized independently within each data block and column. MeshFix entries are shown in gray and excluded from heatmap normalization because it reconstructs rather than locally edits the input mesh.

	Quad-equivalent face set				Edge	Deletion			Addition		
Method	P↑	R↑	F1↑	Δ	F1↑	P↑	R↑	F1↑	P↑	R↑	F1↑
Scripted defects											
Draft (unedited)	.901	.901	.901	–	.922	0	0	0	0	0	0
+ MeshFix†	.601	.071	.118	−.783	.141	.107	.999	.193	.101	.053	.059
+ trimesh hole fill	.890	.916	.902	+.002	.921	0	0	0	.502	.133	.203
+ Non-manifold face removal	.929	.854	.890	−.011	.931	.427	.340	.375	0	0	0
+ PyMeshLab close holes	.791	.889	.837	−.064	.902	.442	.336	.379	.147	.277	.190
+ Blender cleanup + hole fill	.867	.936	.900	−.001	.919	.256	.011	.020	.423	.324	.357
+ MeshRepair (ours)	.990	.973	.981	+.081	.989	.985	.988	.984	.828	.702	.724
Generated drafts, MeshCarve											
Draft (unedited)	.838	.859	.848	–	.829	0	0	0	0	0	0
+ MeshFix†	.726	.256	.327	−.521	.333	.467	.727	.538	.098	.009	.016
+ trimesh hole fill	.834	.863	.847	.000	.829	0	0	0	.237	.005	.009
+ Non-manifold face removal	.860	.842	.851	+.003	.829	.602	.070	.117	0	0	0
+ PyMeshLab close holes	.816	.850	.832	−.016	.818	.603	.067	.111	.097	.010	.016
+ Blender cleanup + hole fill	.818	.863	.839	−.009	.824	.319	.008	.015	.161	.008	.014
+ MeshRepair (ours)	.873	.865	.868	+.020	.831	.644	.223	.305	.341	.093	.130
Generated drafts, LATO2											
Draft (unedited)	.268	.334	.294	–	.764	0	0	0	0	0	0
+ MeshFix†	.245	.077	.101	−.193	.274	.803	.788	.771	.026	.001	.003
+ trimesh hole fill	.268	.334	.294	.000	.764	0	0	0	.025	.000	.001
+ Non-manifold face removal	.297	.292	.294	.000	.764	.907	.210	.311	0	0	0
+ PyMeshLab close holes	.280	.299	.289	−.006	.754	.910	.199	.300	.022	.003	.004
+ Blender cleanup + hole fill	.264	.334	.292	−.002	.760	.460	.009	.017	.028	.001	.002
+ MeshRepair (ours)	.346	.336	.340	+.046	.784	.916	.352	.484	.149	.024	.038

on MeshCarve. Addition remains the harder half of the task, since added faces are fewer and less precise than deletions. This matches the analysis in Section 4.4. On the scripted defects the same pattern appears with a larger gain. MeshRepair deletes nearly all inserted faces and recovers most of the removed ones, while the repair tools leave face F1 flat or lower it. The scripted defects emulate two error classes that mesh repair surveys document on polygonal models: holes left by surface reconstruction, and redundant faces that produce non-manifold edges (Ju, 2009; Attene et al., 2013). Appendix A reports surface distances. MeshRepair leaves MeshCarve drafts unchanged within per-pair sampling noise and moves LATO2 drafts closer to the reference. MeshFix, PyMeshLab, and Blender move the surface away from the reference.

Qualitative results. Figure 3 shows the same behavior on individual drafts. MeshFix returns a remeshed fragment or no mesh at all. Non-manifold removal opens new boundaries where it deletes faces, and hole closing covers openings with fan triangulations. MeshRepair removes stray and duplicated faces and fills missing ones with faces that follow the artist layout, while the rest of the draft passes through unchanged.

4.4 Empirical Analysis

Deletion vs. addition. In every block of Table 1, the deletions of MeshRepair are more precise and more complete than its additions. On LATO2 drafts it deletes at a precision of .916, while its additions reach a precision of .149 and a recall of .024. Deletion picks from a fixed set, the draft faces, and only has to decide when to stop. Addition has a much larger action space. It must pick three compatible vertices in order, decide whether a vertex is missing, and stop without repeating a

Table 2: **Ablations.** (a) Pointer vs. coordinate heads at a smaller scale (340M parameters, 30k steps, 200 reconstruction pairs). (b) Face F1 after repeated correction rounds.

