SCALING VERSATILE 3D ASSETS EDITING WITH A MILLION-SCALE DATASET

Anonymous authors
Paper under double-blind review

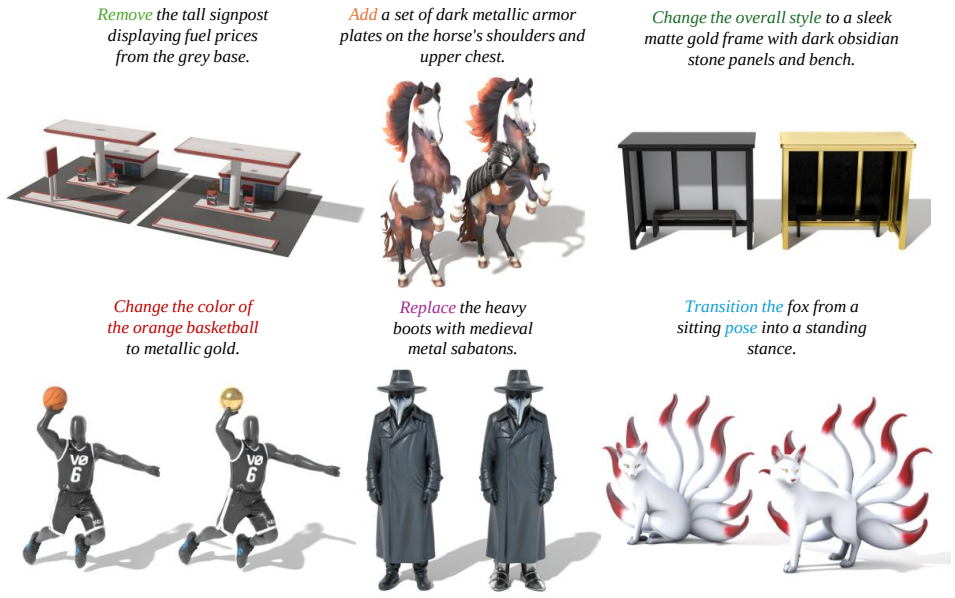

Figure 1: Given a 3D asset and an editing instruction, our method edits the asset accordingly while preserving unrelated regions.

ABSTRACT

Although recent 3D generative models produce increasingly realistic assets, controllable 3D asset editing remains challenging. Existing methods are limited by scarce training data, insufficient source-aware modeling, and a lack of practical evaluation protocols. To address these limitations, we present Alchemy3D, a unified framework for training and evaluating versatile 3D asset editors that covers data construction, model architecture, and benchmark evaluation. Specifically, we curate Alchemy3D-1M, a large-scale 3D editing dataset containing 1.25M assets and 1.38M editing pairs across seven editing types. On this data, we train a family of generative flow models for general-purpose 3D asset editing. The model family supports image- and text-conditioned editing, few-step inference, and transfer to multi-view 3D part segmentation. We further introduce GEdit3D-Bench, a large-scale, open-world benchmark with a multi-dimensional evaluation protocol. Across existing and newly introduced benchmarks, our method outperforms prior methods on most metrics of editing fidelity, source preservation, and visual quality. Our dataset, models, and benchmark will be made publicly available.

1 INTRODUCTION

With the development of diffusion and flow-matching models (Song & Ermon, 2019; Song et al., 2021; Ho et al., 2020; Lipman et al., 2023), recent 3D generative models (Xiang et al., 2025; 2026; Hunyuan3D et al., 2025) can produce high-quality 3D assets from text or image prompts. However,

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.

general-purpose 3D editing remains comparatively underdeveloped because it requires fine-grained spatial awareness. An editing model must not only execute a localized or global modification according to the instruction, but also keep irrelevant regions unchanged.

Compared to generation methods (Xiang et al., 2025; 2026; Hunyuan3D et al., 2025), existing 3D editing models are typically constrained by limitations in training data, model design, and evaluation protocols. First, achieving region-specific revisions requires paired data before and after editing, which is difficult to acquire. Existing 3D editing datasets are limited in scale, diversity, quality, and editing types (Ma et al., 2025; Xia et al., 2025; Weng et al., 2026). Second, recent 3D editors (Ma et al., 2025; Weng et al., 2026) inject the source asset through a ControlNet-style (Zhang et al., 2023) branch, which limits the interaction between source and target and may provide insufficient fine-grained guidance from the source. Finally, existing editing benchmarks are often relatively small, drawn from the same distribution as the training data, or evaluated by similarity to a single automatically generated target (Li et al., 2026b; Zhou et al., 2026b; Ye et al., 2025a; Xia et al., 2025; Weng et al., 2026), which restricts evaluation to in-distribution settings and fails to capture the one-to-many nature of generative editing.

To address these challenges, we present **Alchemy3D**, an integrated 3D editing framework comprising scalable data construction, model architecture design, and benchmark evaluation. For data construction, we select seven representative editing types (addition, removal, replacement, animation, local appearance, global appearance, and segmentation) and design type-specific pipelines to create editing pairs at scale. Candidate pairs are filtered with vision-language models (Qwen Team, 2026b; Bai et al., 2025), yielding 1.38 million editing pairs.

For the model architecture, we concatenate the tokens of the source asset and the edited target asset so that they interact through self-attention, while text or image instructions are injected through cross-attention. To improve inference efficiency, we further adopt a few-step distillation procedure based on MeanFlow (Geng et al., 2025) and DMD (Yin et al., 2024b;a). With this architecture and **Alchemy3D-1M**, we train a family of **Alchemy3D** model variants for different editing scenarios.

For evaluation, we construct **GEdit3D-Bench**, a benchmark built from assets that are independent of our training data. It combines newly synthesized 3D content with recently released real-world assets from Sketchfab[1] to mitigate in-distribution evaluation. Editing instructions are written by a vision-language model, target images are produced by an image editing model (Cao et al., 2025), and all samples are curated through automated and human filtering. Rather than measuring similarity to a single generated target, GEdit3D-Bench jointly assesses view quality, reference alignment, and Multimodal Large Language Model (MLLM)-based scores, providing a more comprehensive evaluation of 3D editors.

Our contributions are summarized as follows:

- To expand the scope of versatile 3D editing, we propose a scalable data construction pipeline spanning 7 diverse editing tasks, yielding **Alchemy3D-1M**, a dataset of 1.38 million high-quality editing pairs covering diverse assets.
- Instead of introducing source asset features through a separate branch, we fuse tokens from the source and edited target models, enabling sufficient interaction between them. We further propose a distillation pipeline that enables efficient, few-step inference for the 3D editing model. Building on this architecture and **Alchemy3D-1M**, we train a series of **Alchemy3D** model variants tailored to different editing use cases.
- To enable a more comprehensive and fair evaluation of 3D editors, we introduce **GEdit3D-Bench**, a large-scale, multi-dimensional benchmark independent of our training data.
- Extensive comparisons on multiple benchmarks confirm that our model achieves significant improvements over existing 3D editing methods, with the distilled model runs about $4\times$ faster than the base model while retaining comparable quality.

[1] https://sketchfab.com/

Table 1: Comparison of existing 3D asset editing datasets. Our Alchemy3D-1M incorporates more diverse editing types and is more than 10× larger in scale than existing related datasets.

Dataset	Scale	PBR	Editing Types					
			Add	Remove	Replace	Appearance	Animation	Segmentation
Steer3D	100K	✗	✓	✓	✗	✓	✗	✗
Nano3D-100K (ICLR'26)	100K	✗	✓	✓	✓	✗	✗	✗
3DEditVerse (ICML'26)	116K	✗	✓	✓	✓	✗	✗	✗
PxForm (SIGGRAPH Asia'26)	102K	✗	✓	✓	✓	✓	✗	✗
Alchemy3D-1M	**1.38M**	✓	✓	✓	✓	✓	✓	✓

2 Related Work

2.1 Native 3D Generation

3D generative models synthesize representations such as point clouds (Luo & Hu, 2021; Nichol et al., 2022), neural fields (Müller et al., 2023), triplanes (Wang et al., 2023), and 3D Gaussian splats (He et al., 2024). TRELLIS (Xiang et al., 2025) introduced a structured latent space that supports meshes, NeRFs (Mildenhall et al., 2021), and 3D Gaussian splats (Kerbl et al., 2023) through a unified VAE (Kingma & Welling, 2013). TRELLIS.2 (Xiang et al., 2026) subsequently introduced the compact O-Voxel representation and decomposed asset generation into sparse-structure, shape, and PBR-material stages. Alchemy3D adopts this representation and transforms the generation pipeline into a source-conditioned editor.

2.2 3D Asset Editing

Early 3D editing methods rely on optimization, either reconstructing assets from images modified by a 2D editor (Haque et al., 2023) or applying score-distillation sampling (Poole et al., 2023) to optimize a 3D representation (Sella et al., 2023; Li et al., 2024; Palandra et al., 2024). Subsequent systems edit multi-view renderings or videos and reconstruct an asset from the modified observations; agentic variants primarily automate view selection (Qi et al., 2024; Huang et al., 2025; Qu et al., 2025; Zheng et al., 2025). Although flexible, these pipelines are computationally expensive and prone to cross-view inconsistency.

Feed-forward approaches improve efficiency by adapting techniques from zero-shot image editing. VoxHammer (Li et al., 2026b) combines TRELLIS (Xiang et al., 2025) with RF-Inversion (Rout et al., 2024) and attention manipulation (Wang et al., 2025a), while Nano3D (Ye et al., 2025a) integrates TRELLIS with FlowEdit (Kulikov et al., 2025). Their reliance on training-free image-editing mechanisms, however, limits robustness on complex 3D transformations. Steer3D (Ma et al., 2025) and 3DEditFormer (Xia et al., 2025) instead train 3D editors on constructed paired datasets, demonstrating the value of task-specific supervision but remaining constrained by data scale and coverage. Concurrently, PartFlow (Weng et al., 2026) constructs editing pairs from part-segmentation datasets and is the closest prior setting to ours. Alchemy3D extends this direction with substantially larger and broader dataset, a unified model family, and open-world evaluation.

3 Alchemy3D-1M Dataset

As shown in Table 1, existing datasets for 3D asset editing remain limited in both scale and editing types. Among them, Steer3D (Ma et al., 2025) independently reconstructs the before and after assets from corresponding image pairs, which can lead to poor identity preservation. Nano3D-100K and 3DEditVerse (Ye et al., 2025a; Xia et al., 2025) focus primarily on structural edits, such as addition, removal, and replacement. PxForm, in contrast, curates editing pairs from labeled 3D part datasets (Dong et al., 2025), limiting the diversity and complexity of the assets. In contrast, Alchemy3D-1M, to the best of our knowledge, is the first million-scale dataset for 3D asset editing.

The construction of Alchemy3D-1M is divided into three distinct categories. For structural editing data which cover addition, removal, replacement, as well as local and global appearance modifications, the dataset generation follows a structured three-stage pipeline consisting of *preparation*,

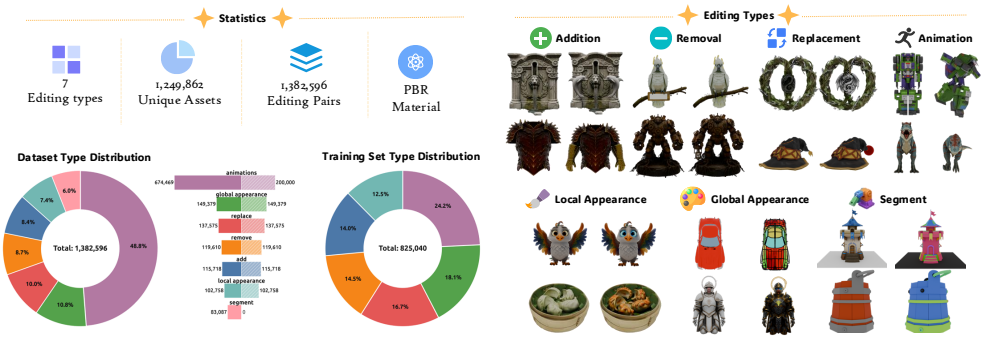

Figure 2: Statistics of Alchemy3D-1M dataset. Alchemy3D-1M includes 1.25M unique assets and 1.38 M editing pairs, across 7 different editing types.

construction, and *filtering*. (1) In the data *preparation* stage, source images are rendered from public 3D datasets (Deitke et al., 2023; Fu et al., 2021; Chang et al., 2015; Collins et al., 2022; Zhang et al., 2025; Khanna et al., 2024) or generated using a text-to-image model (Labs, 2025). The paired 3D source assets are re-generated with TRELLIS.2 (Xiang et al., 2026) according to the source images, where the sampling trajectories are stored for subsequent operations. Then, we use Qwen3-VL (Bai et al., 2025) to write an editing text instruction for these source images and introduce FLUX.2-Dev-Turbo to generate target edited images. (2) During data *construction*, for removal and addition types, we detect (Jiang et al., 2026) and segment (Carion et al., 2025) 2D edit-region masks based on the text instruction and the rendered views from reconstructed assets. These masks are then lifted into 3D using SegViGen (Li et al., 2026a). For replacement type, the 3d editing mask is instead estimated by training-free editing method (Kulikov et al., 2025). Guided by these 3D masks, trajectory-aware 3D inpainting modifies the target region while maintaining the original diffusion trajectories in unedited areas to produce the edited 3D model. (3) Finally, in the *filtering* stage, the generated 3D and image editing pairs are checked and recaptioned with Qwen3.6-27B (Qwen Team, 2026b), where the low quality assets are discarded.

