Seeing Both the Forest and the Trees: Reusing Holistic 3D Priors for Part-Decomposed Generation

Yiqun Zhao[1,2] **Binbin Huang**[1,2] **Haobin Duan**[1,2] **Zibo Zhao**[3] **Shenghua Gao**[1,2,4*]
[1]The University of Hong Kong [2]Transengram
[3]Tencent [4]Shenzhen Loop Area Institute



Figure 1: Diverse decomposed 3D objects generated by our approach using images from recent open-source image generation models [1, 2]. Among them, the guns, train, crab, and motorcycle are specifically displayed in exploded views.

Abstract

Generating 3D assets with explicit parts requires two properties at once: geometric detail inside each part and structural coherence across the whole object. Recent image-conditioned 3D generation models adopt a *structured 3D representation* that anchors tokens to explicit spatial positions, capturing both properties, but their output remains a single fused mesh. We ask how a part-decomposed generation model can inherit both properties of these holistic priors, and answer with StructParts, a framework whose core technical contribution is a *dual-frame coordinate interleaving* mechanism: each shape latent is interpreted under both a part-local and an object-global reference frame, and processed through a global-local interleaved transformer that simultaneously reuses the prior's geometric detail at the part scale and its structural coherence at the whole-object scale. As its structural input, StructParts uses a *segmented mesh scaffold*, a coarse mesh paired with a face-level segmentation mask kept entirely within the prior's native 3D representation. With only lightweight finetuning, both intra-part detail and inter-part coherence emerge from the same pretrained weights. On PartObjaverse-Tiny, StructParts matches or surpasses prior part-decomposed models at a fraction of their training cost, while naturally supporting part-mesh refinement and part-level editing.

*Corresponding author. Project page: https://zhao-yiqun.github.io/structparts/

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Part-decomposed 3D asset generation is essential for downstream tasks such as localized editing and animation. A part-decomposed generation model must produce two properties at once: *geometric detail* inside each part, and *structural coherence* across the whole object. Recent image-conditioned 3D generation models [3, 4] adopt a **structured 3D representation** that anchors tokens to explicit spatial positions, jointly scaling to high-fidelity global structure and local geometric detail. The native 3D priors of such holistic models therefore already encode what part-decomposed generation needs, even though they emit a single fused mesh. Yet directly extending these priors to part-decomposed generation has proven non-trivial.

Approaches that adapt holistic 3D generation models to part-decomposed generation can be grouped into three families, and none of them fully inherits both properties of the holistic prior. (i) Methods that feed a point cloud [5, 6, 7], sometimes with extra abstract layout such as bounding boxes [5], supply an input the holistic prior was never trained to take; the part model must therefore relearn its conditioning almost from scratch, incurring substantial retraining cost [5, 7]. (ii) Two-stage methods [8, 9] repurpose the holistic generation model itself to predict structural abstractions such as bounding boxes or cuboid meshes in the first stage; the resulting objective shift drags the prior into a non-geometric latent space and erases the very geometric detail the pretraining was meant to provide. (iii) Methods that keep the original holistic input but predict each part directly in the unnormalized object frame [10, 11] preserve the input format but mismatch the prior's normalized pretraining space at the output side, where tiny parts occupy only a small fraction of the canvas and degrade accordingly. Across all three, the prior is forced to operate outside the conditions it was pretrained on, leaving neither its geometric detail nor its structural coherence fully inherited. This motivates our central question:

How can a part-decomposed generation model inherit both the structural coherence and the geometric detail of a holistic 3D prior?

Answering this question requires recognizing a fundamental characteristic of 3D data: its inherent hierarchical structure. A hat is both a component of a human character and a complete shape in its own right. This dual identity is not incidental but ubiquitous, and a holistic 3D prior trained on diverse 3D assets has implicitly observed each part as both a piece and a whole. To honor this duality, a part-decomposed model should not inherit the prior at one scale only: it should expose both scales simultaneously, treating each part as a complete shape locally (so per-part geometric detail is reused exactly as during pretraining) and as part of the whole object globally (so structural coherence is preserved across the assembly). We summarize this principle as seeing both **the forest and the trees**: just as a tree is itself a complete shape and at the same time part of a larger forest, a model should reason at the part scale and the object scale at once. The duality is mutually reinforcing: global context constrains parts toward coherence, while high-quality parts elevate the geometric fidelity of the whole.

Honoring this duality requires the model to read the prior at both scales simultaneously, which in turn forces a careful choice of structural input. We propose StructParts, whose first design choice is to keep the structural input format that the holistic prior was already pretrained on. Recent holistic 3D generation models [3, 12] condition shape synthesis on a coarse 3D mesh via voxel-anchored positional embeddings; we recognize that this same format already carries everything a part model needs to anchor its latents in 3D space, with the addition of a single face-level segmentation mask sufficing to expose part boundaries. Together, the coarse mesh and segmentation mask form a **segmented mesh scaffold**: a structural input the holistic prior reads without any objective shift, and one that off-the-shelf upstream stages produce as a by-product of holistic generation and segmentation.

With the scaffold in place, the core mechanism of StructParts is **dual-frame coordinate interleaving**, which inherits both prior properties at once. Each scaffold-anchored latent is interpreted under two complementary frames: a *local* frame normalized within each part, and a *global* frame normalized across the whole object. These yield distinct local and global positional embeddings, processed through a global-local interleaved transformer that alternates between intra-part and inter-part attention. In the local frame, each part is processed as a standalone shape, reusing the prior's geometric detail as during pretraining; in the global frame, the parts are coordinated as one anchored assembly, preserving the prior's structural coherence. Both properties therefore emerge

from the same pretrained weights with only lightweight finetuning, rather than requiring extensive retraining [7, 11, 5].

In practice, our pipeline stays simple. We train on segmented ground-truth watertight meshes; at inference, scaffolds come from a single image via structure generation [13] and 3D segmentation models [14, 15]. We retain the image condition alongside the scaffold; together with the holistic prior for geometric details, this enables part-level mesh refinement even when the scaffold is coarse. The scaffold also supports part-level editing via merge/split through segmentation regrouping in a training-free manner. On PartObjaverse-Tiny, STRUCTPARTS matches or surpasses prior part-decomposed models with substantially lower training costs, showing that strong part-decomposed 3D generation can emerge from a minimal reinterpretation of a holistic prior rather than a part-specific model trained from scratch; we demonstrate generation examples from STRUCTPARTS in Fig. 1.

In summary, our contributions are three-fold:

- We propose a *dual-frame coordinate interleaving* mechanism that lets a part-decomposed generation model inherit both the geometric detail and structural coherence of a holistic 3D prior with only lightweight finetuning.
- We adapt the coarse-mesh conditioning from holistic 3D generation models by adding a face-level segmentation mask, yielding a segmented mesh scaffold that exposes part boundaries while remaining in the prior's native 3D representation.
- On PartObjaverse-Tiny, our method matches or surpasses prior part-aware methods at a fraction of their training cost, while supporting part-mesh refinement and part-level editing.

2 Related Work

3D holistic shape generation. Recent 3D shape generation has converged on two representation paradigms. *Sparse-Voxel* methods [16, 4] restrict computation to active surface voxels, yielding a structured latent space well-suited for editing, but the active-voxel sequence grows rapidly with resolution and demands sparse convolution and specialized attention. *VecSet*-based approaches [17, 18, 19, 20, 21, 13] compress a shape into a small set of feature vectors via cross-attention between a dense point cloud and sparser point queries, fitting cleanly into standard transformer attention. Recent localizable VecSet variants such as LATTICE [3, 12] anchor each token to an explicit 3D position, inheriting both VecSet's efficiency and Sparse-Voxel's spatial grounding and capturing both global structure and local geometric detail. Despite this progress, all such holistic models still emit a single fused mesh, motivating a part-decomposed counterpart that directly reuses these native 3D priors rather than disrupting them.

