PAGA:PRIMITIVE ASSEMBLY FROM GENERATIVE ANCHORS

Anonymous authors
Paper under double-blind review



Figure 1: **Primitive assemblies generated by PAGA.** Colors distinguish primitive instances within each assembly. Primitive types include boxes, ellipsoids, cylinders, tori, etc.

ABSTRACT

Compact, editable primitive assemblies are important game assets, yet faithfully representing complex surfaces with few primitives remains challenging. Geometry-driven approximations often require many elements, while direct primitive generation can miss small parts or distort intersecting structures. We introduce **PAGA** (Primitive Assembly from Generative Anchors), which separates part discovery from precise geometry fitting. PAGA generates sparse geometric anchors to propose candidate parts, then grounds them on the original surface to select and fit primitives. Fitting visible surfaces while allowing interior completion improves fidelity; merging compatible primitives and removing redundant bodies under surface coverage checks maintains compactness. On two test sets of 2,000 near-primitive surfaces and 2,000 Objaverse meshes, PAGA reduces mean CD-L1 by approximately **55%** and **68%**, respectively, relative to PrimitiveAnything, while reducing average primitive counts by approximately **47%** and **69%**.

1 INTRODUCTION

Representing 3D objects with compact primitive assemblies is important for game asset generation and editable geometric proxies (Fig. 1). Unlike a dense mesh, an assembly describes a shape through simple elements whose position, size, and proportions can be changed with a few parameters. Yet editable parameters alone do not make an object easy to edit. **Compactness and clear** organization make an assembly easier to understand and modify, reducing the number of elements that users must inspect and manipulate. The task therefore has two goals: preserve the original surface faithfully while maintaining a compact, well-organized assembly.

Existing approaches pursue these goals through **geometric approximation** or **direct primitive generation**, but face different bottlenecks. Decomposition and fitting methods (Li et al., 2019; Liu et al., 2023) use the input geometry directly, but face a trade-off between surface fidelity and representation complexity: local approximations can fragment a part, whereas restricting the number of elements can sacrifice geometric detail. Direct generation methods such as PrimitiveAnything (Ye et al., 2025) learn to produce editable assemblies, but predicting complete primitives from an observed exterior remains ambiguous when parts overlap or are only partially exposed. Generated assemblies can consequently miss small components, distort local geometry, or introduce redundant elements. We seek to combine faithful surface fitting with the structural guidance of generation.

Our design is motivated by an asymmetry in the input: visible surface geometry is directly observed, whereas the hidden portions of overlapping primitives are not. Autoregressive generation conditioned on surface points and normals can describe this visible geometry without first committing to complete primitive bodies. However, generating detailed surfaces requires many tokens. This leads to our key

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

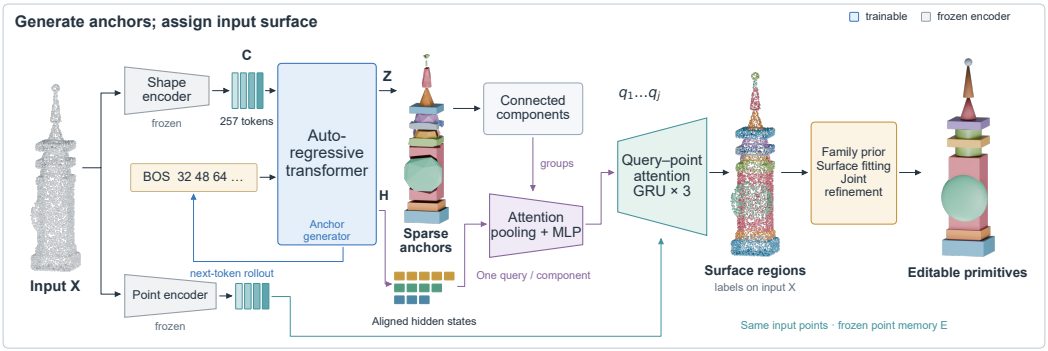

Figure 2: **Overview.** The same oriented points feed frozen shape and point encoders. Autoregression generates sparse anchors; the pooled hidden states of each connected component guide assignment of the input surface, followed by family-guided fitting and joint refinement.

insight: part discovery and precise primitive fitting need not use the same representation. We generate sparse geometric anchors to propose candidate parts, then use them to identify corresponding regions on the original surface. The anchors guide where to look, while the original surface provides the evidence for choosing and fitting the final primitives. The anchors therefore need not encode every geometric detail. To give candidate regions clear boundaries, we cut the sparse training surfaces along primitive contacts, teaching the model to generate disconnected anchors. These cuts separate visible surface regions, not the object's interior, and leave the dense input surface intact.

We build on this insight with PAGA (Fig. 2). Given a mesh, PAGA first generates sparse anchors that propose a variable number of candidate parts and their rough locations. It then associates each anchor with its supporting region on the original surface. A type predictor suggests candidate primitive families, which are verified through geometric fitting. Fitting encourages each primitive to explain its assigned surface region while penalizing unsupported outward protrusions; portions lying inside the object are allowed, accommodating the hidden geometry of overlapping bodies. Finally, PAGA merges compatible primitives and removes redundant bodies when surface coverage is preserved. The result is a compact assembly fitted to the input shape; the anchors serve only as an intermediate guide.

Our contributions are threefold. First, we introduce **a generative-anchor framework** that connects part discovery with surface-grounded primitive fitting. Second, we combine **fitting that allows interior completion with assembly-level consolidatio**n to preserve surface fidelity while reducing fragmentation and redundancy. Third, we construct a **training representation that pairs sparse, contact-separated surface anchors with intact dense geometry**. On two test sets comprising 2,000 near-primitive surfaces and 2,000 Objaverse meshes, respectively, PAGA reduces mean CD-L1 by approximately 55% and 68% relative to PrimitiveAnything, while reducing average primitive counts by approximately 47% and 69%. These results show that better surface fidelity need not require a larger assembly.

2 Related Work

Primitive assembly and shape programs. There are numerous approaches to representing complex shapes with primitives, including primitive surface generation, constructive solid geometry (CSG) program generation, and solid primitive assembly generation. CADDreamer (Li et al., 2025) and CADKnitter (Le et al., 2026) represent surfaces with B-rep faces rather than assemblies of independently parameterized solid primitives. CSG-based methods construct solids through complex Boolean operations, including intersection and subtraction (Yu et al., 2024; Lin et al., 2026; Wang et al., 2026), unlike the compact, union-only primitive proxies considered here. Among geometry-conditioned primitive generators, PrimitiveAnything (Ye et al., 2025) is closest to our setting, but supports only cuboids, ellipsoids, and elliptical cylinders. In our evaluation, it produces larger surface errors and more primitives than PAGA (Table 2). PAGA supports a broader primitive catalog and combines surface-guided fitting with consolidation to achieve both fidelity and compactness.

Primitive decomposition and fitting. Convex decomposition methods (Luo et al., 2025; Yang et al., 2026) produce accurate hull-based approximations but do not enforce a predefined primitive

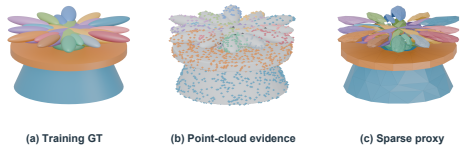

Figure 3: A training assembly, dense visible-surface evidence, and the shared sparse proxy. Colors show the source-primitive assignment.

