UNIVEDIT: UNIFIED VISUAL EDITING ACROSS IMAGES, VIDEOS, AND 3D SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in unified visual editing have begun to consolidate image and video editing into a single model, yet 3D scene editing remains largely isolated. A major challenge for 3D editing is the scarcity of high-quality paired training data, which limits the learning of generalizable editing capabilities. Meanwhile, image, video, and 3D editing share common operations and editing semantics, making knowledge transfer across the three modalities possible. Motivated by this observation, we present **UniVEdit**, a unified framework for image, video, and 3D scene editing within a single model. UniVEdit unifies the three modalities through a shared sequence representation and progressive multimodal training, allowing abundant image and video editing knowledge to benefit data-scarce 3D editing. However, shared editing semantics alone do not guarantee geometrically consistent outputs. We therefore introduce geometry-aware post-training, using a pretrained geometry reconstruction model to supervise depth and camera consistency. We further investigate how geometric supervision should be applied across modalities, and find that applying it only to 3D samples yields the best overall results. Under this setting, although image and video editing receive no direct geometric supervision, they also show gains, suggesting that geometric regularization learned from 3D data can transfer through the shared backbone. Further analysis shows that image and video training mainly transfers editing semantics to 3D, while limited 3D training primarily improves multi-view consistency; geometry-aware post-training further improves overall 3D editing performance.

1 INTRODUCTION

Instruction-driven visual editing has rapidly progressed from images to videos and 3D scenes. Recent works have begun to unify image and video editing within a single model, enabling editing capabilities to be shared across the two modalities (Ju et al., 2026; Mou et al., 2025; Xia et al., 2025; Wei et al., 2026). Meanwhile, generalizable 3D editing increasingly builds on pretrained image or video models to propagate edits consistently across multiple viewpoints (Zhao et al., 2026; Chen et al., 2026b; Huang et al., 2025; Chen et al., 2026c). These developments suggest that editing knowledge may extend beyond individual visual modalities. This motivates a broader question: *can image, video, and 3D editing be learned jointly within a single model, allowing knowledge to be shared across all three domains?*

Two complementary opportunities make such unification promising, as illustrated in Figure 1. *First, editing semantics are largely shared across modalities, while editing supervision is highly asymmetric.* Operations such as object addition and removal, appearance modification, and style transfer enable editing knowledge to transfer across modalities (Ju et al., 2026; Mou et al., 2025). At the same time, large-scale image and video editing datasets are increasingly available (Brooks et al., 2023; Yu et al., 2025; Wu et al., 2025b; Bai et al., 2026), whereas high-quality paired multiview editing data remain scarce (Chen et al., 2026c; Wang et al., 2026). This combination of shared semantics and asymmetric supervision makes cross-modal transfer particularly valuable: a unified model can learn general editing semantics from image and video data, while using comparatively limited 3D data to learn how these edits should remain consistent across viewpoints.

Second, shared editing semantics do not by themselves ensure geometric consistency. Image editing requires geometrically valid spatial structure, while video and multiview editing further require consistency across frames or viewpoints. Prior work has shown that explicit geometric cues and

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

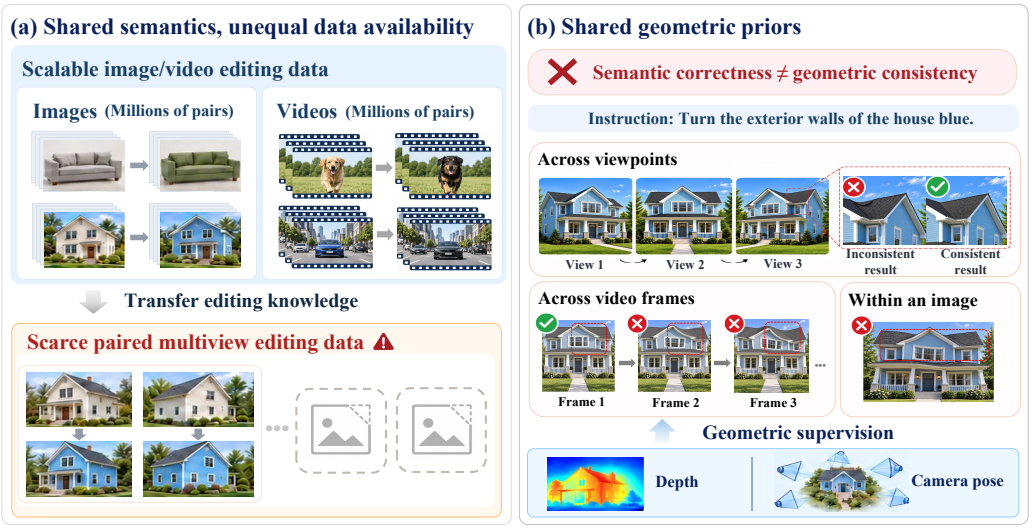

Figure 1: Image and video editing data are abundant while paired multiview data are scarce, and semantically correct edits can still be geometrically inconsistent.

geometry foundation models can improve structural consistency in image, video, and multiview generation (Wu et al., 2024; Li et al., 2025b; Parihar et al., 2025; Du et al., 2026; Yin et al., 2026; Wang et al., 2026; Ma et al., 2026). This suggests that geometric knowledge can serve as a shared structural prior across modalities, complementing semantic transfer within a unified editor.

Based on these insights, we introduce **UniVEdit**, a unified framework for image, video, and 3D editing. UniVEdit represents all three modalities as conditional frame sequences and jointly trains them with a shared backbone and modality-specific tokens, enabling editing knowledge from image and video data to transfer to data-scarce multiview editing. We further introduce *geometry-aware post-training* with depth and camera-pose supervision provided by a frozen geometry teacher, and investigate how geometric objectives should be applied across the three modalities. Together, UniVEdit combines cross-modal editing transfer with explicit geometric regularization in a single model.

Our experiments reveal a clear division of roles across modalities. Image and video training transfers general editing semantics to multi-view inputs even without 3D supervision, whereas adding limited 3D data mainly strengthens cross-view consistency. As a result, UniVEdit outperforms dedicated 3D editing baselines in multi-view consistency while remaining competitive on image and video editing. We further find that geometry-aware post-training works best when restricted to 3D samples, improving overall 3D editing performance while also yielding gains on image and video through the shared backbone.

The main contributions of this paper are:

- We introduce UniVEdit, a unified framework that formulates image, video, and multiview 3D editing as conditional frame-sequence generation within a single shared model.
- We jointly train UniVEdit across modalities to transfer editing knowledge from images and videos to data-scarce multiview editing, and use geometry-aware post-training that distills depth and camera cues from a frozen geometry teacher to regularize structural consistency.
- We evaluate UniVEdit on image, video, and 3D editing, showing cross-modal transfer under limited 3D supervision and the effect of geometry-aware post-training under different supervision scopes.

2 Related Work

Unified Visual Editing. Early attempts at cross-modal visual editing adapt pretrained image diffusion models to structured visual inputs such as panoramas, videos, and 3D scenes through shared inference-time mechanisms (Kim et al., 2023; Kwon et al., 2024). While these methods provide a

common editing procedure across modalities, they do not learn shared editing capabilities through joint multimodal training. More recent works unify image and video generation and editing within a single model using shared sequence representations and generative backbones (Ju et al., 2026; Wei et al., 2026; Mou et al., 2025; Chen et al., 2026a; Pan et al., 2026), enabling editing knowledge to transfer between images and videos. Unified visual models have also begun to incorporate geometric prediction and novel-view synthesis (Yang et al., 2026a), but instruction-driven 3D scene editing remains largely separate from image–video editing. UniVEdit bridges this gap by jointly learning image, video, and multiview 3D editing within a single shared model.

3D Scene Editing. 3D editing commonly transfers powerful 2D editing priors to multiple rendered views and propagates the edits back to an underlying 3D representation. Early methods iteratively optimize NeRFs or 3D Gaussians using edited views (Haque et al., 2023; Chen et al., 2024b), with later approaches improving cross-view consistency through joint attention, feature fusion, or explicit geometric constraints (Chen et al., 2024a; Lee et al., 2025; Wu et al., 2024; Li et al., 2025b). Despite strong editing quality, these approaches typically require per-scene optimization or iterative 2D-3D updates. Recent methods instead learn generalizable multiview editors or exploit pretrained video priors to generate consistent edited views in a feed-forward manner (Zhao et al., 2026; Chen et al., 2026b; Huang et al., 2025; Chen et al., 2026c; Wang et al., 2026; Zhu et al., 2026). Related work further shows that treating multiview observations as pseudo-video sequences can combine temporal priors with explicit geometric correspondences (Ma et al., 2026). These methods adapt pretrained models into dedicated 3D editors, whereas UniVEdit keeps image and video editing data as joint supervision and retains all three tasks in one model.

Geometry-Aware Visual Generation. Generative models trained primarily for visual fidelity may still produce geometrically inconsistent structures, motivating the integration of explicit geometric supervision. One line of work injects geometric information directly into generative models through depth conditioning, geometric feature alignment, or auxiliary dense prediction (Wu et al., 2024; 2026; Bai et al., 2025c; Rao et al., 2026). A complementary line of work uses pretrained geometry or reconstruction models as teachers, evaluators, or reward models for generative post-training (Go et al., 2026; Du et al., 2026; Yin et al., 2026; An et al., 2026; Wang et al., 2026). Such approaches demonstrate that geometry foundation models can provide effective supervisory signals beyond conventional reconstruction objectives. UniVEdit extends this idea to unified visual editing, using a frozen geometry teacher to derive target-conditioned depth and camera supervision applicable to image, video, and multiview editing, and studying its supervision scope.

3 METHOD

3.1 UNIFIED SEQUENCE REPRESENTATION

UniVEdit unifies image, video, and 3D editing within a shared frame-sequence representation. Across modalities, we denote the source and target samples as x^m_{src} and x^m_{tgt}, where $m \in \{\text{image}, \text{video}, \text{3D}\}$. Images are represented as static frame sequences, videos preserve their temporal frame order, and 3D scenes are represented by multi-view sequences ordered according to the sampled camera trajectory. This formulation allows all three modalities to share the same backbone and sequence-processing pipeline. However, the shared sequence representation does not explicitly distinguish static images, temporally evolving videos, and multi-view observations. We therefore introduce three learnable modality tokens $\tau^m \in \{\tau_{\text{img}}, \tau_{\text{vid}}, \tau_{\text{3D}}\}$, registered in the MLLM vocabulary, to encode modality identity. For each sample, the conditioning information is $c^m = (\ell^m, \tau^m, r^m)$, where ℓ^m denotes the editing instruction and r^m is an optional visual reference for reference-based editing tasks. The three editing tasks can therefore be expressed under a unified conditional generation formulation:

$$p_\theta \left(x^m_{\text{tgt}} \mid x^m_{\text{src}}, c^m \right). \tag{1}$$

3.2 MODEL ARCHITECTURE

As shown in Figure 2, given a source sample x^m_{src}, an editing instruction ℓ^m, a modality token τ^m, and an optional reference r^m, the MLLM encodes the textual instruction together with sampled source and reference frames into semantic conditioning features:

$$h^m = E_{\text{MLLM}} \left(\ell^m, \tau^m, x^m_{\text{src}}, r^m \right), \tag{2}$$

