PARTSYNTH: GENERATIVE SPARSE PART STRUCTURES FOR COMPOSITIONAL 3D GENERATION WITH PBR MATERIALS

Anonymous authors
Paper under double-blind review

ABSTRACT

High-fidelity compositional 3D assets require separable parts, detailed geometry, and physically based rendering materials. Existing methods either constrain part counts or incur inference costs that grow with the number of parts. Monolithic material generators also lack explicit part awareness. We introduce PARTSYNTH, an image-conditioned framework for compositional 3D geometry and native part-aware PBR generation. Its core representation, the *Sparse Part Structure* (SPS), uses reusable set-indexed latent slots to represent multi-part structure, jointly generating each slot's activity and feature. Multiple active slots at the same spatial location represent overlapping components, while reusing set indices across disconnected components supports flexible part counts. Connected components of the decoded set-indexed volumes define individual parts. The SPS-guided geometry and PBR stages adapt pretrained monolithic generators to produce detailed multi-part geometry with PBR materials. Experiments show improved part-level geometry and PBR material quality over part-generation baselines, while maintaining competitive whole-shape fidelity.

1 INTRODUCTION

Multi-part 3D assets support editing, reuse, articulation, and assembly in visual and interactive applications. Unlike monolithic shapes, they vary in part count and structure, and their components may contact or overlap (common in game assets). Their generation must jointly produce a plausible whole shape, a flexible multi-part structure, coherent part geometry, and distinct PBR materials.

Recent generative 3D models have advanced high-quality shape and appearance synthesis, with several approaches additionally supporting native PBR materials (Zhang et al., 2024; Xiang et al., 2025; 2026; Hunyuan3D, 2025). However, existing approaches constrain part counts (Lin et al., 2025; Dong et al., 2025; Yang et al., 2025; Ding et al., 2026), incur the costs of sequential generation (Chen et al., 2026; Li et al., 2026a; Liao et al., 2026), depend on segmentation and completion (Chen et al., 2025b; Yang et al., 2026; Yan et al., 2026), or merge components to fit two packed volumes (Tang et al., 2025). Monolithic PBR generators such as TRELLIS.2 (Xiang et al., 2026) also lack explicit part awareness and can exhibit material bleeding across interfaces between nearby components. These limitations leave efficient joint generation of flexible part structures with preserved complex contacts and part-aware PBR insufficiently addressed.

Meeting these requirements calls for a compact representation of an unordered and potentially overlapping set of parts. We address this with a coarse-to-fine factorization that generates a *Sparse Part Structure* (SPS) before detailed geometry and materials. Given a generated coarse sparse scaffold, SPS assigns K set-indexed slots to each occupied location and jointly generates their activity and features. Its decoder reconstructs up to K sparse volumes, allowing multiple sets to occupy the same location. Following the dual-volume packing idea (Tang et al., 2025), contacting parts use different sets, while disconnected components can reuse a set and are identified by voxel connectivity in the decoded sub-scaffolds before detailed geometry generation. This reuse allows a bounded set space to represent many parts without fixing the output part count. We use $K{=}8$, which supports more complex part relationships than dual-volume packing covers most assets without part merging and

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

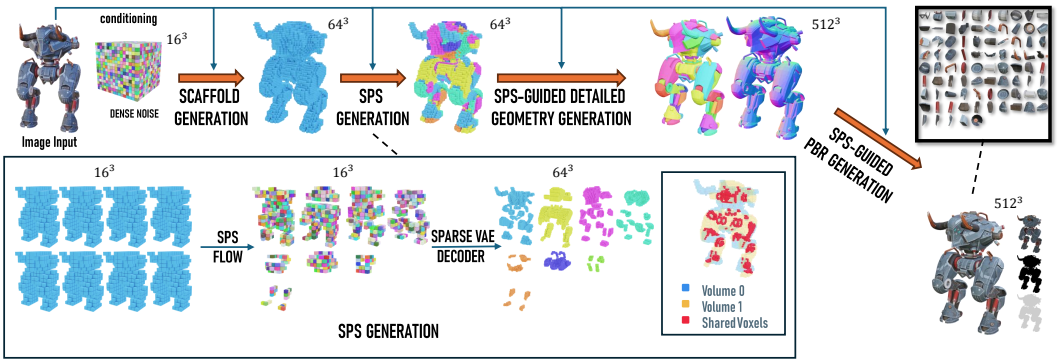

Figure 1: **Framework of PARTSYNTH.** Given an input image and a noisy 16^3 latent, PARTSYNTH generates a sparse 64^3 scaffold, a Sparse Part Structure (SPS), detailed geometry, and PBR materials. The lower panel expands the encoded scaffold into $K{=}8$ set-indexed slots per occupied location; one decoded set volume is empty in this example. The inset highlights voxels shared by two set volumes in red. The upper panel shows the geometry in part colors and as a normal-map render, a gallery of separated, individually size-normalized parts (top right), and the final PBR render with base-color, metallic, and roughness maps (bottom right).

geometry modification. SPS remains efficient because its K-dependent generation operates only on the sparse scaffold.

Concretely, PARTSYNTH is a four-stage coarse-to-fine framework: *given a single image, it first generates a coarse whole-shape scaffold (Section 3.1), expands its sparse support into K set-indexed latent slots and jointly generates their activity and features (Section 3.2), synthesizes detailed geometry for the set-indexed volumes, and finally generates PBR materials for the multi-part geometry* (Section 3.4). Fig. 1 illustrates this process. Our formulation applies to voxel-based generators that provide aligned sparse structure, geometry, and material latent spaces; we instantiate it with TRELLIS.2. The scaffold stage uses its pretrained sparse-structure model, while the SPS-guided geometry and PBR stages initialize from the corresponding pretrained models and extend them with our sparse part structure. Experiments show that PARTSYNTH improves part-level geometry and PBR quality over the representative part-generation baselines while maintaining competitive whole-shape fidelity.

Our contributions are twofold:

- A **generative Sparse Part Structure** that represents multi-part structure with reusable set-indexed latent slots and jointly generates each slot's activity and latent feature. This compact representation supports flexible part counts and overlapping components.
- A **SPS-guided framework for detailed part geometry and part-aware PBR generation** that adapts pretrained monolithic models with SPS guidance, improving part-level geometry and material quality while reducing measured material error near part interfaces.

2 RELATED WORK

Monolithic 3D Generation Dense volumetric generation incurs cubic storage growth, motivating compact latent sets (Zhang et al., 2023; 2024; Zhang & Wonka, 2025) and sparse or coarse-to-fine spatial representations (Zheng et al., 2023; Ren et al., 2024; Li et al., 2026b; He et al., 2025; Lai et al., 2026). TRELLIS (Xiang et al., 2025) generates a sparse structure before its structured latent features, while TRELLIS.2 (Xiang et al., 2026) represents geometry and PBR attributes with native sparse O-Voxels. These approaches generate whole assets in a monolithic manner.

PBR Material Generation Existing methods obtain surface appearance or PBR materials through diffusion-guided optimization or multi-view synthesis followed by projection or reconstruction (Chen et al., 2023; Zeng et al., 2024; Huang et al., 2025; Engelhardt et al., 2025; Yeo et al., 2026; Bala et al., 2024; Siddiqui et al., 2024; Ye et al., 2025; Chen et al., 2025d), or generate native 3D

PBR materials directly (Chen et al., 2025c; Xiang et al., 2026). These methods generally treat an asset monolithically, without explicit part-indexed material variables, and may blur material boundaries between nearby components. Our method instead generates part-aware PBR latents, using set-indexed coordinates to distinguish components sharing spatial support.

