JOINTEDIT3D: FEED-FORWARD 3D SCENE EDITING IN A UNIFIED LATENT SPACE

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing 3D scene editing methods typically rely on per-scene optimization over explicit 3D representations or cascaded edit-and-reconstruct pipelines, resulting in high test-time cost, limited 3D awareness, and structural inconsistencies. To couple appearance synthesis with geometry prediction, we adapt a pretrained unified RGB-geometry latent space to feed-forward scene editing. Given a source video and an edited reference image, **JointEdit3D** performs asymmetric latent inpainting: it observes only the edited RGB reference latent and jointly generates the remaining RGB latents and the entire geometry latent along the source trajectory. JointEdit3D introduces a dedicated SceneAnchor Branch to inject source-scene structure without forcing direct copying, and adopts edit/background-aware losses to balance edited-region fidelity with unedited-content preservation. To address the lack of paired resources for standardized 3D scene editing evaluation, we introduce **SceneEdit3D-15K**, a dataset with 15K paired editing samples and renderer-provided 3D annotations, together with **SceneEdit3D-Bench**, a curated 100-sample benchmark. Experiments show that JointEdit3D improves edited-region quality and 3D structural completeness over prior baselines while maintaining competitive background preservation.

1 INTRODUCTION

Editing a 3D scene requires modifying its appearance and geometry consistently across views while preserving unaffected scene content. Moving an object, for example, means placing it at a new location and reconstructing the surfaces revealed at its original position. We study this problem in a feed-forward setting: given a source video and an edited reference image, we jointly predict edited appearance and geometry along the source trajectory.

Existing methods often rely on 2D editing guidance or separate view editing from 3D reconstruction. Optimization-based methods edit NeRFs (Mildenhall et al., 2021) or 3D Gaussian Splatting (Kerbl et al., 2023) with guidance from pretrained 2D diffusion models (Haque et al., 2023; Chen et al., 2024b; Wu et al., 2024; Lee et al., 2025). Although the representation is 3D, editing guidance is derived from 2D edits of rendered views. These methods often require precise masks, and their per-scene optimization is costly. Edits can also affect regions beyond the specified mask, for example through changes in shadows and reflections. Recent feed-forward methods reduce test-time cost through multi-view inpainting, video propagation followed by reconstruction, or edited-view alignment before a 3D stage (Cao et al., 2024; Qu et al., 2025; Chen et al., 2026; Liu et al., 2025). Although these methods can incorporate geometric cues or 3D priors, inconsistencies introduced during view editing can carry into the 3D output.

A natural way to reduce this separation is to edit in a latent space that represents 3D structure. Prior work studies voxel-structured latents for object editing (Xiang et al., 2025; Jin et al., 2025; Li et al., 2026) and geometry-aware latents for scene generation and reconstruction (Yang et al., 2025; Ren et al., 2025; Wu et al., 2026; Huang et al., 2026), but these methods are not designed for reference-guided scene editing. Our key idea is to let edit decisions, appearance changes, and geometry updates share the same latent state. We therefore adapt a pretrained unified RGB-geometry latent space (Huang et al., 2026) to 3D scene editing and propose **JointEdit3D**, which jointly generates edited appearance and geometry rather than editing views first and reconstructing afterward. Given a source video, a single edited reference frame, and an optional language instruction, JointEdit3D

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.

predicts a complete edited RGB-geometry latent and decodes it into RGB video and a scene point cloud along the source trajectory. We formulate this as asymmetric RGB-geometry latent inpainting: the edited RGB reference provides the observed target content, while the model jointly generates the remaining RGB latents and the entire geometry latent; the reference provides explicit appearance, placement, and local context for propagating the edit across views.

The source video provides reconstruction evidence but conflicts with the edited reference in regions that should change. We introduce a *SceneAnchor Branch* that receives edit cues from the evolving main-branch state and injects source RGB features into the frozen generator through interleaved residual conditioning, treating the source as an edit-aware preservation prior rather than a target to copy. A standard MSE objective also averages over all latent positions, so sparse edits are underweighted relative to unchanged background; we introduce a set of edit-aware losses that separate edited and background regions in both RGB and geometry latents, while emphasizing positions with larger latent changes and frames farther from the reference.

Progress in 3D scene editing also requires paired supervision and consistent evaluation protocols. Existing 3D editing resources rarely provide paired before/after scene edits with 3D annotations, and evaluations are often limited to 2D metrics or single-operation settings. To address this, we construct **SceneEdit3D-15K**, a dataset of 15K paired scene edits with RGB videos and renderer-provided 3D annotations, and further curate **SceneEdit3D-Bench**, a 100-sample held-out benchmark evaluating edit fidelity, background preservation, and 3D structure.

Our contributions are summarized as follows:

- We formulate reference-guided 3D scene editing as asymmetric inpainting in a pretrained unified RGB-geometry latent space, jointly predicting edited appearance and geometry along the source trajectory.
- We introduce JointEdit3D, combining a SceneAnchor Branch conditioned on the evolving edit state with edit-aware losses to propagate the edit specified by a single reference across views while preserving unedited content, without edit-region masks at inference.
- We introduce **SceneEdit3D-15K**, a dataset of 15K paired scene edits with renderer-provided 3D annotations, and **SceneEdit3D-Bench** for evaluating edit fidelity, background preservation, and 3D structure.

2 Related Work

3D scene editing. Existing methods differ in how they connect edit guidance with 3D representations. Optimization-based methods update NeRFs or 3D Gaussians per scene with text, image, mask, or segmentation guidance (Mirzaei et al., 2023; Haque et al., 2023; Zhuang et al., 2023; Chen et al., 2024b), often adding grouping, multi-view consistency, or control mechanisms (Ye et al., 2024; Wang et al., 2024; Wu et al., 2024; Chen et al., 2024a; Lee et al., 2025; Zhuang et al., 2024; Zhang et al., 2025a). Recent feed-forward methods differ in their conditioning inputs and how edited views are converted into 3D outputs. MVInpainter inpaints masked multi-view RGB (Cao et al., 2024). EditCast3D propagates a single-frame 2D edit through video and then reconstructs 3D (Qu et al., 2025). Omni-3DEdit edits in 2D diffusion latents before decoding 3D (Chen et al., 2026). Edit3r applies a 2D editor to the input views and then predicts edited 3D Gaussians in a feed-forward pass (Liu et al., 2025). JointEdit3D instead edits directly in a unified RGB-geometry latent, jointly generating aligned RGB and geometry while preserving unedited source content.

3D latent editing and geometry-aware generation. At the object level, prior work explores controllable generation and editing in voxel-structured 3D latent spaces (Xiang et al., 2025; Jin et al., 2025; Li et al., 2026). A related line of scene generation and reconstruction couples synthesis with 3D structure via camera, point-cloud, 3D-cache, or reconstruction features (Zhou et al., 2025; Li et al., 2024; Yang et al., 2025; Ren et al., 2025; Zhang et al., 2025b; Wu et al., 2026; Cao et al., 2025), but is not designed for editing existing scenes under an edited reference. Building on Gen3R's pretrained unified RGB-geometry latent space (Huang et al., 2026), our approach introduces reference-guided scene editing with source-scene anchoring to preserve unedited content.

Editing datasets and benchmarks. Existing image and video editing datasets provide edit supervision but lack paired 3D annotations (Brooks et al., 2023; Zhang et al., 2023a; Fu et al., 2026). General 3D scene datasets provide real or synthetic data for reconstruction and scene understanding (Dai et al., 2017; Straub et al., 2019; Roberts et al., 2021; Fu et al., 2021; Yeshwanth et al., 2023) but are not built as before/after editing corpora. Real-scene inpainting benchmarks such as SPIn-NeRF (Mirzaei et al., 2023) and 360-USID (Wu et al., 2025) focus on object removal. SceneEdit3D-15K provides large-scale paired scene edits across multiple operations with renderer-provided 3D annotations, while SceneEdit3D-Bench supports standardized evaluation.

3 METHOD

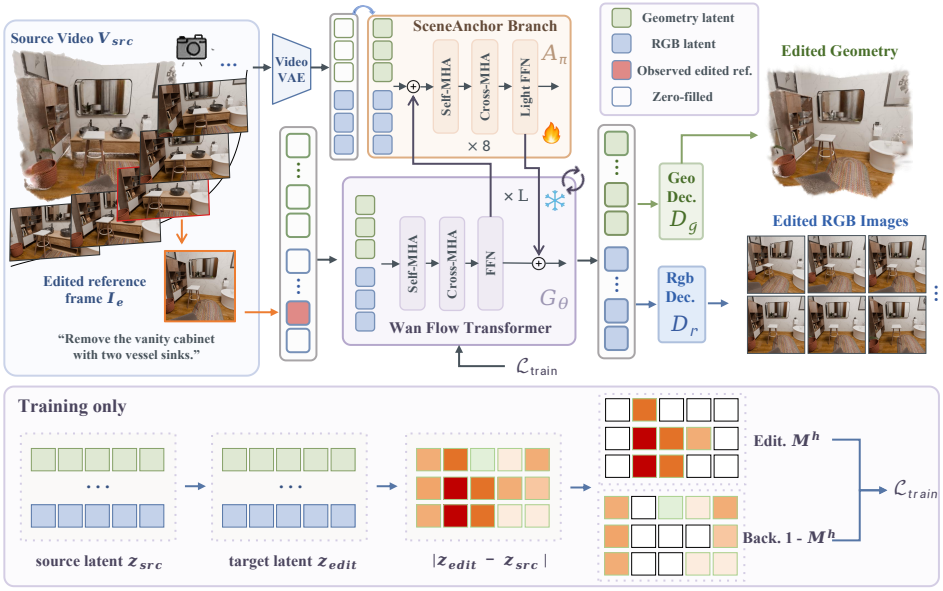

Figure 1: JointEdit3D pipeline: RGB-geometry latent inpainting from a source video and one edited reference frame; source-scene anchoring preserves unedited structure.

3.1 PRELIMINARIES

We briefly review how Gen3R (Huang et al., 2026), a 3D scene reconstruction model, forms a joint RGB-geometry latent space. Given a multi-view video $\mathcal{V} = \{I^i\}_{i=1}^N$, Gen3R encodes appearance and geometry through two aligned streams:

$$\mathbf{z}^{\text{rgb}} = E_{\text{rgb}}(\mathcal{V}), \qquad \mathbf{h}^{\text{vggt}} = \Phi_{\text{vggt}}(\mathcal{V}), \qquad \mathbf{z}^{\text{geo}} = A_\psi^{\text{enc}}(\mathbf{h}^{\text{vggt}}), \tag{1}$$

where E_{rgb} is the pretrained video VAE encoder (Wan et al., 2025), Φ_{vggt} extracts VGGT (Wang et al., 2025) features, and A_ψ^{enc} is the encoder of the learned Geometry Adapter that maps VGGT tokens into the video latent domain, giving $\mathbf{z}^{\text{rgb}}, \mathbf{z}^{\text{geo}} \in \mathbb{R}^{C\times F\times H\times W}$. Gen3R concatenates the two latents along the spatial width and fine-tunes a Wan-based diffusion backbone to generate the resulting RGB-geometry grid:

$$\mathbf{z}(\mathcal{V}) = [\,\mathbf{z}^{\text{rgb}};\ \mathbf{z}^{\text{geo}}\,] \in \mathbb{R}^{C\times F\times H\times 2W}, \tag{2}$$

where the first and second spatial halves store RGB and geometry under matched indices, so generation is joint rather than RGB-only. In JointEdit3D, G_θ denotes this joint-space Wan backbone post-trained by Gen3R (Huang et al., 2026) and $G_{\theta,\pi}$ denotes the full editor after adding the SceneAnchor Branch parameters π. We reuse the pretrained Geometry Adapter to compute $\mathbf{z}^{\text{geo}}$, and use $\mathbf{z}_{\text{src}} = \mathbf{z}(\mathcal{V}_{\text{src}})$ and $\mathbf{z}_{\text{edit}} = \mathbf{z}(\mathcal{V}_{\text{edit}})$ as the source and edited target latents, respectively, where $\mathcal{V}_{\text{edit}}$ is the edited target video paired with $\mathcal{V}_{\text{src}}$ in our training data. The RGB half of $\mathbf{z}_{\text{edit}}$ comes from the video VAE and its geometry half from the frozen VGGT and Geometry Adapter, so the geometry half of $\mathbf{z}_{\text{edit}}$ is a training pseudo-target and is not provided at inference.