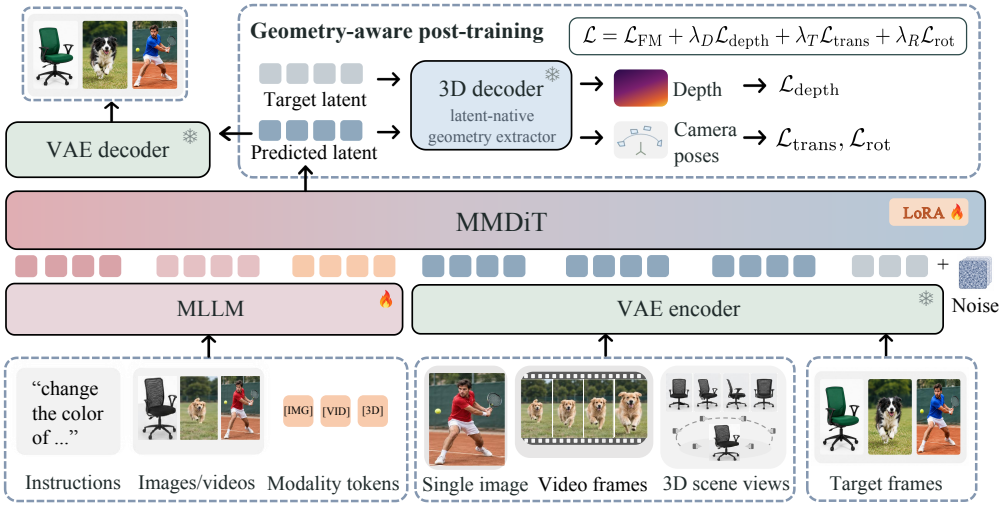

Figure 2: Overview of UniVEdit. Geometry-aware post-training uses a frozen 3D decoder to estimate depth and camera poses from predicted and target latents.

where the reference input is omitted when unavailable. In parallel, a video VAE encodes the complete source and reference sequences into dense visual latents:

$$z^m_{\text{src}} = E_{\text{VAE}}(x^m_{\text{src}}), \qquad z^m_{\text{ref}} = E_{\text{VAE}}(r^m), \tag{3}$$

where z^m_{ref} is omitted for tasks without a visual reference. The MMDiT serves as the shared generative backbone, jointly incorporating the semantic features (i.e., h^m) and dense visual latents to generate the edited target latents. Edited images and videos are directly decoded by the VAE decoder, while 3D scenes are reconstructed from the generated multi-view images using a 3D decoder.

3.3 Progressive Cross-Modal Training

We optimize all three modalities using the same conditional flow-matching objective (Lipman et al., 2023). Given a target latent z_0, we sample an intermediate latent z_t along the flow path at timestep t and train the model to predict the corresponding target velocity v_t:

$$\mathcal{L}_{\text{FM}} = \mathbb{E}\left[\left\| v_\theta(z_t, t, z^m_{\text{src}}, z^m_{\text{ref}}, h^m) - v_t \right\|_2^2\right], \tag{4}$$

where h^m encodes the conditioning information c^m defined in Section 3.1. For training, we adopt a progressive learning strategy that extends the model from image editing to video editing and finally to 3D editing. At each stage, data from the newly introduced modality are added to the existing training set, while the corresponding modality token is learned jointly.

3.4 Geometry-Aware Post-Training

The unified sequence representation enables knowledge sharing across modalities, but does not explicitly enforce the spatial structure or geometric consistency of the edited outputs. We therefore introduce geometry-aware post-training using a frozen latent-native geometry extractor built with the model-stitching strategy of Go et al. (2026). The extractor predicts depth maps and camera poses directly from both predicted and target latents, allowing the geometry extracted from the target latents to supervise the predicted geometry.

Depth consistency. Inspired by the scale-normalized geometric supervision (Wang et al., 2024; Keetha et al., 2026), we adopt a scale-invariant depth loss. We independently normalize the predicted and target depth maps by their sequence-level mean z-depth, denoted by d_{pred} and d_{tgt}, respectively, and apply a Smooth L1 loss ρ:

$$\mathcal{L}_{\text{depth}} = \mathbb{E}_p\left[\rho\left(\frac{D_{\text{pred}}(p)}{d_{\text{pred}}} - \frac{D_{\text{tgt}}(p)}{d_{\text{tgt}}}\right)\right]. \tag{5}$$

Camera trajectory consistency. For video and 3D multi-view sequences, we additionally constrain camera trajectories. We first align the predicted and target trajectories to their respective first frames and supervise both rotation and translation. For rotation, we measure the discrepancy between predicted and target orientations using the scaled chordal distance (Hartley et al., 2013):

$$\mathcal{L}_{\text{rot}} = \mathbb{E}_{i\geq 1}\left[\frac{\left\|\bar{R}_i^{\text{pred}} - \bar{R}_i^{\text{tgt}}\right\|_F}{\sqrt{2}}\right]. \tag{6}$$

For translation, we supervise both adjacent-frame displacements and first-frame-relative positions: displacement errors can accumulate into trajectory drift, whereas position supervision alone provides weaker local constraints (Sturm et al., 2012). Let $\bar{t}_i$ denote the first-frame-aligned camera translation and d the sequence-level mean depth. We define $\tilde{t}_i = \bar{t}_i/d$ and $\Delta\tilde{t}_i = \tilde{t}_i - \tilde{t}_{i-1}$, and optimize

$$\mathcal{L}_{\text{trans}} = \mathbb{E}_i\left[\lambda_{\text{rel}}\rho\left(\Delta\tilde{t}_i^{\text{pred}} - \Delta\tilde{t}_i^{\text{tgt}}\right) + \lambda_{\text{abs}}\rho\left(\tilde{t}_i^{\text{pred}} - \tilde{t}_i^{\text{tgt}}\right)\right]. \tag{7}$$

where the first term constrains local motion and the second anchors the global trajectory. Coordinate alignment and scale normalization are detailed in Appendix B.2. The overall objective combines the flow-matching and geometric losses:

$$\mathcal{L} = \mathcal{L}_{\text{FM}} + \lambda_D\mathcal{L}_{\text{depth}} + \lambda_T\mathcal{L}_{\text{trans}} + \lambda_R\mathcal{L}_{\text{rot}}. \tag{8}$$

We consider depth supervision for image editing, and both depth and camera supervision for video and 3D editing. We then study how these geometric objectives should be applied across modalities. In our final configuration, both depth and camera supervision are applied only to 3D samples. Appendix B provides additional normalization details and hyperparameters.

4 Experiments

4.1 Settings

Datasets. We curate our training set from existing image, video, and 3D editing datasets (Yu et al., 2025; Kuprashevich et al., 2026; Song et al., 2026; He et al., 2026; Lin et al., 2026; Ye et al., 2026; Ling et al., 2024), resulting in a total of 2.53M training samples. The detailed dataset composition is summarized in Table 6 in Appendix A. Since large-scale 3D scene editing data are not readily available, we additionally construct a large-viewpoint 3D scene editing dataset containing 8K samples. The data construction pipeline is provided in Appendix D.

Baselines and evaluation metrics. For image editing, we select ICEdit (Zhang et al., 2025), VAREdit (Mao et al., 2026), and ChordEdit (Lu et al., 2026) as instruction-based baselines. ACE++ (Mao et al., 2025) and FLUX.1 Kontext (Black Forest Labs et al., 2025) represents methods that support both instruction and reference conditions. Our video-editing baselines include the instruction-based methods VideoCoF (Yang et al., 2026c) and Omni-Video 2 (Yang et al., 2026b), together with VACE (Jiang et al., 2025), UniVideo (Wei et al., 2026), VINO (Chen et al., 2026a), Kiwi-Edit (Lin et al., 2026), and OmniWeaving (Pan et al., 2026), which support both instruction and reference conditions. For 3D editing, the selected baselines consist of GaussianEditor (Chen et al., 2024b), DGE (Chen et al., 2024a), and EditSplat (Lee et al., 2025). For evaluation, image editing is assessed in terms of structural preservation and editing-semantic consistency. Video editing additionally considers temporal quality, while 3D editing focuses on instruction alignment and multi-view consistency. Detailed evaluation protocols are provided in Appendix C.

Training details. We initialize UniVEdit with OmniWeaving (Pan et al., 2026), which uses HunyuanVideo-1.5 (Wu et al., 2025a) as the generative backbone and Qwen2.5-VL (Bai et al., 2025b) as the MLLM. We apply LoRA (Hu et al., 2022) to the DiT and add modality tokens to the MLLM vocabulary. Stage-1 follows a progressive training schedule: we first train on image-editing data for 10K steps, then incorporate video-editing data for another 7K steps, and finally add 3D multi-view editing data for 3K steps. Stage-2 starts from the Stage-1 checkpoint and performs geometry-aware post-training for 500 steps using a frozen latent-native geometry extractor, built following the model-stitching strategy of Go et al. (2026). See Appendix A for further details.

Table 1: Quantitative comparison on ***Image Editing***. Struct. Dist.: Structure Distance. Ref.: Reference. Best in **bold**, second-best underlined.

Method	Consistency					CLIP Semantics		
	Struct. Dist. ↓ $\times 10^2$	PSNR ↑	MSE ↓ $\times 10^3$	LPIPS ↓ $\times 10^2$	SSIM ↑ $\times 10^2$	Whole ↑ $\times 10^2$	Edited ↑ $\times 10^2$	Ref. ↑ $\times 10^2$
Instruction-based editing								
ICEdit	3.54	**27.88**	**6.01**	**5.76**	**89.39**	24.24	21.94	/
VAREdit	5.75	23.52	11.02	8.19	81.75	25.47	**22.74**	/
ChordEdit	**2.88**	22.60	8.04	12.35	76.52	24.59	22.06	/
Ours	8.44	24.20	17.19	10.07	85.84	**25.70**	22.45	/
Instruction-based editing & Reference-based editing								
ACE++	10.98	27.75	9.20	5.99	86.43	21.07	18.79	67.96
FLUX.1 Kontext	**9.26**	**36.49**	**0.32**	**1.44**	**97.07**	**24.27**	**21.92**	**78.68**
Ours	12.39	21.95	29.12	14.86	81.91	23.55	21.10	64.40



Figure 3: Comparison with image-editing baselines: (a) instruction-based, (b) reference-based.

4.2 Main Results

Image Editing. We evaluate whether unified multimodal training retains competitive image-editing capability. As shown in Table 1, UniVEdit achieves the highest whole-image CLIP score and the second-highest edited-region CLIP score for instruction-based editing, demonstrating strong semantic alignment, as also illustrated in Figure 3(a). For reference-based editing, UniVEdit remains competitive in semantic alignment, while showing a gap in reference fidelity and consistency. As illustrated in Figure 3(b), this gap may partly reflect the stronger tendency of FLUX.1 Kontext to directly reproduce the reference appearance, whereas UniVEdit follows the editing instruction while adapting the reference appearance to the source scene.

Video Editing. We next examine video editing capability. As shown in Table 2, UniVEdit achieves the highest whole-video CLIP score for instruction-based editing and ranks second across all consistency and temporal-quality metrics, demonstrating a strong balance between semantic editing and temporal preservation. For reference-based editing, UniVEdit remains competitive across semantic and temporal metrics, with the second-best SSIM and comparable CLIP scores to the strongest unified baselines. Figure 4 further shows that UniVEdit faithfully follows category-level editing instructions (i.e., "Jeep") and integrates the reference subject (i.e., "robot") into the target scene while maintaining coherent results across frames.

3D Editing. We evaluate whether UniVEdit can effectively transfer editing capability to 3D while preserving cross-view consistency. As shown in Table 3, UniVEdit achieves the best 3D consistency

Table 2: Quantitative comparison on ***Video Editing***. Best in **bold**, second-best underlined.