Multi-Part Generation Early methods represent part relations with hierarchies, graphs, or sequences and are commonly category-specific (Li et al., 2017; Mo et al., 2019; Wu et al., 2020). Other approaches decompose existing shapes (Liu et al., 2025; Yang et al., 2024; Ma et al., 2025) or reconstruct parts from segmentation-derived conditions (Liu et al., 2024; Chen et al., 2025b; Yang et al., 2026; Yan et al., 2026). Recent end-to-end methods either jointly generate component-specific latent groups (Lin et al., 2025; Dong et al., 2025; Li et al., 2025b), jointly infer geometry and latent segmentation before part refinement (He et al., 2026), generate box layouts before geometry (Yang et al., 2025; Ding et al., 2026), or generate parts sequentially at a cost that grows with the part count (Chen et al., 2026; Li et al., 2026a; Liao et al., 2026).

Volume Packing PartPacker (Tang et al., 2025) packs parts into two SDF/VecSet volumes, requiring part merging for non-bipartite contact graphs and assuming watertight geometry. KaiNinja (Yu et al., 2026) (concurrent to our work) transfers this dual-volume design to two TRELLIS.2 O-Voxel streams that persist through geometry generation, while its material model remains part-agnostic. In contrast, SPS uses K reusable sets only in a compact generated structural latent; its decoded set volumes guide joint detailed geometry and part-aware PBR generation. This separation accommodates more complex contacts with significantly lower memory and computation overhead (Appendix C).

3 METHODOLOGY

3.1 SPARSE WHOLE-SHAPE SCAFFOLD GENERATION

Directly generating a collection of parts, represented by boxes or independent part latents, requires a model to determine the number, scale, and placement of all components simultaneously. This becomes difficult for assets with many heterogeneous or overlapping parts. We instead first establish the coarse support of the whole shape with an image-conditioned monolithic generator and then generate the part structure only over this support.

In our implementation, the pretrained TRELLIS.2 sparse-structure model generates a *sparse whole-shape scaffold* U, a sparse binary grid at 64^3 resolution. A sparse variational autoencoder, $\mathrm{VAE}_{\mathrm{sparse}}$, compresses U into a feature-bearing latent support Ω on a sparse 16^3 grid (see Appendix A.1 for details).

3.2 GENERATIVE SPARSE PART STRUCTURE

Set-Indexed Sub-Scaffold Construction For each multi-part training asset, we rasterize every part surface on the scaffold grid. A voxel may belong to multiple parts when their surfaces intersect the same cell. We construct a part-contact graph whose nodes are parts and whose edges connect parts that either co-occur in a voxel or occupy face-adjacent voxels. A graph-coloring algorithm (Brélaz, 1979) assigns contacting parts to different sets within a budget of K. Propagating these assignments to the voxelized parts decomposes the whole-shape scaffold into $U = \bigcup_{k=0}^{K-1} U_k$, where each U_k contains mutually non-contacting parts. Different set volumes may overlap, *i.e.*, $U_i \cap U_j \neq \varnothing$ for some $i \neq j$, preserving parts that occupy the same spatial location. Spatially disconnected parts may reuse a set index and can later be separated through connected-component extraction. We call this decomposition the *set-indexed sub-scaffold*.

The required number of sets depends on the asset's contact structure. PartPacker's dual-volume packing (Tang et al., 2025) corresponds to $K{=}2$ and must contract non-bipartite contact graphs by merging parts. We use a fixed maximum budget of $K{=}8$, which covers most processed decompositions while keeping the set dimension compact (see Section 4.1 for this choice).

Sparse Part Structure Encoding each sub-scaffold U_k with $\mathrm{VAE}_{\mathrm{sparse}}$ produces an active latent support $\Omega_k \subseteq \Omega$ and a feature vector $\mathbf{f}_{k,\mathbf{u}} \in \mathbb{R}^d$ at each $\mathbf{u} \in \Omega_k$. We represent all sets on the shared

latent support using the *Sparse Part Structure* (SPS)

$$\mathcal{Z}_{\mathrm{SPS}} = \{(k, \mathbf{u}, a_{k,\mathbf{u}}, \mathbf{f}_{k,\mathbf{u}}) \,|\, k \in \{0, 1, \ldots, K-1\},\ \mathbf{u} \in \Omega\}\,, \tag{1}$$

where $a_{k,\mathbf{u}} \in \{0, 1\}$ is the slot activity and $\mathbf{f}_{k,\mathbf{u}}$ is its latent feature. Decoding the active slots for each non-empty set k with $\mathrm{VAE}_{\mathrm{sparse}}$ reconstructs the corresponding sub-scaffold U_k. Here, the set index k is randomly assigned to each set; no semantic meaning is implied by its specific value.

SPS Flow Model Given the input image I and scaffold support Ω, the SPS flow-matching model jointly generates the activity and feature of every slot in Eq. (1). It instantiates K candidates at each occupied location and associates them with 4D coordinates (k, x, y, z). Self-attention uses 4D rotary positional encoding (RoPE) to distinguish both spatial locations and set indices. We follow standard flow-matching training (Lipman et al., 2023) to recover the clean slot features and activities from noisy observations. Appendix A.1 details the architecture and implementation.

Scaffold Conditioning Encoding U with the VAE produces a scaffold latent representation spatially aligned with the set-indexed slots. We broadcast each scaffold feature to all set-indexed slots at the same spatial location and concatenate it with their noisy activity and latent features along the channel dimension before the flow model's input projection. We find that this conditioning improves consistency between the generated and target set features and accelerates convergence.

Set-wise Decoding After flow sampling, we discard inactive slots and decode the retained features of each set independently with the shared $\mathrm{VAE}_{\mathrm{sparse}}$ decoder. This produces up to K sparse volumes $\{\mathcal{V}_k\}_{k=0}^{K-1}$ at 64^3 resolution. We define each part as a voxel-connected component within a decoded set volume. These part identities are retained during detailed geometry generation (Section 3.3), even if a part yields multiple disconnected surface components: such disconnections may be genuine or caused by decoding-induced artifacts, which final-mesh connectivity alone cannot distinguish. The decoded set volumes guide the subsequent geometry and PBR stages.

3.3 SPS-Guided Detailed Geometry Generation

We instantiate this stage by adapting the corresponding pretrained TRELLIS.2 flow-matching model to generate detailed multi-part geometry. The original TRELLIS.2 model uses 3D RoPE over spatial coordinates (x, y, z). We instead combine all set volumes into one sequence and extend the coordinates to (k, x, y, z). Joint attention allows information exchange across the complete asset while preserving set identity. We initialize the model from the pretrained geometry weights and fine-tune it on our multi-part dataset. After sampling, we decode each set independently with the pretrained TRELLIS.2 geometry VAE. The resulting mesh facets inherit their part labels from the decoded set-indexed volumes of the previous stage.

3.4 SPS-Guided PBR Generation

We adapt the pretrained TRELLIS.2 PBR flow model to jointly generate PBR attributes for all parts. The generated multi-part geometry provides the set-indexed spatial support and geometry hierarchy on which the material latents are generated. The pretrained TRELLIS.2 PBR VAE then decodes these latents into high-resolution material attributes over the corresponding multiresolution grids.

Following Section 3.3, we jointly process material tokens from all sets using coordinates (k, x, y, z). Each noisy material token is conditioned on the geometry latent from the same set and spatial location. We initialize the model from the pretrained PBR weights and fine-tune it on our dataset. After sampling, we decode each set independently with the pretrained PBR VAE and associate the resulting attributes with the parts identified in the decoded sub-scaffolds.

3.5 Monolithic Latent Conditioning

To decompose an existing monolithic asset into parts, we optionally encode its geometry and materials using the pretrained VAEs. The resulting geometry latent conditions SPS and detailed geometry generation, while the material latent conditions PBR generation. For SPS, we downsample the geometry latent through spatial-to-channel rearrangement, align it with the sparse support,

and concatenate it with the noisy slot features before the input projection. For geometry and PBR generation, we inject spatially aligned projections of the corresponding monolithic latents through zero-initialized learnable gates. Random condition dropping during training enables each model to operate with or without its monolithic latent. Implementation details are provided in Appendix A.2.

