COMPOSE: COMPOSITIONAL SYNTHESIS AND EDITING OF 3D SHAPES VIA PART-AWARE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Creating and editing high-quality 3D content remains a central challenge in computer graphics. We address this challenge by introducing **CompoSE**, a novel method for **Compo**sitional **S**ynthesis and **E**diting of 3D shapes via part-aware control. Our method takes as input a set of coarse geometric primitives (e.g., bounding boxes) that represent distinct object parts arranged in a particular spatial configuration, and synthesizes as output part-separated 3D objects that support localized granular (i.e., compositional) editing of individual parts. The key insight that enables our method is our use of a diffusion transformer architecture that alternates between processing each part locally and aggregating contextual information across parts globally, and features a novel conditioning technique that ensures strong adherence to the user's input. Importantly, our method learns to infer part semantics and symmetries directly from the user's coarse layout guidance, and does not require part-level text prompts. We demonstrate that our method enables powerful part-level editing capabilities, including context-aware substitution, addition, and style-preserving resizing operations. We show through extensive experiments that our method significantly outperforms existing approaches on guided synthesis, as measured by objective metrics and LLM-based evaluations.

1 INTRODUCTION

Creating and editing high-quality 3D content remains a central challenge in many creative industries, including film, design, simulation, and manufacturing. In response, the computer graphics community has developed a variety of generative AI methods for synthesizing 3D shapes from high-level user input [113, 112, 67, 14, 82, 73, 68, 75, 102, 32, 25, 71, 50, 111, 69, 108, 41, 96, 70, 87, 86, 34, 15, 91, 92, 93]. However, many of these methods rely on language descriptions to guide their output, and the resulting user interfaces can be notoriously unpredictable [1], especially when a user is attempting to specify the precise spatial configuration of individual object parts using text prompts.

In an effort to provide more precise user controls, a variety of methods have been proposed that accept richer forms of guidance, such as 2D images [56, 57, 80, 105, 115, 92, 93, 91, 66, 45, 38], a global bounding box [115, 81], or proxy shapes [12, 84, 115, 10, 11, 23]. Additionally, methods have been proposed to generate part-separated outputs [110, 90, 59, 22, 109, 116] to facilitate downstream applications. However, although some of these methods generate part-separated *outputs*, none of them accepts part-separated *input layouts*. The choice made by existing methods to model the user's intent through a single monolithic object (i.e., without individual parts) also limits predictability, because small changes in the user's input can lead to significant changes in the synthesized output, potentially altering the entire generated 3D shape. More generally, no existing method supports *compositional synthesis and editing* (i.e., synthesizing part-separated objects in a way that allows for localized granular editing of individual parts), which is essential in professional content creation scenarios.

In this work, we introduce **CompoSE**, a novel method for **Compo**sitional **S**ynthesis and **E**diting of 3D shapes via part-aware control. Our method synthesizes part-separated 3D shapes from a set of coarse geometric primitives (e.g., bounding boxes) in a particular spatial configuration, and a global text prompt. In our method, each geometric primitive represents a distinct object part that the user

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.

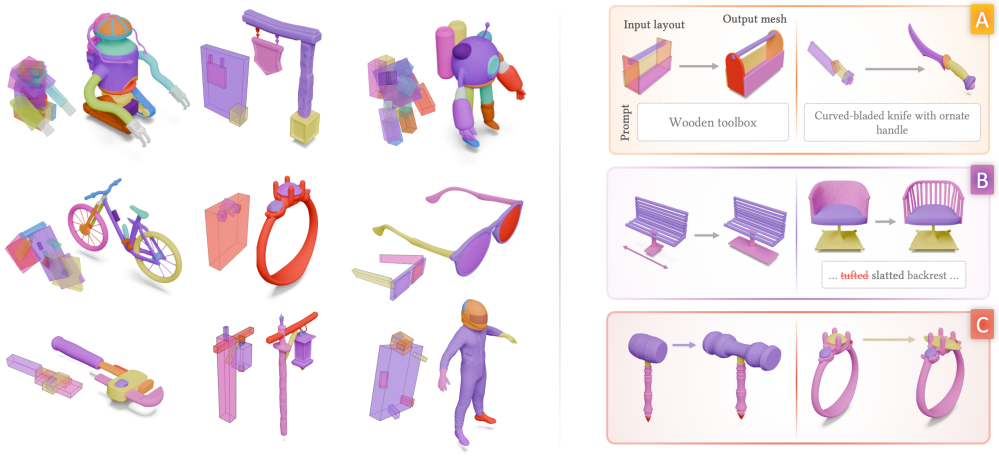


Figure 1: Our method for synthesizing 3D shapes takes as input a text prompt and a set of coarse geometric primitives (e.g., bounding boxes) that represent distinct object parts that a user would like to manipulate individually. From this input, our method synthesizes part-separated 3D objects that align with the user's geometric primitives and text prompt (A and additional results on the left). Our method enables *compositional synthesis and editing*, i.e., we synthesize 3D shapes in a way that allows for localized granular editing of individual parts. For example, our method enables style-preserving resizing (B , left) and text-based editing of individual parts (B , right), as well as context-aware part substitution (C).

can manipulate individually. The key insight that enables compositional synthesis and editing in our method is our use of a diffusion transformer (DiT) architecture [95, 40, 74] that processes each input geometric primitive in parallel, and strategically alternates a single DiT block between two modes. In *local stages*, the block processes spatial layout information for individual primitives, and provides high-fidelity control over the geometry of each synthesized object part; while in *global stages*, it aggregates context cues and text information across object parts to ensure stylistic coherence and structural validity.

Our architecture design enables compositional editing by allowing the user to spatially manipulate a specific geometric primitive that is given as input to a local stage, while optionally freezing the latent variables associated with the other parts at inference time. The 3D shapes resulting from this inference procedure adhere to user edits, while also preserving the identity of existing object parts exactly. To summarize, our main contributions are as follows:

1. We present the first method to synthesize part-separated 3D shapes directly from coarse part layouts, featuring a novel conditioning technique that ensures strong adherence to the user's input layout, and a high degree of control over individual object parts.
2. We enable a versatile set of compositional editing capabilities, including substitution, addition, and style-preserving resizing of individual object parts.
3. We provide a fully automated data processing pipeline that enables training on large-scale 3D shape datasets, and does not require manual part-level segmentations or text annotations.

We demonstrate through extensive experiments that our method significantly outperforms existing approaches on guided synthesis, as measured by objective metrics and LLM-based evaluations.

2 Related Work

Controllable Shape Synthesis. The advent of diffusion models [40, 74] has significantly advanced the field of 3D shape synthesis. Due to the expansive literature, we refer readers to a recent survey by Wang et al. [98]. We review the most representative methods in section 1, where we categorize them according to each method's choice of user guidance. A common thread among these methods is that they lack the ability to accept part-separated user guidance, which is a gap we address here.

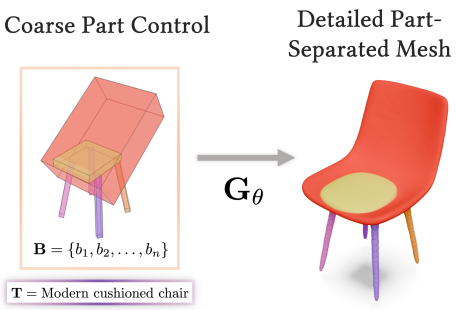


Figure 2: **Our problem setting** for shape synthesis via part-aware control. We propose to synthesize a 3D shape from a set of coarse geometric primitives (e.g., oriented bounding boxes) arranged in a particular spatial configuration, and a global text prompt. In our problem setting, we do not rely on image inputs or part-level text prompts.

Part-Aware Shape Synthesis. Early efforts in part-aware synthesis introduced part-disentangled representations for language- or sketch-guided generation [35, 53, 104]. However, these pioneering methods were often limited by their reliance on category-specific models and typically produced low-quality geometry. Recent approaches leverage DiT-based diffusion models [74] to generate part-separated output shapes, starting either from an input 3D mesh [110] or a 2D image [90, 59, 22, 109, 116].

While these image-conditioned methods have shown impressive results, their design imposes fundamental limitations on editability and controllability. First, conditioning on 2D image features — reinjected at every stage of the pipeline — globally couples all parts together, so resizing, substituting, or iterating on individual parts requires reconditioning on a new image and regenerating the entire shape. Second, the conditioning image is itself typically produced by a 2D diffusion model, which offers no part-level control and provides no information about occluded or internal geometry, restricting the user to parts visible from a single camera viewpoint. Although some of these methods (e.g., OmniPart [109]) include an intermediate bounding-box stage, the boxes are axis-aligned, not user-controllable, and cannot be used for part-level editing, since image features are still reinjected downstream. In contrast, our method accepts user-controlled oriented bounding boxes paired with a global text prompt as its primary guidance, enabling specification of parts invisible from any camera angle and supporting compositional editing — adding, resizing, and replacing parts — while keeping all other parts frozen.

Distinct from methods that synthesize geometry directly, methods based on *3D shape programs* [46, 42, 29] generate executable code capable of enforcing structural constraints like symmetry, but are often limited to specific categories and produce lower-fidelity geometry than diffusion-based methods. Our work combines the fine-grained control associated with 3D shape programs with the high geometric fidelity and cross-category generality of diffusion-based methods.

3D Shape Editing. A common approach for editing 3D shapes involves leveraging 2D diffusion models, either through per-scene optimization via Score Distillation Sampling [9, 72, 114, 120, 49, 18, 21, 20, 60, 121, 97, 106, 83], or by using multi-view diffusion models to generate edits that are then lifted to 3D [4, 55, 17, 6, 3, 26, 78, 101, 13, 100, 33, 94, 8]. Another common approach is to directly leverage a 3D diffusion prior and use an existing proxy shape for guidance [115, 11, 23, 28, 85] while synthesizing geometric details across the entire shape. More recent work has achieved localized editing by leveraging grid-based latent representations [54, 105, 45, 65, 76]. However, these methods either achieve editing through text or image instructions, or operate on monolithic shapes whose parts cannot be edited independently. In contrast, our work provides an intuitive interface where users can add, resize, and replace individual object parts by manipulating bounding boxes.

3 Part-Aware Control, Synthesis, and Editing

Our goal in this section is to design a latent diffusion transformer model that enables compositional synthesis and editing of 3D shapes, where the user's intent is communicated through a global text prompt describing an object, and a spatial layout of coarse geometric primitives (e.g., oriented bounding boxes) describing the arrangement of the object's individual parts.

Technical Requirements. Enabling compositionality in our model imposes three requirements on its design. First, the model must accept coarse geometric primitives (e.g., oriented bounding boxes) that specify the intended structure and part layout. Second, the model must fuse part-level geometric conditions with a global text prompt so that each synthesized part respects its local geometric constraints while remaining semantically coherent with the overall description. Third,

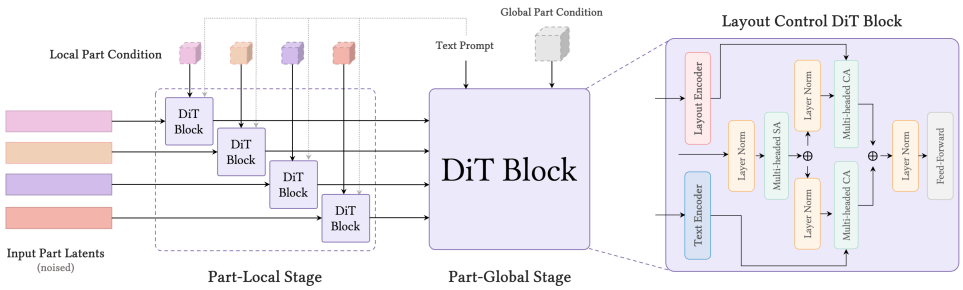


Figure 3: **Our diffusion transformer (DiT) architecture** for compositional synthesis and editing with part-aware control. Our architecture alternates between local and global stages. Local stages spatially ground each individual part to its location, and global stages aggregate context cues and text information across primitives.

the model must support localized editing of individual parts through manipulation of individual geometric primitives at inference time. We discuss our editing procedure in appendix D.

Problem Setting. Motivated by the technical requirements in the previous paragraph, we consider the following formal problem setting, which we illustrate in fig. 2. Given a conditioning set $\mathbf{B} = \{b_1, b_2, \ldots, b_n\}$ of coarse oriented boxes and a shape-level text prompt T, we want to synthesize *separate parts* p_i for each box in $\mathbf{B}$, such that the synthesized parts $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$ together form a coherent 3D shape aligning with T. Formally, we want:

$$\mathbf{P} = \mathbf{G}_\theta\left(\mathcal{E}_\mathbf{B}(\mathbf{B}), \mathcal{E}_\mathbf{T}(T)\right) \tag{1}$$

where $\mathbf{G}_\theta$ is a generative model parameterized by θ, $\mathcal{E}_\mathbf{B}$ is a learned box encoder that maps each box b_i to an appropriate latent representation, and $\mathcal{E}_\mathbf{T}$ is a pre-trained text encoder [79].

3.1 Layout-Controlled Diffusion Transformer

Shared Latent Space. We encode each part p_i with a pre-trained shape VAE into a latent $\mathbf{z}_i \in \mathbb{R}^{D\times d}$ of D tokens of width d, which the decoder turns back into an SDF. For the box encoder $\mathcal{E}_\mathbf{B}$, we use that *same* VAE encoder, so that a box is encoded exactly as a part would be:

$$\mathbf{c}_i = \mathcal{E}_\mathbf{B}(b_i) = \textsc{Vae}_{\textsc{enc}}(\textsc{Sample}(b_i)) \in \mathbb{R}^{D\times d}. \tag{2}$$

The conditions and the denoising targets then share one space, so the model does not have to learn how they relate from scratch. We encode the minimal oriented box $b_\star$ enclosing all primitives in the same way, as $\mathbf{c}_\star = \mathcal{E}_\mathbf{B}(b_\star)$.

Layout Control DiT Block. The user gives a single global prompt, but the model must fill several primitives, each with a different spatial role. We therefore inject layout and text conditioning through two *parallel* cross-attention branches (see fig. 3). For a block input $\mathbf{u}$, layout condition $\mathbf{c}$, and text tokens $\mathbf{y} = \mathcal{E}_\mathbf{T}(T)$, the block computes

$$\begin{aligned} \mathbf{h} &= \mathbf{u} + \mathrm{SA}(\mathrm{LN}(\mathbf{u}),\, \mathbf{M})\,, \\ \mathbf{h}' &= \mathbf{h} + \alpha_c\, \mathrm{CA}(\mathrm{LN}(\mathbf{h}),\, \mathbf{c}) + \mathrm{CA}(\mathrm{LN}(\mathbf{h}),\, \mathbf{y})\,, \end{aligned} \tag{3}$$

where LN is layer normalization and $\mathbf{M}$ a self-attention mask that depends on the stage (described next). The scalar α_c sets the layout control strength in each layer: we keep $\alpha_c = 1$ during training, and anneal it for the negative samples at inference time (appendix H.2 of the supplementary).

Part-Local and Part-Global Stages. The same block runs in two modes, differing only in the attention mask and in the layout condition we inject (fig. 3). Local stages process each part on its own, conditioned on its primitive $\mathbf{c}_i$, which grounds the part spatially: its geometry is synthesized at the position, scale, and orientation of its target box. Global stages process all parts jointly over $\mathbf{Z} = [\mathbf{z}_1; \ldots; \mathbf{z}_n]$, conditioned on the enclosing primitive $\mathbf{c}_\star$, so that the parts fit together into

one coherent object. These stages also carry the semantics of the injected text prompt $\mathbf{y}$ across parts, and let the model pick up relations between them, such as symmetries, without any explicit supervision. We interleave the two: local stages keep control fine-grained, while global stages keep the parts mutually consistent.

Training. Our generative model $\mathbf{G}_\theta$ is a conditional flow model trained end-to-end. Let $\mathbf{C} = \{\mathbf{c}_1, \ldots, \mathbf{c}_n, \mathbf{c}_\star\}$ denote the layout conditions read by the layout branches, $\mathbf{y}$ the text prompt, and $\mathbf{Z}_t = (1-t)\mathbf{Z}_0 + t\boldsymbol{\epsilon}$ the part latents interpolated toward noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$. Following rectified flow [61], we train the velocity network $\mathbf{v}_\theta$ with

$$\mathcal{L}(\theta) = \mathbb{E}_{(\mathbf{Z}_0,\, \mathbf{C},\, \mathbf{y}),\, \boldsymbol{\epsilon},\, t} \left[\left\| \mathbf{v}_\theta(\mathbf{Z}_t,\, \mathbf{C},\, \mathbf{y}\,;\, t) - (\boldsymbol{\epsilon} - \mathbf{Z}_0) \right\|_2^2 \right], \tag{4}$$

so that each denoising step is conditioned on both the part layout (via $\mathbf{C}$) and the prompt (via $\mathbf{y}$). Full training details are in appendix H.4.

Part-Aware Editing. Our model supports part-level edits at inference time by freezing the latents of untouched parts, which controls *where* an edit applies, and reinjecting cached attention keys and values, which controls *how much* of the edited part's original identity is preserved. We provide full details in appendix D.

4 Experiments

We evaluate our proposed method on 1,000 part layout and prompt pairs unseen during training, extracted from the Objaverse [19] dataset, following the preprocessing pipeline described in appendix I.1.

4.1 Compositional Synthesis and Editing

Synthesis. In fig. 4, we show diverse shape generations for given target layouts and text prompts, where we demonstrate strong alignment between our generated shapes and the provided layout constraints and prompts. As shown in the top rows of fig. 4, our method successfully generates varied objects, such that synthesized parts clearly correspond to the input bounding boxes and the overall structure matches the user's intent.

Editing. In fig. 5, we demonstrate our method's versatility through diverse editing operations. In the first two rows, we showcase edit sequences starting from a base shape: we iteratively add parts by creating boxes or substitute them by rescaling without style preservation. In the third row, we compare with Prox-E [85] on the same edit chain. Since it cannot keep unedited parts fixed, it regenerates the whole object at each step, so the parts are no longer separated and the shape drifts from the original. In rows four to six, we show examples of *identity-preserving edits*, where we rescale parts while preserving their geometric identity, as described in appendix D. We compare our method with anisotropic scaling, and provide a quantitative evaluation against anisotropic scaling and part substitution in table 4 of the supplementary. Finally, the last row shows *text-based part editing*: changing the prompt under a fixed layout yields localized edits that preserve the shape's structure and style, unlike the global edit in the middle of the row.

4.2 Comparisons

Metrics. We evaluate generation quality and controllability using several metrics. To measure visual quality, we compute the Fréchet Inception Distance (FID) [37, 89] on rendered normal maps at both the whole-object level (**Object-FID**) and for individual, normalized parts (**Part-FID**). Spatial alignment is quantified by Intersection-over-Union (IoU) [48]. **Object-IoU** measures the overlap between the generated object's bounding box and the union of all control boxes. We also report **Voxel-IoU** on a 64^3 grid, which is useful for evaluating baselines that do not produce part segmentations, and **Part-IoU**, which averages the bounding-box overlap at the individual part level. Finally, to measure text-shape alignment, we compute the CLIP-Score [79, 36] between the input prompt and rendered views.

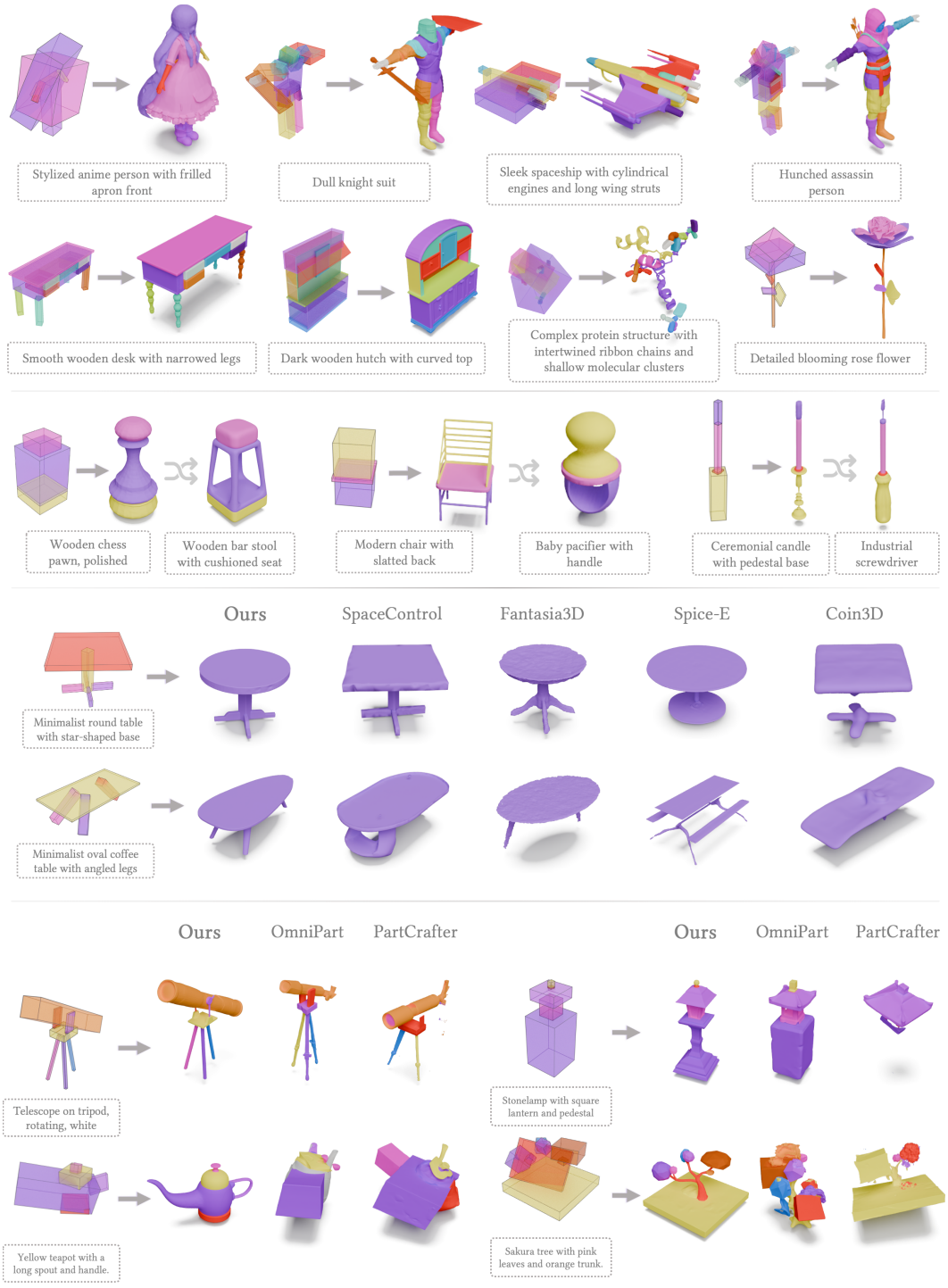


Figure 4: **Part-Controlled Synthesis.** We demonstrate our method's ability to generate diverse shapes that satisfy part layout constraints while matching input text prompts. We showcase multiple generations for various prompt-layout pairs (first two rows) using our highest-resolution model, followed by variations in text prompts for a fixed layout (third row). We then compare against proxy-guided methods (rows four and five), which receive the same coarse guidance shape and text prompt as our method, and against image-conditioned part synthesis methods (rows six and seven), which we condition on an image of the object generated by Nano Banana Pro [30] from a rendering of the target layout.