3.2 METHOD OVERVIEW

Given a source video $\mathcal{V}_{\text{src}} = \{I_s^i\}_{i=1}^N$ and an editing condition $\mathcal{C} = (I_e, p)$, comprising an edited reference frame and an optional language instruction, JointEdit3D jointly predicts edited RGB views and geometry along the source trajectory by denoising the unified RGB-geometry latent (Appendix D.4). It treats editing as joint latent inpainting rather than RGB-only generation followed by reconstruction, with the edited reference frame I_e specifying the target edit and the source RGB condition providing evidence for preserving unedited content. As shown in Figure 1, the framework combines single-frame-guided inpainting (§3.3), source-scene preservation through a SceneAnchor Branch (§3.4), and region-decomposed training objectives (§3.5). We write $\mathbf{ctx} = (\phi_{\text{CLIP}}(I_e),\ \phi_{\text{T5}}(p))$ for the cross-attention context derived from the reference image and instruction.

3.3 EDIT CONDITIONING VIA SINGLE-FRAME INPAINTING

In Gen3R reconstruction, every RGB frame is observed and geometry is inferred from it. Editing is asymmetric: only one target RGB frame is observed; the remaining edited RGB views and the entire geometry stream are generated jointly, conditioned on the reference and source video. The source video disagrees with the edited target where changes are required, so it enters through a separate branch (§3.4) rather than as observed target content. We implement this asymmetric observation with latent inpainting: the video VAE encodes I_e alone into $\hat{\mathbf{z}}_e^{\text{rgb}}$, which we place in a zero-filled tensor $\mathbf{y}_{\text{ref}}$ at latent step $k = \lceil r/4 \rceil$, where r is the zero-based reference clip index:

$$\mathbf{y}_{\text{ref}}[:, k, :, : W] = \hat{\mathbf{z}}_e^{\text{rgb}}, \quad \text{all other entries} = 0. \tag{3}$$

A four-channel observation mask $\mathbf{m}_{\text{ref}}$ represents the supplied RGB reference under Wan's $4\times$ temporal packing; geometry remains unobserved. We concatenate

$$\mathbf{y} = [\,\mathbf{y}_{\text{ref}};\ \mathbf{m}_{\text{ref}}\,] \in \mathbb{R}^{(C+4)\times F\times H\times 2W} \tag{4}$$

with the noisy latent $\mathbf{z}_t$ along the channel dimension before the transformer. This observation mask specifies the known frame, not an edit region; temporal grouping and mask packing are detailed in Appendix B.1.

3.4 SOURCE-SCENE PRESERVATION VIA SCENEANCHOR BRANCH

The edited reference specifies the target change, while the source video provides evidence for preserving unedited scene content. Direct source conditioning must balance preservation against changes specified by the reference, while removing source conditioning weakens background preservation. We therefore use a dedicated *SceneAnchor Branch* A_π that receives the evolving denoising state from the frozen main branch and injects source-scene features back as residual corrections. Architecturally, this branch is related to residual adapter and control-branch designs (Zhang et al., 2023b; Mou et al., 2024; Ye et al., 2023), and its role here is source-scene anchoring rather than generic spatial control or image-prompt injection.

Architecture. We freeze G_θ and train only the SceneAnchor Branch parameters π to learn residual source-scene adaptation. The branch consumes only the RGB half of the source. Concretely, we feed $\mathbf{c}_{\text{src}} = [\,\mathbf{z}_{\text{src}}^{\text{rgb}};\ \mathbf{0}\,]$, where $\mathbf{z}_{\text{src}}^{\text{rgb}}$ encodes $\mathcal{V}_{\text{src}}$ via the video VAE and the geometry half is zeroed since no source geometry is assumed at inference. The full source latent $\mathbf{z}_{\text{src}}$ from §3.1 is computed only during training to derive latent-difference masks and weights (§3.5). At inference, the conditioning inputs are $\mathbf{c}_{\text{src}}$, $\mathbf{y}$, and $\mathbf{ctx}$; the full source latent $\mathbf{z}_{\text{src}}$ and target latent $\mathbf{z}_{\text{edit}}$ are training-only. A 3D convolutional patch embedding with the same kernel and stride as G_θ projects this anchor input into an anchor state $\mathbf{c}_0$. The branch follows G_θ's block structure at hidden width d_a=512, with learned projections between the branch and backbone widths; each anchor block attaches to a selected backbone layer (Appendix B.1).

Interleaved residual fusion. During denoising, A_π is interleaved with selected layers of G_θ, so source conditioning adapts to the current edit state rather than acting as a static copy signal. Let $\mathbf{x}_{\ell_j}$ denote the main-branch hidden tokens at layer ℓ_j. At the j-th injected layer, the current denoising state is projected into the anchor stream and processed by the anchor block:

$$\mathbf{c}_j \leftarrow \text{AnchorBlock}_j(\mathbf{c}_{j-1} + W_{\text{in}}^{(j)}(\mathbf{x}_{\ell_j}),\ W_e^{(j)}(\mathbf{e}),\ W_{\text{ctx}}^{(j)}(\mathbf{ctx})), \tag{5}$$

where $\mathbf{e}$ is the diffusion timestep embedding. The anchor output is mapped back to the main width and added as a residual update:

$$\mathbf{x}_{\ell_j} \leftarrow \mathbf{x}_{\ell_j} + W_{\text{out}}^{(j)}(\mathbf{c}_j). \tag{6}$$

All $W_{\text{out}}^{(j)}$ are zero-initialized, so the editor reduces to G_θ at the start of training. Because the anchor stream consumes both the source RGB condition and the evolving main-branch state, its residual updates depend on the current edit hypothesis. This provides a mechanism for balancing source preservation with the requested change without an input edit-region mask. Disabling the feedback degrades both edit-region and background metrics (Table 5), supporting the use of the evolving main state for source conditioning.

3.5 Training Objectives

The base objective follows flow matching (Lipman et al., 2023) over the edited target latent $\mathbf{z}_{\text{edit}}$. A standard MSE averages over all positions, giving sparse edited regions a small share of the loss weight; we therefore decompose the loss into edit and background regions in each modality. Given noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and timestep $t \in (0, 1)$, we set $\mathbf{z}_t = (1-t)\mathbf{z}_{\text{edit}} + t\boldsymbol{\epsilon}$. The full editor predicts a velocity $G_{\theta,\pi}(\mathbf{z}_t, t, \mathbf{y}, \mathbf{c}_{\text{src}}, \mathbf{ctx})$ supervised toward $\boldsymbol{\epsilon} - \mathbf{z}_{\text{edit}}$. For each position i,

$$l_i = \frac{1}{C} \left\| G_{\theta,\pi}(\mathbf{z}_t, t, \mathbf{y}, \mathbf{c}_{\text{src}}, \mathbf{ctx})_i - (\boldsymbol{\epsilon}_i - \mathbf{z}_{\text{edit},i}) \right\|_2^2. \tag{7}$$

A training-only temporal weight a_i upweights positions temporally farther from the reference (Appendix B.1); we write $\tilde{l}_i = a_i l_i$ and apply region weighting separately to the RGB and geometry halves.

Edit-aware region decomposition. To prevent the background from dominating the loss, we estimate edited positions from source-target latent changes in each half. For $h \in \{\text{rgb}, \text{geo}\}$ and positions Ω_h,

$$d_i^h = \frac{\frac{1}{C} \left\| \mathbf{z}_{\text{edit},i}^h - \mathbf{z}_{\text{src},i}^h \right\|_1}{\max_{j \in \Omega_h} \frac{1}{C} \left\| \mathbf{z}_{\text{edit},j}^h - \mathbf{z}_{\text{src},j}^h \right\|_1 + \varepsilon_{\text{den}}}, \qquad M_i^h = \mathbf{1}[d_i^h > \tau]. \tag{8}$$

This produces separate masks for appearance and geometry so the loss can emphasize modality-specific latent changes. These latent-difference masks are used only to construct the training loss; they are not available or required at inference. Appendix A shows training-time masks, inference-time edit responses, and the calibration of τ.

Within-region weighting. Inside the edit region, a linear weight $w_i^h = 1 + \alpha d_i^h$ assigns larger relative loss weights to larger latent changes:

$$\mathcal{L}_{\text{edit}}^h = \frac{\sum_{i \in \Omega_h} w_i^h \tilde{l}_i M_i^h}{\sum_{i \in \Omega_h} w_i^h M_i^h + \varepsilon_{\text{den}}}, \quad \mathcal{L}_{\text{bg}}^h = \frac{\sum_{i \in \Omega_h} \tilde{l}_i (1 - M_i^h)}{\sum_{i \in \Omega_h} (1 - M_i^h) + \varepsilon_{\text{den}}}, \tag{9}$$

and the two regions are combined per half as $\mathcal{L}_h = \lambda \mathcal{L}_{\text{edit}}^h + (1-\lambda)\mathcal{L}_{\text{bg}}^h$. The final objective $\mathcal{L}_{\text{train}} = \mathcal{L}_{\text{rgb}} + \gamma_{\text{geo}} \mathcal{L}_{\text{geo}}$ balances RGB and geometry-latent supervision, where λ controls the edit/background balance and γ_{geo} the relative weight of geometry-latent supervision.

4 SceneEdit3D-15K Dataset

Paired editing data with 3D ground truth remain limited. Existing 2D editing datasets (Brooks et al., 2023; Zhang et al., 2023a) lack paired geometry and multi-view supervision for evaluating scene structure. We introduce **SceneEdit3D-15K**, a large-scale dataset of 15K paired scene editing samples with language instructions, edited reference frames, edit masks, and renderer-provided 3D annotations, together with **SceneEdit3D-Bench**, a curated benchmark of 100 held-out samples.

SceneEdit3D-15K is built from composable indoor scenes in Imaginarium (Zhu et al., 2025) whose objects can be individually manipulated in Blender. A VLM analyzes rendered views, object metadata, and layout relations to propose canonical deletion and relocation tasks, while Blender-side

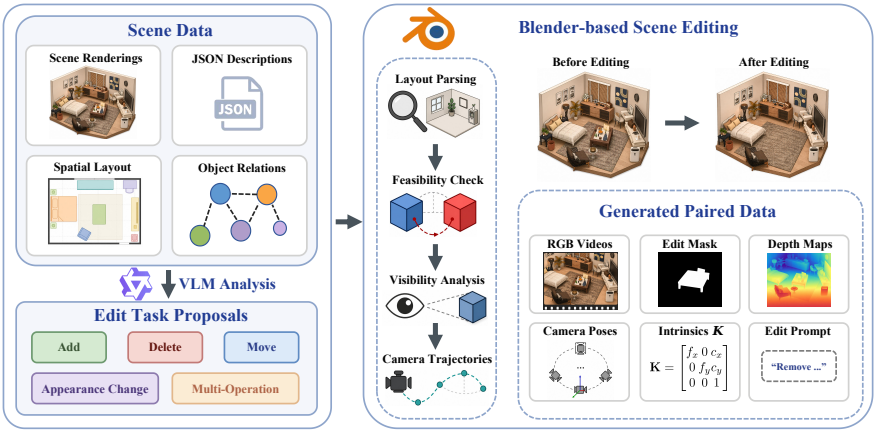

Figure 2: SceneEdit3D-15K data generation pipeline.

procedures generate material/texture appearance changes; Blender then executes each edit deterministically and renders paired source and target videos under identical camera trajectories, with addition and reverse-relocation samples derived by swapping before/after states and reversing frame order. Multi-operation samples combine edits to distinct objects. Training uses the paired RGB videos, one edited reference frame, and a natural-language instruction. Renderer annotations support data preparation, calibration, and evaluation, as detailed in the appendix.

The dataset contains 13,799 training samples from 135 scenes and 1,520 held-out samples from 16 separate scenes. The held-out pool is split at the sample level into 1,420 validation samples and 100 test samples forming SceneEdit3D-Bench, stratified across the five operation types and scene categories, with varied object scales, image locations, and partial occlusions, supporting region-aware evaluation of edited-region quality, background preservation, and 3D structural correctness; generation protocol, quality control, and split details are in Appendix C.