For animation data, we sample pairs of frames from motion sequences in existing animation datasets (Deitke et al., 2023; Zhang et al., 2025; Li et al., 2021; Inc., 2025) and articulation datasets (Iliash et al., 2026). For articulated assets without explicit motion sequences, we simulate motions with a physics simulator (Xiang et al., 2020). We then compute inter-frame feature similarities with DINOv3 (Siméoni et al., 2025) and discard pairs without noticeable motion.

For segmentation data, we collect assets from 3D part segmentation datasets (Mo et al., 2019; Wang et al., 2025b; Ding et al., 2025) and apply a custom color assignment algorithm to colorize individual parts. Rendered images of these colorized assets then serve as visual instructions to guide the segmentation. The detailed construction pipeline is provided in Appendix A.2.

As shown in Table 1 and Fig. 2, Alchemy3D-1M contains **1,249,862 unique 3D assets** and **1,382,596 annotated editing pairs** spanning **seven editing categories**: addition, removal, replacement, local appearance, global appearance, animation, and segmentation. For the training set, we downsample the animation pairs to 200k to balance the distribution across editing types and reserve the segmentation pairs exclusively for downstream evaluation. This yields a final training set of **825,040 editing pairs** for unified training. Compared to existing 3D editing datasets, Alchemy3D-1M is substantially larger, more diverse, and covers a broader spectrum of editing tasks, serving as a valuable asset for the 3D vision community.

4 Alchemy3D Model

4.1 Architecture.

3D asset editing is inherently a generative task: one text or image instruction can yield multiple valid edited assets. While current editing approaches (Xia et al., 2025; Weng et al., 2026; Zhou et al., 2026b; Ma et al., 2025) train generative models to modify 3D assets, they treat the source

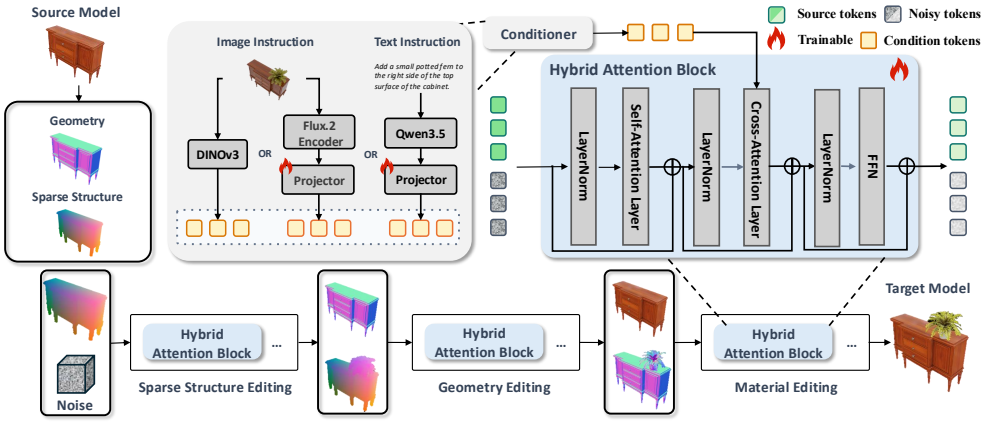

Figure 3: Illustration of Alchemy3D model. Following Trellis.2, we design a three-stage editing pipeline that sequentially edits sparse structure (voxel occupancy), geometry (fine-grained surface), and material (PBR attributes). Each stage comprises hybrid attention blocks that denoise noisy target tokens by attending to source tokens from the source asset alongside condition tokens from image/text instructions. For clarity, we display decoded outputs from denoised tokens at each stage.

asset strictly as external guidance via an auxiliary branch such as ControlNet (Zhang et al., 2023), which may miss crucial cross-feature interactions between the source and target assets.

As illustrated in Fig. 3, rather than introducing an auxiliary branch to inject source features, we directly incorporate tokens encoded from both the source asset and the noisy target state into the hybrid attention blocks, enabling in-depth feature interaction. Following Trellis.2 (Xiang et al., 2026), we organize these attention blocks into a three-stage flow transformer that sequentially edits sparse structure (voxel occupancy), geometry (fine-grained surface), and materials (PBR attributes).

The requested edit is specified by an external condition, which can be either a text instruction or an edited image instruction. To support diverse usage scenarios, we adopt multiple encoders to process different conditional signals. For *text-conditioned* editing, we encode the text instruction with Qwen3.5-2B (Qwen Team, 2026a) followed by a lightweight trainable projector, and the resulting features provide semantic guidance for specific editing operations. For *image-conditioned* editing, we use DINOv3 (Siméoni et al., 2025) to extract the condition tokens. Since DINO focuses on extracting semantic features instead of low-level vision features, we optionally trained a separate texture editing model with FLUX.2 encoder (Labs, 2025) followed by a lightweight projector for the material editing stage, which can help produce better color and texture fidelity. Comparison between DINOv3 and FLUX.2 encoder is presented in Sec. 6.2.

4.2 Training Objectives.

Flow Matching. During training, a pair of source and target assets is sampled and encoded into the corresponding latent representations, which can be sparse voxel, geometry, or material latents depending on the stage. The target latent X_0 is then interpolated with Gaussian noise according to $X_t = (1-t)X_0 + tX_1$, and concatenated token-wise with the source latent X'. The generative flow model v_θ predicts the velocity field from the concatenated input tokens $[X'\ X_t]$, conditioned on the image or text instruction c and the sampled timestep t. We then train the model with the standard optimal transport flow-matching objective (Lipman et al., 2023):

$$\mathcal{L}_{\text{FM}} = \mathbb{E}_{t, X_t \sim p_t}\left[||(X_1 - X_0) - v_\theta(X', X_t, c, t)||_2^2 \right], \tag{1}$$

Few-Step Distillation. While effective, concatenating source and target tokens roughly doubles the sequence length and consequently increases the computational cost of attention. We recover this efficiency through a two-stage distillation procedure inspired by few-step video distillation (Gu et al., 2026). In the first stage, we train a continuous-step flow map following MeanFlow (Geng

et al., 2025), with the objective:

$$\mathcal{L}_{\text{MF}}(\theta) = \mathbb{E}\left[\|u_\theta(X', X_t, c, r, t) - \text{sg}(u_{\text{tgt}})\|_2^2\right],$$
$$u_{\text{tgt}} = v(X_t, t) - (t-r)\frac{du_\theta(X', X_t, c, r, t)}{dt}, \tag{2}$$

where the flow map u_θ, which differs slightly from the pretrained flow model, predicts the target flow conditioned on an additional timestep r. Here, $\text{sg}(\cdot)$ denotes the stop-gradient operation, and v denotes the pretrained flow model defined in Eq. 1. Following the transition model (Wang et al., 2026), we approximate the time derivative as

$$\frac{du_\theta(X', X_t, c, r, t)}{dt} \approx \frac{u_\theta(X', X_{t+\Delta t}, c, r, t+\Delta t) - u_\theta(X', X_{t-\Delta t}, c, r, t-\Delta t)}{2\Delta t}. \tag{3}$$

Starting from this flow map, we then perform on-policy distribution matching distillation following DMD (Yin et al., 2024b;a). Given an initial Gaussian noise $X_1 \sim \mathcal{N}(0, I)$, the student model first approximates a clean sample through its inference trajectory $\hat{X}_0 = f_\theta(X_1)$, which is then re-noised at a sampled timestep t as $X_t = (1-t)\hat{X}_0 + t\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$. The DMD gradient is given by

$$\nabla_\theta \mathcal{L}_{\text{DMD}} = -\mathbb{E}_{t, X_1}\left[\left(s_{\text{real}}(X_t, t) - s_{\text{fake}}(X_t, t)\right)\frac{\partial f_\theta(X_1)}{\partial \theta}\right], \tag{4}$$

where s_{real} and s_{fake} denote the score functions of the pretrained teacher and the student-generated distributions, respectively. We replace the adversarial loss in DMD2 Yin et al. (2024a) with the MeanFlow objective in Eq. 2. Specifically, we optimize

$$\mathcal{L}_{\text{on-policy}} = \mathcal{L}_{\text{DMD}} + \mathcal{L}_{\text{MF}}. \tag{5}$$

Throughout the few-step distillation process, only the attached LoRA modules are trainable.

5 GEDIT3D-BENCH

Existing benchmarks for 3D asset editing have several fundamental limitations. Edit3D-Bench, Eval3DEdit, and TANGOEdit (Li et al., 2026b; Zhou et al., 2026b; Lim et al., 2026) each collect only about 100 assets from existing 3D datasets (Deitke et al., 2023; Downs et al., 2022; Yang et al., 2024), which is too small for a comprehensive evaluation. Another line of work, including Steer3D, Nano3D, 3DEditFormer, and PartFlow (Ma et al., 2025; Ye et al., 2025a; Xia et al., 2025; Weng et al., 2026), builds test sets by splitting the training data of the corresponding method, which risks leakage and overfitting to that specific distribution. The protocols are also narrow. The first line of works evaluates only CLIP or DINO similarity between rendered views and the input image. The second provides a low-quality ground-truth asset and measures alignment to it, which overlooks the quality of that reference and the one-to-many nature of generative editing. **GEdit3D-Bench** instead provides large-scale, open-world data and evaluates complementary aspects of editing quality.

Construction. Each sample contains a source asset $\mathcal{A}$, source and target captions $\mathcal{C}_{\text{src}}$ and $\mathcal{C}_{\text{tgt}}$, a source rendered image $\mathcal{I}_{\text{src}}$, an editing instruction $\mathcal{I}$, and a target edited image $\mathcal{I}_{\text{tgt}}$. We first assemble synthetic assets generated with Hunyuan3D V3.1 and recently released Sketchfab assets from diverse categories. Then, we caption multi-view renderings, generate instructions for six editing types, and edit one source view to produce the visual target. Finally, automated checks and human review will be introduced to remove inconsistent or low-quality samples and generate target captions for text-based evaluation. More details can be found in Appendix A.3.

Metrics. Given an edited asset $\mathcal{A}'$, we evaluate its multi-view renderings without treating a generated 3D target as ground truth. **View Quality** averages aesthetic-predictor-v2.5, MANIQA (Yang et al., 2022), and MUSIQ (Ke et al., 2021) scores across views (Chen & Mo, 2022). **Reference Alignment** comprises image alignment A_I and text alignment A_T, instantiated with complementary encoders including EVA-CLIP (Sun et al., 2023), SigLIP (Zhai et al., 2023), BLIP (Li et al., 2022), DINOv3 (Siméoni et al., 2025), and Uni3D (Zhou et al., 2024), where applicable. Following common practice in image and video evaluation (Zhou et al., 2026a; Liang et al., 2026; Jiang et al., 2024; Ye et al., 2025b), we employ **MLLM (Multimodal Large Language Model) as judge**: *Success Rate (SR)* is a binary measure of whether the requested edit succeeds, while *Instruction Following (IF)*, *Identity Preservation (IP)*, and *Visual Quality (VQ)* are scored from 1 to 100.

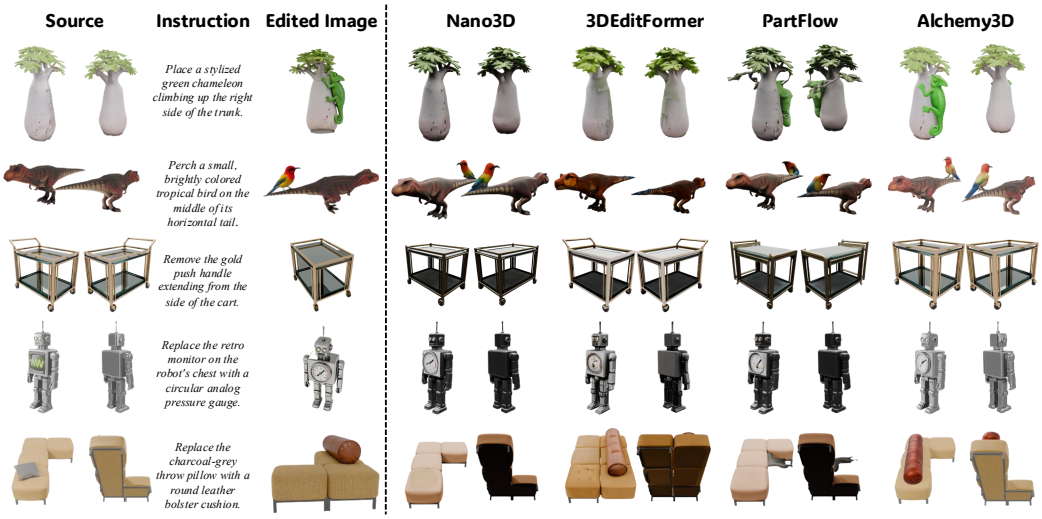

Figure 4: Qualitative comparison between Alchemy3D and baselines. We show only addition, removal, and replacement, the editing types supported by all baselines.

6 Experiments

6.1 Evaluation Setting.

We evaluate five variants of our model: (1) **Alchemy3D**, our default image-conditioned editor based on the DINOv3 encoder; (2) **Alchemy3D-Flux**, which uses the FLUX.2 image encoder to improve color and texture fidelity; (3) **Alchemy3D-Instruct**, which uses Qwen3.5-2B (Qwen Team, 2026a) to support text-instruction input; (4) **Alchemy3D-Turbo**, a few-step image-conditioned editor distilled from Alchemy3D; and (5) **Alchemy3D-Segment**, which is fine-tuned from Alchemy3D for multi-view-guided 3D part segmentation.