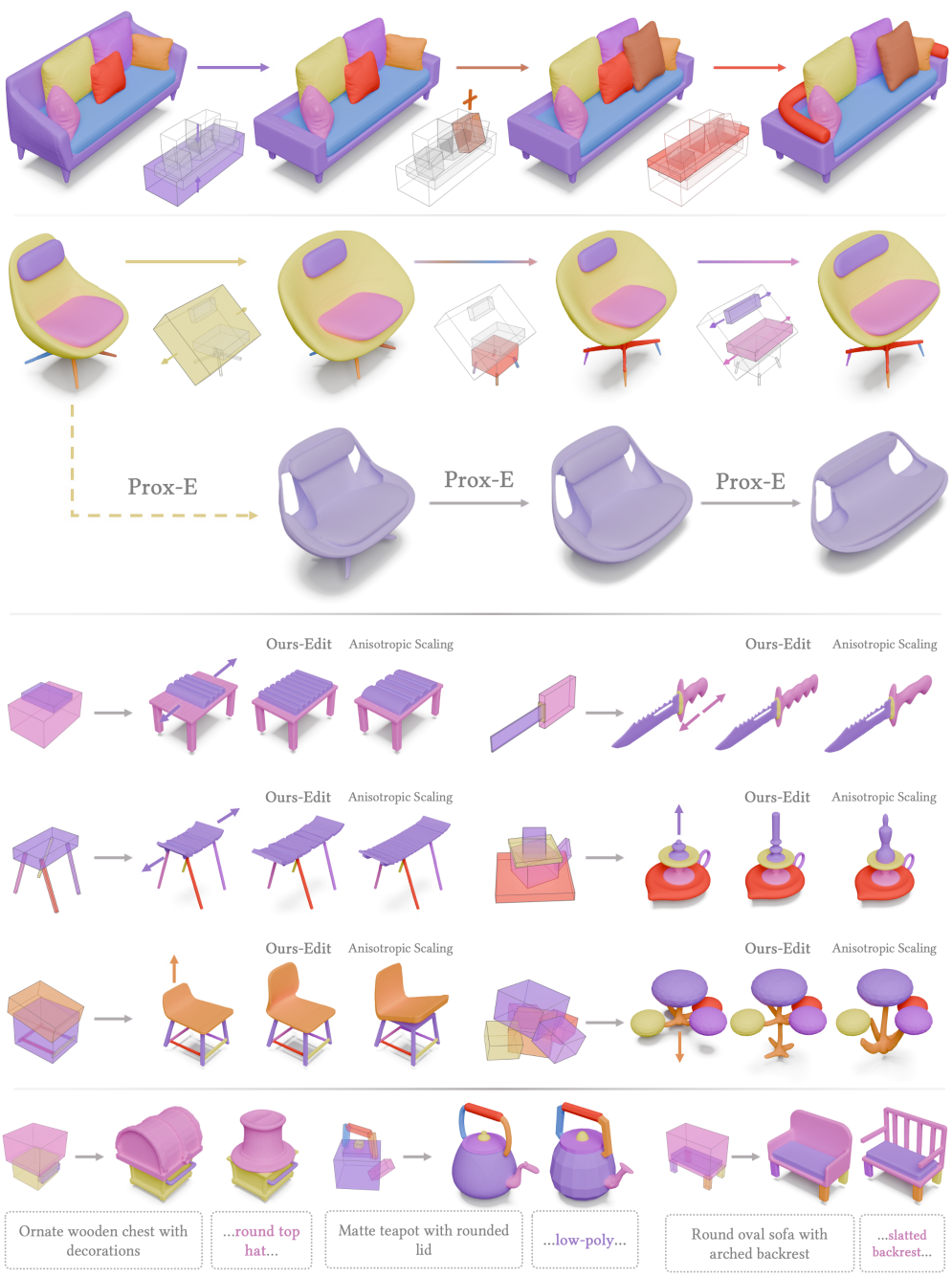


Figure 5: **Part-Level Editing.** We demonstrate our method's versatility through diverse editing operations. In the first two rows, we showcase edit sequences starting from a base shape. We iteratively add parts by creating new boxes or substitute them by rescaling without style preservation. In the third row, we compare with Prox-E [85] on the same edit chain, using the variant that obtains its primitives by inverting the input layout (Prox-E$_{\text{PRIMINVERT}}$). Since it cannot keep unedited parts fixed, it regenerates the whole object at each step: parts that were not edited change as well, and the shape drifts further from the original along the chain. In the following three rows, we show examples of identity-preserving edits, where we rescale parts while preserving their geometric identity, as described in appendix D. We compare our method with Anisotropic Scaling, and provide a quantitative evaluation against Anisotropic Scaling and Part Substitution in table 4 of the supplementary. Finally, the last row shows text-based part editing: changing the prompt under a fixed layout yields localized edits that preserve the shape's structure and style.

Table 1: **Guided Synthesis on Objaverse**. We compare our method with proxy-guided methods — Fantasia3D [12], Coin3D [23] + ControlNet [117], Spice-E [84] and SpaceControl [28] — which, except for Spice-E†, support open-vocabulary text-to-3D guided synthesis similarly to ours, and with the image-conditioned part synthesis methods PartCrafter [59] and OmniPart [109]. For image-conditioned part synthesis methods, which do not natively support 3D part layout control, we first render the 3D layout and generate a matching 2D image of the object by prompting Nano Banana Pro [30], which is then provided as input. †Spice-E evaluation was computed on a smaller subset containing only table, chair, and airplane categories, since it only supports these classes.

Method	Paradigm	Conditioning	FID ↓		IoU ↑			CLIP↑
			Object	Part	Object	Voxel	Part	
Proxy-Guided Synthesis								
Fantasia3D [12]	SDS [75]	Proxy Mesh	77.64	⊗	0.259	0.164	⊗	27.22
Coin3D [23]	SDS [75]	Proxy Mesh	95.14	⊗	0.168	0.095	⊗	25.10
Spice-E [84] †	Neural Field Diff. [47]	Proxy Mesh	90.30	⊗	0.640	0.237	⊗	26.40
SpaceControl [28]	Structured Latents [105]	Superquadrics	42.96	⊗	0.807	**0.776**	⊗	26.73
Image-Conditioned Part Synthesis								
PartCrafter [59] + Nano Banana [30]	VecSet [113]	Image	21.10	31.54	0.408	0.203	0.148	28.81
OmniPart [109] + Nano Banana [30]	Structured Latents [105]	Image	12.73	16.38	0.513	0.269	0.262	29.64
Ours	VecSet [113]	Part Layout (fig. 2)	**10.58**	**9.24**	**0.812**	0.550	**0.819**	**31.36**

Table 2: **GPT-Evaluation of Shape Generation and Editing**. We compare our method against each baseline by prompting GPT-4o with the target layout, the text prompt, and both outputs, asking which one is better along three axes, with ties allowed; each comparison is evaluated under both A/B orderings to cancel position bias. We report our win rate in % (higher is better ↑), with **Overall** averaging the three axes. **Quality** is overall realism, **Layout Adherence** agreement with the target layout, **Prompt Adherence** conformance to the text instruction, and **Identity Preservation** fidelity to the style, details and untouched parts of the initial shape, which GPT-4o is additionally shown for editing.

Controllable Shape Generation				
Ours vs.	Quality ↑	Layout Adherence ↑	Prompt Adherence ↑	Overall ↑
Proxy-Guided Synthesis				
Fantasia3D [12]	90.0	86.7	78.3	85.0
Coin3D [23]	100.0	94.2	91.7	95.3
Spice-E [84]	93.3	68.3	90.0	83.9
SpaceControl [28]	90.0	54.2	86.7	76.9
Image-Conditioned Part Synthesis				
PartCrafter [59] + Nano Banana [30]	85.0	75.0	76.0	78.7
OmniPart [109] + Nano Banana [30]	60.5	54.0	59.5	58.0
Shape Editing				
Ours vs.	Quality ↑	Layout Adherence ↑	Identity Preservation ↑	Overall ↑
Prox-E$_{\text{PRIMINVERT}}$ [85]	87.0	65.2	91.3	81.2
Prox-E$_{\text{SUPERDEC}}$ [85]	91.3	67.4	93.5	84.1

Baselines. To our knowledge, there is no prior work that accepts granular part-based oriented controls in combination with text prompts as we do. We therefore compare against two families of synthesis methods, reported as separate blocks in table 1, and against a dedicated editing baseline.

Proxy-guided synthesis methods accept coarse, non-part-separated shape guidance: Spice-E [84] (diffusion structural guidance), Fantasia3D [12] (SDS initialization), and Coin3D [23] combined with ControlNet [117] (proxy mesh conditioning). We additionally compare to SpaceControl [28], which accepts superquadric conditioning by instantiating one superquadric per control box.

Image-conditioned part synthesis methods decompose a single input image into separate 3D parts: PartCrafter [59] and OmniPart [109]. Unlike our method, these approaches take no 3D layout control, so we first render the target layout and use Nano Banana Pro [30] to generate an image of the object from it, which we provide as input. This comparison quantifies the cost of routing 3D struc-

tural intent through a 2D image: the conditioning image is an imperfect and ambiguous proxy for the layout, and any structure it fails to convey — in particular occluded parts and precise part extents — cannot be recovered downstream.

Editing baselines. For part-level editing, we compare against Prox-E [85], which edits shapes through coarse primitive-based abstractions, in two variants that differ only in how its primitives are obtained: Prox-E$_{\text{PRIMINVERT}}$ isolates the editing mechanism under matched control, while Prox-E$_{\text{SUPERDEC}}$ reflects how the method is used in practice. We report both comparisons in the editing block of table 2.

We detail the evaluation protocol we follow for every baseline — metric coverage, layout-to-image conditioning, part matching, and the two Prox-E variants — in appendix E.1 of the supplementary.

As shown in table 1, our method achieves the best Object-FID (10.58), Part-FID (9.24), Object-IoU (0.812), Part-IoU (0.819) and CLIP-Score (31.36), losing only Voxel-IoU to SpaceControl (0.776), which densely fills its superquadric proxies rather than synthesizing meaningful part geometry. The image-conditioned part synthesis methods fail in the opposite way: strong image priors give them competitive visual quality (12.73 Object-FID for OmniPart), but they follow the requested structure poorly, reaching at most 0.513 Object-IoU and 0.262 Part-IoU against our 0.812 and 0.819. This gap is a direct consequence of routing 3D intent through a 2D image, and controlling parts natively in 3D, as we do, removes this bottleneck.

GPT Evaluation. To complement our quantitative metrics with a holistic perceptual assessment, we conduct an automated evaluation using GPT-4o [44] as a judge. We report both evaluations in table 2, and describe the full protocol, prompts, and sample grids in appendix E.2.

For generation, our method is preferred over every baseline on every axis, with Overall win rates from 95.3% against Coin3D down to 58.0% against OmniPart, where Layout Adherence is the tightest axis (54.0%). For editing, our edits win 81.2% and 84.1% overall against the two Prox-E variants, with the largest gap on Identity Preservation (91.3% and 93.5%); the advantage holds even under matched control, so it comes from our editing mechanism rather than from the control signal.

Inference Time. Our method offers competitive inference time compared to prior work, scaling gracefully with the number of parts N_p; we report a full comparison against baselines in appendix J.4 of the supplementary.

5 LIMITATIONS

Our method has several limitations that motivate future work. First, slight artifacts can appear in synthesized parts due to the out-of-distribution nature of part segments for the VAE, requiring a filtering stage; fine-tuning the VAE on part-decomposed data could help. Second, the method is capped at 16 parts per shape, but can be trained on higher part counts by leveraging more memory-efficient attention schemes [51, 31, 7]. Finally, physical stability is not guaranteed: missing contacts or asymmetries between parts can compromise structural integrity, which could be addressed by fine-tuning against a stability reward from a physical simulator [24, 77].

6 CONCLUSION

We introduced CompoSE, the first method to synthesize part-separated 3D shapes from coarse oriented part layouts, thereby enabling compositional editing of individual object parts. Our method relies on a conditioning technique that ensures strong adherence to input part layouts, a diffusion transformer architecture that fuses part conditioning with global context, and a data processing pipeline that enables training on large 3D shape datasets without manual part annotations. It supports a versatile set of editing capabilities, including substitution, addition, and style-preserving resizing of individual parts, and outperforms existing approaches as measured by objective metrics and LLM-based evaluations. In the future, richer structured guidance, such as proxy meshes with animation rigs and physical affordances, could enable the synthesis of photorealistic, simulation-ready 3D assets. Beyond 3D, similar proxies inferred from videos could guide the generation of physically grounded video variations.

REFERENCES

[1] Maneesh Agrawala. Unpredictable Black Boxes are Terrible Interfaces. https://magrawala.substack.com/p/unpredictable-black-boxes-are-terrible, 2023.

[2] Omri Avrahami, Or Patashnik, Ohad Fried, Egor Nemchinov, Kfir Aberman, Dani Lischinski, and Daniel Cohen-Or. Stable Flow: Vital Layers for Training-Free Image Editing. In *CVPR*, 2025.

[3] Roi Bar-On, Dana Cohen-Bar, and Daniel Cohen-Or. EditP23: 3D Editing via Propagation of Image Prompts to Multi-View. *arXiv preprint arXiv:2506.20652*, 2025.

[4] Amir Barda, Matheus Gadelha, Vladimir G Kim, Noam Aigerman, Amit H Bermano, and Thibault Groueix. Instant3dit: Multiview Inpainting for Fast Editing of 3D Objects. In *CVPR*, 2025.

[5] Shariq Farooq Bhat, Niloy Mitra, and Peter Wonka. LooseControl: Lifting ControlNet for Generalized Depth Conditioning. In *SIGGRAPH*, 2024.

[6] Hansheng Chen, Bokui Shen, Yulin Liu, Ruoxi Shi, Linqi Zhou, Connor Z Lin, Jiayuan Gu, Hao Su, Gordon Wetzstein, and Leonidas Guibas. 3D-Adapter: Geometry-Consistent Multi-View Diffusion for High-Quality 3D Generation. *arXiv preprint arXiv:2410.18974*, 2024.

[7] Junsong Chen, Jincheng Yu, Yitong Li, Shuchen Xue, Haozhe Liu, Jingyu Xin, Yuyang Zhao, Tian Ye, Zhangjie Wu, Zian Wang, Daquan Zhou, Ping Luo, Song Han, and Enze Xie. SANA-Video 2.0: Hybrid Linear Attention with Attention Residuals for Efficient Video Generation. *arXiv preprint arXiv:2607.21553*, 2026.

[8] Minghao Chen, Iro Laina, and Andrea Vedaldi. DGE: Direct Gaussian 3D Editing by Consistent Multi-view Editing. In *ECCV*, 2024.

[9] Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. SHAP-EDITOR: Instruction-guided Latent 3D Editing in Seconds. In *CVPR*, 2024.

[10] Qimin Chen, Zhiqin Chen, Vladimir G. Kim, Noam Aigerman, Hao Zhang, and Siddhartha Chaudhuri. DECOLLAGE: 3D Detailization by Controllable, Localized, and Learned Geometry Enhancement. In *ECCV*, 2024.

[11] Qimin Chen, Yuezhi Yang, Wang Yifan, Vladimir G. Kim, Siddhartha Chaudhuri, Hao Zhang, and Zhiqin Chen. ART-DECO: Arbitrary Text Guidance for 3D Detailizer Construction. In *SIGGRAPH Asia*, 2025.

[12] 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.

[13] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting. In *CVPR*, 2024.

[14] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. MeshAnything: Artist-Created Mesh Generation with Autoregressive Transformers. In *ICLR*, 2025.

[15] Yiwen Chen, Zhihao Li, Yikai Wang, Hu Zhang, Qin Li, Chi Zhang, and Guosheng Lin. Ultra3D: Efficient and High-Fidelity 3D Generation with Part Attention. *arXiv preprint arXiv:2507.17745*, 2025.

[16] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. InternVL: Scaling up Vision Foundation Models and Aligning for Generic Visual-Linguistic Tasks. In *CVPR*, 2024.

[17] Yufeng Chi, Huimin Ma, Kafeng Wang, and Jianmin Li. DisCo3D: Distilling Multi-View Consistency for 3D Scene Editing. *arXiv preprint arXiv:2508.01684*, 2025.

[18] Dale Decatur, Itai Lang, Kfir Aberman, and Rana Hanocka. 3D Paintbrush: Local Stylization of 3D Shapes with Cascaded Score Distillation. In *CVPR*, 2024.

[19] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A Universe of Annotated 3D Objects. In *CVPR*, 2023.

[20] Nam Anh Dinh, Itai Lang, Hyunwoo Kim, Oded Stein, and Rana Hanocka. Geometry in Style: 3D Stylization via Surface Normal Deformation. In *CVPR*, 2025.

[21] Shaocong Dong, Lihe Ding, Zhanpeng Huang, Zibin Wang, Tianfan Xue, and Dan Xu. Interactive3D: Create What You Want by Interactive 3D Generation. In *CVPR*, 2024.

[22] Shaocong Dong, Lihe Ding, Xiao Chen, Yaokun Li, Yuxin Wang, Yucheng Wang, Qi Wang, Jaehyeok Kim, Chenjian Gao, Zhanpeng Huang, et al. From One to More: Contextual Part Latents for 3D Generation. In *ICCV*, 2025.

[23] Wenqi Dong, Bangbang Yang, Lin Ma, Xiao Liu, Liyuan Cui, Hujun Bao, Yuewen Ma, and Zhaopeng Cui. Coin3D: Controllable and Interactive 3D Assets Generation with Proxy-Guided Conditioning. In *SIGGRAPH*, 2024.

[24] Yuan Dong, Qi Zuo, Xiaodong Gu, Weihao Yuan, Zhengyi Zhao, Zilong Dong, Liefeng Bo, and Qixing Huang. GPLD3D: Latent Diffusion of 3D Shape Generative Models by Enforcing Geometric and Physical Priors. In *CVPR*, 2024.

[25] Slava Elizarov, Ciara Rowles, and Simon Donné. Geometry Image Diffusion: Fast and Data-Efficient Text-to-3D with Image-Based Surface Representation. In *ICLR*, 2025.

[26] Ziya Erkoç, Can Gümeli, Chaoyang Wang, Matthias Niessner, Angela Dai, Peter Wonka, Hsin-Ying Lee, and Peiye Zhuang. PrEditor3D: Fast and Precise 3D Shape Editing. In *CVPR*, 2025.

[27] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling Rectified Flow Transformers for High-Resolution Image Synthesis. In *ICML*, 2024.

[28] Elisabetta Fedele, Francis Engelmann, Ian Huang, Or Litany, Marc Pollefeys, and Leonidas Guibas. SpaceControl: Introducing Test-Time Spatial Control to 3D Generative Modeling. In *ICLR*, 2026.

[29] Aditya Ganeshan, R. Kenny Jones, and Daniel Ritchie. Improving Unsupervised Visual Program Inference with Code Rewriting Families. In *ICCV*, 2023.

[30] Google DeepMind. Nano Banana Pro: Gemini 3 Pro Image Model from Google DeepMind. `https://blog.google/innovation-and-ai/products/nano-banana-pro/`, 2025. Accessed: 2025-11-20.

[31] Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In *COLM*, 2024.

[32] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry Images. *ACM TOG (Proc. SIGGRAPH)*, 2002.

[33] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions. In *ICCV*, 2023.

[34] 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.

[35] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. *ACM TOG*, 2022.

[36] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. CLIPScore: A Reference-free Evaluation Metric for Image Captioning. In *EMNLP*, 2021.

[37] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *NeurIPS*, 2017.

[38] HexaGen. HexaGen: Community Creations. `https://www.hexagen.ai`, 2024. Accessed: 2024-11-12.

[39] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[40] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *NeurIPS*, 2020.

[41] Hugues Hoppe. Overview of Recent Work on Geometry Images. In *Geometric Modeling and Processing*, 2004.

[42] Wentao Hu, Jia Zheng, Zixin Zhang, Xiaojun Yuan, Jian Yin, and Zihan Zhou. PlankAssembly: Robust 3D Reconstruction from Three Orthographic Views with Learnt Shape Programs. In *ICCV*, 2023.

[43] Team Hunyuan3D, Bowen Zhang, Chunchao Guo, Dongyuan Guo, Haolin Liu, Hongyu Yan, Huiwen Shi, Jiaao Yu, Jiachen Xu, Jingwei Huang, Kunhong Li, Lifu Wang, Linus, Penghao Wang, Qingxiang Lin, Ruining Tang, Xianghui Yang, Yang Li, Yirui Guan, Yunfei Zhao, Yunhan Yang, Zeqiang Lai, Zhihao Liang, and Zibo Zhao. HY3D-Bench: Generation of 3D Assets. *arXiv preprint arXiv:2602.03907*, 2026.

[44] Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. GPT-4o System Card. *arXiv preprint arXiv:2410.21276*, 2024.

[45] Hyper3D. Rodin AI: Free AI 3D Model Generator for Everyone. `https://hyper3d.ai/rodin`, 2024. Accessed: 2024-11-12.

[46] R. Kenny Jones, Theresa Barton, Xianghao Xu, Kai Wang, Ellen Jiang, Paul Guerrero, Niloy J. Mitra, and Daniel Ritchie. ShapeAssembly: Learning to Generate Programs for 3D Shape Structure Synthesis. *ACM TOG*, 2020.

[47] Heewoo Jun and Alex Nichol. Shap-E: Generating Conditional 3D Implicit Functions. *arXiv preprint arXiv:2305.02463*, 2023.

[48] Jukka Jylänki. An Exact Algorithm for Finding Minimum Oriented Bounding Boxes, 2015. Technical report.

[49] Hiromichi Kamata, Yuiko Sakuma, Akio Hayakawa, Masato Ishii, and Takuya Narihira. Instruct 3D-to-3D: Text Instruction Guided 3D-to-3D Conversion. *arXiv preprint arXiv:2303.15780*, 2023.

[50] Animesh Karnewar, Andrea Vedaldi, David Novotny, and Niloy J Mitra. HoloDiffusion: Training a 3D Diffusion Model using 2D Images. In *CVPR*, 2023.

[51] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *ICML*, 2020.

[52] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2Video-Zero: Text-to-Image Diffusion Models are Zero-Shot Video Generators. In *ICCV*, 2023.

[53] Juil Koo, Seungwoo Yoo, Minh Hieu Nguyen, and Minhyuk Sung. SALAD: Part-Level Latent Diffusion for 3D Shape Generation and Manipulation. In *ICCV*, 2023.

[54] Lin Li, Zehuan Huang, Haoran Feng, Gengxiong Zhuang, Rui Chen, Chunchao Guo, and Lu Sheng. VoxHammer: Training-Free Precise and Coherent 3D Editing in Native 3D Space. In *3DV*, 2026.