4 Experimental Analysis

4.1 Experimental Setup

Training Data We construct MegaPart3D from TRELLIS-500K (Xiang et al., 2025) and TexVerse (Zhang et al., 2025) by automatically extracting connected components from filtered GLB assets (Appendix B). The resulting 330 K multi-part assets contain $2 \sim 8412$ parts, with 39.5 parts on average. Unlike human-refined datasets such as PartVerse (Dong et al., 2025) and PartVerse-XL (Ding et al., 2026), our construction requires no part annotation and preserves source components. Following TRELLIS.2, we render 24 images per asset at 1024^2 using predefined camera poses and randomized lighting, and sample one render as the image condition.

Choice of K We evaluate graph coloring on a random sample of 1000 assets from MegaPart3D. The success rates for budgets $K \in \{2, 4, 8, 16\}$ are 30.2 %, 66.2 %, 92.5 %, and 96.4 %, respectively. A two-set budget fails on most of these assets; we choose $K = 8$ to cover most of the sample and balance computational cost for the networks.

Implementation, Training and Inference Implementation details appear in Appendix A. On a single NVIDIA A100 GPU (80 GB), the scaffold, SPS, detailed geometry, and PBR stages take 2.5 s, 2.3 s, 2.2 s, and 2.6 s per asset on average, respectively, without monolithic latent conditioning. For single-image 3D generation and fixed-geometry PBR synthesis (Sections 4.2 and 4.3), we enable monolithic PBR latent conditioning (Section 3.5). TRELLIS.2 synthesizes the whole-object PBR latent for the sparse scaffold without encoding GT materials. In contrast, compositional conversion encodes the supplied asset's PBR materials (Section 4.4). The examples in Fig. 5 show closer appearance alignment with the input image; synthesizing the latent adds approximately 1.7 s.

Evaluation Dataset Following prior work (Yang et al., 2024; Liu et al., 2025; Yang et al., 2026; Chen et al., 2026; Li et al., 2025b), we evaluate on PartObjaverse-Tiny (Yang et al., 2024), which contains 200 Objaverse assets with human-curated semantic part annotations.

4.2 Single-Image 3D Generation

We evaluate PartSynth on PartObjaverse-Tiny against two groups of baselines: monolithic generators, including TRELLIS.2, Hunyuan3D 2.1 (Hunyuan3D, 2025), and Step1X-3D (Li et al., 2025a); and multi-part generators, including PartCrafter (Lin et al., 2025), OmniPart (Yang et al., 2025), and PartPacker. We exclude FullPart (Ding et al., 2026) because its part-bounding-box generation code is unavailable, preventing end-to-end evaluation. The detailed baseline adaptations are provided in Appendix D.1.

Given a conditioning image $\mathcal{I}$, its ground-truth multi-part asset $\mathcal{G}$, and a generated asset $\mathcal{C}$, we evaluate the quality with the following metrics (see additional details in Appendix E).

Image Alignment Following TRELLIS.2, we measure alignment with the conditioning image $\mathcal{I}$ using CLIP similarity on PBR-shaded renders, CLIP_N on normal-map renders, and point-cloud–image embedding similarity from ULIP-2 (Xue et al., 2024) and Uni3D (Zhou et al., 2024).

Whole-Shape Geometry Fidelity We compare $\mathcal{C}$ and $\mathcal{G}$ using Chamfer distance (CD) and F-Score at distance thresholds 0.05 and 0.1 (F@0.05 and F@0.1). Because generated assets may differ in orientation, we evaluate four canonical yaw rotations and select the CD-minimizing alignment. The same selected alignment is used for all geometry metrics.

Part-Level Geometry Fidelity Because the TRELLIS.2 O-Voxel representation supports open surfaces, volumetric IoU is not generally reliable for our decoded geometry. We instead use CD_p,

Table 1: **Single-image 3D generation on PartObjaverse-Tiny. Bold** and underlined values indicate the best and second-best results, respectively. Dashes denote inapplicable metrics.

METHOD	IMAGE ALIGNMENT				GEOMETRY				PBR	
	$\text{CLIP}_\text{N}\uparrow$	ULIP-2 ↑	Uni3D ↑	CLIP ↑	CD ↓	F@0.05/@0.1 ↑	$\text{CD}_\text{p}\downarrow$	$\text{CD}^\star_\text{p}\downarrow$	LPIPS ↓	SSIM ↑
TRELLIS.2	0.660	0.430	**0.379**	0.770	**0.111**	**0.672/0.856**	–	–	**0.182**	**0.841**
Hunyuan3D 2.1	0.654	0.419	0.368	0.767	0.135	0.605/0.809	–	–	0.200	0.823
Step1X-3D	0.655	0.417	0.368	**0.772**	0.124	0.585/0.820	–	–	0.200	0.828
PartPacker	0.654	0.425	0.371	–	0.118	0.621/0.843	0.338	0.345	–	–
PartCrafter	0.631	0.368	0.324	–	0.175	0.436/0.684	0.406	0.443	–	–
OmniPart	0.656	0.426	0.369	0.761	0.135	0.577/0.807	0.322	0.371	0.210	0.828
PARTSYNTH	**0.661**	**0.431**	**0.379**	0.768	0.112	0.669/0.854	**0.291**	**0.277**	0.184	0.840

a symmetric nearest-part Chamfer distance that does not require known one-to-one correspondence: $\text{CD}_p = \frac{1}{2}\left(\frac{1}{m}\sum_{i=1}^{m}\min_j C_{ij} + \frac{1}{n}\sum_{j=1}^{n}\min_i C_{ij}\right)$, where m and n are the numbers of generated and ground-truth parts, respectively, and C_{ij} is the CD between generated part i and ground-truth part j. Here we report CD_p against the human-curated instance annotations in PartObjaverse-Tiny. However, because their granularity may differ from component-based decompositions, we additionally report $\text{CD}^\star_p$ against component-level references, as a complementary evaluation.

Whole-Shape PBR Consistency We render $\mathcal{C}$ and $\mathcal{G}$ with PBR shading from matched viewpoints under the CD-selected alignment and compare the resulting images using LPIPS and SSIM.

As shown in Table 1, PARTSYNTH matches or closely approaches the strongest monolithic baselines on whole-shape geometry, image alignment, and PBR quality, while outperforming all evaluated part-aware baselines across all applicable metrics. It achieves the lowest CD_p and $\text{CD}^\star_p$, indicating stronger part-surface fidelity under both human-curated instance and component-derived reference granularities. Moreover, $\text{CD}^\star_p < \text{CD}_p$ is consistent with greater fidelity to fine-grained component structure retained by our data preprocessing. In contrast, the other multi-part methods typically exhibit $\text{CD}^\star_p > \text{CD}_p$ because their training data merge component-level details into coarser parts.

Fig. 2 compares single-image 3D generation on diverse image inputs (the truck image is from PartObjaverse-Tiny; the others are AI-generated), including characters, machines, thin structures, and cluttered scenes. In these examples, PARTSYNTH shows finer part structure than the other part-aware methods, alongside detailed geometry and plausible PBR appearance. Fig. 3 shows additional results for AI-generated input images, with PBR renders of the complete assets and their separated parts. Further comparative examples appear in Appendix G.1.

4.3 PBR SYNTHESIS ON FIXED MULTI-PART GEOMETRY

To isolate material quality from geometry-generation errors, we evaluate image-conditioned PBR synthesis using fixed ground-truth multi-part geometry from PartObjaverse-Tiny. For PARTSYNTH, the supplied parts provide the set-indexed spatial support and part-geometry condition for material generation. We compare against the material-generation modules of TRELLIS.2, Hunyuan3D 2.1, Step1X-3D, OmniPart, and FullPart. TRELLIS.2 and Hunyuan3D 2.1 predict PBR materials for prescribed geometry, whereas the other three baselines provide only RGB appearance; unsupported material channels use the GLB exporter defaults. OmniPart and FullPart instead regenerate geometry from coarse part inputs, so we transfer their generated materials back to the original surfaces. Baseline adaptations are detailed in Appendix D.2. We use the following metrics to evaluate PBR synthesis quality (see details in Appendix F).

