3DE-AGENT: TOWARD UNIVERSAL INSTRUCTION-DRIVEN AGENTIC 3D EDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Due to the lack of generalizable 3D editing protocols and paired cross-task 3D assets, existing instruction-driven 3D editing methods, based on either explicit representations or implicit models, are typically limited to simple and specific editing tasks, making it difficult to support diverse and complex editing intents through a universal instruction interface. In this paper, we propose **3DE-Agent**, a universal agentic framework to perform various and composite 3D editing tasks. To achieve this goal, we first design and encapsulate off-the-shelf models into tools according to their functional roles in 3D editing, then build a three-stage harness that enables VLM to fully exploit inherent reasoning and tool-use capabilities for understanding and performing versatile and complex editing instructions. In addition, we observe that agentic operation sometimes produces undesired artifacts, such as holes after object removal or blurs after deformation. To mitigate these issues, we curate pseudo paired 3D multi-view data to repurpose Stable Virtual Camera (SEVA) as a general-purpose 3D refiner via hybrid-policy self-distillation, improving both visual quality and cross-view consistency of editing results. Extensive experiments demonstrate the effectiveness and generalization of 3DE-Agent in tackling diverse and complex instructions. Codes and models will be publicly available.

1 INTRODUCTION

Instruction-driven 3D editing provides a user-friendly interface that allows users to modify existing 3D scenes through natural-language instructions while producing multi-view consistent editing results. Common editing tasks include appearance modification (Wang et al., 2023), object removal (Mirzaei et al., 2023), geometric deformation (Bengtson et al., 2026), $etc.$ Existing 3D editing methods can be categorized into two main directions. One line of studies (Chen et al., 2024c; Cheng et al., 2025b;c; Zheng et al., 2025) aims to develop dedicated editing workflows that directly operate on explicit 3D representations, such as NeRFs (Mildenhall et al., 2021) or Gaussians (Kerbl et al., 2023). These workflows are typically guided by pre-trained 2D multimodal models (Brooks et al., 2023; Radford et al., 2021) and perform specific editing operations through iterative optimization (Wang et al., 2022; Haque et al., 2023). For example, InstructN2N (Haque et al., 2023) iteratively updates editing views as supervision, while SPInNeRF (Mirzaei et al., 2023) erases targeted foreground 3D Gaussians to achieve 3D removal. Another line of studies (Li et al., 2025; Qi et al., 2024; Hong et al., 2023; Zhang et al., 2024) attempts to learn editing mappings in an implicit manner, $e.g.$, in 3D latent or multi-view spaces (Chen et al., 2026). Specifically, DGE (Chen et al., 2024a) learns a multi-view editing diffusion model for appearance modification. ShapeEditor (Chen et al., 2024b) guides object-level editing in Shap-E (Jun & Nichol, 2023) latent.

However, the above-mentioned methods support only a limited set of editing tasks and struggle with under-specified and complex editing instructions, failing to provide a universal instruction interface. For explicit manipulation on 3D representations, different editing tasks can differ substantially in both objectives and operation workflows, making it difficult to unify them into a universal predefined workflow. For example, instructions to transfer an object across scenes require that the object should be first extracted and then inserted into the target scene with an appropriate scale. In comparison, learning 3D editing mappings implicitly has the potential to support more diverse and complex instructions, but training such a universal model heavily relies on large-scale paired cross-task 3D editing data. Unfortunately, curating real paired 3D assets is extremely challenging, and existing diffusion models remain limited in understanding complex instructions.

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

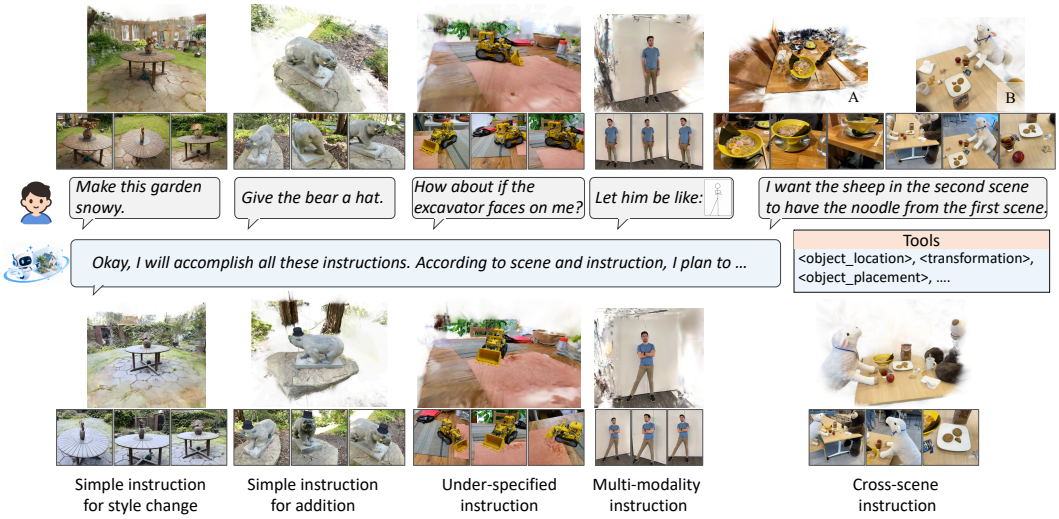


Figure 1: **3DE-Agent applications.** Besides simple tasks that previous methods can achieve, 3DE-Agent can handle instructions involving under-specification, cross-modality, and complex targets.

In this paper, we propose **3DE-Agent**, which formulates universal instruction-driven 3D editing as an agentic tool-composition problem. Instead of encoding all editing behaviors into a monolithic model or manually designing a fixed workflow for each task, we first identify common functional substeps across previous workflows for different 3D editing tasks, and encapsulate off-the-shelf models ($e.g.$, Trellis2 (Xiang et al., 2025), DA3 (Lin et al., 2025), SAM3 (Carion et al., 2025)) and operations ($e.g.$, Rendering, Warping, Reconstruction (Kerbl et al., 2023)) as reusable tools. Then we build a unified three-stage harness around callable tools (Ning et al., 2026b), which requires 3DE-Agent to achieve 3D editing in order of *Perception and Planning*, *Execution*, and *Verification*. In this way, 3DE-Agent can explicitly exploit different tools to interpret instructions, compose different tools, and verify results for tackling different editing tasks while using a unified instruction interface. Fig. 1 demonstrate that 3DE-Agent can tackle diverse and complex instruction.

Furthermore, we observe that 3DE-Agent operating on 3D Gaussians sometimes introduces editing artifacts, such as holes after object removal or blur caused by geometric deformation. No existing models can refine these 3D artifacts consistently. To address this issue, we construct samples with diverse editing artifacts and pair them with the corresponding clean multi-view ground-truth from existing 3D multi-view datasets (Reizenstein et al., 2021; Ling et al., 2024; Xia et al., 2024). These data are used to train a multi-view refinement model, named *Refiner*, in On-Policy Self-Distillation (OPSD (Zhao et al., 2026)) paradigm. Specifically, we initialize Refiner from pretrained Stable Virtual Camera (SEVA) (Zhou et al., 2025) as the student model and use it to generate on-policy multi-view refinement trajectories. Meanwhile, an additional SEVA is frozen as the teacher model and fed with ground-truth multi-view as privileged information, allowing it to offer richer target supervision for Refiner to achieve artifact refinement.

Our main contributions are threefold. First, we propose 3DE-Agent, a 3D editing harness that leverages VLM reasoning and tool-use capability to dynamically compose function-oriented tools for diverse and complex instructions. Second, we introduce an OPSD-based Refiner to correct diverse editing artifacts and improve the consistency of edited multi-view observations. Third, extensive experiments demonstrate the effectiveness and generality of 3DE-Agent, which can tackle diverse and complex instruction with high-quality editing.

2 Related Work

Explicit Editing on 3D Representation. Directly editing 3D representations, such as NeRFs (Mildenhall et al., 2021) and 3D Gaussians (Kerbl et al., 2023), are the primary focus of early 3D editing studies (Wang et al., 2024b; Zhuang et al., 2023). CLIP-NeRF (Wang et al., 2022), NeRF-Art (Wang et al., 2023), Instruct-NeRF2NeRF (Haque et al., 2023), and GaussianEditor (Chen et al., 2024c) typically rely on pre-trained 2D multimodal models (Radford et al., 2021; Brooks et al.,

2023) to provide editing supervision within predefined workflows, thereby guiding the iterative update of the 3D representation (Haque et al., 2023). This paradigm has made considerable progress in 3D removal and object appearance editing (Cao et al., 2024b; Cheng et al., 2025a), as these tasks involve only minor or simple geometric modification. However, most of these works target only specific simple task and cannot handle instructions involving deformation or addition, even less capable of under-specification or complex instructions.

Implicit Editing on 3D Latent. Efforts have been made to learn editing mappings in 3D latent spaces or multi-view spaces. Powerful pre-trained 3D encoders (Jun & Nichol, 2023; Nichol et al., 2022) or visual encoders (Rombach et al., 2022; Zhou et al., 2025) enable direct learning of instruction-conditioned editing mappings in the encoded latent space. V2Edit (Zhang et al., 2025), Pro3D-Editor (Zheng et al., 2025), and DiGA3D (Pan et al., 2025) instantiate this paradigm through render-edit-reconstruct, progressive view propagation, and geometry-aware inpainting, respectively. By learning in structured latent spaces, these methods have the potential to support a broader range of editing tasks (Chen et al., 2026). However, due to the scarcity of 3D editing assets (Deitke et al., 2023) and the limited understanding capabilities of base models, the supported tasks are limited to object-level editing and difficult to scale and generalize.

Harness for 3D Agent. Recently, building effective agent harnesses (Ning et al., 2026a; DeepSeek-AI, 2026) for VLMs to manage reasoning, tool invocation, and execution feedback has attracted increasing attention and has been applied to 2D image (Shen et al., 2026), video generation (Hu et al., 2026; Huang et al., 2026a), and 3D world (Luo et al., 2026; Yang et al., 2026). GenClaw (Ye et al., 2026) uses code as an intermediate canvas and organizes scene conceptualization, structural sketching, and image generation into a staged workflow. WorldClaw (Guo et al., 2026) employs multiple agents to plan scene structures and invoke different tools, constructing large-scale 3D worlds in a coarse-to-fine manner. Programmable World Model (Huang et al., 2026b; Hu et al., 2024) translates natural-language instructions into executable programs to maintain explicit world states and control video models for rendering. Inspired by these work, we bring harness engineering to 3D editing, aiming to support diverse and complex instructions through a unified interface. Concurrent work DesignAgent3d (Liu et al., 2026b) and AGE (Qin et al., 2026) shares similar motivations with ours, but DesignAgent3d focuses on limited appearance editing, while AGE directly exposes existing models as executable skills and lacks an explicit multi-view refinement module to correct tool-induced artifacts and inconsistencies.