[55] Peng Li, Suizhi Ma, Jialiang Chen, Yuan Liu, Congyi Zhang, Wei Xue, Wenhan Luo, Alla Sheffer, Wenping Wang, and Yike Guo. CMD: Controllable Multiview Diffusion for 3D Editing and Progressive Generation. In *SIGGRAPH*, 2025.

[56] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. CraftsMan3D: High-fidelity Mesh Generation with 3D Native Diffusion and Interactive Geometry Refiner. In *CVPR*, 2025.

[57] Yangguang Li, Yuanbo Yang, Kaicheng Yu, Libo Zhou, Hongsheng Li, Guangming Zhu, Yuzhong Zhao, Peilin Zhao, Zhou Zhao, and Lei Zhang. TripoSG: High-Fidelity 3D Shape Synthesis using Large-Scale Rectified Flow Models. *IEEE TPAMI*, 2025.

[58] Yuchen Li, Ujjwal Upadhyay, Habib Slim, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT: Composition of Materials on Parts of 3D Things. In *ECCV*, 2022.

[59] 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.

[60] Fangfu Liu, Hanyang Wang, Weiliang Chen, Haowen Sun, and Yueqi Duan. Make-Your-3D: Fast and Consistent Subject-Driven 3D Content Generation. In *ECCV*, 2024.

[61] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *ICLR*, 2023.

[62] William E. Lorensen and Harvey E. Cline. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. In *SIGGRAPH*, 1987.

[63] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *ICLR*, 2019.

[64] Shalini Maiti, Lourdes Agapito, and Filippos Kokkinos. Gen3DEval: Using vLLMs for Automatic Evaluation of Generated 3D Objects. In *CVPR*, 2025.

[65] Nissim Maruani, Wang Yifan, Matthew Fisher, Pierre Alliez, and Mathieu Desbrun. ShapeShifter: 3D Variations Using Multiscale and Sparse Point-Voxel Diffusion. In *CVPR*, 2025.

[66] Meshy AI. Meshy. `https://www.meshy.ai`, 2024. Accessed: 2024-11-12.

[67] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. AutoSDF: Shape Priors for 3D Completion, Reconstruction and Generation. In *CVPR*, 2022.

[68] George Kiyohiro Nakayama, Mikaela Angelina Uy, Jiahui Huang, Shi-Min Hu, Ke Li, and Leonidas Guibas. DiffFacto: Controllable Part-Based 3D Point Cloud Generation with Cross Diffusion. In *ICCV*, 2023.

[69] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An Autoregressive Generative Model of 3D Meshes. In *ICML*, 2020.

[70] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A System for Generating 3D Point Clouds from Complex Prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[71] Michael Niemeyer and Andreas Geiger. GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields. In *CVPR*, 2021.

[72] Jangho Park, Gihyun Kwon, and Jong Chul Ye. ED-NeRF: Efficient Text-Guided Editing of 3D Scene with Latent Space NeRF. In *ICLR*, 2024.

[73] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation. In *CVPR*, 2019.

[74] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *ICCV*, 2023.

[75] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*, 2023.

[76] Rolandos Alexandros Potamias, Michail Tarasiou, Stylianos Ploumpis, and Stefanos Zafeiriou. ShapeFusion: A 3D Diffusion Model for Localized Shape Editing. In *ECCV*, 2024.

[77] Ava Pun, Kangle Deng, Ruixuan Liu, Deva Ramanan, Changliu Liu, and Jun-Yan Zhu. Generating Physically Stable and Buildable Brick Structures from Text. In *ICCV*, 2025.

[78] Zhangyang Qi, Yunhan Yang, Mengchen Zhang, Long Xing, Xiaoyang Wu, Tong Wu, Dahua Lin, Xihui Liu, Jiaqi Wang, and Hengshuang Zhao. Tailor3D: Customized 3D Assets Editing and Generation with Dual-Side Images. *arXiv preprint arXiv:2407.06191*, 2024.

[79] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*, 2021.

[80] 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*, 2024.

[81] Roblox Foundation AI Team. Cube: A Roblox View of 3D Intelligence. *arXiv preprint arXiv:2503.15475*, 2025.

[82] Aditya Sanghi, Rao Fu, Vivian Liu, Karl D.D. Willis, Hooman Shayani, Amir H. Khasahmadi, Srinath Sridhar, and Daniel Ritchie. CLIP-Sculptor: Zero-Shot Generation of High-Fidelity and Diverse Shapes from Natural Language. In *CVPR*, 2023.

[83] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-E: Text-guided Voxel Editing of 3D Objects. In *ICCV*, 2023.

[84] Etai Sella, Gal Fiebelman, Noam Atia, and Hadar Averbuch-Elor. Spice-E: Structural Priors in 3D Diffusion using Cross-Entity Attention. In *SIGGRAPH*, 2024.

[85] Etai Sella, Hao Phung, Nitay Amiel, Or Litany, Or Patashnik, and Hadar Averbuch-Elor. Prox-E: Fine-Grained 3D Shape Editing via Primitive-Based Abstractions. In *SIGGRAPH*, 2026.

[86] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Niessner. MeshGPT: Generating Triangle Meshes with Decoder-Only Transformers. In *CVPR*, 2024.

[87] Habib Slim and Mohamed Elhoseiny. ShapeWalk: Compositional Shape Editing through Language-Guided Chains. In *CVPR*, 2024.

[88] Habib Slim, Xiang Li, Yuchen Li, Mahmoud Ahmed, Mohamed Ayman, Ujjwal Upadhyay, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, and Mohamed Elhoseiny. 3DCoMPaT++: An Improved Large-scale 3D Vision Dataset for Compositional Recognition. *arXiv preprint arXiv:2212.11715*, 2023.

[89] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, 2015.

[90] 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.

[91] Tencent Hunyuan3D Team. Hunyuan3D 1.0: A Unified Framework for Text-to-3D and Image-to-3D Generation. *arXiv preprint arXiv:2411.02293*, 2024.

[92] Tencent Hunyuan3D Team. Hunyuan3D 2.0: Scaling Diffusion Models for High Resolution Textured 3D Assets Generation. *arXiv preprint arXiv:2501.12202*, 2025.

[93] Tencent Hunyuan3D Team. Hunyuan3D 2.5: Towards High-Fidelity 3D Assets Generation with Ultimate Details. *arXiv preprint arXiv:2506.16504*, 2025.

[94] Cyrus Vachha and Ayaan Haque. Instruct-GS2GS: Editing 3D Gaussian Splats with Instructions. `https://instruct-gs2gs.github.io/`, 2024.

[95] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *NeurIPS*, 2017.

[96] Mathias Vogel, Keisuke Tateno, Marc Pollefeys, Federico Tombari, Marie-Julie Rakotosaona, and Francis Engelmann. P2P-Bridge: Diffusion Bridges for 3D Point Cloud Denoising. In *ECCV*, 2024.

[97] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. SV3D: Novel Multi-view Synthesis and 3D Generation from a Single Image using Latent Video Diffusion. In *ECCV*, 2024.

[98] Chen Wang, Hao-Yang Peng, Ying-Tian Liu, Jiatao Gu, and Shi-Min Hu. Diffusion Models for 3D Generation: A Survey. *Computational Visual Media*, 2025.

[99] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM TOG*, 2019.

[100] Minghao Wen, Shengjie Wu, Kangkan Wang, and Dong Liang. InterGSEdit: Interactive 3D Gaussian Splatting Editing with 3D Geometry-Consistent Attention Prior. In *ICCV*, 2025.

[101] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Prisacariu. GaussCtrl: Multi-View Consistent Text-Driven 3D Gaussian Splatting Editing. In *ECCV*, 2024.

[102] Lemeng Wu, Dilin Wang, Chengyue Gong, Xingchao Liu, Yunyang Xiong, Rakesh Ranjan, Raghuraman Krishnamoorthi, Vikas Chandra, and Qiang Liu. Fast Point Cloud Generation with Straight Flows. In *CVPR*, 2023.

[103] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. GPT-4V(ision) is a Human-Aligned Evaluator for Text-to-3D Generation. In *CVPR*, 2024.

[104] Zijie Wu, Yaonan Wang, Mingtao Feng, He Xie, and Ajmal Mian. Sketch and Text Guided Diffusion Model for Colored Point Cloud Generation. In *ICCV*, 2023.

[105] 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*, 2025.

[106] Teng Xu, Jiamin Chen, Peng Chen, Youjia Zhang, Junqing Yu, and Wei Yang. TIGER: Text-Instructed 3D Gaussian Retrieval and Coherent Editing. *arXiv preprint arXiv:2405.14455*, 2024.

[107] Yanbo Xu, Yu Wu, Sungjae Park, Zhizhuo Zhou, and Shubham Tulsiani. Temporal Score Rescaling for Temperature Sampling in Diffusion and Flow Models. In *ICML*, 2026.

[108] Xingguang Yan, Han-Hung Lee, Ziyu Wan, and Angel X Chang. An Object is Worth 64×64 Pixels: Generating 3D Object via Image Diffusion. In *3DV*, 2025.

[109] 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 *SIGGRAPH Asia*, 2025.

[110] 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.

[111] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent Point Diffusion Models for 3D Shape Generation. In *NeurIPS*, 2022.

[112] Biao Zhang, Matthias Niessner, and Peter Wonka. 3DILG: Irregular Latent Grids for 3D Generative Modeling. In *NeurIPS*, 2022.

[113] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *ACM TOG*, 2023.

[114] Hao Zhang, Yao Feng, Peter Kulits, Yandong Wen, Justus Thies, and Michael J Black. TECA: Text-Guided Generation and Editing of Compositional 3D Avatars. In *3DV*, 2024.

[115] 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. In *SIGGRAPH*, 2024.

[116] Longwen Zhang, Qixuan Zhang, Haoran Jiang, Yinuo Bai, Wei Yang, Lan Xu, and Jingyi Yu. BANG: Dividing 3D Assets via Generative Exploded Dynamics. In *SIGGRAPH*, 2025.

[117] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *ICCV*, 2023.

[118] Xinlu Zhang, Yujie Lu, Weizhi Wang, An Yan, Jun Yan, Lianke Qin, Heng Wang, Xifeng Yan, William Yang Wang, and Linda Ruth Petzold. GPT-4V(ision) as a Generalist Evaluator for Vision-Language Tasks. *arXiv preprint arXiv:2311.01361*, 2023.

[119] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A Modern Library for 3D Data Processing. *arXiv preprint arXiv:1801.09847*, 2018.

[120] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. DreamEditor: Text-Driven 3D Scene Editing with Neural Fields. In *SIGGRAPH Asia*, 2023.

[121] Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. TIP-Editor: An Accurate 3D Editor Following Both Text-Prompts and Image-Prompts. *ACM TOG*, 2024.

CompoSE: Supplementary Material

A Release Statement 18

B LLM Use Disclosure 18

C User Interface 18

D Part-Aware Editing 19

E Evaluation Details 19

E.1 Baselines 19

E.2 GPT-4o Evaluation 20

F Additional Samples 22

F.1 Edit Chains and Identity-Preserving Edits 22

F.2 Controllable Part Synthesis 23

F.3 Comparison with Controllable Baselines 28

F.4 Comparison with Part Synthesis Baselines 30

G Failure Cases 41

H Implementation Details 42

H.1 Architecture Details 42

H.2 CFG Control Annealing 43

H.3 Layout Optimization and Artifact Filtering 45

H.4 Training Details 47

I Dataset 48

I.1 Dataset Creation 48

I.2 Statistics 49

J Additional Ablations 50

J.1 SpaceControl Classifier-Free Guidance Scale 50

J.2 Identity-Preservation under Part Rescaling 51

J.3 Model Ablation Study 51

J.4 Inference Time 51

A Release Statement

Our code and dataset will be made publicly available upon publication.

B LLM Use Disclosure

We used large language models to aid and polish the writing of this paper, and a vision-language model (InternVL3-8B [16]) to generate the text captions of our synthetic training dataset, as described in appendix I.1.

C User Interface

For the purposes of easily interacting with our method, we have developed a simple user interface (UI) that allows users to input text prompts and define part layouts by manipulating boxes, using the Streamlit framework[1]. The UI provides direct feedback on the generated 3D shapes, enabling users to visualize and refine their designs interactively. We show the interface we have developed in fig. 6.

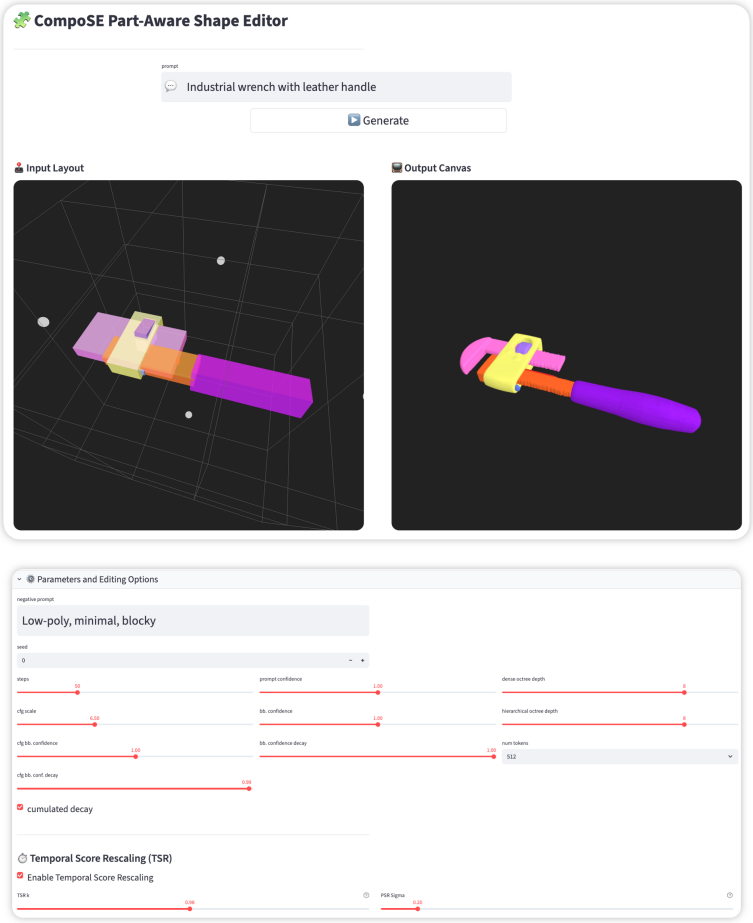


Figure 6: **CompoSE User Interface.** Our user interface allows users to input text prompts and define part layouts using bounding boxes. We show the layout editor and the output visualizer (top), and the parameter panel (bottom) for fine-tuning generation settings.

[1] https://streamlit.io/

D PART-AWARE EDITING

When performing part-aware editing operations (see fig. 5), we freeze part latents to control which parts are updated and which parts are preserved. Specifically, when a user edits only a subset of parts, we freeze the latents of all other parts by restoring their latents from the *reference shape* (i.e., the shape that has been synthesized by our model prior to any editing operations) at every diffusion time-step. This technique effectively locks the unedited parts in place, in the sense that they contribute to global consistency of the edited shapes, but do not themselves change their geometry or semantics.

In cases where the user wants to edit the geometry of a part while preserving its style and overall appearance, such as rescaling a part (fig. 5 rows 4–6), latent freezing is not sufficient. To enable this type of appearance-preserving edit, we use a *KV-reinjection* technique [5, 2, 52]. Specifically, during our model's forward pass for the reference shape, we cache the attention keys and values produced in the input attention layer and the text attention layer of every DiT block, and at every diffusion time-step. Then, during editing, we recompute keys and values from the modified latents, but we blend them with the cached keys and values in a time-step-dependent manner. This weighted reinjection anchors the part's visual identity, ensuring that changes to the part's corresponding geometric primitive (e.g., rescaling) do not cause the appearance of the part to drift (fig. 5 rows 4–6). As shown in fig. 5, our method yields rescaled parts whose style, surface detail, and geometric character remain faithful to the reference shape.

Together, part-latent freezing and KV-reinjection provide a unified and flexible editing framework. Latent freezing controls *where* the edit occurs, while KV-reinjection controls *how much* of the original identity should be preserved. This combination enables expressive operations: adding parts, substituting them through text prompts, or performing identity-preserving geometric adjustments, all while maintaining global coherence and respecting the user-specified layout.

E EVALUATION DETAILS

E.1 BASELINES

Here we detail the evaluation protocol we follow for each family of baselines.

Proxy-Guided Synthesis. Since Fantasia3D, Coin3D, and Spice-E do not synthesize decomposed parts, we only evaluate object-level metrics for these baselines; additionally, since Spice-E is fine-tuned only on table, chair, and airplane categories, we evaluate it on a filtered subset of these categories. Coin3D fails on roughly 10% of test instances despite using the hyperparameters recommended by its authors; these failures are excluded from its reported metrics. For SpaceControl, we report a single guidance scale in table 1 of the main paper, and provide a full sweep over this parameter, as well as a variant augmented with image conditioning via ControlNet [117], as an ablation in appendix J.1.

Image-Conditioned Part Synthesis. Since PartCrafter and OmniPart accept no 3D layout control, we must express the target layout in the only modality they accept: we render each test layout from a canonical viewpoint and prompt Nano Banana Pro [30] to synthesize an image of the object that follows both the rendered part arrangement and the text prompt, which we then provide as input. Because these methods output part-separated geometry, we evaluate them on part-level metrics as well. For fairness, we do not penalize them for producing a different part count than requested: we establish a one-to-one correspondence between generated and ground-truth parts via Hungarian matching, and average part-level metrics over the matched pairs only. We further score each generated asset under 25 canonical orientations and retain the best-scoring one, so that no method is penalized for a global rigid misalignment.

Editing Baselines. Both Prox-E [85] baselines run the method end to end, and differ only in how its primitives are obtained: Prox-E$_{\text{PRIMINVERT}}$ inverts the input primitive layout into the method's proxy representation, while Prox-E$_{\text{SUPERDEC}}$ follows the method's standard route and recovers primitives from the input shape through superquadric decomposition. In both cases the edit intent is expressed on those primitives and the edit itself is performed entirely by Prox-E.

E.2 GPT-4o Evaluation

Our use of a multimodal model as a judge is motivated by prior work demonstrating that such models are scalable, human-aligned evaluators for 3D [103, 64] and vision-language generation [118]. Rather than ranking all methods jointly, we run pairwise comparisons: for each test instance and each baseline, we compose a grid image showing the target layout alongside our output and the baseline's, and ask GPT-4o [44] which of the two is better along three axes, allowing ties. For generation, the axes are **Quality** (overall realism), **Layout Adherence** (agreement with the target layout), and **Prompt Adherence** (conformance to the text instruction); for editing, the grid additionally shows the initial shape, and Prompt Adherence is replaced by **Identity Preservation** (fidelity to the style, details, and untouched parts of the initial shape). Every comparison is evaluated under both A/B orderings so that any positional bias cancels out, and we report our win rate in %, counting each tie as half a win (i.e., $(\text{wins} + 0.5 \times \text{ties})/\text{comparisons}$), with **Overall** averaging the three axes. For Layout Adherence, we explicitly instruct GPT-4o to treat the input boxes as loose bounding volumes rather than target shapes. Without this instruction, we found that the judge tends to reward outputs that simply fill each box with a block-like shape, which trivially maximizes spatial overlap while producing implausible geometry; the instruction aligns the criterion with the intended use of layouts as coarse guidance, consistent with our IoU-based metrics which measure bounding-box overlap rather than volumetric filling. The results are given in table 2 of the main paper.

Controllable Generation Evaluation. We provide the complete prompt used for our GPT-4o controllable generation evaluation in fig. 8, alongside samples of the image grids shown to GPT-4o for evaluation in fig. 7.

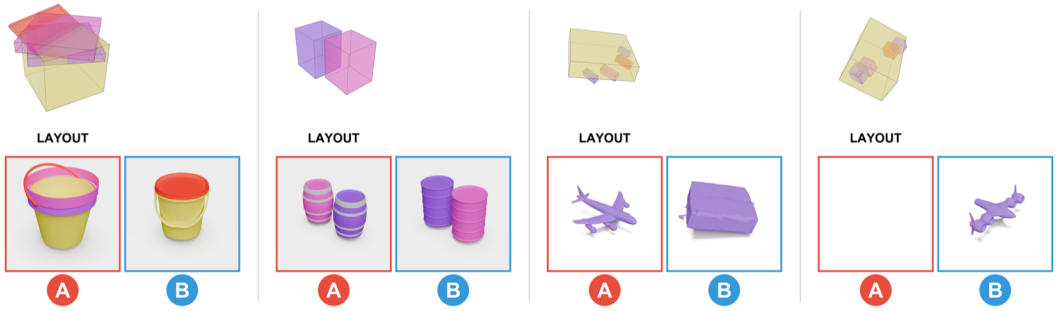


Figure 7: **Generated Grids for GPT-4o Pairwise Comparison.** We show samples of the image grids shown to GPT-4o for evaluating controllable generation, which contain the target layout (top), and the shapes generated by our method and one baseline (bottom). Each sample is evaluated under both A/B orderings of the two methods, so that any bias towards a spatial location cancels out. When a method fails (rightmost sample, tile A), we replace the corresponding image with a white tile.

GPT-4o Controllable Generation Evaluation Prompt

```
This image shows a 3D shape generation task.
Top-left (LAYOUT): the target bounding-box layout of the shape's parts, given as
generation guidance.
Bottom row: two 3D shapes generated by two different methods, labeled A and B.
The intended text description of the shape is: "{text description}"
Compare A and B. For each question, answer "A", "B", or "tie" if neither is clearly
better.
Ignore differences in color and rendering style.

Q1 (Overall Quality): Which generated shape has better overall quality and realism?
Q2 (Layout Adherence): Which generated shape better matches the target layout? Each
part (identified by its color-coded bounding box) should be in the position, scale,
and orientation shown in the layout. The boxes are only loose bounding volumes for
each part, not target shapes to fill --- a part that is stretched or blocked out into
a rectangular shape merely to fill its box completely should NOT be rewarded for this;
correct part geometry within a roughly matching box is what matters. A shape that is
completely distorted (broken, melted, missing parts, or no longer recognizable as the
object) must always be judged worse, even if its rough proportions match the layout.
Q3 (Prompt Adherence): Which generated shape better conforms to the text description
above?

Reply ONLY with a JSON object of the form
{"1": "A", "2": "tie", "3": "B"},
with no extra text, no markdown, no code fences.
```

Figure 8: Prompt provided to GPT-4o for comparing two generated 3D shapes against a target layout and text description. GPT-4o is asked to choose the better shape (or a tie) along three axes: overall quality (Q1), layout adherence (Q2), and prompt adherence (Q3). The model is constrained to respond with a strict JSON object containing one verdict per question.

Shape Editing Evaluation. For editing, we manually collect editing examples that each involve one to two parts, resulting in 200 evaluations in total. We provide the complete prompt used for our GPT-4o shape editing evaluation in fig. 9.