Whole-Shape and Part-Level PBR Consistency We compare PBR-shaded renders of $\mathcal{C}$ and $\mathcal{G}$ from matched viewpoints using CLIP, LPIPS, and SSIM. We report these metrics for both the assembled whole shape and individually rendered parts, using the subscript p for part-level scores, such as CLIP_p. Here, CLIP compares predicted and GT renders, different from the conditioning-image CLIP used in Section 4.2.

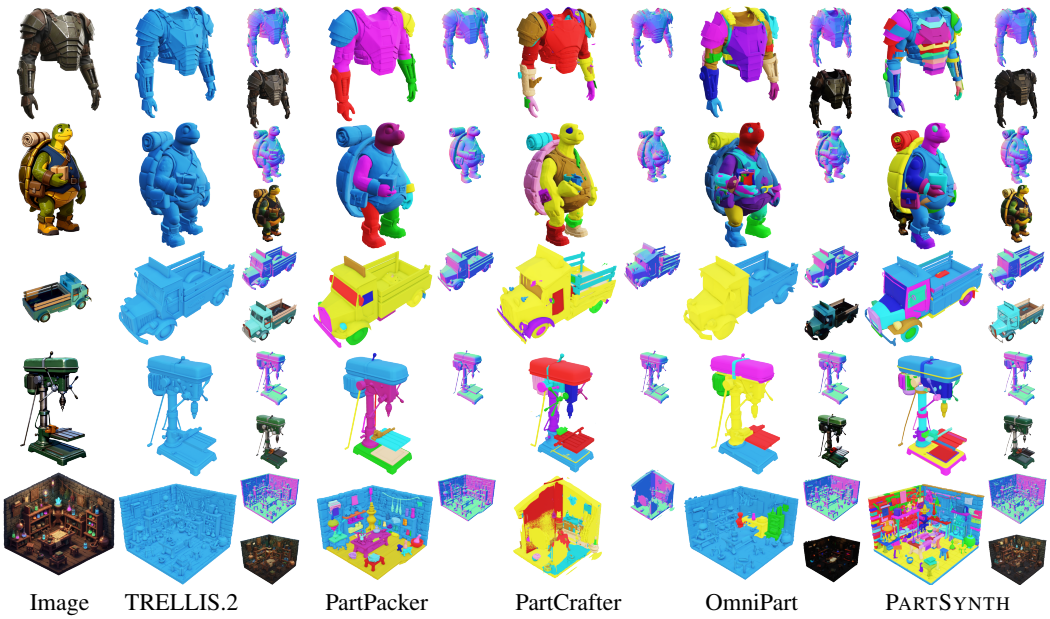

Figure 2: **Qualitative comparison of single-image 3D generation.** Each row shows a conditioning image and five generated results. For each method, the large geometry view is colored by part when available; the smaller views show a normal map (top) and, where available, a PBR render (bottom).



Figure 3: **Enlarged views of generated assets and their parts.** For each example, from left to right: the input image, a PBR render of the generated asset, and its separated, individually size-normalized parts rendered with their generated materials.

Part-Interface Material Error Material bleeding across part interfaces can be observed qualitatively without GT materials; however, quantitative comparison requires a common reference. We therefore assess interface bleeding using qualitative visualizations together with a GT-referenced

Table 2: **Conditional PBR synthesis on fixed ground-truth geometry.** Metrics without and with subscript p evaluate whole-shape and part PBR quality, respectively. PARTSYNTH-PerSet is a training-free ablation (see Section 4.5).

METHOD	Whole-Shape PBR			Part-Level PBR			
	CLIP ↑	LPIPS ↓	SSIM ↑	CLIP_p ↑	LPIPS_p ↓	SSIM_p ↑	PI-ME ↓
TRELLIS.2	**0.927**	0.046	0.945	0.886	0.082	0.933	0.071
Hunyuan3D 2.1	0.911	0.056	0.936	0.843	0.103	0.924	0.097
Step1X-3D	0.896	0.065	0.931	0.843	0.105	0.925	0.121
OmniPart	0.865	0.075	0.923	0.812	0.122	0.907	0.127
FullPart	0.887	0.069	0.929	0.823	0.121	0.914	0.117
PARTSYNTH	0.926	**0.045**	**0.948**	**0.902**	**0.064**	**0.945**	**0.061**
PARTSYNTH-PerSet	0.898	0.064	0.934	0.854	0.098	0.925	0.105



Figure 4: **PBR synthesis on fixed multi-part geometry.** For the character (top) and camera (bottom), the Input column shows the conditioning image above the supplied part-colored geometry. The Ref. column shows GT PBR renders of the whole asset and an isolated part; GT material attributes are not provided to any method. The remaining columns show corresponding whole-asset and isolated-part renders which help visualize material consistency across parts.

metric, *Part-Interface Material Error* (PI-ME), which measures material error near GT part interfaces. Given N surface samples from GT-defined interface regions, we compute $\text{PI-ME} = \frac{1}{6N}\sum_{i=1}^{N}\left\|\hat{\mathbf{m}}_{p_i}(\mathbf{x}_i) - \mathbf{m}^{\text{GT}}_{p_i}(\mathbf{x}_i)\right\|_2^2$,, where $\hat{\mathbf{m}}$ and $\mathbf{m}^{\text{GT}}$ are the predicted and ground-truth six-channel PBR material vectors. PI-ME may also capture other local material errors.

As shown in Table 2, PARTSYNTH performs best on all reported metrics except whole-shape CLIP, where it ranks second. It reduces PI-ME from 0.071 for TRELLIS.2 to 0.061, indicating lower material error near part interfaces. OmniPart and FullPart are additionally affected by geometry regeneration and material transfer.

Fig. 4 compares whole-asset and isolated-part PBR renders on two examples. Several baselines show apparent cross-part material bleeding issues, with hair color extending onto the character's head or lens material onto the camera body. PARTSYNTH shows cleaner part boundaries while retaining the assets' overall appearance.

4.4 CONVERSION TO COMPOSITIONAL ASSETS

Given a monolithic asset with geometry and PBR materials, we use its geometry as the sparse whole-shape scaffold and encode it to condition SPS and detailed geometry generation. We also encode its PBR materials to condition part-aware PBR synthesis. Scaffold sampling is skipped. Fig. 6 shows three conversions of assets generated by TRELLIS.2 from single images. The outputs expose separable parts while retaining the inputs' overall shape and appearance.

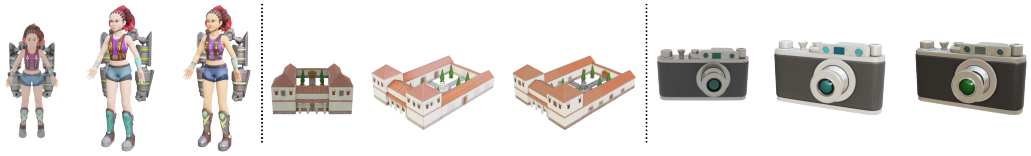

Figure 5: **Ablation study on monolithic PBR latent conditioning.**

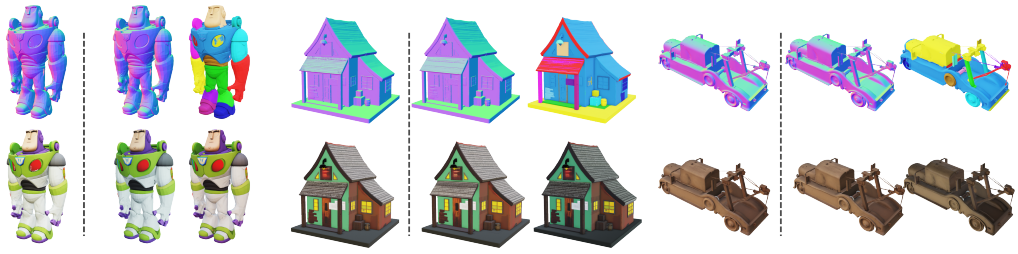

Figure 6: **Converting monolithic assets into compositional assets.** For each example, the left column shows normal and appearance renders of the input asset from the same viewpoint. The four panels to the right show the decomposed result via PARTSYNTH: a normal map render and part-colored geometry above two PBR renders under different lighting.