Method	Consistency			CLIP Semantics			Temporal Quality		
	PSNR ↑	LPIPS ↓ $\times 10^2$	SSIM ↑ $\times 10^2$	Whole ↑ $\times 10^2$	Edited ↑ $\times 10^2$	Ref. ↑ $\times 10^2$	MFS ↑ $\times 10^2$	MFS-Edit ↑ $\times 10^2$	DOVER ↑ $\times 10^2$
Instruction-based editing									
VideoCoF	**25.64**	**10.23**	**81.23**	26.97	**21.44**	/	**85.42**	**83.69**	**58.46**
OmniVideo 2	11.28	49.92	39.39	23.02	18.83	/	35.61	33.28	28.85
Ours	17.33	33.87	61.21	**27.14**	20.50	/	76.70	74.56	48.41
Instruction-based editing & Reference-based editing									
VACE	17.42	28.16	59.65	26.54	20.49	**66.95**	**82.02**	**80.67**	56.44
UniVideo	15.56	33.69	48.59	**27.44**	21.17	61.88	60.94	55.78	48.01
VINO	14.59	30.49	47.37	26.99	**21.47**	65.99	72.99	69.96	**61.44**
Kiwi-Edit	18.49	**22.04**	57.90	26.32	21.13	59.28	78.30	75.88	54.90
OmniWeaving	**19.57**	30.07	**65.24**	26.50	20.16	55.82	80.67	78.91	44.66
Ours	17.32	33.78	61.25	26.95	20.54	60.60	76.80	74.69	48.56

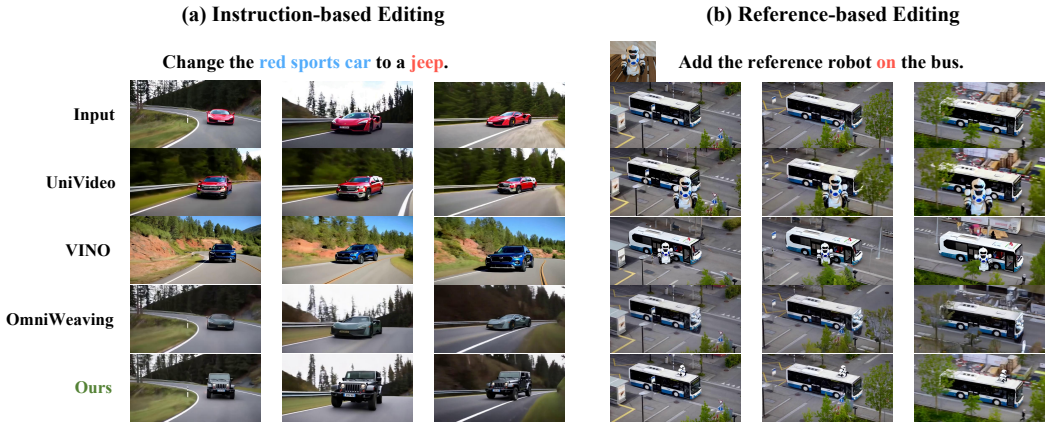

Figure 4: Comparison with unified-model baselines (UniVideo, VINO, OmniWeaving): (a) instruction-based, (b) reference-based.

and lowest Ph-Loss, while ranking second on both CLIPDir and CLIPSim. These results show that UniVEdit maintains strong semantic editing performance while substantially improving multi-view consistency. Figure 5 further demonstrates this capability across diverse edits: UniVEdit consistently changes the person's clothing to a suit, transforms the horse statue into a jade-like material across viewpoints, and propagates the snowy appearance throughout the bonsai scene.

5 In-depth Analysis

5.1 Cross-Modal Transfer to 3D Editing

Since videos and 3D multi-view content can both be represented as frame sequences, we first investigate whether editing capability learned from image and video data transfers to multi-view inputs, and then whether a short stage of 3D training improves cross-view consistency.

As shown in Figure 6, OmniWeaving already exhibits non-trivial editing capability on multi-view inputs, and Stage-1 training without 3D data further improves CLIPDir. This supports the cross-modal transfer of editing capability to multi-view sequences. After introducing 3D data, CLIPDir improves only moderately, whereas 3D Consistency increases substantially. This indicates that limited 3D training primarily improves cross-view consistency.

5.2 Ablation of Geometry-Aware Post-Training

We investigate how the type and scope of geometric supervision affect the unified editor. Table 4 compares no geometric supervision (*w/o Geo*), depth-only supervision on all modalities (*w/o Cam-*

Table 3: Quantitative comparison on ***3D Editing***. Best in **bold**, second-best underlined.

Method	CLIPDir ↑ $\times 10^2$	CLIPSim ↑ $\times 10^2$	3D Consistency ↑ $\times 10^2$	Ph-Loss ↓ $\times 10^2$
GaussianEditor	9.43	26.84	64.14	20.93
DGE	11.27	28.12	64.79	19.86
EditSplat	**13.76**	**28.60**	66.30	19.69
Ours	13.15	28.43	**74.34**	**18.88**

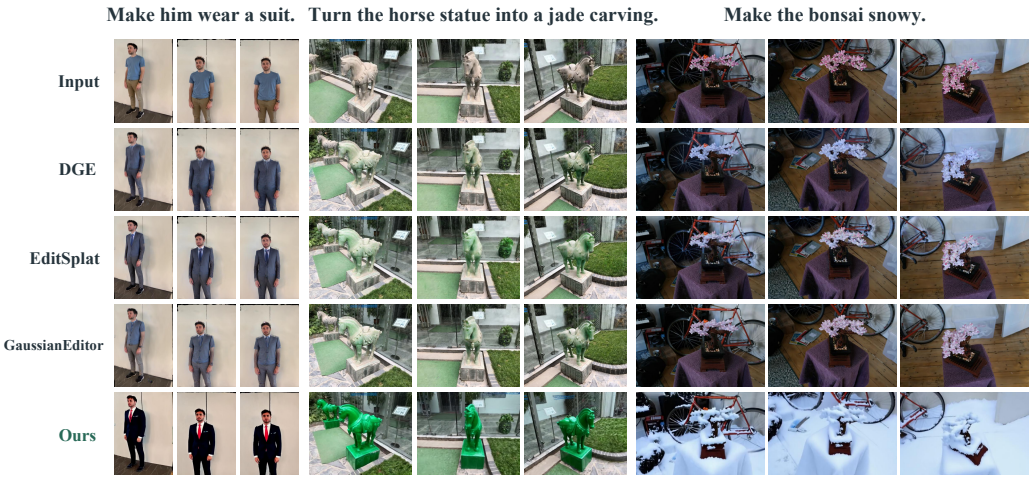

Figure 5: Qualitative comparison against all 3D-editing baselines.

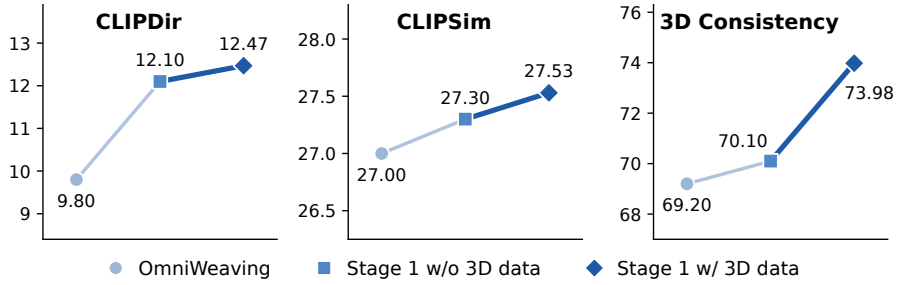

Figure 6: Cross-modal transfer to 3D editing. Stage-1 training on image and video data substantially improves editing semantics on multi-view inputs, while incorporating 3D data yields further gains primarily in cross-view consistency.

era), camera-only supervision on video and 3D (*w/o Depth*), full supervision on all applicable modalities (*Geo-All*), and full supervision only on 3D (*Geo-3D*).

Using *w/o Geo* as the baseline, both depth-only and camera-only supervision improve several video metrics but lower CLIPDir and 3D Consistency, while CLIPSim remains nearly unchanged, revealing a trade-off across modalities. Applying both signals to all applicable modalities does not resolve this conflict: *Geo-All* still underperforms the baseline on all reported 3D metrics and brings little change to image and video performance.

In contrast, restricting both losses to 3D samples (*Geo-3D*) improves all reported metrics over *w/o Geo* and outperforms *Geo-All* on every reported image, video, and 3D metric, showing that the supervision scope is critical. The image and video gains arise without any direct geometric supervision on these modalities, suggesting that geometric regularization learned from 3D data can transfer through the shared backbone.

We hypothesize that the supervision scope matters because geometric signals align differently with each modality. In multi-view 3D data, depth and camera poses directly constrain cross-view scene structure, whereas for single images and videos they are less directly aligned with the editing objective; for videos, this mismatch is further complicated by object motion. We therefore adopt *Geo-3D* as our final configuration.

Table 4: Ablation of geometry-aware post-training. Values in parentheses show changes from Ours (Geo-3D), with ↑ and ↓ directions considered. Red indicates degradation after ablation, while gray indicates improvement. 3D Consist. denotes 3D Consistency.

Method	Image Editing		Video Editing			3D Editing		
	LPIPS ↓	Whole ↑	LPIPS ↓	Whole ↑	MFS ↑	CLIPDir ↑	CLIPSim ↑	3D Consist. ↑
- w/o **Geo**	14.88 (-0.02)	23.48 (-0.07)	34.20 (-0.42)	26.90 (-0.05)	76.05 (-0.75)	12.47 (-0.68)	27.53 (-0.90)	73.98 (-0.36)
- w/o **Camera**	15.02 (-0.16)	23.48 (-0.07)	34.01 (-0.23)	26.92 (-0.03)	76.38 (-0.42)	11.76 (-1.39)	27.49 (-0.94)	72.17 (-2.17)
- w/o **Depth**	15.19 (-0.33)	23.57 (+0.02)	33.72 (+0.06)	26.92 (-0.03)	77.05 (+0.25)	12.10 (-1.05)	27.54 (-0.89)	72.45 (-1.89)
Geo-All	14.96 (-0.10)	23.50 (-0.05)	34.35 (-0.57)	26.89 (-0.06)	76.07 (-0.73)	11.62 (-1.53)	27.33 (-1.10)	72.82 (-1.52)
Ours (Geo-3D)	14.86	23.55	33.78	26.95	76.80	13.15	28.43	74.34



Figure 7: Four-frame replication per view (right) preserves sharper, more saturated detail than a single frame per view (left).

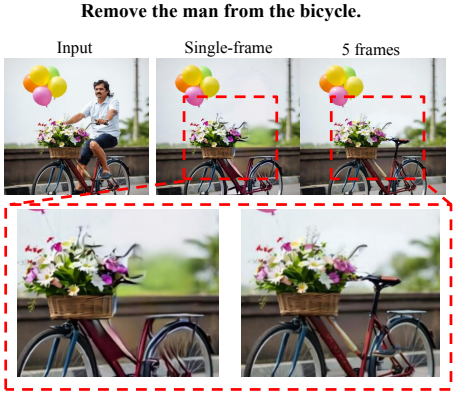

Figure 8: Five-frame static replication (right) recovers finer structural detail than single-frame input (middle) for image editing.