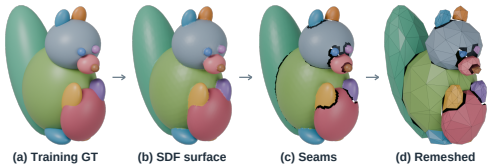

Figure 4: From a training assembly to a token-efficient proxy: visible exterior, contact seams, sparse remeshing.

catalog. Analytic fitting methods use parameterized bodies: Light-SQ (Wang et al., 2025) fits superquadrics with shape exponents, while SuperFit (Ganeshan et al., 2026) fits SuperFrusta with rounding, taper, and hollowing controls. Marching-Primitives (Liu et al., 2023) can fit hard primitives such as ellipsoids without these controls, but their shape range is limited and SDF grids can miss thin structures. PAGA fits a broader catalog of *hard primitives* to the original surface and consolidates them into compact, high-fidelity proxies.

Autoregressive mesh generation. MeshGPT (Siddiqui et al., 2024) and MeshAnything (Chen et al., 2025a) model meshes as autoregressive token sequences. MeshAnything V2 (Chen et al., 2025b) uses face adjacency to shorten sequences and preserve local continuity. Maintaining local continuity while generating low-density meshes aligns with our goal of token-efficient geometric anchors and motivates applying autoregressive mesh generation to primitive fitting. Rather than use a complete generated mesh as the final output, PAGA generates sparse geometric anchors to propose candidate parts, then uses the original surface to determine their precise primitive types and parameters.

3D segmentation. Query-based methods predict masks from point features (Schult et al., 2023; Kolodiazhnyi et al., 2024). Other approaches use multi-view masks (Tang et al., 2024), learned part features (Yang et al., 2024; Liu et al., 2025), or language-conditioned mask tokens (Huang et al., 2025; He et al., 2024). PAGA avoids the complexity of multi-view fusion and language-model inference. It adopts Slot Attention-inspired query updates (Locatello et al., 2020), but initializes queries from generated anchors and derives their number from the generated components. This separates part discovery from surface localization: the segmentation head only grounds existing part hypotheses on the input surface for primitive fitting.

3 Method

Overview. Fig. 2 is the architecture of the method. Let $X = \{(\mathbf{x}_i, \mathbf{n}_i)\}_{i=1}^{N}$ be oriented samples of an observed surface. We assemble a variable-size partition $\widehat{\mathcal{R}} = \{\widehat{R}_j\}_{j=1}^{J}$ and analytic primitives $\widehat{\mathcal{P}}$ whose assembled exterior explains X. Neither J nor the types are given.

3.1 Dual-resolution representation

Each training assembly is kept as two surfaces (Fig. 3). Assignment and fitting use the dense surface X, sampled from the intact visible exterior, so contact between neighboring parts remains. Autoregression does not target this dense surface: writing every sample as a token would spend the sequence on coordinates. Its target is a second, sparse mesh, cut along primitive contacts and covered by a few triangles per exposed region (Sec. 3.2). Only that sparse mesh is cut, so the decoded components separate neighboring parts, while X stays intact. Both encoders read the same samples under the same indices. Points whose ownership is uncertain remain in X and are not labeled as any part. The sparse mesh is the sequence target used in training, not the geometry returned by the model.

3.2 Constructing the proxy

If two adjacent faces share one complete edge, our program treats them as the same part. After coordinates are quantized to 128 bins, two solids that merely touch fall on the same vertices, so two parts become one query. Each training assembly is therefore turned into a proxy by the three steps in Fig. 4.

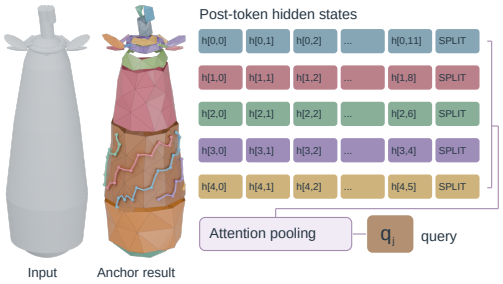

Figure 5: **From a generated component to a query.** Decoder states of one connected component, possibly spanning several strips, pool into $\mathbf{q}_j$. Traversal restarts are not part identities.

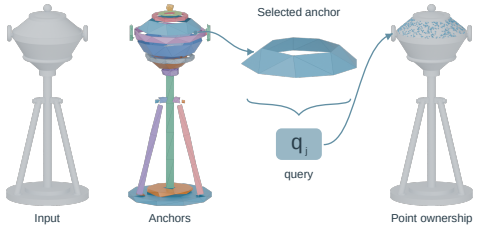

Figure 6: **Grounding on input points.** Query $\mathbf{q}_j$ finds its support on the dense observation. Colored points indicate ownership.

Visible exterior. Keep only the surface that the assembly exposes, and record which source primitive each piece comes from. The Boolean form is the exact surface of the assembled solids. The SDF form builds an SDF exterior from the same assembly and then perturbs it, so the surface leaves the exact primitive exterior and the generator is also trained on less regular inputs. A fully hidden surface is not a sequence target; the generator should propose exposed parts. Supervising interior vertices, or asking the decoder to emit non-surface points, made the ShapeVAE-conditioned sequence less stable. The encoder receives oriented samples of the exterior, and its regression target is that continuous visible surface.

Quantization-safe contact seams. Cut a seam only on the proxy, along where primitives meet: after quantization, neighboring parts no longer share a complete edge, while the sparse mesh still covers the exposed region. The checks used during remeshing are in Appendix B.

Sparse remeshing. Rebuild each separated region with a few triangles, keeping coverage and holes and controlling local scale, so tokens are spent on part structure rather than on dense coordinates.

Cuts do not modify X. The autoregressive sequence targets the proxy (the anchor), not precise geometry. Fitting that proxy directly misuses it (Sec. 4.3, Fig. 10). Sparse triangulation and quantization erase type cues, and small spheres and boxes are especially hard to tell apart. The contact gaps are constructed seams; a direct fit treats them as missing geometry and separates bodies that should meet. Assignment and fitting use the observation that was not cut.

The training assemblies are built from primitives, and the model can likewise run on non-primitive surfaces. The network being trained is a segmentation model: component queries are supervised to partition the observed surface, and that objective is sensitive to the junctions where parts meet rather than to a global primitive template. The perturbed SDF exteriors move the same supervision off the exact primitive surface. Adaptation trains hard cross-entropy on $0.5\,p_B + 0.5\,p_M$ from a shared prefix, where B is a lightly retrained reference and M is the branch continued on the assemblies (Appendix A).

3.3 Autoregressive part anchors

The number of region hypotheses J is the number of connected components of the decoded mesh. A frozen ShapeVAE (Zhao et al., 2023) reads the 8192 oriented samples as 257 condition tokens, giving $C = E_s(X)$. The decoder has 24 layers and hidden width 1024. Let $Z = (z_t)_{t=1}^{T}$ be the token sequence, z_t the token at step t, T its length, and θ the decoder. Conditioned on C, the decoder generates

$$p_\theta(Z \mid X) = \prod_{t=1}^{T} p_\theta(z_t \mid z_{<t}, C), \qquad \mathcal{L}_{\text{AR}} = -\frac{1}{T}\sum_{t=1}^{T} \log p_\theta(z_t \mid z_{<t}, C). \tag{1}$$

After decoding, connected components $C_1, \ldots, C_J$ of the mesh are the part hypotheses. Emission order need not follow parts: one primitive may appear as several visible patches. The part is that component, not the order of its faces. The region queries stay with this readout: decoded connectivity groups those hidden states into queries on X, and J equals the component count. The sequence stays near the visible surface. Pooling follows attention-based set aggregation (Lee et al., 2019), and the groups are the decoded components. Each component must enter the assignment head as one query.