(a) Pointer vs. coordinate heads Deletion / addition head	Del. F1	Add. F1
pointer / pointer	.662	.015
coordinates / pointer	.091	.003
pointer / coordinates	.669	.002

(b) Correction rounds Rounds	MeshCarve	LATO2
0 (draft)	.848	.294
1	.868	.340
2	.869	.341
3	.868	.341

Table 3: **Correction by defect size.** Generated drafts binned by the edit ratio r, the number of edits the reference requires under the quad equivalence (Section 4.1) divided by its face count. LATO2 is pooled over all drafts. Face F1 of the draft, its gain from MeshRepair, the share of drafts that improve or get worse (%), and deletion and addition precision and recall.

		Face			Deletion		Addition	
Drafts by r	n	F1	Δ	$\uparrow/\downarrow$	P	R	P	R
MeshCarve, < 0.02	70	.998	$-.005$	24/9	.180	.024	.124	.002
0.02–0.05	43	.983	$+.004$	70/16	.510	.050	.357	.019
0.05–0.15	170	.950	$+.012$	71/23	.656	.128	.382	.056
0.15–0.3	279	.887	$+.025$	79/18	.714	.193	.351	.075
0.3–0.5	195	.799	$+.026$	69/29	.679	.302	.360	.136
0.5–1	147	.692	$+.035$	69/31	.696	.426	.350	.187
LATO2, all	938	.294	$+.046$	83/15	.916	.352	.149	.024

face. Addition also faces a stricter success criterion: when an added face requires a new vertex, the predicted vertex must match the reference voxel exactly for the face to receive credit. Nevertheless, the qualitative results in Figure 3 show that MeshRepair produces plausible additions that repair local defects. The gain in face F1 therefore comes mostly from deletions, and missing faces remain the harder half of the task.

Correction by defect size. Table 3 bins MeshCarve drafts with $r < 1$ by the edit ratio r, the number of required edits divided by the reference face count. On MeshCarve the gain in face F1 grows with r over the reported range, from $+.004$ at r between 0.02 and 0.05 to $+.035$ at r between 0.5 and 1, while deletion precision stays above $.5$ for $r \geq 0.02$. Almost every LATO2 draft has $r \geq 1$, and 22% of its edges are non-manifold (Appendix B), yet MeshRepair still raises its face F1 by $.046$. Nearly correct MeshCarve drafts show mild over-correction ($-.005$ at $r < 0.02$).

Ablations. Table 2(a) replaces each pointer head with a head that emits face coordinates, at a smaller scale. Coordinates for deletion drop deletion F1 from $.662$ to $.091$, since naming a face by its coordinates is far harder than pointing at it. Coordinates for addition drop addition F1 from $.015$ to $.002$ and leave deletion unchanged. Table 2(b) repeats the correction on its own output. One round captures nearly all of the benefit. Further rounds change face F1 by at most $.001$ on either generator, so they are not worth their cost.

5 Limitations

MeshRepair corrects topology only. It preserves existing vertex coordinates, so coordinate drift, misplaced parts, and coarse shape errors remain the generator's responsibility. It also relies on enough local evidence in the draft; extremely sparse, disconnected, or distorted inputs may exceed its capacity. Addition and new-vertex recovery remain harder than deletion, and the model can delete correct faces or leave holes, raising boundary edges on LATO2 (Appendix B). It therefore guarantees neither watertightness, manifoldness, nor exact reference recovery, although its edit sequence can be restricted to a selected region before conventional cleanup.

6 Conclusion

We presented MeshRepair, a face-level prefix-LM that corrects the topology of generated artisan meshes in place. It reads a draft mesh as a prefix to generate deletion and addition sequences, leaving most faces untouched. It requires no other inputs and costs a fraction of full generation. On scripted defects and on drafts from two mesh generators, MeshRepair raises the quad-equivalent face F1 where repair tools leave it flat or lower it. It also removes nearly all non-manifold edges. Its precise deletions remove erroneous faces, while plausible face additions restore missing connectivity and improve the corrected topology. We see MeshRepair as a plug-in stage after any mesh generator that turns an almost right draft into an artist-ready mesh.