4.5 ABLATION STUDIES

Monolithic PBR Latent Conditioning Fig. 5 compares our PBR synthesis with and without the whole-object PBR latent. Each pair uses the same generated multi-part geometry, isolating the effect of material conditioning. In these examples, conditioning better preserves the input's appearance, including the character's boots, the camera lens, and details of the building.

Joint vs. Per-Set Generation We compare our jointly fine-tuned geometry and PBR models with PARTSYNTH-*PerSet*, a training-free baseline that applies the pretrained TRELLIS.2 models independently to each set. Geometry evaluation uses the same conditioning images and GT set-indexed volumes for both methods; PBR evaluation uses the same images and fixed GT multi-part geometry. Relative to PARTSYNTH-PerSet, PARTSYNTH reduces whole-shape CD from 0.041 to 0.038 and part-level CD_p from 0.034 to 0.028. On fixed geometry, it also improves all reported whole-shape and part-level PBR metrics (Table 2). PARTSYNTH-PerSet scores below monolithic TRELLIS.2 on these metrics, suggesting that independently applying the pretrained models to each set is not sufficient. Qualitative comparisons are provided in Appendix G.2.

5 CONCLUSION

We presented PARTSYNTH, a framework for generating multi-part 3D assets with part-aware PBR attributes from a single image. Its central result is that a bounded Sparse Part Structure with reusable set-indexed slots can support flexible part structures without fixing the output part count or generating parts sequentially. Across single-image 3D generation and PBR synthesis on fixed multi-part geometry, PARTSYNTH maintains competitive whole-shape quality while improving part-level geometry and material quality, including fidelity near part interfaces.

Limitations PARTSYNTH inherits the component-level granularity of its automatically processed training data. Very small components remain difficult to recover at the current 512^3 detailed-geometry resolution, and the generated parts are not necessarily semantic or functionally meaningful. The model also lacks explicit mechanisms for maintaining consistency among symmetric or repeated components, inferring articulation or assembly constraints, or controlling part count and granularity. Appendix G.1 also illustrates some of these limitations. We leave exploration of higher-resolution generation and text- or semantic-mask-based structural control to future work.

STATEMENTS

AI USE STATEMENT

In this work, we used generative AI tools (GPT) for language editing, including grammatical correction, sentence rephrasing, and improving the readability of the manuscript. We did not use generative AI tools to generate research ideas, formulate hypotheses, design methodologies or experiments, interpret results, conduct literature searches, or create scientific figures.

Generative AI tools were used to assist with creating utility scripts for automation and batch processing, and to provide suggestions during debugging. They were not used to develop or implement the core methods, algorithms, models, or experiments presented in this work.

All AI-assisted text and code were reviewed, revised, tested, and verified by the authors. The authors take full responsibility for all content of this work, including its scientific claims, methods, results, conclusions, code, and artifacts.

ETHICS STATEMENT

This work presents a method for generating multi-part 3D assets with PBR materials, with potential applications in computer graphics, design, virtual reality, and digital content creation. As with other generative models, the method could be misused to create misleading, inappropriate, or copyright-infringing content, and its outputs may reflect biases in the training data. We do not identify ethical risks specific to our method beyond these broader concerns. Any deployment should include appropriate safeguards and comply with applicable legal, ethical, and intellectual property requirements.

REPRODUCIBILITY STATEMENT

To support reproducibility, the paper and supplementary materials describe the problem formulation, optimization objectives, network architectures, training procedures, hyperparameters, datasets, and evaluation metrics. They also document the dataset construction and preprocessing procedures, data splits, loss functions, optimization settings, and training schedules. Our experiments use publicly available software frameworks and standard hardware platforms. Upon publication, we plan to release the source code, trained models, and processed datasets, subject to applicable licensing and organizational requirements.

REFERENCES

Maciej Bala, Yin Cui, Yifan Ding, Yunhao Ge, Zekun Hao, Jon Hasselgren, Jacob Huffman, Jingyi Jin, J.P. Lewis, Zhaoshuo Li, Chen-Hsuan Lin, Yen-Chen Lin, Tsung-Yi Lin, Ming-Yu Liu, Alice Luo, Qianli Ma, Jacob Munkberg, Stella Shi, Fangyin Wei, Donglai Xiang, Jiashu Xu, Xiaohui Zeng, and Qinsheng Zhang. Edify 3D: Scalable high-quality 3D asset generation. arXiv:2411.07135, 2024.

Daniel Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4): 251–256, 1979.

Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models. In *ICLR*, 2025a.

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. In *CVPR*, pp. 5881–5892, 2025b.

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. In *NeurIPS*, 2026.

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling geometry and appearance for high-quality text-to-3D content creation. In *ICCV*, 2023.

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, Liang Pan, Dahua Lin, and Ziwei Liu. 3DTopia-XL: Scaling high-quality 3D asset generation via primitive diffusion. In *CVPR*, pp. 26576–26586, 2025c.

Zilong Chen, Yikai Wang, Wenqiang Sun, Feng Wang, Yiwen Chen, and Huaping Liu. MeshGen: Generating PBR textured mesh with render-enhanced auto-encoder and generative data augmentation. In *CVPR*, pp. 5835–5848, 2025d.

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. In *ICLR*, 2026.

Shaocong Dong, Lihe Ding, Xiao Chen, Yaokun Li, Yuxin Wang, Yucheng Wang, Qi Wang, Jaehyeok Kim, Chenjian Gao, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. From one to more: Contextual part latents for 3D generation. In *ICCV*, pp. 8230–8240, 2025.

Andreas Engelhardt, Mark Boss, Vikram Voleti, Chun-Han Yao, Hendrik Lensch, and Varun Jampani. SViM3D: Stable video material diffusion for single image 3D generation. In *ICCV*, pp. 28428–28439, 2025.

Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. SparseFlex: High-resolution and arbitrary-topology 3D shape modeling. In *ICCV*, 2025.

Xufan He, Yushuang Wu, Xiaoyang Guo, Chongjie Ye, Jiaqing Zhou, Tianlei Hu, Xiaoguang Han, and Dong Du. UniPart: Part-level 3D generation with unified 3D geom-seg latents. In *CVPR*, 2026.

Xin Huang, Tengfei Wang, Ziwei Liu, and Qing Wang. Material anything: Generating materials for any 3D object via diffusion. In *CVPR*, pp. 26556–26565, 2025.

Tencent Hunyuan3D. Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready pbr material. arXiv:2506.15442, 2025.

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. In *ICCV*, pp. 4015–4026, 2023.

Zeqiang Lai, Yunfei Zhao, Zibo Zhao, Haolin Liu, Qingxiang Lin, Jingwei Huang, Chunchao Guo, and Xiangyu Yue. LATTICE: Democratize high-fidelity 3D generation at scale. In *CVPR*, pp. 19982–19992, 2026.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. GRASS: Generative recursive autoencoders for shape structures. *ACM Trans. Graph.*, 36(4):52:1–52:14, 2017.

Sikuang Li, Chen Yang, Jiemin Fang, Jiazhong Cen, Yuhe Wei, Jichen Pang, Wei Shen, and Qi Tian. SCULPT: Subtractive composition for 3D part generation. arXiv:2608.13541, 2026a.