GPT-4o Shape Editing Evaluation Prompt

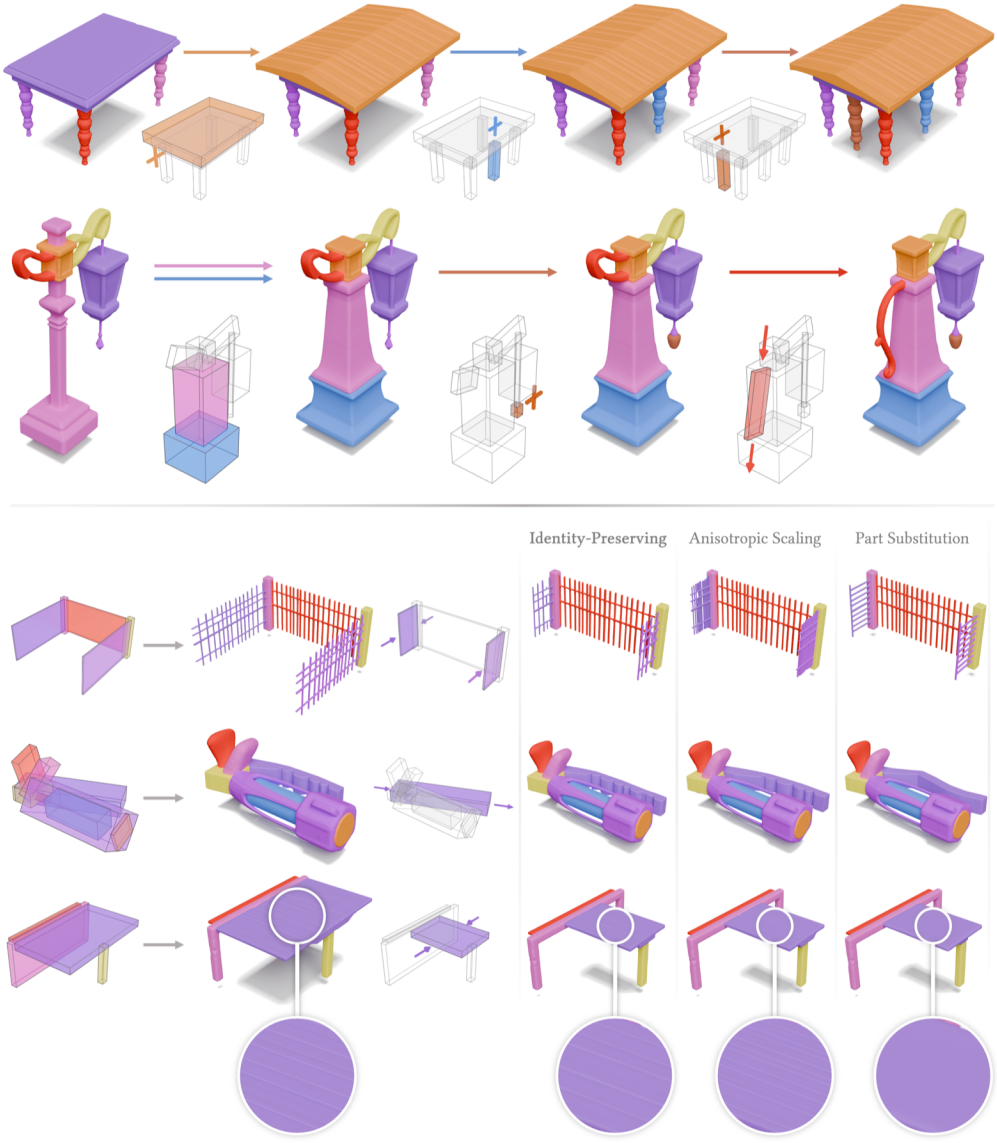
```
This image shows a 3D shape editing task.
Top-left (INITIAL): the initial 3D shape before editing.
Top-right (LAYOUT): the edit instruction, shown as the target bounding-box layout of the
shape's parts; arrows indicate how parts should be moved or resized.
Bottom row: two edited 3D shapes produced by two different methods, labeled A and B.
Compare A and B. For each question, answer "A", "B", or "tie" if neither is clearly
better.
Ignore differences in color and rendering style.

Q1 (Overall Quality): Which edited shape has better overall quality and realism?
Q2 (Layout Adherence): Which edited shape better matches the edit layout? A shape that
is completely distorted (broken, melted, missing parts, or no longer recognizable as the
object) must always be judged worse, even if its rough proportions match the layout.
Q3 (Identity Preservation): Which edited shape better preserves the identity of the
initial shape (its style, details, and the parts not targeted by the edit)?

Reply ONLY with a JSON object of the form
{"1": "A", "2": "tie", "3": "B"},
with no extra text, no markdown, no code fences.
```

Figure 9: Prompt provided to GPT-4o for comparing two 3D shape edits against the initial shape and the target layout. GPT-4o is asked to choose the better edit (or a tie) along three axes: overall quality (Q1), layout adherence (Q2), and identity preservation (Q3). The model is constrained to respond with a strict JSON object containing one verdict per question.

F Additional Samples

F.1 Edit Chains and Identity-Preserving Edits

We show additional results for our method's editing capabilities. The top two rows illustrate chained edits, where a sequence of localized part-level operations (e.g., addition, substitution, and resizing) is applied successively to the same base shape, with each step's target part highlighted alongside the resulting edit; the shape's identity and previously edited parts are preserved throughout the chain. The bottom three rows compare our identity-preserving part rescaling against Anisotropic Scaling and Part Substitution, with a zoomed-in inset highlighting how well each method preserves the original part's fine surface detail after resizing.

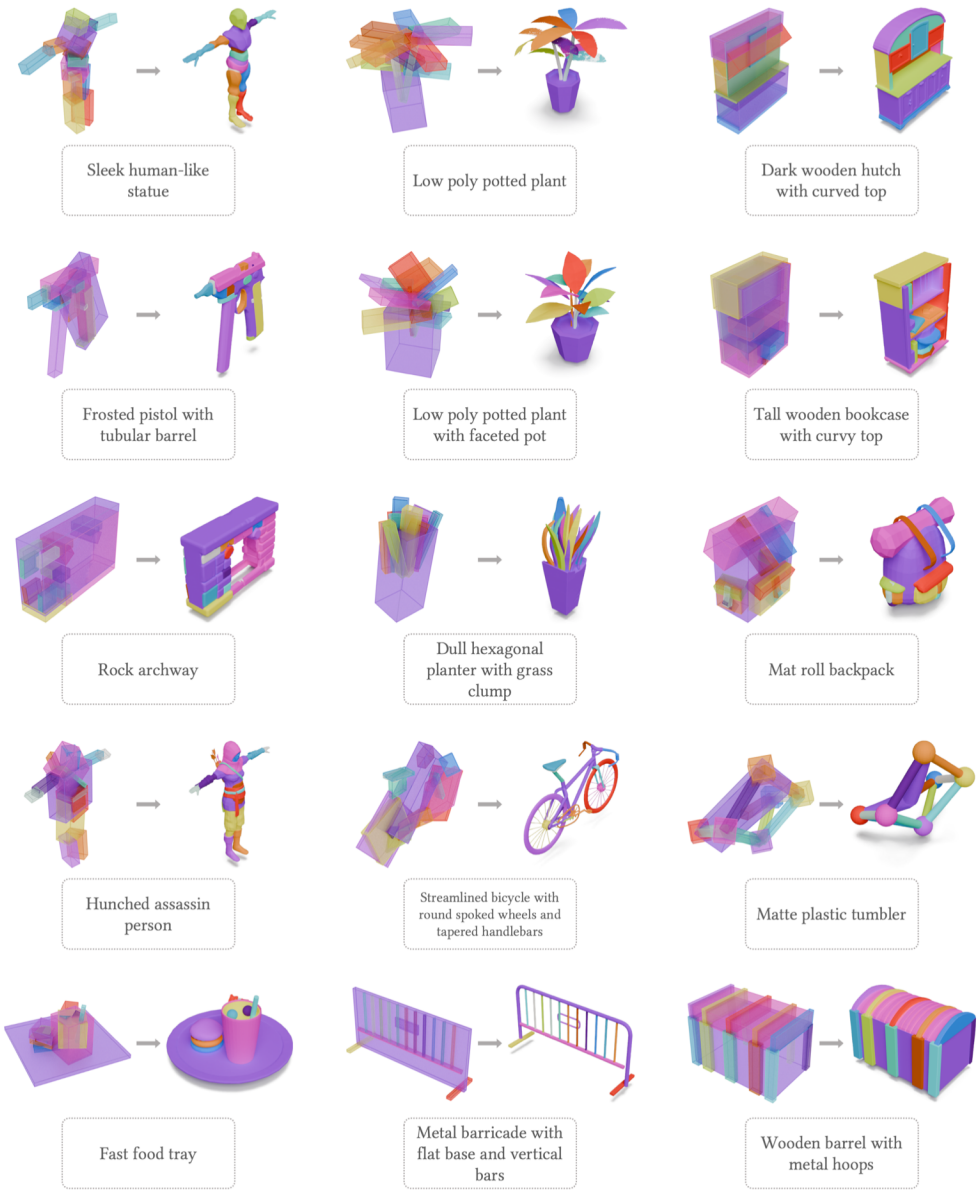


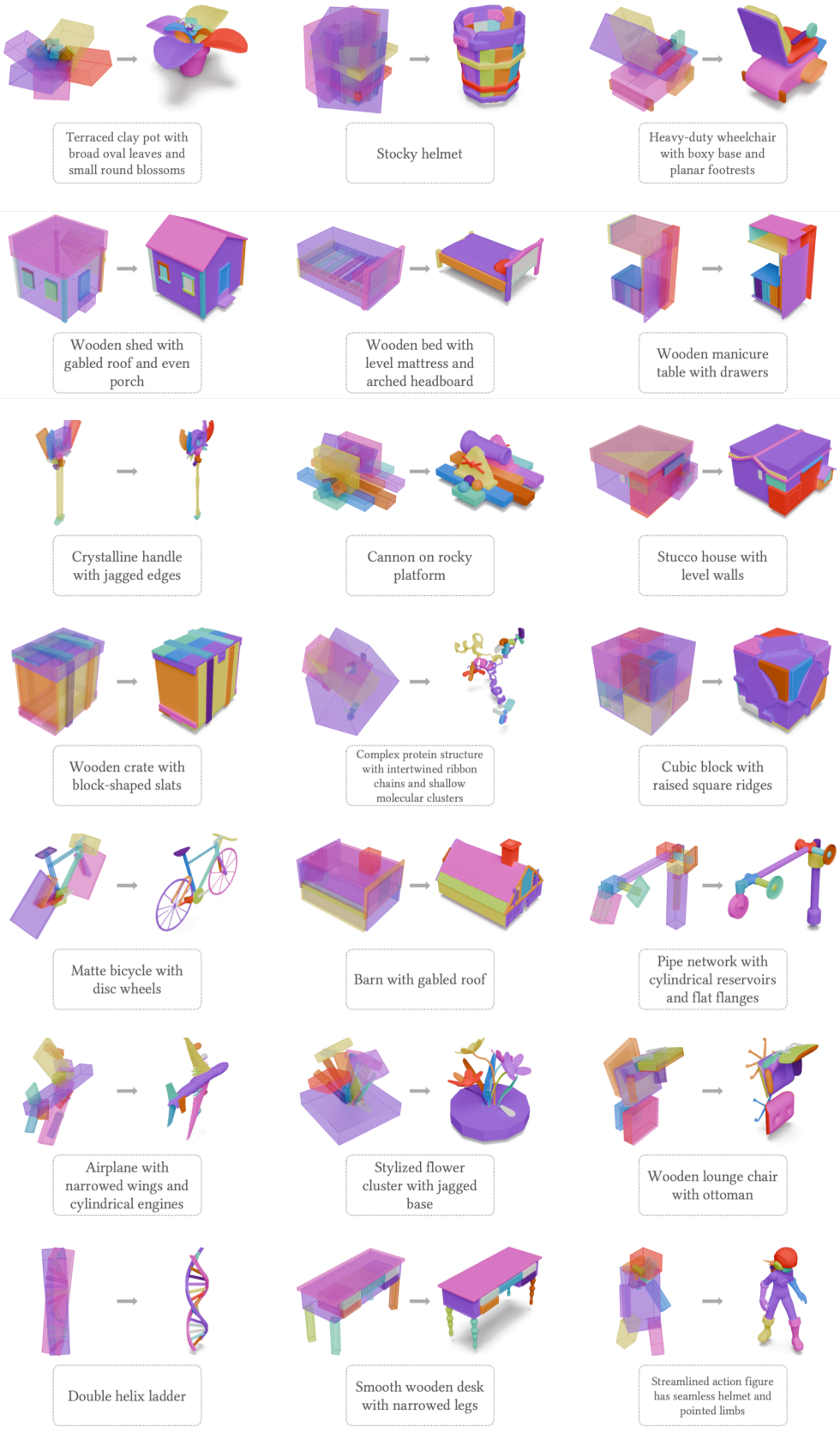
F.2 Controllable Part Synthesis

We provide additional qualitative results of our method's controllable part synthesis capabilities, sampling shapes with varying numbers of parts and levels of geometric detail from the input layout.

F.2.1 16-Part Generation at High Resolution (2048 Tokens)

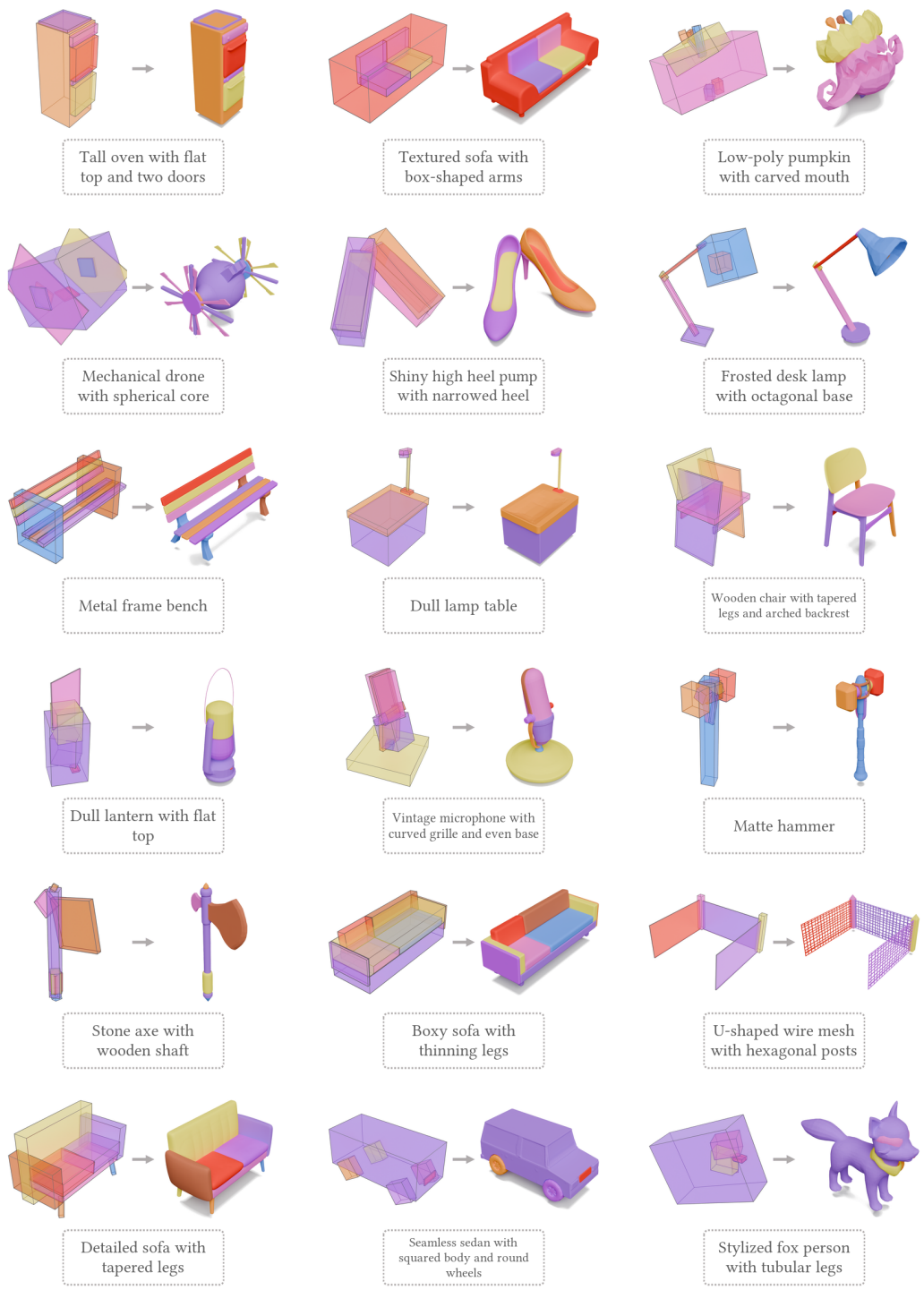
We show additional results generated with $D = 2048$ tokens per part, using input layouts composed of up to 16 parts. This higher-token-resolution variant is able to capture finer geometric detail across a larger number of parts per shape.

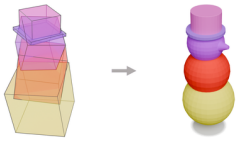



Terraced clay pot with broad oval leaves and small round blossoms
Stocky helmet
Heavy-duty wheelchair with boxy base and planar footrests
Wooden shed with gabled roof and even porch
Wooden bed with level mattress and arched headboard
Wooden manicure table with drawers
Crystalline handle with jagged edges
Cannon on rocky platform
Stucco house with level walls
Wooden crate with block-shaped slats
Complex protein structure with intertwined ribbon chains and shallow molecular clusters
Cubic block with raised square ridges
Matte bicycle with disc wheels
Barn with gabled roof
Pipe network with cylindrical reservoirs and flat flanges
Airplane with narrowed wings and cylindrical engines
Stylized flower cluster with jagged base
Wooden lounge chair with ottoman
Double helix ladder
Smooth wooden desk with narrowed legs
Streamlined action figure has seamless helmet and pointed limbs

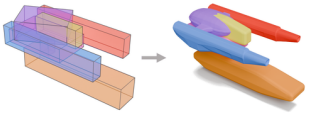
F.2.2 8-PART GENERATION AT BASE RESOLUTION (512 TOKENS)

We show additional results generated with our base model, using $D = 512$ tokens per part and input layouts composed of up to 8 parts.

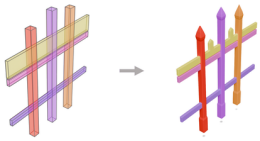




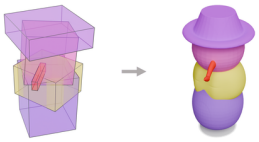
Snowman with spherical stacked spheres



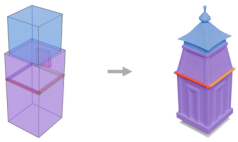
Smooth ship with angular fins and missile



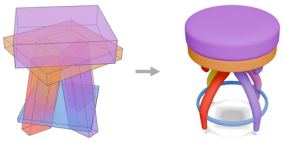
Matte fence section



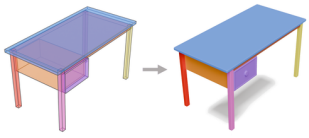
Snowman with curved head and conical base



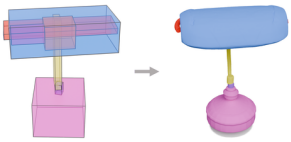
Tiered pagoda with square base and conical roof



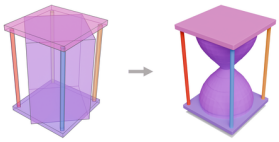
Padded stool with bent legs



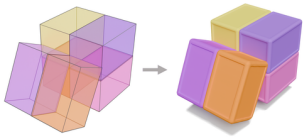
Streamlined table with flat top and four straight legs



Brass desk lamp with round base



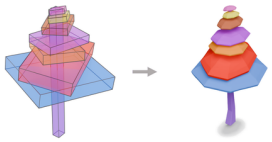
Frosted hourglass with boxy frame



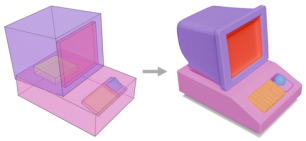
Stacked cargo crates



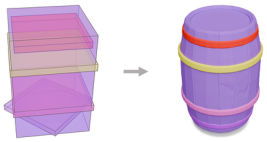
Bowed claw blade



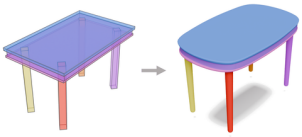
Low-poly tree with conical stacked layers



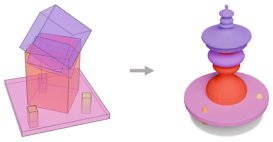
Vintage computer console with rounded casing and slatted vents



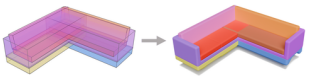
Wooden barrel with planar top and ring bands



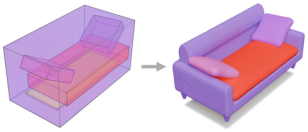
Elongated oval dining table with four turned legs



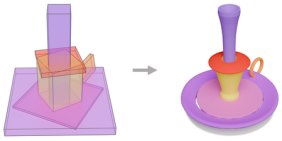
Ornate incense burner



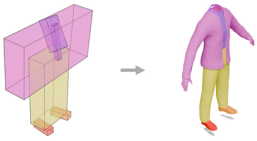
L-shaped sectional sofa with tapered legs



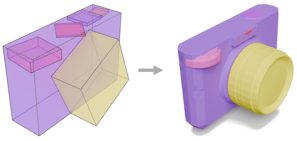
Plush sofa with narrowed legs



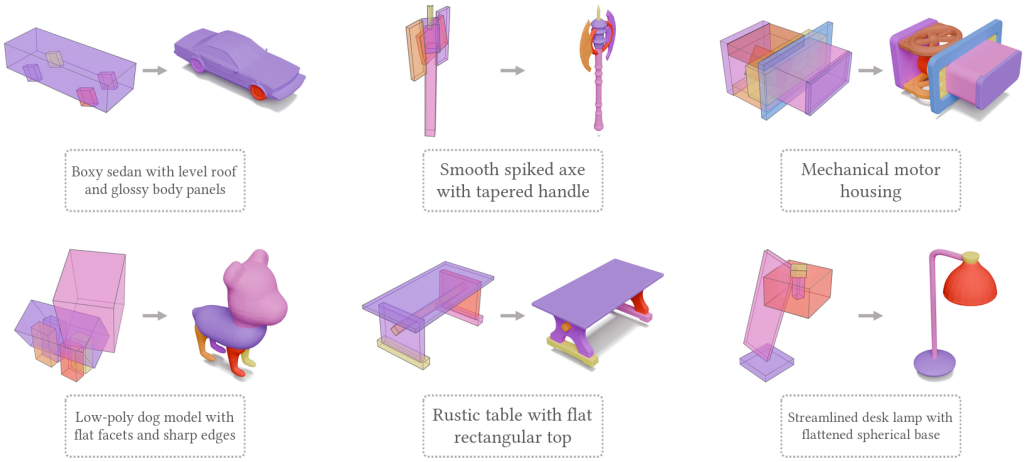
Matte brass candlestick with ring handle