Let $\mathbf{h}_t$ be the decoder hidden state of face t, $\mathcal{I}_j$ the faces of component C_j, and $\mathbf{q}_*$ a shared learned query. Learned maps W_s, W_k, W_v project these states, d is the attention width, $\mathbf{m}_t$ is the projected state, and a_{jt} is its pooling weight. They form the initial query $\mathbf{q}_j^0$ (Fig. 5):

$$\mathbf{m}_t = W_s\,\mathrm{LN}(\mathbf{h}_t), \qquad a_{jt} = \frac{\exp(\mathbf{q}_*^\top W_k \mathbf{m}_t/\sqrt{d})}{\sum_{u\in\mathcal{I}_j}\exp(\mathbf{q}_*^\top W_k \mathbf{m}_u/\sqrt{d})}, \tag{2}$$

$$\mathbf{q}_j^0 = \mathrm{MLP}\Big(\mathbf{q}_* + \sum_{t\in\mathcal{I}_j} a_{jt} W_v \mathbf{m}_t\Big). \tag{3}$$

Different from unconditional slot attention, $\mathbf{q}_j^0$ already carries the spatial evidence of a proposed region.

3.4 Grounding on the observed surface

Assignment reads the original samples, not the cut proxy (Fig. 6). Point features come from the frozen Point Transformer V3 and have width 1088. The assignment head has width 256 and runs for $L = 3$ rounds. For point i, its feature, coordinate, and normal form the vector $\mathbf{e}_i$, used as a key and a value (Wu et al., 2025):

$$\mathbf{e}_i = W_p\,\mathrm{LN}(\mathbf{f}_i) + \mathrm{MLP}_x([\mathbf{x}_i;\mathbf{n}_i]) \tag{4}$$

In each round $\ell = 0,\dots,L-1$, the query looks at all points, pools them into $\mathbf{u}_j^\ell$ with weights α_{ji}^ℓ, and updates itself with a GRU and a feed-forward layer:

$$\alpha_{ji}^\ell = \mathrm{softmax}_i\big(\mathrm{LN}(\mathbf{q}_j^\ell)^\top W_K \mathbf{e}_i/\sqrt{d}\big), \qquad \mathbf{u}_j^\ell = \sum_i \alpha_{ji}^\ell W_V \mathbf{e}_i, \tag{5}$$

$$\widetilde{\mathbf{q}}_j^{\ell+1} = \mathrm{GRU}(\mathbf{u}_j^\ell, \mathbf{q}_j^\ell), \qquad \mathbf{q}_j^{\ell+1} = \widetilde{\mathbf{q}}_j^{\ell+1} + \mathrm{FFN}(\widetilde{\mathbf{q}}_j^{\ell+1}), \tag{6}$$

The updated query $\mathbf{q}_j^L$ scores point i by s_{ji}. The scores are normalized over queries, and the point is assigned to the highest-scoring query:

$$s_{ji} = (W_q \mathbf{q}_j^L)^\top (W_e \mathbf{e}_i)/\sqrt{d}, \qquad \pi_{ji} = \mathrm{softmax}_j(s_{ji}), \qquad \hat{y}_i = \arg\max_j \pi_{ji}. \tag{7}$$

Labels are placed on the input points, so a displaced proxy can still send points to the right region.

During training, several generated components may correspond to the same labeled part k; call them $\mathcal{J}_k$. The loss first adds their scores, giving g_{ki} and p_{ki}. Inference does not use this step:

$$g_{ki} = \log \sum_{j\in\mathcal{J}_k} \exp(s_{ji}), \qquad p_{ki} = \mathrm{softmax}_k(g_{ki}), \tag{8}$$

Only points with a clear owner and a supporting query are supervised. Ω is that set, y_i is the labeled owner of point i, and $\mathcal{K}_+$ is the set of owners that appear. The loss is pointwise cross-entropy plus a Dice term balanced across regions. The weight λ_D and the smoother ϵ are in Appendix A.

$$\mathcal{L}_{\text{point}} = -\frac{1}{|\Omega|}\sum_{i\in\Omega}\log p_{y_i i} + \frac{\lambda_D}{|\mathcal{K}_+|}\sum_{k\in\mathcal{K}_+}\left(1 - \frac{2\sum_{i\in\Omega} p_{ki}\mathbf{1}[y_i=k]+\epsilon}{\sum_{i\in\Omega} p_{ki} + \sum_{i\in\Omega}\mathbf{1}[y_i=k]+\epsilon}\right) \tag{9}$$

Points that cannot be assigned to one part stay in X unlabeled. Inference uses Eq. (7), and each query is its own region.

3.5 Visibility-aware, compact primitive fitting

Assignment decides which points belong together. Fitting decides which solid explains them. The type head reads rotation-invariant statistics $\phi(\widehat{R}_j)$ of those points, combines them with the refined query $\mathbf{q}_j^L$, and predicts a family preference $\boldsymbol{\tau}_j$ (Table 1). The geometric channel is $\mathrm{LN}(15) \to 64 \to 256$, and the classifier has width 512. $t_j^\star$ is the training family, with loss $\mathcal{L}_{\text{type}} = \sum_j \mathrm{CE}(\boldsymbol{\tau}_j, t_j^\star)$.

$$\mathbf{g}_j = \mathrm{MLP}_g\big(\phi(\widehat{R}_j)\big), \qquad \boldsymbol{\tau}_j = \mathrm{softmax}\big(W_\tau\,\mathrm{LN}(\mathbf{q}_j^L + \mathbf{g}_j)\big), \tag{10}$$

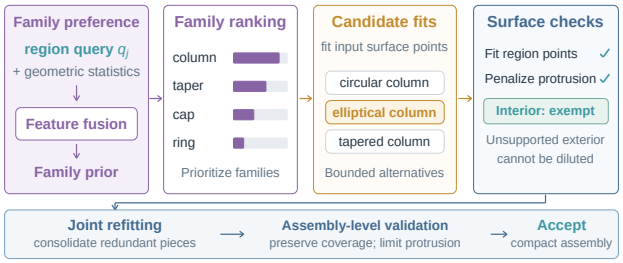

Figure 7: **Family-guided fitting.** Learned preferences guide candidate fitting on input surface points. A ring-cut sphere becomes two coincident spheres that are then merged; a ring-cut cylinder is merged into one longer cylinder. A primitive fully contained in another is removed.

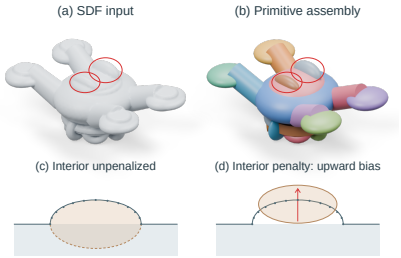

Figure 8: **Surface fitting and interior completion.** Red circles mark embedded ellipsoids in the input and assembly. The cross-sections illustrate interior completion versus the outward bias induced by penalizing buried surfaces.