Weiyu Li, Xuanyang Zhang, Zheng Sun, Di Qi, Hao Li, Wei Cheng, Weiwei Cai, Shihao Wu, Jiarui Liu, Zihao Wang, Xiao Chen, Feipeng Tian, Jianxiong Pan, Zeming Li, Gang Yu, Xiangyu Zhang, Daxin Jiang, and Ping Tan. Step1X-3D: Towards high-fidelity and controllable generation of textured 3D assets. arXiv:2505.07747, 2025a.

Zhihao Li, Yufei Wang, Heliang Zheng, Yihao Luo, and Bihan Wen. Sparc3D: Sparse representation and construction for high-resolution 3D shapes modeling. In *NeurIPS*, 2026b.

Zhiqi Li, Wenhuan Li, Tengfei Wang, Zhenwei Wang, Junta Wu, Haoyuan Wang, Yunhan Yang, Zehuan Huang, Yang Li, Peidong Liu, and Chunchao Guo. MoCA: Mixture-of-components attention for scalable compositional 3D generation. arXiv:2512.07628, 2025b.

Manwen Liao, Xinyu Lian, Jian Mao, Kaixu Chen, Li Luo, Jinghao Yan, Wanshui Gan, Qiao Yu, Weitian Zhang, Chunhua Shen, Guang Chen, Bo Dai, Xudong Xu, and Zhaoyang Lyu. MegaParts: Scaling part-aware 3D object generation to 300 parts via token-efficient autoregressive modeling. arXiv:2608.14783, 2026.

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. In *NeurIPS*, 2025.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *ICLR*, 2023.

Anran Liu, Cheng Lin, Yuan Liu, Xiaoxiao Long, Zhiyang Dou, Hao-Xiang Guo, Ping Luo, and Wenping Wang. Part123: Part-aware 3d reconstruction from a single-view image. In *SIGGRAPH Conference Papers*, pp. 24:1–24:12, 2024.

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 *ICCV*, pp. 9704–9715, 2025.

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:2509.06784, 2025.

Meshy. Meshy: AI 3D Model Generator. `https://www.meshy.ai/`, 2026. Accessed: 2026-09.

Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas J Guibas. StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Trans. Graph.*, 38(6): 242:1–242:19, 2019.

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. XCube: Large-scale 3D generative modeling using sparse voxel hierarchies. In *CVPR*, pp. 4209–4219, 2024.

Yawar Siddiqui, Tom Monnier, Filippos Kokkinos, Mahendra Kariya, Yanir Kleiman, Emilien Garreau, Oran Gafni, Natalia Neverova, Andrea Vedaldi, Roman Shapovalov, and David Novotny. Meta 3D AssetGen: Text-to-mesh generation with high-quality geometry, texture, and PBR materials. In *NeurIPS*, volume 37, pp. 9532–9564, 2024.

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. In *NeurIPS*, 2025.

Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. PQ-NET: A generative part Seq2Seq network for 3D shapes. In *CVPR*, pp. 829–838, 2020.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *CVPR*, pp. 21469–21480, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3D generation. In *CVPR*, 2026.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. UlIP-2: Towards scalable multimodal pre-training for 3D understanding. In *CVPR*, pp. 27091–27101, 2024.

Xinhao Yan, Jiachen Xu, Yang Li, Changfeng Ma, Yunhan Yang, Chunshi Wang, Zibo Zhao, Zeqiang Lai, Yunfei Zhao, Zhuo Chen, and Chunchao Guo. X-Part: high fidelity and structure coherent shape decomposition. In *CVPR*, 2026.

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:2411.07184, 2024.

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. In *ACM Trans. Graph.*, 2025.

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. In *ICLR*, 2026.

Jingrui Ye, Lingting Zhu, Runze Zhang, Zeyu Hu, Yingda Yin, Lanjiong Li, Lequan Yu, and Qingmin Liao. Large material Gaussian model for relightable 3D generation. arXiv:2509.22112, 2025.

Kyeongmin Yeo, Yunhong Min, Jaihoon Kim, and Minhyuk Sung. MatLat: Material latent space for PBR texture generation. In *CVPR*, 2026.

Ruihan Yu, Lian Fu, Muyao Niu, Zheng hui Huang, Yu-Ju Tsai, Sho Kuno, Fengbo Lan, Yonghao Yu, Erwin Wu, Ming-Hsuan Yang, Kaipeng Zhang, and Zhixiang Wang. Kaininja: Extending Native 3D Generators to the Part Level. arXiv:2609.15659, 2026.

Xianfang Zeng, Xin Chen, Zhongqi Qi, Wen Liu, Zibo Zhao, Zhibin Wang, Bin Fu, Yong Liu, and Gang Yu. Paint3D: Paint anything 3D with lighting-less texture diffusion models. In *CVPR*, pp. 4252–4262, 2024.

Biao Zhang and Peter Wonka. LaGeM: A large geometry model for 3D representation learning and diffusion. In *ICLR*, 2025.

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Trans. Graph.*, 42(4):92:1–92:16, 2023.

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 Trans. Graph.*, 43(4):120:1–120:20, 2024.

Yibo Zhang, Li Zhang, Rui Ma, and Nan Cao. TexVerse: A universe of 3D objects with high-resolution textures. arXiv:2508.10868, 2025.

Xinyang Zheng, Hao Pan, Pengshuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally Attentional SDF diffusion for controllable 3D shape generation. *ACM Trans. Graph.*, 42(4):91:1–91:13, 2023.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3D: Exploring unified 3D representation at scale. In *ICLR*, 2024.

A Implementation Details

A.1 SPS Network Architecture

Sparse VAE The sparse VAE encodes the sparse whole-shape scaffold U, a binary 64^3 grid, into a feature-bearing latent support Ω on a sparse 16^3 grid with an eight-dimensional feature vector at each active location. Its decoder reconstructs U. The VAE combines sparse 3D convolutions with DC-AE-style compression (Chen et al., 2025a) and contains 29.61M parameters. At sparse grid resolutions 64^3, 32^3, and 16^3, the encoder uses $(64, 128, 256)$ channels and $(1, 1, 3)$ residual blocks, respectively; only its final stage uses two-head self-attention. The decoder mirrors these stages, with $(256, 128, 64)$ channels and $(3, 1, 1)$ residual blocks at 16^3, 32^3, and 64^3. Each residual block contains a convolution and a feed-forward layer; these counts exclude the downsampling and upsampling modules. The number of active tokens at each resolution depends on the input.

The encoder parameterizes a diagonal Gaussian posterior whose mean is used for flow training and inference. Two spatial-to-channel downsampling modules and symmetric upsampling modules connect the three grid resolutions. At each upsampling level, an MLP predicts the binary occupancy

Table 3: **Current inference sampling parameters.** All stages use Euler sampling. w is the image CFG scale, ρ is the guidance-rescale coefficient, and r is the time-rescaling factor.

Stage	Steps	CFG w	Rescale ρ	CFG interval	Time scale r
Whole-shape scaffold	12	7.5	0.7	$[0.6, 1.0]$	5.0
Sparse Part Structure (SPS)	12	7.5	0.7	$[0.6, 1.0]$	5.0
Detailed geometry	12	7.5	0.5	$[0.6, 1.0]$	3.0
PBR materials	12	1.0	0.0	$[0.6, 0.9]$	3.0

of eight child voxels per parent. Training uses cross-entropy for these occupancy decisions and a KL-divergence regularizer with weight $\lambda = 0.001$. All sets share the decoder but are decoded independently.

Slot Features and Activity Let $S = |\Omega|$ be the number of active locations in the encoded scaffold support. We instantiate KS slots at coordinates (k, x, y, z), using $K = 8$. Each flow target contains eight normalized VAE features and an activity label repeated across eight channels, with $+1$ for active and -1 for inactive slots; inactive latent features are set to zero. The scaffold latent is replicated across sets and concatenated with the noisy flow state, producing 24 input channels before optional monolithic conditioning. The model predicts a 16-channel velocity using mean squared error. After sampling, we average the eight activity channels, retain slots with positive mean activity, undo latent normalization, and decode the retained features.