3D parts reasoning. 3D part reasoning extracts part structure from existing 3D shapes. *Traditional supervised* methods [22] train segmentation networks on labeled benchmarks but struggle to generalize beyond seen categories. *2D-lifting* methods [23, 24] render multiple views and propagate masks from 2D foundation models [25, 26] back to the 3D surface; this enables open-vocabulary segmentation but suffers from view inconsistency and unstable post-processing. *Native 3D data-engine* methods [27, 15, 28, 29, 14] instead distill 2D foundation features into a 3D backbone for direct prediction. None of these synthesize new geometry; they yield part masks on a given mesh. Our work directly exploits this readily available signal at inference time, treating the segmented mesh as a structural scaffold injected into a holistic generation model, rather than training a separate part-prediction head from scratch or coupling decomposition and geometry generation within a single model [11, 10].

3D decomposed shape generation. Existing part-decomposed generation models acknowledge native 3D priors but exploit them only indirectly, falling into four patterns. (i) Conditioning a learned model on a point cloud, sometimes with additional abstract layout [5, 6, 7]; because such input formats differ from how holistic shape models are conditioned, the prior is not preserved and substantial retraining is required [5, 7]. (ii) Repurposing the generation model to first synthesize structural abstractions like bounding boxes or cuboid meshes [8, 9]; the first-stage objective shift drags the prior into a latent space sharply different from the native shape prior. (iii) Keeping the original holistic input but predicting each part in the unnormalized object frame [10, 11, 30]; this spatial-resolution mismatch hurts tiny parts that occupy only a small fraction of the object volume. (iv) Lifting 2D multi-view part predictions to 3D [31]; this sidesteps the native 3D latent at the cost

of view consistency and post-processing complexity. In contrast, StructParts stays in the native geometric domain, exposes part boundaries through a segmented mesh scaffold, and uses dual-frame coordinate interleaving to reason at both scales with only lightweight finetuning rather than full retraining of the underlying generation model.

3 Method

This section instantiates StructParts. As illustrated in Fig. 2, given a single image and a segmented mesh scaffold, we sample per-part shape latents anchored to the scaffold, attach dual-frame positional embeddings (one local to each part, one global across the object), and process them with a global-local interleaved transformer; the image condition is retained throughout, so fine detail flows from the image into each part even when the scaffold is coarse. We review the structured latent representation we build on (Sec. 3.1), detail the scaffold-guided global-local interleaving (Sec. 3.2), and describe its training-free applications (Sec. 3.3).

3.1 Preliminaries: structured shape latents for 3D diffusion

Structured shape latents. We build on recent 3D shape generation models [3, 12] that adopt a *structured* 3D representation, in which each latent token is anchored to an explicit spatial position. Given a watertight mesh, the encoder takes a curvature-aware sample [19, 21] of point attributes $\mathbf{X} \in \mathbb{R}^{N\times 7}$ (positions, normals, sharp-edge flags). We quantize $\mathbf{X}$ onto a regular grid of resolution R, yielding a query set $\mathbf{X}_0 \in \mathbb{R}^{N_0\times 7}$ with integer voxel indices $\mathbf{V}_0 \in \{0,\dots,R-1\}^{N_0\times 3}$ (one token per occupied cell), providing stable spatial information for the 3D DiT's positional embeddings (PE).

The transformer-based VAE encoder maps the shape into these structured latents:

$$\mathbf{Z} = \mathcal{E}(\mathbf{X}) = \mathrm{SelfAttn}^{8}\big(\mathrm{CrossAttn}(\mathrm{PE}(\mathbf{X}_0), \mathrm{PE}(\mathbf{X}))\big), \tag{1}$$

where PE denotes a Fourier positional embedding and $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{N_0}$ is the set of N_0 shape latents. The VAE decoder takes $\mathbf{Z}$ and a query point $\mathbf{q}$ as input to decode the implicit surface-field value at $\mathbf{q}$:

$$\mathcal{D}(\mathbf{q}, \mathbf{Z}) = \mathrm{CrossAttn}(\mathrm{PE}(\mathbf{q}), \mathrm{SelfAttn}^{16}(\mathbf{Z})). \tag{2}$$

3D shape diffusion. Following [21, 17], we apply a flow-based diffusion model [32] on top of these structured latents, with the integer voxel indices $\mathbf{V}_0$ providing a Rotary Positional Embedding (RoPE) [33] for each latent token.

3.2 Scaffold-guided global-local interleaving in transformer

Scaffold construction. Given a single image, we build a **segmented mesh scaffold** $\mathcal{S} = (\mathcal{V}, \mathcal{F}, \mathcal{L})$, where $(\mathcal{V}, \mathcal{F})$ is a coarse mesh produced by an off-the-shelf holistic 3D generation model [13] and $\mathcal{L}$ is a face-level segmentation mask from an off-the-shelf 3D segmentation model such as P3-SAM [14], partitioning the scaffold into K semantic parts. This format extends the coarse-mesh conditioning that holistic 3D models already accept with only a face-level mask, keeping the input in the prior's native 3D representation.

During training, each ground-truth part $k \in \{1,\dots,K\}$ is encoded into a per-part latent $\mathbf{Z}^{(k)}$ via the encoder $\mathcal{E}$ in Sec. 3.1; the full object latent is then the union $\mathbf{Z} = \bigcup_{k=1}^{K} \mathbf{Z}^{(k)}$. At inference, per-part latents are initialized from Gaussian noise and generated by the diffusion model, while the segmented scaffold supplies their spatial anchors.

Dual-frame positional embeddings. To read the prior at both scales, we interpret each shape latent under two complementary coordinate frames. For each part k, we sample dense points $\mathbf{X}^{(k)}$ on its scaffold surface, normalize them into a unit cube using the part's bounding box, and voxelize to obtain local voxel indices $\mathbf{I}_L^{(k)}$; inverting the bounding-box transform lifts the same points to the object's global frame, yielding global voxel indices $\mathbf{I}_G^{(k)}$. From these we compute two RoPE [33] embeddings:

$$\mathrm{RoPE}(\mathbf{x}, \mathbf{I}) = \mathbf{x} \odot \cos(\mathbf{\Theta_I}) + \mathbf{x}^{*} \odot \sin(\mathbf{\Theta_I}), \tag{3}$$

where $\mathbf{I}$ is a 3D voxel-index input, $\odot$ is element-wise multiplication, $\mathbf{x}^{*}$ pairs adjacent feature dimensions for rotation, and $\mathbf{\Theta_I}$ holds the rotary phases derived from $\mathbf{I}$. For the latent tokens in

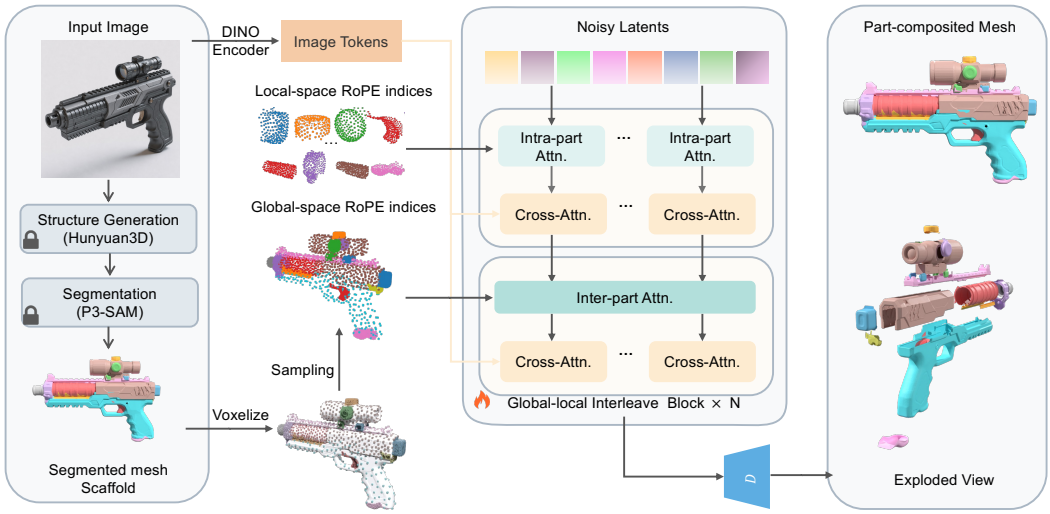


Figure 2: **Overview of StructParts's pipeline.** Given a single image, our method first constructs a 3D segmented scaffold using the structure generation model followed by a segmentation model. We then sample points from each part surface to construct the voxel indices in both the per-part normalized space and the shared global-shape space. These voxel indices serve as position embeddings to guide the diffusion model's sampling process. The segmented scaffold and part meshes depicted here are dataset-provided ground truth (GT), used to illustrate the part-wise data flow.