5.3 Analysis of Implementation Choices

3D: Four-Frame Replication per View. During Stage-1 training, we observed substantial detail loss in 3D editing. We suspected that temporal VAE compression contributed to this problem, since adjacent frames in 3D sequences often contain much larger viewpoint changes than those in ordinary videos. To verify this, we reconstructed the original 3D sequences using the HunyuanVideo-1.5 (Wu et al., 2025a) VAE. The reconstructed sequences achieve 26.02 dB PSNR and retain only 41.9% of the high-frequency content. We therefore represent the first view with a single frame and each subsequent view with four identical frames to match the VAE's temporal compression ratio. This increases reconstruction PSNR to 30.24 dB and high-frequency retention to 66.5%, producing visibly sharper results, as shown in Figure 7. We adopt this representation for 3D training.

Image: Five-Frame Replication. We found that directly modeling image editing with a single frame produced weaker results. Since the backbone is pretrained for multi-frame generation, we hypothesize that image editing can also benefit from sequence modeling. We therefore repeat each image five times to form a static sequence, which is encoded into two latent frames. Then we take the first edited frame as the result. This simple change improves the editing quality, as shown in Figure 8, and we adopt it for image training.

6 Conclusion

We presented **UniVEdit**, a unified framework for image, video, and 3D scene editing, together with geometry-aware post-training for improving geometric consistency. Our experiments demonstrate positive transfer of editing capabilities from image and video to 3D, while limited 3D training primarily improves cross-view consistency. We further find that applying geometry-aware post-training only to 3D samples gives the best overall results among the tested supervision scopes and also brings modest gains to image and video editing through the shared backbone. Overall, UniVEdit maintains competitive image and video editing performance while enabling strong and consistent 3D editing.

AI USE STATEMENT

In preparing this manuscript, we used the generative AI assistant Claude to draft portions of the paper, polish the writing, and retrieve relevant literature. We also used the generative model Kling-o1 to generate synthetic datasets used in this study. We did not use generative AI to design the research method, analyze experimental results, or draw the paper's scientific conclusions. The authors reviewed all AI-assisted text and generated materials and take responsibility for the final content, claims, and artifacts.

REFERENCES

Zhaochong An, Orest Kupyn, Théo Uscidda, Andrea Colaco, Karan Ahuja, Serge Belongie, Mar Gonzalez-Franco, and Marta Tintore Gazulla. VGGRPO: Towards World-Consistent Video Generation with 4D Latent Reward. In *Proceedings of the ECCV*, 2026.

Qingyan Bai, Qiuyu Wang, Hao Ouyang, Yue Yu, Hanlin Wang, Wen Wang, Ka Leong Cheng, Shuailei Ma, Yanhong Zeng, Zichen Liu, Yinghao Xu, Yujun Shen, and Qifeng Chen. Scaling Instruction-Based Video Editing with a High-Quality Synthetic Dataset. In *Proceedings of the CVPR*, 2026.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, 2025b.

Yunpeng Bai, Shaoheng Fang, Chaohui Yu, Fan Wang, and Qixing Huang. GeoVideo: Introducing Geometric Regularization into Video Generation Model. In *Proceedings of NeurIPS*, 2025c.

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded Anti-Aliased Neural Radiance Fields. In *Proceedings of the CVPR*, 2022.

Black Forest Labs, Stephen Batifol, Andreas Blattmann, Frederic Boesel, Saksham Consul, Cyril Diagne, Tim Dockhorn, Jack English, Zion English, Patrick Esser, et al. Flux.1 kontext: Flow matching for in-context image generation and editing in latent space. *arXiv preprint arXiv:2506.15742*, 2025.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. InstructPix2Pix: Learning To Follow Image Editing Instructions. In *Proceedings of the CVPR*, 2023.

Junyi Chen, Tong He, Zhoujie Fu, Pengfei Wan, Kun Gai, and Weicai Ye. Vino: A unified visual generator with interleaved omnimodal context. *arXiv preprint arXiv:2601.02358*, 2026a.

Liyi Chen, Ruihuang Li, Guowen Zhang, Pengfei Wang, and Lei Zhang. Fast Multi-view Consistent 3D Editing with Video Priors. In *Proceedings of the AAAI*, 2026b.

Liyi Chen, Pengfei Wang, Guowen Zhang, Zhiyuan Ma, and Lei Zhang. Omni-3DEdit: Generalized Versatile 3D Editing in One-Pass. In *Proceedings of CVPR*, 2026c.

Minghao Chen, Iro Laina, and Andrea Vedaldi. DGE: Direct Gaussian 3D Editing by Consistent Multi-view Editing. In *Proceedings of ECCV*, 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 CVPR*, 2024b.

Hongyang Du, Junjie Ye, Xiaoyan Cong, Runhao Li, Jingcheng Ni, Aman Agarwal, Zeqi Zhou, Zekun Li, Randall Balestriero, and Yue Wang. VideoGPA: Distilling Geometry Priors for 3D-Consistent Video Generation. In *Proceedings of the ICML*, 2026.

Hyojun Go, Dominik Narnhofer, Goutam Bhat, Prune Truong, Federico Tombari, and Konrad Schindler. Text-to-3D by Stitching a Multi-view Reconstruction Network to a Video Generator. In *Proceedings of the ICLR*, 2026.

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

Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 2013.

Haoyang He, Jie Wang, Jiangning Zhang, Zhucun Xue, Xingyuan Bu, Qiangpeng Yang, Shilei Wen, and Lei Xie. OpenVE-3M: A Large-Scale High-Quality Dataset for Instruction-Guided Video Editing. In *Proceedings of the ECCV*, 2026.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *Proceedings of the ICLR*, 2022.

Junchao Huang, Xinting Hu, Shaoshuai Shi, Zhuotao Tian, and Li Jiang. Edit360: 2D Image Edits to 3D Assets from Any Angle. In *Proceedings of the ICCV*, 2025.

Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. VACE: All-in-One Video Creation and Editing. In *Proceedings of the ICCV*, 2025.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. PnP Inversion: Boosting Diffusion-based Editing with 3 Lines of Code. In *Proceedings of the ICLR*, 2024.

Xuan Ju, Tianyu Wang, Yuqian Zhou, He Zhang, Qing Liu, Nanxuan Zhao, Zhifei Zhang, Yijun Li, Yuanhao Cai, Shaoteng Liu, Daniil Pakhomov, Zhe Lin, Soo Ye Kim, and Qiang Xu. EditVerse: Unifying Image and Video Editing and Generation with In-Context Learning. In *Proceedings of the ICLR*, 2026.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is Better to Track Together. In *Proceedings of the ECCV*, 2024.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, Jonathon Luiten, Manuel Lopez-Antequera, Samuel Rota Bulò, Christian Richardt, Deva Ramanan, Sebastian Scherer, and Peter Kontschieder. MapAnything: Universal Feed-Forward Metric 3D Reconstruction. In *Proceedings of the 3DV*, 2026.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 2023.

Subin Kim, Kyungmin Lee, June Suk Choi, Jongheon Jeong, Kihyuk Sohn, and Jinwoo Shin. Collaborative Score Distillation for Consistent Visual Editing. In *Proceedings of the NeurIPS*, 2023.

Kling Team, Jialu Chen, Yuanzheng Ci, Xiangyu Du, Zipeng Feng, Kun Gai, Sainan Guo, Feng Han, Jingbin He, Kang He, et al. Kling-omni technical report. *arXiv preprint arXiv:2512.16776*, 2025.

Maksim Kuprashevich, Grigorii Alekseenko, Irina Tolstykh, Georgii Fedorov, Bulat Suleimanov, Vladimir Dokholyan, and Aleksandr Gordeev. NoHumansRequired: Autonomous High-Quality Image Editing Triplet Mining. In *Proceedings of the WACV*, 2026.

Gihyun Kwon, Jangho Park, and Jong Chul Ye. Unified editing of panorama, 3d scenes, and videos through disentangled self-attention injection. *arXiv preprint arXiv:2405.16823*, 2024.

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 CVPR*, 2025.

Minghan Li, Chenxi Xie, Yichen Wu, Lei Zhang, and Mengyu Wang. FiVE-Bench: A Fine-grained Video Editing Benchmark for Evaluating Emerging Diffusion and Rectified Flow Models. In *Proceedings of the ICCV*, 2025a.

Ruihuang Li, Liyi Chen, Zhengqiang Zhang, Varun Jampani, Vishal M Patel, and Lei Zhang. Sync-Noise: Geometrically Consistent Noise Prediction for Instruction-based 3D Editing. In *Proceedings of the AAAI*, 2025b.

Yiqi Lin, Guoqiang Liang, Ziyun Zeng, Zechen Bai, Yanzhe Chen, and Mike Zheng Shou. Kiwi-Edit: Versatile Video Editing via Instruction and Reference Guidance, 2026.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, Xuanmao Li, Xingpeng Sun, Rohan Ashok, Aniruddha Mukherjee, Hao Kang, Xiangrui Kong, Gang Hua, Tianyi Zhang, Bedřich Beneš, and Aniket Bera. DL3DV-10K: A Large-Scale Scene Dataset for Deep Learning-based 3D Vision. In *Proceedings of the CVPR*, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *Proceedings of the ICLR*, 2023.

Liangsi Lu, Xuhang Chen, Minzhe Guo, Shichu Li, Jingchao Wang, and Yang Shi. ChordEdit: One-Step Low-Energy Transport for Image Editing. In *Proceedings of the CVPR*, 2026.

Yue Ma, Xinyu Wang, Qianli Ma, Qinghe Wang, Mingzhe Zheng, Xiangpeng Yang, Hao Li, Chongbo Zhao, Jixuan Ying, Harry Yang, Hongyu Liu, and Qifeng Chen. Group Editing: Edit Multiple Images in One Go. In *Proceedings of the CVPR*, 2026.

Chaojie Mao, Jingfeng Zhang, Yulin Pan, Zeyinzi Jiang, Zhen Han, Yu Liu, and Jingren Zhou. ACE++: Instruction-Based Image Creation and Editing via Context-Aware Content Filling. In *Proceedings of the ICCV Workshops*, 2025.

Qingyang Mao, Qi Cai, Yehao Li, Yingwei Pan, Mingyue Cheng, Ting Yao, Qi Liu, and Tao Mei. Visual Autoregressive Modeling for Instruction-Guided Image Editing. In *Proceedings of the ICLR*, 2026.

Chong Mou, Qichao Sun, Yanze Wu, Pengze Zhang, Xinghui Li, Fulong Ye, Songtao Zhao, and Qian He. Instructx: Towards unified visual editing with mllm guidance. *arXiv preprint arXiv:2510.08485*, 2025.

Kaihang Pan, Qi Tian, Jianwei Zhang, Weijie Kong, Jiangfeng Xiong, Yanxin Long, Shixue Zhang, Haiyi Qiu, Tan Wang, Zheqi Lv, et al. Omniweaving: Towards unified video generation with free-form composition and reasoning. *arXiv preprint arXiv:2603.24458*, 2026.

Yulin Pan, Xiangteng He, Chaojie Mao, Zhen Han, Zeyinzi Jiang, Jingfeng Zhang, and Yu Liu. ICE-Bench: A Unified and Comprehensive Benchmark for Image Creating and Editing. In *Proceedings of the ICCV*, 2025.

Rishubh Parihar, Sachidanand VS, and R. Venkatesh Babu. Zero-Shot Depth Aware Image Editing with Diffusion Models. In *Proceedings of the ICCV*, 2025.

Qwen Team. Qwen3.5: Towards Native Multimodal Agents, 2026.

Zhefan Rao, Bin Zou, Haoxuan Che, Xuanhua He, Chong Hou Choi, Yanheng Li, Rui Liu, and Qifeng Chen. From dense prediction to visual editing: Structured supervision for unified image and video creation. *arXiv preprint arXiv:2608.14740*, 2026.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment Anything in Images and Videos. *arXiv preprint arXiv:2408.00714*, 2024.