5 Experiments

5.1 Experimental Setup

Implementation details. We instantiate JointEdit3D with a frozen Gen3R-tuned image-camera-conditioned Wan2.1 1.3B DiT backbone G_θ (L=32 blocks) (Huang et al., 2026). The SceneAnchor Branch A_π is a trainable branch with hidden width d_a=512 and K=8 anchor blocks, totaling 70M trainable parameters. We use AdamW with initial learning rate 1×10^{-4}, bf16 mixed precision, and gradient checkpointing. For the region-decomposed objective: edit-mask threshold τ=0.15, within-region weight α=5.0, temporal weight η=0.5, edit/background balance λ=0.5, and geometry weight γ_{geo}=1.5. Training runs on the SceneEdit3D-15K split for 10K steps with batch size 10 per device on 8 NVIDIA H200 GPUs. Further implementation details are provided in Appendix B.1.

Baselines. We compare with representative optimization-based and feed-forward methods: SPIn-NeRF (Mirzaei et al., 2023), Gaussian Grouping (Ye et al., 2024), GScream (Wang et al., 2024), GaussianEditor (Chen et al., 2024b), MVInpainter (Cao et al., 2024), and Omni-3DEdit (Chen et al., 2026), and include SEVA (Zhou et al., 2025) as a protocol-mismatched novel-view synthesis diagnostic. Table 1 summarizes interfaces and operation coverage. A check under Unified weights means one shared set of editor weights covers every supported operation. Baselines use official implementations; input adaptations, comparison scope, and frame budgets are in Appendix D.3.

Evaluation benchmark and metrics. We evaluate on **SceneEdit3D-Bench** and use 360-USID (Wu et al., 2025) for real-scene deletion. Trained only on synthetic SceneEdit3D-15K, JointEdit3D also shows qualitative generalization on DAVIS, ScanNet++, CO3D, and casual captures (Appendix B.3). We report PSNR↑/LPIPS↓ on edit/background regions and 3D metrics against renderer-provided edited geometry (Appendix D.4).

Table 1: Capability comparison of editing methods (Opt./FF: optimization-based/feed-forward; ✓: supported; ○: protocol-mismatched; –: unsupported under the released interface).

Method	Type	Properties			Edit operations				
		Mask-free	Pose-free	Unified weights	Delete	Add	Move	Appear.	Multi-op
SPIn-NeRF (Mirzaei et al., 2023)	Opt.	–	–	–	✓	–	–	–	–
Gaussian Grouping (Ye et al., 2024)	Opt.	–	–	–	✓	○	–	○	–
GScream (Wang et al., 2024)	Opt.	–	–	–	✓	–	–	–	–
GaussianEditor (Chen et al., 2024b)	Opt.	–	–	–	✓	○	–	○	–
SEVA (Zhou et al., 2025)	FF	✓	–	✓	○	○	○	○	○
MVInpainter (Cao et al., 2024)	FF	–	✓	–	✓	✓	✓	✓	✓
Omni-3DEdit (Chen et al., 2026)	FF	✓	–	–	✓	✓	✓	✓	✓
Ours	FF	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Delete edit-region results (PSNR↑/LPIPS↓).

Method	SceneEdit3D-Bench	360-USID
SPIn-NeRF (Mirzaei et al., 2023)	19.20 / 0.528	15.89 / 0.4826
Gaussian Grouping (Ye et al., 2024)	20.41 / 0.377	16.67 / 0.4241
GScream (Wang et al., 2024)	20.38 / 0.449	14.37 / 0.5725
GaussianEditor (Chen et al., 2024b)	20.26 / 0.451	16.23 / 0.3975
MVInpainter (Cao et al., 2024)	21.37 / 0.406	15.60 / 0.5677
Omni-3DEdit (Chen et al., 2026)	24.86 / 0.307	17.78 / 0.3655
Omni-3DEdit + SceneEdit3D FT	28.99 / 0.230	18.02 / 0.3637
JointEdit3D (Ours)	31.92 / 0.151	18.57 / 0.3426

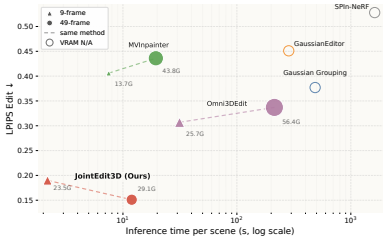

Figure 3: Efficiency-quality plot.

5.2 Main Results

Quantitative comparison. We organize comparisons by operation coverage, with best and second-best shaded: Table 2 reports edit-region quality in the shared deletion setting on SceneEdit3D-Bench and 360-USID, and Table 3 reports operation-wise results for methods supporting all five edit types, with separate PSNR↑/LPIPS↓ on the renderer-defined edit ("Edit") and background ("Bg.") regions.

Tables 2 and 3 show that JointEdit3D achieves the strongest edited-region quality on the deletion benchmark and all five SceneEdit3D-Bench operations, with competitive background preservation. Mask-guided baselines receive additional edit-localization information and attain higher background PSNR on some operations. JointEdit3D substantially improves both edit and background metrics over the mask-free Omni-3DEdit. Fine-tuning Omni-3DEdit on SceneEdit3D-15K improves every operation, demonstrating that the dataset benefits another editing architecture. JointEdit3D retains its lead after both models are adapted on the same dataset for 10K updates. Move and Multi-op remain challenging. Gains on real-world 360-USID are smaller than on SceneEdit3D-Bench, likely because its simpler scenes with centrally located target objects leave less room for improvement. Additional object-removal results on Inpaint360GS are in Appendix D.2. Figure 3 shows leading edit quality and faster inference under the reported settings.

Qualitative deletion results. Figure 4 compares synthetic SceneEdit3D-Bench and real Instruct-NeRF2NeRF/DL3DV captures (Haque et al., 2023; Ling et al., 2024). SPIn-NeRF and GaussianEditor leave residuals or blur around removed objects. MVInpainter retains boundary discontinuities and shadows, while Omni-3DEdit changes background colors and misses the small photo-frame edit in the cluttered shelf. JointEdit3D removes the targets while preserving nearby structures, including the lamp and shelf contents. Together with the 360-USID metrics, these examples support transfer to real inputs despite synthetic-only training.

3D geometry quality. Table 4 evaluates geometry against the renderer-provided edited cloud using the Gen3R protocol (Huang et al., 2026) (Appendix D.4). Native joint output and *JointEdit3D RGB + VGGT* share the generated RGB sequence but differ in geometry prediction. Joint output lowers edit-region CD and retains comparable background CD. Without any confidence filtering, the native output achieves better Completeness and F-score than *JointEdit3D RGB + VGGT*. This likely reflects the generative model's ability to infer additional geometry in occluded regions and at object boundaries, where point locations are less certain. Filtering these predictions at a confidence threshold of 0.85 improves Accuracy over the unfiltered output while retaining better Completeness and F-score than the RGB + VGGT reconstruction. This GT-adjusted setting is a post-hoc diagnostic (†; Appendix B.5). Against Omni-3DEdit + VGGT, both edit-region and background CD improve.

Table 3: Operation-wise PSNR↑/LPIPS↓ on Bg./Edit regions; Average is the mean over all 100 samples (Appendix D.3).

Reg.	Method	Delete	Add	Move	Appearance	Multi-op	Average
Bg.	SEVA (Zhou et al., 2025)	17.89 / 0.421	17.74 / 0.426	17.69 / 0.411	19.00 / 0.378	17.54 / 0.445	17.93 / 0.418
	MVInpainter (Cao et al., 2024)	33.57 / 0.080	33.64 / 0.076	30.97 / 0.081	33.76 / 0.073	30.06 / 0.091	32.76 / 0.080
	Omni-3DEdit (Chen et al., 2026)	25.00 / 0.215	24.24 / 0.217	24.40 / 0.210	25.88 / 0.190	23.81 / 0.235	24.65 / 0.214
	Omni-3DEdit + SceneEdit3D FT	26.43 / 0.207	27.24 / 0.195	26.76 / 0.192	27.07 / 0.196	26.44 / 0.218	26.80 / 0.201
	JointEdit3D (Ours)	32.62 / 0.079	32.59 / 0.081	31.22 / 0.069	33.80 / 0.059	30.84 / 0.095	32.33 / 0.077
Edit	SEVA	23.39 / 0.399	15.92 / 0.523	17.87 / 0.450	17.50 / 0.464	16.93 / 0.496	18.72 / 0.465
	MVInpainter	21.37 / 0.406	20.04 / 0.362	16.99 / 0.437	18.94 / 0.349	17.92 / 0.423	19.55 / 0.392
	Omni-3DEdit	24.86 / 0.307	20.63 / 0.293	18.55 / 0.377	20.92 / 0.230	17.03 / 0.461	21.10 / 0.323
	Omni-3DEdit + SceneEdit3D FT	28.99 / 0.230	23.20 / 0.247	22.75 / 0.279	24.28 / 0.153	21.97 / 0.336	24.80 / 0.246
	JointEdit3D (Ours)	31.92 / 0.151	23.88 / 0.195	23.19 / 0.244	27.67 / 0.115	23.77 / 0.263	26.63 / 0.187



Figure 4: Deletion comparison on synthetic SceneEdit3D-Bench and real Instruct-NeRF2NeRF / DL3DV captures (Haque et al., 2023; Ling et al., 2024). Red boxes mark edited regions.

Real-scene editing across operations. Figure 5 shows dynamic object removal, relocation, and multi-operation editing. MVInpainter leaves the camel's shadow after removal and color discontinuities near multi-edit boundaries. Both baselines struggle with relocation; Omni-3DEdit's moved object appears and disappears across views. JointEdit3D produces cleaner edits while preserving nearby structures across these cases. Appearance editing, occluded removal, and native point-cloud outputs are in Appendix B.3.

5.3 Ablations and Design Comparisons

Due to computational constraints, all variants in Table 5, including the full model, use 2K training steps. Appendices B.1 and B.6 detail adaptation configurations and the controlled main-state feedback ablation.

Table 4: 3D geometry comparison. Confidence filtering uses a GT-adjusted threshold (†; Appendix B.5). Oracle and filtered rows are diagnostics excluded from ranking.

Method	Acc.↓	Comp.↓	F-score↑	Edit CD↓	Bg. CD↓
Target RGB + VGGT (oracle)	0.0724	0.0950	0.9619	0.1329	0.0914
SEVA + VGGT	0.1433	0.2244	0.8722	0.2184	0.1825
Omni-3DEdit + VGGT	0.1165	0.1313	0.9143	0.1878	0.1191
JointEdit3D RGB + VGGT	0.0891	0.1211	0.9357	0.1421	0.0958
JointEdit3D (joint output)	0.1153	0.0886	0.9702	0.1352	0.0966
JointEdit3D, confidence-filtered†	0.0872	0.1153	0.9653	0.1382	0.0827



Figure 5: Real-scene dynamic object removal, relocation, and multi-operation editing. Red boxes mark edited regions.

Table 5: Component ablations and alternative adaptation designs for JointEdit3D.

	Edit Region		Background		3D Structure
Variant	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	CD↓
JointEdit3D (full)	26.82	0.209	30.69	0.103	0.1053
w/o main-state feedback to SceneAnchor	24.35	0.312	29.73	0.114	0.1325
Direct source conditioning (rank-16 LoRA)	23.53	0.292	30.88	0.086	0.1194
w/o source observation (zero SceneAnchor input)	19.40	0.510	18.91	0.562	0.1829
w/o GEO in Wan Flow (+VGGT)	24.75	0.381	27.85	0.337	0.1506
w/o Text Condition	26.37	0.214	30.66	0.109	0.1064
w/o Region Decomposition	23.69	0.324	30.16	0.159	0.1257
w/o Within-region Weighting	25.35	0.234	29.94	0.119	0.1126
w/o Temporal Weighting	26.12	0.238	29.91	0.124	0.1148

In Table 5, removing main-to-anchor feedback degrades editing and geometry, supporting SceneAnchor conditioning on the evolving denoising state. Replacing SceneAnchor with direct source-conditioning LoRA improves background preservation but weakens editing; its parameterization also differs. Without source video, background degrades most, suggesting the reference alone insufficiently constrains unedited content across views. Replacing joint geometry generation with RGB-only generation followed by VGGT degrades RGB and reconstruction but also changes the pretrained latent layout (Appendix B.1). Separating edit and background losses contributes most among loss variants, consistent with counteracting background dominance. Within-region and temporal weighting further help by emphasizing larger latent changes and positions farther from the reference in time. Text removal has less impact than removing source observations or region decomposition. Adding estimated source geometry worsens edit LPIPS and CD without helping background (Table 8), so richer inputs need not improve conditioning. Source-corruption robustness is evaluated in Appendix B.8.