part k, we use local embeddings $\mathrm{RoPE}_L(\cdot) = \mathrm{RoPE}(\cdot, \mathbf{I}_L^{(k)})$ and global embeddings $\mathrm{RoPE}_G(\cdot) = \mathrm{RoPE}(\cdot, \mathbf{I}_G^{(k)})$, which are then injected into the shape latents described next.

Global-local interleaving attention. The dual frames let the network reason about each part as a standalone shape locally while still coordinating parts into one coherent object globally. For every latent token $\mathbf{Z}_i \in \mathbf{Z}^{(k)}$, we identify its nearest active voxel in the part's normalized frame and assign it both RoPE_L and RoPE_G. As illustrated in the middle of Fig. 2, our diffusion transformer alternates layer-by-layer between intra-part and inter-part attention, both implemented through one unified attention operation:

$$\mathrm{Attn}_F(\mathbf{Z}^{(k)}, \mathbf{Z}_{kv}) = \mathrm{softmax}\left(\frac{\mathrm{RoPE}_F(\sigma_q(\mathbf{Z}^{(k)}))\,\mathrm{RoPE}_F(\sigma_k(\mathbf{Z}_{kv}))^\top}{\sqrt{d}}\right)\sigma_v(\mathbf{Z}_{kv}), \tag{4}$$

where $\sigma_q, \sigma_k, \sigma_v$ are the query, key, and value projection layers and d is the per-head hidden dimension. In the *intra-part* attention layers we set $F = L$ and $\mathbf{Z}_{kv} = \mathbf{Z}^{(k)}$, so each part is processed as a standalone shape (**the trees**), reusing the prior's geometric detail exactly as during pretraining. In the *inter-part* attention layers we set $F = G$ and $\mathbf{Z}_{kv} = \mathbf{Z}$, coordinating the parts as one anchored assembly (**the forest**) and preserving the prior's structural coherence across the whole object. Both properties therefore emerge from the same pretrained weights, and only the alternating positional injection differs from holistic generation.

Image conditioning. In each block, per-part latents cross-attend (without RoPE) to image features $\mathbf{C}_{\text{img}}$ extracted by DINOv2 [26], so fine geometric detail can flow directly from the image into each part even when the scaffold itself is coarse.

Training. We optimize the network with a flow-matching [34, 32] objective. Given a ground-truth latent $\mathbf{Z}_0$, Gaussian noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and time step $t \sim \mathcal{U}[0, 1]$, we form the interpolated state $\mathbf{Z}_t = t\mathbf{Z}_0 + (1-t)\epsilon$ and supervise the network f_θ to predict the velocity $\mathbf{Z}_0 - \epsilon$ from $(\mathbf{Z}_t, t, \mathbf{C}_{\text{img}})$:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, \mathbf{Z}_0, \epsilon}\left[\left\| f_\theta(\mathbf{Z}_t, t, \mathbf{C}_{\text{img}}) - (\mathbf{Z}_0 - \epsilon)\right\|_2^2\right]. \tag{5}$$

3.3 Downstream tasks

Because StructParts retains the holistic prior intact and is scaffold-guided, two capabilities emerge without any additional training:

Part-mesh refinement from a coarse scaffold. The dual-frame embeddings only require coarse voxel indices, and the image condition is retained at every block. As a result, the model can refine a

low-fidelity scaffold into high-quality decomposed parts at inference time, with fine geometric detail recovered directly from the image rather than from the scaffold itself.

Part-level editing via scaffold regrouping. The segmentation mask $\mathcal{L}$ in the scaffold gives users a direct handle on decomposition granularity. Inspired by RePaint [35] and X-Part [5], we support training-free *merge* and *split* operations: *merge* fuses adjacent scaffold segments into a single coherent part by relabeling their faces in $\mathcal{L}$, while *split* subdivides an existing segment into distinct sub-parts. At inference, the modified $\mathcal{L}$ determines which latents are resampled by the flow-matching process; tokens in unedited regions are kept so that the rest of the object remains consistent.

4 Experiments

We first detail our implementation (Sec. 4.1) and experimental setup (Sec. 4.2). Next, we evaluate our approach against state-of-the-art methods through comprehensive quantitative and qualitative comparisons (Sec. 4.3). We then validate our core design choices via ablation studies (Sec. 4.4), and finally showcase the versatility of our pipeline through various downstream applications (Sec. 4.5).

4.1 Implementation Details

Backbone and initialization. StructParts uses a 21-layer DiT backbone following Hunyuan3D-2.1 [13], with self-attention layers alternating between intra-part (local frame, 128^3 part-normalized RoPE) and inter-part (global frame, 512^3 shape-global RoPE) positional conditioning. We initialize from open-source Hunyuan3D-2.1 [13] weights and finetune on part-level data.

Training data. We train on Partverse-XL [9] ($\sim$36K objects, $\sim$240K parts). Raw part meshes are remeshed to watertight via cuBVH [11] and 24 conditioning views per object are rendered following Trellis [16]; segmented watertight scaffolds for the full object are obtained by baking multi-view rendered segmentation labels onto a watertight remesh of the whole shape (full details in App. A.1).

Training compute. The DiT is trained for 40K steps on 8 H200 GPUs over $\sim$4 days, for a total allocation of 768 H200-hours. Full network configuration (hidden dim, MoE setup, NTK-aware RoPE base) and hyperparameters (optimizer, precision, gradient clipping, per-part token budget) are deferred to App. A.1.

4.2 Experimental Setup

We evaluate on PartObjaverse-Tiny [27] (200 high-quality objects with detailed segmentation labels), whose UIDs are strictly excluded from our training set.

Metrics. We report Chamfer Distance (CD) and F-Score at thresholds 0.05 and 0.10 (F-Score@0.05, F-Score@0.10) at two granularities: *part-level*, where each part is normalized into a $[-1, 1]$ bounding box, and *shape-level*, where the assembled object is normalized into a $[-1, 1]$ bounding box.

Settings and baselines. Because baselines differ in whether they accept layout controls, we use two complementary protocols. **(i) GT-layout-conditioned, part-level.** Methods that consume a layout (HoloPart [6], X-Part [5], FullPart [9], OmniPart [8]) receive ground-truth bounding boxes or segmentation masks, yielding a one-to-one correspondence between predicted and ground-truth parts and a direct part-level geometric assessment. **(ii) Image-to-decomposed-parts, shape-level.** Direct-generation baselines [8, 10, 11] synthesize parts from a single image with uncontrollable output layouts, so all methods are evaluated on the assembled shape; the layout-dependent methods are run with a standardized predicted pipeline using Hunyuan3D-2.1 [13] for the scaffold mesh and P3-SAM [14] for the part labels.