Wensong Song, Hong Jiang, Zongxin Yang, Zheqiao Cheng, Ruijie Quan, and Yi Yang. Insert Anything: Image Insertion via In-Context Editing in DiT. In *Proceedings of the AAAI*, 2026.

Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. VGGT: Visual Geometry Grounded Transformer. In *Proceedings of the CVPR*, 2025.

Jiyuan Wang, Chunyu Lin, Lei Sun, Zhi Cao, Yuyang Yin, Lang Nie, Zhenlong Yuan, Xiangxiang Chu, Yunchao Wei, Kang Liao, and Guosheng Lin. Edit in 2D, Verify in 3D: Reinforcement Learning for Multi-view Consistent Scene Editing. In *Proceedings of the ECCV*, 2026.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. DUSt3R: Geometric 3D Vision Made Easy. In *Proceedings of the CVPR*, 2024.

Cong Wei, Quande Liu, Zixuan Ye, Qiulin Wang, Xintao Wang, Pengfei Wan, Kun Gai, and Wenhu Chen. UniVideo: Unified Understanding, Generation, and Editing for Videos. In *Proceedings of the ICLR*, 2026.

Bing Wu, Chang Zou, Changlin Li, Duojun Huang, Fang Yang, Hao Tan, Jack Peng, Jianbing Wu, Jiangfeng Xiong, Jie Jiang, et al. Hunyuanvideo 1.5 technical report. *arXiv preprint arXiv:2511.18870*, 2025a.

Haoning Wu, Erli Zhang, Liang Liao, Chaofeng Chen, Jingwen Hou, Annan Wang, Wenxiu Sun, Qiong Yan, and Weisi Lin. Exploring Video Quality Assessment on User Generated Contents from Aesthetic and Technical Perspectives. In *Proceedings of the ICCV*, 2023.

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 *Proceedings of the ICLR*, 2026.

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 *Proceedings of the ECCV*, 2024.

Yuhui Wu, Liyi Chen, Ruibin Li, Shihao Wang, Chenxi Xie, and Lei Zhang. InsViE-1M: Effective Instruction-based Video Editing with Elaborate Dataset Construction. In *Proceedings of the ICCV*, 2025b.

Bin Xia, Jiyang Liu, Yuechen Zhang, Bohao Peng, Ruihang Chu, Yitong Wang, Xinglong Wu, Bei Yu, and Jiaya Jia. Dreamve: Unified instruction-based image and video editing. *arXiv preprint arXiv:2508.06080*, 2025.

Ceyuan Yang, Zhijie Lin, Yang Zhao, Fei Xiao, Hao He, Qi Zhao, Chaorui Deng, Kunchang Li, Zihan Ding, Yuwei Guo, et al. Context unrolling in omni models. *arXiv preprint arXiv:2604.21921*, 2026a.

Hao Yang, Zhiyu Tan, Jia Gong, Luozheng Qin, Hesen Chen, Xiaomeng Yang, Yuqing Sun, Yuetan Lin, Mengping Yang, and Hao Li. Omni-video 2: Scaling mllm-conditioned diffusion for unified video generation and editing. *arXiv preprint arXiv:2602.08820*, 2026b.

Xiangpeng Yang, Ji Xie, Yiyuan Yang, Yue Ma, Yan Huang, Min Xu, and Qiang Wu. VideoCoF: Unified Video Editing with Temporal Reasoner. In *Proceedings of the CVPR*, 2026c.

Yao Yao, Zixin Luo, Shiwei Li, Jingyang Zhang, Yufan Ren, Lei Zhou, Tian Fang, and Long Quan. BlendedMVS: A Large-Scale Dataset for Generalized Multi-View Stereo Networks. In *Proceedings of the CVPR*, 2020.

Junliang Ye, Shenghao Xie, Ruowen Zhao, Zhengyi Wang, Hongyu Yan, Wenqiang Zu, Lei Ma, and Jun Zhu. Nano3D: A Training-Free Approach for Efficient 3D Editing Without Masks. In *Proceedings of the ICLR*, 2026.

Tengjiao Yin, Jinglei Shi, Heng Guo, and Xi Wang. VIGOR: VIdeo Geometry-Oriented Reward for Temporal Generative Alignment. In *Proceedings of the ECCV*, 2026.

Qifan Yu, Wei Chow, Zhongqi Yue, Kaihang Pan, Yang Wu, Xiaoyang Wan, Juncheng Li, Siliang Tang, Hanwang Zhang, and Yueting Zhuang. AnyEdit: Mastering Unified High-Quality Image Editing for Any Idea. In *Proceedings of the CVPR*, 2025.

Zechuan Zhang, Ji Xie, Yu Lu, Zongxin Yang, and Yi Yang. Enabling Instructional Image Editing with In-Context Generation in Large Scale Diffusion Transformer. In *Proceedings of the NeurIPS*, 2025.

Canyu Zhao, Xiaoman Li, Tianjian Feng, Zhiyue Zhao, Hao Chen, and Chunhua Shen. TINKER: Diffusion's Gift to 3D–Multi-View Consistent Editing From Sparse Inputs without Per-Scene Optimization. In *Proceedings of the ICLR*, 2026.

Guangcong Zheng, Teng Li, Xianpan Zhou, and Xi Li. Realcam-vid: High-resolution video dataset with dynamic scenes and metric-scale camera movements. *arXiv preprint arXiv:2504.08212*, 2025.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo Magnification: Learning View Synthesis using Multiplane Images. *ACM Transactions on Graphics*, 2018.

Xinnan Zhu, Ruijie Xu, Jiayu Ying, Daoguo Dong, Jiachen Xu, Yuan Xie, and Xin Tan. JointEdit3D: Feed-Forward 3D Scene Editing in a Unified Latent Space, 2026.

Appendix

Content

A Implementation Details **15**
B Geometry Loss Details **16**
B.1 Depth Loss 16
B.2 Camera Loss 16
B.3 Hyperparameters 18
C Evaluation Protocols **19**
C.1 Image Editing 19
C.1.1 Benchmark 19
C.1.2 Metrics 19
C.2 Video Editing 20
C.2.1 Benchmark 20
C.2.2 Metrics 20
C.3 3D Editing 21
C.3.1 Benchmark 21
C.3.2 Metrics 21
D Construction of Multi-view Editing Data **21**
D.1 Source Sequence Preparation 21
D.2 Target Selection and Instruction Construction 22
D.3 Paired Edit Generation 23
D.4 Prompts 23
E Additional Results **27**
F Limitation **27**

A Implementation Details

Table 5 summarizes the training schedule and hyperparameters. Both training stages use 360p resolution and run on 8 NVIDIA H200 GPUs. The LoRA rank and alpha are set to 32. During Stage 1, we update only the DiT LoRA and modality tokens added to the MLLM vocabulary. For Stage 2, we construct the geometry extractor by retraining a latent-native 3D decoder for HunyuanVideo-1.5 (Wu et al., 2025a) latent space, following the model-stitching strategy of Go et al. (2026). During this stage, the DiT LoRA is updated, while the geometry extractor and modality tokens remain frozen.

Table 5: Training schedule and hyperparameters. I, V, and 3D denote image, video, and 3D multi-view editing, respectively. A dash indicates an inactive modality.

Setting	**Stage 1: progressive training**			**Stage 2**
	Ph. 1 (I)	**Ph. 2 (I+V)**	**Ph. 3 (I+V+3D)**	**Geometry-aware**
Training steps	10K	7K	3K	500
Learning rate (LoRA)	5×10^{-5}	5×10^{-5}	2×10^{-5}	1×10^{-5}
Learning rate (modality tokens)	1×10^{-5}	1×10^{-5}	5×10^{-6}	Frozen
Effective batch size (image)	200	480	480	128
Effective batch size (video)	/	120	120	32
Effective batch size (3D)	/	/	120	32
Sequence length (image)	1	5	5	5
Sequence length (video)	/	65	65	65
Sequence length (3D)	/	/	101	65

Table 6: Training data used for image, video, and 3D multi-view editing.

Modality	Data Source	Task Type	Samples
Image	AnyEdit (Yu et al., 2025)	Instruct-based Editing	∼2M
	NHR-Edit (Kuprashevich et al., 2026)	Instruct-based Editing	
	AnyInsertion (Song et al., 2026)	Reference-based Editing	
Video	OpenVE (He et al., 2026)	Instruct-based Editing	∼500K
	Kiwi-Edit (Lin et al., 2026)	Reference-based Editing	
3D	Self-built scene editing dataset	Instruct-based Editing	∼8K
	Nano3D (Ye et al., 2026)	Instruct-based Editing	∼20K
	DL3DV (Ling et al., 2024)	Non-editing	∼2K

B Geometry Loss Details

B.1 Depth Loss

Normalization. Predicted and target geometries are reconstructed independently and may differ in their overall scale. We normalize each reconstruction by its own mean depth:

$$d^b = \frac{1}{NHW} \sum_{i=0}^{N-1} \sum_{u=1}^{H} \sum_{v=1}^{W} D_i^b(u, v), \qquad b \in \{\text{pred}, \text{tgt}\}, \tag{9}$$

where N is the number of supervised frames and H, W are the depth-map dimensions. Sharing this factor across the sequence removes a common multiplicative scale while preserving relative depth scales between frames. Independent per-frame normalization would instead remove these inter-frame scale differences. We do not subtract a depth offset, since additive shifts change depth ratios, and centering would remove sensitivity to these changes.

We retain gradients through the predicted mean depth, so differentiation accounts for both the depth values and the normalization factor. Consequently, the gradient has zero component along the overall scale direction, consistent with the invariance of normalized depth to a common multiplicative rescaling.

Supervision setup. The predicted and target branches use the same frame indices. For a video or multiview sequence of 65 frames, we supervise the 17 frames indexed by $0, 4, \ldots, 64$. For image editing, only the first latent frame is passed to the geometry extractor, and normalization uses the resulting depth map. The target branch uses the latent given by the VAE posterior mode to avoid randomness from posterior sampling and is evaluated without gradients.

B.2 Camera Loss

Camera supervision requires a common coordinate convention and comparable reconstruction scales. We first align the coordinate systems, then compare rotations and depth-normalized translations. Translation supervision combines local displacement and global position constraints.

Coordinate alignment. Independent reconstructions may use different world coordinate systems. Direct pose comparison would therefore include coordinate-frame differences in the trajectory error. We express each trajectory in its own first-camera coordinate system:

$$\bar{T}_i^{\text{pred}} = \text{sg}(T_0^{\text{pred}})^{-1} T_i^{\text{pred}}, \qquad \bar{T}_i^{\text{tgt}} = (T_0^{\text{tgt}})^{-1} T_i^{\text{tgt}}, \tag{10}$$

where T_i is a camera-to-world pose and sg denotes stop-gradient. The reference transform is treated as fixed during backpropagation.

We check the extractor outputs to determine whether this alignment is already enforced. The median angular deviation of the target first-camera rotation from the identity is 0.73° for 3D and 0.23° for video, indicating that explicit alignment is needed to enforce this convention. We also test invariance by left-multiplying all poses of one trajectory by the same rigid transformation Q, while keeping the other trajectory unchanged. The aligned camera loss remains unchanged, consistent with

$$(QT_0)^{-1}(QT_i) = T_0^{-1} T_i. \tag{11}$$

Rotation distance. After alignment, we compare rotations using the scaled chordal distance. For a relative rotation angle $\theta \in [0, \pi]$,

$$\frac{\|\bar{R}_i^{\text{pred}} - \bar{R}_i^{\text{tgt}}\|_F}{\sqrt{2}} = 2\sin(\theta/2). \tag{12}$$

This distance increases monotonically with θ, approximates the angular error at small angles, and can be computed directly from rotation matrices without evaluating an inverse cosine. The aligned first-camera rotation is the identity, so the rotation loss is evaluated only on the remaining frames.