6 Conclusion

We presented JointEdit3D, a feed-forward framework that jointly predicts edited appearance and geometry in a pretrained unified RGB-geometry latent space. Asymmetric latent inpainting propagates a single reference edit along the source trajectory, while the SceneAnchor Branch and edit-aware losses balance edit fidelity with preservation of unedited content. We also introduced SceneEdit3D-15K and SceneEdit3D-Bench to support paired training and standardized evaluation of edit fidelity, background preservation, and 3D structure. Experiments show improved edited-region quality and structural completeness with competitive background preservation and efficient inference. Extending quantitative evaluation beyond real-scene object removal and improving robustness to imperfect references remain directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools to help construct SceneEdit3D-15K: a vision–language model proposes renderer-executable deletion and relocation targets from scene renderings and layout metadata. We have not used generative AI tools to formulate mathematical claims, implement JointEdit3D, clean or reformat evaluation data, or interpret the reported quantitative results, and the remaining required disclosure tasks (theoretical proofs, survey or interview instruments, and qualitative thematic analysis) are not applicable. Additionally, we used an image editing model to obtain some edited reference frames for real-scene qualitative examples, and we used generative AI tools to draft and edit manuscript text, improve readability, check internal consistency of tables and captions, format bibliography entries, and discuss presentation choices. All AI-assisted writing was reviewed by the authors. VLM-proposed dataset tasks are retained only after deterministic Blender validation, and all numbers, claims, and figures in this paper were verified by the authors. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

The model architecture and training objective are described in Section 3. Dataset generation and annotations are detailed in Section 4 and Appendix C, while the training configuration, evaluation protocol, and metrics are reported in Section 5.1 and Appendices B.1, D.3, and D.4.

REFERENCES

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, Chaohui Yu, Fan Wang, Xiangyang Xue, and Yanwei Fu. MVInpainter: Learning multi-view consistent inpainting to bridge 2d and 3d editing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=XIScpCMUse.

Chenjie Cao, Jingkai Zhou, Shikai Li, Jingyun Liang, Chaohui Yu, Fan Wang, Xiangyang Xue, and Yanwei Fu. Uni3c: Unifying precisely 3d-enhanced camera and human motion controls for video generation. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

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 (CVPR)*, 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.

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, 2024b.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10933–10942, 2021.

Yang Fu, Yike Zheng, Ziyun Dai, and Henghui Ding. Effecterase: Joint video object removal and insertion for high-quality effect erasing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 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.

Jiaxin Huang, Yuanbo Yang, Bangbang Yang, Lin Ma, Yuewen Ma, and Yiyi Liao. Gen3r: 3d scene generation meets feed-forward reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Xuancheng Jin, Rengan Xie, Wenting Zheng, Rui Wang, Hujun Bao, and Yuchi Huo. Fuse3d: Generating 3d assets controlled by multi-image fusion. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pp. 1–12, 2025.

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

Dong In Lee, Hyeongcheol Park, Jiyoung Seo, Eunbyung Park, Hyunje Park, Ha Dam Baek, Sangheon Shin, Sangmin Kim, and Sangpil Kim. Editsplat: Multi-view fusion and attention-guided optimization for view-consistent 3d scene editing with 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 11135–11145, 2025.

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 *Thirteenth International Conference on 3D Vision*, 2026. URL https://openreview.net/forum?id=UhHNN5lW67.

Xinyang Li, Zhangyu Lai, Linning Xu, Yansong Qu, Liujuan Cao, Shengchuan Zhang, Bo Dai, and Rongrong Ji. Director3d: Real-world camera trajectory and 3d scene generation from text. *Advances in neural information processing systems*, 37:75125–75151, 2024.

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.

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. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jiageng Liu, Weijie Lyu, Xueting Li, Yejie Guo, and Ming-Hsuan Yang. Edit3r: Instant 3d scene editing from sparse unposed images. *arXiv preprint arXiv:2512.25071*, 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*, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20669–20679, 2023.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(5):4296–4304, 2024. doi: 10.1609/aaai.v38i5.28226.

Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 724–732, 2016.

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.

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 *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10901–10911, 2021.

Xuanchi Ren, Tianchang Shen, Jiahui Huang, Huan Ling, Yifan Lu, Merlin Nimier-David, Thomas Müller, Alexander Keller, Sanja Fidler, and Jun Gao. Gen3c: 3d-informed world-consistent video generation with precise camera control. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6121–6132, 2025.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10912–10922, 2021.

Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 5294–5306, 2025.

Shaoxiang Wang, Shihong Zhang, Christen Millerdurai, Rüdiger Westermann, Didier Stricker, and Alain Pagani. Inpaint360GS: Efficient object-aware 3D inpainting via gaussian splatting for 360-degree scenes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2026.

Yuxin Wang, Qianyi Wu, Guofeng Zhang, and Dan Xu. Learning 3d geometry and feature consistent gaussian splatting for object removal. In *European conference on computer vision*, pp. 1–17. Springer, 2024.

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, 2025.

Haoyu Wu, Diankun Wu, Tianyu He, Junliang Guo, Yang Ye, Yueqi Duan, and Jiang Bian. Geometry forcing: Marrying video diffusion and 3d representation for consistent world modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ULXYZCms41.

Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European conference on computer vision*, pp. 55–71. Springer, 2024.

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*, pp. 21469–21480, 2025.

Yuanbo Yang, Jiahao Shao, Xinyang Li, Yujun Shen, Andreas Geiger, and Yiyi Liao. Prometheus: 3d-aware latent diffusion models for feed-forward text-to-3d scene generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2857–2869, 2025.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *arXiv preprint arXiv:2308.06721*, 2023.

Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pp. 162–179. Springer, 2024.

Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12–22, 2023.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. Magicbrush: A manually annotated dataset for instruction-guided image editing. *Advances in Neural Information Processing Systems*, 36:31428–31449, 2023a.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 3836–3847, 2023b.

Qihang Zhang, Yinghao Xu, Chaoyang Wang, Hsin-Ying Lee, Gordon Wetzstein, Bolei Zhou, and Ceyuan Yang. 3ditscene: Editing any scene via language-guided disentangled gaussian splatting. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=iKDbLpVgQc.

Qihang Zhang, Shuangfei Zhai, Miguel Angel Bautista Martin, Kevin Miao, Alexander Toshev, Joshua Susskind, and Jiatao Gu. World-consistent video diffusion with explicit 3d modeling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 21685–21695, 2025b.

Jensen 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12405–12414, 2025.

Xiaoming Zhu, Xu Huang, Qinghongbing Xie, Zhi Deng, Junsheng Yu, Yirui Guan, Zhongyuan Liu, Lin Zhu, Qijun Zhao, Ligang Liu, et al. Imaginarium: Vision-guided high-quality 3d scene layout generation. *ACM Transactions on Graphics (TOG)*, 44(6):1–24, 2025.

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. *ACM Transactions on Graphics (ToG)*, 43(4):1–12, 2024.

A Response Diagnostics and Calibration

This section visualizes the complementary latent cues used for region-decomposed supervision, the denoiser response to the edited reference, and threshold calibration. The latent-difference masks are used only during training.

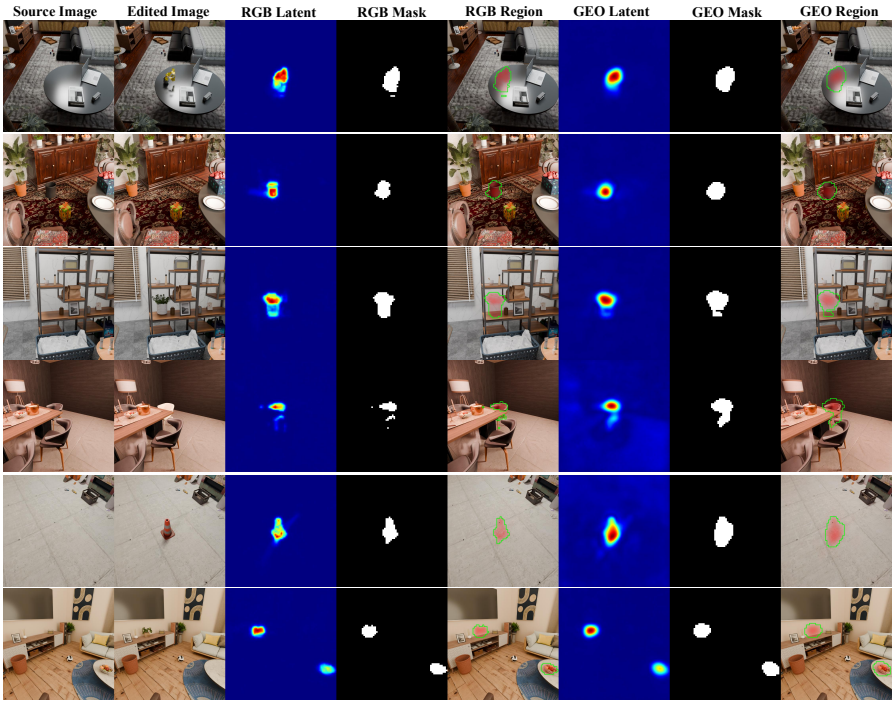

Figure 6: Latent-difference visualization for region-decomposed supervision. RGB and geometry latent deltas provide complementary edit cues, which are converted into latent masks for separating edited and background regions.

Geometry deltas emphasize structural changes around the edited object, while RGB deltas capture appearance changes that extend beyond its silhouette. These complementary cues allow supervision to account for both the object edit and its surrounding effects. We therefore derive training masks from latent differences and assign larger within-region weights to positions with larger changes.

A.1 Edit-Condition Impact Visualization

To visualize where the edited reference affects generation, we fix the same noisy latent, timestep, prompt, and source-video control branch, and compare two denoising forward passes: one with the full edit condition and one with the edited reference condition removed. We compute

$$\Delta_{\text{edit}} = G_{\theta,\pi}(z_t, c_{\text{source}}, c_{\text{edit}}) - G_{\theta,\pi}(z_t, c_{\text{source}}, c_{\text{no ref}}), \qquad M_{\text{edit}} = \text{mean}_c(|\Delta_{\text{edit}}|),$$

then upsample and normalize M_{edit} to obtain the heatmap shown in Figure 7.

Both passes retain the source-video branch and prompt; $c_{\text{no ref}}$ denotes removal of the edited reference condition only. The heatmaps show that the edited reference guides denoiser responses toward the intended edit and nearby affected context.