On-policy Distillation and Self-distillation. In LLMs, OPD (Li et al., 2026b) supervises student-generated trajectories with a teacher, while OPSD (Zhao et al., 2026) uses the same model with privileged teacher context. In visual generation, OPSD-V (Liu et al., 2026a), D-OPSD (Jiang et al., 2026), and DiffusionOPSD (Zhou et al., 2026a) use self-distillation to improve generation quality or adaptation, while DiffusionOPD (Li et al., 2026a), Flow-OPD (Fang et al., 2026), and DanceOPD (Zhou et al., 2026b) consolidate the capabilities of multiple experts into one model. Different from these works focusing on capability improvement or many-to-one distillation, we use OPSD to re-purpose a generative model (Zhou et al., 2025) as a multi-view refinement model, using clean views as privileged information to guide artifact repair.

3 METHOD

3.1 OVERVIEW OF 3DE-AGENT

To enable 3DE-Agent to handle diverse and complex editing instructions, we build a unified harness and encapsulate a collection of function-oriented tools that can be freely invoked by the VLM. More specifically, we follow GenClaw (Ye et al., 2026) and organize the harness into three stages: *Perception and Planning*, *Execution*, and *Verification*. This design encourages 3DE-Agent to first establish a tool-use plan, then execute the corresponding operations, and finally verify whether the edited scene satisfies the instruction. In addition, we observe that directly exposing existing models such as SAM3 (Carion et al., 2025) and TRELLIS.2 (Xiang et al., 2025) as individual tools often leads to suboptimal intermediate results (Qin et al., 2026). For example, with direct access to SAM3, 3DE-Agent may independently segment the target in each view and project the resulting masks into 3D, which can suffer from cross-view segmentation inconsistencies. We therefore construct tools from a functional perspective by identifying commonly required substeps ($i.e.$, object location) across different 3D editing tasks and encapsulating them into higher-level tools.

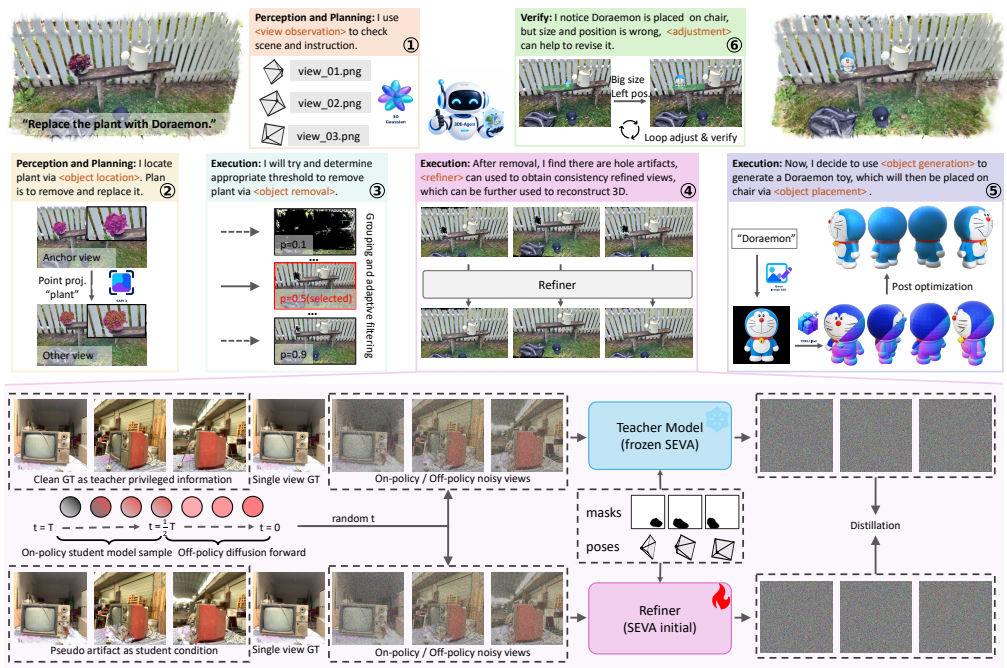


Figure 2: **Overview of 3DE-Agent and Refiner Training.** Top: example of editing trajectory organized into perception and planning, execution, and verification. Encapsulated tools are dynamically invoked by 3DE-Agent. Bottom: Refiner training combines on-policy rollout states with off-policy noised states. A frozen teacher provides supervision using clean views as privileged information.

However, when invoking tools to perform 3D operations, 3DE-Agent often introduces artifacts such as holes after object removal and floating Gaussians after transformation. To correct these artifacts, we collect pseudo artifact data to train a multi-view refinement model named *Refiner*, which is taken as an optional tool and can be invoked by 3DE-Agent whenever refinement is required.

Fig. 2 shows how 3DE-Agent handles a complex instruction. Given the input scene metadata, including rendered observations and registered cameras, 3DE-Agent first interprets the instruction and determines a tool-chain plan. It then executes the planned operations by invoking the corresponding tools, including the Refiner when necessary, and finally verifies the plausibility of the edited scene and whether the editing objective has been satisfied.

3.2 Harness with Function-Oriented Tools

Perception and Planning. 3DE-Agent first jointly interprets the editing instruction and the current 3D scene to determine the editing objective, identifies the content to be modified or preserved, and establishes an initial execution plan. At this stage, 3DE-Agent is allowed to inspect the scene from arbitrary viewpoints together with the corresponding camera poses, or extract a target object from the 3D scene for closer inspection. This stage may involve invoking tools for 3D Gaussian rendering, object segmentation, and localization. Note that this plan is not a hard constraint; 3DE-Agent may revise the tool chain in later stages according to intermediate observations and execution results.

Execution. At this stage, 3DE-Agent follows the plan to invoke editing tools. Since our tools are encapsulated to provide high-level functionality and can return substeps' targets directly, 3DE-Agent does not need to handle the internal implementation details of tools. For example, invoking *object generation* returns a specified 3D object, while *transformation* directly modifies an object's scale or rotation using given parameters. For each tool call, 3DE-Agent inspects the returned result and decides whether to accept it, discard it, or invoke the tool again with adjusted parameters. Meanwhile, the harness stores and provides the history of tool calls and intermediate results as additional context for subsequent decisions.

Verification. After the execution stage, 3DE-Agent verifies the current result by inspecting the rendered multi-view images. It checks whether the editing objective has been achieved and whether

noticeable artifacts, incorrect object interaction, or other visual inconsistencies remain. In most cases, 3DE-Agent invokes additional tools to directly adjust the edited 3D Gaussians. This adjustment may involve multiple tool calls until the result satisfies the user instruction, as the Doraemon adjustment illustrated in Fig. 2. If the verification indicates that the remaining errors cannot be resolved through simple adjustments, 3DE-Agent revises the tool-use plan and restarts either from the beginning or from an intermediate state in the execution stage.

Tools. The tools are encapsulated according to recurring functional substeps summarized from different 3D editing processes, such as *object location*, *object generation*, and *object placement*. We identify several gaps between off-the-shelf models and their intended functional roles in 3D editing. For instance, directly applying the 2D segmentation model SAM3 to 3D object localization can lead to over- or under-segmentation across views; objects generated by TRELLIS.2 may exhibit noticeable color drift that degrades the appearance of the edited scene; and the depth predicted by DA3 cannot be directly used for warping due to resolution and alignment mismatches. We mitigate these issues by incorporating lightweight post-processing and explicit 3D priors into the corresponding tools. Details of the encapsulated tools are provided in Appendix A.1.

Remarks. Recent work uses code to call interfaces such as Blender for 3D generation and layout (Yang et al., 2026). Applying these methods to captured scenes can require the preparation of editable geometry, materials, and lighting. Our 3DE-Agent operates directly on Gaussians, reducing asset preparation while preserving captured appearance and supporting photorealistic rendering.

3.3 Multi-View Refinement via On-Policy Self-Distillation

Refiner. When 3DE-Agent calls tools to manipulate 3D Gaussians, these operations can introduce different artifacts such as holes after object removal and floating Gaussians after deformation. Relying on editing models (Wu et al., 2025a) cannot consistently repair these artifacts across views. To address this issue, we follow Omni-3DEdit (Chen et al., 2026) and take SEVA (Zhou et al., 2025) as the starting point, which is a generative model supporting novel-view synthesis from ground-truth view inputs. We employ OPSD (Zhao et al., 2026) to repurpose SEVA as a multi-view Refiner with artifact view inputs. We curate training data by performing 3DE-Agent on available 3D assets.

OPSD with Hybrid Sampling. We initialize a trainable student model f_S ($i.e.$, the Refiner) and a frozen teacher f_T from pretrained SEVA. In each training sample, let $\mathcal{I}_{\text{pseudo}} = \{I^i_{\text{pseudo}}\}_{i=1}^N$ denote the constructed 3D views containing artifacts and $\mathcal{I}_{\text{gt}} = \{I^i_{\text{gt}}\}_{i=1}^N$ their corresponding clean ground-truth views. The camera poses of views are denoted by $\mathcal{P} = \{P^i\}_{i=0}^N$, which are shared by $\mathcal{I}_{\text{pseudo}}$ and $\mathcal{I}_{\text{gt}}$. $\mathcal{M} = \{M^i\}_{i=1}^N$ is the binary mask indicating the artifact region to be repaired. We take $\mathcal{I}_{\text{gt}}$ as privileged information, building conditions $c_S = (\mathcal{I}_{\text{pseudo}}, I^0_{\text{gt}}, \mathcal{P}, \mathcal{M})$ and $c_T = (\mathcal{I}_{\text{gt}}, I^0_{\text{gt}}, \mathcal{P}, \mathcal{M})$ for the student and teacher models, respectively. Note that I^0_{gt} is additionally provided as a reference view to prevent ill-posed refinements.