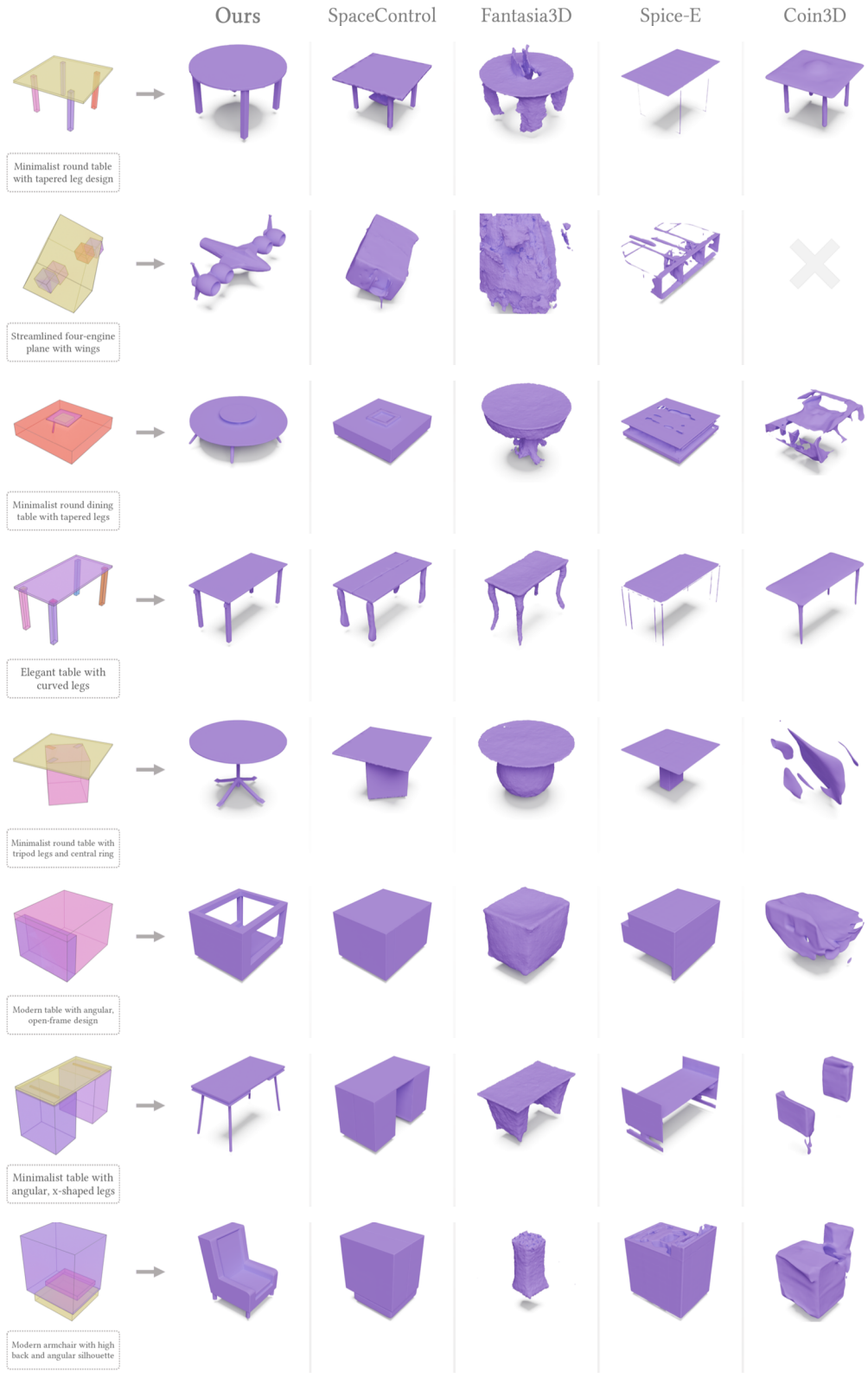
Headless suit with open collar and straight arms

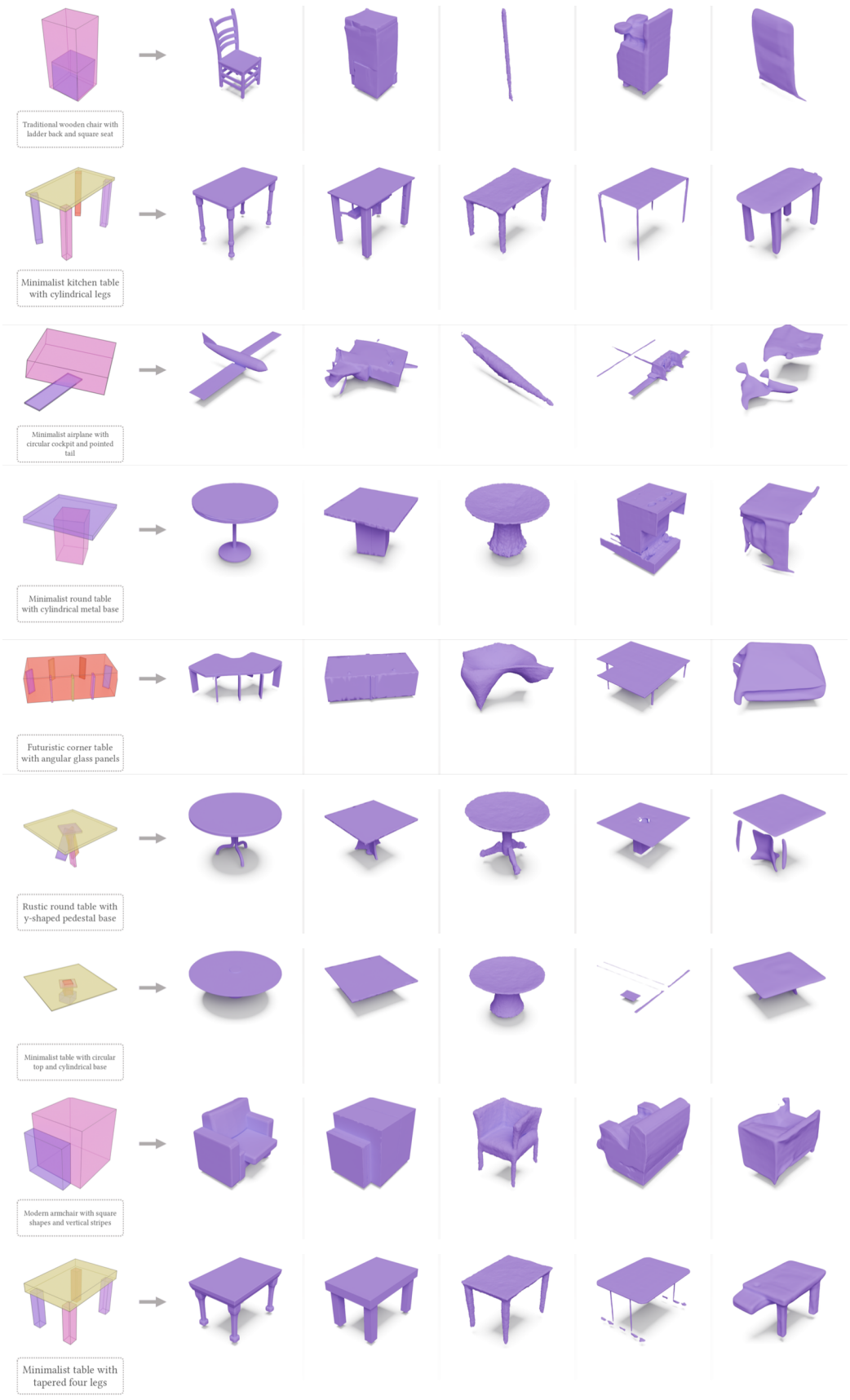


Vintage camera with block-shaped body and barrel-like lens

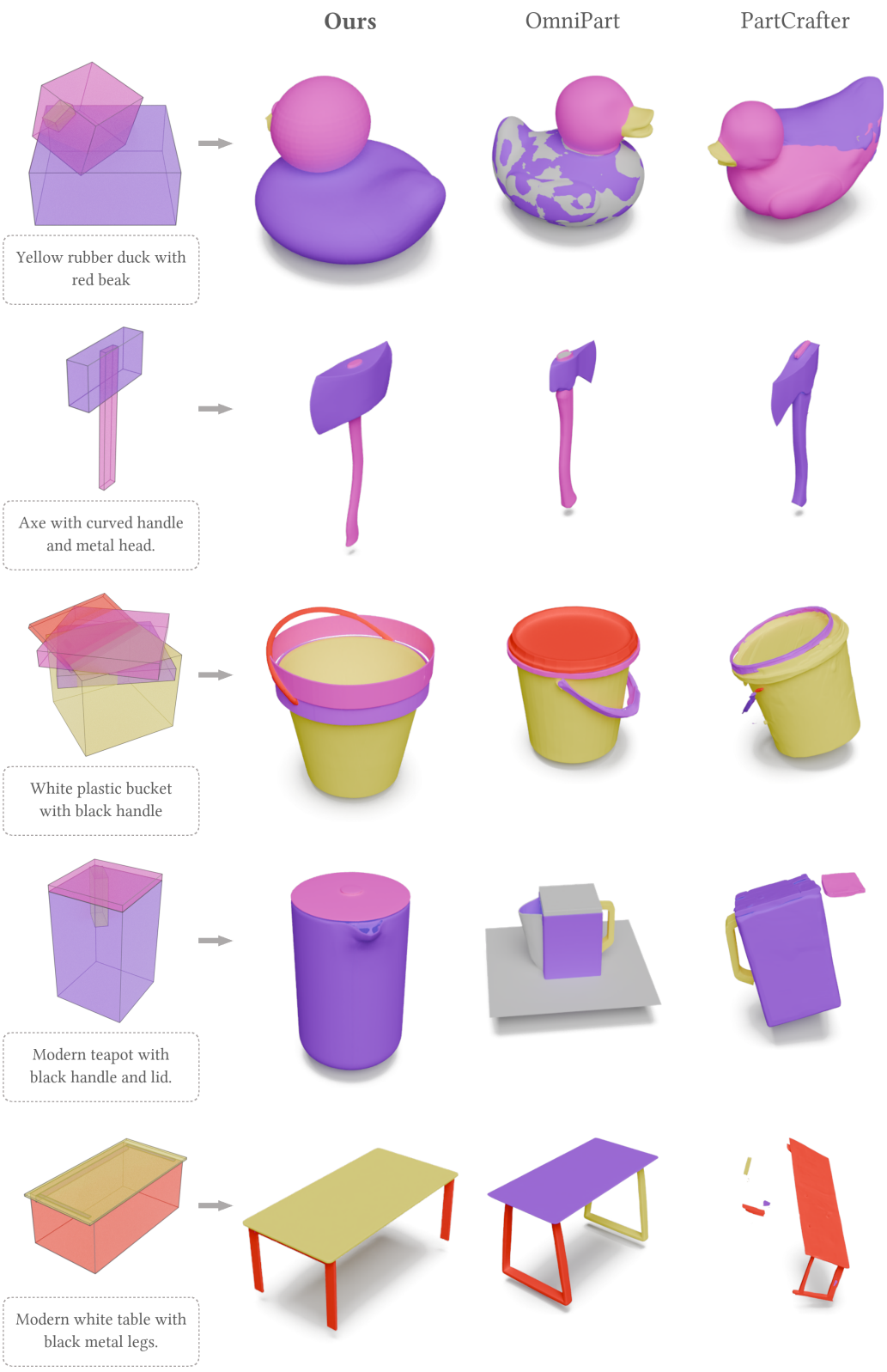

Boxy sedan with level roof and glossy body panels
Smooth spiked axe with tapered handle
Mechanical motor housing
Low-poly dog model with flat facets and sharp edges
Rustic table with flat rectangular top
Streamlined desk lamp with flattened spherical base

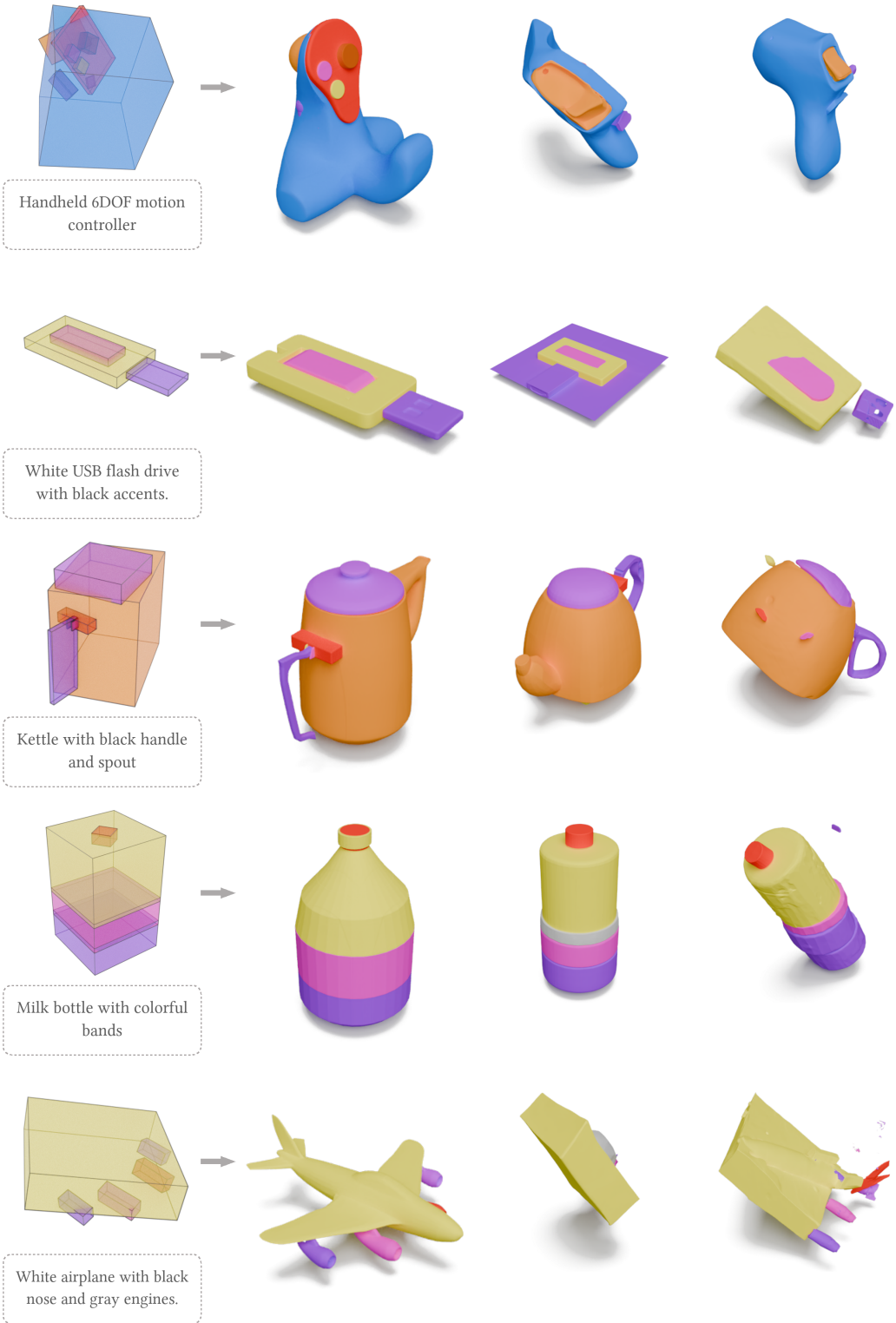
F.3 Comparison with Controllable Baselines




Traditional wooden chair with ladder back and square seat
Minimalist kitchen table with cylindrical legs
Minimalist airplane with circular cockpit and pointed tail
Minimalist round table with cylindrical metal base
Futuristic corner table with angular glass panels
Rustic round table with y-shaped pedestal base
Minimalist table with circular top and cylindrical base
Modern armchair with square shapes and vertical stripes
Minimalist table with tapered four legs

F.4 Comparison with Part Synthesis Baselines




Handheld 6DOF motion controller
White USB flash drive with black accents.
Kettle with black handle and spout
Milk bottle with colorful bands
White airplane with black nose and gray engines.



White chess pawn, 3D model, isolated

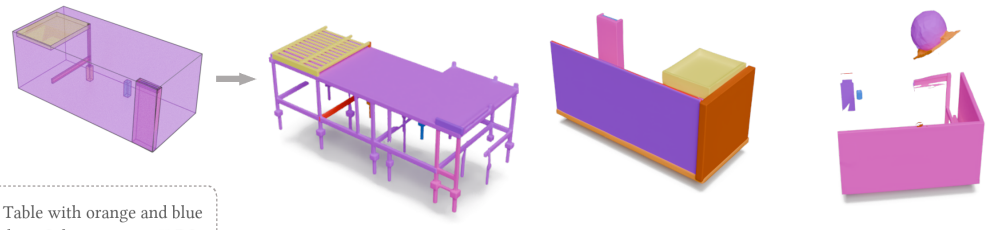
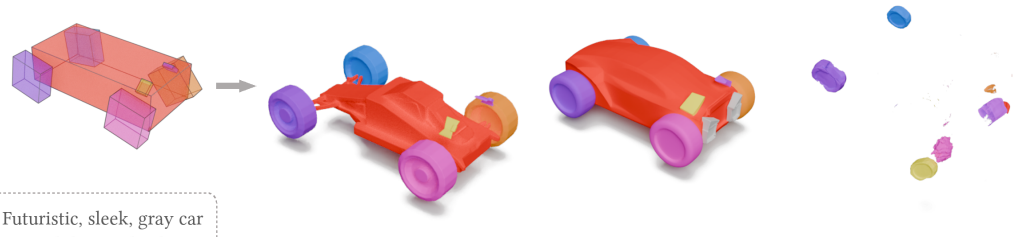
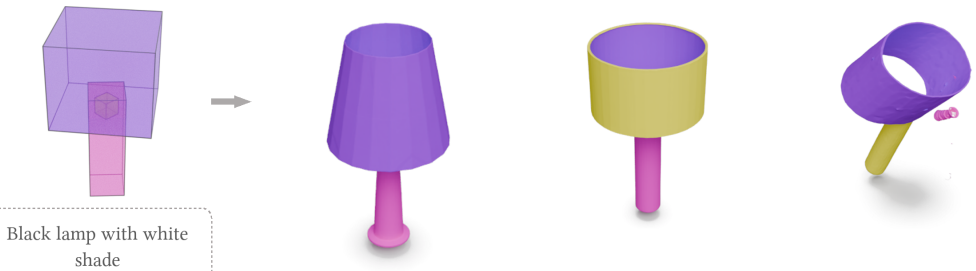


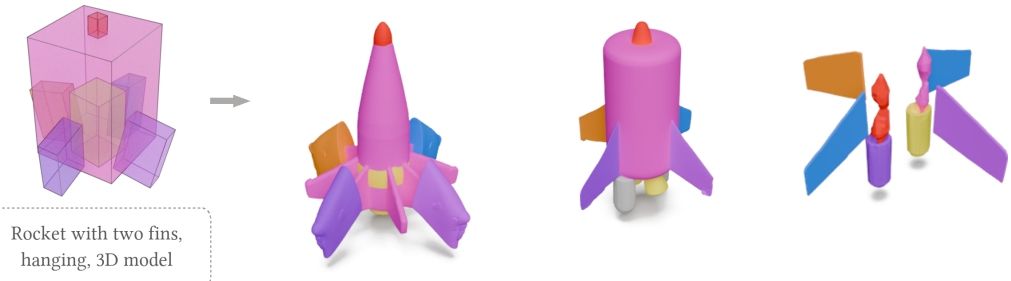
Table with orange and blue legs, Subestação DNIT-RO



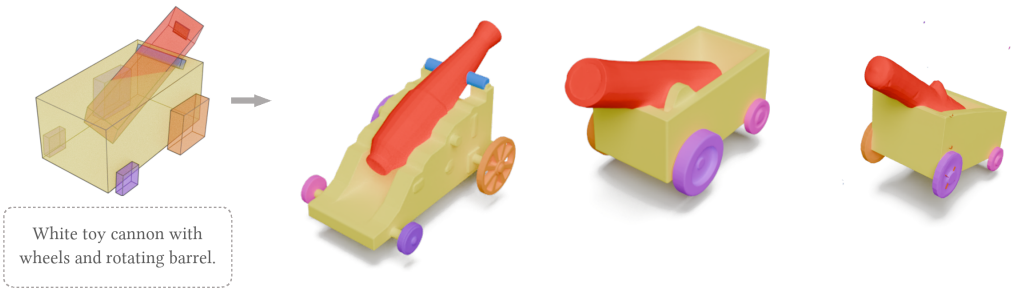
Futuristic, sleek, gray car with large wheels.



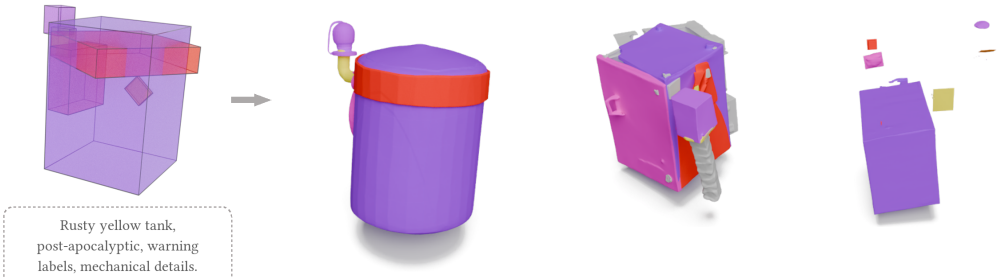
Black lamp with white shade



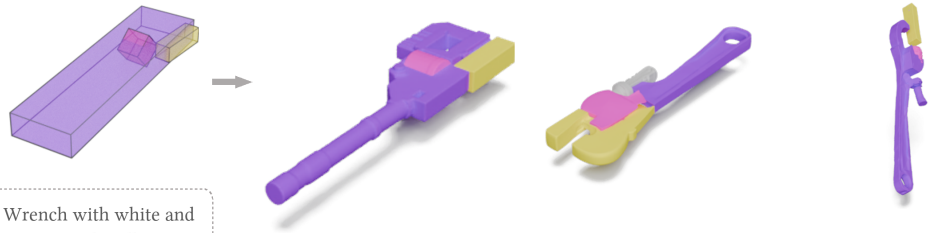
Rocket with two fins, hanging, 3D model



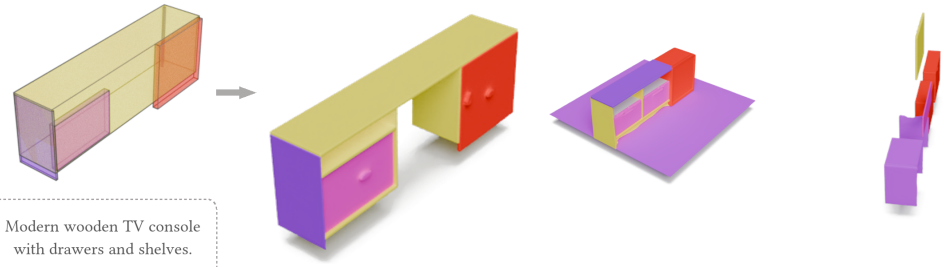
White toy cannon with wheels and rotating barrel.