Matched-budget comparison. In addition to released baseline checkpoints, we retrain X-Part with the same Hunyuan3D-2.1 initialization, Partverse-XL data, 512 tokens per part, and 768-H200-hour allocation as StructParts. Tab. 1 reports this result separately; implementation details, training schedules, checkpoint comparisons, and per-object compute are given in App. A.2.

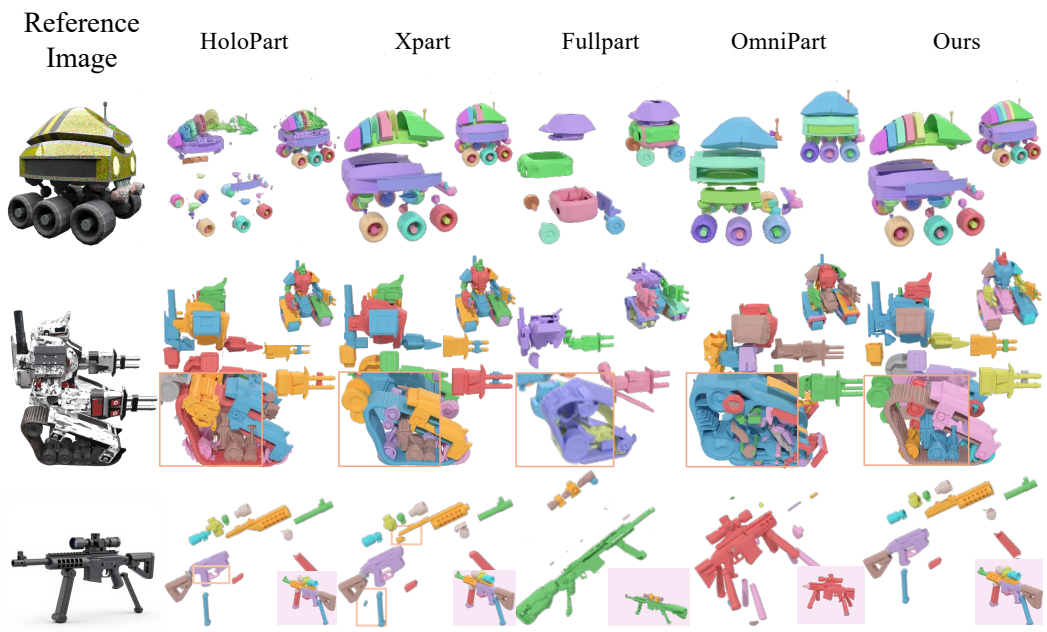


Figure 3: **Layout-conditioned part decomposition.** Comparison among methods that accept an explicit layout (HoloPart, X-Part, FullPart, OmniPart): each row shows assembled shapes for part coherency and exploded views for per-part geometry. Baselines drop fine details, leave incomplete shapes, or introduce floaters; StructParts preserves both by reusing the holistic prior.

4.3 Main Results

Layout-conditioned evaluation (part-level). Tab. 1 shows StructParts achieves the best decomposition quality across all metrics, with qualitative comparisons in Fig. 3. The advantage also holds over X-Part retrained under the same GPU-hour budget, providing a controlled comparison of training efficiency. FullPart and OmniPart often fail on complex structures even with ground-truth layouts. HoloPart and XPart follow the given layouts but introduce input signals that disrupt the native 3D prior, yielding missing geometric detail (XPart, row 2), incomplete shapes (row 1 for both), or floaters within part bounding boxes (XPart, row 3). StructParts avoids these failure modes by reusing the holistic prior in its native representation.

Image-to-decomposed-parts evaluation (shape-level). Tab. 2 shows StructParts is comparable to released X-Part and outperforms the other baselines by a clear margin; the matched-budget comparison is reported separately in Tab. 1. Fig. 4 reveals each baseline's failure mode: PartPacker and PartCrafter cannot produce semantically meaningful decompositions as they synthesize parts without explicit controllability; FullPart and OmniPart's bbox-conditioned formulation causes collisions and mismatches at part boundaries that propagate into the assembled shape. StructParts, XPart and HoloPart share the same scaffold input, yet only StructParts leaves the holistic prior undisturbed, preserving fine geometric detail in the final decomposed shape generation.

Additional validation. On a fixed 200-object ABO subset evaluated with identical inputs and metrics, StructParts obtains the lowest mean and median CD and matches released X-Part at F@0.05. Category-wise results, sample counts, and paired bootstrap intervals are reported in App. A.4. A controlled comparison of Hunyuan3D-2.1 and UltraShape initialization at the same training point is given in App. A.5.

4.4 Ablation Studies

We validate the dual-frame design in Tab. 3 by asking: (1) is the dual frame itself necessary, and (2) is local RoPE the right local-frame mechanism, compared with the per-part descriptors used in prior work [5, 10]? We compare three variants: **LRoPE only** (local RoPE within each part), **GRoPE only** (global RoPE across the whole object), and **GRoPE + part embed.** (global RoPE with per-part embeddings, as in [5]). Fig. 5 answers both. *LRoPE only* treats every part identically in its

Table 1: **Part decomposition.** Part-level comparison on PartObjaverse-Tiny. †Released X-Part [5] uses ∼2M closed-source objects/128 GPUs versus our ∼36K open-source objects/8 GPUs; matched X-Part uses our initialization, data, token and compute budgets (App. A.2).

Method	CD ↓	F-0.05 ↑	F-0.10 ↑
OmniPart	0.238	0.43	0.60
HoloPart	0.182	0.48	0.64
X-Part†	0.114	0.63	0.75
X-Part (matched)	0.169	0.55	0.69
Fullpart	0.375	0.24	0.39
StructParts	0.107	0.67	0.77

Table 2: **Decomposed shape generation.** Quantitative shape-level comparison on PartObjaverse-Tiny using released baseline checkpoints. †The released X-Part model uses the larger training resources noted in Tab. 1; the matched-budget comparison is reported separately in Table 1.

Method	CD ↓	F-0.05 ↑	F-0.10 ↑
OmniPart	0.088	0.56	0.75
PartCrafter	0.105	0.47	0.67
PartPacker	0.068	0.56	0.76
HoloPart	0.092	0.55	0.73
X-Part†	0.043	0.65	0.83
Fullpart	0.051	0.58	0.78
StructParts	0.045	0.63	0.83

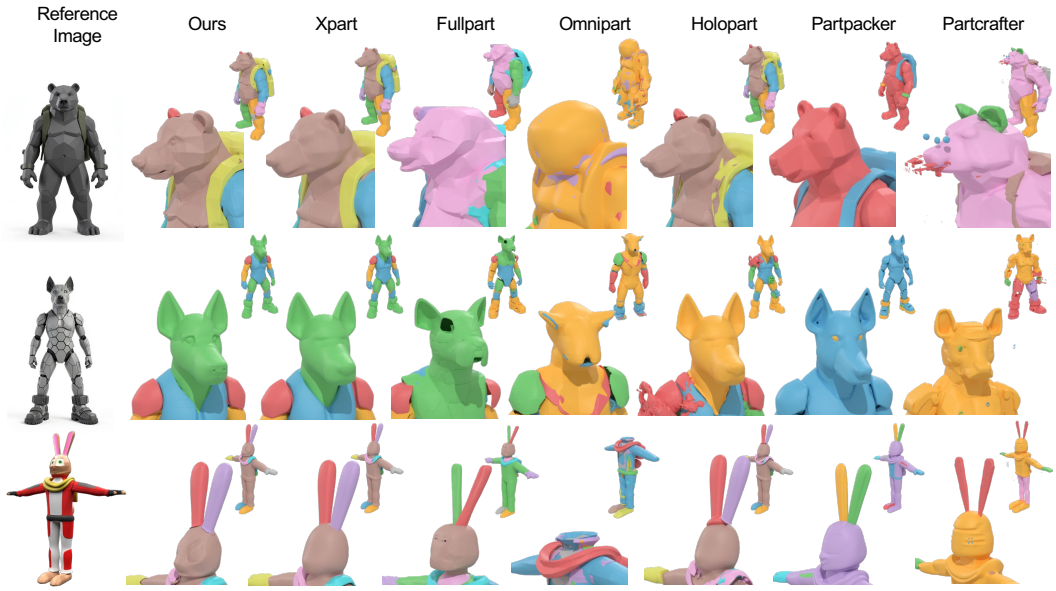


Figure 4: **Image-to-decomposed-parts comparison.** Top two rows: in-the-wild prompts generated by flux-2 [2]; bottom row: an in-distribution test sample. For each input we show the assembled shape and a zoom-in view. PartPacker / PartCrafter struggle to produce semantically meaningful decompositions; FullPart and OmniPart suffer from boundary collisions;

Table 3: **Ablation Study of dual-frame positional embedding strategies.** Quantitative results evaluated at both part-level and shape-level.