To obtain noisy multi-view latent $\hat{z}_t$ at timestep $t \in \{1, \dots, T\}$, the student model performs on-policy rollout ($i.e.$, sampling) starting from z_T via Euler updates (Karras et al., 2022):

$$z^{\text{on}}_T = \sigma_T \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{1}$$

$$z^{\text{on}}_{t-1} = z^{\text{on}}_t + \Delta\sigma_t \frac{z^{\text{on}}_t - f_S(z^{\text{on}}_t, t; c_S)}{\sigma_t}, \tag{2}$$

where ϵ is Gaussian noise, σ_t is the noise level, and $\Delta\sigma_t = \sigma_{t-1} - \sigma_t$. We collect noisy latent $\hat{z}_t$ along the sampling trajectory after stop-gradient with $\hat{z}_t = \text{sg}(z^{\text{on}}_t)$. Both student and teacher models receive the same $\hat{z}_t$ under their respective conditions, and the teacher takes advantage of privileged information to provide target-prediction supervision. We train the student to match the target prediction from the teacher as follows [1]:

$$\mathcal{L}_{\text{OPSD}} = \mathbb{E}\left[\frac{1}{\sigma_t^2} \left\| f_S(\hat{z}_t, t; c_S) - \text{sg}[f_T(\hat{z}_t, t; c_T)] \right\|^2\right]. \tag{3}$$

[1] For simplicity, we omit the teacher classifier-free calculation. Pseudo code is in Appendix A.2.

Although on-policy sampling enables supervision on states actually visited by the student, accumulated rollout errors progressively induce a distribution shift from the teacher, degrading the quality of teacher supervision (Ziheng et al., 2026). Numerically, we perform a pilot study by randomly sampling 500 training examples and collecting their student on-policy trajectories $\hat{z}_t$. The average PSNR of the teacher prediction $f_T(\hat{z}_t, t; c_T)$ with respect to the clean target is reported in Fig. 3. The teacher prediction quality initially improves as denoising proceeds, but starts to decrease after approximately $\frac{1}{2}T$, indicating degraded teacher supervision along long student rollouts.

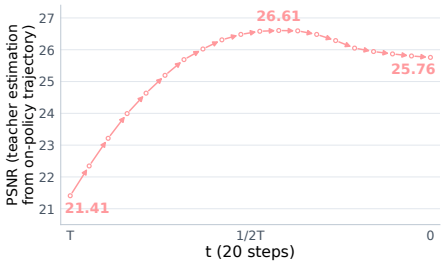


Figure 3: **Student trajectory drift.** The teacher output quality decreases after $\frac{1}{2}T$ along the student trajectory.

We therefore propose a hybrid sampling strategy that retains on-policy states in the early, high-noise steps and uses standard diffusion forward noising to obtain off-policy noisy latents in the later, low-noise steps. Let $z_{\text{gt}}^{\text{off}}$ denote the ground-truth latent obtained by encoding $\mathcal{I}_{\text{gt}}$ with the VAE encoder (Rombach et al., 2022). We replace the purely on-policy input with:

$$\hat{z}_t = \begin{cases} \text{sg}(z_t^{\text{on}}), & \frac{1}{2}T < t \leq T, \\ z_{\text{gt}}^{\text{off}} + \sigma_t \epsilon, & 1 \leq t \leq \frac{1}{2}T. \end{cases} \tag{4}$$

This hybrid sampling strategy mitigates student trajectory drift while retaining on-policy training at high noise levels. Both off-policy and on-policy samples share the same loss in Eq. 3.

Agentic Pseudo-paired Data Curation. We construct training data based on existing 3D multi-view datasets (Reizenstein et al., 2021; Ling et al., 2024; Xia et al., 2024), directly using the original 3D observations as $\mathcal{I}_{\text{gt}}$, and construct a data pipeline to introduce pseudo artifacts into the 3D scenes to obtain $\mathcal{I}_{\text{pseudo}}$ and $\mathcal{M}$. Instead of building data construction pipelines from scratch, we first reconstruct the original scenes as 3D Gaussians and fully leverage 3DE-Agent to simulate artifacts generated by real editing operations. Specifically, we propose two complementary data construction strategies. To simulate missing regions exposed after object removal or movement, we employ 3DE-Agent to identify planar surfaces in the scene and randomly select a local patch on them ($e.g.$, a circular region on the ground). The Gaussians within the selected patch are removed, and the remaining Gaussians are rendered to obtain $\mathcal{I}_{\text{pseudo}}$, while the removed Gaussians are rendered to obtain the corresponding $\mathcal{M}$. To simulate floating Gaussians and blur introduced by object interaction, we use 3DE-Agent to place randomly generated objects at random locations in the scene. By applying random operations such as translation and rotation, the objects are removed by 3DE-Agent. The edited scene is then rendered to obtain $\mathcal{I}_{\text{pseudo}}$, while the Gaussians of the incompletely removed objects are rendered to obtain $\mathcal{M}$. We provide more details in Appendix A.3.

Inference. We use the trained student as the Refiner and expose it as a tool to 3DE-Agent, which invokes the Refiner whenever obvious artifacts are detected. For the Refiner output, we retain only the predictions inside the repair masks, while directly copying the original coarse observations outside the masked regions. The refined multi-view images, together with the registered cameras, are then used to reconstruct the Gaussian scene.

4 Experiments

4.1 Experimental Setup

Benchmarks and Metrics. Since there is not an existing 3D editing benchmark covering diverse editing tasks, we construct a manually curated benchmark containing both simple atomic instructions and complex instructions . We conduct both qualitative and quantitative comparisons with previous methods. In addition to CLIP-D (Radford et al., 2021) and image aesthetic assessment DeQA score (You et al., 2025), we employ Claude Opus 4.6 as an independent VLM-based evaluator to measure editing results in three dimensions. Please refer to Appendix A.3 for the benchmark details and VLM-based evaluation. We additionally evaluate the effectiveness of our Refiner on the existing 3D removal benchmark 360-USID (Wu et al., 2025b) with PSNR and LPIPS as metrics.

Table 1: **Quantitative comparison on our 3D editing benchmark.** We evaluate both simple atomic instructions and complex instructions using CLIP direction similarity, image aesthetic assessment DeQA score, and Claude Opus 4.6 based evaluation. Green cells indicate unsupported editing tasks.

Method	Simple Atomic Instructions												Complex Instructions											
	Add			Remove			Appearance			Deform			Composition			Under-spec.			Cross-scene			Multimodal		
	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑	CLIP ↑	DeQA ↑	Claude ↑
DGE (Chen et al., 2024a)	0.030	3.373	1.72	0.016	2.620	1.67	0.192	3.035	3.17	-0.003	3.339	2.33	0.099	**3.495**	2.00	0.056	3.358	2.83	0.023	3.123	1.75	0.037	3.143	2.00
MVInpainter (Cao et al., 2024b)	0.036	3.227	1.94	0.058	2.603	1.50	0.132	2.616	2.06	0.009	2.991	1.94	**0.195**	2.770	2.17	0.060	2.724	1.75	0.002	**3.701**	2.00	0.030	2.082	1.83
TIP-Editor (Zhuang et al., 2024)	0.049	3.648	2.28	0.037	**2.994**	1.78	0.151	**3.129**	2.50	**0.029**	3.380	2.22	0.191	2.785	2.58	0.027	3.012	2.33	0.013	3.602	1.92	0.088	2.785	2.25
Omni-3DEdit (Chen et al., 2026)	0.079	2.808	1.78	0.031	2.172	1.33	0.077	2.751	2.06	-0.017	2.776	1.67	0.047	2.748	1.42	0.036	2.729	2.08	0.100	2.390	1.83	0.017	2.868	2.33
RL3DEdit (Wang et al., 2026)	0.092	2.903	1.94	0.054	2.359	2.00	0.174	2.628	2.67	0.005	3.028	2.33	0.136	2.975	2.50	**0.082**	2.822	2.67	0.061	2.511	1.75	0.068	2.738	2.50
3DE-Agent (Ours)	**0.219**	**3.788**	**3.17**	**0.062**	2.964	**2.56**	**0.194**	3.055	**3.44**	0.017	**3.504**	**2.56**	0.169	3.148	**3.00**	0.041	**3.595**	**3.17**	**0.138**	3.508	**3.33**	**0.098**	**3.609**	**3.08**

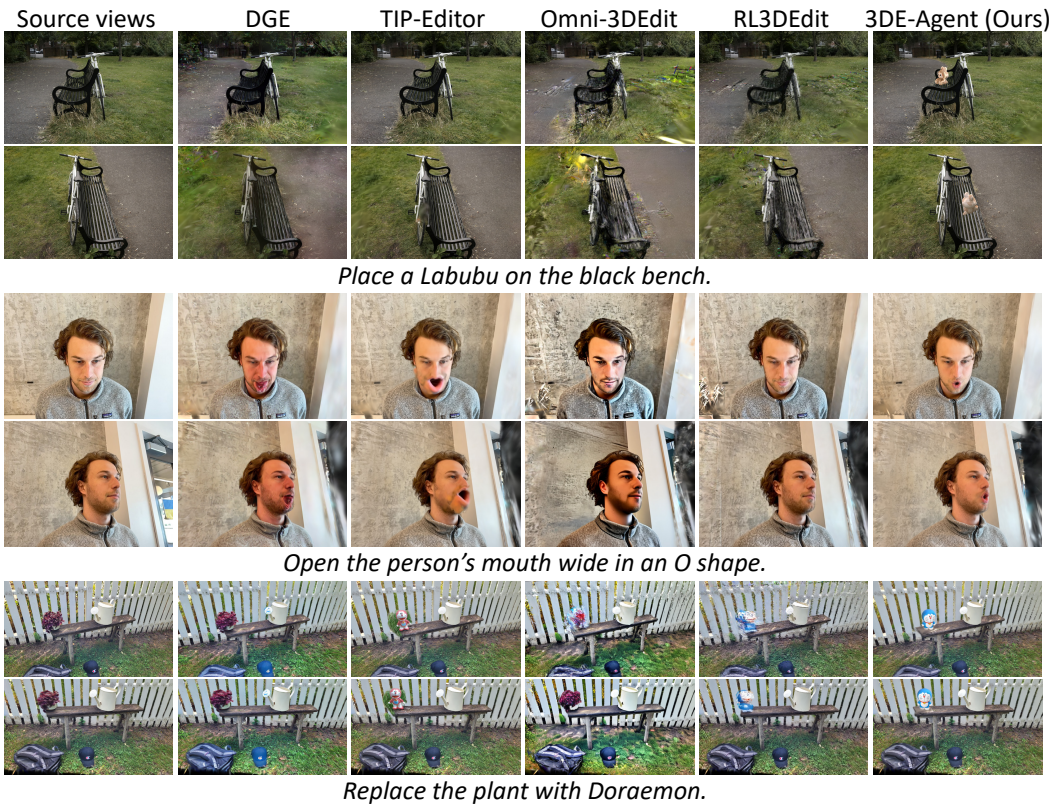


Figure 4: **Qualitative comparisons.** 3DE-Agent successfully accomplishes object addition, deformation, and composition instructions with better visual quality than previous methods.