Table 1: Primitive families and shape parameters. Polygonal prisms allow 3–8 cross-section vertices and must retain a constant cross-section along a straight extrusion axis.

Family	Primitive body	Shape parameters
Polyhedral	Box, prism, pyramid	Extents, profile vertices, height
Column / taper	Circular or elliptical cylinder, cone, frustum	Section radii, height, end-section sizes
Ellipsoid / cap	Ellipsoid, closed half-ellipsoid, teardrop cap	Semi-axes, cap taper
Ring / tube	Torus, elliptical ring, restricted swept tube, hollow cylinder or frustum	Centerline dimensions, section radii, inner/outer radii, height
Bounded sector	Column sector, annular sector, frustum-wall sector, ring arc	Full-body dimensions, start angle, sweep angle

Note. All bodies additionally have position and orientation. Rows group fitter parameterizations geometrically.

Types in one family look alike, so they are fitted to the region's points in the predicted order of preference (Fig. 7). Fitting first finds a coarse primitive. From the position, normals, and principal directions of the assigned points, it estimates a cylinder axis and radius, or a box orientation and size. Least squares and other numerical optimization then adjust position, rotation, and size so the input points lie near the primitive surface. Part of the refinement uses finite differences to measure the effect of a parameter change. The whole primitive is then checked: a candidate must pass two checks. The input points assigned to the query should lie near the primitive surface; enclosing them inside the solid is not enough. Surface that sticks out of the input object without observed support is penalized. Surface buried inside the object is not penalized for lying away from the exterior. This allows interior completion and limits unsupported outward protrusions. In Fig. 8, the circled ellipsoid follows the visible surface and extends into the fuselage. Penalizing that buried surface pushes the fit outward and away from the body. A candidate is kept only when both checks pass; otherwise only a limited set of other types is tried.

After fitting, redundant primitives are checked. For example, when a sphere is cut by a ring, different regions of the same sphere each receive an anchor and are completed on their own, producing two nearly coincident spheres; this triggers a merge. A cylinder cut by a ring likewise yields two anchors, which we merge into one longer cylinder. Other primitives that can be merged are merged as well. A primitive fully contained in another is deleted. For a region that still cannot be fit by a primitive, we build a very low-density triangle mesh of that region and replace each face by a thin triangular prism or frustum, so the result meets the primitive constraint.

4 Experiments

The evaluation has three parts. Section 4.1 fixes the data, the metrics, and the baselines. Section 4.2 compares assembly quality and runtime with methods that emit their own geometry. Section 4.3 tests the autoregressive anchors and the learned surface assignment. Training configuration is in Appendix A.

4.1 Experimental setup

Dataset construction. Training surfaces come from primitive assemblies. Each assembly is stored as a dense exterior for assignment and as a contact-separated sparse proxy for the autoregressive target, so the token sequence records part structure rather than a dense mesh (Sec. 3.2). The training pool contains 110,402 sources.

Testing uses two fixed sets of $2{,}000$ inputs, kept separate. Near-primitive surfaces are drawn from the held-out split, which asks whether a compact primitive explanation generalizes to surfaces withheld from training. Randomly sampled Objaverse (Deitke et al., 2023) meshes ask the same metrics of surfaces that were not built as primitive assemblies. Within a set, every method receives the same input and the same geometric reference. The shape encoder and the point encoder read the same 8192 oriented samples.

Metrics. One distance does not separate a compact assembly from an oversized or incomplete one. Surface distance is reported as both CD-L1 and CD-L2^2. CD-L1 is the mean bidirectional nearest-neighbor distance, as a percentage of the reference bounding-box diagonal D; it measures average surface agreement. CD-L2^2 is the mean square of those same nearest-neighbor distances, in units of $10^{-4}D^2$, so larger local deviations stand out more. F1 asks whether that agreement holds in both directions at a fixed fraction of D. With area-uniform samples A of the reference S, recall at tolerance τ is

$$r(\tau) = \frac{1}{|A|} \sum_{x \in A} \mathbf{1}\big[d(x, \widehat{S}) \le \tau\big], \tag{11}$$

and precision swaps the two surfaces. F1@1% and F1@2% are the harmonic means at $\tau = 0.01D$ and $\tau = 0.02D$. Recall without precision is an oversized hull; precision without recall is a missing part. Normal consistency (NC) is the mean absolute agreement of normals at those nearest neighbors, in percent. Where a ground-truth assembly exists, volume IoU is the intersection over union of the output solid and that assembly, in percent. $\bar{K}$ is the mean output count and is the compactness of the result; a smaller count is not better unless the surface metrics hold. Mean time is the cost of producing the output.

Baselines. We compare methods that read 3D geometry directly. Image- or text-conditioned assembly, and synthesis inside a modeling tool, use different inputs and are excluded. PrimitiveAnything (Ye et al., 2025) outputs assemblies of cubes, spheres, and cylinders. Official MeshAnything V2 (Chen et al., 2025b) is a mesh baseline; its connected-component count is not a primitive count. Marching-Primitives (Liu et al., 2023) converts a watertight SDF into superquadrics, without mapping those types onto our catalog. SAMesh (Tang et al., 2024) is the region source in the ablation of Sec. 4.3.

4.2 Comparative analysis

Assembly quality. Table 2 aggregates every test input. Figs. 13, 14, and 9 show the qualitative comparisons. All objects in Fig. 9 are from Objaverse. The SAMesh column in Figs. 13 and 14 is read in Sec. 4.3. On near-primitive surfaces, our CD-L1 is 0.69, F1@1% is 85.92, F1@2% is 92.06, and NC is 88.67, each better than PrimitiveAnything, MeshAnything V2, and Marching-Primitives in the upper block, at a mean count of 15.87. Volume IoU against the ground-truth assembly is 92.39 for ours, 68.83 for PrimitiveAnything, 75.95 for MeshAnything V2, 83.33 for Marching-Primitives, and 79.63 for SAMesh plus fitting. Marching-Primitives extracts superquadrics from a watertight SDF, and that construction tends to make the solids slightly too large. Its F1@1% is 49.41 while F1@2% is 85.47, the widest gap in the upper block, so the surface sits near the reference but outside the tighter band. The volume IoU of 83.33 shows that these enlarged bodies still cover much of the ground-truth solid. Ours keeps the tighter surface and the higher volume overlap. PrimitiveAnything uses 29.76 elements for a weaker surface match. Under complex overlap it emits many primitives and the assembly is disordered; on the randomly sampled Objaverse meshes its mean count is 88.48. On that set our CD-L1 is 0.95, F1@1% is 78.63, F1@2% is 89.82, NC is 82.31, and the mean count is 27.01. Compared with the other methods, our primitives are arranged in a more orderly and pleasing way, closer to how a person would judge the assembly. Ten people in the lab viewed the outputs with method names hidden, comparing every baseline and one ablation; all ten judged ours the most orderly and the most visually pleasing. The sixth object in Fig. 9 triggers an unfittable region: the region is first turned into a very low-density triangle mesh, and each face is then replaced by a thin triangular prism or frustum.