Method	Part-level			Shape-level		
	CD ↓	F-0.05 ↑	F-0.10 ↑	CD ↓	F-0.05 ↑	F-0.10 ↑
StructParts	0.107	0.67	0.77	0.008	0.93	0.97
GRoPE + part embed.	0.125	0.63	0.74	0.012	0.83	0.91
GRoPE only	0.143	0.63	0.75	0.019	0.70	0.90
LRoPE only	0.331	0.20	0.36	0.054	0.51	0.73

own normalized frame, severing it from the holistic prior and producing incoherent global structure. *GRoPE only* recovers global layout on simple compositions (top row) but degrades geometry on small or off-center parts under unnormalized coordinates; together these confirm that the dual frame is necessary. Adding per-part embeddings on top of GRoPE still fails to restore fine geometric detail, showing that local RoPE – not a part descriptor – is the key driver of per-part geometric quality.

Scaffold robustness. With all other settings fixed, we add normal-direction vertex noise at 1%, 2%, and 5% of the scaffold bounding-box diagonal, or relabel 5%, 10%, and 20% of boundary voxels. Whole-shape performance remains close to clean at 1% geometry noise, but F@0.05 falls from 0.63



Figure 5: **Ablation on positional-embedding strategies.** Three variants compared against StructParts' dual frame: *LRoPE only* loses global structure (parts treated identically in their own normalized frames); *GRoPE only* recovers global layout but degrades small or off-center parts under unnormalized coordinates; *GRoPE + part embed.* (as in prior work) still misses fine geometric detail. Only the dual-frame design recovers both global structure and per-part detail.

to 0.51 at 5%. Because assembled-shape metrics can hide incorrect part assignments, we additionally corrupt GT labels and evaluate corresponding parts: F@0.05 remains 0.67 at 5% corruption but falls to 0.58 at 20%. Comparing input scaffolds and generated shapes under identical normalization and surface sampling shows comparable overall geometry, without establishing high-frequency detail gains from aggregate metrics alone. Full geometry-preservation and perturbation results are in App. A.3. These tests measure local error sensitivity; substantial natural scaffold failures remain a limitation.

4.5 Applications

As discussed in Sec. 3.3, the scaffold-conditioned formulation naturally supports downstream tasks. We demonstrate two: decomposed-mesh refinement and part-level editing.

Decomposed-mesh refinement. As shown in Fig. 6, when the input scaffold [13] is coarse, X-Part [5] rigidly follows the input geometry and fails to recover detail, while StructParts synthesizes fine geometric details that faithfully match the input image. By aligning each part with the holistic shape latent space, our dual-frame design inherits the detail-recovery ability of the underlying generation prior rather than forcing the model into a new part-specific interface, suggesting potential to scale to high-quality mesh generation.

Part-level shape editing. As shown in Fig. 7, splitting or merging segmentation groups on the scaffold surface naturally yields part decompositions at different granularities.

5 Conclusion

We presented StructParts, a part-decomposed 3D generation framework that inherits both the geometric detail and the structural coherence of a pretrained holistic 3D prior. Its core mechanism,

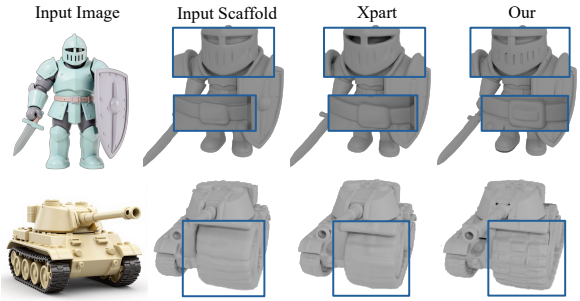


Figure 6: **Part-mesh refinement.** Given a coarse input scaffold, X-Part rigidly follows the input geometry, while StructParts recovers fine geometric detail directly from the image by leveraging the holistic prior.



Figure 7: **Part-level editing.** Merging or splitting segmentation groups on the scaffold yields decompositions at different granularities, training-free at inference time.

dual-frame coordinate interleaving, interprets each shape latent under both a part-local and an object-global reference frame, processed through a global-local interleaved transformer that reads the prior at both scales while remaining in the prior's native 3D representation. The structural input is a *segmented mesh scaffold* borrowed from coarse-mesh conditioning in holistic 3D generation, requiring only a face-level segmentation mask as adaptation. With only lightweight finetuning, StructParts matches or surpasses prior part-decomposed methods on PartObjaverse-Tiny at a fraction of their training cost, and naturally supports part-mesh refinement and part-level editing.

Limitations and future work. StructParts depends on the accuracy of the input scaffold. Although it can recover geometric detail, it does not reliably correct an inaccurate scaffold, and scaffold errors can propagate to the generated parts. The controlled perturbations in App. A.3 quantify local sensitivity, rather than the full range of naturally occurring scaffold inaccuracies. Representative examples of inaccurate scaffolds and their generated parts are shown in Fig. 8. Promising next steps include extending dual-frame interleaving to deeper hierarchies (e.g., sub-parts) and applying structured-representation reuse to articulated-object generation. Broader societal impacts are discussed in App. A.6.

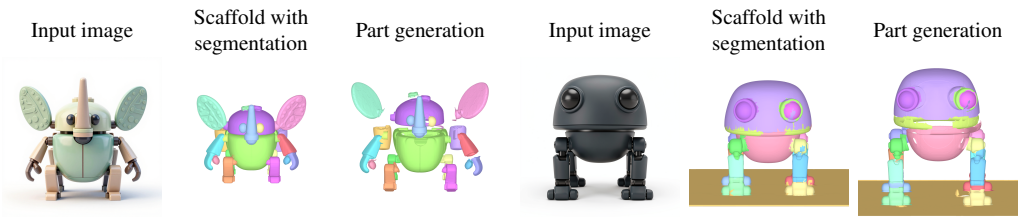


Figure 8: **Limitations: inaccurate scaffolds.** Each example shows the input image, scaffold with segmentation, and exploded generation. Although our method generates geometric detail, inaccuracies in the input scaffold can persist in the generated parts.

Acknowledgments and Disclosure of Funding

This work is partially supported by funding and computing resources provided by the HKU-Shanghai Intelligent Computing Research Center. This work was supported by the General Research Fund of the Research Grants Council of Hong Kong under Grant No. 17200725 and the JC STEM Lab funded by The Hong Kong Jockey Club Charities Trust.