Implementation Details. We use GPT-5.6-Sol (Singh et al., 2025) and GPT-Image-2 as the VLM and 2D editor of 3DE-Agent. During the Refiner training, we adopt a 20-step sampling schedule and set the hybrid sampling boundary to $\frac{1}{2}T$. We construct 10K pseudo-paired training samples in total, including both missing-region simulation and floating-Gaussians simulation. The Refiner is trained for 6K iterations using AdamW with a batch size of 8. Following Omni-3DEdit (Chen et al., 2026), we adopt the same multi-view input setting and process up to 10 artifact views per sample.

4.2 Comparison with Previous Methods

Quantitative Comparison. In our constructed benchmark, we categorize editing instructions into simple atomic instructions and complex instructions according to their complexity. Simple atomic instructions consist of four basic editing types: object addition, removal, appearance editing, and deformation. Complex instructions further include compositional editing, which requires multiple operations to be executed sequentially, $e.g.$, removing an object followed by modifying 3D scene style. We also consider under-specified, multimodal, and cross-scene instructions, which require a stronger understanding of user intent and scene context.

As shown in Tab. 1, we compare 3DE-Agent with recent 3D editing methods, where green cells indicate instruction types that are inherently unsupported by the corresponding method. For example,

Table 2: **Quantitative comparison (PSNR ↑ / LPIPS ↓) on 360-USID benchmark.**

Methods	Carton	Cone	Cookie	Newcone	Plant	Skateboard	Sunflower	Average
SPIn-NeRF (Mirzaei et al., 2023)	16.66 / 0.539	15.44 / 0.389	11.88 / 0.521	17.13 / 0.519	16.85 / 0.401	15.65 / 0.675	23.54 / 0.206	16.73 / 0.464
LeftRefill (Cao et al., 2024a)	14.67 / 0.560	14.93 / 0.380	11.15 / 0.519	16.26 / 0.448	16.18 / 0.463	14.91 / 0.572	18.85 / 0.331	15.28 / 0.468
Gaussian Grouping (Ye et al., 2024b)	16.70 / 0.502	14.55 / 0.366	11.56 / 0.731	16.75 / 0.533	16.18 / 0.440	16.00 / 0.577	20.79 / 0.209	16.07 / 0.480
GScream (Wang et al., 2024a)	14.69 / 0.587	14.66 / 0.476	12.73 / 0.429	13.66 / 0.605	16.24 / 0.437	12.94 / 0.626	18.47 / 0.436	14.76 / 0.514
Infusion (Liu et al., 2024)	14.19 / 0.555	14.16 / 0.439	12.05 / 0.486	9.56 / 0.624	16.13 / 0.406	13.62 / 0.638	21.20 / 0.238	14.42 / 0.484
AuraFusion360 (Wu et al., 2025b)	17.68 / 0.473	15.63 / 0.332	12.84 / 0.434	17.54 / 0.426	18.00 / **0.322**	**17.01** / 0.559	**24.94 / 0.173**	17.66 / 0.388
Omni-3DEdit (Chen et al., 2026)	**18.58 / 0.452**	16.30 / 0.360	13.18 / 0.465	17.83 / 0.336	17.60 / 0.411	16.76 / 0.521	23.80 / 0.224	17.72 / 0.395
Refiner (Ours)	18.29 / 0.489	**16.73 / 0.250**	**13.58 / 0.313**	**18.03 / 0.302**	**18.34** / 0.406	16.77 / **0.503**	23.42 / 0.201	**17.88 / 0.352**

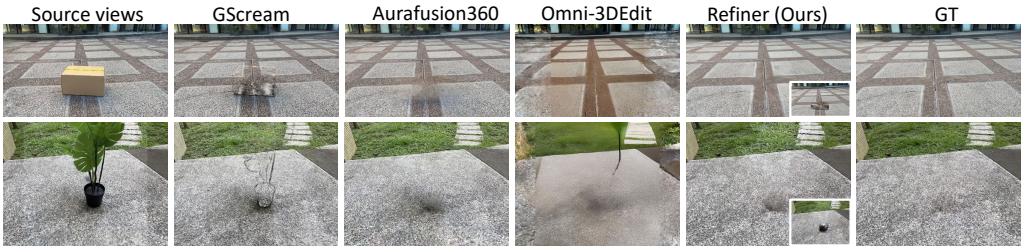


Figure 5: **Qualitative results on 360-USID benchmark.** The bottom-right image in Refiner is its input. Our Refiner outperforms other methods with fine-grained details.

DGE (Chen et al., 2024a) and MVInpainter (Cao et al., 2024b) are largely restricted to specific editing tasks and therefore provide limited coverage over the benchmark. TIP-Editor (Zhuang et al., 2024), RL3DEdit (Wang et al., 2026), and Omni-3DEdit (Chen et al., 2026) exhibit stronger generalization and can handle some compositional instructions through sequential execution. Nevertheless, their performance remains suboptimal. 3DE-Agent outperforms these methods in both simple and complex instructions across visual quality (DeQA) and VLM evaluation (Claude).

Qualitative Comparison. We further provide qualitative comparisons on several editing instructions to demonstrate the advantages of 3DE-Agent, as shown in Fig. 4. For simple tasks such as object addition and deformation, our method still produces satisfactory results under large viewpoint changes, whereas previous methods often suffer from noticeable artifacts or even fail to complete the edit. For the case of *"Replace the plant with Doraemon"*, TIP-Editor (Zhuang et al., 2024) leaves visible holes after removing the original plant, while Omni-3DEdit (Chen et al., 2026) and RL3Dedit (Wang et al., 2026) suffer from semantic ambiguity and inconsistency on Doraemon, respectively. In comparison, 3DE-agent dynamically selects and executes different tool chains according to the editing task, thereby accomplishing the target edit more accurately and cleanly.

Refiner Comparison on 3D Removal. To study the performance of Refiner, we compare it with previous methods focusing on 3D removal on 360-USID benchmark. Tab. 2 reports the quantitative results across seven scenes. Our Refiner outperforms previous methods with best PSNR and LPIPS. Fig. 5 presents the visual comparisons. One can observe that GScream (Wang et al., 2024a) fails to fully remove object Gaussians, Aurafusion360 (Wu et al., 2025b) and Omni-3DEdit (Chen et al., 2026) suffer from blur and appearance drift, respectively. In comparison, Refiner achieves high-quality removal with fine-grained details.

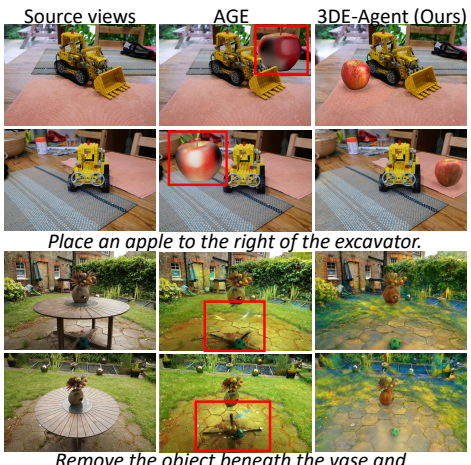
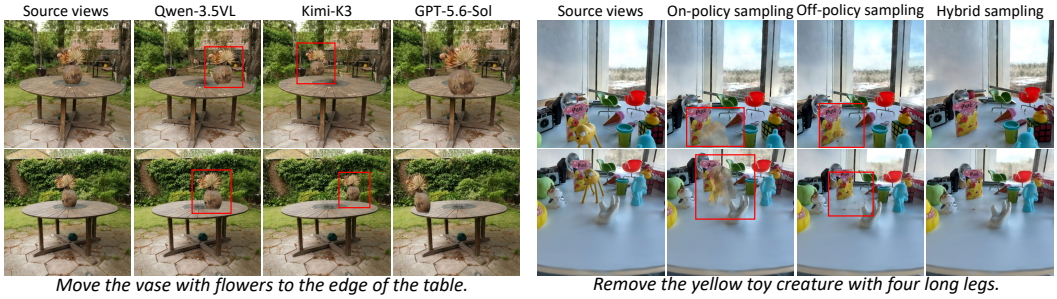


Figure 6: **Comparison with AGE.** 3DE-Agent achieves accurate placement and clearer ground.

Comparison with AGE. We further compare our method with the concurrent work AGE (Qin et al., 2026), which directly employs a multi-agent framework to invoke 3D editing models such as Trellis2. Since AGE has not released its code, we follow the editing instructions reported in its paper and present qualitative comparisons in Fig. 6. AGE directly exposes heterogeneous models to the agents without a unified tool abstraction and does not explicitly address artifacts introduced by intermediate 3D operations. As a result, the two examples exhibit an incorrectly positioned, plastic-like apple and an incompletely removed chair, respectively. In



(a) Comparison of different 3DE-Agent VLM results. (b) Comparison of different Refiner sampling results.

Figure 7: **Qualitative study on the effect of VLM and Refiner sampling strategies.** (a) Qwen-3.5VL and Kimi-K3 fail to place the vase at the correct edge of the table. (b) Refiners trained with either full on-policy or full off-policy sampling leave noticeable artifacts.

comparison, 3DE-Agent produces more photorealistic apples nearby the excavator, and achieves a cleaner removal without the table remaining in the garden scene.

4.3 Diagnostic Studies

Effect of 3DE-Agent VLM. To study the effect of different VLM on 3DE-Agent, we replace GPT-5.6-Sol with the open-source Qwen3.5VL-27B (Qwen Team, 2026) and Kimi-K3 (Team et al., 2026) multimodal models and evaluate them on the instruction *"Move the vase with flowers to the edge of the table."*. As shown in Fig. 7(a), Qwen3.5-VL-based 3DE-Agent adopts a relatively conservative strategy: although it successfully moves the vase, it does not place it close enough to the table edge. In contrast, Kimi-K3 moves the vase along the table plane beyond the table boundary, yet the result is still accepted in verification stage and returned as the final output. GPT-5.6-Sol performs accurate adjustments and eventually places the vase at the desired position.