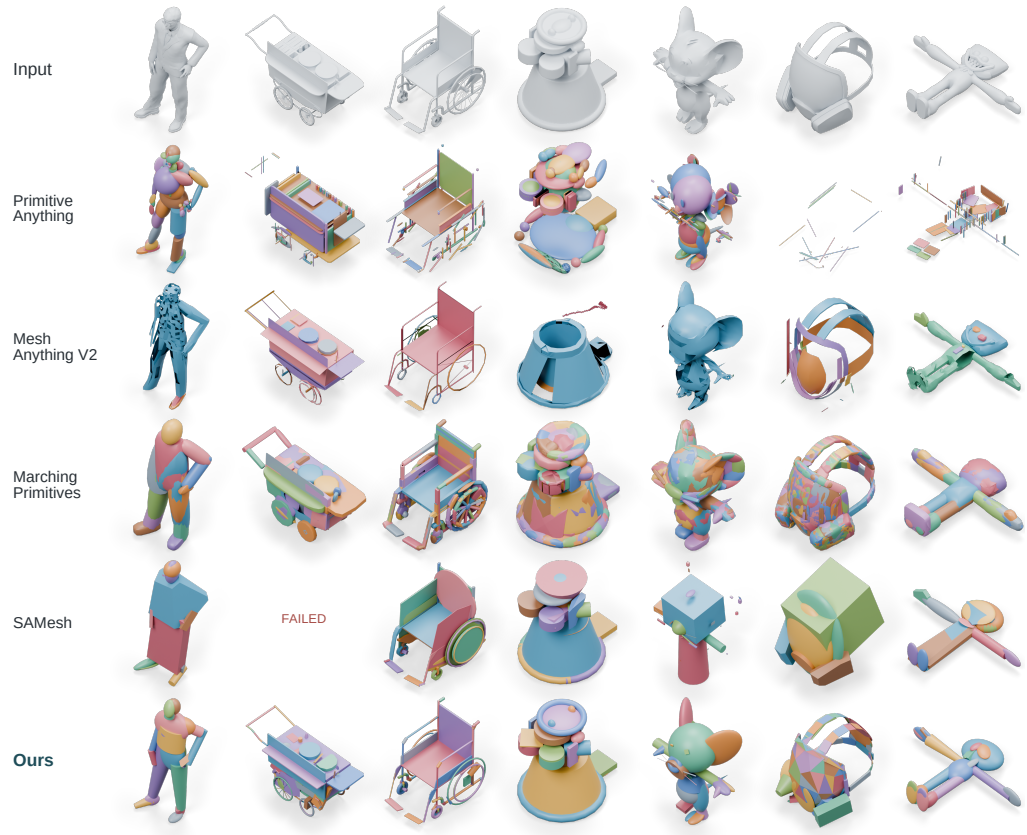

Figure 9: Columns correspond to objects; rows show the input and five methods, with ours last. Each object uses the same camera and scale across methods, with consistent lighting and materials. FAILED denotes an unavailable method output.

Table 2: Surface comparison. Upper block: 2,000 near-primitive surfaces. Lower block: 2,000 randomly sampled Objaverse meshes. n counts valid outputs. CD-L1 is a percentage of the reference diagonal D. CD-L2^2 is in units of $10^{-4}D^2$. F1, normal consistency (NC), and volume IoU are percentages; time is seconds. Volume IoU is reported only in the upper block, where a ground-truth assembly exists. $\bar{K}$ is the mean output count.

Method	n	$\bar{K}$	CD-L1 ↓	CD-L2^2 ↓	F1@1% ↑	F1@2% ↑	NC ↑	Vol. IoU ↑	Mean s ↓
Near-primitive surfaces									
PrimitiveAnything	2000	29.76	1.52	6.65	56.82	77.62	75.48	68.83	0.3
MeshAnything V2	2000	–	2.16	29.92	57.79	77.52	79.50	75.95	78.6
Marching-Primitives	2000	20.41	1.49	8.41	49.41	85.47	76.47	83.33	118.0
SAMesh + fit	1999	11.99	0.79	4.08	82.84	91.32	87.56	79.63	462.5
Ours	2000	15.87	0.69	2.02	85.92	92.06	88.67	92.39	174.8
Randomly sampled Objaverse									
PrimitiveAnything	2000	88.48	2.98	50.30	48.24	66.78	69.45	–	0.6
MeshAnything V2	2000	–	1.96	29.63	65.96	81.47	80.67	–	92.9
Marching-Primitives	1998	36.87	1.57	13.23	51.22	85.19	74.89	–	113.1
SAMesh + fit	1567	33.21	1.09	7.15	74.63	87.54	83.66	–	548.6
Ours	2000	27.01	0.95	4.48	78.63	89.82	82.31	–	252.5

Efficiency. Mean time in Table 2 is the cost of one output. PrimitiveAnything is the fastest comparison entry (0.3 s and 0.6 s). MeshAnything V2 takes 78.6 s and 92.9 s. Our generator takes 174.8 s on the near-primitive surface set and 252.5 s on Objaverse. Of that time, autoregressive generation accounts for 62.18%, geometric fitting for 37.72%, and assignment plus the type head, including feature preparation, for 0.10%. Marching-Primitives takes 118.0 s and 113.1 s.

4.3 Ablations

Replacing the anchors. SAMesh segments a surface from multi-view masks and does not require a chosen part count. To test the effectiveness of the autoregressive anchors, we replace those anchors with SAMesh regions and keep the same fitting pipeline (Table 2). On the near-primitive surface set, SAMesh plus fitting reaches CD-L1 0.79, F1@1% 82.84, F1@2% 91.32, and NC 87.56, all below our row, at a mean count of 11.99. On Objaverse its 1567 valid outputs reach CD-L1 1.09 and F1@1% 74.63, against our 0.95 and 78.63, at a mean count of 33.21 against our 27.01. Beyond the metrics, the visual gap is larger: our primitive arrangement is much more orderly.

Learned surface assignment. Figure 10 compares three ways of grounding the same generated anchors on one held-out near-primitive surface, with type guidance and the fitting procedure held fixed. Fitting the anchor surfaces directly inherits their gaps and incomplete part boundaries, leaving broken components. Assigning each input point to the nearest anchor vertex places the dense samples on those anchors, but distance alone mixes neighboring regions and yields an irregular arrangement. Learned assignment gathers, for each anchor, the input surface that belongs to it; the fit uses that surface and produces more complete components in a more coherent assembly, and can also correct cases in which an anchor's geometric position is wrong.

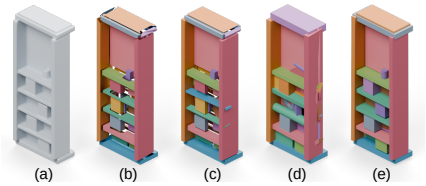

Figure 10: One held-out near-primitive surface. (a) Input; (b) anchors; (c) direct anchor fit; (d) nearest-vertex fit; (e) learned-assignment fit.

4.4 Optional primitive stylization

Some inputs contain intricate geometry that is difficult to express with a compact primitive assembly. Primitive stylization offers an optional way to simplify their appearance before decomposition. This step is not tied to a particular algorithm: any procedure that returns a suitable primitive-like surface can supply the input. We give one possible route (Appendix C). The same assembly pipeline then processes that surface without retraining. Figure 11 illustrates this choice on a slender lattice structure. Direct reconstruction works poorly and looks disordered on this example. Stylization deliberately changes the fine structure into a simpler interpretation.

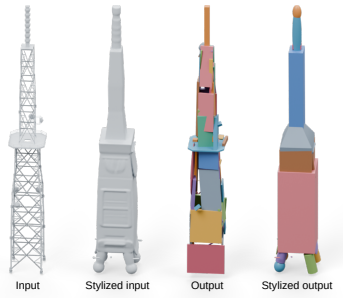

Figure 11: Original input, primitive-stylized surface, direct output, and our output after stylization, from left to right.