For baselines and benchmarks, We compare image-conditioned Alchemy3D with Nano3D (Ye et al., 2025a), 3DEditFormer (Xia et al., 2025), and PartFlow (Weng et al., 2026). For instruction-conditioned editing, we compare with Steer3D (Ma et al., 2025), In this work, we adopt GEdit3D-Bench (Sec. 5) as our primary evaluation benchmark. For a more comprehensive comparison, we additionally evaluate on existing datasets, including Eval3DEdit (Zhou et al., 2026b), 3DEditVerse (Xia et al., 2025), Nano3D-100K (Ye et al., 2025a), and the Edit3D-Bench introduced by VoxHammer (Li et al., 2026b). Since Nano3D does not release an official test split, we randomly sample 1.5K pairs from its data for evaluation. For text-conditioned editors, we conduct comparisons on GEdit3D-Bench and on the Edit3D-Bench introduced by Steer3D.

Unless otherwise stated, MLLM scores use Gemini-3.8-Flash.[2] On GEdit3D-Bench, a random 400-sample subset is used for MLLM scoring and the user study. Alchemy3D and Alchemy3D-Instruct are sampled with 12 steps and classifier-free guidance on sparse structure and shape. Alchemy3D-Turbo uses 3 steps, with guidance fused during training. Please refer to Sec. A.4.1 for more details.

6.2 Experimental Results

Image-conditioned editing. As demonstrated in Tables 2 and 3, **Alchemy3D** and **Alchemy3D-Turbo** outperform existing methods across both alignment and MLLM scores, with human preference for our methods significantly exceeding that of the baselines. More details on the human user study are provided in Sec. A.4.1. Qualitative comparisons in Fig. 4 indicate our methods faithfully execute the requested changes while well preserving the identity of the source asset. In terms of sampling efficiency, Alchemy3D-Turbo requires 1.231 seconds, making it faster than all baseline methods. Although PartFlow uses a lower-complexity ControlNet-style architecture, it samples

[2] https://deepmind.google/models/model-cards/gemini-3-8-flash/

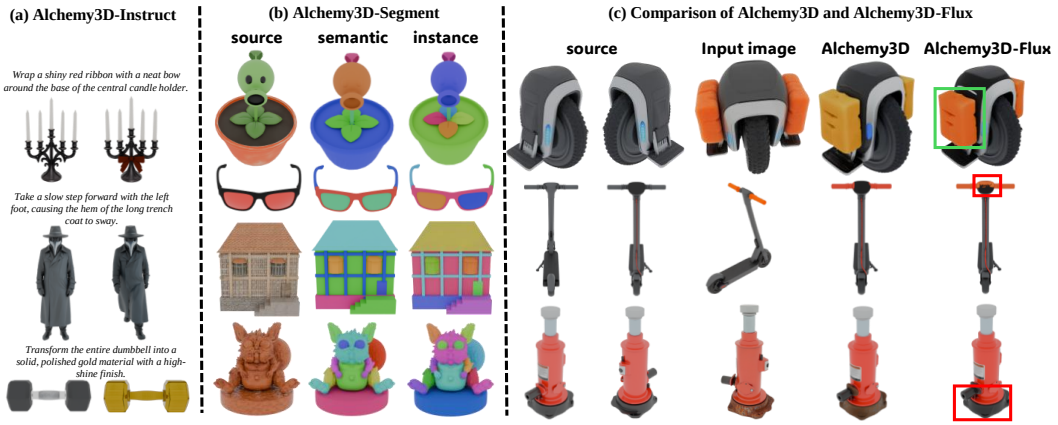

Figure 5: (a) Examples from Alchemy3D-Instruct. (b) Segmentation results from Alchemy3D-Segment, where the segmentation criterion (e.g., semantic or instance) is controlled by the image instruction. (c) Comparison between Alchemy3D and Alchemy3D-Flux.

Table 2: Quantitative results on GEdit3D-Bench for addition, removal, and replacement. Per-type results are reported in Table 9. Human preference scores do not sum to 100% because participants could also select a "tie" option when neither result was clearly better. Sampling time is measured on a single NVIDIA H200 GPU and averaged over 10 examples.

	View Qual.		Ref. Alignment			MLLM				Human	Time
Method	Aes. ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_T^{CLIP} ↑	A_I^{DINO} ↑	SR ↑	VQ ↑	IF ↑	IP ↑	**Pref.**	**(s)** ↓
Nano3D	4.40	72.87	81.59	65.10	69.15	51.14%	70.16	53.84	72.09	5.1%	8.629
3DEditFormer	4.34	70.87	81.83	63.19	68.07	58.90%	66.31	59.79	63.39	5.3%	3.262
PartFlow	4.31	70.96	81.25	62.24	67.78	55.50%	65.27	56.20	66.60	6.5%	7.184
Alchemy3D-Turbo	**4.58**	**72.94**	82.36	**68.31**	72.88	88.18%	69.91	70.71	73.22	–	**1.231**
Alchemy3D	4.43	72.36	**84.91**	67.47	**73.18**	**89.55%**	**72.74**	**71.38**	**76.19**	**70.7%**	5.202

with 50 steps by default and therefore incurs a substantial inference cost. The remaining per-type results are reported in Table 10 and Table 9. As a side comparison, we also report image alignment scores on 3DEditVerse, the Edit3D-Bench introduced by VoxHammer, and Nano3D-100K in Table 4. Alchemy3D and Alchemy3D-Turbo still outperform the baselines.

Instruction-conditioned editing. In this section, we compare text-conditioned editing between Alchemy3D-Instruct and Steer3D (Ma et al., 2025). Since Steer3D (Ma et al., 2025) provides neither text captions nor image instructions, we evaluate editing performance using the aesthetic score for visual quality, together with the MLLM-based metrics SR, IF, and IP described in Sec. 5. As shown in Table 6, Alchemy3D-Instruct, trained across all editing types, outperforms Steer3D's type-specific models on both GEdit3D-Bench and Steer3D's Edit3D-Bench. Examples of Alchemy3D-Instruct are shown in Fig. 5(a). Interestingly, the text-conditioned model underperforms the image-conditioned one, likely due to the ambiguity of text instructions. Motivated by this observation, we explore an agentic text-conditioned editing pipeline that automatically selects a 2D view, edits it, and then invokes the image-conditioned Alchemy3D. A detailed analysis is presented in Sec. A.5.1.

Ablation on Encoder. As shown in Fig. 5(c), replacing DINOv3 with the FLUX.2 encoder improves color and style fidelity, while reducing local editing quality. Though the standard performances of the two models are close (Table 5), Alchemy3D achieves much better performance when adapting to part segmentation with limited data (Table 7). We therefore retain Alchemy3D as our default foundation and release Alchemy3D-Flux for appearance-critical use cases.

Downstream part segmentation. To validate whether the priors acquired through unified editing transfer to downstream applications, we adapt Alchemy3D using LoRA (Hu et al., 2022) to create **Alchemy3D-Segment**, a model designed for 3D part segmentation guided by 1 to 8 segmented views. Results on PartObjaverse-Tiny (Yang et al., 2024) (Table 7) show that **Alchemy3D-Segment** substantially outperforms prior baselines and initialization variants trained from scratch or from TRELLIS.2. These improvements demonstrate that unified 3D editing serves as a powerful pretext

Table 3: Quantitative results on Eval3DEdit. Since Eval3DEdit originally only evaluates CLIP similarities, we additionally report aesthetic score for view quality and SR for MLLM rating.

	Add / Remove / Replace				Action				Style			
Method	Aes.	A_I	A_T	SR	Aes.	A_I	A_T	SR	Aes.	A_I	A_T	SR
Nano3D	4.60	82.31	49.43	40.00%	–	–	–	–	–	–	–	–
3DEditFormer	4.55	81.78	49.24	70.97%	4.06	75.80	44.45	20.00%	–	–	–	–
PartFlow	4.51	81.59	49.32	67.86%	–	–	–	–	4.58	78.10	50.68	40.00%
Alchemy3D-Turbo	**4.64**	82.67	**52.56**	**90.00%**	**4.33**	79.96	**51.46**	**90.00%**	4.61	81.15	52.51	**100.00%**
Alchemy3D	4.60	**83.77**	52.10	83.87%	4.29	**80.84**	51.15	80.00%	**4.64**	**82.40**	**54.41**	90.00%

Table 4: Comparison on other benchmark data.

Method	$A_I^{\text{CLIP}}\uparrow$	$A_I^{\text{SigLIP}}\uparrow$	$A_I^{\text{DINO}}\uparrow$	$A_I^{\text{Uni3D}}\uparrow$
3DEditVerse				
3DEditFormer	69.18	79.18	64.60	36.76
PartFlow	68.08	78.60	63.74	36.46
Alchemy3D-Turbo	72.10	82.05	69.44	36.36
Alchemy3D	**73.10**	**82.18**	**72.20**	**37.41**
Edit3D-Bench (VoxHammer)				
3DEditFormer	76.91	84.99	66.90	38.33
PartFlow	77.55	85.95	69.02	38.93
Alchemy3D-Turbo	79.90	87.28	72.91	38.25
Alchemy3D	**80.91**	**88.44**	**75.04**	**39.06**
Nano3D-100K				
3DEditFormer	66.30	78.06	60.38	36.57
PartFlow	65.42	78.61	61.18	36.20
Alchemy3D-Turbo	69.86	80.81	66.58	37.04
Alchemy3D	**72.22**	**82.37**	**71.11**	**38.06**

Table 5: Comparison between Alchemy3D and Alchemy3D-Flux on GEdit3D-Bench.

Model	Aes. ↑	$A_I^{\text{CLIP}}\uparrow$	$A_T^{\text{CLIP}}\uparrow$	$A_I^{\text{DINO}}\uparrow$
Alchemy3D-Flux	**4.420**	83.43	**66.36**	72.26
Alchemy3D	4.419	**83.53**	**66.36**	**72.52**

Table 6: Quantitative comparison between Steer3D and Alchemy3D-Instruct on GEdit3D-Bench and Edit3D-Bench (Steer3D).

Method	**Aes.**	**SR**	**IF**	**IP**
GEdit3D-Bench				
Steer3D	3.84	12.37%	38.55	35.28
Alchemy3D-Instruct	**4.60**	**45.67%**	**51.81**	**72.56**
Edit3D-Bench (Steer3D)				
Steer3D	3.78	33.73%	38.69	53.94
Alchemy3D-Instruct	**3.94**	**49.60%**	**67.81**	**71.76**

Table 7: Quantitative results for semantic part segmentation on PartObjaverse-Tiny. Trial experiments compare different strategies for training single-view conditioned segmentation.

Method	**mIoU** ↑
Trial Experiments	
From Scratch (1-view)	28.54
From TRELLIS.2 (1-view)	58.68
From Alchemy3D (1-view)	60.08
From Alchemy3D-Flux (LoRA+1-view)	52.55
From Alchemy3D (LoRA+1-view)	61.10
Baselines	
Find3D	19.35
PartField	52.01
P3-SAM	46.58
SegViGen	53.49
Ours	
Alchemy3D-Segment(1 to 8-views)	
(1-view)	63.10
(2-view)	65.23
(4-view)	66.98
(8-view)	**68.70**

task for downstream part understanding. Interestingly, training Alchemy3D-Segment with variable 1-to-8 view inputs yields better 1-view evaluation performance than training on single views alone (Alchemy3D LoRA+1-view), highlighting the value of multi-view data augmentation.

7 Conclusion

In this work, we present **Alchemy3D**, a unified framework spanning scalable data construction, model architecture, and benchmark evaluation for versatile 3D editing. We construct **Alchemy3D-1M**, a dataset of 1.38 million editing pairs across seven editing types, and introduce an editing architecture that fuses tokens from source and edited target 3D representations via self-attention, with text/image instructions injected via cross-attention. Together with a few-step distillation procedure, this architecture yields a series of **Alchemy3D** model variants tailored to different editing use cases. To enable fair, comprehensive evaluation, we also introduce **GEdit3D-Bench**, a large-scale, multi-dimensional benchmark entirely independent of our training data. Extensive experiments show that our models achieve significant improvements over existing 3D editing methods, with the distilled variant achieving a $4\times$ acceleration while maintaining superior performance.

8 AI USE STATEMENT

In this work, we used generative AI tools to edit the manuscript for readability and to assist with writing and debugging code. We have reviewed all AI-assisted work. Two authors checked the revised text, and the AI-assisted code was tested for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Siyu Cao, Hangting Chen, Peng Chen, Yiji Cheng, Yutao Cui, Xinchi Deng, Ying Dong, Kipper Gong, Tianpeng Gu, Xiusen Gu, et al. Hunyuanimage 3.0 technical report. *arXiv preprint arXiv:2509.23951*, 2025.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. SAM 3: Segment anything with concepts. 2025.

Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. 2015.

Chaofeng Chen and Jiadi Mo. IQA-PyTorch: Pytorch toolbox for image quality assessment. [Online]. Available: `https://github.com/chaofengc/IQA-PyTorch`, 2022.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F Yago Vicente, Thomas Dideriksen, Himanshu Arora, et al. Abo: Dataset and benchmarks for real-world 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 2023.

Lihe Ding, Shaocong Dong, Yaokun Li, Chenjian Gao, Xiao Chen, Rui Han, Yihao Kuang, Hong Zhang, Bo Huang, Zhanpeng Huang, et al. Fullpart: Generating each 3d part at full resolution. *arXiv preprint arXiv:2510.26140*, 2025.

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

Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas B McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017.

Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 2021.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 2025.

Georgia Gkioxari, Justin Johnson, and Jitendra Malik. Mesh R-CNN. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, 2019.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation, 2026.

Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

Xianglong He, Junyi Chen, Sida Peng, Di Huang, Yangguang Li, Xiaoshui Huang, Chun Yuan, Wanli Ouyang, and Tong He. Gvgen: Text-to-3d generation with volumetric representation. In *European Conference on Computer Vision*, 2024.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33*, 2020.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, 2022.

Junchao Huang, Xinting Hu, Shaoshuai Shi, Zhuotao Tian, and Li Jiang. Edit360: 2d image edits to 3d assets from any angle. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Team Hunyuan3D, Shuhui Yang, Mingxin Yang, Yifei Feng, Xin Huang, Sheng Zhang, Zebin He, Di Luo, Haolin Liu, Yunfei Zhao, et al. Hunyuan3d 2.1: From images to high-fidelity 3d assets with production-ready pbr material. *arXiv preprint arXiv:2506.15442*, 2025.

Denys Iliash, Jiayi Liu, Egor Fokin, Qirui Wu, Ali Mahdavi-Amiri, Manolis Savva, and Angel X. Chang. Artiverse: A diverse and physically grounded dataset for articulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Adobe Systems Inc. Mixamo. `https://www.mixamo.com/`, 2025.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 1991.

Dongfu Jiang, Max Ku, Tianle Li, Yuansheng Ni, Shizhuo Sun, Rongqi Fan, and Wenhu Chen. Genai arena: An open evaluation platform for generative models. *Advances in Neural Information Processing Systems*, 2024.

Qing Jiang, Junan Huo, Xingyu Chen, Yuda Xiong, Zhaoyang Zeng, Yihao Chen, Tianhe Ren, Junzhi Yu, and Lei Zhang. Detect anything via next point prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Junjie Ke, Qifei Wang, Yilin Wang, Peyman Milanfar, and Feng Yang. Musiq: Multi-scale image quality transformer, 2021. URL https://arxiv.org/abs/2108.05997.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 2023.

Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X Chang, and Manolis Savva. Habitat synthetic scenes dataset (hssd-200): An analysis of 3d scene scale and realism tradeoffs for objectgoal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: benchmarking large-scale scene reconstruction. *ACM Trans. Graph.*, 2017.

Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. https://bfl.ai/blog/flux-2, 2025.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*. PMLR, 2022.

Lin Li, Haoran Feng, Zehuan Huang, Haohua Chen, Wenbo Nie, Shaohua Hou, Keqing Fan, Pan Hu, Sheng Wang, Buyu Li, et al. Segvigen: Repurposing 3d generative model for part segmentation. *arXiv preprint arXiv:2603.16869*, 2026a.

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 *2026 International Conference on 3D Vision (3DV)*, 2026b.

Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Yuhan Li, Yishun Dou, Yue Shi, Yu Lei, Xuanhong Chen, Yi Zhang, Peng Zhou, and Bingbing Ni. Focaldreamer: Text-driven 3d editing via focal-fusion assembly. In *Proceedings of the AAAI conference on artificial intelligence*, 2024.

Hanwen Liang, Yuyang Yin, Dejia Xu, Hanxue Liang, Zhangyang Wang, Konstantinos N. Plataniotis, Yao Zhao, and Yunchao Wei. Diffusion4d: Fast spatial-temporal consistent 4d generation via video diffusion models. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Sen Liang, Cong Wang, Zhentao Yu, Fengbin Guan, Zhengguang Zhou, Teng Hu, Youliang Zhang, Yuan Zhou, Xin Li, Qinglin Lu, et al. Goku: A million-scale universal dataset and benchmark for instruction-based video editing. *arXiv preprint arXiv:2606.30599*, 2026.

Siwoo Lim, Sunjae Yoon, Gwanhyeong Koo, Hyeonseo Yun, and Chang D Yoo. Tango: Training-free 3d editing via tangent-space guidance and optimization. *arXiv preprint arXiv:2607.14927*, 2026.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021.

Ziqi Ma, Hongqiao Chen, Yisong Yue, and Georgia Gkioxari. Feedforward 3d editing via text-steerable image-to-3d. *arXiv preprint arXiv:2512.13678*, 2025.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 2021.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019.

Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

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.

Francesco Palandra, Andrea Sanchietti, Daniele Baieri, and Emanuele Rodola. Gsedit: Efficient text-guided editing of 3d objects via gaussian splatting. *arXiv preprint arXiv:2403.05154*, 2024.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023.

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. *CoRR*, 2024.

Huaizhi Qu, Ruichen Zhang, Shuqing Luo, Luchao Qi, Zhihao Zhang, Xiaoming Liu, Roni Sengupta, and Tianlong Chen. Editcast3d: Single-frame-guided 3d editing with video propagation and view selection. *arXiv preprint arXiv:2510.13652*, 2025.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL `https://qwen.ai/blog?id=qwen3.5`.

Qwen Team, 2026b. URL `https://qwen.ai/blog?id=qwen3.6-27b`.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL `https://arxiv.org/abs/2408.00714`.

Litu Rout, Yujia Chen, Nataniel Ruiz, Constantine Caramanis, Sanjay Shakkottai, and Wen-Sheng Chu. Semantic image inversion and editing using rectified stochastic differential equations. *arXiv preprint arXiv:2410.10792*, 2024.

Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *9th International Conference on Learning Representations*, 2021.

Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*, 2023.

Jiangshan Wang, Junfu Pu, Zhongang Qi, Jiayi Guo, Yue Ma, Nisha Huang, Yuxin Chen, Xiu Li, and Ying Shan. Taming rectified flow for inversion and editing. In *International Conference on Machine Learning*, 2025a.

Penghao Wang, Yiyang He, Xin Lv, Yukai Zhou, Lan Xu, Jingyi Yu, and Jiayuan Gu. Partnext: A next-generation dataset for fine-grained and hierarchical 3d part understanding. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego,CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December5, 2025*, 2025b.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.*, 2004.

Zidong Wang, Yiyuan Zhang, Xiaoyu Yue, Xiangyu Yue, Yangguang Li, Wanli Ouyang, and Lei Bai. Transition models: Rethinking the generative learning objective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Jiawei Weng, Saining Zhang, Zhenxin Diao, Peishuo Li, Henghaofan Zhang, Junhao Chen, and Hao Zhao. Feedforward 3d editing learns from semantic-part transformation. *arXiv preprint arXiv:2605.27351*, 2026.

Ruihao Xia, Yang Tang, and Pan Zhou. Towards scalable and consistent 3d editing. *arXiv preprint arXiv:2510.02994*, 2025.

Fanbo Xiang, Yuzhe Qin, Kaichun Mo, Yikuan Xia, Hao Zhu, Fangchen Liu, Minghua Liu, Hanxiao Jiang, Yifu Yuan, He Wang, Li Yi, Angel X. Chang, Leonidas J. Guibas, and Hao Su. SAPIEN: A simulated part-based interactive environment. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, et al. Native and compact structured latents for 3d generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Sidi Yang, Tianhe Wu, Shuwei Shi, Shanshan Lao, Yuan Gong, Mingdeng Cao, Jiahao Wang, and Yujiu Yang. Maniqa: Multi-dimension attention network for no-reference image quality assessment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

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

Junliang Ye, Shenghao Xie, Ruowen Zhao, Zhengyi Wang, Hongyu Yan, Wenqiang Zu, Lei Ma, and Jun Zhu. Nano3d: A training-free approach for efficient 3d editing without masks. *arXiv preprint arXiv:2510.15019*, 2025a.

Yang Ye, Xianyi He, Zongjian Li, Bin Lin, Shenghai Yuan, Zhiyuan Yan, Bohan Hou, and Li Yuan. Imgedit: A unified image editing dataset and benchmark. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025b.

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Frédo Durand, and Bill Freeman. Improved distribution matching distillation for fast image synthesis. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a.

Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Frédo Durand, William T. Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, 2024b.

Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. Anyedit: Mastering unified high-quality image editing for any idea. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018.

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

Yang Zheng, Mengqi Huang, Nan Chen, and Zhendong Mao. Pro3d-editor: A progressive-views perspective for consistent and precise 3d editing. *Advances in Neural Information Processing Systems*, 2025.

Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. In *International Conference on Learning Representations*, 2024.

Yu Zhou, Xiaoyan Yang, Bojia Zi, Lihan Zhang, Ruijie Sun, Weishi Zheng, Haibin Huang, Chi Zhang, and Xuelong Li. Point2insert: Video object insertion via sparse point guidance. *arXiv preprint arXiv:2602.04167*, 2026a.

Zhenglin Zhou, Fan Ma, Chengzhuo Gui, Xiaobo Xia, Hehe Fan, Yi Yang, and Tat-Seng Chua. Anchorflow: Training-free 3d editing via latent anchor-aligned flows. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026b.

A Appendix

A.1 Model Training Details and Compute Report

A.1.1 Hyperparameters

LoRA. For Alchemy3D-Segment, we attach a LoRA module to the query, key, value, and output projections of each transformer block, with rank 96. For Alchemy3D-Turbo, in addition to these projection layers, we attach LoRA modules to the FFN, AdaLN modulation, input and output layers, and the two time embedders, with rank 256.

Classifier-free Guidance. Both MeanFlow (Geng et al., 2025) and DMD (Yin et al., 2024b;a) fuse classifier-free guidance into the model during training. We follow most of the guidance settings used when sampling the pretrained generator. We found, however, that a MeanFlow model with guidance disabled, as used in the original sampler, produced noisy textures. We therefore set the guidance strength to 3.0 when training the third-stage PBR editing model.

A.1.2 Training Compute Report

We report the core training configurations and compute resources in Tab. 8 to facilitate reproduction.

Table 8: Training configurations of our models. Stage 1 refers to sparse-structure editing, stage 2 to Shape SLat editing, and stage 3 to PBR SLat editing.

Model Name	Stage	Initialization	Training Steps	Batch Size	# GPUs (NVIDIA H200)	Training Time (hours)
Alchemy3D	1	TRELLIS.2	62.5K	192	16	∼83.33
Alchemy3D	2	TRELLIS.2	65K	128	16	∼133.61
Alchemy3D	3	TRELLIS.2	65K	128	16	∼162.50
Alchemy3D-Flux	3	TRELLIS.2-Flux	50K	128	16	∼125
Alchemy3D-Instruct	1	TRELLIS.2-Instruct	75K	192	16	∼105.83
Alchemy3D-Instruct	2	TRELLIS.2-Instruct	40K	128	16	∼100
Alchemy3D-Instruct	3	TRELLIS.2-Instruct	60K	96	16	∼100
Alchemy3D-Segment (Trial)	3	Alchemy3D	10K	64	8	∼10.83
Alchemy3D-Segment	3	Alchemy3D-Segment (Trial)	20K	64	8	∼28.89
Alchemy3D-MF	1	Alchemy3D	15K	288	24	∼74.44
Alchemy3D-MF	2	Alchemy3D	15K	192	24	∼65.83
Alchemy3D-MF	3	Alchemy3D	10K	192	24	∼55.83
Alchemy3D-Turbo	1	Alchemy3D-MF	12K	288	24	∼56.67
Alchemy3D-Turbo	2	Alchemy3D-MF	12K	96	24	∼37.78
Alchemy3D-Turbo	3	Alchemy3D-MF	14K	96	24	∼43.33

A.2 Alchemy3D-1M Construction Pipeline

In this section, we describe the construction pipeline of Alchemy3D-1M (Sec. 3). It has three stages: asset preparation, type-specific construction, and data filtering with condition generation.

A.2.1 Stage 1: Asset Preparation

Animation. For animation editing, we collect assets with temporal motion from multiple public datasets, including the animated subset of Objaverse-XL (Deitke et al., 2023) filtered by Diffusion4D (Liang et al., 2024), TexVerse (Zhang et al., 2025), and DeformingThings4D (Li et al., 2021). To further diversify character-motion combinations, we randomly pair rigged characters from Mixamo (Inc., 2025) with different motion sequences. We additionally incorporate articulated assets from ArtiVerse (Iliash et al., 2026) and PartNet-Mobility (Mo et al., 2019).

Segmentation. For part segmentation, we collect assets with dense semantic part annotations from PartNet (Mo et al., 2019), PartNeXT (Wang et al., 2025b), and PartVerseXL (Ding et al., 2025). These annotations are directly used to construct the corresponding segmentation supervision.

Other editing types. For the remaining five edit categories, namely addition, removal, replacement, local appearance editing, and global appearance editing, we adopt a shared preparation pipeline. We first construct a large pool of source images by rendering assets from existing 3D datasets (Deitke et al., 2023; Fu et al., 2021; Chang et al., 2015; Collins et al., 2022; Zhang et al., 2025; Khanna et al., 2024), together with internally generated asset-centric images synthesized by a text-to-image foundation model (Labs, 2025). Given each source image, Qwen3-VL (Bai et al., 2025) generates edit instructions according to edit-type-specific prompting templates. The source image and generated instruction are then provided to FLUX.2-Dev-Turbo (Labs, 2025) to synthesize the corresponding edited target image. The generated source-target image pairs are subsequently verified using Qwen3-VL to remove semantically inconsistent or low-quality edits.

For each retained source image, we reconstruct the corresponding source 3D asset using TRELLIS.2 (Xiang et al., 2026). To ensure the quality of the reconstructed assets, we render multiple views and employ Qwen3-VL to assess their geometry and appearance, discarding assets with noticeable reconstruction artifacts or quality issues. For the accepted source assets, we additionally preserve the intermediate diffusion trajectories, which are later reused for trajectory-aware 3D inpainting to preserve the unedited regions during editing.