Effect of Different Sampling Strategies. We further evaluate Refiners trained with different sampling strategies on the instruction "*Remove the yellow toy creature with four long legs.*". As illustrated in Fig. 7(b), full on-policy sampling denotes that the student generates the entire 20-step denoising trajectory, while full off-policy sampling completely discards the student trajectory and constructs z_t by forward noising the clean latent z_{gt} as in typical diffusion forward process. One can observe that full on-policy sampling suffers from trajectory drift, which makes the teacher supervision increasingly unreliable and results in blurred refinement outputs. In contrast, full off-policy sampling lacks exposure to student-visited states and therefore leaves noticeable residual artifacts. Our proposed hybrid sampling strategy combines the advantages of both settings and produces the cleanest refinement results. More experimental studies are provided in Appendix A.4.

5 Conclusion

We presented 3DE-Agent, a unified agentic framework for instruction-driven 3D scene editing. By organizing heterogeneous editing capabilities into function-oriented tools within a VLM harness, 3DE-Agent dynamically composed editing operations without relying on predefined task-specific workflows. We further introduced a multi-view Refiner trained with hybrid on-policy self-distillation and agentically constructed pseudo-paired data to correct artifacts introduced during tool execution. Experiments on diverse scenes demonstrated the effectiveness of 3DE-Agent on both standard object removal and simple and complex editing tasks. It is expected that 3DE-Agent can facilitate the emerging research of agentic 3D editing.

Limitation. 3DE-Agent has several limitations. First, as discussed in our diagnostic studies, its performance depends on the multimodal perception and reasoning capabilities of the VLM for tool invocation and error correction. Our current reliance on proprietary VLM limits local deployment and fully self-contained execution. Second, the harness in this work is manually designed. Although we leverage 3DE-Agent to automatically construct training data for the Refiner, how to extend such automation to the harness itself or other tools, toward a self-improving 3D agentic editing system, needs more exploration.

REFERENCES

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5470–5479, 2022.

Josef Bengtson, Yaroslava Lochman, and Fredrik Kahl. Gem-nr: Geometry-aware multi-view editing for nonrigid scene changes, 2026. URL https://arxiv.org/abs/2606.05142.

Tim Brooks, Aleksander Holynski, and Alexei A Efros. Instructpix2pix: Learning to follow image editing instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18392–18402, 2023.

Chenjie Cao, Yunuo Cai, Qiaole Dong, Yikai Wang, and Yanwei Fu. Leftrefill: Filling right canvas based on left reference through generalized text-to-image diffusion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024a.

Chenjie Cao, Chaohui Yu, Fan Wang, Xiangyang Xue, and Yanwei Fu. Mvinpainter: Learning multi-view consistent inpainting to bridge 2d and 3d editing. *arXiv preprint arXiv:2408.08000*, 2024b.

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. URL https://arxiv.org/abs/2511.16719.

Liyi Chen, Pengfei Wang, Guowen Zhang, Zhiyuan Ma, and Lei Zhang. Omni-3dedit: Generalized versatile 3d editing in one-pass. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12640–12650, 2026.

Minghao Chen, Iro Laina, and Andrea Vedaldi. Dge: Direct gaussian 3d editing by consistent multi-view editing. In *European Conference on Computer Vision*, pp. 74–92. Springer, 2024a.

Minghao Chen, Junyu Xie, Iro Laina, and Andrea Vedaldi. Shap-editor: Instruction-guided latent 3d editing in seconds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26456–26466, 2024b.

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21476–21485, 2024c.

Yuxin Cheng, Binxiao Huang, Taiqiang Wu, Wenyong Zhou, Chenchen Ding, Zhengwu Liu, Graziano Chesi, and Ngai Wong. Perspective-aware 3d gaussian inpainting with multi-view consistency. *arXiv preprint arXiv:2510.10993*, 2025a.

Yuxin Cheng, Binxiao Huang, Taiqiang Wu, Wenyong Zhou, Chenchen Ding, Zhengwu Liu, Graziano Chesi, and Ngai Wong. Perspective-aware 3d gaussian inpainting with multi-view consistency. *arXiv preprint arXiv:2510.10993*, 2025b.

Yuxin Cheng, Binxiao Huang, Taiqiang Wu, Wenyong Zhou, Chenchen Ding, Zhengwu Liu, Graziano Chesi, and Ngai Wong. Perspective-aware 3d gaussian inpainting with multi-view consistency. *arXiv preprint arXiv:2510.10993*, 2025c.

DeepSeek-AI. Deepseek harness: Everything is a plugin. https://github.com/deepseek-ai/deepseek-harness, 2026.

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

Zhen Fang, Wenxuan Huang, Yu Zeng, Yiming Zhao, Shuang Chen, Kaituo Feng, Yunlong Lin, Lin Chen, Zehui Chen, Shaosheng Cao, et al. Flow-opd: On-policy distillation for flow matching models. *arXiv preprint arXiv:2605.08063*, 2026.

Chunchao Guo, Jinpeng Li, Yang Li, and Zilong Huang. Worldclaw: Agentic 3d open-world generation at scale. *arXiv preprint arXiv:2608.05248*, 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*, pp. 19740–19750, 2023.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

Haobo Hu, Qi Mao, Yuanhang Li, and Libiao Jin. Camera artist: A multi-agent framework for cinematic language storytelling video generation. *arXiv preprint arXiv:2604.09195*, 2026.

Ziniu Hu, Ahmet Iscen, Aashi Jain, Thomas Kipf, Yisong Yue, David A Ross, Cordelia Schmid, and Alireza Fathi. Scenecraft: An llm agent for synthesizing 3d scenes as blender code. In *ICML*, pp. 19252–19282, 2024.

Lingxuan Huang, Sizhe He, Hengji Zhou, Liqiang Nie, Lianghao Xia, and Chao Huang. Vimax: Agentic video generation. *arXiv preprint arXiv:2606.07649*, 2026a.

Zheng-Hui Huang, Guixu Lin, Jiacheng Lin, Yi-Chuan Huang, Ruihan Yu, Muyao Niu, Siqi Yang, Yu-Lun Liu, Yung-Yu Chuang, Kaipeng Zhang, and Zhixiang Wang. Programmable world model. *arXiv preprint arXiv:2609.10540*, 2026b.

Dengyang Jiang, Xin Jin, Dongyang Liu, Zanyi Wang, Mingzhe Zheng, Ruoyi Du, Xiangpeng Yang, Qilong Wu, Zhen Li, Peng Gao, Harry Yang, and Steven Hoi. D-opsd: On-policy self-distillation for continuously tuning step-distilled diffusion models. *arXiv preprint arXiv:2605.05204*, 2026.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Justin Kerr, Chung Min Kim, Ken Goldberg, Angjoo Kanazawa, and Matthew Tancik. Lerf: Language embedded radiance fields. In *International Conference on Computer Vision (ICCV)*, 2023.

Peng Li, Suizhi Ma, Jialiang Chen, Yuan Liu, Congyi Zhang, Wei Xue, Wenhan Luo, Alla Sheffer, Wenping Wang, and Yike Guo. Cmd: Controllable multiview diffusion for 3d editing and progressive generation. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–10, 2025.

Quanhao Li, Junqiu Yu, Kaixun Jiang, Yujie Wei, Zhen Xing, Pandeng Li, Ruihang Chu, Shiwei Zhang, Yu Liu, and Zuxuan Wu. Diffusionopd: A unified perspective of on-policy distillation in diffusion models. *arXiv preprint arXiv:2605.15055*, 2026a.

Yaxuan Li, Yuxin Zuo, Bingxiang He, Jinqian Zhang, Chaojun Xiao, Cheng Qian, Tianyu Yu, Huanang Gao, Wenkai Yang, Zhiyuan Liu, et al. Rethinking on-policy distillation of large language models: Phenomenology, mechanism, and recipe. *arXiv preprint arXiv:2604.13016*, 2026b.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views. *arXiv preprint arXiv:2511.10647*, 2025.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

Hongyu Liu, Chun Wang, Feng Gao, Xuanhua He, Yue Ma, Ziyu Wan, Yong Zhang, Xiaoming Wei, and Qifeng Chen. Opsd-v: On-policy self-distillation for post-training few-step autoregressive video generators. *arXiv preprint arXiv:2607.08766*, 2026a.

Xiujin Liu, Tianyu Yang, Yilun Zhao, and Xiangliang Zhang. Designagent3d: Interactive 3d scene editing via designer-like multimodal reasoning. *arXiv preprint arXiv:2608.21438*, 2026b.

Zhiheng Liu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jie Xiao, Kai Zhu, Nan Xue, Yu Liu, Yujun Shen, and Yang Cao. Infusion: Inpainting 3d gaussians via learning depth completion from diffusion prior. *arXiv preprint arXiv:2404.11613*, 2024.

Jun Luo, Jiaxiang Tang, Ruijie Lu, and Gang Zeng. Sceneassistant: A visual feedback agent for open-vocabulary 3d scene generation. *arXiv preprint arXiv:2603.12238*, 2026.

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*, 65(1):99–106, 2021.

Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G. Derpanis, Jonathan Kelly, Marcus A. Brubaker, Igor Gilitschenski, and Alex Levinshtein. SPIn-NeRF: Multiview segmentation and perceptual inpainting with neural radiance fields. In *CVPR*, 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.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, et al. Code as agent harness. *arXiv preprint arXiv:2605.18747*, 2026a.

Xuying Ning, Katherine Tieu, Dongqi Fu, Tianxin Wei, Zihao Li, Yuanchen Bei, Jiaru Zou, Mengting Ai, Zhining Liu, Ting-Wei Li, et al. Code as agent harness. *arXiv preprint arXiv:2605.18747*, 2026b.

Jingyi Pan, Dan Xu, and Qiong Luo. Diga3d: Coarse-to-fine diffusional propagation of geometry and appearance for versatile 3d inpainting. *arXiv preprint arXiv:2507.00429*, 2025.

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, 2024. URL `https://arxiv.org/abs/2407.06191`.

Hao Qin, Tesi Lin, Mingwei Wei, Yukai Sun, Mengxu Lu, Ming Kong, and Qiang Zhu. Age: Agentic gaussian editing in 3d scenarios. In *European Conference on Computer Vision (ECCV)*, 2026.