5 Conclusion

This paper assembles an observed surface into a small set of editable primitives. Predicting primitive parameters directly must fill in interiors the observation does not show. Segmentation alone tends to drop small exposed parts. An autoregressive mesh first proposes those parts. On the original observation we then decide where each part lies and which analytic body represents it. The sparse mesh only decides how the parts separate. The final geometry comes from the observed surface.

On surfaces with a ground-truth assembly, our surface lies closer to the reference and the volume overlap is higher. On Objaverse meshes that were not built as primitive assemblies, the surface is likewise closer and the primitive arrangement is more orderly.

Limitations. A part that has been proposed can still be corrected by later localization and fitting. A region that was never proposed cannot be supplied later. Once two parts have been gathered into one component, they are difficult to split apart. Shapes that are very fragmented, or rarely seen in the training data, remain limited by sequence length and by how broadly the training data covers such shapes.

REFERENCES

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. In *International Conference on Learning Representations*, volume 2025, pp. 51369–51389, 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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13922–13931. IEEE, 2025b.

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

Aditya Ganeshan, Matheus Gadelha, Thibault Groueix, Zhiqin Chen, Siddhartha Chaudhuri, Vladimir G. Kim, Wang Yifan, and Daniel Ritchie. Residual primitive fitting of 3D shapes with SuperFrusta. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://bardofcodes.github.io/superfit/.

Shuting He, Henghui Ding, Xudong Jiang, and Bihan Wen. Segpoint: Segment any point cloud via large language model. In *European Conference on Computer Vision*, pp. 349–367. Springer, 2024.

Kuan-Chih Huang, Xiangtai Li, Lu Qi, Shuicheng Yan, and Ming-Hsuan Yang. Reason3d: Searching and reasoning 3d segmentation via large language model. In *2025 International Conference on 3D Vision (3DV)*, pp. 1177–1186. IEEE, 2025.

Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. Oneformer3d: One transformer for unified point cloud segmentation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20943–20953. IEEE, 2024.

Black Forest Labs. FLUX.1: A 12B parameter rectified flow transformer for text-to-image. https://blackforestlabs.ai/, 2024.

Tri Le, Khang Nguyen, Baoru Huang, Tung D. Ta, and Anh Nguyen. CADKnitter: Compositional CAD generation from text and geometry guidance. In *SIGGRAPH Asia 2026 Conference Papers*, 2026. URL https://cadknitter.github.io/.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 2647–2655. IEEE, 2019.

Yuan Li, Cheng Lin, Yuan Liu, Xiaoxiao Long, Chenxu Zhang, Ningna Wang, Xin Li, Wenping Wang, and Xiaohu Guo. CADDreamer: CAD object generation from single-view images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21448–21457, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Li_CADDreamer_CAD_Object_Generation_from_Single-view_Images_CVPR_2025_paper.html.

Youtian Lin, Yikang Yang, Zhanpeng Hu, Mengqi Zhou, Feihu Zhang, Xun Cao, Jiaheng Liu, and Yao Yao. Procedura: Agentic 3D modeling with procedural control. *arXiv preprint arXiv:2608.26238*, 2026. URL https://arxiv.org/abs/2608.26238.

Minghua Liu, Mikaela Angelina Uy, Donglai Xiang, Hao Su, Sanja Fidler, Nicholas Sharp, and Jun Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9704–9715. IEEE, 2025.

Weixiao Liu, Yuwei Wu, Sipu Ruan, and Gregory S Chirikjian. Marching-primitives: Shape abstraction from signed distance function. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8771–8780. IEEE, 2023.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Yuzhe Luo, Zherong Pan, Kui Wu, Xingyi Du, Yun Zeng, Xiangjun Tang, Yiqian Wu, Xiaogang Jin, and Xifeng Gao. RL-ACD: Reinforcement learning-based approximate convex decomposition. *ACM Transactions on Graphics*, 44(6):1–12, 2025. URL https://xjtang.com/publication/2025-RL-ACD-Reinforcement-Learning-based-Approximate-Convex-Decomposition.

Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8216–8223. IEEE, 2023.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19615–19625. IEEE, 2024.

George Tang, William Zhao, Logan Ford, David Benhaim, and Paul Zhang. Segment any mesh. *arXiv preprint arXiv:2408.13679*, 2024.

Rui-Huan Wang, Si-Tong Wei, Jia-Qi He, Heng-Yi Wei, Baoquan Chen, and Peng-Shuai Wang. aDSL: Agentic 3D creation via joint agent-program design. *arXiv preprint arXiv:2608.17975*, 2026. URL https://arxiv.org/abs/2608.17975.

Yuhan Wang, Weikai Chen, Zeyu Hu, Runze Zhang, Yingda Yin, Ruoyu Wu, Keyang Luo, Shengju Qian, Yiyan Ma, Hongyi Li, Yuan Gao, Yuhuan Zhou, Hao Luo, Wan Wang, Xiaobin Shen, Zhaowei Li, Kuixin Zhu, Chuanlang Hong, Yueyue Wang, Lijie Feng, Xin Wang, and Chen Change Loy. Light-SQ: Structure-aware shape abstraction with superquadrics for generated meshes. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, 2025. URL https://johann.wang/Light-SQ/.

Xiaoyang Wu, Daniel DeTone, Duncan Frost, Tianwei Shen, Chris Xie, Nan Yang, Jakob Engel, Richard Newcombe, Hengshuang Zhao, and Julian Straub. Sonata: Self-supervised learning of reliable point representations. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22193–22204. IEEE, 2025.

Yuezhi Yang, Qixing Huang, Mikaela Angelina Uy, and Nicholas Sharp. Learning convex decomposition via feature fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. URL https://research.nvidia.com/labs/sil/projects/learning-convex-decomp/.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024.

Jingwen Ye, Yuze He, Yanning Zhou, Yiqin Zhu, Kaiwen Xiao, Yong-Jin Liu, Wei Yang, and Xiao Han. Primitiveanything: Human-crafted 3d primitive assembly generation with auto-regressive transformer. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–12, 2025.

Fenggen Yu, Yiming Qian, Xu Zhang, Francisca Gil-Ureta, Brian Jackson, Eric Bennett, and Hao Zhang. DPA-Net: Structured 3D abstraction from sparse views via differentiable primitive assembly. In *European Conference on Computer Vision*, pp. 454–471. Springer, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/10442.pdf.

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.

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

A Implementation and evaluation modes

Illegal next tokens are masked by parser state. In Eq. (9), $\lambda_D = 0.25$ and $\epsilon = 1$. Both branches consume the same indexed sample. The type-head channel sends no gradient into the surface assignment network.

Training. We train the generator, assignment head, and type head in stages. The reference branch B is a lightly retrained MeshAnything; the trainable branch M continues for many steps on the assemblies. Both branches read the same prefix, and the training objective is hard cross-entropy on the mixture $0.5\,p_B + 0.5\,p_M$, which keeps part of B's behavior on non-primitive surfaces. We then train pooling and assignment with a fixed decoder replay source and frozen point encoder, followed by the type head with grouping frozen. Evaluation uses freely generated sequences and predicted connectivity.