A.2 Threshold Calibration for Edit/Background Separation

We calibrate the binary edit-region threshold on a held-out subset with renderer-provided object masks. Let G denote the affected-object mask. For each threshold τ, we obtain a predicted edit mask from the normalized latent difference and measure object-region coverage and responses outside the

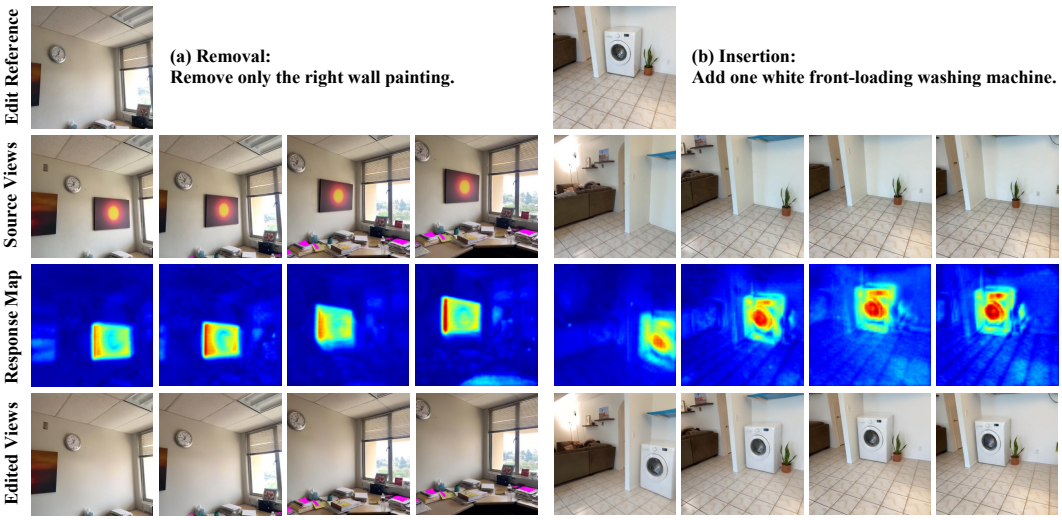

Figure 7: Edit-condition impact visualization. Heatmaps show the denoiser response difference with and without the edited reference condition, keeping the prompt and source-video branch fixed. Warmer regions indicate where the edited reference condition most strongly changes the model prediction.

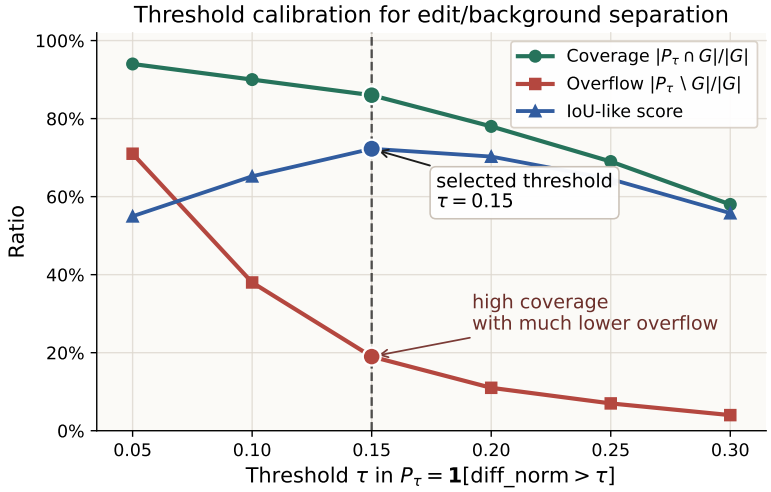

Figure 8: Threshold calibration for latent-difference edit/background decomposition over $\tau \in \{0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$. We choose $\tau = 0.15$, near the coverage–overflow elbow.

object mask:

$$P_\tau = \mathbf{1}[\text{diff_norm} > \tau], \qquad \text{Coverage} = \frac{|P_\tau \cap G|}{|G|},$$

$$\text{Overflow} = \frac{|P_\tau \setminus G|}{|G|}, \qquad \text{Score} = \frac{\text{Coverage}}{1 + \text{Overflow}}.$$

For each mask pair, Score equals the intersection-over-union $|P_\tau \cap G|/|P_\tau \cup G|$. Overflow measures responses outside the object mask, including valid secondary effects; it is a spatial calibration measure rather than an editing-error rate. We set $\tau = 0.15$, near the coverage–overflow elbow, retaining high object coverage while limiting spatial spillover. Larger thresholds suppress weaker edit responses.

B Additional Experimental Results

B.1 Implementation Details

Input representation and conditioning. All videos are represented as 49 RGB frames at 560×560 resolution. The Wan video VAE encodes each sequence into 16-channel RGB latents with 13 temporal steps and 70×70 spatial size. Following Gen3R, the geometry stream is encoded by frozen VGGT and Geometry Adapter modules; we concatenate RGB and geometry latents along the width dimension, producing a target latent of shape $16\times13\times70\times140$. Each sample provides one edited reference frame I_e at zero-based clip index $r \in \{0, \dots, 48\}$. For synthetic data, this frame is selected from views with high edit-mask coverage. Under Wan's causal temporal packing, latent step 0 corresponds to the first frame, while step $j \geq 1$ packs frames $4j-3$ through $4j$. Thus, $k = \lceil r/4 \rceil$, including $k = 0$ for $r = 0$. We encode I_e alone and place its latent at step k; the clip is not reordered. To construct the four-channel observation mask, we mark frame r in pixel time, repeat the first-frame entry four times, and pack consecutive groups of four into channels. For $r = 0$, all four channels at step 0 are marked; for $r > 0$, only channel $(r-1) \bmod 4$ at step k is marked. These channels encode a single observed RGB frame. Remaining RGB steps and the full geometry half stay masked. The source-video RGB latent, concatenated with a zero geometry half, serves as the SceneAnchor Branch control input.

The single-image encoding is an auxiliary condition in $\mathbf{y}$ (Equation 4); it is not substituted into the noisy state $\mathbf{z}_t$ or the target video latent. Training supervises the full edited-video latent.

Training. For the full JointEdit3D model, we train only the SceneAnchor Branch and keep the Gen3R-tuned Wan transformer, VAE, text encoder, CLIP image encoder, VGGT, and Geometry Adapter frozen. Cached latents and text/image features are used to reduce training cost. The main model is trained for 10K steps on 8 NVIDIA H200 GPUs with batch size 10 per device, bf16 mixed precision, gradient checkpointing, AdamW with learning rate 10^{-4}, cosine decay, and EMA. We apply light condition dropout to the edited image and text prompt to improve robustness.

The main results use the 10K-step model. Due to computational constraints, all component ablations in Table 5 use a common 2K-step schedule, including the full-model baseline.

Architecture and ablations. SceneAnchor blocks are injected at layers $\mathcal{S} = \{0, 4, 8, 12, 16, 20, 24, 28\}$. The "Direct source conditioning (rank-16 LoRA)" variant in Table 5 replaces SceneAnchor with direct source conditioning and adapts the frozen Gen3R-tuned transformer using rank-16 LoRA. Both designs are trained for 2K steps, with different conditioning pathways and trainable parameterizations: 70M SceneAnchor parameters for the full model versus LoRA adapters for direct conditioning. Appendix B.6 separately tests main-state feedback within the SceneAnchor architecture. The "w/o GEO in Wan Flow (+VGGT)" variant generates only RGB latents, reducing the grid from $16 \times 13 \times 70 \times 140$ to $16 \times 13 \times 70 \times 70$, and reconstructs its RGB output using the VGGT backend from Table 4. This ablation changes both the generated modalities and the pretrained latent layout.

Temporal weighting. The training-only temporal weight is

$$a_i = 1 + \eta \frac{|\rho(i) - k|}{\max_j |\rho(j) - k| + \varepsilon_{\text{den}}}, \qquad \tilde{l}_i = a_i l_i, \tag{10}$$

where $\rho(i)$ is the latent temporal index of position i and k is the reference step. The weight emphasizes positions farther from the reference in time, with $a_i \in [1, 1.5]$ for $\eta = 0.5$. Temporal distance is a proxy for viewpoint change and does not account for revisits or loop closures. The remaining loss settings are τ=0.15, α=5.0, λ=0.5, and γ_{geo}=1.5.

Inference. Inference uses the same 49-frame preprocessing and single-reference conditioning. Unless otherwise stated, quantitative evaluation uses FlowMatch Euler sampling with 5 denoising steps, classifier-free guidance scale 1.0, control scale 1.0, seed 42, and zero camera embeddings. Zero camera embeddings disable explicit camera guidance; the ordered source RGB observations provide trajectory evidence through SceneAnchor, alongside the reference and pretrained model priors. Camera predictions are outputs rather than input pose constraints. After denoising, the RGB half is

decoded by the Wan VAE, and the geometry half is decoded by the Geometry Adapter to recover depth, point maps, valid masks, and camera predictions.

B.2 Hyperparameter Sensitivity

Parameter and checkpoint selection. Loss hyperparameters and the checkpoint selection rule are fixed before evaluation, with τ calibrated separately in Appendix A.2. Table 6 varies one hyperparameter at a time while keeping the others at their defaults. All variants follow the 2K-step protocol of Table 5 and are evaluated on the 100-sample SceneEdit3D-Bench test set.

Table 6: Hyperparameter sensitivity on SceneEdit3D-Bench. Each block varies one objective hyperparameter around the default setting used by JointEdit3D.

		Edit Region		Background		3D Structure
Hyperparameter	Value	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓	CD↓
Within-region scale α	2.5	25.71	0.252	30.02	0.129	0.1184
	5.0	26.82	0.209	30.69	0.103	0.1053
	7.5	26.18	0.228	29.93	0.126	0.1136
Temporal scale η	0.25	25.96	0.239	29.87	0.131	0.1161
	0.50	26.82	0.209	30.69	0.103	0.1053
	0.75	26.20	0.229	30.05	0.123	0.1124
Edit/background balance λ	0.40	25.89	0.247	30.91	0.096	0.1145
	0.50	26.82	0.209	30.69	0.103	0.1053
	0.60	26.47	0.221	29.86	0.132	0.1118
Geometry loss weight γ_{geo}	1.0	26.19	0.231	30.12	0.124	0.1208
	1.5	26.82	0.209	30.69	0.103	0.1053
	2.0	25.98	0.238	29.95	0.128	0.1109

The default settings achieve the best edited-region quality and geometry accuracy across the tested values. Reducing λ favors background preservation at the expense of edit fidelity, reflecting its role in balancing the two regions. Intermediate values of α, η, and γ_{geo} outperform both lower and higher settings, supporting balanced emphasis on latent changes, temporal distance, and geometry supervision.

B.3 Additional Qualitative Comparisons

Relocation, multi-operation, and deletion comparisons are in Figures 5 and 4. The figures below add appearance, occluded removal, multi-reference editing, and native point clouds on DAVIS, ScanNet++, CO3D, and casual captures (Perazzi et al., 2016; Yeshwanth et al., 2023; Reizenstein et al., 2021). Unless otherwise stated, real-scene examples use a cropped clip, one randomly selected keyframe, and a Nano Banana edit of that frame, demonstrating transfer from synthetic training to real captures.

Figure 9 recolors a backpack on DAVIS. All methods receive the same edited reference; MVInpainter additionally receives a SAM3 backpack mask, while Omni-3DEdit and JointEdit3D use no input mask. MVInpainter removes the backpack instead of recoloring it. Omni-3DEdit produces inconsistent backpack colors across frames and alters the background appearance. JointEdit3D maintains the red backpack across views while preserving the person and background.

Figure 10 removes a monitor cluster on ScanNet++ while keeping the bottle behind it. MVInpainter replaces the cluster with blur, while Omni-3DEdit leaves monitors or cables in some views. JointEdit3D removes the foreground monitors, preserves the bottle behind them, and restores continuous cabinet and wall surfaces across views.

Although trained with a single edited reference, JointEdit3D accepts multiple edited references at inference by placing their encodings at the corresponding conditioning positions. Figure 11 demonstrates combined removal and insertion across different parts of a room, propagating both reference edits while preserving the surrounding scene.

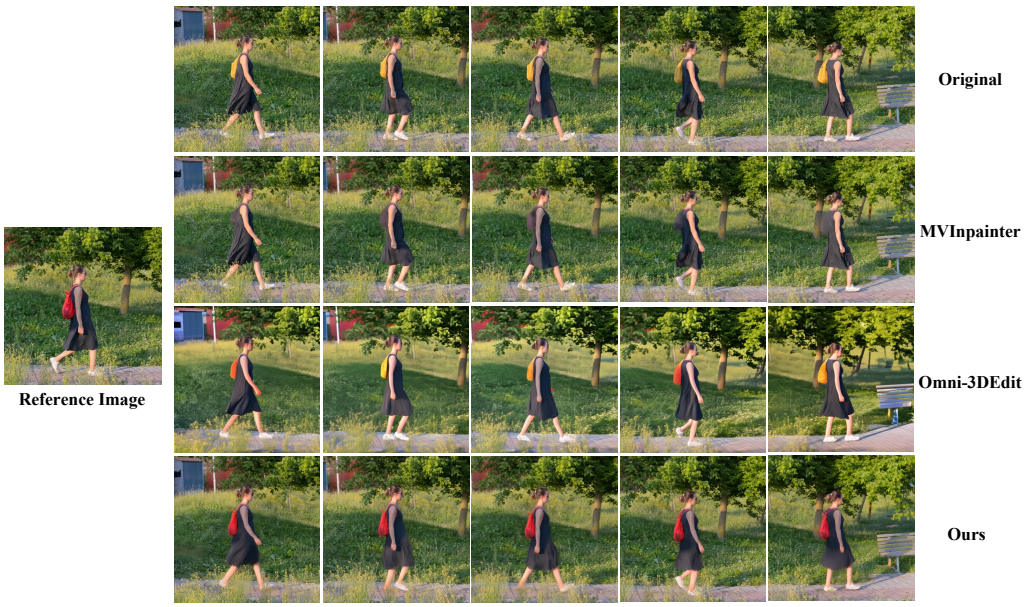

Figure 9: Reference-guided backpack recoloring on DAVIS.