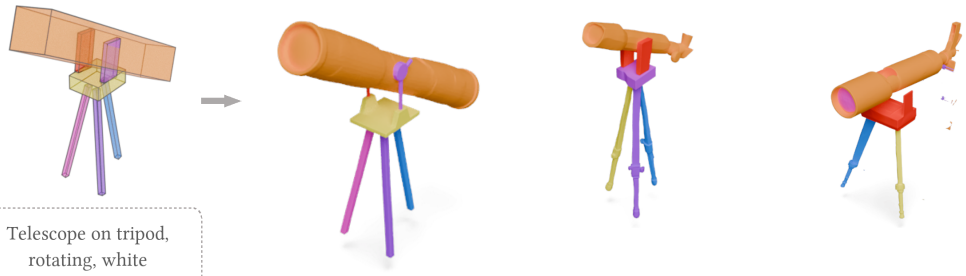
Rusty yellow tank, post-apocalyptic, warning labels, mechanical details.



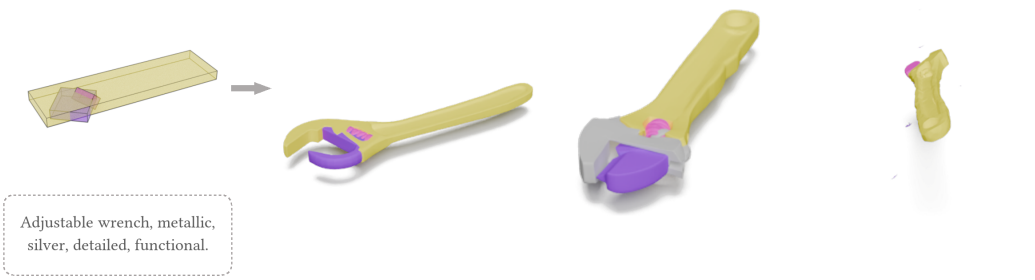
Wrench with white and gray handle



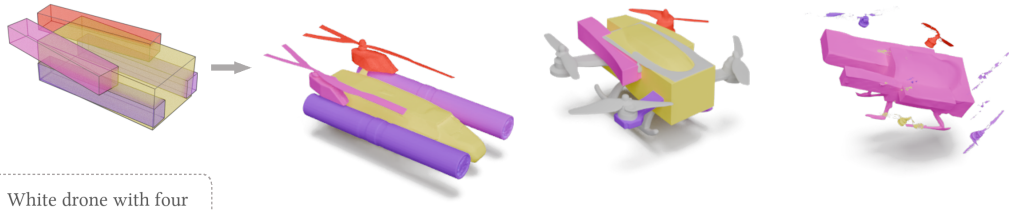
Modern wooden TV console with drawers and shelves.



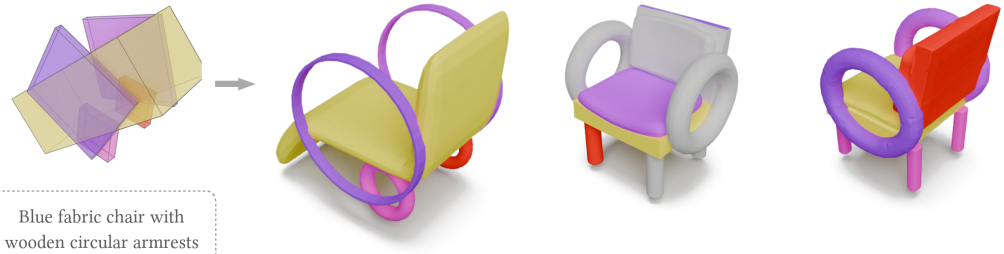
Telescope on tripod, rotating, white



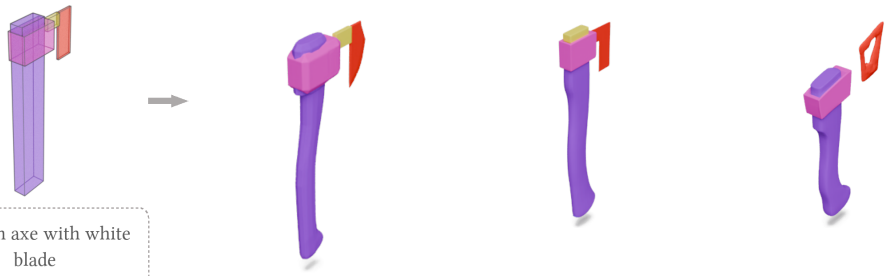
Adjustable wrench, metallic, silver, detailed, functional.



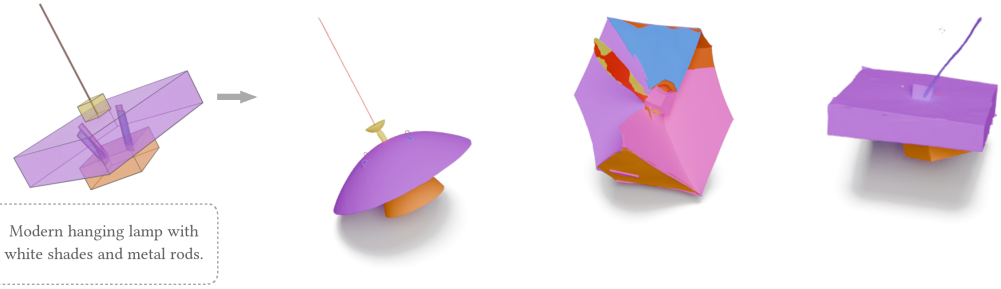
White drone with four rotors and landing gear.



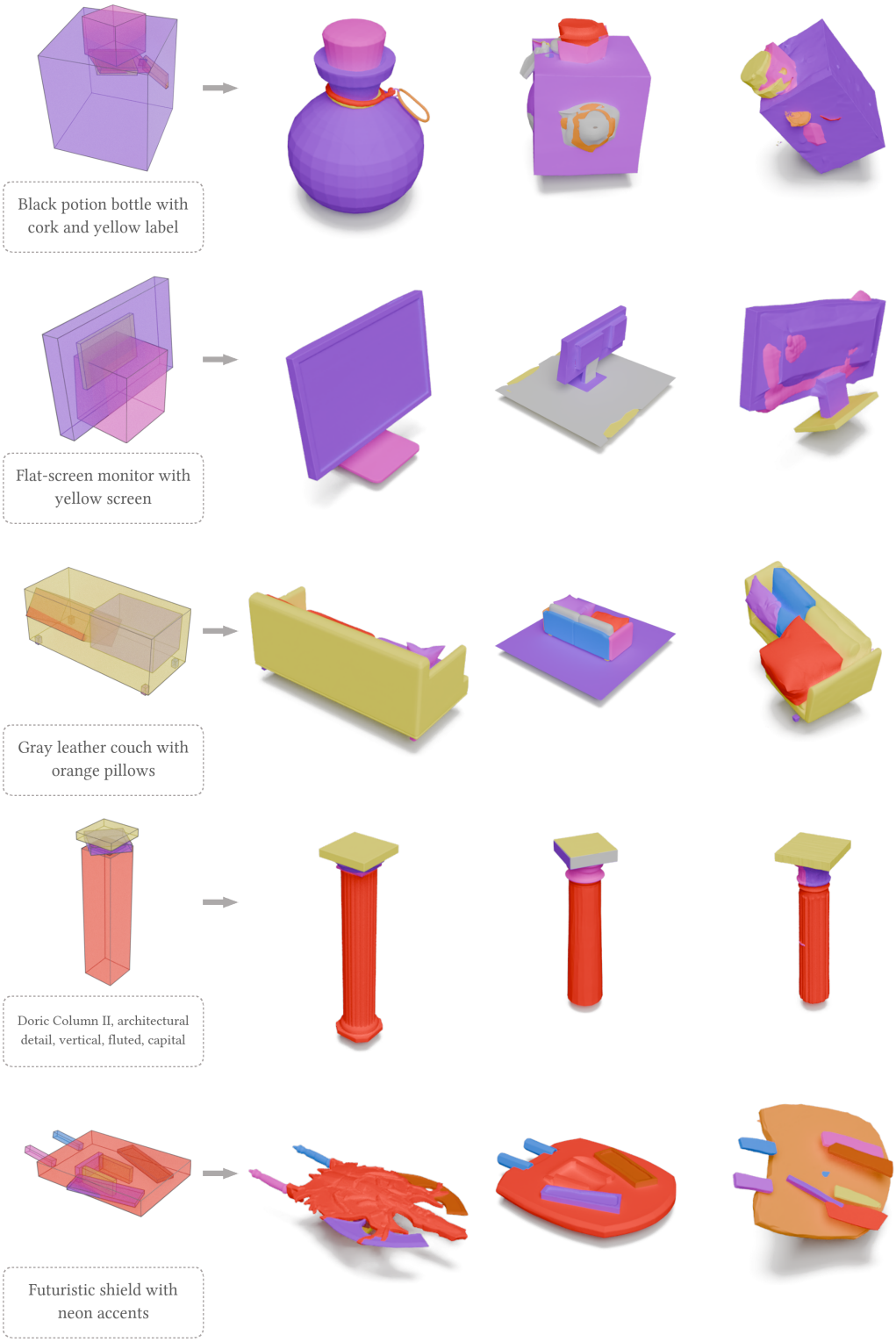
Blue fabric chair with wooden circular armrests

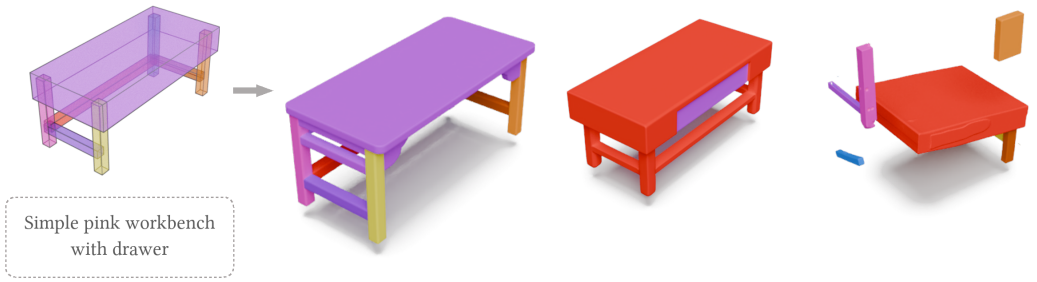


Wooden axe with white blade

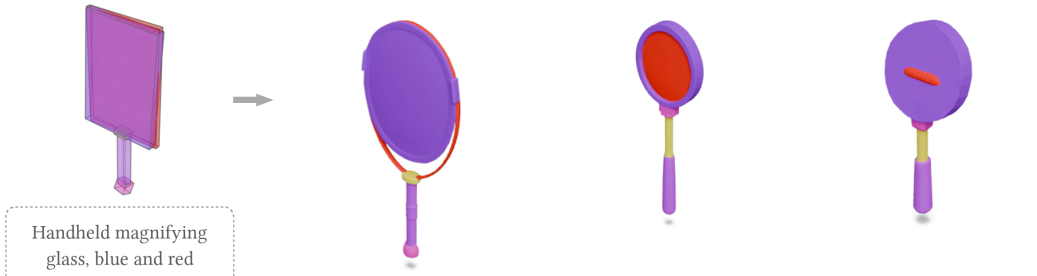


Modern hanging lamp with white shades and metal rods.

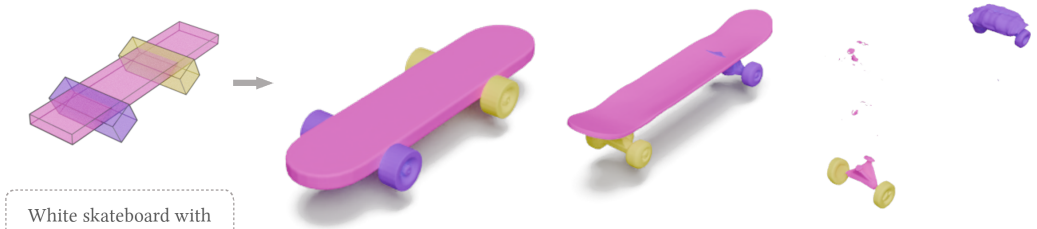

Black potion bottle with cork and yellow label
Flat-screen monitor with yellow screen
Gray leather couch with orange pillows
Doric Column II, architectural detail, vertical, fluted, capital
Futuristic shield with neon accents



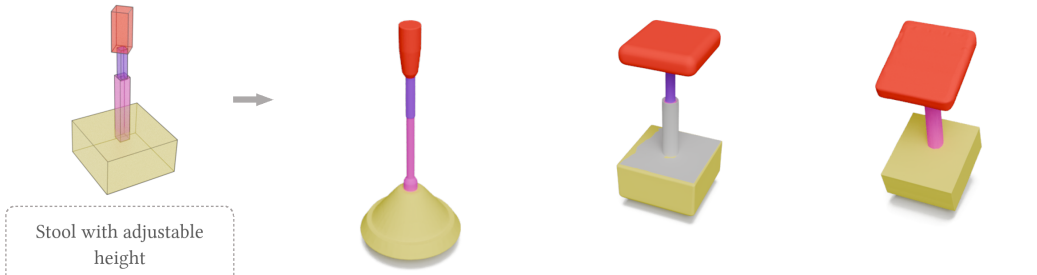
Simple pink workbench with drawer



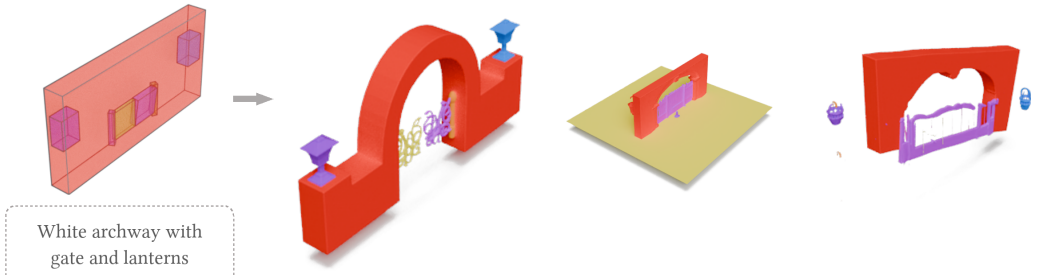
Handheld magnifying glass, blue and red



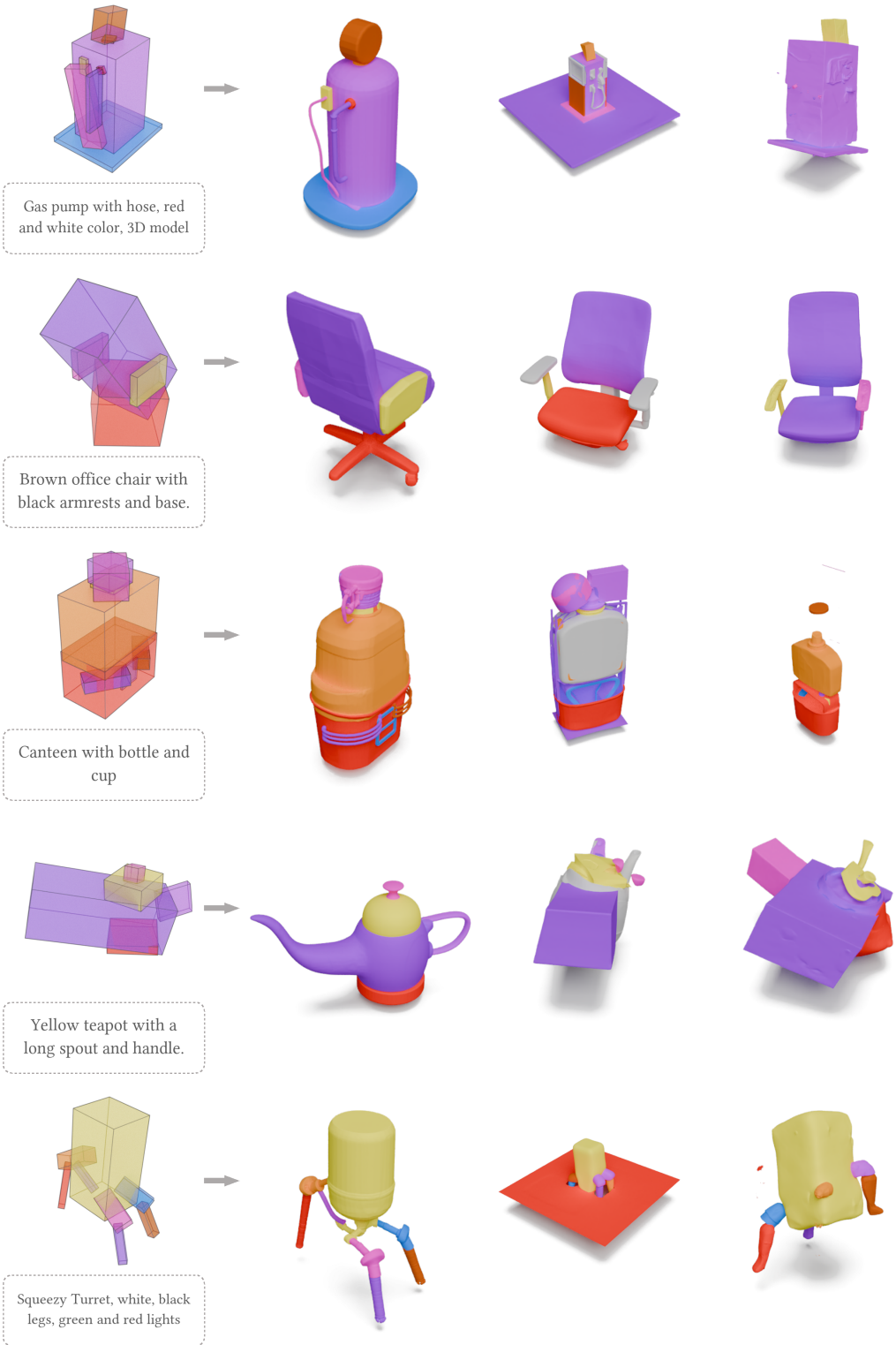
White skateboard with black wheels.

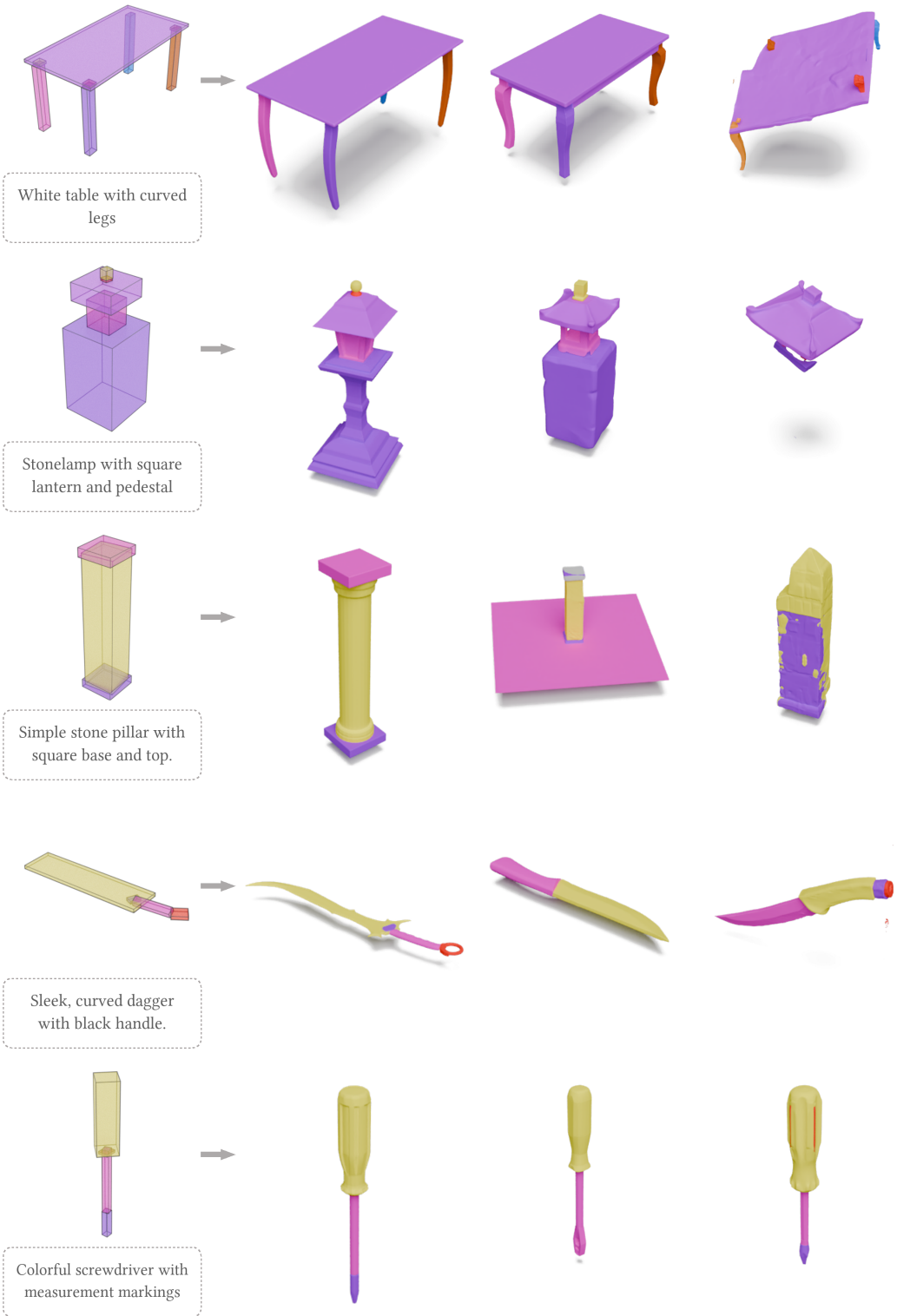


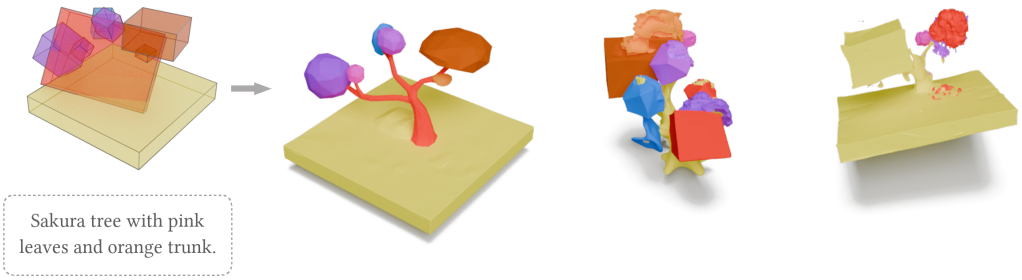
Stool with adjustable height



White archway with gate and lanterns


Gas pump with hose, red and white color, 3D model
Brown office chair with black armrests and base.
Canteen with bottle and cup
Yellow teapot with a long spout and handle.
Squeezy Turret, white, black legs, green and red lights

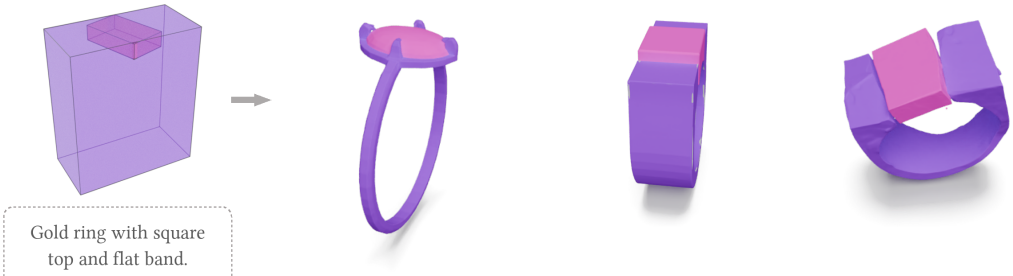

White table with curved legs
Stonelamp with square lantern and pedestal
Simple stone pillar with square base and top.
Sleek, curved dagger with black handle.
Colorful screwdriver with measurement markings



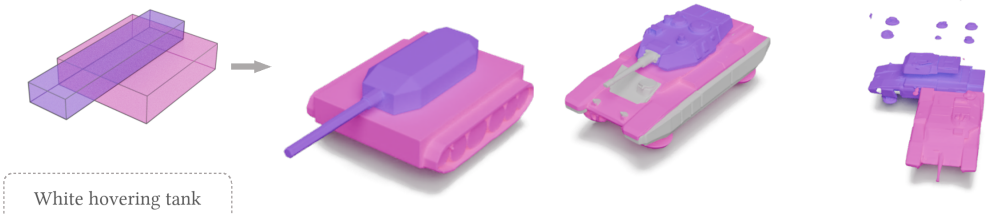
Sakura tree with pink leaves and orange trunk.