AI USE STATEMENT

We used a generative AI coding assistant. It helped with writing and language polishing of the manuscript. It helped with the code of the evaluation, rendering and figure-generation scripts. We did not use generative AI to generate research ideas, to design the method, or to produce any experimental result. All AI-assisted text was revised by the authors, all AI-assisted code was verified and tested by the authors, and every number reported was computed by our own evaluation pipeline. We take responsibility for the final content of this work.

REFERENCES

Anonymous. MeshCarve: Artisan Mesh Generation with Flow Matching in Compact Geometry and Topology Latent Space. In *Submitted to The Fifteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=LfKYZHX5GT. under review.

Marco Attene. A Lightweight Approach to Repairing Digitized Polygon Meshes. *The Visual Computer*, 26(11):1393–1406, 2010.

Marco Attene, Marcel Campen, and Leif Kobbelt. Polygon Mesh Repairing: An Application Perspective. *ACM Computing Surveys*, 45(2):15:1–15:33, 2013. doi: 10.1145/2431211.2431214.

Blender Online Community. Blender, a 3D modelling and rendering package. https://www.blender.org, 2025.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. MeshAnything: Artist-Created Mesh Generation with Autoregressive Transformers. In *International Conference on Learning Representations (ICLR)*, 2025a.

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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025b.

Dawson-Haggerty et al. trimesh. https://trimesh.org/, 2019.

Tao Ju. Fixing Geometric Errors on Polygonal Models: A Survey. *Journal of Computer Science and Technology*, 24(1):19–29, 2009. doi: 10.1007/s11390-009-9206-7.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. LATTICE: Democratize High-Fidelity 3D Generation at Scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Haoxuan Li, Ziya Erkoç, Lei Li, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshPad: Interactive Sketch-Conditioned Artist-Reminiscent Mesh Generation and Editing. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Weiyu Li, Antoine Toisoul, Tom Monnier, Roman Shapovalov, Rakesh Ranjan, Ping Tan, and Andrea Vedaldi. MeshFlow: Efficient Artistic Mesh Generation via MeshVAE and Flow-based Diffusion Transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Stefan Lionar, Jiabin Liang, and Gim Hee Lee. TreeMeshGPT: Artistic Mesh Generation with Autoregressive Tree Sequencing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26608–26617, 2025.

Hang Long, Tianhao Zhao, Junkai Lin, Youjia Zhang, Huipeng Guo, Rendong Liang, Jiale Xu, Jozef Hladký, Matthias Nießner, Yuanming Hu, and Wei Yang. LATO.2: Factorized 3D Mesh Generation with Vertex and Topology Flow, 2026.

Alessandro Muntoni and Paolo Cignoni. PyMeshLab, 2021.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. PolyGen: An Autoregressive Generative Model of 3D Meshes. In *International Conference on Machine Learning (ICML)*, 2020.

Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. SpaceMesh: A Continuous Representation for Learning Manifold Surface Meshes. In *SIGGRAPH Asia Conference Papers*, 2024.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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. In *International Conference on Learning Representations (ICLR)*, 2025.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer Networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

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. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a.

Hanxiao Wang, Ying-Tian Liu, Yuan-Chen Guo, Qi-Yuan Feng, Zi-Xin Zou, Ding Liang, Biao Zhang, and Yan-Pei Cao. Nexus: Native Mesh Generation with Diffusion. *ACM Transactions on Graphics (TOG)*, 45(4), 2026b. doi: 10.1145/3811344.

Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, Tong Zhang, Shenghua Gao, and C. L. Philip Chen. Scaling Mesh Generation via Compressive Tokenization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11093–11103, 2025.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D Latents for Scalable and Versatile 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21469–21480, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and Compact Structured Latents for 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14419–14429, 2026.

Rui Xu, Tianyang Xue, Qiujie Dong, Le Wan, Zhe Zhu, Peng Li, Zhiyang Dou, Cheng Lin, Shiqing Xin, Yuan Liu, Wenping Wang, and Taku Komura. MeshMosaic: Scaling Artist Mesh Generation via Local-to-Global Assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20003–20013, 2026.

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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Tianhao Zhao, Youjia Zhang, Hang Long, Jinshen Zhang, Wenbing Li, Yang Yang, Gongbo Zhang, Jozef Hladký, Matthias Nießner, and Wei Yang. LATO: 3D Mesh Flow Matching with Structured Topology Preserving Latents. In *International Conference on Machine Learning (ICML)*, 2026.