References

[1] Z-Image Team, Alibaba Group. Z-image: An efficient image generation foundation model with single-stream diffusion transformer, 2025. 1

[2] Black Forest Labs. Flux.2: Frontier visual intelligence. https://bfl.ai/blog/flux-2, 2025. Accessed: 2026-05-02. 1, 8

[3] 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. *CVPR*, 2026. 2, 3, 4

[4] 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. *Tech report*, 2025. 2, 3

[5] Xinhao Yan, Jiachen Xu, Yang Li, Changfeng Ma, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zeqiang Lai, Yunfei Zhao, Zhuo Chen, et al. X-part: High fidelity and structure coherent shape decomposition. *CVPR*, 2026. 2, 3, 6, 7, 8, 9, 14

[6] Yunhan Yang, Yuan-Chen Guo, Yukun Huang, Zi-Xin Zou, Zhipeng Yu, Yangguang Li, Yan-Pei Cao, and Xihui Liu. Holopart: Generative 3d part amodal segmentation. *arXiv preprint arXiv:2504.07943*, 2025. 2, 3, 6

[7] Longwen Zhang, Qixuan Zhang, Haoran Jiang, Yinuo Bai, Wei Yang, Lan Xu, and Jingyi Yu. Bang: Dividing 3d assets via generative exploded dynamics. *ACM Transactions on Graphics (TOG)*, 44(4):1–21, 2025. 2, 3

[8] Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, Zi-Xin Zou, Yukun Huang, Ying-Tian Liu, Hao Xu, Ding Liang, Yan-Pei Cao, and Xihui Liu. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. *SIGGRAPH Asia*, 2025. 2, 3, 6

[9] Lihe Ding, Shaocong Dong, Yaokun Li, Chenjian Gao, Xiao Chen, Rui Han, Yihao Kuang, Hong Zhang, Bo Huang, Zhanpeng Huang, Zibin Wang, Dan Xu, and Tianfan Xue. Fullpart: Generating each 3d part at full resolution. *ICLR*, 2026. 2, 3, 6, 14

[10] Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Yiqiang Feng, Yadong Mu, and Katerina Fragkiadaki. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. *NeuRIPS*, 2025. 2, 3, 6, 7

[11] Jiaxiang Tang, Ruijie Lu, Zhaoshuo Li, Zekun Hao, Xuan Li, Fangyin Wei, Shuran Song, Gang Zeng, Ming-Yu Liu, and Tsung-Yi Lin. Efficient part-level 3d object generation via dual volume packing. *NeuRIPS*, 2025. 2, 3, 6, 14

[12] Tanghui Jia, Dongyu Yan, Dehao Hao, Yang Li, Kaiyi Zhang, Xianyi He, Lanjiong Li, Jinnan Chen, Lutao Jiang, Qishen Yin, Long Quan, Ying-Cong Chen, and Li Yuan. Ultrashape 1.0: High-fidelity 3d shape generation via scalable geometric refinement. *arXiv preprint arXiv:2512.21185*, 2025. 2, 3, 4, 17

[13] Tencent Hunyuan3D Team. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material, 2025. 3, 4, 6, 9, 14

[14] Changfeng Ma, Yang Li, Xinhao Yan, Jiachen Xu, Yunhan Yang, Chunshi Wang, Zibo Zhao, Yanwen Guo, Zhuo Chen, and Chunchao Guo. P3-sam: Native 3d part segmentation. *arXiv preprint arXiv:2509.06784*, 2025. 3, 4, 6, 14

[15] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. 3

[16] 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. *CVPR*, 2024. 3, 6, 14

[17] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3, 4

[18] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *ACM Transactions on Graphics (TOG)*, 43(4):1–20, 2024. 3

[19] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3d shape variational auto-encoders. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR)*, pages 16251–16261, June 2025. 3, 4

[20] Tencent Hunyuan3D Team. Hunyuan3d 1.0: A unified framework for text-to-3d and image-to-3d generation, 2024. 3

[21] Tencent Hunyuan3D Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025. 3, 4

[22] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 3

[23] Mutian Xu, Xingyilang Yin, Lingteng Qiu, Yang Liu, Xin Tong, and Xiaoguang Han. Sampro3d: Locating sam prompts in 3d for zero-shot instance segmentation. In *International Conference on 3D Vision (3DV)*, 2025. 3

[24] Songyou Peng, Kyle Genova, Chiyu Max Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. *arXiv:2304.02643*, 2023. 3

[26] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 3, 5, 14

[27] 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. 3, 6

[28] Yuchen Zhou, Jiayuan Gu, Tung Yen Chiang, Fanbo Xiang, and Hao Su. Point-SAM: Promptable 3d segmentation model for point clouds. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[29] Yiqun Zhao, Jiachen Xu, Yihao Zhi, Yunfei Zhao, Zibo Zhao, Xintong Han, Chunchao Guo, and Shenghua Gao. Flexpart: Text-controlled semantics-aware 3d part segmentation at flexible granularity. *arXiv preprint arXiv:2610.23659*, 2026. 3

[30] Minghao Chen, Jianyuan Wang, Roman Shapovalov, Tom Monnier, Hyunyoung Jung, Dilin Wang, Rakesh Ranjan, Iro Laina, and Andrea Vedaldi. Autopartgen: Autogressive 3d part generation and discovery. *arXiv preprint arXiv:2507.13346*, 2025. 3

[31] Minghao Chen, Roman Shapovalov, Iro Laina, Tom Monnier, Jianyuan Wang, David Novotny, and Andrea Vedaldi. Partgen: Part-level 3d generation and reconstruction with multi-view diffusion models. *arXiv preprint arXiv:2412.18608*, 2024. 3

[32] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code, 2024. 4, 5

[33] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2021. 4, 14

[34] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 5

[35] Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11461–11471, June 2022. 6

[36] Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. YaRN: Efficient context window extension of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. 14

A Technical appendices and supplementary material

A.1 Implementation Details

Dataset and scaffold construction. We train on Partverse-XL [9], which contains 36K objects with about 240K parts in total. Because the raw per-part meshes in Partverse-XL are not watertight, we apply cuBVH-based watertight remeshing [11] to each part before training. For each object, 24 conditioning views are rendered following Trellis [16].

To obtain segmented watertight scaffolds for the full object, we (i) watertight-remesh the entire shape, (ii) bake multi-view rendered segmentation labels from the raw (non-watertight) mesh onto the watertight surface using a multi-view depth-fusion procedure [16], and (iii) filter out edge labels that fall outside the ground-truth per-part bounding boxes to clean up boundary noise. The result is a 3D-consistent watertight mesh whose face labels match the original part decomposition.

Network architecture. Our DiT backbone follows Hunyuan3D-2.1 [13]: 21 layers, hidden dimension 2048, 16 attention heads. Self-attention alternates by layer parity: even layers are intra-part (tokens attend only within their own part) and odd layers are inter-part (tokens attend across all parts). Each token carries a 3D voxel coordinate encoded by two-scale rotary positional embeddings: a part-local 128^3 grid in intra-part layers and a shape-global 512^3 grid in inter-part layers. The shape-global RoPE uses NTK-aware scaling [36, 33] with rotary base $\theta = 40$k, a $4\times$ scaling of the default $\theta = 10$k chosen to accommodate the larger 512^3 position range. The last 6 blocks replace the dense feed-forward layer with a Mixture-of-Experts feed-forward (8 experts, top-2 routing) to enhance latent-space capacity. All attention layers use FlashAttention-3 kernels.

Training procedure. The DiT is initialized from open-source Hunyuan3D-2.1's ShapeDiT [13] weights and fine-tuned on part-level data using a rectified-flow-matching objective with a linear noise schedule and velocity prediction. We use AdamW with learning rate $1e{-}4$ and weight decay 10^{-2}, in bfloat16 mixed precision (forward/backward in bf16; master weights and optimizer states in fp32) with gradient-norm clipping at 1.0. During training we fix the per-part token budget to 512, though the model can scale to more tokens at inference. The DiT is trained on 8 GPUs for 40K steps within a 768-H200-hour allocation (8 H200 GPUs for four days).

VAE finetuning. The VAE component is finetuned with a token length of 512 to ensure it can encode and decode individual parts effectively at the per-part scale used during DiT training.