Figure 10: Monitor removal on ScanNet++ with preservation of the bottle behind the monitors.

Figure 12 shows the source clip, the edited sequence, and the decoded point cloud on ScanNet++ and casual indoor captures. Object additions and removals appear consistently in the generated views and native point clouds, while the surrounding furniture layout is preserved.

B.4 Operation-Specific Diagnostics

Table 7 complements the main evaluation with operation-wise SSIM and depth metrics. JointEdit3D achieves depth threshold accuracy above 93% across all five operations. Relocation has the

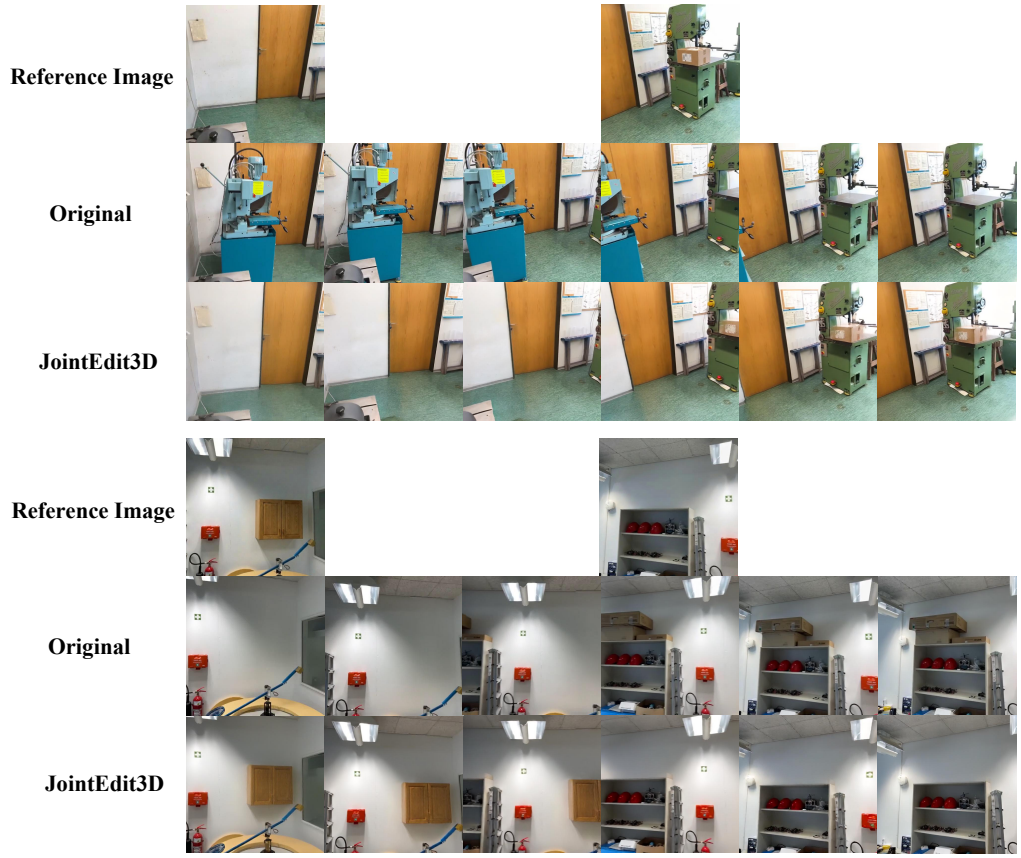

Figure 11: Multi-reference editing in real scenes: removing a machine and adding a box (top), and adding a wall cabinet and removing boxes above the shelving unit (bottom).

largest depth error, while multi-operation editing yields the lowest edited-region SSIM, reflecting the additional demands of changing spatial layout and coordinating multiple edits.

Table 7: Supplementary operation-specific SSIM and depth metrics for JointEdit3D on SceneEdit3D-Bench.

Operation	Edit SSIM↑	Bg. SSIM↑	Abs Rel↓	δ<1.25 ↑
Delete	0.836	0.914	0.0540	0.9819
Add	0.698	0.921	0.0592	0.9741
Move	0.698	0.913	0.0858	0.9305
Appearance	0.840	0.926	0.0606	0.9719
Multi-op	0.672	0.892	0.0683	0.9628

B.5 Region-wise Geometry and Confidence Filtering

Regional geometry metrics in Table 4 use renderer-defined edit and background regions with the same full-scene Sim(3) alignment. JointEdit3D lowers CD in both regions relative to Omni-3DEdit + VGGT, supporting structural editing together with scene preservation.

Confidence filtering controls the accuracy–completeness trade-off of the native point cloud. At a threshold of 0.85, filtering lowers prediction-to-reference error relative to the unfiltered output while reducing coverage. The filtered output outperforms *JointEdit3D RGB + VGGT* on all five metrics reported in Table 4. The threshold was adjusted using renderer-referenced geometry scores, so the filtered row is marked as a post-hoc diagnostic (†) and excluded from ranking.

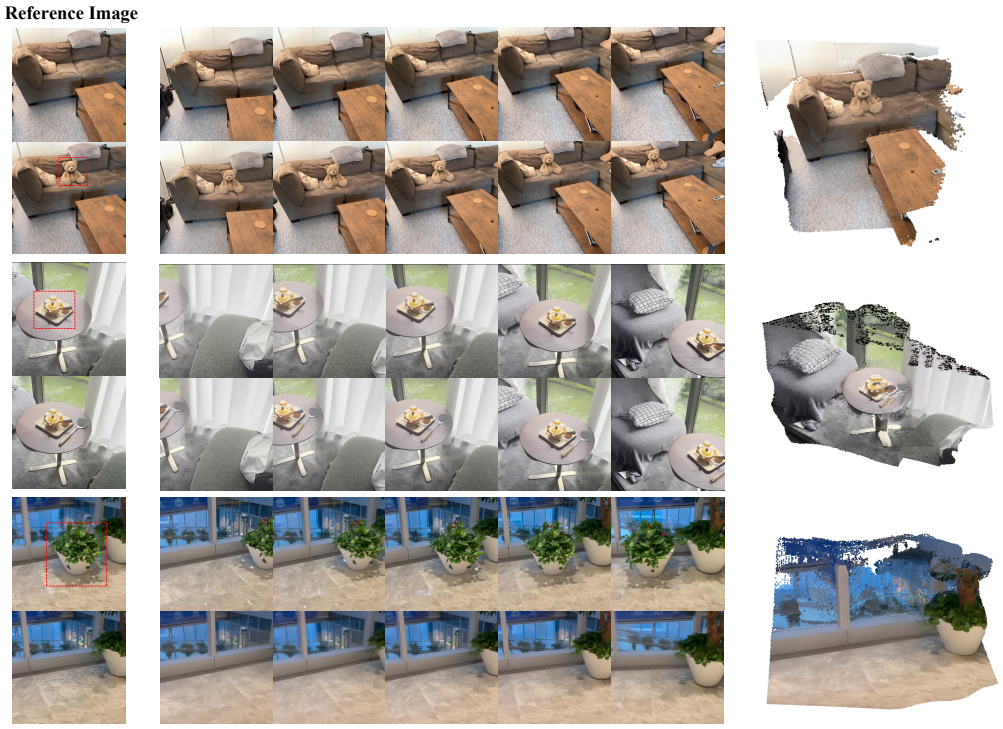

Figure 12: Real-scene editing with jointly generated RGB views and native point clouds on ScanNet++ and casual indoor captures.

B.6 Main-State Feedback Ablation

The feedback ablation in Table 5 tests the main-state-to-SceneAnchor feedback term in Equation 5. Introduce a fixed gate $b \in \{0, 1\}$:

$$\mathbf{c}_j = \mathrm{AnchorBlock}_j\Big(\mathbf{c}_{j-1} + b\, W_{\mathrm{in}}^{(j)}(\mathbf{x}_{\ell_j}),\ W_e^{(j)}(\mathbf{e}),\ W_{\mathrm{ctx}}^{(j)}(\mathbf{ctx})\Big). \tag{11}$$

The full model uses $b = 1$; the feedback-disabled variant uses $b = 0$ at every injected layer throughout both training and inference. Source RGB conditioning, timestep and image/text context, and the anchor-to-main residual updates in Equation 6 remain enabled.

Both configurations use the same pretrained checkpoint, corresponding parameter initialization, branch width and depth, injection layers, zero-initialized output projections, data order, random seeds, optimizer, batch size, losses, 2K-step schedule, and evaluation settings. Results are reported only in Table 5, alongside the existing 2K-step full-model reference.

B.7 RGB versus RGB+Geometry SceneAnchor Input

Table 8 tests whether adding estimated source geometry improves SceneAnchor conditioning under the 2K-step protocol of Table 5. The full model uses $[\mathbf{z}_{\mathrm{src}}^{\mathrm{rgb}};\ 0]$; the alternative fills the geometry half with an estimated source geometry latent, $\hat{\mathbf{z}}_{\mathrm{src}}^{\mathrm{geo}}$. Both variants retain joint RGB-geometry generation, so this experiment examines source conditioning rather than removal of the generated geometry stream.

RGB-only conditioning achieves better edit fidelity and geometry quality, with comparable background preservation. Explicit source geometry may reinforce the original structure in regions that need to change, making reference-guided structural edits harder. These results support using RGB observations for source-scene anchoring while allowing the joint generator to predict the edited geometry.

Table 8: SceneAnchor input ablation under the 2K-step protocol of Table 5.

SceneAnchor input	Edit LPIPS↓	Bg. LPIPS↓	CD↓
$[\mathbf{z}_{\text{src}}^{\text{rgb}};\, 0]$	0.209	0.103	0.1053
$[\mathbf{z}_{\text{src}}^{\text{rgb}};\, \hat{\mathbf{z}}_{\text{src}}^{\text{geo}}]$	0.284	0.105	0.1124

B.8 Source-Observation Robustness

We corrupt only SceneEdit3D-Bench source videos (motion blur, sensor-like noise, JPEG). The edited reference and the evaluation target stay clean, so Table 9 measures the editor's sensitivity to source degradation with a clean reference.

Table 9: Native geometry under source-only corruption on SceneEdit3D-Bench.

Source input	Acc.↓	Comp.↓	CD↓	F-score↑
Clean	0.1153	0.0886	0.1020	0.9702
Motion blur	0.1243	0.0891	0.1067	0.9627
Sensor-like noise	0.1218	0.0894	0.1056	0.9615
JPEG artifacts	0.1193	0.0884	0.1039	0.9649

JointEdit3D maintains stable geometry under the tested source corruptions, with modest changes in CD and F-score. Completeness remains nearly unchanged, indicating that source degradation primarily affects point accuracy rather than scene coverage.

B.9 Runtime and Memory Details

Table 10 provides the runtime breakdown and peak memory for the 49-frame full-3D setting in Figure 3. Inference runs on a single NVIDIA H200 GPU. RGB/video baselines include VGGT reconstruction as post-processing; optimization methods include per-scene fitting.

Table 10: Runtime and memory details for the 49-frame full-3D efficiency comparison.