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

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *International Conference on Computer Vision*, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Fei Shen, Chengyu Xie, Lihong Wang, Zhanyi Zhang, Xin Jiang, Xiaoyu Du, and Jinhui Tang. Imagagent: Orchestrating multi-turn image editing via constraint-aware planning and reflection. *arXiv preprint arXiv:2603.29602*, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, Jianfeng Cai, Xinyuan Cai, Peizhou Cao, Yuxuan Cao, Ziwei Chai, Y Charles, et al. Kimi k3: Open frontier intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3835–3844, 2022.

Can Wang, Ruixiang Jiang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Nerf-art: Text-driven neural radiance fields stylization. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

Jiyuan Wang, Chunyu Lin, Lei Sun, Zhi Cao, Yuyang Yin, Lang Nie, Zhenlong Yuan, Xiangxiang Chu, Yunchao Wei, Kang Liao, et al. Geometry-guided reinforcement learning for multi-view consistent 3d scene editing. *arXiv preprint arXiv:2603.03143*, 2026.

Yuxin Wang, Qianyi Wu, Guofeng Zhang, and Dan Xu. Gscream: Learning 3d geometry and feature consistent gaussian splatting for object removal. In *ECCV*, 2024a.

Yuxuan Wang, Xuanyu Yi, Zike Wu, Na Zhao, Long Chen, and Hanwang Zhang. View-consistent 3d editing with gaussian splatting. In *European conference on computer vision*, pp. 404–420. Springer, 2024b.

Zehan Wang, Ziang Zhang, Jiayang Xu, Jialei Wang, Tianyu Pang, Chao Du, Hengshuang Zhao, and Zhou Zhao. Orient anything v2: Unifying orientation and rotation understanding. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025a. URL https://arxiv.org/abs/2508.02324.

Chung-Ho Wu, Yang-Jung Chen, Ying-Huan Chen, Jie-Ying Lee, Bo-Hsu Ke, Chun-Wei Tuan Mu, Yi-Chuan Huang, Chin-Yang Lin, Min-Hung Chen, Yen-Yu Lin, et al. Aurafusion360: Augmented unseen region alignment for reference-based 360deg unbounded scene inpainting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 16366–16376, 2025b.

Hongchi Xia, Yang Fu, Sifei Liu, and Xiaolong Wang. Rgbd objects in the wild: Scaling real-world 3d object learning from rgb-d videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22378–22389, 2024.

Jianfeng Xiang, Xiaoxue Chen, Sicheng Xu, Ruicheng Wang, Zelong Lv, Yu Deng, Hongyuan Zhu, Yue Dong, Hao Zhao, Nicholas Jing Yuan, and Jiaolong Yang. Native and compact structured latents for 3d generation. *Tech report*, 2025.

Yixuan Yang, Zhen Luo, Wanshui Gan, Jinkun Hao, Junru Lu, Jinghao Yan, Zhaoyang Lyu, and Xudong Xu. Code-as-room: Generating 3d rooms from top-down view images via agentic code synthesis. *arXiv preprint arXiv:2605.18451*, 2026.

Junyan Ye, Jun He, Zilong Huang, Dongzhi Jiang, Xuan Yang, Rui Chen, and Weijia Li. Genclaw: Code-driven agentic image generation. *arXiv preprint arXiv:2605.30248*, 2026.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024a.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *ECCV*, 2024b.

Zhiyuan You, Xin Cai, Jinjin Gu, Tianfan Xue, and Chao Dong. Teaching large language models to regress accurate image quality scores using score distribution. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14483–14494, 2025.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2024.

Yanming Zhang, Jun-Kun Chen, Jipeng Lyu, and Yu-Xiong Wang. V2edit: Versatile video diffusion editor for videos and 3d scenes. *arXiv preprint arXiv:2503.10634*, 2025.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. *arXiv preprint arXiv:2601.18734*, 2026.

Yang Zheng, Mengqi Huang, Nan Chen, and Zhendong Mao. Pro3d-editor: A progressive-views perspective for consistent and precise 3d editing. *arXiv preprint arXiv:2506.00512*, 2025.

Jensen (Jinghao) Zhou, Hang Gao, Vikram Voleti, Aaryaman Vasishta, Chun-Han Yao, Mark Boss, Philip Torr, Christian Rupprecht, and Varun Jampani. Stable virtual camera: Generative view synthesis with diffusion models. *arXiv preprint arXiv:2503.14489*, 2025.

Wei Zhou, Xiongwei Zhu, Lingdong Kong, Bo Chen, Lei Zhang, Yongyuan Liang, Xiaoxia Hou, Ye Tian, Xian Sun, Yingshuo Wang, et al. On-policy self-distillation in diffusion models. *arXiv preprint arXiv:2608.24646*, 2026a.

Wei Zhou, Xiongwei Zhu, Zelin Xu, Bo Dong, Lixue Gong, Yongyuan Liang, Meng Chu, Leigang Qu, Lingdong Kong, Wei Liu, et al. Danceopd: On-policy generative field distillation. *arXiv preprint arXiv:2606.27377*, 2026b.

Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pp. 1–10, 2023.

Jingyu Zhuang, Di Kang, Yan-Pei Cao, Guanbin Li, Liang Lin, and Ying Shan. Tip-editor: An accurate 3d editor following both text-prompts and image-prompts. *arXiv preprint arXiv:2401.14828*, 2024.

Zhou Ziheng, Jiaqi Li, Huacong Tang, Ying Nian Wu, and Demetri Terzopoulos. Less is more: Early stopping rollout for on-policy distillation. *arXiv preprint arXiv:2605.27028*, 2026.

A APPENDIX

We provide the following materials in this appendix:

- Section A.1: More function-oriented tools details.
- Section A.2: More implementation details of 3DE-Agent.
- Section A.3: More implementation details of experimental settings.
- Section A.4: More experimental results.
- Section A.5: Declaration of LLM assistance.

A.1 FUNCTION-ORIENTED TOOLS DETAILS

View Observation. This tool is designed to return specific images for 3DE-Agent to observe. Instead of requiring 3DE-Agent to call reconstruction and rasterization functions directly, the tool wraps these operations and maintains the correspondence between 3D camera poses, rendered images, and image IDs for the retrieved images. 3DE-Agent usually calls this tool for scene understanding and anchor-view selection in the early editing stage, or for checking editing results and identifying problems in intermediate stages.

Object Location and Segmentation. To locate foreground objects or background regions related to the instruction in a 3D scene, a VLM is first called to observe different views and select the view in which the largest portion of the target is visible as the anchor. We then use SAM3 (Carion et al., 2025) to convert the foreground tag into a mask for the anchor image. We select the registered 3D points from COLMAP whose projections fall within this mask and project them into other views to provide approximate locations and visibility guidance. Specifically, in the other views, the semantic tag and approximate location are jointly provided to SAM3 to obtain the corresponding masks. If the projected target is outside the image or occluded, we skip segmentation in that view and exclude it from the target's mask supervision.

Finally, we follow Gaussian Grouping (Ye et al., 2024a) to learn semantic features for the Gaussians using the accepted masks. This brings two benefits: (1) Gaussian Grouping supports training multiple targets in parallel without significantly increasing the optimization time; (2) the classification probabilities p of the optimized Gaussians allow a VLM to adaptively select the best classification threshold. Specifically, we provide rendered results obtained with different filtering thresholds $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. 3DE-Agent compares these results and selects the threshold with the best visual result for foreground extraction or removal.

Object Generation. This tool converts an object in a single image or a textual concept into 3D Gaussians. We combine an image generation model (Wu et al., 2025a) with the powerful image-to-3D model TRELLIS.2 (Xiang et al., 2025) to obtain multi-view images of the object and its corresponding mesh. However, we observe that the generated mesh shows significant color drift from the input image and lacks a realistic appearance. These issues cannot be resolved by adjusting the lighting. We therefore cluster the object's appearance colors and apply color correction during post-processing. The corrected mesh is then converted into a Gaussian representation.

Object Placement. This tool first determines the bottom region and upward direction of the object to be placed. Specifically, if the object is produced by *object generation*, we can directly identify its bottom Gaussians, since objects generated by TRELLIS.2 are upright and level by default. If the object comes from a flat region in the original scene, the tool calls *object location* to locate the supporting object. It then fits a plane to the 3D points using RANSAC to determine the object's bottom Gaussians and orientation. If neither condition applies, 3DE-Agent makes a visual judgment using the outputs of Orient Anything (Wang et al.). For the target placement location, we also use RANSAC to fit a plane and determine the orientation and position of the object's Gaussians. If the target region is non-planar, such as a bear's head, 3DE-Agent performs dynamic *pose adjustment*.

Position, Scale and Pose Adjustment. This tool is designed to interact frequently with 3DE-Agent to adjust an object's position, size, and pose in a 3D scene. Specifically, 3DE-Agent observes a view divided into a dense grid and provides grid coordinates. The object is then placed on a flat region within the selected grid cell. 3DE-Agent also provides specific parameters for size and pose adjustments, such as doubling the object's size. The tool uses the object's center as the pivot

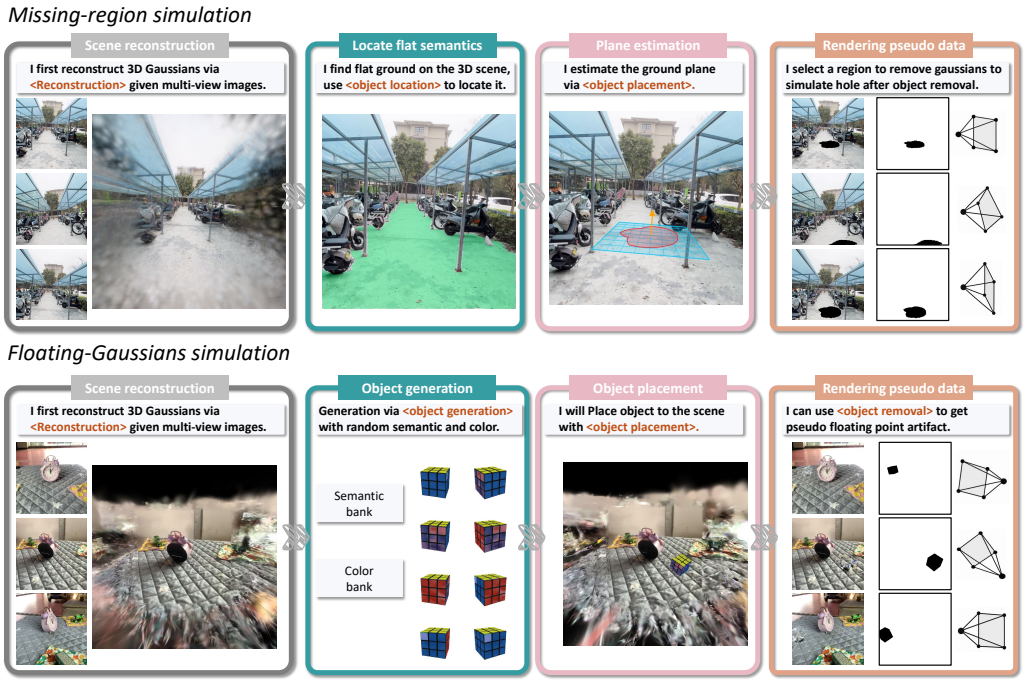


Figure 8: **Data curation pipeline.** We leverage 3DE-Agent to simulate two types of artifacts, including region-missing and floating-Gaussians.

for these transformations. Scaling proportionally changes the distances between Gaussians, while rotation changes their positions around the center to achieve the requested pose. Note that 3DE-Agent does not need to know the object's absolute size or rotation coordinates. It can determine the correct adjustment factors by interpreting feedback over multiple rounds.