A.2.2 Stage 2: Type-specific Construction Pipeline

Animation. For assets with existing motion sequences, we uniformly sample keyframes at fixed temporal intervals and randomly pair different keyframes of the same asset to construct animation editing pairs. For articulated assets, we use SAPIEN (Xiang et al., 2020) to simulate joint interactions for each annotated joint, obtain the resulting motion trajectories, and similarly sample and randomly pair keyframes from the same asset.

Segmentation. For each asset with K annotated parts, we construct a K-color palette with well-separated hues. We first randomly sample the initial hue h_0 and generate the remaining hues using the golden-ratio increment:

$$\begin{aligned} h_0 &\sim \mathcal{U}\,[0,1)\,, \\ h_{i+1} &= (h_i + \phi) \bmod 1, \end{aligned} \tag{6}$$

where $\phi = \frac{\sqrt{5}-1}{2} \approx 0.618$ is the golden-ratio increment. For each part, the saturation and value are independently sampled as $s_i \sim \mathcal{U}\,[0.7, 0.9]$ and $v_i \sim \mathcal{U}\,[0.8, 0.95]$. The resulting HSV colors are converted to RGB and assigned to mesh faces according to their annotated part labels, producing the corresponding part segmentation representation.

Addition/Removal. Addition and removal are constructed using a shared segment-and-remove pipeline. We independently construct image-editing vocabularies for the two editing types rather than deriving one exclusively by reversing the other, thereby avoiding the limited diversity and potential distributional bias introduced by relying on a single editing direction.

For each image pair, we reconstruct the asset in the state where the edited object is present using TRELLIS.2. For removal, this reconstructed asset directly serves as the source asset. For addition, it instead serves as the target asset, and the resulting pair is reversed after constructing its object-absent counterpart.

We render multiple views of the reconstructed asset and employ Rex-Omni (Jiang et al., 2026) to identify the object to be removed. SAM3 (Carion et al., 2025) is subsequently used to obtain dense 2D segmentations. Among all rendered views, we select the view that jointly maximizes the visible area and the number of detected instances of the segmented object, thereby retaining as many target components as possible. The selected rendering is then converted into a two-color map, where the segmented object instances are colored white and the remaining objects are colored dark gray. Together with the corresponding 3D asset, this map is provided to SegViGen (Li et al., 2026a) to recover the editable region in 3D. We further apply binary clustering to the predicted segmentation,

assuming two clusters corresponding to the editable and unedited regions, and use the resulting cluster assignment to construct the final binary 3D mask.

Given the recovered 3D mask, we construct the object-absent counterpart through trajectory-aware 3D inpainting. Instead of re-noising the reconstructed asset during sampling, we directly reuse the diffusion trajectory recorded when generating the original asset. At each sampling step, the latent in the unedited region is replaced with the corresponding latent from the stored trajectory:

$$\begin{aligned} \tilde{z}_t &= x_t \odot (1 - M) + z'_t \odot M, \\ z'_{t-\Delta t} &= \tilde{z}_t - \Delta t \, v_\theta(\tilde{z}_t, t, c), \end{aligned} \tag{7}$$

where $\Delta t > 0$ is the sampling interval, M is the binary 3D mask with $M = 1$ indicating the editable region, x_t is the latent from the stored diffusion trajectory of the original asset at timestep t, and z'_t is the current latent generated under the edited condition c. The mask is projected to the corresponding representations, and the same procedure is applied throughout the sparse-structure, Shape SLat, and PBR SLat generation stages. Consequently, the unedited regions remain explicitly constrained by the original generation trajectory throughout the reconstruction process.

Replacement. Unlike addition and removal, replacement edits simultaneously involve the removal of an existing object and the introduction of a new one, making the edited region more challenging to localize through direct segmentation. We therefore estimate the editable region by comparing the sparse structures before and after editing.

Specifically, we employ FlowEdit (Kulikov et al., 2025) to approximately transform the source sparse structure conditioned on the edited image. The spatial differences between the resulting structure and the original source structure are used to estimate the editable region, from which we construct a binary 3D mask. Importantly, FlowEdit is used only for estimating the spatial extent of the edit rather than generating the final edited asset. The estimated mask is subsequently used in the trajectory-aware 3D inpainting procedure described above to generate the final geometry and appearance while preserving the unedited regions of the source asset.

Local Appearance. For local appearance editing, we adopt the same 3D mask construction procedure as in addition and removal to localize the edited region. We preserve the sparse structure of the source asset and perform mask-guided 3D inpainting only during the Shape SLat and PBR SLat generation stages. We do not hold the Shape SLat fixed, because appearance modifications, such as material or style changes, may also require adjustments to fine geometric details while preserving the overall structure of the asset.

Global Appearance. For global appearance editing, no spatial mask is required. We preserve the sparse structure of the source asset and regenerate the Shape SLat and PBR SLat conditioned on the edited image, allowing the appearance of the asset to be globally modified while maintaining its overall structure.

A.2.3 Stage 3: Data Filtering and Condition Generation

The preceding stages produce candidate editing pairs, which may still contain reconstruction artifacts, insufficient editing changes, or other quality issues. We therefore design edit-type-specific filtering procedures to remove undesirable samples and improve the overall quality of Alchemy3D-1M. For the retained samples, we further generate editing instructions and image-based conditions for subsequent model training.

Animation. To remove samples with insufficient motion changes, we render eight fixed circular views of both the source and target assets. For each corresponding view, we compute the cosine similarity between their DINOv3 (Siméoni et al., 2025) features:

$$s_v = \cos(f_{\text{DINO}}(R_v(X_s)), f_{\text{DINO}}(R_v(X_t))), \quad v = 1, \ldots, 8. \tag{8}$$

We retain a pair only if

$$\min_{v=1,\ldots,8} s_v < \tau, \tag{9}$$

where τ is a predefined similarity threshold. This filtering removes pairs for which the motion change is insufficiently apparent from all considered viewpoints. For each retained pair, the eight

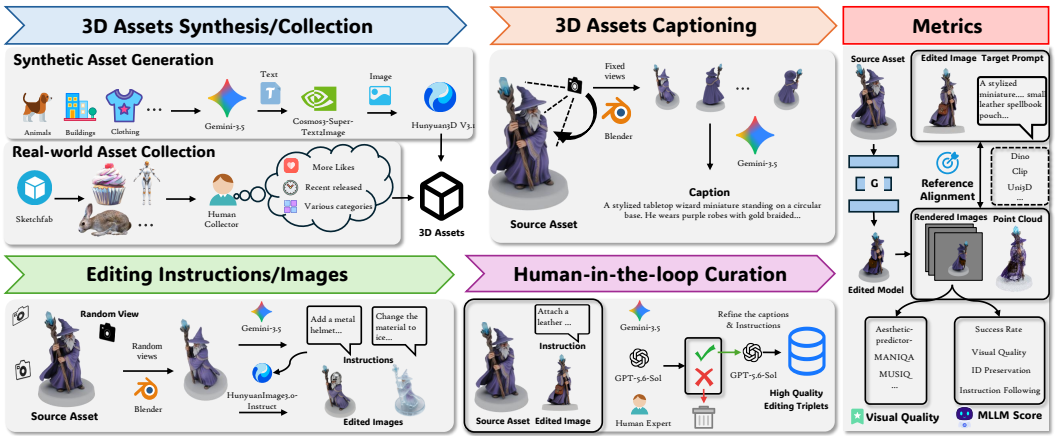

Figure 6: Overview of the construction pipeline and evaluation dimensions of GEdit3D-Bench.

source and target renderings are concatenated into a grid and provided to Qwen3.6-27B (Qwen Team, 2026b) to generate editing instructions describing the motion changes. To increase linguistic diversity, we generate 3–5 instructions for each pair using different phrasings while preserving the same editing semantics. For image-conditioned editing, we additionally render eight random views of the target asset as image conditions.

Segmentation. For segmentation editing, we render eight random views of the part-colored assets using flat colors rather than conventional lighting. This rendering scheme is designed to mimic practical 2D segmentation-map conditions, where the regions predicted by a 2D segmentation model (Ravi et al., 2024; Carion et al., 2025) can be represented as manually assigned flat colors. The resulting multi-view renderings are used as image conditions for segmentation editing.

Other editing types. For addition, removal, replacement, local appearance, and global appearance editing, we render multiple views of each editing pair and concatenate them into grid images for quality filtering and recaptioning with Qwen3.6-27B (Qwen Team, 2026b). For addition, removal, replacement, and local appearance editing, we additionally render 16 random views of each target asset and use Qwen3.6-27B to determine whether the edited region is visible in each view. Views in which the edited region is not sufficiently visible are discarded. For global appearance editing, where the change is not spatially localized, we render eight random views instead.

A.3 GEdit3D-Bench

A.3.1 Limitations of Existing Benchmarks

The quality of a benchmark and its evaluation protocol can strongly influence the development of a research field. A meaningful benchmark should provide a fair, reliable, and realistic assessment of model capabilities while remaining aligned with practical applications. However, existing benchmarks for 3D asset editing still suffer from several fundamental limitations.

Limited Scale and Diversity. Existing benchmarks typically evaluate models on a relatively small number of manually curated assets. For example, Edit3D-Bench (Li et al., 2026b) selects only 100 assets from GSO (Downs et al., 2022) and PartObjectVerse-Tiny (Yang et al., 2024). Similarly, Eval3DEdit (Zhou et al., 2026b) and TANGOEdit (Lim et al., 2026) curate approximately 100 assets from datasets including Objaverse-XL (Deitke et al., 2023) and GSO (Downs et al., 2022). While these benchmarks provide valuable initial evaluations, their limited scale restricts the coverage of asset categories, appearances, geometric structures, and editing scenarios, making it difficult to assess the robustness of 3D editing models in diverse real-world settings.

Low-Quality Ground Truth and In-distribution Evaluation. Another line of work (Ma et al., 2025; Ye et al., 2025a; Xia et al., 2025; Weng et al., 2026) constructs evaluation sets by splitting

the training data of the corresponding method into training and test subsets. Models are then evaluated by measuring the similarity between generated outputs and the provided edited 3D assets. This protocol has several limitations. First, existing 3D editing datasets contain annotation and generation errors, including Alchemy3D-1M (Sec. 3). Treating generated or reconstructed assets as ground truth can therefore pass those artifacts into the score. Second, 3D asset editing is a conditional generation task with multiple valid solutions, so similarity to a single reference is an incomplete measure of editing quality. Third, a random split of the same distribution used for training measures in-distribution fit. A model may score well by matching dataset-specific patterns without generalizing to unseen assets and edits.

We provide visualizations of samples for the Edit3D-Bench proposed by Steer3D in Fig. 11, for Nano3D-100K in Fig. 12, and for 3DEditVerse in Fig. 13.

Insufficient Evaluation Dimensions. Existing evaluation protocols also provide limited coverage of the diverse requirements of 3D asset editing. In particular, measuring similarity to an edited reference alone does not fully characterize whether a model successfully executes the requested modification, preserves the identity and irrelevant attributes of the source asset, or produces a perceptually high-quality result. A comprehensive benchmark should therefore jointly assess multiple aspects of editing quality, including editing success, instruction following, source identity preservation, visual quality, and semantic alignment with the desired edit.

A.3.2 Construction Pipeline

Each sample in GEdit3D-Bench consists of a source 3D asset, a rendered source image, a natural-language editing instruction, and a corresponding target edited image, along with the text captions for source and target assets. Given a source asset $\mathcal{A}$ and an editing instruction $\mathcal{I}$ or edited image $\mathcal{I}_{\text{tgt}}$, a 3D editing model is expected to generate an edited asset $\mathcal{A}'$ that satisfies the requested modification while preserving the identity and irrelevant attributes of the source asset. The construction pipeline consists of four stages.

3D Asset Synthesis and Collection. We first define 21 high-level asset categories covering a broad range of semantic domains, including characters, animals, vehicles, furniture, architecture, and daily objects. For each category, Gemini-3.5-Flash[3] generates diverse asset descriptions, which are subsequently converted into high-quality images using Cosmos3-Super-Text2Image (Agarwal et al., 2026). These images are then transformed into 3D assets using Hunyuan3D V3.1. To further improve the diversity and realism of the benchmark, we additionally collect real-world 3D assets from Sketchfab. During collection, we consider semantic categories, user engagement, and release timestamps. By prioritizing recently released assets, we reduce potential overlap with existing large-scale 3D generation datasets, such as Objaverse-XL (Deitke et al., 2023) and TexVerse (Zhang et al., 2025). Combining generated and real-world assets enables GEdit3D-Bench to cover a broad spectrum of use cases, object categories, visual styles, and geometric structures.

3D Asset Captioning. For each collected asset, we render multiple predefined viewpoints using Blender and concatenate them into multi-view image grids. These rendered views are provided to Gemini-3.5-Flash to generate detailed semantic descriptions of the assets. The resulting captions describe object categories, appearances, materials, structures, and distinctive characteristics, and serve as semantic references for subsequent instruction generation.

Editing Instruction and Target Image Generation. Given each source asset, we randomly sample rendered viewpoints and provide them together with the source captions to Gemini-3.5-Flash. Using category-specific prompting templates, Gemini-3.5-Flash generates diverse editing instructions covering addition, removal, replacement, local and global appearance modification, and animation. For each instruction, HunyuanImage3.0-Instruct (Cao et al., 2025) performs image-level editing on the rendered source views to generate corresponding target images. These edited images provide visual references for evaluating whether a 3D editing model correctly interprets and executes the requested modification.