Inference pipeline. At inference, the segmented mesh scaffold is obtained from a single image via two off-the-shelf upstream stages: a holistic 3D generation model [13] produces the coarse mesh, and a 3D segmentation model [14] provides the per-face part labels. The image is encoded by DINOv2 [26]; per-part shape latents are sampled by the rectified-flow scheduler conditioned on dual-frame RoPE indices derived from the scaffold and on the image features. Because the dual-frame embeddings only require coarse voxel indices and the image condition is retained in every block, low-fidelity scaffolds are sufficient: missing geometric detail is recovered directly from the image (see the part-mesh refinement application in Sec. 3.3).

A.2 Matched-Budget Comparison with X-Part

To verify training efficiency under a controlled resource budget, we retrain X-Part [5] using the same initialization, training data, and token budget as StructParts (Sec. 4.2). Both runs use 8 H200 GPUs for four days. The X-Part implementation and hyperparameters follow the setup confirmed with its authors; the retrained model is distinguished from its released checkpoint throughout the evaluation.

Computation and conditioning. To measure the cost of the two adaptation strategies, we profile both methods with the same PyTorch profiler, including forward and backward computation. Tab. 4 summarizes the shared training allocation and per-object costs. StructParts uses 32.5% fewer TFLOPs per object and less memory. X-Part uses learned global-object and per-part local-geometry conditioners, fuses semantic features into both, and injects two 4,096-token streams through separate cross-attention branches in every DiT block. StructParts instead derives dual-frame coordinates from scaffold voxel indices and reuses object-level image conditioning across parts. Computing X-Part conditioning online per part partly explains its approximately $1.5\times$ longer step time. We

retain the author-confirmed online point-sampling setup, since fixing points offline can introduce a train–inference mismatch.

Table 4: **Training allocation and computational cost.** Both methods use 512 tokens per part and the same GPU-hour allocation. TFLOPs/object include forward and backward computation under the same PyTorch profiler. Optimizer steps and processed-part counts are reported with checkpoint quality in Tab. 5.

Quantity	StructParts	X-Part (matched)
GPUs	8 H200	8 H200
Total allocation	768 GPU-hours	768 GPU-hours
Trainable parameters	3.051 B	3.315 B
TFLOPs/object	92.7	137.4
Allocated GPU memory	82.71 GB	115 GB

Checkpoint comparison. To verify whether the quality advantage requires more optimization steps, we evaluate the 20K and 40K StructParts checkpoints alongside the 28K matched-budget X-Part model using the same part-level protocol. Tab. 5 reports quality, optimizer steps, and processed-part counts together. Even the 20K StructParts checkpoint performs better on all four metrics after fewer steps and fewer processed parts; quality improves further at 40K. The full-budget comparison supports training efficiency, while the earlier checkpoint shows that the advantage does not require a higher optimizer-step count.

Table 5: **Checkpoint quality and training-schedule accounting.** Part-level evaluation on PartObjaverse-Tiny. The 40K StructParts and 28K X-Part runs use the same GPU-hour allocation; the 20K StructParts checkpoint is included to compare quality after fewer optimizer steps and processed parts.

Run	Steps	Processed parts	Mean CD ↓	Median CD ↓	F@0.05 ↑	F@0.10 ↑
StructParts	20K	2.56M	0.129	0.013	0.64	0.76
X-Part (matched)	28K	3.59M	0.169	0.028	0.55	0.69
StructParts	40K	5.12M	0.107	0.014	0.67	0.77

A.3 Scaffold Robustness and Geometry Preservation

Geometry preservation and whole-shape sensitivity. To verify geometry preservation and quantify sensitivity to upstream errors, we evaluate the input Hunyuan scaffold and generated outputs against the same GT objects with identical normalization and surface sampling. Tab. 6 reports the input scaffold, clean output, and outputs under two controlled perturbations. With all other settings fixed, we add normal-direction vertex noise at 1%, 2%, or 5% of the scaffold bounding-box diagonal, or relabel 5%, 10%, or 20% of boundary voxels. The clean output preserves overall geometry; aggregate metrics do not isolate high-frequency detail, which is assessed qualitatively in Fig. 6. Geometry noise causes little change at 1%, moderate degradation at 2%, and substantial degradation at 5%. Label corruption changes whole-shape metrics only slightly, indicating that assembly can conceal incorrect part boundaries.

Part-level sensitivity. To expose segmentation errors hidden by assembly, we apply the same label-corruption rates to GT part labels and compare each generated part with its corresponding GT part. GT labels retain a known correspondence, whereas P3-SAM part IDs cannot be reliably matched to GT part IDs. Tab. 7 shows nearly unchanged performance at 5% corruption and progressively worse part consistency at 10–20%. These controlled perturbations assess local error sensitivity; the dependence on reasonable coarse topology and the effects of natural structural failures are discussed in Sec. 5.

A.4 Cross-Dataset and Category-Wise Evaluation

Cross-dataset evaluation. To verify generalization beyond PartObjaverse-Tiny, we evaluate a fixed 200-object ABO subset using identical inputs and the same whole-shape CD/F-score protocol for all

Table 6: **Geometry preservation and scaffold perturbations.** The input scaffold and generated outputs are evaluated against the same GT shapes with identical normalization and surface sampling. Noise is normal-direction vertex displacement as a percentage of the bounding-box diagonal; label corruption reassigns boundary voxels. All other inference settings are fixed.

Method / condition	Mean CD ↓	Median CD ↓	F@0.05 ↑	F@0.10 ↑
Input Hunyuan scaffold	0.049	0.038	0.63	0.84
StructParts (clean)	0.045	0.037	0.63	0.83
Geometry noise 1%	0.046	0.037	0.63	0.83
Geometry noise 2%	0.047	0.039	0.61	0.80
Geometry noise 5%	0.052	0.041	0.51	0.73
Label corruption 5%	0.045	0.037	0.62	0.83
Label corruption 10%	0.046	0.037	0.61	0.82
Label corruption 20%	0.047	0.036	0.61	0.82

Table 7: **Part-level sensitivity to segmentation errors.** We corrupt GT boundary labels to retain a known correspondence between each generated and GT part. These results are distinct from the assembled-shape experiment in Tab. 6.

Method / condition	Mean CD ↓	Median CD ↓	F@0.05 ↑	F@0.10 ↑
Clean	0.107	0.014	0.67	0.77
Label corruption 5%	0.108	0.014	0.67	0.77
Label corruption 10%	0.113	0.019	0.61	0.73
Label corruption 20%	0.117	0.038	0.58	0.71

methods. Tab. 8 separates released and matched-budget X-Part checkpoints. StructParts achieves the lowest mean and median CD, matches released X-Part at F@0.05, and is within 0.01 at F@0.10.

Table 8: **Cross-dataset evaluation on ABO.** A fixed 200-object subset is evaluated with identical inputs and the same whole-shape CD/F-score protocol for all methods. Released and matched-budget X-Part checkpoints are distinguished.

Method / condition	Mean CD ↓	Median CD ↓	F@0.05 ↑	F@0.10 ↑
StructParts	**0.021**	**0.012**	**0.68**	0.79
X-Part (released)	0.023	0.013	**0.68**	**0.80**
X-Part (matched)	0.045	0.037	0.58	0.69
HoloPart	0.026	0.018	0.62	0.70
FullPart	0.028	0.021	0.66	0.71
OmniPart	0.095	0.057	0.49	0.49

Category consistency and data separation. To assess whether the gain is concentrated in a few categories, we report part-level CD for the six largest evaluation categories in Tab. 9, and object counts, paired-part counts, and F-scores in Tab. 10. StructParts has the lowest median CD in all six categories, the lowest mean CD in five, and leads both F-scores in four. The higher Animals mean nevertheless indicates a heavy-tailed failure. All evaluation UIDs are excluded from training, yielding zero object overlap; PartVerse-XL lacks an aligned taxonomy, so semantic-category overlap cannot be measured objectively.