Appearance Adjustment. This tool is used for local or global color adjustments without changing the geometry. Inspired by previous 3D editing methods such as GaussianEditor (Chen et al., 2024c) and Instruct-NeRF2NeRF (Haque et al., 2023), we enable optimization of the color and spherical harmonic (SH) attributes of Gaussians within the specified editing region. Other attributes, such as positions and covariances, remain frozen. To obtain supervision, we apply a 2D editor to the original rendered views one by one to modify their appearance while preserving geometry. We then use these edited views to optimize the 3D Gaussians through iterative dataset updating.

Projection-based Editing Propagation. This tool propagates a single-image edit from the current view to another registered view and is often used to support non-rigid deformation. Following GEMNR (Bengtson et al., 2026), we provide three views—the source anchor view, the edited anchor view, and the source target view—to DA3 (Lin et al., 2025) for camera estimation. We constrain the source and edited anchor views to share the same camera pose. This allows us to obtain the canonical depth of the edited anchor view. We then apply depth-based warping and use an image editor to complete the details, obtaining a consistent edited target view. This warping procedure can be called iteratively to cover a wider range of viewpoints.

Others Tools. We also implement several lightweight tools. For example, the *editor* employs GPT-Image-2 for single-view editing, while *merge* combines multiple 3D Gaussian assets in a shared canonical space.

A.2 More implementation details of 3DE-Agent

Pseudo Code of Refiner Training. We provide pseudocode to further clarify how hybrid sampling is used to train the Refiner. As shown in Alg. 1, we additionally specify how classifier-free guidance (CFG) is applied to the teacher when constructing the supervision target. After distillation, the student model ($i.e.$, the Refiner) does not demand CFG during inference.

Algorithm 1 Refiner OPSD Training with Hybrid Sampling

Require: Student f_S, frozen teacher f_T, training pair $(\mathcal{I}_{\text{pseudo}}, \mathcal{I}_{\text{gt}}, \mathcal{P}, \mathcal{M})$, noise schedule $\{\sigma_t\}_{t=1}^{T}$, CFG scale w

Construct conditions $c_S = (\mathcal{I}_{\text{pseudo}}, I^0_{\text{gt}}, \mathcal{P}, \mathcal{M})$ and $c_T = (\mathcal{I}_{\text{gt}}, I^0_{\text{gt}}, \mathcal{P}, \mathcal{M})$
Encode $\mathcal{I}_{\text{gt}}$ to obtain clean latent z_{gt}
Sample timestep $t \sim \mathcal{U}\{1, \dots, T\}$
if $t > \frac{1}{2}T$ **then**
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $z_T^{\text{on}} \leftarrow \sigma_T \epsilon$
  for $k = T, T-1, \dots, t+1$ **do**
    $\hat{z}_{0,S} \leftarrow f_S(z_k^{\text{on}}, k; c_S)$
    $z_{k-1}^{\text{on}} \leftarrow z_k^{\text{on}} + (\sigma_{k-1} - \sigma_k)\frac{z_k^{\text{on}} - \hat{z}_{0,S}}{\sigma_k}$
  end for
  $\hat{z}_t \leftarrow \text{sg}(z_t^{\text{on}})$
else
  Sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$
  $\hat{z}_t \leftarrow z_{\text{gt}} + \sigma_t \epsilon$
end if
$\hat{z}_{0,S} \leftarrow f_S(\hat{z}_t, t; c_S)$
$\hat{z}_{0,T}^{\text{cond}} \leftarrow f_T(\hat{z}_t, t; c_T)$
$\hat{z}_{0,T}^{\text{uncond}} \leftarrow f_T(\hat{z}_t, t; c_\emptyset)$
$\hat{z}_{0,T}^{\text{cfg}} \leftarrow \hat{z}_{0,T}^{\text{uncond}} + w\left(\hat{z}_{0,T}^{\text{cond}} - \hat{z}_{0,T}^{\text{uncond}}\right)$
$\mathcal{L}_{\text{OPSD}} \leftarrow \frac{1}{\sigma_t^2}\left\|\hat{z}_{0,S} - \text{sg}(\hat{z}_{0,T}^{\text{cfg}})\right\|_2^2$
Update f_S using $\nabla\mathcal{L}_{\text{OPSD}}$

Data Curation Pipeline. We elaborate on the training data construction for the Refiner. To simulate two common artifact types: missing-regions after object removal and floating-Gaussians introduced by Gaussian transformations, we design two corresponding data curation pipelines, as shown in Fig. 8.

For *missing-region simulation*, 3DE-Agent first identifies approximately planar semantic regions in the scene, such as walls or the ground. A randomly shaped local region is then selected, and the Gaussians within it are removed to mimic holes exposed after object removal.

For *floating-Gaussians simulation*, we maintain a semantic bank containing 100 object categories and a color bank containing 25 color variants. We randomly sample an object category and color to generate a 3D object, which is subsequently placed into the scene by 3DE-Agent. The inserted object is then removed using *object removal*. The incompletely removed Gaussians naturally exhibit floating-Gaussian artifact patterns, which are used as corrupted training observations for the Refiner.

A.3 More implementation details of experimental settings

VLM-based Score. We employ Claude Opus 4.6 as a VLM-based evaluator under a role-playing setting, where it is instructed to act as an experienced 3D editing inspector. The edited 3D Gaussians are evaluated from three dimensions: *Instruction Completion*, *Scene Plausibility*, and *Scene Quality*. For each dimension, the VLM judges the rendered views together with the corresponding editing instruction and assigns a score from 1 to 5. The final score is computed as the average over the three dimensions. The complete role-playing prompt is shown in Fig. 11.

Benchmark Details. We provide the detailed benchmark composition in Tab. 5. Instead of collecting new scenes from scratch, we integrate publicly available 3D scene datasets, including Mip-NeRF 360 (Barron et al., 2022), SPIn-NeRF (Mirzaei et al., 2023), Instruct-NeRF2NeRF (Haque et al., 2023), and LERF (Kerr et al., 2023). For each task category, we design task-specific prompts covering both simple atomic instructions and complex instructions. The scene and editing instruction associated with each test sample are reported in the table.

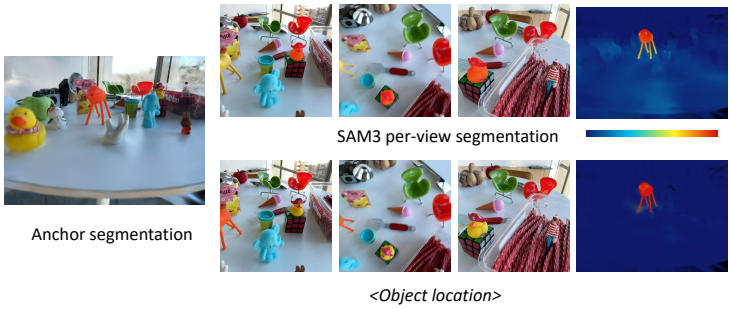


Figure 9: **Comparison between SAM3 and object location.**

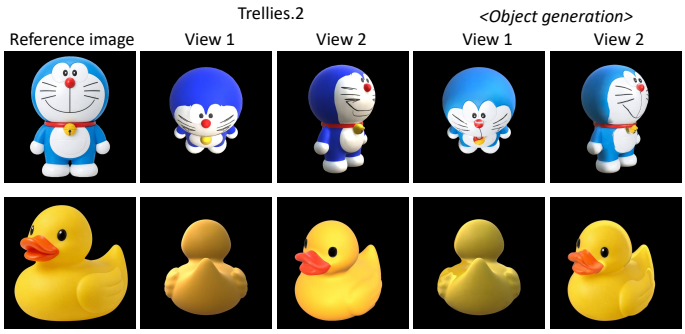


Figure 10: **Comparison between Trellies.2 and object geneartion.**

A.4 More experimental results

Effect of Function-oriented Tools. We further provide a visual comparison with directly using off-the-shelf models for 3D operations. As shown in Fig. 9 and 10, directly applying these models for object localization or object generation often leads to inaccurate localization or noticeable appearance drift in the generated objects. Such errors can accumulate throughout subsequent tool executions and ultimately degrade the performance of 3DE-Agent. We additionally study the performance of directly exposing the models to the 3DE-Agent, the results are presented in Tab. 3. Performance drops significantly due to accumulated intermediate errors.

Ablation study on Refiner. We provide a numerical comparison of different OPSD sampling strategies on the 360-USID benchmark. As shown in Tab. 4, directly performing object removal without the Refiner yields the worst PSNR and LPIPS. Among different Refiner training strategies, hybrid sampling consistently outperforms both full on-policy and full off-policy sampling, achieving the best quantitative performance. These results are consistent with the qualitative observations in Fig. 7(b), further validating the effectiveness of our hybrid sampling strategy.

Diagnostic Study on Harness. We provide an example to illustrate the necessity of our three-stage harness. As shown in Fig. 12, which presents a case of "*Turn the person into a clown with clown makeup and a fluffy, curly afro.*", the harness first interprets the editing requirements and establishes an initial plan in the *Perception and Planning* stage, where an anchor-view edit is propagated to additional viewpoints using *editing propagation*. The *Execution* stage then follows this plan to invoke the corresponding tools. Although the resulting per-view edits are individually of high quality, reconstructing them into 3D produces multi-view renders with noticeable blur and artifacts. These issues are identified during the *Verification* stage, where 3DE-Agent further observes that the main inconsistencies are concentrated around the face. It therefore proposes a revised plan that applies appearance editing specifically to the facial texture while preserving the underlying geometry. After a second round of tool execution, the resulting scene exhibits substantially improved consistency and is accepted as the final output.