[3] https://deepmind.google/models/model-cards/gemini-3-5-flash/

Multi-stage Quality Curation. To ensure the reliability of benchmark samples, each editing triplet, consisting of a source rendering, editing instruction, and edited image, undergoes multi-stage quality verification. Specifically, we employ Gemini-3.5-Flash, GPT-5.6-Sol, and human experts to filter samples with incorrect semantics, unrealistic modifications, inconsistent object identities, or low-quality editing results. After quality control, we randomly sample the remaining high-quality triplets to form the final benchmark set. Finally, source captions and editing instructions are provided to GPT-5.6-Sol to generate target captions describing the expected edited assets.

A.3.3 Benchmark Statistics

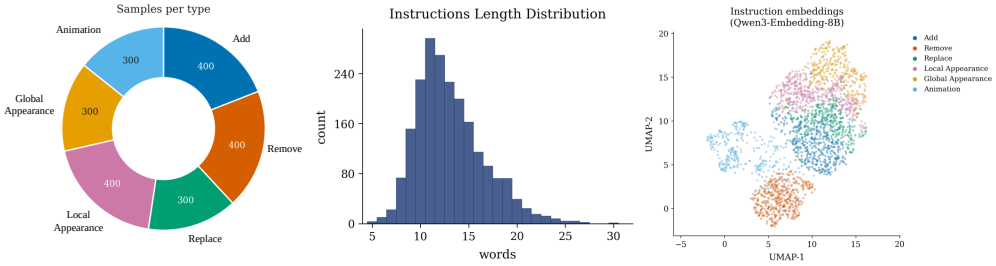

Figure 7: Statistics of GEdit3D-Bench. The left panel shows the number of samples per editing type, the middle panel shows the distribution of instruction length, and the right panel visualizes instruction embeddings from Qwen3-Embedding-8B.

As shown in Fig. 7, GEdit3D-Bench contains 400 samples for addition, removal, and local appearance, and 300 samples for replacement, animation, and global appearance. The editing instructions are about a dozen words long, as in our training set, and describe short atomic edits. We embed these instructions with Qwen3-Embedding-8B[4] and visualize them with UMAP in the right panel of Fig. 7. The embedding distribution indicates that the instructions cover diverse edits.

A.3.4 Evaluation Protocol Details

View Quality Assessment. To assess the perceptual quality of edited assets, we render multiple views and compute image-quality scores for each view. Specifically, we employ the aesthetic-predictor-v2.5[5] and established image quality assessment models, including MANIQA (Yang et al., 2022) and MUSIQ (Ke et al., 2021), implemented through pyiqa (Chen & Mo, 2022). Scores are averaged across the rendered views to obtain an overall view-quality assessment.

Reference Alignment. Pretrained encoders are widely used to evaluate semantic alignment between generated results and desired targets. We consider image alignment A_I and text alignment A_T, instantiated with EVA-CLIP (Sun et al., 2023), SigLIP (Zhai et al., 2023), BLIP (Li et al., 2022), DINOv3 (Siméoni et al., 2025), and Uni3D (Zhou et al., 2024), where applicable.

MLLM-based Evaluation. Recent benchmarks for video and image generation and editing (Zhou et al., 2026a; Liang et al., 2026; Ye et al., 2025b; Huang et al., 2024) have increasingly adopted modern multimodal large language models (MLLMs) for evaluation due to their strong visual understanding and reasoning capabilities. Following this direction, we instruct MLLMs to evaluate each editing result along four complementary dimensions: **Success Rate (SR)**, **Instruction Following (IF)**, **Identity Preservation (IP)**, and **Visual Quality (VQ)**. SR is a binary judgment of whether the requested edit is successfully achieved, while IF, IP, and VQ are scored on a 1–100 scale. Specifically, IF measures how faithfully the result follows the editing instruction, IP measures whether the identity and irrelevant properties of the source asset are preserved, and VQ measures the perceptual quality and plausibility of the resulting asset. SR and VQ use a shared system prompt. IF and IP use an edit-type-specific prompt, because the editing types differ substantially in granularity.

[4] https://huggingface.co/Qwen/Qwen3-Embedding-8B
[5] https://github.com/discus0434/aesthetic-predictor-v2-5

A.4 More Evaluation Details and Results

A.4.1 Details of Evaluation Setting

Main experiments. Unless otherwise stated, the CLIP model used in evaluation is EVA-CLIP-18B[6], the SigLIP model is siglip2-giant[7], the DINO model is DINOv3 ViT-L[8], and the Uni3D model is uni3d-giant[9]. For reference alignment, each asset is rendered from ten views with camera radius 1.8 and a $49°$ field of view. Eight views use a pitch of $10°$ and yaw angles spaced by $45°$. The other two are front views, with yaw fixed to the front and pitch set to $+30°$ and $-30°$. MLLM scores use the same random 400-sample subset of GEdit3D-Bench as the user study.

Instruction-driven editing by Alchemy3D. Noisy instruction captions make Alchemy3D-Instruct weaker than image-conditioned Alchemy3D, a gap also observed in text-conditioned 3D generation (Xiang et al., 2025). We therefore design a complementary pipeline that uses image-conditioned Alchemy3D for instruction-driven editing. Given a source asset and an editing instruction, we render 16 random views and ask a VLM to select the view best suited to the requested edit. A 2D image editor modifies that view, after which Alchemy3D edits the source asset using the modified image as its reference. We use Qwen3.8-27B[10] for view selection and FLUX.2-Klein-9B[11] for image editing.

User Study. To evaluate the methods in a realistic setting, we conducted a user study with 24 participants from nine institutions. We used the same subset as for MLLM scoring and randomly assigned 50 tasks to each participant. For each task, the source asset and outputs from all methods were loaded into Google Model Viewer[12], allowing participants to inspect each asset interactively. Participants selected the best result or chose "Hard to Select", which we counted as "Cannot Decide". An example page is shown in Fig. 8.

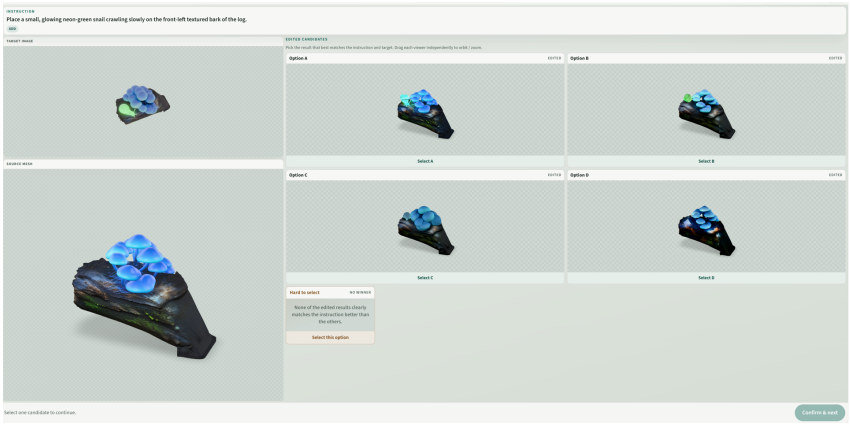

Figure 8: Example page from the user study.

A.5 More Experimental Results

Fig. 9 reports user-selection rates across all six editing types. Alchemy3D receives the highest preference for all editing types: addition (69.2%), removal (71.8%), replacement (73.1%), animation (78.3%), local appearance (44.3%), and global appearance (39.8%). The "Cannot Decide" rate

[6] https://huggingface.co/BAAI/EVA-CLIP-18B
[7] https://huggingface.co/google/siglip2-giant-opt-patch16-384
[8] https://huggingface.co/facebook/dinov3-vitl16-pretrain-lvd1689m
[9] https://huggingface.co/BAAI/Uni3D/tree/main/modelzoo/uni3d-g
[10] https://huggingface.co/Qwen/Qwen3.8-27B
[11] https://huggingface.co/black-forest-labs/FLUX.2-klein-9B
[12] https://github.com/google/model-viewer

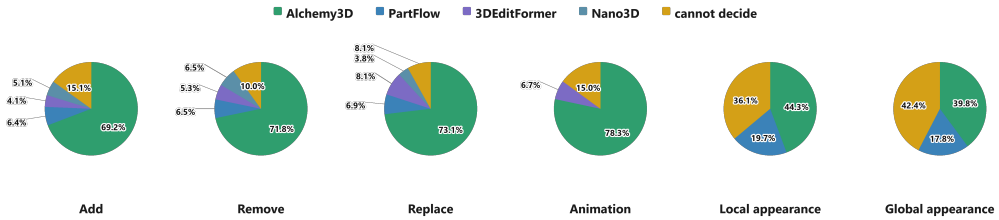

Figure 9: Results of the user study.

Table 9: Quantitative comparison on GEdit3D-Bench. Best results are in **bold** and second-best are underlined.

Method	View Quality			Ref. Alignment						MLLM			
	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_T^{CLIP} ↑	A_I^{SigLIP} ↑	A_I^{DINO} ↑	A_I^{Uni3D} ↑	A_T^{Uni3D} ↑	SR ↑	VQ ↑	IF ↑	IP ↑
add													
Nano3D	4.40	0.552	73.04	80.11	65.34	86.96	66.70	36.48	80.70	56.2%	70.1	55.3	72.5
3DEditFormer	4.36	0.537	70.91	79.98	63.44	86.17	65.01	36.06	79.55	53.8%	63.8	56.1	59.4
PartFlow	4.32	0.543	70.98	79.89	62.74	86.50	65.20	36.21	79.34	57.5%	63.4	56.5	65.8
Alchemy3D-Turbo	**4.59**	0.541	**73.49**	82.01	**69.66**	88.59	70.80	36.24	80.32	85.0%	68.6	68.3	71.8
Alchemy3D	4.47	**0.578**	72.94	**84.84**	68.66	**90.28**	**71.63**	**36.87**	**81.31**	**86.2%**	**72.2**	**69.8**	**76.3**
remove													
Nano3D	4.39	0.553	**72.44**	82.53	64.12	89.28	70.39	**37.32**	81.05	41.2%	70.3	48.8	70.5
3DEditFormer	4.33	0.548	70.69	83.27	61.80	89.20	69.93	37.06	79.63	52.5%	68.3	58.0	65.9
PartFlow	4.30	0.550	70.73	82.58	61.05	89.08	69.25	36.97	79.44	50.0%	66.1	53.5	65.1
Alchemy3D-Turbo	**4.56**	0.535	72.26	82.19	**66.77**	89.54	**74.28**	36.66	80.61	**90.0%**	70.7	71.7	73.9
Alchemy3D	4.41	**0.572**	71.64	**84.85**	65.93	**91.23**	73.67	37.15	**81.24**	**90.0%**	**73.0**	**72.8**	**75.6**
replace													
Nano3D	4.41	0.557	**73.20**	82.29	66.16	88.28	70.75	**37.69**	82.91	57.6%	70.0	58.7	73.7
3DEditFormer	4.34	0.545	71.06	82.38	64.77	88.19	69.66	37.68	82.83	74.6%	67.0	67.2	65.4
PartFlow	4.32	0.548	71.23	81.29	63.28	88.14	69.28	37.32	81.32	60.3%	66.7	59.6	69.7
Alchemy3D-Turbo	**4.58**	0.540	73.11	83.05	**68.91**	89.62	73.80	37.13	82.36	90.0%	70.6	**72.6**	74.2
Alchemy3D	4.43	**0.578**	72.53	**85.10**	68.22	**90.67**	**74.60**	37.41	**82.99**	**93.3%**	**73.2**	71.5	**76.8**
local_appearance													
PartFlow	4.32	0.552	70.53	81.67	61.36	89.21	71.31	**36.76**	79.42	38.8%	69.7	45.2	68.5
Alchemy3D-Turbo	**4.60**	0.541	**72.93**	81.30	**66.73**	89.30	**73.60**	36.45	80.43	**61.3%**	70.0	**57.4**	75.0
Alchemy3D	4.44	**0.579**	71.97	**83.45**	65.56	**90.44**	72.78	36.64	**80.88**	54.4%	**72.4**	53.4	**76.0**
global_appearance													
PartFlow	4.36	0.542	70.14	72.04	55.31	82.01	64.38	34.20	73.50	8.3%	68.7	36.1	75.0
Alchemy3D-Turbo	**4.56**	0.502	**71.20**	74.57	**63.17**	83.62	67.09	**34.51**	**75.35**	**50.0%**	68.3	**53.2**	76.8
Alchemy3D	4.45	**0.565**	71.05	**75.72**	60.79	**84.46**	**67.82**	34.36	74.99	28.3%	**72.7**	43.7	**77.7**
animation													
3DEditFormer	4.19	0.535	70.59	83.83	65.88	88.91	70.60	36.79	79.38	49.2%	63.9	54.9	61.6
Alchemy3D-Turbo	**4.42**	0.533	**72.78**	83.62	**69.97**	89.96	73.88	36.44	80.59	**68.3%**	64.8	65.1	69.2
Alchemy3D	4.31	**0.567**	72.34	**86.41**	69.29	**91.44**	**75.23**	**37.31**	**82.02**	66.7%	**70.8**	**66.6**	**72.8**

increases from 8.1–15.1% for structural and animation edits to 36.1% and 42.4% for local and global appearance edits, respectively, indicating greater ambiguity in evaluating appearance changes.

A.5.1 Image-conditioned Editing

We first report complete image-conditioned results on Eval3DEdit and GEdit3D-Bench, followed by evaluations on 3DEditVerse, the Edit3D-Bench of VoxHammer, and Nano3D-100K. Nano3D supports only addition, removal, and replacement. 3DEditFormer does not support appearance edits, and PartFlow does not support animation edits. We evaluate each baseline only on the editing types it supports.