Method	Time (s)	Gen./Opt. (s)	Post. (s)	Peak alloc. (GB)	Peak reserved (GB)
JointEdit3D (Ours)	11.94	11.94	0.00	29.07	50.33
MVInpainter (Cao et al., 2024)	43.25	19.45	23.81	43.80	66.88
Omni-3DEdit (Chen et al., 2026)	237.15	213.34	23.81	56.40	73.08
SPIn-NeRF (Mirzaei et al., 2023)	1619.00	1619.00	0.00	–	–
Gaussian Grouping (Ye et al., 2024)	486.00	486.00	0.00	–	–
GaussianEditor (Chen et al., 2024b)	285.07	285.07	0.00	–	–

JointEdit3D uses five denoising steps followed by RGB and geometry decoding in one inference run, with no per-scene optimization or VGGT reconstruction post-process. Memory is blank where the timing logs did not record peak allocation.

C SceneEdit3D-15K Generation Details

C.1 Scene Sources and Candidate Extraction

We build SceneEdit3D-15K from composable indoor scenes in Imaginarium (Zhu et al., 2025). Each scene is provided as a Blender file with object-level meshes, parent-child relations, physically based materials, lighting, camera metadata, and captions, so edits change the underlying 3D scene state rather than rendered images. We complete missing structural elements (e.g., walls and ceilings) to maintain complete indoor structure along the rendered camera trajectories, and we build an object table from scene metadata and rendered visibility statistics, recording each candidate's exact Blender name, semantic class, caption, parent, number of children, visible image area, maximum 3D extent, height, and support-likeness; structural objects (floors, ceilings, cameras, walls) are excluded from the editable set.

C.2 VLM-Guided Edit Proposal

We use GPT-5.5 as the VLM to propose canonical deletion and relocation tasks. Given scene renderings and a JSON layout file derived from the scene graph, the VLM analyzes object visibility,

Representative task-proposal prompt

System. You are a senior 3D scene edit planner. Your output is consumed by an automatic Blender renderer, so object names, operation names, and JSON syntax must be exact. Return strict JSON only; do not output markdown or natural language outside JSON.

Inputs. Inspect the rendered scene views and the accompanying JSON layout file before selecting tasks. Use the renderings to judge visibility, occlusion, room boundaries, and visual plausibility, and use the JSON layout to reason about object names, support, containment, proximity, and parent-child relations.

Task goal. Generate exactly K realistic, visually meaningful, and physically plausible canonical tasks. The allowed operations in this prompt are `delete` and `move`; do not generate `add`, `move_reverse`, or appearance-change tasks here. These inverse or appearance tasks are produced later by the deterministic rendering/augmentation pipeline. Use only object names from the JSON layout file, copied character-for-character. Diversify object classes, scene regions, and object scales.

Operation-specific rules. *Delete:* choose a clearly visible and semantically meaningful removable object; avoid structural elements, room-defining objects that are too large, tiny objects whose removal is visually ambiguous, and support objects with many children unless deleting the whole group is plausible. Prefer objects whose removal creates a visible but physically plausible empty region. *Move:* choose a non-support leaf object with visible free space nearby; move it only a moderate distance within the same functional area, keep it supported after moving, avoid collisions or wall penetration, and ensure the object is visible both before and after the move. *Appearance:* do not select appearance tasks in this prompt; material/texture changes are sampled by Blender from compatible material or texture candidates.

Trajectory rule. Select one template from `trajectory_templates`. The selected trajectory should view the affected objects clearly and avoid severe occlusion; include a short reason.

Output schema.

```
{"tasks": [{"operation": "delete|move",
  "target_object": "exact_object_name",
  "edit_text_en": "short edit description",
  "instruction": "renderer-facing concise instruction",
  "trajectory_reason": "why this view works",
  "trajectory_template": "one provided template"}]}
```

Figure 13: Representative VLM prompt for proposing canonical renderer-executable tasks. The VLM analyzes rendered scene views and a JSON layout file before selecting deletion and relocation targets under physical and visibility constraints. Add and move-reverse samples are derived later by state/time reversal; appearance changes use Blender-side material/texture replacement.

physical support, free space, and scene relations, then selects deletion and relocation targets with exact object names and proposes a camera-trajectory template. For relocation, Blender samples and validates the final target position under geometric constraints. Addition and move-reverse samples are obtained by reversing valid deletion and move pairs, and appearance changes are generated separately in Blender by replacing a target object's material or texture with a compatible alternative. Figure 13 shows the task-proposal prompt; Figure 14 shows instruction refinement performed after rendering. These representative templates omit implementation-specific examples and repair logs for readability.

Raw VLM responses are validated rather than accepted directly: each parsed task is checked against the candidate table and layout and is rejected for unknown object names, repeated targets, violated operation-specific constraints, deletion targets that are too small or structurally important, or move targets without sufficient nearby free space and support. When too few tasks are valid, a repair prompt returns the rejection reasons and the previous response, asking the VLM to keep valid tasks and replace only invalid ones; remaining gaps are filled by rule-based diversification over the validated candidate pool.

C.3 Deterministic Scene Execution

Accepted canonical tasks are executed in Blender under a deterministic before/after protocol, and inverse tasks are derived by state/time reversal. Deletion hides the target object and its descendants in the edited scene (child lights are preserved to keep light-source settings fixed); the addition sample swaps the source and edited states and reverses the rendered sequence. Relocation keeps the target visible in both scenes and moves it to a new collision-free position within the navigable room

Editing-instruction refinement prompt

Role. You are generating natural editing instructions to fine-tune a video diffusion model for scene editing. You are shown three rendered inputs from the same viewpoint: **before editing**, **after editing**, and an **edit-region mask** where white marks the changed area. You also receive the operation type and a draft renderer instruction.

Visual analysis. Compare the before/after images and use the mask only to localize the change. Identify the edited object, its appearance, nearby landmarks, support surface, and the visible result after the edit. Do not mention the mask, frame index, image pair, renderer, or any technical term in the final instruction.

Prompt style. Generate one forward instruction and one reverse instruction. Each prompt should be 15–40 words, start with an action verb such as remove, add, move, or replace, include spatial context, describe the visual outcome, and include color/material/size cues when visible.

Output schema.

```
{"forward": "before-to-after editing prompt",
 "reverse": "after-to-before editing prompt"}
```

Figure 14: Representative prompt for converting rendered before/after evidence into natural user-facing editing instructions. This stage grounds the language in the actual rendered change rather than only in object identifiers.

bounds, with the reverse sample derived by swapping states and reversing frame order. Appearance changes are sampled by Blender without VLM proposal, replacing a material or texture with a compatible alternative while preserving mesh and transform. Multi-op samples apply two or more atomic operators to *distinct* objects in one execution; same-object conflicts (e.g., move then delete the same instance) are rejected, and the camera path is centered on the affected set.

Camera trajectories are generated from fixed templates (horizontal sweeps, diagonal forward, rising, and drop-forward motions) centered on the edited object or, for composite edits, on the midpoint of the affected objects. Relocation candidates are tested for room-bound validity, collision, support plausibility, and visibility from the final path, and tasks whose targets are poorly visible are rejected or repaired, so the source and target videos differ primarily by the intended edit rather than by accidental occlusion or camera drift.

C.4 Rendering, Annotations, and Prompt Refinement

Every accepted edit is rendered as a source video and an edited target video with identical camera poses, intrinsics, frame count, resolution, lighting configuration, and sampling settings, so frame i of the two sequences forms a strictly paired before/after observation. We additionally export camera intrinsics, relative poses, depth maps, and task metadata. The edited reference frame I_e is selected from the target sequence and paired with the final instruction p as the conditioning input used by JointEdit3D.

Edit masks are rendered through Blender object-index passes and track the affected object: deleted or relocated objects, the added object in the edited branch, or the material-replaced object; for composite edits, branch-specific masks are combined when a single edit-region mask is needed for prompt refinement or evaluation. Because the masks come from the renderer, they align exactly with the RGB and depth frames and require no separate segmentation model.

The final language instruction is refined after rendering: a VLM receives the edited reference frame, its source counterpart at the same viewpoint, and the combined mask, and generates forward and reverse prompts in a compact JSON format. The refined instructions describe the rendered edit and its visible outcome, converting renderer-facing identifiers into natural language with spatial context and appearance cues. For example, "remove object chair_12" becomes "Remove the brown lounge chair beside the sofa, revealing the wooden floor and wall behind it." The before/after pair and mask support offline instruction generation; the editor receives the source RGB video, edited reference, and final instruction.

C.5 Dataset Organization and Release Format

The dataset is organized into separate JSONL manifests for training, validation, and test. Each entry records a sample ID, RGB/mask and geometry shard keys, the edited-reference frame index, an edit

Algorithm 1: Deterministic Blender Execution for One Edit

Input: Blender scene $\mathcal{S}_0$, metadata $\mathcal{M}$, edit task q, JSON layout file $\mathcal{L}$, seed s. For an atomic edit, q is a validated delete/move proposal or a Blender-sampled appearance task.
Output: paired source/target videos, masks, depth maps, camera poses, intrinsics, and task metadata.

1. Parse the JSON layout file $\mathcal{L}$ and scene metadata $\mathcal{M}$ to recover object identities, support relations, parent-child links, approximate object extents, and navigable room bounds.
2. Resolve affected object roots from q, expand them with the parent-child graph, and run layout validity checks to reject structural, too-small, or unsupported targets.
3. For relocation, sample candidate target positions and run collision detection, support-surface checks, room-bound checks, and free-space validation before accepting a placement.
4. Generate a seeded camera trajectory $P = \{(R_i, t_i)\}_{i=1}^{N}$ from fixed templates such as horizontal sweeps, diagonal motions, rising motions, and drop-forward motions, centered on the affected object set.
5. Run operation-specific visibility checks along P: deletion requires target visibility before editing, while relocation and appearance changes require visibility in both states. Reject or repair tasks that fail these checks; inverse additions inherit visibility from the deletion source.
6. Save the initial object states and camera state so both branches start from the same scene.
7. For each branch $b \in \{\text{before}, \text{after}\}$: restore the saved scene state, apply the branch-specific edit operator E_q^b, attach the same camera path P, and render RGB/depth frames.
8. Select branch-specific mask objects: deleted objects for deletion, moved objects in both relocation branches, and appearance-changed objects in both appearance branches. Render binary masks through Blender object-index passes.
9. Export RGB videos, depth maps, masks, relative camera poses, intrinsics, task JSON, and completion sentinels.
10. Derive inverse samples for deletion and relocation by swapping before/after branches and reversing frame, depth, mask, and pose order: delete $\rightarrow$ add and move $\rightarrow$ move-reverse.
11. Discard samples with missing files, corrupted frames, invalid physical placement, failed operation-specific visibility checks, or empty masks in branches where the edited object should be visible.

$$E_q^b(\mathcal{S}_0) = \begin{cases} \text{hide}(o), & q = \text{delete},\ b = \text{after}, \\ \text{move}(o, \Delta), & q = \text{move},\ b = \text{after}, \\ \text{replace_mat}(o, m'), & q = \text{appearance},\ b = \text{after}, \\ \text{identity}, & \text{otherwise}. \end{cases}$$

Figure 15: Algorithmic view of the deterministic Blender execution protocol. The operators shown define atomic edits; multi-operation tasks compose operators on distinct objects. Both rendering branches reuse the same accepted camera path and render settings to produce paired observations under identical viewpoints.

prompt, and the number of frames. RGB frames and masks are stored in one shard family; depth maps and camera parameters are stored in the corresponding geometry shards under the same sample ID. The dataset license is CC BY-NC-SA 4.0 with attribution to the Imaginarium (Zhu et al., 2025) authors, matching the source scenes' ShareAlike terms.