Table 3: **Ablation study of encapsulated tools.**

	Add			Removal		
Method	**CLIP-D ↑**	**DeQA ↑**	**Claude ↑**	**CLIP-D ↑**	**DeQA ↑**	**Claude ↑**
w/ models	0.198	3.467	2.94	0.041	2.305	1.94
w/ tools	**0.219**	**3.788**	**3.17**	**0.062**	**2.964**	**2.56**

Table 4: **Ablation study on the 360-USID benchmark.** We report PSNR ↑ and LPIPS ↓.

Method	PSNR ↑	LPIPS ↓
3DE-Agent w/o Refiner	13.773	0.529
Refiner with:		
On-policy sampling	17.34	0.381
Off-policy sampling	16.77	0.601
Hybrid sampling	**17.88**	**0.352**

Edited 3D Video. We provide supplementary videos demonstrating that 3DE-Agent offers a unified interface for handling diverse editing instructions and produces high-quality edited 3D assets.

A.5 Declaration of LLM assistance

During the preparation of this manuscript, we utilized GPT-6 to assist in improving its linguistic quality. Specifically, after completing the initial draft, we provided the model with selected passages to obtain suggestions for grammar, clarity, and conciseness. All AI-assisted revisions were rigorously reviewed and edited by the authors, who assume full responsibility for the final accuracy and scholarly appropriateness of the content.

Role-playing prompt for evaluating 3D editing results

```
# Role
You are an experienced 3D editing evaluator with expertise in instruction following, 3D scene plausibility, and reconstruction quality. Your task is to critically
inspect the edited 3D scene and assign scores based only on the provided instruction, source/edited view pairs, and references.
Evaluate the edited 3D scene using the user's instruction, up to 10 matched SOURCE/EDITED view pairs, and any provided references. Inspect all pairs and
assign one integer score (1-5) per dimension.

1. Instruction completion
Are the requested objects, edits, poses, style, placements and relationships achieved? Check every major requirement. Cap this score at 3 if any major
requirement is clearly unmet, and at 2 if the core goal is unachieved. Accept reasonable interpretations of ambiguous instructions without inventing requirements.

2. Scene plausibility
Are object styles, materials, sizes, proportions, layout, contact and support relationships coherent with the intended scene? Do not penalize explicitly
requested stylization, exaggeration or surreal combinations, or merely the presence of toys/cartoon characters in real scenes.

3. Scene quality
Are the final renders clear and complete, without obvious holes, reconstruction blur, floating fragments, ghosting, duplicate structures, abnormal transparency
or seams? Judge absolute final quality, including inherited defects, but not intentional stylistic effects.

# Scoring scale
1.0 = Severe failure
2.0 = Major problems
3.0 = Partial success or noticeable problems
4.0 = Good with minor problems
5.0 = Fully satisfactory with no obvious problems

# Evaluation rules
- Score dimensions independently; attractive images do not compensate for unmet instructions.
- Inspect both edited targets and the whole scene; intact backgrounds must not hide serious target defects.
- Use all informative views, not only the best one. Normal occlusion is not failure.
- Do not mechanically penalize the same issue across dimensions unless it genuinely affects each.

# Output
Return only the JSON below, with one brief reason per dimension. Compute final_score as the equal-weight mean of the three scores, rounded to two decimal
places.
{
  "instruction_completion": {"score": 1.0, "reason": "..."},
  "scene_plausibility": {"score": 1.0, "reason": "..."},
  "scene_quality": {"score": 1.0, "reason": "..."},
  "final_score": 1.00
}
```

Figure 11: **Role-playing prompt for Opus 4.6 to evaluate 3D editing results.** Edited 3D scenes are evaluated by considering three dimensions of Instruction completion, scene plausibility, and scene quality.

Table 5: **Editing instructions in our benchmark.**

Type	Scene	Editing instruction
Add	Kitchen	Place an apple to the right of the excavator.
	Garden	Place a Doraemon on the table.
	Bear	Give the bear a hat.
	Bicycle	Place a Labubu on the black bench.
	Face	Give him sunglasses.
	Teatime	Place a small yellow rubber duck beside the white mug.
Remove	Garden	Remove the vase and all its contents.
	Bicycle	Remove the bicycle.
	Figurines	Remove the yellow long-legged figurine from the table.
	Teatime	Remove the red apple from the table.
	spinnerf10	Remove the watering can from the bench.
	Bear	Remove the bear stone.
Appearance	Face	Turn the person into a clown with clown makeup and a fluffy, curly afro.
	Bear	Make the bear black.
	Bicycle	Turn it into an autumn scene.
	Fangzhou	Turn the person into a plaster statue.
	Kitchen	Paint the excavator red while preserving its black tracks.
	Garden	Make this garden snowy.
Deform	Kitchen	Raise the bucket.
	Room	Arrange the shoes neatly.
	Face	Open the person's mouth wide in an O shape.
	Kitchen	Make the bucket wider.
	Person	Raise his left arm.
	Bear	Make the bear bigger.
Composition	spinnerf10	Replace the plant with Doraemon.
	Garden	Remove the object beneath the vase and transform the scene into Van Gogh painting style.
	Kitchen	Place a red apple to the right of the excavator and paint the excavator blue.
	Person	Give him a suit, and place a basketball by his feet.
Under-specified	Garden	This scene needs to be colder.
	Kitchen	How about if the excavator faces on me?
	Garden	Move the vase with flowers to the edge of the table.
	Figurines	Make the tabletop look tidier.
Cross-scene	Kitchen → Garden	Transfer the yellow toy bulldozer to the round wooden table in Garden.
	Ramen → Teatime	I want the sheep in the second scene to have the noodle from the first scene.
	Stump → Garden	Make the garden in the same season as Stump.
	Figurines → Kitchen	Transfer the blue elephant figurine from Figurines to the excavator left.
Multimodal	Person	Let him be like: *[reference image]*.
	Bear	Make the scene in the style of *[reference image]*.
	Kitchen	Paint the excavator using the colors shown in *[reference image]*.
	Face	Give the person the hairstyle shown in *[reference image]*.

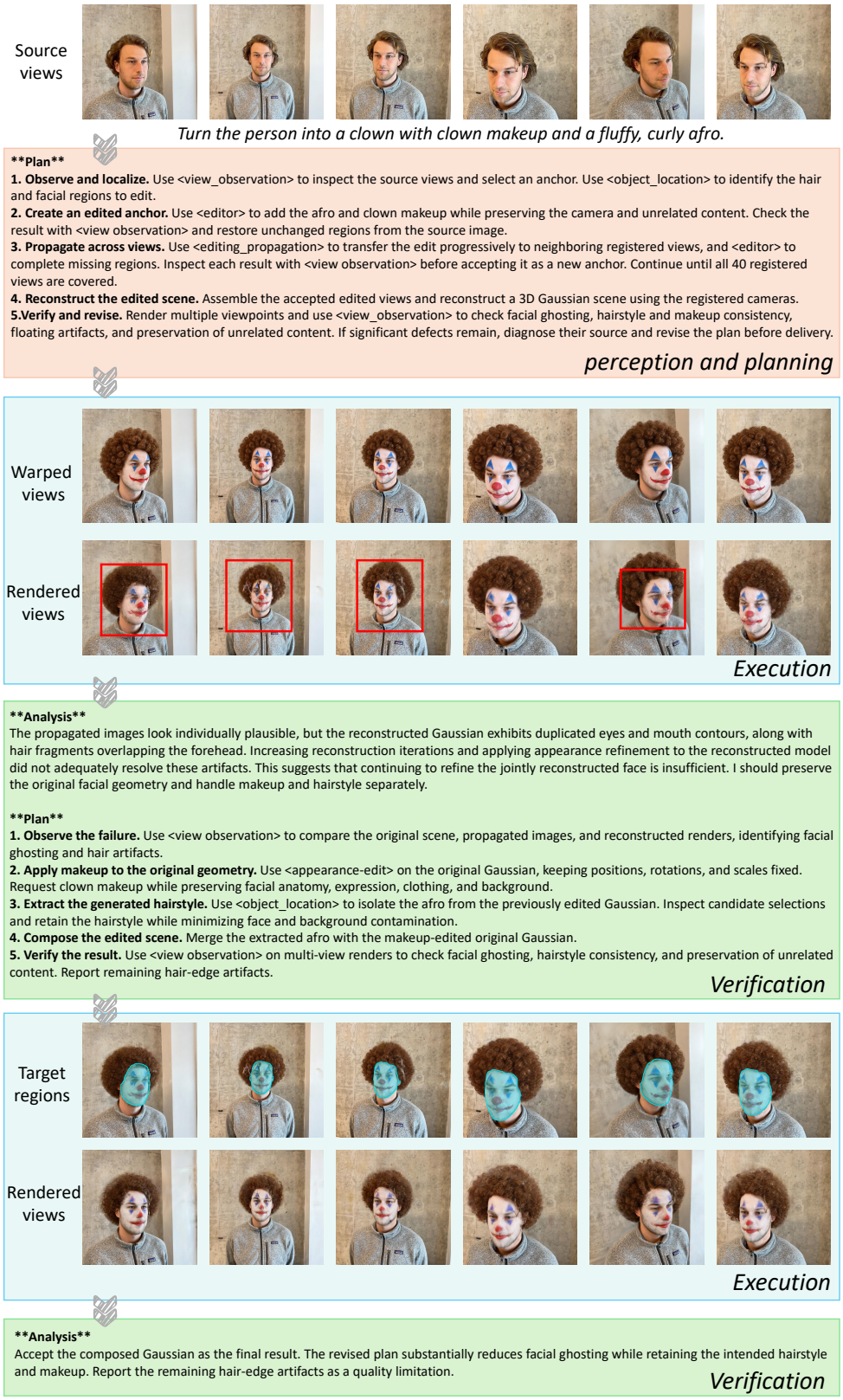


Figure 12: **Demonstration of our harness during 3D editing.** The three stages are performed by 3DE-Agent in order. An extra plan is proposed to revise the observed errors.