On addition, removal, and replacement in Table 9, Alchemy3D and Alchemy3D-Turbo lead in success rate, instruction following, and most alignment scores. Removal is the clearest case: both of our models reach 90% success, against 41.2% for Nano3D and about 50% for 3DEditFormer and PartFlow. Animation follows the same ranking. Alchemy3D-Turbo reaches 68.3% success and Alchemy3D 66.7%, against 49.2% for 3DEditFormer, and Alchemy3D is also highest on image alignment and identity preservation.

Appearance edits remain weaker than structural edits. PartFlow reaches 38.8% success on local appearance and 8.3% on global appearance. Alchemy3D-Turbo reaches 61.3% and 50.0%, and Alchemy3D reaches 54.4% and 28.3%, while both preserve identity better than PartFlow. We at-

Table 10: Quantitative comparison on Eval3DEdit. Best results are in **bold** and second-best are underlined.

	View Quality			Ref. Alignment						MLLM Score			
Method	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_T^{CLIP} ↑	A_I^{SigLIP} ↑	A_I^{DINO} ↑	A_I^{Uni3D} ↑	A_T^{Uni3D} ↑	SR ↑	VQ ↑	IF ↑	IP ↑
add													
Nano3D	4.65	0.547	72.09	80.58	46.70	87.89	70.46	37.94	61.59	27.3%	68.9	38.3	66.9
3DEditFormer	4.57	0.523	72.05	79.49	45.56	86.72	67.06	37.29	59.88	45.5%	66.2	45.8	62.3
PartFlow	4.60	0.549	72.25	80.63	48.47	87.37	68.44	36.72	60.96	62.5%	63.0	55.0	69.9
Alchemy3D-Turbo	**4.75**	0.537	**74.80**	81.99	53.05	88.93	74.61	38.80	66.20	80.0%	68.4	**64.5**	**77.0**
Alchemy3D	4.68	**0.559**	74.08	**83.21**	**53.06**	**89.70**	**74.89**	**38.86**	**66.28**	**81.8%**	**74.1**	63.7	72.5
remove													
Nano3D	**4.65**	0.543	72.37	**84.27**	**49.12**	90.48	75.47	40.90	**62.51**	20.0%	66.2	37.8	74.6
3DEditFormer	4.57	0.518	71.46	82.94	47.98	89.35	73.31	39.99	58.14	70.0%	65.0	60.7	67.8
PartFlow	4.49	0.531	71.45	82.45	47.10	89.05	70.25	39.91	58.71	80.0%	62.9	66.2	69.9
Alchemy3D-Turbo	4.64	0.554	**74.40**	83.09	47.09	89.93	76.70	**41.06**	60.66	**90.0%**	**69.5**	70.6	**77.1**
Alchemy3D	4.59	**0.560**	72.73	84.05	46.69	**90.94**	**78.12**	40.95	59.61	80.0%	69.3	**71.0**	72.0
replace													
Nano3D	4.51	0.542	72.20	82.06	52.64	88.20	73.65	38.73	67.49	77.8%	69.0	64.3	74.8
3DEditFormer	4.50	0.535	71.93	82.90	54.18	88.49	72.89	38.66	66.76	**100.0%**	65.1	**74.0**	73.6
PartFlow	4.43	0.542	72.44	81.70	52.37	88.28	72.77	38.77	68.68	60.0%	67.1	58.7	71.9
Alchemy3D-Turbo	4.54	0.536	**73.94**	82.95	**57.54**	88.84	74.72	39.62	**71.84**	**100.0%**	**71.2**	73.2	**76.4**
Alchemy3D	**4.54**	**0.553**	73.46	**84.04**	56.56	**89.75**	**77.63**	**39.96**	71.72	90.0%	67.0	69.5	73.5
style													
3DEditFormer	4.61	0.507	70.84	81.05	51.72	88.57	67.24	**39.17**	66.35	90.0%	67.2	–	–
PartFlow	4.58	0.532	71.63	78.10	50.68	86.29	64.58	37.17	65.70	40.0%	61.7	–	–
Alchemy3D-Turbo	4.61	0.502	71.32	81.15	52.51	87.74	71.97	38.23	65.78	**100.0%**	65.1	–	–
Alchemy3D	**4.64**	**0.552**	**72.53**	**82.40**	**54.41**	**89.89**	**73.34**	38.16	**67.47**	90.0%	**71.2**	–	–
action													
3DEditFormer	4.06	0.509	70.32	75.80	44.45	83.56	65.69	32.86	54.34	20.0%	55.2	40.0	57.5
Alchemy3D-Turbo	**4.33**	0.513	**73.58**	79.96	**51.46**	86.85	73.76	36.85	63.89	**90.0%**	64.4	**70.5**	69.5
Alchemy3D	4.29	**0.537**	72.83	**80.84**	51.15	**88.09**	**76.49**	**37.01**	**66.34**	80.0%	**70.8**	**70.5**	**75.6**

Table 11: Image-conditioned comparison on 3DEditVerse. Shaded columns measure alignment with the provided reconstructed target and are not treated as primary metrics.

	View Quality			Ref. Alignment				Target Alignment					
Method	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_I^{SigLIP} ↑	A_I^{DINO} ↑	A_I^{Uni3D} ↑	LPIPS ↓	SSIM ↑	PSNR ↑	CD ↓	F1 ↑	NC ↑
3DEditFormer	4.191	0.533	68.96	69.18	79.18	64.60	36.76	0.0797	0.9112	**23.81**	**13.36**	**72.18**	**0.852**
PartFlow	4.145	0.537	69.08	68.08	78.60	63.74	36.46	**0.0786**	**0.9148**	23.75	17.65	66.92	0.844
Alchemy3D-Turbo	**4.239**	0.524	**71.43**	72.10	82.05	69.44	36.36	0.1016	0.8967	20.65	22.71	50.28	0.773
Alchemy3D	4.213	**0.558**	70.78	**73.10**	**82.18**	**72.20**	**37.41**	0.1000	0.8961	22.06	24.48	44.15	0.757

tribute the remaining gap to task conflict under joint training: most training pairs modify both geometry and appearance, so appearance-only edits conflict with the majority of the training distribution (Sec. A.6).

Table 10 separates Eval3DEdit by editing type. On addition, Alchemy3D reaches 81.8% success, against 27.3% for Nano3D, 45.5% for 3DEditFormer, and 62.5% for PartFlow. On removal, Nano3D keeps a high CLIP score but only 20% success, so similarity to the source is not the same as completing the edit. Replacement is the closest comparison: 3DEditFormer and Alchemy3D-Turbo both reach 100% success, while Alchemy3D is slightly lower at 90% and remains stronger on image alignment. On style, PartFlow falls to 40% success, whereas Alchemy3D-Turbo reaches 100%. On action, 3DEditFormer succeeds on 20% of cases, against 90% for Alchemy3D-Turbo and 80% for Alchemy3D.

We next evaluate on 3DEditVerse (Xia et al., 2025), the Edit3D-Bench of VoxHammer (Li et al., 2026b), and Nano3D-100K (Ye et al., 2025a). Nano3D does not release an official test list, so we randomly sample 1.5K pairs. View quality and reference alignment follow Sec. 5 and do not use a reconstructed 3D target. For completeness, we also report *target alignment*: perceptual similarity to rendered views of the provided target asset through LPIPS (Zhang et al., 2018), SSIM (Wang et al., 2004), and PSNR, and geometric similarity to its mesh through Chamfer distance (Fan et al., 2017), F-score (Knapitsch et al., 2017), and normal consistency (Gkioxari et al., 2019). These columns are shaded. As discussed in Sec. 5, a reconstructed asset is not an independent reference, and similarity to one such asset ignores the one-to-many nature of editing. Alchemy3D performs better on the unshaded metrics but worse on the shaded metrics, consistent with prior methods being trained or selected against the same reconstructed targets.

Table 12: Image-conditioned comparison on Edit3D-Bench (VoxHammer). Target-alignment metrics are not available for this benchmark.

	View Quality			Ref. Alignment			
Method	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_I^{SigLIP} ↑	A_I^{DINO} ↑	A_I^{Uni3D} ↑
3DEditFormer	4.099	0.553	73.72	76.91	84.99	66.90	38.33
PartFlow	4.112	0.568	74.25	77.55	85.95	69.02	38.93
Alchemy3D-Turbo	4.124	0.553	73.48	79.90	87.28	72.91	38.25
Alchemy3D	**4.207**	**0.580**	**74.61**	**80.91**	**88.44**	**75.04**	**39.06**

Table 13: Image-conditioned comparison on Nano3D-100K. Shaded columns measure alignment with the provided reconstructed target and are not treated as primary metrics.

	View Quality			Ref. Alignment				Target Alignment					
Method	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_I^{CLIP} ↑	A_I^{SigLIP} ↑	A_I^{DINO} ↑	A_I^{Uni3D} ↑	LPIPS ↓	SSIM ↑	PSNR ↑	CD ↓	F1 ↑	NC ↑
3DEditFormer	4.228	0.500	71.27	66.30	78.06	60.38	36.57	0.1084	0.8740	19.92	**15.21**	**69.88**	0.814
PartFlow	4.192	0.509	71.93	65.42	78.61	61.18	36.20	**0.0970**	**0.8887**	**20.79**	17.58	69.43	**0.823**
Alchemy3D-Turbo	4.266	0.487	71.24	69.86	80.81	66.58	37.04	0.1296	0.8603	18.79	24.99	49.81	0.726
Alchemy3D	**4.304**	**0.539**	**73.27**	**72.22**	**82.37**	**71.11**	**38.06**	0.1327	0.8572	18.40	29.12	42.00	0.699

Across Tables 11, 12, and 13, Alchemy3D ranks first and Alchemy3D-Turbo usually second on the unshaded metrics.

Instruction-conditioned editing. Table 14 separates direct instruction editing from the agentic pipeline in Sec. A.4.1. Alchemy3D-Instruct outperforms Steer3D on every editing type that Steer3D supports. The margin is largest on addition, where Steer3D has zero success, and on identity preservation, where Steer3D remains near 30–50 while Alchemy3D-Instruct remains above 65. The agentic variant edits one selected view and then applies image-conditioned Alchemy3D, which further raises success: 75.0% versus 57.5% on addition, 87.1% versus 78.1% on replacement, 52.5% versus 30.0% on local appearance, 50.0% versus 6.7% on global appearance, and 70.0% versus 46.7% on animation. Global appearance remains the weakest setting for the instruction model, consistent with noise in the recaptioned instructions and with the task conflict in Sec. A.6. Identity preservation is sometimes higher for Alchemy3D-Instruct than for the agentic pipeline, because the intermediate 2D edit changes the asset more strongly.

Multi-turn Editing. Fig. 10 illustrates multi-turn, long-horizon editing with Alchemy3D. A language model (Qwen3.8-27B) decomposes a complex instruction into short atomic edits. The agentic pipeline in Sec. A.4, which selects a view, edits that image, and then edits the 3D asset, is applied recursively to each atomic instruction.

A.6 Limitations and Future Work

Imperfect verification. Although Alchemy3D-1M (Sec. 3) is a million-scale 3D asset editing dataset, and training Alchemy3D on it supports its utility, several limitations remain. As with prior datasets in image, video, and 3D asset editing (Yu et al., 2025; Liang et al., 2026; Ye et al., 2025a; Xia et al., 2025; Weng et al., 2026) that verify and caption data automatically with vision-language models, our dataset may contain low-quality samples because current VLMs are imperfect. For example, they can struggle to distinguish the left and right sides of an asset, and may accept low-quality samples as valid training pairs.

Task conflict. In this work, we follow TRELLIS.2 (Xiang et al., 2026) in adopting a dense cross-attention transformer architecture. However, during training, we found that the goals and granularities of different editing types are not always mutually beneficial when trained together. For example, local and global appearance changes are learned poorly when trained jointly with a large number of geometry-changing edits. A mixture-of-experts model (Jacobs et al., 1991) may separate editing types that operate at different granularities.

Table 14: Instruction-driven comparison on GEdit3D-Bench. Steer3D does not support replacement or animation edits. † Alchemy3D is evaluated with the agentic view selection and image editing described in Sec. A.4.1.