Current training configuration. The anchor generator is trained on four nodes, each with eight NVIDIA A800 80 GB GPUs and a 60-core CPU quota (Intel Xeon Platinum 8374C at 2.70 GHz), using 32-GPU distributed data parallelism. The surface assignment network is trained separately on eight GPUs. The main training set contains $110{,}402$ sources. Generator training uses AdamW with batch size 128, a decoder learning rate of 10^{-5}, and a condition-projection learning rate of 3×10^{-5}. Surface-assignment training uses AdamW with batch size 8 and learning rate 2×10^{-5}. The software stack is Python 3.11.9, PyTorch 2.3.1, CUDA 12.1, and FlashAttention 2.6. The generator is initialized from MeshAnything V2, and the point branch uses frozen Sonata features. Evaluation uses freely generated meshes and predicted connectivity. Generation at evaluation uses B and M at step 134000, with the same mixture coefficient as training, 0.5.

B Remeshing constraints for anchor supervision

The autoregressive anchors are trained on a sparse proxy of each training assembly (Sec. 3.2). This proxy is the component sequence emitted by the decoder. Its geometry is the visible exterior of the assembled solid, realized either as the exact Boolean surface or as a perturbed SDF of that exterior (Sec. 3.2). Source ownership partitions the exterior into regions. The region of a primitive consists of the portion of this cut exterior that the primitive exposes; surface lying inside another primitive is excluded, and a fully hidden primitive contributes no region.

Dense samples X are drawn from the same exterior and remain unchanged. Within each exposed region, the boundary is simplified, the vertices on the region's surface are rearranged, and the region is remeshed subject to the constraints below. Acceptance is evaluated in the integer frame $q = \mathrm{round}(64x + 63.5) \in \{0, \ldots, 127\}$, where one bin equals $1/64$ of the normalized coordinate. A region enters the anchor sequence only when every constraint holds.

- The region mesh contains no zero-area face, no duplicate face, no nonmanifold edge, and no branching boundary.
- The region mesh contains no T-junction, no coincident vertex, no self-intersection, and no coplanar overlap of positive area.
- Every edge has length in $[1, 30]$ bins. Every vertex lies within one bin of the dense exterior and within one bin of the local plane of its face.
- The remeshing retains every edge of the region's original boundary. The Euler characteristic of the region and the number of boundary edges stay unchanged.
- The proxy boundary remains within the accepted deviation of that region's reference boundary, and the proxy covers the reference exposed region.

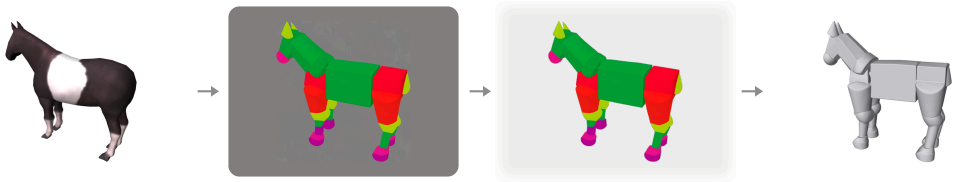

Figure 12: **Optional redraw.** Left to right: one perspective control view from a camera restricted to a fixed family, a type-colored redraw into flat, seam-cut parts on a plain backdrop, an exact border flood-fill that removes that backdrop, and an image-to-3D lift to a gray exterior with texture painting disabled. The bottom band is the closed set of fills: a color names a shape family rather than object albedo, and the closest pair sits 149 apart in RGB against a measured 42 paint drift, so a fill decodes by lookup. No step emits analytic parameters.

- After quantization, two regions that belong to different source primitives share no full edge. The construction keeps the boundary that faces a contact open, so the quantized owners stay separate.

C Optional redraw for difficult inputs

The main method samples an observed surface directly. An exterior that lies far from a primitive assembly, or that is too complex to explain with a small set of primitives, may first be redrawn toward the assembly exteriors on which the anchor generator was conditioned. Surfaces that already resemble primitive assemblies are sampled without this redraw. The redraw returns a part-biased surface and emits no analytic parameters. Fig. 12 specifies the route and the closed set of fills. Because the resulting X is a reconstruction, redraw and lifting error propagates into assignment and fitting.

Control view. For a mesh we render one perspective control at 1024^2 (raised elevation, fixed three-quarter azimuth, white backdrop), and image inputs bypass this step. The control camera is restricted to this family because the abstraction adapter has seen only perspective renders of this form, never orthographic or top-down views.

Type-colored redraw. FLUX.1 (Labs, 2024), fine-tuned with a rank-128 LoRA, driven by a fixed trigger and type-color caption, redraws the subject as a packed collage of flat-filled parts on a plain gray backdrop. Ten saturated palette colors encode shape families rather than object albedo (Fig. 12): color is a discrete channel that survives reconstruction as a visible region boundary, whereas a shaded repaint would leave type unreadable. The caption also fixes granularity, asking for a coarse build of a few dozen large parts rather than many small ones, so proposals arrive at the scale the anchor generator expects. We do not ask the model to restore object-true color; a deterministic recoloring exists but is unnecessary here, since the generator consumes the drawing itself.

Reconstruction and interface. A border flood-fill removes the backdrop instead of a learned matting network: the backdrop is a single flat gray and the subject may contain gray, so a connected flood from the image border is exact where a generic matting model guesses. Hunyuan3D (Zhao et al., 2025) then reconstructs a gray mesh with texture painting disabled, since only a geometric exterior is needed downstream.

D Statements

AI use statement

Humans designed the method and the experiments. While writing code, the authors used generative AI to carry out some implementation steps; every such change was reviewed by a human. Generative AI was also used to polish the paper and to help produce some of the data. The authors take responsibility for the final text, claims, code, and data.

Ethics statement

This work converts an observed 3D surface into an editable primitive assembly. It does not involve human-subject studies and does not collect personal data. The assets used for training and evaluation are existing 3D models, including publicly released Objaverse meshes, together with primitive assemblies constructed for supervision. The intended use is geometric editing and user-generated 3D content. The method does not identify people or make decisions about individuals.

Reproducibility statement

The architecture, the autoregressive objective, the assignment model, and the fitting procedure are in Secs. 3.3–3.5. Dual-resolution inputs and proxy construction are in Secs. 3.1–3.2, with the remeshing constraints in Appendix B. Hardware, optimizer, learning rates, the training mixture, and the evaluated checkpoint are in Appendix A. Datasets, metrics, and comparisons are in Sec. 4. Code and checkpoints will be released upon acceptance.

E Qualitative comparisons

Figures 13 and 14 compare the input surfaces and reconstructed outputs on near-primitive surfaces and Objaverse meshes, respectively, complementing the quantitative results in Table 2.