Paired uncertainty estimates. To assess statistical reliability against released X-Part, we perform a paired bootstrap over objects with complete matched predictions. We resample objects with replacement 10,000 times, retaining all paired parts within each sampled object. Tab. 11 reports StructParts-minus-X-Part differences and 95% intervals; negative CD and positive F-score differences favor StructParts. Only the F@0.05 interval excludes zero. Statistical conclusions use the unrounded intervals; displayed F-score values use two decimals.

Table 9: **Category-wise part-level CD on PartObjaverse-Tiny.** Each cell reports mean / median CD, using the six largest evaluation categories. X-Part denotes its released checkpoint.

Category	FullPart	HoloPart	OmniPart	X-Part	StructParts
Animals	0.395/0.317	0.120/0.031	0.182/0.205	**0.040**/0.013	0.095/**0.009**
Buildings & Outdoor	0.483/0.389	0.166/0.068	0.403/0.386	0.100/0.027	**0.068/0.013**
Daily-used	0.259/0.122	0.199/0.043	0.229/0.191	0.119/0.024	**0.066/0.011**
Electronics	0.453/0.256	0.150/0.061	0.165/0.146	0.152/0.029	**0.137/0.022**
Human-shaped	0.332/0.273	0.081/0.028	0.177/0.148	0.049/0.011	**0.041/0.009**
Transportation	0.377/0.238	0.181/0.065	0.332/0.211	0.208/0.044	**0.144/0.022**

Table 10: **Category counts and paired-part F-scores.** Object counts describe the evaluation set; paired parts have complete matched predictions. F-score pairs are F@0.05 / F@0.10. X-Part denotes its released checkpoint.

Category	Objects	Paired parts	StructParts	X-Part
Human-Shape	29	423	0.74/0.83	0.74/0.85
Animals	23	344	**0.74**/0.82	0.72/0.84
Daily-Use	25	335	**0.71/0.81**	0.62/0.74
Buildings & Outdoor	25	370	**0.67/0.79**	0.61/0.72
Transportation	38	615	**0.59/0.70**	0.52/0.64
Electronics	34	523	**0.63/0.74**	0.58/0.71

Table 11: **Paired bootstrap differences against released X-Part.** Objects with complete paired predictions are resampled 10,000 times, retaining all paired parts within each sampled object. Differences are StructParts minus X-Part. F-score estimates and interval bounds use two decimals; a negative bound that rounds to zero retains its sign. Statistical conclusions use the unrounded intervals.

Metric difference	Estimate	95% paired bootstrap interval
Mean CD	-0.0089	$[-0.0198, +0.0009]$
F@0.05	$+0.04$	$[+0.02, +0.06]$
F@0.10	$+0.03$	$[-0.00, +0.05]$

A.5 Initialization Dependence and Method Comparison

Alternative initialization. To verify dependence on the Hunyuan3D-2.1 prior, we also initialize StructParts from UltraShape [12], whose DiT is pretrained with a coarse-mesh anchor. Since the compatible UltraShape DiT requires retraining, we compare both initializations at the same 20K halfway training point using part-level metrics. UltraShape achieves mean/median CD of 0.123/0.015 and F@0.05/F@0.10 of 0.64/0.77; the Hunyuan3D-2.1 baseline is the 20K StructParts checkpoint in Tab. 5. The nearly identical F-scores support adaptation of both initializations at this common training point.

Comparison with UniPart. To clarify the distinction from dual-space part representations, we compare the two formulations. UniPart first generates and segments a whole-object geometry–segmentation latent, then predicts separate global and canonical part latents through two-stage diffusion. StructParts instead processes one latent within one DiT under alternating part-local and object-global frames, accepting an editable segmented scaffold for training-free merge/split editing. This comparison is conceptual; UniPart code was unavailable at the time of evaluation, preventing a matched quantitative comparison.

A.6 Social Impact

Positive impact. StructParts produces explicitly part-decomposed 3D assets, enabling part-aware downstream workflows such as localized editing, animation rigging, retopology, and semantic asset reuse. By reusing a pretrained holistic 3D prior with only lightweight finetuning, the method avoids the energy and data costs of training a part-aware generative model from scratch, lowering the carbon footprint of part-decomposed 3D research and making this capability accessible to teams without large-scale compute infrastructure. The scaffold-conditioned interface and training-free editing

operations (merging and splitting parts via segmentation regrouping) further lower the barrier for non-experts to iterate on 3D content, with potential applications in education, indie game development, and rapid prototyping for design and engineering.

Negative impact and mitigation. Like other generative 3D models, STRUCTPARTS can be misused, for example to produce unauthorized reproductions of copyrighted 3D assets or to fabricate misleading 3D media. Because the model inherits a pretrained holistic prior, it also inherits any biases present in the upstream training data, such as over-representation of common object categories or culturally skewed design styles. To mitigate these risks, we train and release our model only with publicly available, license-compliant datasets, and we encourage downstream users to attribute reused assets and to verify generated outputs for intellectual-property and ethical concerns before deployment. The model is not designed for, and we do not recommend its use in, identity-related or surveillance applications.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 state our three contributions (the dual-frame coordinate interleaving mechanism, the segmented mesh scaffold adaptation, and the empirical evaluation on PartObjaverse-Tiny), each of which is supported by the corresponding sections in Sec. 3 and Sec. 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: Limitations are discussed in Sec. 5 and the supplementary appendix. In particular, our method depends on the quality of the upstream segmented mesh scaffold (a coarse mesh from a holistic 3D generator plus a 3D segmentation model), and inherits any failure modes of those upstream components.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results; our contributions are an architectural mechanism (dual-frame coordinate interleaving) and an empirical evaluation.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The architecture (segmented mesh scaffold and dual-frame coordinate interleaving), training procedure, hyperparameters, and evaluation protocol are described in Sec. 3 and Sec. 4; further implementation details (data preprocessing, scaffold construction, evaluation setup) are provided in App. A.1.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We will release our code and trained model checkpoints upon acceptance, along with instructions for reproducing the main experiments. The training and evaluation datasets used (Objaverse-XL, PartVerse-XL, PartObjaverse-Tiny) are publicly available, as documented in Sec. 4.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).

- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4 reports the optimizer (AdamW, learning rate 10^{-4}, weight decay 10^{-2}), training schedule (rectified-flow matching, linear schedule, velocity prediction, bfloat16 mixed precision, gradient-norm clipping at 1.0), batch size, number of training steps, dataset splits, and evaluation metrics; additional details are in App. A.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Following standard practice in 3D generative modeling, we report metrics from a single training run, since rerunning the full training pipeline with multiple seeds would substantially exceed our compute budget. We mitigate this by evaluating with multiple complementary metrics (Chamfer Distance and F-Score at thresholds 0.05 and 0.1) at both part-level and object-level granularities, as detailed in Sec. 4.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).

- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 4 reports our training compute (8 GPUs, $\sim$4 days, batch size 128 per rank, $\sim$32 GPU-days total). For comparison, baseline training-compute figures are reproduced from each method's original publication.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The work uses publicly released 3D-asset datasets and pretrained models, and conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Both positive and negative societal impacts are discussed in App. A.6.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The model performs object-centric 3D reconstruction from images and does not pose high-risk dual-use concerns beyond the standard generative-modeling considerations noted in App. A.6

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All used datasets (Objaverse-XL, PartVerse-XL,) and pretrained models (Ultrashape, DINOv2) are publicly available under their respective licenses and are properly cited; baselines (X-part, HoloPart, Omnipart, Fullpart, Partpacker, Partcrafter) are cited and used per their public release terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We will release the trained our model checkpoints together with the codebase upon acceptance. Documentation of the architecture, training pipeline, and inference protocol is provided in Sec. 3.1–4 and the supplementary appendices.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were not used as an important, original, or non-standard component of the method.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.