Transformer and Image Conditioning The SPS flow model contains 30 transformer blocks with hidden width 1536, and has 1.295B parameters excluding the sparse VAE and image encoder. Each block uses 12 attention heads of dimension 128 and a $1536 \to 8192 \to 1536$ GELU feed-forward network. We use pre-layer normalization and residual connections, with RMS normalization on attention queries and keys. Sinusoidal timestep embeddings modulate the self-attention and feed-forward branches. Cross-attention uses DINOv3 ViT-L/16 features extracted from the input image at 512×512 resolution.

A.2 Monolithic Conditioning Details

For SPS, the monolithic geometry latent has 32 channels on a 64^3 sparse grid. Two spatial-to-channel rearrangements reduce it to the 16^3 scaffold latent grid with 2048 channels. We align these features with the sparse support, fill absent locations with zeros, broadcast them across the K sets, and concatenate them before the input projection.

For detailed geometry and PBR generation, the corresponding monolithic geometry and material latents each have 32 channels on a 32^3 grid. We align each latent with the set-indexed tokens and inject it through a learned $32 \to 1536$ projection with a zero-initialized scalar gate. Geometry and PBR use separate projections and gates. These optional monolithic pathways are distinct from the scaffold condition used by SPS and the same-set geometry condition used by PBR generation.

A.3 Training and Inference Setup

All three flow models are trained with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.01, and $\epsilon = 10^{-8}$), a constant learning rate of 10^{-4} without warm-up, and a global batch size of 256. SPS and detailed geometry are trained on 64 Nvidia A100 GPUs (80 GB) with four samples per GPU, while PBR generation uses 32 GPUs with eight samples per GPU, processed in two microbatch splits. We use BF16 mixed-precision training, an exponential moving average with decay 0.9999, and adaptive gradient clipping with maximum norm 1.0 at the 95th percentile. Flow timesteps are sampled uniformly with $\sigma_{\min} = 10^{-5}$. During SPS training, we randomly permute the set indices to prevent index-specific semantics. During training, we drop the monolithic latent condition with probabilities 0.15, 0.4, and 0.4 for SPS, detailed geometry, and PBR generation, respectively.

At inference, all stages use Euler flow sampling and non-EMA weights. Detailed geometry and PBR generation use output resolution 512 and eight-color positional encoding. Stage-specific guidance and rescaling parameters appear in Table 3.

Table 4: Data statistics of multi-part 3D asset datasets. #Assets denotes the number of assets in each dataset. #Parts per Asset shows statistics of part counts per asset including mean, standard deviation, 25th percentile, median, 75th percentile, and maximum values.

Dataset	#Assets	#Parts per Asset					
		Mean	STD	25%	50%	75%	Max
PARTVERSE	11797	7.36	6.43	4	6	9	139
PARTVERSE-XL	37449	8.83	7.53	4	7	11	198
MEGAPART3D	328982	39.5	107.3	4	11	34	8412

B DATASET CONSTRUCTION

We construct MEGAPART3D from TRELLIS-500K (Xiang et al., 2025) and TexVerse (Zhang et al., 2025). Following the TRELLIS.2 filtering protocol, we remove assets without PBR materials or with low aesthetic scores. For each retained GLB, we extract submeshes from its node hierarchy, weld duplicate boundary vertices, separate connected components, and assign each component a unique source part identifier.

Table 4 compares MEGAPART3D with PartVerse (Dong et al., 2025) and PartVerse-XL (Ding et al., 2026), which obtain semantically meaningful parts through mesh pre-segmentation followed by human refinement. In contrast, our automatic pipeline preserves source component instances without manual part annotation, enabling construction at a larger scale.

C COMPLEXITY OF VOLUME-PACKING-BASED APPROACHES

We compare how the number of set volumes K affects token counts, attention costs, and model size in PartPacker, KaiNinja, and PARTSYNTH.

PartPacker PartPacker concatenates two latent sequences of $N = 4096$ tokens before a shared transformer, giving an 8192-token sequence (Tang et al., 2025). A direct K-volume extension of this design would contain

$$L_{\mathrm{PP}} = KN = 4096K \tag{2}$$

tokens. Full self-attention would require $O(K^2N^2)$ attention scores, while token storage grows as $O(KN)$. The shared transformer backbone need not grow with K, although a generalized implementation would require additional stream identifiers.

KaiNinja KaiNinja's layout stage uses a dense 16^3 latent sequence and a separate transformer copy for each volume (Yu et al., 2026). Extending this layout architecture to K volumes would process

$$L_{\mathrm{KN}} = KN, \qquad N = 16^3 = 4096 \tag{3}$$

tokens. Across the stream-specific blocks, self-attention requires $O(KN^2)$ scores and the layout-transformer parameters grow linearly with K. Each cross-volume block attends over all KN tokens and therefore uses $O(K^2N^2)$ scores. KaiNinja's three-volume ablation reports $1.5\times$ the per-step cost of its two-volume model, consistent with the added stream. Its refinement stage shares transformer weights across streams, so this linear parameter growth applies to the layout stage rather than the entire pipeline.

PARTSYNTH PARTSYNTH first generates a single scaffold latent independently of K. Let $S = |\Omega| = \mathcal{O}(16^2)$ be the number of active locations in the encoded scaffold support. The SPS stage instantiates K set-indexed slots at each active location, giving

$$L_{\mathrm{SPS}} = KS. \tag{4}$$

Because one transformer is shared across all slots, its parameter count is independent of K. Token storage is $O(KS)$, while full self-attention uses $O(K^2S^2)$ scores. Unlike methods that allocate a dense $N = 4096$ token sequence to every volume, PARTSYNTH instantiates slots only on the input-dependent sparse support. When $S < N$, its attention-score size is smaller by a factor of

$(S/N)^2$ at the same K. The dependence on K remains quadratic; the practical advantage comes from the smaller support and the absence of per-stream transformer replication, which make the chosen budget $K = 8$ tractable.

D Baseline Implementation Details

D.1 Single-Image 3D Generation

Monolithic Generators For TRELLIS.2, Hunyuan3D 2.1, and Step1X-3D, we use the released inference implementations and pretrained models, with the same conditioning image used for the multi-part generators.

Multi-Part Generators For PartPacker, we use the official implementation and recommended settings. For PartCrafter, we set the requested number of parts to the GT count, thereby providing privileged information about the target decomposition. For OmniPart (Yang et al., 2025), we generate its segmentation input automatically with the Segment Anything Model (SAM) (Kirillov et al., 2023), without using GT masks.

D.2 PBR Synthesis on Fixed Multi-Part Geometry

Monolithic Material Models For TRELLIS.2, Hunyuan3D 2.1, and Step1X-3D, we merge the supplied parts into a complete mesh before appearance generation. For TRELLIS.2, we sample the decoded sparse material field at the original surface positions. For Hunyuan3D 2.1 and Step1X-3D, we retain source-triangle correspondence and rebake the generated textures onto the original primitives, with Hunyuan remeshing disabled. These procedures preserve the supplied geometry and decomposition.

OmniPart Adaptation We bypass OmniPart's layout-prediction stage and run only its second-stage part-synthesis model. We normalize the complete input object in the model's coordinate frame and surface-voxelize it on a 64^3 grid. The conditioning contains the whole-object occupancy and, for each prescribed part, the occupied cells within its bounding box. We decode the generated part outputs and transfer their colors to the corresponding original surfaces using the part-restricted projection described below.

FullPart Adaptation We use the supplied GT part bounding boxes as inputs to FullPart's released final structured-latent generation stage. Each part is centered and scaled in its own canonical frame and surface-voxelized on a 64^3 grid. We provide the resulting voxel coordinates, the bounding boxes in the shared object frame, and the conditioning image. We decode one geometry and appearance output per input part and transfer its colors to the corresponding original surface.