Translation scale normalization. Camera translations must also be normalized to account for differences in reconstruction scale. Using the mean inter-frame displacement as the denominator introduces an inverse dependence on camera-motion magnitude into the normalization derivative. This can amplify gradients when camera motion is small.

We instead use mean depth, which shares the reconstruction scale of camera translation without using motion magnitude as the denominator:

$$p_i^{\text{pred}} = \frac{\bar{t}_i^{\text{pred}}}{\text{sg}(d^{\text{pred}})}, \qquad p_i^{\text{tgt}} = \frac{\bar{t}_i^{\text{tgt}}}{d^{\text{tgt}}}. \tag{13}$$

This removes the common reconstruction scale and avoids the explicit inverse-motion factor. The depth denominator is treated as a fixed scale reference during backpropagation, so translation supervision does not propagate through the depth scale.

Local displacement and global position. Local displacement errors can either accumulate or cancel along a trajectory. Their magnitudes alone therefore do not determine the global position error. We complement adjacent-frame displacement supervision with a position anchor relative to the first frame. Writing the normalized position error as

$$e_i = p_i^{\text{pred}} - p_i^{\text{tgt}}, \tag{14}$$

the local residual is $e_i - e_{i-1}$, whereas the anchor residual is e_i after alignment.

An analytical comparison illustrates the distinction. We fix the target trajectory and construct two predicted error patterns. One adds the same error vector δ at every step. The other alternates the error increment between δ and $-\delta$. Let $M = N - 1$ be even and set $e_0 = 0$. As shown in Table 7, both patterns have the same local loss because SmoothL1 is symmetric with respect to the residual sign. Their anchor losses differ because only the first pattern accumulates position error.

Table 7: Analytical comparison of translation error patterns. Here, ϕ denotes SmoothL1 applied to each coordinate and averaged over the three coordinates. Equal local losses can correspond to different accumulated position errors.

Quantity	Same-direction accumulation	Alternating errors
Position error e_i	$i\delta$	$\frac{1-(-1)^i}{2}\delta$
Displacement error	δ	$(-1)^{i+1}\delta$
Mean local loss	$\phi(\delta)$	$\phi(\delta)$
Mean anchor loss	$\frac{1}{M}\sum_{i=1}^{M}\phi(i\delta)$	$\frac{1}{2}\phi(\delta)$
Final position error	$M\delta$	0

We further examine whether model predictions exhibit larger position errors later in the sequence. We select 10 source samples from each of the 3D editing, OpenVE, and Kiwi-Edit datasets and generate predictions from three checkpoints, using 50 sampling steps and three random seeds per checkpoint. This yields 90 prediction–target pairs for 3D and 180 for video across checkpoints and seeds. For each trajectory, we divide the non-reference frames into two halves and compute the mean position error $\|e_i\|_2$ in each half. The median ratio of second-half to first-half error is 2.23 for 3D and 1.43 for video. These larger errors in the second half further motivate explicit position supervision alongside the local displacement loss.

B.3 HYPERPARAMETERS

We calibrate weights using gradient norms to control the relative influence of the loss terms. Below, we describe the balance within the translation objective, the balance among geometry components, and the overall geometry strength relative to flow matching. The first uses gradients with respect to predicted camera extrinsics. The latter two use gradients with respect to trainable DiT LoRA parameters.

Balance within the translation objective. We use the prediction–target trajectory pairs described in Section B.2 to measure the gradients of the local and anchor terms with respect to predicted camera extrinsics P. For each modality, we compute

$$q_m = \mathrm{median}_k \frac{\|\nabla_P \mathcal{L}_{\mathrm{abs}}^{(m,k)}\|_2}{\|\nabla_P \mathcal{L}_{\mathrm{rel}}^{(m,k)}\|_2}, \tag{15}$$

where m indexes modalities and k indexes prediction–target pairs. With the local weight fixed to 1, an anchor weight of q_m^{-1} approximately balances the two gradient norms.

The resulting balancing weights are 1.27 for 3D and 1.33 for video, as reported in Table 8. Their similarity motivates the shared setting $\lambda_{\mathrm{rel}} = 1$ and $\lambda_{\mathrm{abs}} = 1.3$.

Table 8: Calibration of the anchor weight using gradients with respect to predicted camera extrinsics.

Calibration quantity	**3D**	**Video**
Number of prediction–target pairs	90	180
Unweighted anchor/local gradient ratio q_m	0.79	0.75
Balancing weight q_m^{-1}	1.27	1.33
Selected anchor weight	1.30	1.30

Balance among geometry components. We use gradient norms with respect to trainable DiT LoRA parameters to balance rotation against translation and camera supervision against depth supervision. During initial calibration, equally weighted translation gradients are approximately 8.9 times larger than rotation gradients. We therefore assign internal weights of 0.9 to rotation and 0.1 to translation. We then set the camera multiplier so that the camera and depth objectives have comparable parameter gradient norms during calibration. After component and modality scaling, the resulting gradients remain balanced; Table 9 reports these effective coefficients.

Overall geometry strength. We use the flow-matching objective as the reference for the overall strength of geometry supervision:

$$r_m = \mathrm{median}_k \frac{\|\nabla_\theta \mathcal{L}_{\mathrm{geo}}^{(m,k)}\|_2}{\|\nabla_\theta \mathcal{L}_{\mathrm{FM}}^{(m,k)}\|_2}, \tag{16}$$

where θ denotes the trainable LoRA parameters, $\mathcal{L}_{\mathrm{geo}}$ includes all geometry weights, and k indexes gradient measurements during training. We use approximately 5%–10% as the calibration target so that geometry supervision acts as an auxiliary objective. The median reduces the influence of occasional large gradients, and separate modality multipliers account for differences in gradient magnitude.

Final configuration. The SmoothL1 threshold is 0.02 for depth and 0.001 for both translation terms. Table 9 reports these thresholds and the final weights. The effective coefficients λ_D, λ_T, and λ_R include all component scaling factors and modality multipliers. The local and anchor weights specify the relative contributions within the translation objective. The flow-matching weight is 1.

Table 9: Geometry loss hyperparameters. Effective loss weights include all calibration and modality-specific multipliers and are rounded to two decimal places. SmoothL1 thresholds are reported exactly. The flow-matching weight is 1. A dash indicates an inapplicable setting.

Category	Parameter	Image	Video	3D
SmoothL1 thresholds	Depth β_d	0.02	0.02	0.02
	Translation β_t	–	0.001	0.001
Translation weights	Local λ_{rel}	/	1.00	1.00
	Anchor λ_{abs}	/	1.30	1.30
Effective loss weights	Depth λ_D	0.47	0.22	0.27
	Translation λ_T	/	1.26	1.57
	Rotation λ_R	/	0.89	1.11

C Evaluation Protocols

C.1 Image Editing

C.1.1 Benchmark

We evaluate image editing on the PIE-Bench (Ju et al., 2024) and the reference-based editing subset of ICE-Bench (Pan et al., 2025). For evaluation, the edited images produced by the baseline methods are resized to match the dimensions of their corresponding input images.

The PIE-Bench dataset contains 700 samples across ten editing categories: random editing (140); object replacement, addition, and removal (80 each); content, pose, color, and material changes (40 each); and background and style changes (80 each). Each sample provides a source image, an editing instruction, source and target descriptions, and an editing-region mask. The reference-based editing subset of ICE-Bench contains 518 samples: face swapping (95), reference-guided style editing (186), subject-guided inpainting (140), and virtual try-on (97). Each sample provides a source image, an editing instruction, a target description, and a reference image.

C.1.2 Metrics

We evaluate image editing along two dimensions: *Consistency* and *CLIP Semantics*. Consistency measures how well the edited image preserves the overall structure of the source image and its unedited regions. It includes Structure Distance and PSNR, MSE, LPIPS, and SSIM computed on the unedited regions. CLIP Semantics measures the alignment of the edited image with the target description and the reference image. It includes the similarity between the whole edited image and the target description (Whole), between the edited region and the target description (Edited), and between the edited image and the reference image (Reference). The Reference metric is evaluated only on the ICE-Bench reference-editing samples.

Let I_s and I_e denote the source and edited images, respectively. Let M be the binary mask of the edited region and $U = 1 - M$ the mask of the unedited region. For the unedited-region metrics, we set the edited regions in both images to zero, yielding $U \odot I_s$ and $U \odot I_e$. With image intensities normalized to $[0, 1]$, MSE is computed over these masked images, and PSNR is derived from it:

$$\text{MSE} = \frac{1}{3HW} \sum_{p=1}^{HW} \sum_{c=1}^{3} \big[U_p(I_{s,p,c} - I_{e,p,c})\big]^2, \qquad \text{PSNR} = 10 \log_{10} \frac{1}{\text{MSE}}. \tag{17}$$

SSIM and LPIPS are likewise computed on the two masked images; LPIPS uses a SqueezeNet feature network. All four metrics are computed over the full image dimensions, without normalizing by the area of the unedited region.

Structure Distance measures the overall structural difference between the source and edited images. We extract key features from the final layer of DINO ViT-B/8, form a pairwise cosine-similarity matrix over the tokens of each image, and compute the mean squared difference between the two

matrices:

$$S(I)_{ab} = \frac{\langle k_a(I), k_b(I)\rangle}{\|k_a(I)\|\,\|k_b(I)\|}, \qquad \text{Structure Distance}(I_s, I_e) = \frac{1}{T^2}\sum_{a,b=1}^{T}\big[S(I_s)_{ab} - S(I_e)_{ab}\big]^2, \tag{18}$$

where T is the number of tokens and $k_a(I)$ is the key feature of token a in image I.

For CLIP Semantics, we use CLIP ViT-L/14 and compute the cosine similarity between features, clipped below at zero. Let f_I and f_T denote the image and text encoders, t the target description, and I_r the reference image. The three scores are

$$\begin{aligned} \text{Whole} &= \max\big\{0, \cos\big(f_I(I_e), f_T(t)\big)\big\}, \\ \text{Edited} &= \max\big\{0, \cos\big(f_I(M \odot I_e), f_T(t)\big)\big\}, \\ \text{Reference} &= \max\big\{0, \cos\big(f_I(I_e), f_I(I_r)\big)\big\}. \end{aligned} \tag{19}$$

For Edited, pixels outside the edited region are set to zero before encoding the full image. Reference compares the full edited image with the reference image. Each metric is computed per sample and then averaged over the samples for which it is available.

C.2 Video Editing

C.2.1 Benchmark

We evaluate video editing on FiVE-Bench (Li et al., 2025a) and the reference-based editing subset of VIE-Bench (Mou et al., 2025). Our FiVE-Bench evaluation set comprises 419 editing instances from 100 distinct source videos: rigid object replacement (100), non-rigid object replacement (100), color change (100), material change (100), object addition (9), and object removal (10). The VIE-Bench reference-based editing subset contains 20 instances: reference-guided object addition and object replacement (10 each). Each instance has a source video and a reference image.

To accommodate the input frame requirements of the baseline methods, we uniformly sample 65 frames from each source video and set the frame rate to 24 fps. Source frames and masks are aligned by temporal index, and all methods are evaluated on output frames. Before computing frame-level metrics, we resize the source frames, edited frames, and masks to a common resolution according to the source video's aspect ratio: 624×352 (landscape), 480×480 (square), or 352×624 (portrait).