	View Quality			Ref. Alignment		MLLM			
Method	Aes. ↑	MANIQA ↑	MUSIQ ↑	A_T^{CLIP} ↑	A_I^{Uni3D} ↑	SR ↑	VQ ↑	IF ↑	IP ↑
Add									
Steer3D	3.810	0.522	69.39	46.47	63.38	0.0	47.9	25.7	30.2
Alchemy3D-Instruct	4.602	0.563	74.43	**68.34**	**80.69**	57.5	69.6	57.9	**75.5**
Alchemy3D†	**4.604**	**0.566**	**74.48**	68.33	78.76	**75.0**	**72.4**	**65.6**	69.3
Remove									
Steer3D	3.827	0.523	69.18	46.12	61.84	8.8	53.0	41.7	32.9
Alchemy3D-Instruct	4.537	0.549	72.94	65.15	78.79	**75.0**	66.8	64.9	68.3
Alchemy3D†	**4.590**	**0.557**	**73.11**	**66.34**	**80.69**	72.5	**70.9**	**65.9**	**69.8**
Replace									
Alchemy3D-Instruct	4.443	0.544	**74.04**	65.99	79.28	78.1	67.5	67.1	66.2
Alchemy3D†	**4.564**	**0.564**	73.98	**67.78**	**81.85**	**87.1**	**70.8**	**71.2**	**68.8**
Local Appearance									
Steer3D	3.840	0.528	69.54	50.79	69.64	23.8	55.1	43.1	32.3
Alchemy3D-Instruct	**4.639**	0.558	73.51	65.10	79.75	30.0	71.0	42.0	**72.0**
Alchemy3D†	4.604	**0.563**	**73.52**	**65.72**	**80.03**	**52.5**	**71.2**	**53.2**	68.6
Global Appearance									
Steer3D	3.882	0.523	68.74	49.79	68.78	18.3	53.3	45.3	49.1
Alchemy3D-Instruct	**4.605**	0.538	**72.88**	57.00	71.86	6.7	67.6	33.4	**77.6**
Alchemy3D†	4.580	**0.540**	72.40	**62.23**	**75.22**	**50.0**	**71.2**	**51.8**	75.5
Animation									
Alchemy3D-Instruct	4.431	0.549	73.43	69.20	79.33	46.7	64.4	56.2	65.7
Alchemy3D†	**4.487**	**0.551**	**73.47**	**69.94**	**81.18**	**70.0**	**67.7**	**62.7**	**70.6**

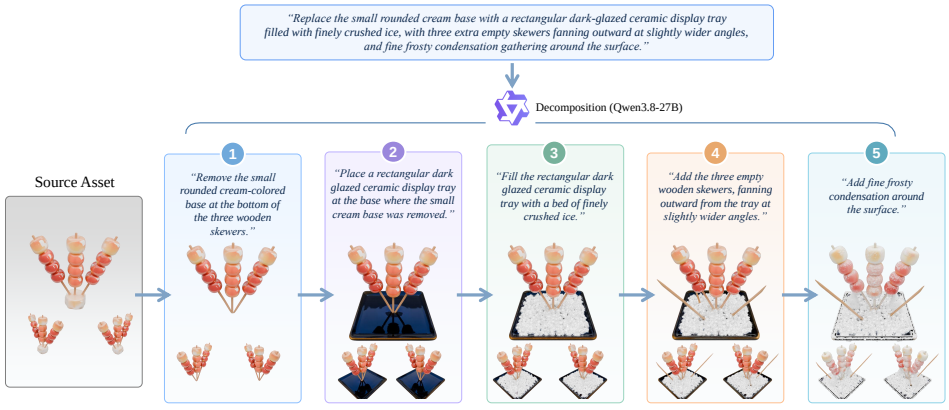

Figure 10: Examples of multi-turn, long-horizon 3D asset editing.

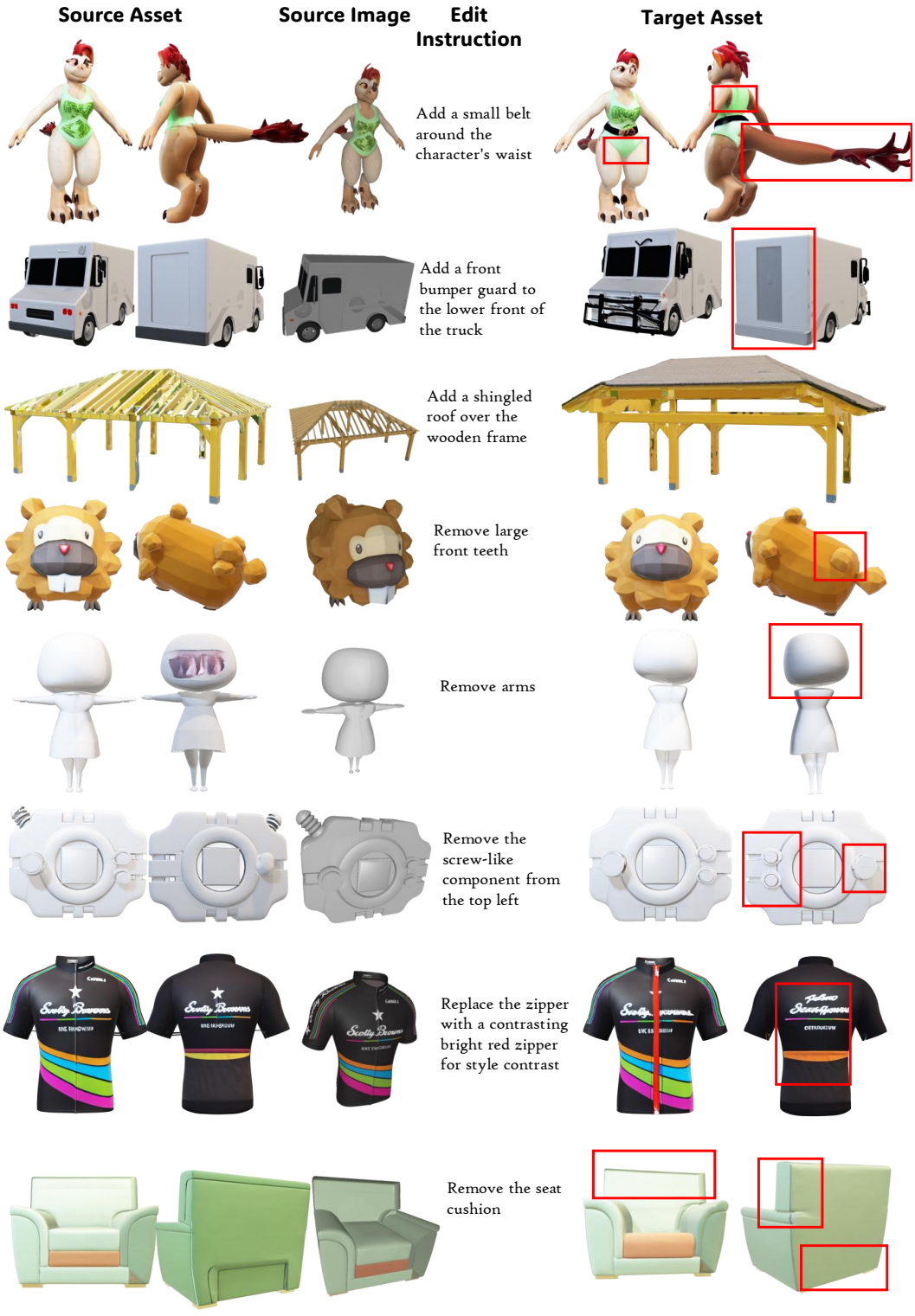

Figure 11: Examples from the Edit3D-Bench of Steer3D.



Figure 12: Examples from Nano3D-100K. Nano3D did not release its test set, so this figure uses the 1.5K-pair subset described in Sec. A.5.1.

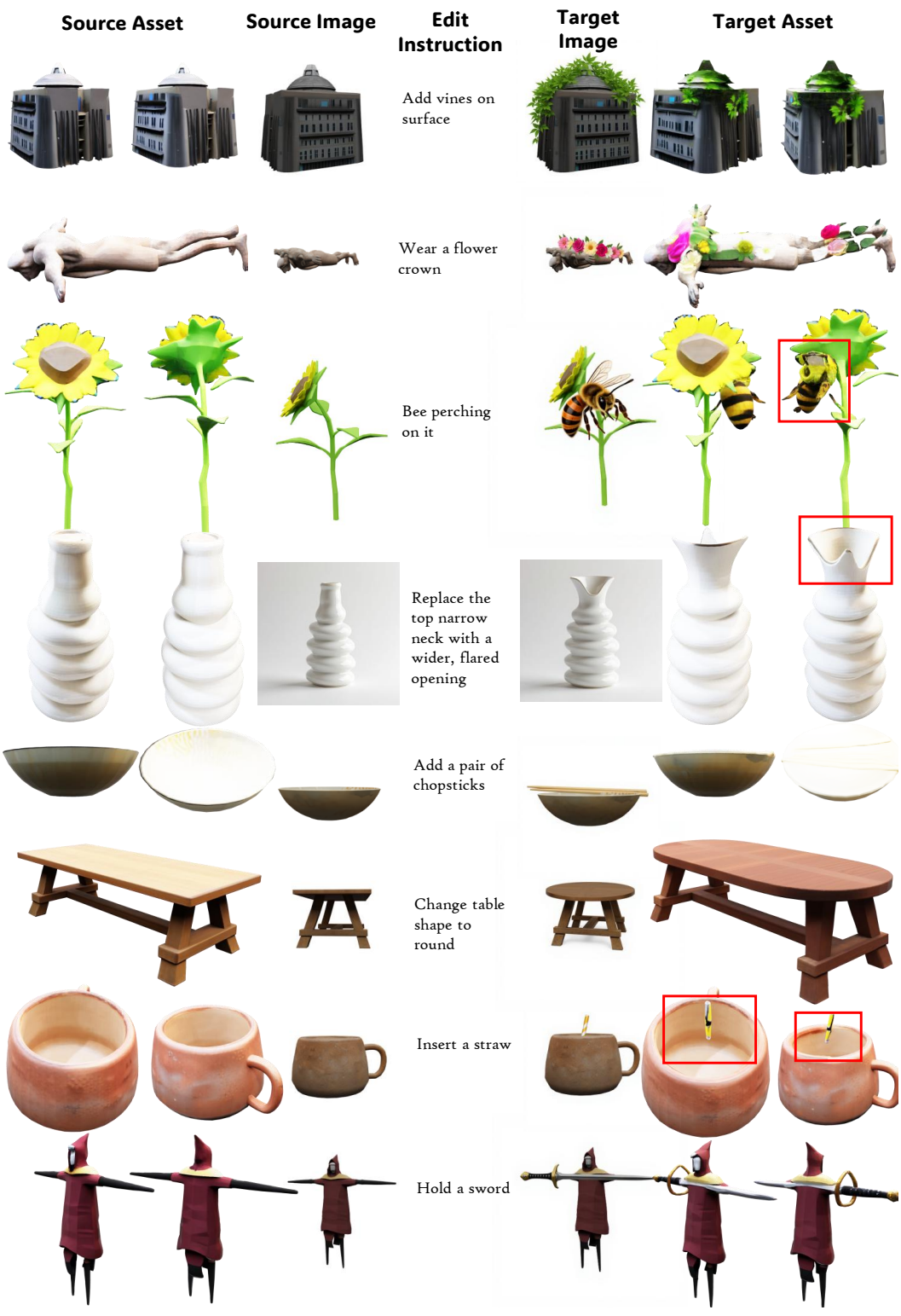

Figure 13: Examples from the 3DEditVerse test set.



Figure 14: Qualitative comparison on addition, removal, and replacement edits.



Figure 15: Qualitative comparison on animation edits.

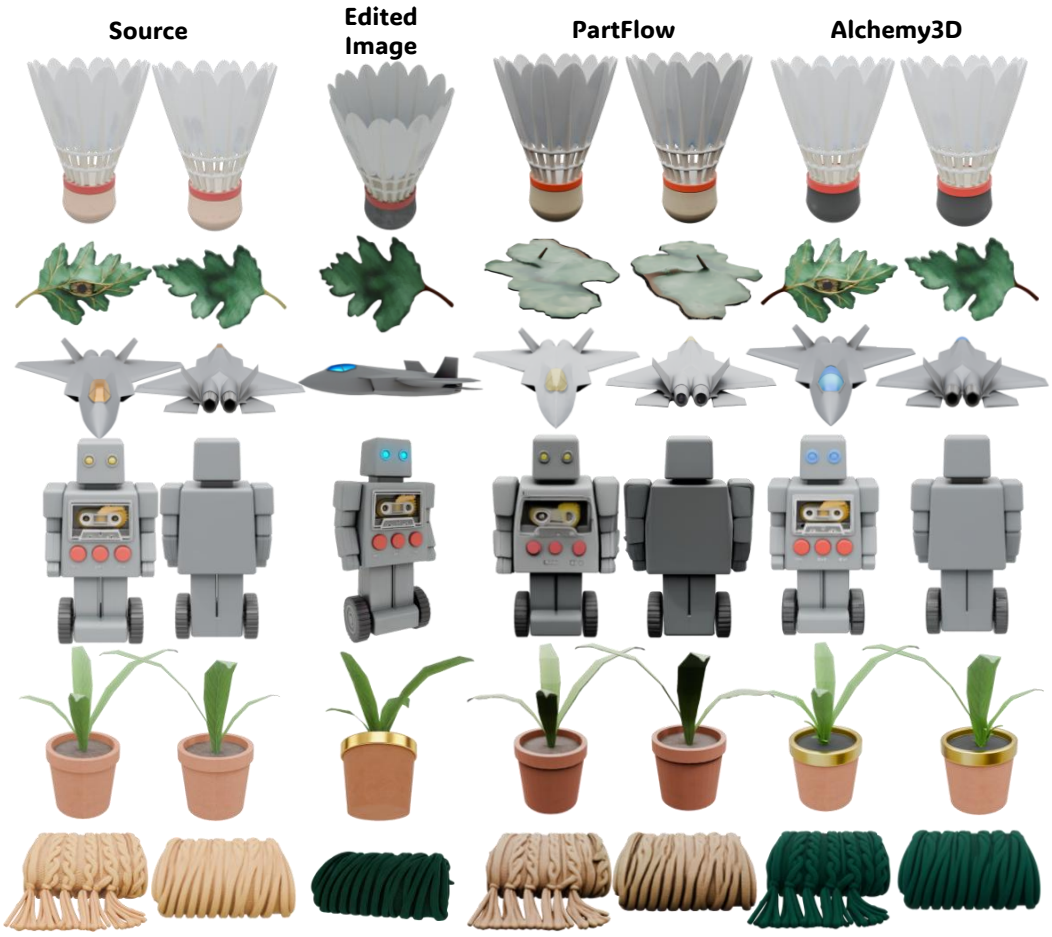

Figure 16: Qualitative comparison on appearance edits.