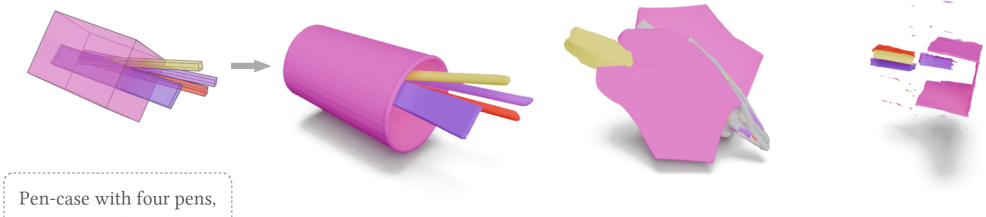
Potion bottle with twisted cap and band.



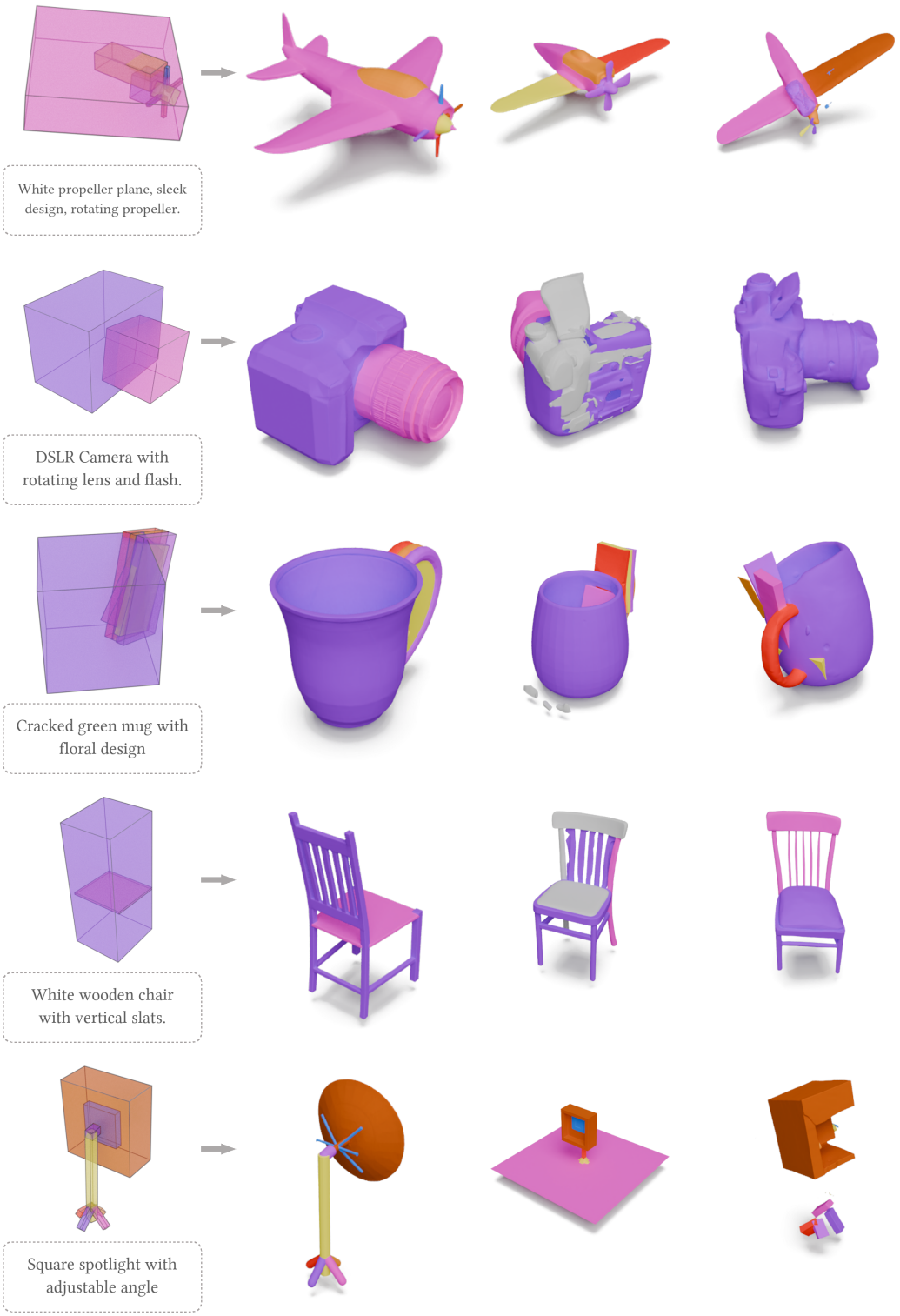
Gold ring with square top and flat band.

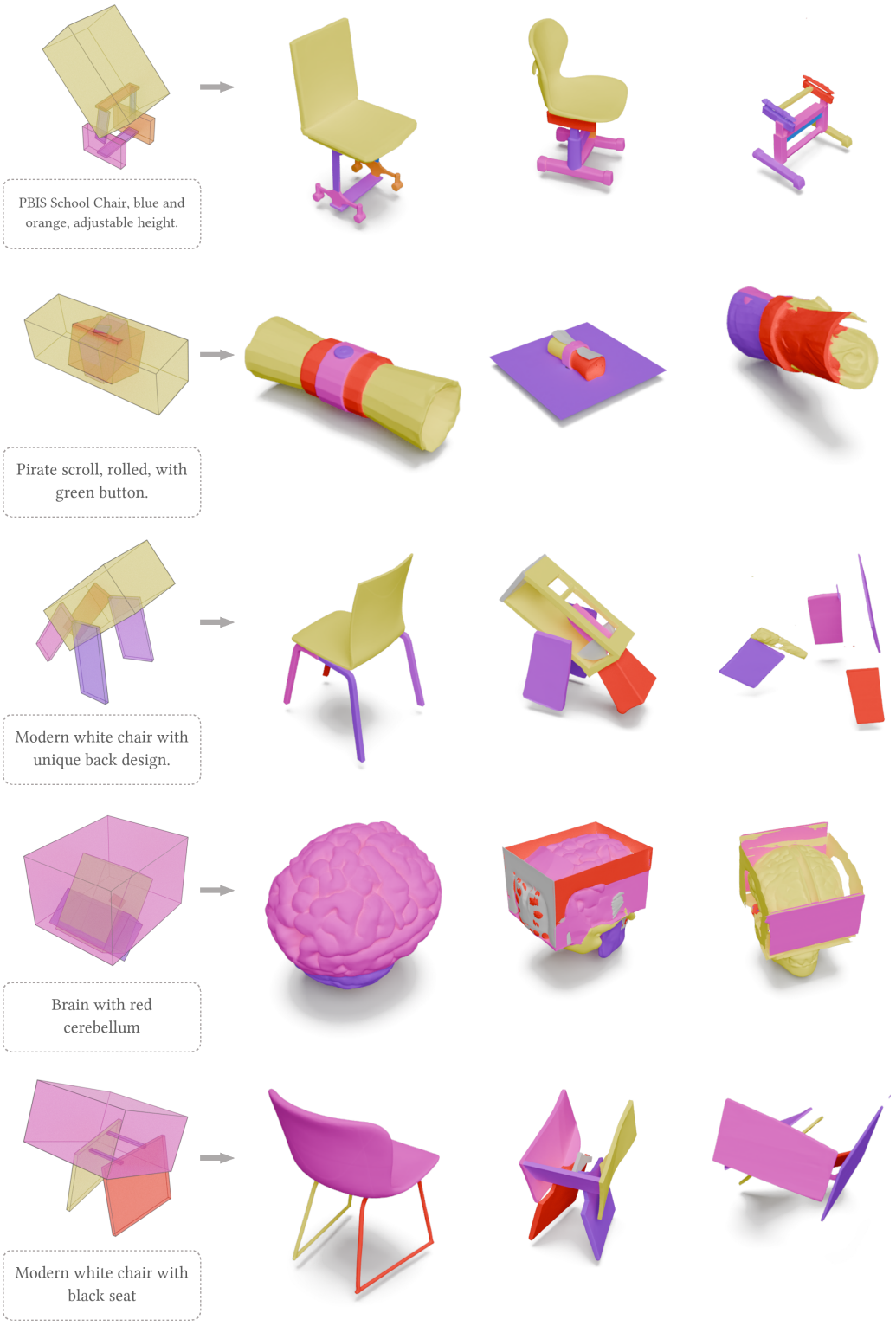


White hovering tank with rotating turret.



Pen-case with four pens, rotating and retracting.


White propeller plane, sleek design, rotating propeller.
DSLR Camera with rotating lens and flash.
Cracked green mug with floral design
White wooden chair with vertical slats.
Square spotlight with adjustable angle



G FAILURE CASES

While our method achieves strong results across a wide range of inputs, we observe a few characteristic failure modes, illustrated in fig. 10. The most common artifacts are mild z-fighting along the boundaries between adjacent parts, which arises when neighboring part surfaces are generated

in close spatial proximity without explicit awareness of one another. We also occasionally observe awkward part geometries in cases with complex, highly overlapping layouts, where the model must reconcile a large number of competing spatial constraints within a small region. Importantly, these issues are localized and do not compromise the overall structural coherence or semantic alignment of the generated shapes, and we expect them to be straightforward to address in future work through improved cross-part consistency mechanisms or light post-processing.

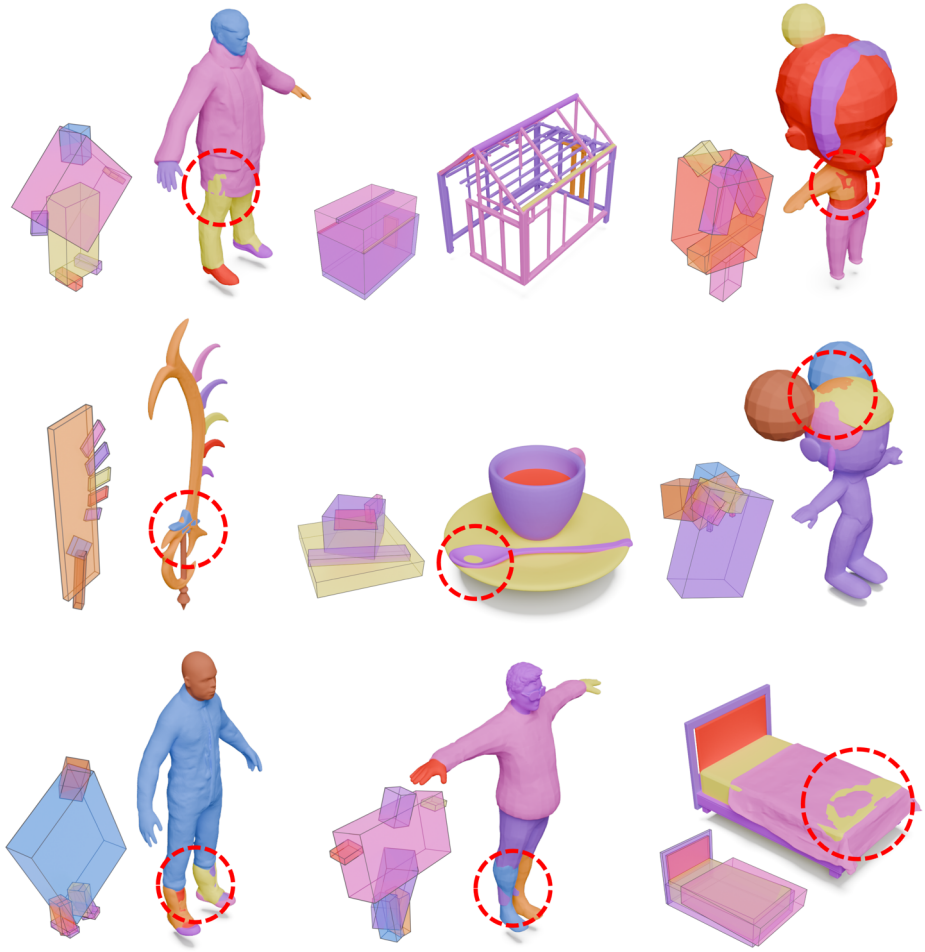


Figure 10: **Failure Case Examples.** We highlight a few failure cases of our method, which mostly include z-fighting artifacts between parts, and awkward part geometries especially prevalent in complex, highly overlapping layouts.

H Implementation Details

H.1 Architecture Details

We provide a complete architecture diagram of our CompoSE model in fig. 11, illustrating the flow from the part-control and layout encoders, through the cross-attention fusion blocks, to the decoder that produces the final output.

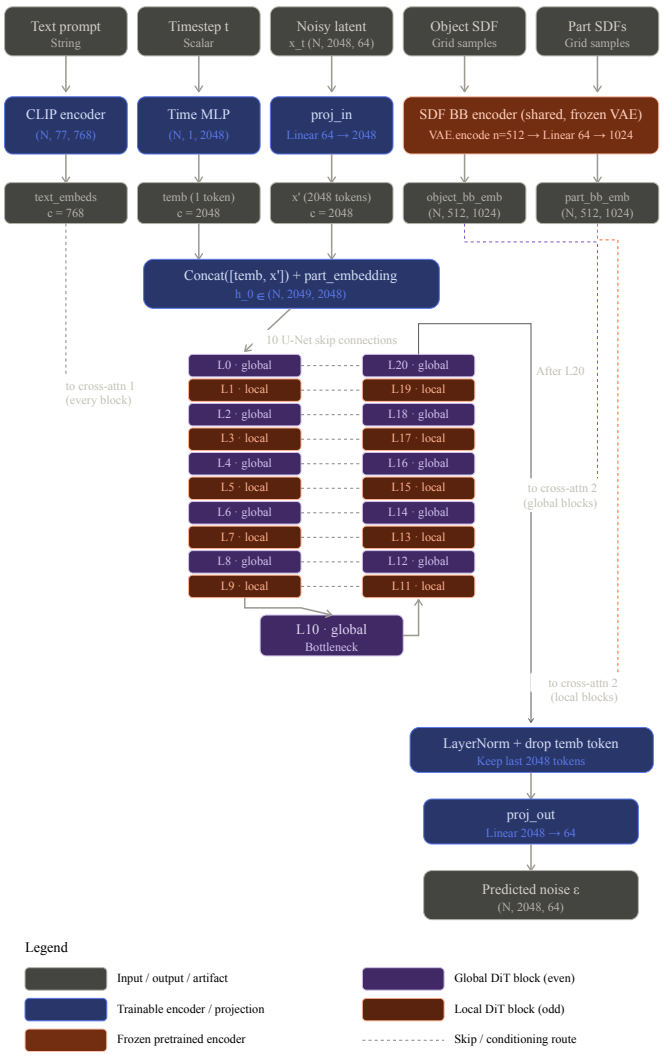


Figure 11: **Complete Architecture Diagram for the CompoSE Transformer.** We provide a complete architecture diagram for our CompoSE model, showing the part-control encoder, the layout encoder, the cross-attention fusion blocks, and the decoder.

H.2 CFG Control Annealing

We employ classifier-free guidance (CFG) [39] for improved sampling quality, but we find that combining it with layout conditioning requires special care: in the dual cross-attention mechanism of our layout control DiT blocks, setting the layout control strength near zero for negative samples causes excessive bounding-box conditioning, whereas a strength of one degrades output quality. Our solution is to anneal the bounding-box conditioning strength with an exponential decay across layers and throughout the denoising process, for negative CFG samples only, which substantially improves visual quality; we ablate this choice in table 5. We detail the resulting schedule below.

CFG. We use classifier-free guidance with a negative prompt to enhance the diversity and quality of the generated parts; the guidance scale and negative prompt are given in appendix H.4.

Control Annealing. Our proposed CFG Control Annealing (CFGCA) is applied as a layer-wise multiplicative decay of the layout control strength α_c during the denoising process, only for negative CFG samples. For positive samples, using $\alpha_c = 0$ leads to poor layout adherence. For negative sam-

ples, when using $\alpha_c = 0$ throughout the denoising process, synthesized shapes tend to be square-like or boxy. We illustrate this effect in fig. 12. This is because negative CFG samples, being unconditioned on layout, are spatially unrestricted. Classifier-free guidance then overcorrects by forcing the generated shape to conform tightly to the layout boundaries, resulting in box-like geometries.

Instead, by gradually reducing the layout control strength for negative samples, we push our model to strongly adhere to the layout in the later denoising steps, while allowing more freedom in the earlier steps for enhanced diversity and shape quality.

Qualitative ablation. We provide additional qualitative results demonstrating the effect of CFG Control Annealing (CFGCA) during inference. We sample shapes starting from the same layout, text prompt and initial noise latent, and show multiple generations with and without control annealing. The results are shown in fig. 13. Overall, we observe that CFGCA helps produce shapes of higher quality without geometric artifacts, while still adhering well to the input layout. These observations confirm the quantitative results reported in table 5.

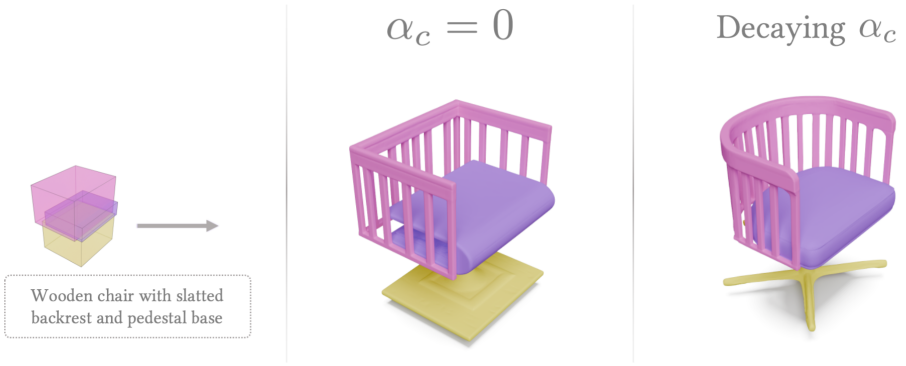


Figure 12: **Removing Control Guidance for Negative Samples.** We illustrate the effect of completely removing layout control for the negative CFG samples during inference (that is, setting $\alpha_c = 0$ for negative samples throughout the denoising process). Without any layout control on negative samples (left), the synthesized shapes tend to exhibit square-like geometries.

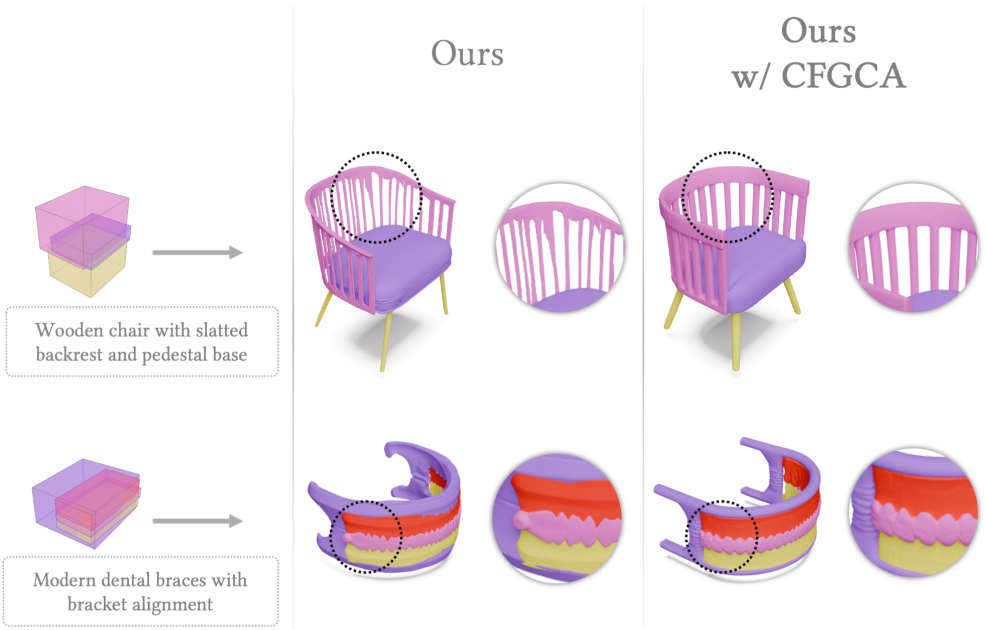


Figure 13: **Effects of CFG Control Annealing.** We illustrate the effect of annealing the layout control strength during inference for the negative CFG samples. Without annealing, synthesized samples can exhibit undesirable artifacts (left). With our proposed CFG Control Annealing (right), the generated shapes display higher quality geometry more consistently. This is especially noticeable in samples containing regular geometric patterns (e.g., backrest slats, dental braces).

H.3 Layout Optimization and Artifact Filtering

Because our method operates in the latent space of a VAE, conversions to and from 3D are subject to reconstruction errors. Since we encode our geometric controls through that same VAE, these errors also distort the conditioning signal and can leave synthesized parts slightly misaligned with the input layout. Fortunately, we know the desired 3D location of each part at inference time, which lets us correct both this misalignment and the artifacts that arise from encoding parts in isolation. We detail the two procedures below.

Misalignment. Due to our synthesis method operating in latent space, and the fact that our layout controls are also encoded into latent features, encoding/decoding inaccuracies can lead to misalignments between the generated shapes and the input part controls. To mitigate these effects, we implement a simple beam-search-based layout optimization procedure during inference. We optimize a shape-preserving transformation (similarity transformation) of the input layout to maximize the part-IoU between the generated parts and the transformed layout. Note that this optimization is performed without any alteration of the synthesized geometry.