C.2.2 Metrics

We evaluate video editing along three dimensions: *Consistency*, *CLIP Semantics*, and *Temporal Quality*. The evaluation uses both frame-level image metrics and video-level metrics: PSNR, LPIPS, SSIM, and CLIP similarities are computed for each frame and averaged over the valid frames of each video, whereas MFS, MFS-Edit, and DOVER (Wu et al., 2023) are computed on the complete video. Consistency measures the preservation of unedited content using PSNR, LPIPS, and SSIM. CLIP Semantics measures alignment with the target description and reference image using Whole, Edited, and Reference. Temporal Quality assesses motion fidelity and video quality using MFS, MFS-Edit, and DOVER. Reference is evaluated only on the VIE-Bench reference-based editing instances.

The frame-level PSNR, LPIPS, SSIM, and CLIP metrics follow the image-editing definitions above. FiVE-Bench provides frame-wise editing masks. For VIE-Bench object replacement, we use manually verified seed boxes to prompt SAM 2 (Ravi et al., 2024) to segment and propagate the source object through the video. For object addition, we use full-frame masks; these instances have no unedited region and are excluded from PSNR, LPIPS, and SSIM. For the CLIP metrics, the target description is taken from `target_prompt` when available and otherwise from the editing instruction. Whole compares the full edited frame with this description, whereas Edited first zeros pixels outside the editing mask. Reference compares the full edited frame with the full reference image.

For Temporal Quality, MFS uses the FiVE-Bench implementation of CoTracker (Karaev et al., 2024) to track points in the source and edited videos and compare the directions of their displacements between adjacent frames. MFS is computed over the full frame. MFS-Edit restricts tracking to the bounding box enclosing the union of all nonempty frame-wise editing masks, yielding a fixed region across frames. DOVER evaluates each edited video and outputs a video quality score.

C.3 3D EDITING

C.3.1 BENCHMARK

Our 3D editing evaluation follows the scene and editing-instruction setup of EditSplat (Lee et al., 2025). The benchmark comprises eight scenes, each with two editing instructions, for a total of 16 editing tasks. The scenes are drawn from Instruct-NeRF2NeRF (Haque et al., 2023), Mip-NeRF 360 (Barron et al., 2022), and BlendedMVS (Yao et al., 2020). We use the corresponding benchmark annotations for the editing instructions and the source and target text descriptions.

To standardize the inputs and rendering conditions, we first reconstruct an unedited 3D Gaussian Splatting (3DGS) representation (Kerbl et al., 2023) for each source scene from the same images and camera data, training it for 30,000 iterations. For the current comparison, we use a fixed set of training camera views: each scene provides 26 views, from which we uniformly select eight for evaluation. All methods render their edited scenes from the same selected cameras, yielding 16 tasks and 128 rendered images per method. When computing the editing direction, we use the rendering of the unedited 3DGS from the same camera as the source image. We follow the EditSplat task set and recompute the scores of all methods under this unified view setting.

C.3.2 METRICS

We evaluate 3D editing in terms of semantic accuracy and multi-view consistency using four metrics. CLIPDir and CLIPSim follow EditSplat (Lee et al., 2025). We additionally use 3D Consistency, adapted from the editing-consistency metric of Group Editing (Ma et al., 2026), and Ph-Loss, the photometric reprojection loss used by RL3DEdit (Wang et al., 2026).

Let I_s^v and I_e^v denote the unedited and edited 3DGS renderings at view v, respectively, and let t_s and t_e be the source and target text descriptions. With CLIP image and text encoders f_I and f_T, respectively, CLIPDir measures the alignment between image and text change directions, whereas CLIPSim measures the similarity between the edited rendering and target description:

$$\begin{aligned} \text{CLIPDir}^v &= \cos\big(f_I(I_e^v) - f_I(I_s^v),\, f_T(t_e) - f_T(t_s)\big), \\ \text{CLIPSim}^v &= \cos\big(f_I(I_e^v), f_T(t_e)\big). \end{aligned} \tag{20}$$

Here I_s^v is rendered from the unedited 3DGS at the same camera view. Both metrics are computed at each of the eight evaluation views and averaged within each editing task; higher scores are better.

For 3D Consistency, we treat the eight edited renderings of each task as a group and average the feature similarities among the edited results to measure cross-view consistency. Higher scores indicate more consistent edits. For Ph-Loss, we evaluate photometric reprojection error over the seven pairs of adjacent views among the eight evaluation views. We use VGGT (Wang et al., 2025) to predict depth from the edited renderings and combine these depths with the dataset camera parameters to reproject an edited image into the adjacent view. We compute $0.15L_1 + 0.85L_{\text{SSIM}}$, where $L_{\text{SSIM}} = (1 - \text{SSIM})/2$, only at pixels whose reprojected depth agrees with the target-view depth. The depth-consistency threshold is set to 5% of the median valid depth in the target view. A view pair with no depth-consistent pixels is excluded from the average. The remaining pairwise errors are averaged to obtain the task-level Ph-Loss; lower values indicate better cross-view appearance and geometric consistency. Finally, we take the arithmetic mean of the valid task-level scores.

D CONSTRUCTION OF MULTI-VIEW EDITING DATA

We construct multi-view editing data from videos of static indoor scenes recorded by a moving camera. Frames from each video show the same scene from different viewpoints. Each sample contains a source sequence, an editing instruction, and an edited sequence. The pipeline consists of source sequence preparation, instruction construction, and paired edit generation.

D.1 SOURCE SEQUENCE PREPARATION

We use the RealEstate10K (Zhou et al., 2018) subset of RealCam-Vid (Zheng et al., 2025) and retain clips at a resolution of 1280×720. For each clip with at least 72 frames, we uniformly sample 72

frames over its full duration. This preserves coverage of the original camera trajectory. We encode the sampled frames as a 3-second video at 24 fps.

We filter the sampled sequences using two measures, computed on grayscale frames I_t resized to a width of 640 pixels. We compute the Laplacian variance for each frame,

$$v_t = \mathrm{Var}[\nabla^2 I_t], \tag{21}$$

where ∇^2 is the discrete Laplacian. We require a median of at least 90 and a 10th percentile of at least 40. No more than 35% of frames may have a variance below 45.

We also compute dense optical flow between consecutive frames. Let m_t denote the mean flow magnitude for each frame pair. We discard sequences whose coefficient of variation,

$$J = \sigma(m)/\mu(m), \tag{22}$$

exceeds 0.85. This measure captures variation in inter-frame flow magnitude.

D.2 Target Selection and Instruction Construction

Target selection. We use Qwen3-VL-30B-A3B (Bai et al., 2025a) to describe the objects in each sequence and estimate how reliably they can be identified across views. We select one target object for local editing. Local operations can modify, move, replace, or remove this object. For object addition, it serves as a reference for placing the new object.

Candidate objects must satisfy the following conditions, based on the model's predictions:

- Recognition confidence of at least 0.45.
- Normalized temporal span of at least 0.55.
- Tracking stability of at least 0.45.
- Center proximity of at least 0.30.
- At least two distinguishing visual cues.
- Identification as a single instance.
- Instance-count confidence of at least 0.60.

The normalized temporal span measures the interval between the object's first and last appearance relative to the sequence duration. Tracking stability indicates how reliably the model identifies the same object across frames. Center proximity indicates how close the object is to the image center.

We rank eligible objects using a weighted score. The score favors salient objects that are identified with high confidence. It also favors objects that remain visible over time, are tracked consistently, and lie near the image center. We penalize objects that may be confused with others and select the highest-ranked candidate. Sequences without an eligible target are discarded.

Editing operations. We consider ten editing operations:

- **Local edits.** Object addition, removal, replacement, and movement. Local edits also include changes to the target's color, material, or texture.
- **Global edits.** Background change, tone transfer, and style change.

We use Qwen3.5-27B (Qwen Team, 2026) to generate four instruction groups for each scene. Each group describes one editing intent using three English paraphrases. The target and operation remain the same within a group. For global edits, the prompt provides six candidates from the corresponding background, tone, or style pool.

Instruction validation and selection. We check whether each instruction describes a specific, observable edit and matches its assigned operation. The checks reject:

- Ambiguous references or references to multiple target objects.
- Internal object identifiers.

- Viewpoint-dependent directional terms.
- Requests that conflict with the assigned operation. For example, a removal instruction must not request a replacement. A replacement instruction must name both the original object and its replacement.

Each instruction is limited to 30 English words and 180 characters. A group is discarded if it contains an invalid instruction or an instruction that duplicates another in the same scene. Each scene retains up to three valid groups, with one paraphrase selected from each group. We then choose one instruction per scene to bring the overall operation distribution closer to the desired proportions.

D.3 Paired Edit Generation

We provide Kling-O1 (Kling Team et al., 2025) with the source sequence and selected instruction to generate the edited sequence. The prompt asks the model to preserve the source scene layout and depth relationships. It also asks the model to follow the original camera trajectory and perspective. Content unrelated to the edit should remain unchanged.

The final dataset contains 8,098 editing triplets. Table 10 reports the operation distribution. For reporting, material and texture changes are grouped together, as are global style and tone changes.

Table 10: Distribution of editing operations in the final dataset.

Editing operation	Count	Share
Material or texture change	2,379	29.4%
Color change	1,439	17.8%
Global style or tone change	1,008	12.4%
Object replacement	829	10.2%
Object removal	810	10.0%
Object addition	747	9.2%
Background replacement	711	8.8%
Object movement	175	2.2%
Total	8,098	100.0%

D.4 Prompts

We give the two vision-language-model prompts used above.

Scene description and target extraction.

Scene Description and Target Extraction Prompt

```
You are a video-understanding model. Given multiple frames, output
strict JSON; no Markdown, no extra explanation. Goal: extract
structured features that support precise single-target editing while
avoiding ambiguous references.
Key principles:
• Object ids must be stable and unique (e.g., obj_001, obj_002).
• Every editable foreground object must be given 2-4 non-positional
  disambiguating cues (clothing, accessories, texture, pose, etc.).
• Do not use left/right/front/back/near/far as the primary basis for
  disambiguation.
• Record each object's temporal persistence and centrality, for later
  instruction selection.
• If a judgment cannot be made reliably, do not fabricate a value;
  use null or lower the confidence.
Required output fields:
```

1. objects: array of key foreground objects, each with: id, name, saliency (0-1), confidence (0-1); attributes: {gender, age_group, clothing, accessories, colors, material, texture, pose, action_state}; quantity: {count_estimate (>=1), count_confidence (0-1), is_single_instance (true/false)}; position: {bbox_norm: [x1,y1,x2,y2], center_norm: [cx,cy], depth_layer: foreground/mid/background}; temporal_presence: {visible_frame_ratio (0-1), first_seen_norm (0-1), last_seen_norm (0-1), track_stability (0-1)}; center_proximity (0-1); unique_cues: [phrases distinguishing this object, at least 2, non-positional preferred]. (quantity is required; objects with count_estimate > 1 are filtered out downstream.)
2. scene: scene information, including location, time, weather (if determinable).
3. actions: array of actions, each with subject_id, verb, object_id_or_name, confidence (0-1).
4. global_style: overall visual style (e.g., realistic / cinematic / cartoon).
5. camera: camera motion (e.g., static/pan/tilt/zoom/handheld).
6. ambiguity_report: array of groups of potentially confusable objects, each with: candidate_ids: [...], confusion_reason, best_disambiguators: [non-positional preferred].

Return only a single JSON object.

Instruction generation.