Part-Restricted Surface Projection For OmniPart and FullPart, we transfer generated colors to the supplied geometry independently for each matched part. We rasterize the original part's UV atlas to recover a 3D position for each texel, project that position onto the closest triangle of the corresponding generated part, interpolate its generated UV coordinates, and sample the color texture. Restricting the search to the matched part prevents cross-part transfer, although geometric discrepancies can still affect the projected material.

Available Material Channels For the RGB-only outputs from Step1X-3D, OmniPart, and FullPart, we set the unsupported channels to metallic 0, roughness 1, and opacity 1, following their exporters. PI-ME uses these fixed values without fitting them to the GT materials.

E IMAGE-TO-3D EVALUATION DETAILS

E.1 GEOMETRY METRICS

Following TRELLIS.2, we apply scene-node transforms, independently center and scale each predicted or GT object to a longest bounding-box edge of 2, and sample 10,000 area-weighted points from its merged surface. All parts of an object share the same normalization.

For each input, we evaluate four generated candidates under yaw rotations $\{0°, 90°, 180°, 270°\}$ and select the generation–rotation pair with minimum whole-object CD. All geometry metrics use this selection; image-alignment metrics use the selected generation in its original orientation.

E.2 IMAGE-ALIGNMENT METRICS

All image-alignment scores use normalized embeddings.

CLIP Rendering Following TRELLIS.2, we render four 512^2 views at yaw angles $0°, 90°, 180°, 270°$ with fixed camera elevation, normalization, and white background. We average the conditioning-image similarity across views using OpenCLIP ViT-B/32 with `laion2b_s34b_b79k` weights. Normal-map CLIP uses the same cameras without lighting.

F PBR EVALUATION DETAILS

F.1 MATCHED WHOLE-SHAPE AND PART RENDERS

We render the outputs of all methods on the same GT surfaces, with matched cameras, lighting, tone mapping, and background. Whole-shape metrics render the assembled asset; part-level metrics render each GT part independently using the same four-view setup as Appendix E.2.

F.2 PART-INTERFACE MATERIAL ERROR

Selecting Part-Interface Regions We normalize each GT object to a longest bounding-box edge of 1 and surface-voxelize every original part on a shared 512^3 grid. Cells intersecting at least two parts define the interface region. We retain a separate entry for each part occupying such a cell, preserving the original part identity rather than its bounded set index.

Surface Samples And Material Queries We sample $N = 100{,}000$ part–cell entries and project each cell center onto the nearest triangle of its own part. The same GT-defined samples are shared by all methods. GT and predicted six-channel PBR material vectors are queried at those surface points, with sRGB base color linearized before comparison. Shapes without interface entries receive zero; missing predictions are excluded.

G ADDITIONAL QUALITATIVE RESULTS AND APPLICATIONS

G.1 ADDITIONAL QUALITATIVE COMPARISONS AND FAILURE CASES

Fig. 7 extends the single-image 3D comparison with additional inputs.

Fig. 8 compares PARTSYNTH with the commercial system Meshy (Meshy, 2026) on three inputs. In these examples, PARTSYNTH generates finer part structures, while Meshy produces better image-aligned geometry.

Fig. 9 shows two failure cases with AI-generated input images: a mechanism with small and complicated components and a crowded architectural scene. In these examples, fine structures are not fully recovered, and some generated parts appear fragmented.



Figure 7: **Additional single-image 3D generation comparisons.** The first and fourth input images are renders of PartObjaverse-Tiny assets; the others are AI-generated. The output views follow the layout in Fig. 2.

G.2 A COMPARATIVE STUDY ON PARTSYNTH VERSUS PARTSYNTH-PERSET

Fig. 10 compares PARTSYNTH and PARTSYNTH-PerSet under matched conditions. For geometry generation, PARTSYNTH-PerSet produces noisier surfaces and an incomplete leg. In the separate PBR comparison on the same fixed GT multi-part geometry, its output places facial material on the body and leaves one leg black. These examples illustrate the cross-part inconsistencies that can arise when each set is processed independently.

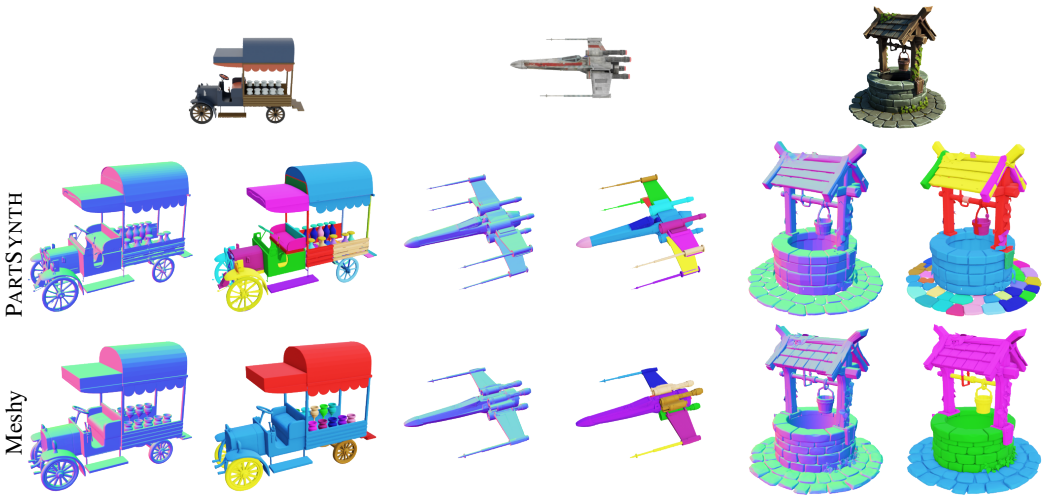

Figure 8: **Single-image 3D generation compared with Meshy.** Each input image (top) is followed by the normal map and part-colored geometry from PartSynth (middle) and Meshy (bottom).



Figure 9: **Failure cases on AI-generated images.** Each triplet shows the input image (left), part-colored generated geometry (middle), and a normal-map render (right).

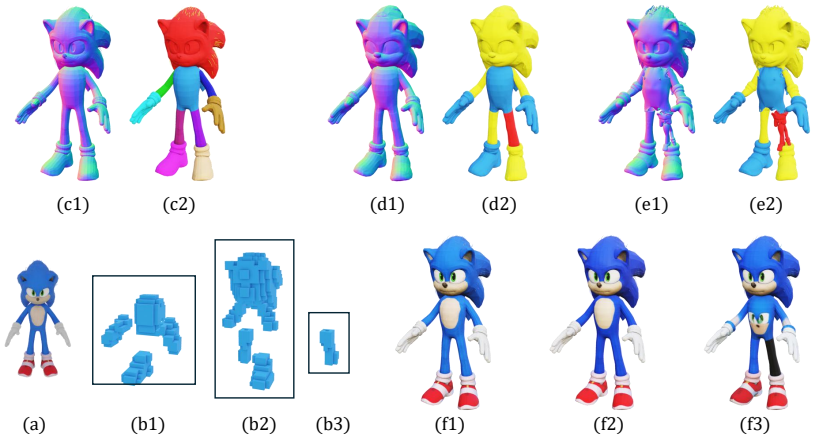

Figure 10: **PartSynth versus PartSynth-PerSet.** (a) Conditioning image. (b1–b3) Three set-indexed volumes derived from the GT part structure and shared by both variants. (c1,c2) GT normal and part-colored renders; (f1) GT PBR render. (d1,d2) Normal and set-colored renders from PartSynth; (e1,e2) the corresponding renders from PartSynth-PerSet. Colors in (d2,e2) identify sets, not individual parts. (f2,f3) PBR renders from PartSynth and PartSynth-PerSet on the same fixed GT multi-part geometry, independently of the generated geometry in (d,e).