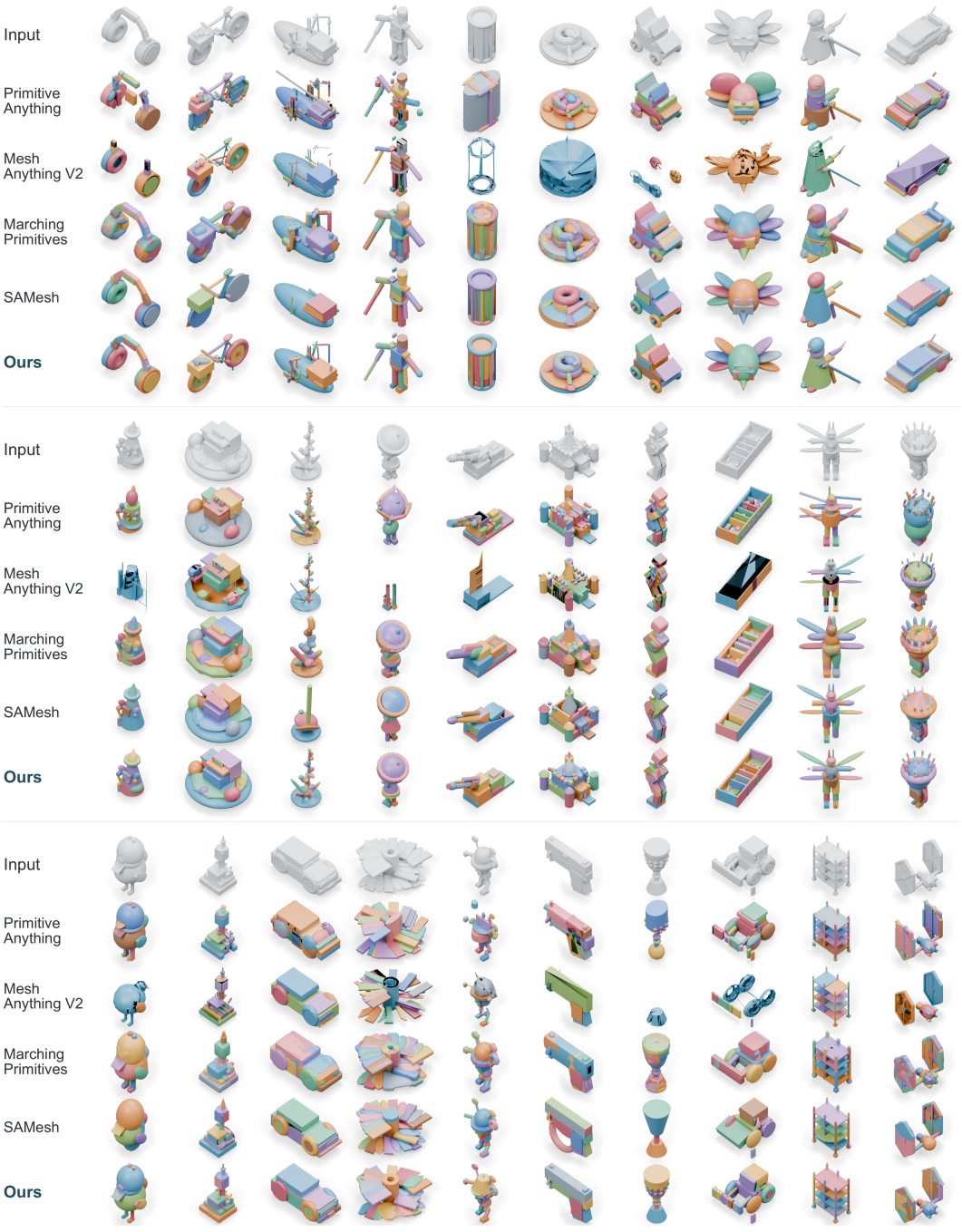

Figure 13: **Near-primitive surfaces.** Qualitative results for the upper block of Table 2. Input and five methods are shown with ours last. Each row uses a common camera, scale, lighting, and materials; empty cells denote method failures. Rendering does not change output geometry.

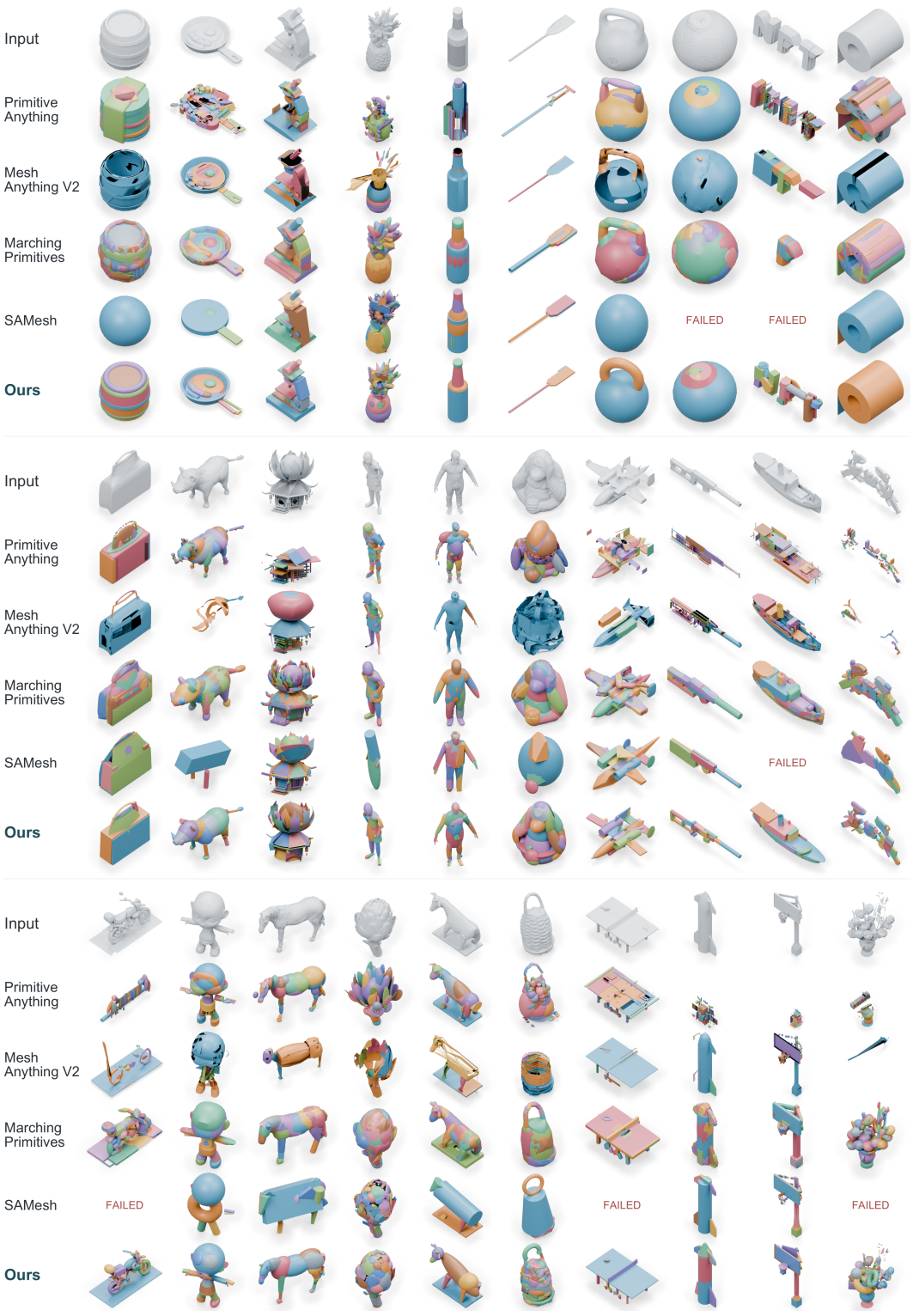

Figure 14: **Randomly sampled Objaverse.** Qualitative results for the lower block of Table 2. Input and five methods share a camera in each row, with ours last. Empty cells denote method failures. Rendering does not change output geometry.