Instruction Generation Prompt

Role. You are a video-editing-instruction designer. Based on the input video's features, generate high-quality, diverse, executable editing instructions. Generation proceeds in three steps: (1) select the editing target, (2) select the editing operator, (3) write the instruction text. The rules below are organized by these three steps.
Output format. If you use extended reasoning, keep it under 300 words and do not draft the full text of each instruction there; after reasoning, you must give the complete final JSON answer. Generate exactly 12 instructions and return strict JSON (no explanatory text). Top-level field: instructions. Each instruction has fields: id, operation_scope, operation, target, instruction, rationale -- and, when n is divisible by 3, also semantic_group, variant_id. operation_scope may only be local or global.
Step 1: Select the editing target (local = single target / global = whole frame).

- A local instruction must be a single-target edit, touching exactly one object.
- Local instructions may only edit the best-selected object, objects[0] (the sole subject); every local instruction must revolve around this single subject.
- Ambiguous or group references are forbidden (e.g., ``a model'' / ``people'' / ``one of them'' / ``all models'').
- Collective-object references are forbidden (e.g., ``blades'' / ``desks'' / ``sofas'' / ``people'').
- Do not use front/back/left/right as the primary means of disambiguation; if unavoidable, use them only as a supplement, together with at least 2 non-positional anchors.
- Target-selection priority: prefer objects with a high visible-frame ratio (visible for all or most of the video); if

several candidates are close, prefer the one with higher center proximity.

- If an object cannot be uniquely localized or its temporal stability is too low, do not generate a local instruction for it -- choose a more stable object, or switch to a global instruction.
- If there are not enough high-confidence single targets, fall back in this order: (a) keep the single-target constraint but relax center-priority from a hard constraint to a preference; (b) slightly relax the temporal-persistence threshold (still requiring unique localizability); (c) use global instructions to fill the gap; (d) if still insufficient, allow only low-risk local edits (e.g., change_color/change_material) and forbid high-ambiguity move-type edits. Ambiguous and group references remain forbidden at every fallback level.

Step 2: Select the editing operator.

- Local-only operators: remove, replace, add,move,change_color, change_material, change_texture.
- Global-only operators: change_background, tone_transfer, style_change.

A local operation must be one of remove/replace/ add/change_color/change_material/move/ change_texture; each is defined as follows:

- **add**: Introduce one new object. Three things must be stated, none omittable -- (1) the new object's specific category (e.g., ``a round wooden side table,'' not a vague term like ``an object'' or ``a prop''); (2) its position relative to an existing object (e.g., ``to the left of / in front of / against the wall behind''); (3) its orientation (e.g., ``facing the camera / facing outward''). *Example:* ``Insert a small potted plant directly behind the {name}, positioned upright against the wall and facing outward, remaining continuously in frame.''
- **remove**: Delete the target object only; pure deletion, with no wording implying a replacement or new insertion (e.g., ``replace it with'' / ``instead'' / ``and add'' / ``put a new'' are all forbidden). *Example:* ``Remove the {name} from the scene while preserving the original background continuity.''
- **move**: Reposition the target object to a different plausible location while preserving its identity, appearance, and scene consistency. The instruction must specify the new position without changing the object itself. *Example:* ``Move the {name} closer to the wall while preserving its appearance and orientation.''
- **replace**: Swap the target for a different object. Before writing, first settle in your own mind exactly what is being swapped for what -- pick a replacement that is plausible for the scene and location (e.g., swapping a kitchen-island stool for a bench, or a dining chair for a sideboard -- something that fits the scene's logic, not an arbitrary mismatched object) -- then the instruction text must name both objects explicitly (``swap the specific original object for the specific new object''), neither omittable. Vague statements that only describe a change in shape without naming the new object (e.g., ``a different-category object,'' ``an alternative design,'' ``something with a different shape'') are not acceptable, because they never say what the new object actually is. The new object's category, shape, or silhouette must differ clearly from the original, and a pure material or color change must not be phrased with ``replace'' (use change_material or change_color instead). *Example:* ``Swap the {name} for a round ottoman with a distinctly different shape and structure, keeping its footprint roughly in the same spot.'' *Counter-example* (not acceptable -- states a shape difference without naming the new object):

''Replace the {name} with a distinctly different shape, featuring a dark tone.''

- **change_color:** Change only the target object's color/hue, not its material, shape, or texture. A specific color must be named (e.g., ''a deep walnut brown,'' ''a matte forest green''), not just ''a different color.'' *Example:* ''Change the color of the {name} to a deep walnut tone with natural shading transitions.''
- **change_material:** Change only the target object's material, not its shape. A specific material must be named (e.g., ''brushed metal,'' ''matte stone,'' ''polished wood''), not just ''a different material.'' *Example:* ''Change the material of the {name} to brushed metal while keeping geometry unchanged.''
- **change_texture:** Change only the target object's surface texture/finish (smoothness, grain, pattern, etc.), not its material category, color, or shape. A specific texture description must be given (e.g., ''a fine-grain finish,'' ''a smoother pattern''). *Example:* ''Modify the texture of the {name} to a fine-grain finish with subtle roughness.''

A global instruction may only change the overall scene style/tone/background mood, and must not name any specific object. A global operation must be one of change_background/tone_transfer/style_change; each is defined as follows:

- **change_background:** Replace only the background and distant scenery, without changing the identity, position, or size of foreground objects. A specific background must be named, chosen -- from the following list curated for this particular scene -- as the option that best matches the scene's content and mood (candidates outside the list are forbidden, and the first candidate should not be chosen reflexively without real comparison): <background candidate 1> / ... / <background candidate 6>. *Example:* ''Set the background to <background candidate 1>, without altering object identities or layout.''
- **tone_transfer:** An overall color/tonal transfer (warm/cool tone, saturation, contrast, etc.), without changing specific objects or background content. A specific tone must be named, chosen from a scene-specific list of 6 candidates in the same way as above. *Example:* ''Apply <tone candidate 1> with clean shadow detail retention across the full scene.''
- **style_change:** An overall stylization of the frame, without changing specific objects or background content, only the rendering style. A specific style must be named, chosen from a scene-specific list of 6 candidates in the same way as above. *Example:* ''Apply <style candidate 1> with clean uncluttered framing.''

The instruction and target fields of a global instruction must not contain an object id or a specific object name (e.g., staircase, table, person).

Step 3: Write the instruction text.

- For a local instruction, the target field must reference the object_id and include 2-4 non-positional disambiguating cues.
- Target field format:
  - local: object_id=<id>; name=<name>; cues=<cue1|cue2|...>; temporal_presence=<ratio>; center_proximity=<score>
  - global: scene_region=<global>; theme=<overall_style_or_atmosphere>
- The instruction text must not contain fragments like ''object_id=...'' or ''obj_001'' (ids may appear only in the target field).
- The instruction must be a short, single sentence, no more than about 22 English words.

- Forbidden vague wording: unverifiable phrases such as ``visually compatible alternative,'' ``suitable counterpart,'' ``context-appropriate,'' ``subtle'' are not allowed.
- Every instruction must describe a clear, observable outcome (a color/material/texture/position change, or a recognizable style keyword).
- Instructions should be as semantically distinct from one another as possible; instruction is written in English, and rationale gives a short explanation of why the choice is reasonable.

Grouping and variant rules (required whenever n is a multiple of 3).

- All 12 instructions are strictly divided into 4 semantic groups (semantic_group = 1..4).
- Each group has exactly 3 instructions, expressing 3 different English phrasings of the same editing intent (variant_id = 1/2/3).
- Within a group, operation_scope, operation, and target must be identical; only the English wording of instruction differs.
- The 4 groups must differ from each other in intent.
- Every instruction carries the semantic_group and variant_id fields.

Quantity requirement. Exactly 12 instructions must be output.

Pre-generation self-check.

1. Does every local instruction uniquely target one object?
2. Are vague and group references avoided?
3. Is reliance on front/back/left/right as the primary anchor avoided?
4. Is the target chosen preferring long-term visibility and centrality?
5. Does the instruction avoid vague wording and state an observable outcome?
6. When n is a multiple of 3, are the grouping and variant constraints satisfied?

Input features (JSON). <the stage-1 output scene-description JSON, restricted to the selected target>

E Additional Results

Figure 9 shows qualitative examples of UniVEdit across image, video, and 3D scene editing. For each example, we show the input and the edited output produced by the given instruction.

F Limitation

One limitation is that the same geometric objectives do not benefit all modalities equally, as shown by our ablation study. Future work could explore modality-adaptive geometric supervision, particularly for videos where camera motion and object motion are entangled. In addition, our current 3D setting focuses on multi-view editing of static scenes. Extending the unified framework to dynamic 3D or 4D scene editing is also an interesting direction for future work.

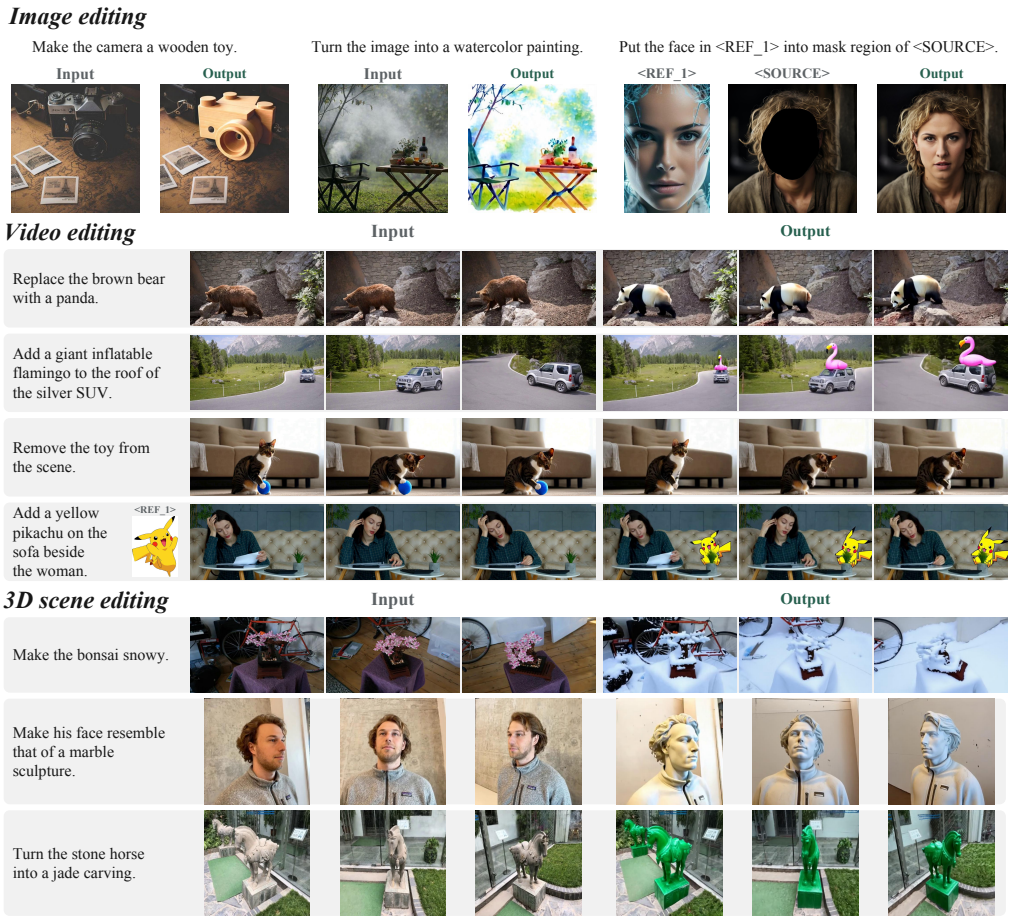

Figure 9: Qualitative examples of UniVEdit across image, video, and 3D scene editing.