C.6 Training and Evaluation Data Flow

SceneEdit3D-15K separates the RGB pairs used for JointEdit3D training from the renderer-side 3D annotations released with the dataset. Each paired source video $\mathcal{V}_{\text{src}}$ and edited target video $\mathcal{V}_{\text{edit}}$ is converted into the unified RGB-geometry latent space of Gen3R (Huang et al., 2026): the RGB half comes from the Wan video VAE, $\mathbf{z}_{\text{edit}}^{\text{rgb}} = E_{\text{rgb}}(\mathcal{V}_{\text{edit}})$, and the geometry half from the frozen VGGT encoder Φ_{vggt} (Wang et al., 2025) and Geometry Adapter encoder, $\mathbf{z}_{\text{edit}}^{\text{geo}} = A_{\psi}^{\text{enc}}(\Phi_{\text{vggt}}(\mathcal{V}_{\text{edit}}))$. Because VGGT predicts camera-aware multi-view geometry features from RGB and the Geometry Adapter maps them onto the Wan latent grid, this yields a geometry-latent pseudo-target aligned with the RGB latent in time and space, giving $\mathbf{z}_{\text{edit}} = [\mathbf{z}_{\text{edit}}^{\text{rgb}}; \mathbf{z}_{\text{edit}}^{\text{geo}}]$. The source video is encoded in

Table 11: SceneEdit3D-15K split statistics and SceneEdit3D-Bench composition. Top: sample, scene, and operation counts; Test is a subset of the Held-out pool. Bottom: edit-mask image-area ratios measured in the randomly selected frame specified in each sample's dataset annotations.

Split	Total	Scenes	Add	Delete	Move	Appearance	Multi-op
Train	13,799	135	4,990	4,987	1,884	1,017	921
Held-out pool	1,520	16	600	600	186	24	110
Test (Bench)	100	15	29	29	14	14	14

Edited-object area ratio	$< 1\%$	1–5%	5–15%	$\geq 15\%$
Count	11	47	29	13

the same space; its RGB half feeds the SceneAnchor Branch, and its full latent is used on the loss side to compute source-target latent-difference masks and weights.

During training, the model is conditioned on the source RGB video, the edited reference frame, and the language instruction, while $\mathbf{z}_{\text{edit}}$ provides the flow-matching supervision: we corrupt it with Gaussian noise and train the editor to predict the flow-matching velocity for both its RGB and VGGT-derived geometry halves. Gradients update only the SceneAnchor Branch; the Wan VAE, VGGT encoder, Geometry Adapter, and Gen3R-tuned backbone remain frozen. The frozen VGGT–Geometry-Adapter encoding pipeline converts the edited RGB target video into geometry-latent supervision. JointEdit3D thus derives appearance and geometry-latent supervision from paired RGB videos while preserving the pretrained joint representation, without using renderer-provided depth maps, point clouds, or camera poses as direct training targets.

Renderer annotations support data preparation, calibration, and evaluation. Object masks are used for visibility and edit-area filtering, instruction refinement, reference-frame selection, and calibration of the latent-difference threshold τ. At evaluation, they define the object and background regions for image metrics. Depth maps, camera intrinsics/extrinsics, and edited scene states provide geometric references for depth and point-cloud comparisons. The training objective uses RGB latents and VGGT-derived geometry latents as targets, with region weights computed from source–target latent differences.

C.7 Quality Control and Benchmark Curation

Quality control runs at three stages: candidate objects are filtered by semantic validity, visibility, size, parent-child structure, and operation-specific safety before rendering; output completeness is checked, with completion sentinels written only after RGB frames, videos, masks, depth maps, poses, intrinsics, and metadata are generated; and samples with missing frames, corrupted files, failed operation-specific visibility checks, or empty or overly small edit regions in branches where the affected object should be visible are removed afterwards.

As Table 11 reports, SceneEdit3D-15K contains 15,319 paired samples: 13,799 training samples from 135 scenes and 1,520 held-out samples from 16 scenes absent from training. The held-out pool is partitioned into 1,420 validation samples and 100 test samples. Training and held-out data are scene-disjoint; validation and test are sample-disjoint subsets of the held-out pool, with no forward/reverse counterparts of the same edit shared across the two splits.

The test split, SceneEdit3D-Bench, draws from 15 held-out scenes and covers all five operation groups, with more samples allocated to addition and deletion. Edited-object area is measured as the renderer-defined edit-mask area divided by the image area in a randomly selected frame specified in each sample's dataset annotations. The benchmark spans sparse ($<1\%$) to large ($\geq 15\%$) edited regions and includes partially occluded, small-object, large-object, and multi-operation cases, supporting evaluation of edit fidelity, background preservation, and 3D structure under a common paired-rendering protocol.

D Evaluation Protocol Details

D.1 360-USID and Real-Scene Quantitative Protocol

We evaluate real-scene object removal on 360-USID (Wu et al., 2025). Before/after 3DGS reconstructions are used solely to obtain a shared source trajectory. The edited reference is an object-free ground-truth frame from the dataset; SAM3-propagated masks provide the inputs required by mask-guided baselines.

Metrics are computed at poses with official after-removal images, using those real images as targets and the official evaluation masks for every method. Table 2 reports edit-region PSNR/LPIPS. The scoring targets are not 3DGS renders, and SAM3-propagated input masks are not used for scoring. Preprocessing failures are the only reason for frame exclusion.

D.2 Inpaint360GS Protocol

Table 12 evaluates object removal on the 11-scene Inpaint360GS benchmark (Wang et al., 2026). Inpaint360GS uses per-scene 3DGS optimization; JointEdit3D uses five denoising steps followed by RGB–geometry decoding. Shared evaluation conditions include the initial reconstruction, scene, poses, ground truth, and metrics, while each method retains its native inference path.

For JointEdit3D, we reconstruct the unedited scene with the official Inpaint360GS 3DGS pipeline and render a source video along a trajectory covering the official `inpaint_camera` test poses. One selected object-free image serves as the editing reference; object-free images at the evaluation poses provide the scoring targets. We evaluate the generated RGB views directly at these poses using full-image and NBS (newly exposed background) metrics. The predicted point cloud remains the native geometry output, without conversion to 3DGS for scoring.

Table 12: Object removal on the 11-scene Inpaint360GS benchmark. NBS denotes the newly exposed background region. Inpaint360GS is a per-scene optimization method; JointEdit3D and Omni-3DEdit are feed-forward.

Method	PSNR↑	LPIPS↓	NBS PSNR↑	NBS LPIPS↓
Inpaint360GS (Wang et al., 2026)	24.40	0.1300	36.29	0.0078
Omni-3DEdit (Chen et al., 2026)	22.38	0.1495	36.77	0.0072
JointEdit3D	25.03	0.1227	37.24	0.0066

JointEdit3D achieves the best full-image and newly exposed-region scores among the compared methods, demonstrating effective removal and background completion on this real-scene benchmark.

D.3 Baseline Interfaces and Aggregation

All methods are mapped to a common prediction layout and scored against fixed target frames and evaluation masks. Method-specific preprocessing follows the released interfaces summarized in Table 13.

Table 1 marks unsupported properties and operations with a dash. Table 2 compares methods on deletion; Table 3 covers methods evaluated on all five operations. Each operation column averages that operation's samples. The Average column averages all 100 SceneEdit3D-Bench samples, using operation counts 29/29/14/14/14 (Delete/Add/Move/Appearance/Multi-op). The edited reference view is excluded from PSNR/LPIPS scoring on SceneEdit3D-Bench.

Evaluation regions. On SceneEdit3D-Bench, Edit denotes the renderer-defined affected-object mask, and Background denotes its complement. Both regions are evaluated against the edited target images. The Background region can include secondary effects outside the object mask, such as changed shadows and reflections; its scores measure agreement with the edited target in that region.

Shared inputs. Every method that can consume them gets the same source video and the same edited reference. If a method recommends a frame budget, we report the better of that budget and the all-frame setting; otherwise we use official defaults and do not tune on SceneEdit3D-Bench.

Table 13: Baseline interfaces used in evaluation. Methods receive source observations and editing guidance according to their native interfaces; SEVA is a protocol-mismatched novel-view-synthesis model that does not consume the source video. Evaluation always uses the fixed benchmark masks and target images.

Method	Inputs used in evaluation	Per-scene fitting	Reporting scope
SPIn-NeRF (Mirzaei et al., 2023)	Edit masks, camera poses, per-scene NeRF assets	Yes	Shared deletion table only
Gaussian Grouping (Ye et al., 2024)	Segmentation/grouping cues, camera poses, per-scene 3DGS assets	Yes	Shared deletion table only
GScream (Wang et al., 2024)	Method-required masks/poses and per-scene 3DGS assets	Yes	Shared deletion table only
GaussianEditor (Chen et al., 2024b)	Method-required masks/text/poses and per-scene 3DGS assets	Yes	Shared deletion table only
SEVA (Zhou et al., 2025)	Edited reference frame + target cameras only; no source video	No	Five-operation table as protocol-mismatched NVS row
MVInpainter (Cao et al., 2024)	Source RGB views, edited reference, edit masks	No	Deletion and five-operation comparisons
Omni-3DEdit (Chen et al., 2026)	Source RGB views, edited reference, camera/pose inputs	No	Deletion and five-operation comparisons
JointEdit3D	Source RGB video, edited reference, optional text prompt	No	Deletion and five-operation comparisons

Mask-guided methods receive the edit masks they cannot run without, and pose-conditioned methods receive benchmark cameras; JointEdit3D uses neither. An operation is protocol-mismatched when producing an output would change the task rather than just the file format—for per-scene NeRF/3DGS editors on add/appearance, that would mean plugging in an extra 2D editor to invent missing target content, which we do not score. SEVA is the one NVS diagnostic: it expands the edited reference to target cameras and never sees the source video.

Edit3r and the reference-input protocol. Edit3r (Liu et al., 2025) applies its own 2D image-editing pipeline to the input views before predicting edited 3D Gaussians. Our reference-guided quantitative comparisons instead provide an externally edited reference frame to methods that support this input. Using Edit3r's own editor would introduce a different source of edit guidance, while adapting its editing stage to enforce the supplied reference would modify the native pipeline. We therefore discuss Edit3r as related work and omit it from the shared-reference quantitative comparison.

MVInpainter on Move and Multi-op. MVInpainter is mask-based multi-view inpainting, not a native Move editor; we include it as Omni-3DEdit did, using MVInpainter-F, which supports limited reference-guided insertion. Both methods get one edited reference, and MVInpainter additionally gets a mask. For Move, the mask composes the old location with a hand-drawn rectangle at the new one, not a ground-truth silhouette; for Multi-op, the edited-object masks are passed together in one forward pass. The Move results in Figure 5 characterize this adaptation of a removal-oriented checkpoint to simultaneous removal and insertion with the required inputs provided.

D.4 3D Metric Protocol

Reference and prediction sources. Table 4 follows the Gen3R point-cloud protocol (Huang et al., 2026). The reference cloud is unprojected from edited-branch depth using renderer intrinsics and camera poses, with the same fixed frame subset and valid-pixel filtering for every method. All RGB reconstruction rows use the same VGGT checkpoint, resolution/crop policy, and inference settings. *Target RGB + VGGT* provides an oracle RGB-input reference whose geometry is estimated by VGGT; it is excluded from ranking. *JointEdit3D RGB + VGGT* reconstructs our generated RGB, while *JointEdit3D* decodes geometry directly from the predicted joint latent.

Alignment and sampling. Each prediction is aligned to the renderer reference using one Sim(3) Umeyama transform estimated from full-scene valid correspondences. After removal of non-finite points, each predicted and reference cloud is subsampled to at most 100K points with a fixed random

seed; smaller clouds retain all valid points. All global metrics use these sampled sets. Region-wise comparisons and confidence filtering in Table 4 reuse the full-scene transform.

Metric definitions. Accuracy is the mean prediction-to-reference nearest-neighbor distance; Completeness measures the reverse direction; CD is their mean. F-score is the harmonic mean of precision and recall at a distance threshold of 0.20 in renderer world units. We interpret this threshold-specific score alongside the unthresholded Accuracy, Completeness, and CD distances. The separate post-hoc confidence-filtering procedure is described in Appendix B.5.

Native output and task scope. JointEdit3D produces edited RGB views and native geometry aligned with the supplied source trajectory, without requiring a pre-existing NeRF or 3DGS as model input. The geometry outputs include depth, point maps, camera predictions, and a scene point cloud. Our evaluation measures edited-view quality, source-structure preservation, and native geometry accuracy. Editing along another trajectory uses source observations from that trajectory in a new inference pass. Rendering arbitrary novel views requires an additional downstream representation, such as a radiance field or 3DGS, beyond the native outputs evaluated here.

E LIMITATIONS

JointEdit3D relies on the edited reference to specify the desired change. Errors or inconsistencies in that reference may propagate across views and into the generated geometry. Improving robustness to imperfect reference edits is a direction for future work.