Artifacts. Furthermore, some generated parts may exhibit artifacts such as floating geometry or disconnected fragments. This is mainly due to the distribution shift between the training data of the VAE, which is mainly composed of normalized and centered holistic shapes, and our fine-tuning data, which consists of unnormalized part segments. Previous works [59, 90] have also observed similar artifacts when using pre-trained VAEs for part-based reconstruction, and proposed to completely retrain a VAE on a large dataset of part segments to mitigate this issue. However, this approach is computationally prohibitive and requires access to a large dataset of parts. In our case, having access to ground-truth part layouts and focusing on part instances allows us to implement a simple artifact filtering strategy based on part-IoU scores. Specifically, since our model generates part instances as single connected components tightly fitting within the input bounding box, mesh

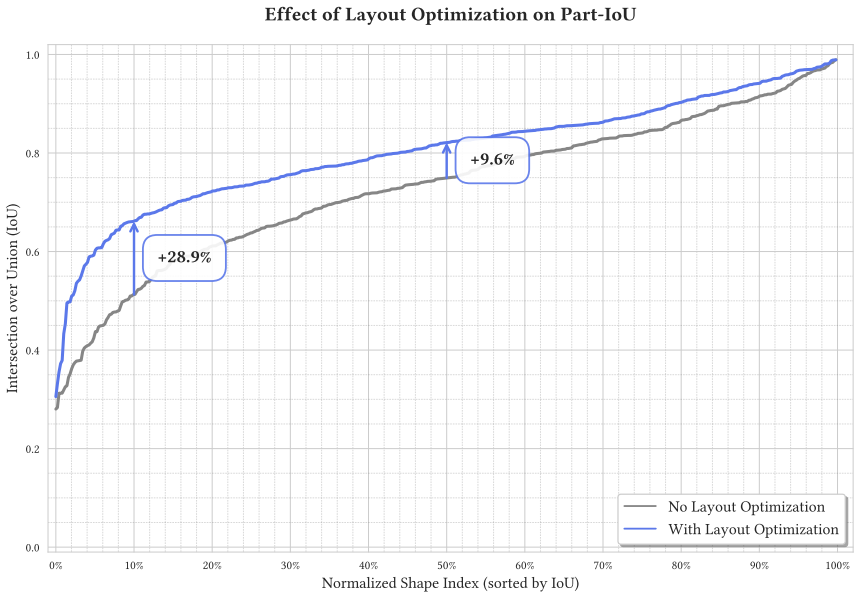


Figure 14: **Effects of Layout Optimization on Part-IoU.** We plot the effects of our layout optimization and artifact filtering, described in this section. We sort shapes by increasing average part-IoU (left to right). We plot the average part-IoU before (gray) and after (blue) layout optimization. For the shapes in the bottom 10% of most misaligned shapes, we gain an improvement of around 28% in part-IoU after optimization. These gains decrease to around 9% for the top 50% best-aligned shapes, showing that our optimization is mostly beneficial for poorly aligned shapes.

components appearing outside their corresponding control boxes are highly likely to be artifacts.

Using this fact, we:

- decompose each synthesized part mesh into its connected components, and only consider parts with more than one component as candidates for artifact removal.
- identify the largest connected component of each part, and compute its IoU with the input bounding box.
- discard all other components and retain only the largest one, if its IoU is above a fixed threshold.

This simple filtering strategy effectively removes floating artifacts in almost all cases, without requiring retraining of the VAE. We illustrate qualitative results of our layout optimization and artifact filtering in fig. 15.

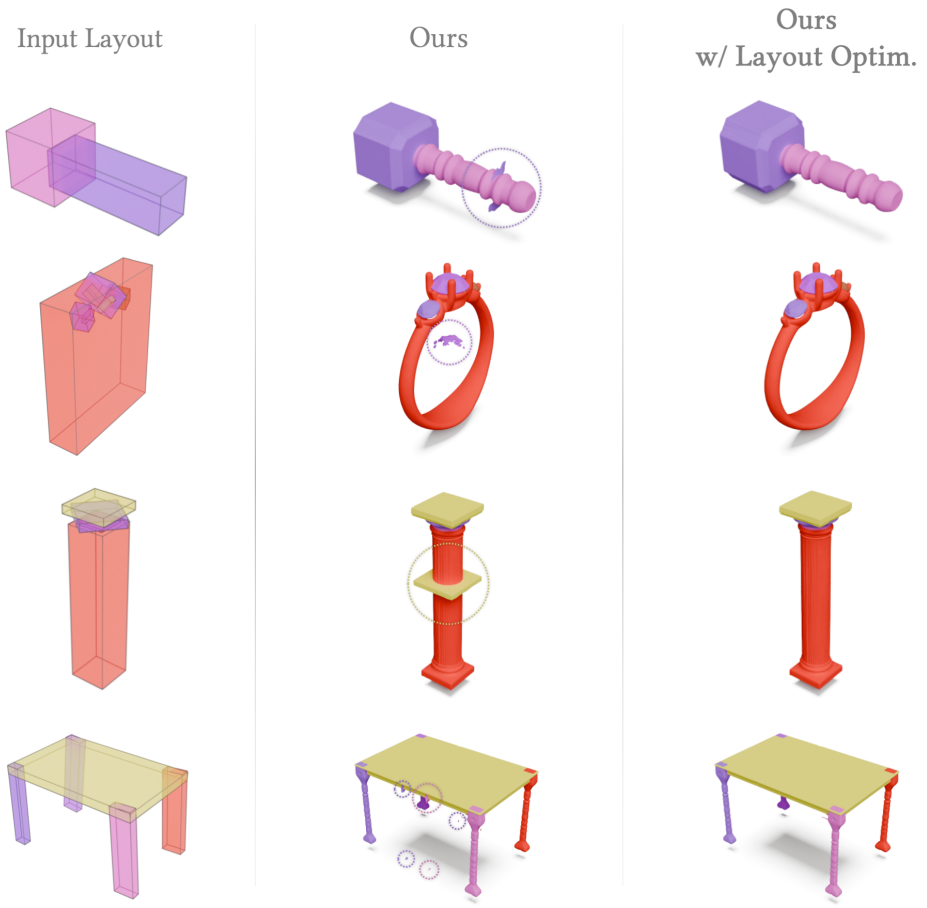


Figure 15: **Effects of Layout Optimization on Part-IoU.** We sample from the shapes most misaligned before optimization to illustrate the effects of our method. Before filtering, some parts exhibit floating geometry or disconnected fragments (middle, circled). These artifacts are especially prevalent in parts with flat or thin geometries (e.g., tabletops, planes). After optimization and filtering, the parts better fit their input bounding boxes, and artifacts are removed consistently (right). Importantly, this process does not affect the initial geometry.

H.4 Training Details

First stage. We train our model end-to-end with a standard rectified flow loss [61]. During training, we use a mix of simple objects (consisting of few parts) and complex ones (consisting of many parts) to improve the generalization ability of our method. Our base model is trained on 68k monolithic shapes and 23k multi-part samples from Objaverse [19], for a total of **92k** objects and **170k** unique parts. Importantly, we do not use additional supervision such as part-level text prompts during training. We train this base model for 90k steps with a batch size of 128 distributed over 8×A100 GPUs, which takes approximately two weeks. We use the AdamW [63] optimizer with a learning rate of 5×10^{-5}, a weight decay of 0.01, and $\epsilon = 10^{-8}$. We clone the weights of a pre-trained TripoSG DiT [57] model for initialization, and modify the DiT blocks to allow for layout conditioning as described in the main paper. Training batches are composed of shape instances sampled randomly from our dataset, in a way that ensures that the total number of parts per batch is equal to our selected batch size. We keep a fixed 15% of training shapes to be monolithic shapes (i.e., with a single part) to minimize forgetting of the base capabilities. We train using $D = 512$ tokens to represent each part latent. We further fine-tune the model with $D = 2048$ tokens in a second stage, described below.

Second stage. For the higher-token-resolution variant with $D = 2048$ tokens, we fine-tune the model for an additional 100k steps on an expanded dataset mix combining Objaverse [19], 3DCoMPaT [58, 88], and HY3DBench [43] samples, all pre-processed with the pipeline described in fig. 16, for a total of **219k** unique shapes and **1.39M** parts. For 3DCoMPaT and HY3DBench, we use the labeled part segments as initial proposals for our merging stage as we find that they are of sufficiently high quality. Finally, we select a further **43k** high-quality multi-part samples from the resulting dataset mix to fine-tune the model for an additional 2k steps, using heuristics based on inter-part IoU. This second training stage is performed on 8×H100 GPUs.

Third stage. To support layouts of up to 16 parts per shape, we further fine-tune the $D = 2048$ model for an additional 50k steps, following the same procedure and dataset mix as in the second stage, but raising the maximum number of parts per training sample to 16.

Inference. We use classifier-free diffusion guidance (CFG [39]) during inference, with a CFG scale of 6.5 for all of our experiments, as we find that this value provides a good balance between text and layout adherence and shape quality. We employ the negative prompt "Low-poly, minimal, blocky" to discourage overly simplistic shapes. Additionally, we use Temporal Score Rescaling [107] with a scale of $k = 0.98$ to further enhance shape detail. For all of our experiments, we use 50 denoising steps with the sampler introduced in Stable Diffusion 3 [27]. Mesh extraction is performed using Marching Cubes [62]. We perform inference with CFG Control Annealing (CFGCA) as described in appendix H.2, with a layer-wise decay factor of $\beta = 0.99$.

I Dataset

Our main first-stage dataset is derived from the Objaverse [19] collection through a fully automated pipeline designed to produce part-segmented shapes and their corresponding text-layout pairs without any manual effort.

I.1 Dataset Creation

There is no off-the-shelf dataset consisting of part-segmented shapes, bounding box layouts, and corresponding text captions that is suitable for our purposes, so we create one by applying a novel fully automated data processing pipeline to Objaverse [19]-sourced samples. In our second training stage, we also process a mix of pre-segmented samples from various sources [58, 88, 43] using the same pipeline to increase the diversity of our training data. We illustrate our processing pipeline in fig. 16. Our pipeline consists of the following stages:

Initial Segmentation. We begin by decomposing each raw 3D mesh into an initial set of low-level segments, which we will merge into semantically meaningful part instances in our next stage. We use the geometrically disconnected components of the mesh as our initial set of segments. We note that the Objaverse GLTF files already contain an artist-defined hierarchy of segments, so we could use these segments as our initial set of segments to merge. However, we found that the artist-defined segments in Objaverse can be spatially incoherent within an object, and inconsistent across objects, so we choose to discard them, a step we refer to as *GLTF flattening*. We then center each mesh at the object-space origin and scale it uniformly so that its longest dimension fits within a unit cube, in a process we refer to as *normalization*. Afterwards, we apply an auto-merging step to fuse segments with planar geometries and negligible volumes with their largest intersecting segment, and we re-normalize.

Progressive Segment Merging. To obtain semantically meaningful parts from our initial segments, we use a progressive merging strategy guided by segment volume. In each iteration, we identify the smallest segments and, for each one, tag it to be merged with its largest connected neighbor. All tagged merges are then executed in a single pass (per iteration). This iterative procedure continues until the number of segments for the shape is within a target range.

Heuristic Object Filtering. To ensure the quality of our training data, we discard unsuitable objects using a series of heuristics. We filter out shapes with poor part segmentation quality based on three criteria: (1) the volume ratios between the largest part and the rest of the parts; (2)

the average intersection-over-union (IoU) between parts; and (3) the geometric integrity of each resulting part. We use empirically tuned thresholds for the first two criteria, and discard any shape in which a final part does not form a single connected surface (e.g., a part consisting of several disjoint triangle islands), to ensure clean, monolithic part instances.

Oriented Bounding Box Extraction. Once we have obtained semantically meaningful parts, we generate a spatial layout of coarse geometric control shapes that will guide our synthesis method. For each part, we compute its minimum oriented bounding box (OBB) using an efficient implementation [119] of the exact minimum bounding box algorithm described by Jylänki [48].

Text Caption Generation. The final component of our training data is the global text prompt. To generate a high-quality caption for each processed object, we render images of the complete object, and we feed these images to the powerful InternVL3-8B [16] vision-language model. We use collages of four views arranged in a 2×2 grid to provide sufficient context for caption generation. We employ a prompt that encourages captions focusing on geometry and part descriptions, avoiding color and texture which are not relevant to our synthesis task. This process provides a clean and descriptive text prompt that is grounded to the final geometry.

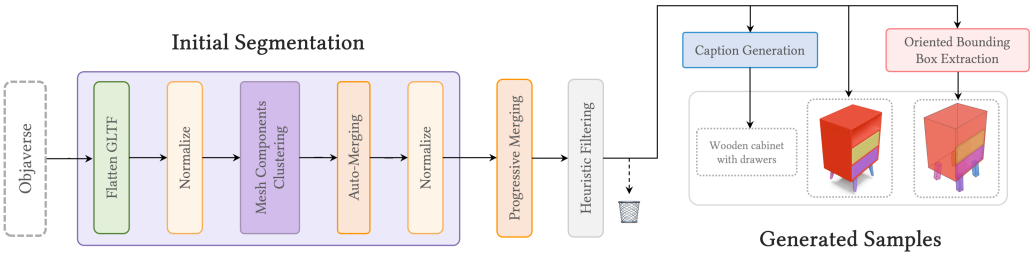


Figure 16: **Dataset Processing Pipeline.** We illustrate our fully automated dataset processing pipeline that converts raw 3D meshes from the 3D asset datasets into part-segmented shapes with corresponding text-layout pairs for training. Layouts are composed of oriented bounding boxes (OBBs) computed for each part segment. Text captions are generated using the InternVL3-8B [16] vision-language model and focus on geometric and part properties.

I.2 Statistics

We plot the distribution of mean part-IoU and largest-to-rest part volume ratio in our processed dataset in fig. 17. Alongside the distributions of mean part-IoU and largest-to-rest part volume ratio, we also plot the cutoff thresholds used for heuristic filtering during dataset processing in red. These cutoffs are selected empirically to remove badly segmented shapes, and more generally poor-quality shapes. We use a maximum mean part-IoU of 0.10 and a maximum largest-to-rest part volume ratio of 3.

Shapes with a high mean part-IoU are not beneficial to the training of our synthesis model, since their parts are highly overlapping and do not provide clear geometric cues. Similarly, heavily imbalanced part layouts are also not useful, as they lead to degenerate training samples where a single part occupies a significant portion of the output space.

In addition to this filtering, we also discard duplicated shapes, and we filter models based on their caption using a list of keywords to maximize the quality of the geometry extracted. After filtering, our final dataset contains **68k** monolithic shapes and **23k** multi-part shapes. Monolithic shapes are rebalanced during training to represent 15% of training samples.

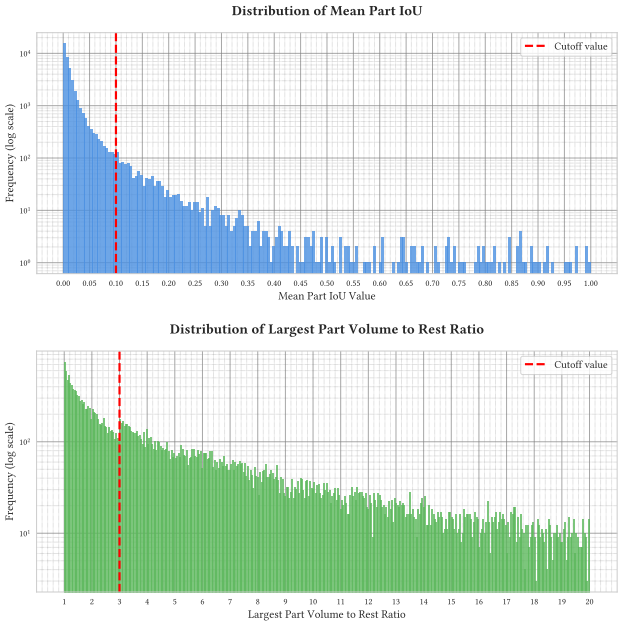


Figure 17: **Dataset Statistics.** We plot the distribution of mean part-IoU (top) and largest-to-rest part volume ratio (bottom) in our processed dataset. Frequencies are shown on a logarithmic scale to better visualize the long tails of the distributions. We also show in red our cutoff thresholds used for heuristic filtering during dataset processing.

J Additional Ablations

We report additional ablations that were omitted from the main text for space: a sweep over the classifier-free guidance scale ω used by SpaceControl [28], an evaluation of our identity-preserving part rescaling procedure against two alternatives, and a model ablation study of our design choices. We also report a comparison of inference time against the baselines.

J.1 SpaceControl Classifier-Free Guidance Scale

In table 1 of the main paper, we report SpaceControl's result at a single operating point, $\omega = 3$, which we selected as it achieves the best Voxel-IoU among all guidance scales we tested, the only metric on which SpaceControl outperforms our method. For completeness, table 3 reports the full sweep over $\omega \in \{3, 6, 7.5, 9\}$, as well as a variant augmented with image conditioning via ControlNet [117]. Increasing ω consistently improves Object-FID at the cost of Object-IoU, as stronger guidance pushes the model toward more detailed, prompt-aligned geometry that deviates from the dense superquadric volume defined by the input proxies. We find $\omega = 7.5$ offers the best trade-off between visual fidelity and layout adherence among the scales we tested, while $\omega = 3$ remains the strongest choice specifically for Voxel-IoU, motivating our choice of operating point in the main comparison.

Table 3: **SpaceControl Classifier-Free Guidance Scale Ablation.** We report SpaceControl's [28] performance across classifier-free guidance scales ω, as well as a variant augmented with image conditioning via ControlNet [117] (CN).

Variant	Object-FID↓	Object-IoU↑	Voxel-IoU↑	CLIP↑
$\omega = 3$	42.96	0.807	**0.776**	26.73
$\omega = 6$	40.20	0.794	0.761	26.83
$\omega = 7.5$	39.27	0.790	0.750	26.86
$\omega = 9$	**38.26**	0.785	0.751	**26.93**
$\omega = 7.5$ + CN [117]	39.55	0.785	0.754	26.77

J.2 Identity-Preservation under Part Rescaling

We ablate our identity-preserving rescaling procedure (appendix D), which adjusts a part's bounding box while preserving its geometric identity via KV-reinjection. We synthesize 70 edited shapes by manually collecting part rescaling operations applied on base shapes synthesized by our method, spanning multiple object categories. For each edit, we compute F-Score at $\tau \in \{0.01, 0.02, 0.05\}$ between the resized part and its counterpart in the unedited shape. We compare against two alternatives sharing the same target box: *Anisotropic Scaling*, which stretches the original geometry, and *Part Substitution*, which regenerates the part from scratch without anchoring to the original latent. As shown in table 4, our method achieves the highest F-Score at every threshold (0.38, 0.58, 0.70), substantially outperforming Anisotropic Scaling which distorts fine details, and Part Substitution which is stylistically consistent but geometrically different, in line with the qualitative results provided in fig. 5 of the main paper and in appendix F.1.

Table 4: **Identity-Preservation under Part Rescaling.** We measure how faithfully each method preserves a part's geometry after rescaling its bounding box, reporting F-Score at $\tau \in \{0.01, 0.02, 0.05\}$ (↑) against the original part.

Method	F@0.01 ↑	F@0.02 ↑	F@0.05 ↑
Anisotropic Scaling	0.23	0.38	0.55
Part Substitution	0.26	0.46	0.64
Identity-Preserving	**0.38**	**0.58**	**0.70**

J.3 Model Ablation Study

We conduct an ablation study to validate our key design choices, with results summarized in table 5.

Part Condition Encoding. We compare our SDF-based encoding against (1) a *parametric encoder* (box parameters via MLP) and (2) a *point-based encoder* (DGCNN [99] on box surface samples, fine-tuned end-to-end). Reusing the SDF-based VAE for guidance boxes ensures latent-space alignment with shape parts — a choice table 5 shows is critical for geometric fidelity and layout adherence.

Inference Strategies. CFG Control Annealing is crucial for visual quality, improving Object-FID from 19.32 to 14.44, while layout optimization — a test-time refinement step detailed in appendix H.3 — raises Part-IoU from 0.740 to 0.805 by correcting minor spatial drifts.

Token Resolution. Scaling per-part tokens from 512 to 2048 clearly improves all metrics, confirming our approach benefits from scale.

J.4 Inference Time

We evaluate inference time against the baselines on an NVIDIA H100 GPU, with results summarized in table 6. At resolution 512, our method is the fastest of all methods at $N_p = 1$ (3.7s), and even

Table 5: **Ablation study.** We ablate various components including choice of part-control encoders, CFG Control Annealing, layout optimization, and token resolution.

Layout Encoding	CFG Control Annealing (appendix H.2)	Layout Optimization (appendix H.3)	Token Resolution	Object-FID ↓	Part-FID ↓	Part-IoU ↑
Parametric			512	20.17	20.14	0.289
DGCNN [99]			512	38.58	43.10	0.008
SDF			512	19.32	21.87	0.729
SDF	✓		512	14.44	19.38	0.740
SDF	✓	✓	512	14.40	19.70	0.805
SDF	✓	✓	2048	**10.58**	**9.24**	**0.819**

at $N_p = 16$ it stays at 20.8s, far below the other proxy-guided baselines (Fantasia3D, Coin3D, and Spice-E), which need between 146.2s and 522.6s for a single part. At resolution 2048, we start at 5.5s, ahead of SpaceControl (9.1s) and PartCrafter (7.0s), but our per-part decoding makes cost grow with the number of parts, reaching 80.3s at $N_p = 16$. OmniPart is the only method whose cost is independent of N_p (4.1s throughout), while PartCrafter scales at a rate comparable to our 2048 model.

Table 6: **Inference Time.** We compare the average inference time (in seconds) of our method and the baselines with a varying number of parts N_p, measured on a single NVIDIA H100 GPU.
[†]Fantasia3D [12] being explicitly optimized for multi-GPU inference, we report its timing using 4 H100 GPUs for fairness.
Rows shown in gray (Image-Conditioned Part Synthesis) report timings that exclude the Nano Banana Pro [30] image-conditioning step required to produce their input; since our method requires no such step, this comparison is not entirely fair.

Method	**Inference Time (sec.) ↓**				
	$N_p = 1$	$N_p = 2$	$N_p = 4$	$N_p = 8$	$N_p = 16$
Proxy-Guided Synthesis					
Fantasia3D [12][†]	245.7 ± 3.0	⊗	⊗	⊗	⊗
Coin3D [23]	522.6 ± 13.8	⊗	⊗	⊗	⊗
Spice-E [84]	146.2 ± 1.7	⊗	⊗	⊗	⊗
SpaceControl [28]	9.1 ± 4.1	⊗	⊗	⊗	⊗
Image-Conditioned Part Synthesis					
PartCrafter [59]	7.0 ± 0.6	10.5 ± 0.1	27.2 ± 0.6	36.6 ± 0.2	75.1 ± 0.1
OmniPart [109]			4.1 ± 0.3		
Ours $_{512}$	3.7 ± 0.1	4.1 ± 0.1	5.8 ± 0.1	10.5 ± 0.1	20.8 ± 0.1
Ours $_{2048}$	5.5 ± 0.1	8.7 ± 0.1	16.0 ± 0.1	33.7 ± 0.1	80.3 ± 0.2