INFINITYEDIT: INFINITE VIDEO EDITING WITH A LIGHTWEIGHT EDIT-IGNITION ADAPTER

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing methods have improved instruction-based video editing. However, most of them adopt an *in-place* editing pattern. They align the edited video with the given source clip over a fixed time span. This pattern fails for open-ended streams, *e.g.*, restyling a live game or applying a camera move to an ongoing shot. In such cases, edits must extend to future frames as they arrive, rather than be applied to a static input clip. In this paper, we study this setting and name it **infinite video editing**: given a preceding segment and an edit request, a model must generate the next segment that continues the stream while applying the edit. This process repeats as an unbounded sequence of edit instructions arrives. This task brings two challenges: the edit must be a faithful continuation rather than a rewrite, and quality must remain stable as edits accumulate. To address them, we propose **InfinityEdit**, a lightweight edit adapter that equips a streaming video generator with unbounded editing ability. The adapter contains three attention modules. History cross-attention guides the denoising frames using the input frames. Temporal causal self-attention keeps temporal cues flowing only from earlier frames to later ones. Edit cross-attention injects the edit request into generation. During inference, the adapter is activated only in the chunk where an edit request arrives. Subsequent chunks are generated by the original model with a reset anchor frame. This scheme applies the edit while preserving the original model's infinite generation ability. Experiments show that InfinityEdit faithfully continues the stream under each edit, and stays stable over unbounded edit sequences.

1 INTRODUCTION

Diffusion transformers can generate high-fidelity videos from text prompts (Kong et al., 2024; Wan et al., 2025b; Seedance et al., 2026). This progress has also improved instruction-based video editing, where a model modifies a given clip according to a natural-language edit request (Wu et al., 2025c; Bai et al., 2025; He et al., 2025; Bian et al., 2025). However, existing editors are built for fixed temporal windows. They follow an *in-place* editing setting. The output has the same duration as the source clip, and is aligned to the same temporal span. This makes editing a rewrite of a finite clip. It does not define how an edit should continue beyond that clip, where future frames have not yet appeared. This fixed-window setting is not suitable for open-ended video streams. In applications such as restyling a live game stream or applying a camera move to an ongoing shot, new content keeps arriving. The edit must therefore extend to future frames, rather than be applied once to a fixed clip. We study this setting and name it **infinite video editing**. Given a preceding video segment and an edit request, a model generates the next segment that continues the stream while applying the edit. This process repeats as an unbounded sequence of edit requests arrives.

Beyond faithfully applying the requested edit, infinite video editing brings two additional challenges. First, the edited segment must be a faithful continuation rather than a frame-wise rewrite. It lies beyond the input window, so the model cannot treat editing as a timestamp-wise transformation of the input clip. At the same time, entities unaffected by the edit should remain consistent with the input. Second, generation quality must remain stable as edits accumulate. Each generated segment becomes the history for the next one, so errors can propagate over time. To meet these requirements, a natural attempt is to reuse a streaming generator and switch its scene prompt when an edit request arrives. However, such generators are trained to continue the current video from its history, not to apply a specified relational edits. Their stability mechanisms can also conflict with the edit. For

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.

example, anchor frames (Zhang & Agrawala, 2025; Yang et al., 2026b) and cached multi-scale history (Yang et al., 2026a; Yuan et al., 2026) tend to pull later segments back toward the unedited content. The edit can therefore be weakened or ignored. This makes a dedicated design necessary for infinite video editing, rather than a simple prompt switch.

To address these challenges, we first build a data-collection pipeline that synthesizes `(source, edit, target)` triplets for this task. Based on this data, we propose **InfinityEdit**, a lightweight edit adapter that equips a streaming video generator with infinite editing ability. We adopt Helios (Yuan et al., 2026) as the backbone and keep it frozen. This preserves its prior for long, stable video, while only the edit adapter is trained. Our adapter is designed with three attention modules. First, history cross-attention conditions the new denoising chunk on the provided input frames. Second, temporal causal self-attention keeps temporal cues flowing only from earlier frames to later ones within the chunk. Third, edit cross-attention effectively injects the edit instruction into the denoising process. To improve stability across multiple rounds of generation, we train the adapter with history corruption, so it can tolerate imperfect generated frames as input. We also adopt a two-phase curriculum: the model first learns to apply the edit and then focuses on refining details. At inference, we uses an "ignite-then-continue" strategy. The adapter is activated only on the chunk where an edit instruction arrives, which ignites the edit in the current generated chunk. The frozen generator then continues generation and carries the edit forward until next edit instruction arrives. An anchored sliding-window history bounds memory and re-anchors generation to the edited content. In this way, InfinityEdit uses the adapter for editing while preserving the long-video prior of the frozen generator. These designs together support an infinite sequence of edits. Experiments on out-of-distribution cases show that InfinityEdit continues the stream under each edit and remains stable over long edit sequences.

Our contributions are summarized as follows:

- We study **infinite video editing**, a task where edits are applied to an ongoing video stream. We identify two key challenges: faithful continuation and stability under repeated editing.
- We design a data-collection pipeline for this task. It synthesizes data with edit-type-aware generation and provides supervision for model training.
- We propose **InfinityEdit**, a lightweight edit adapter that gives a streaming video generator infinite editing ability. The adapter applies each edit when it arrives, and the frozen generator then carries the edited content forward across later chunks.
- Extensive experiments show that InfinityEdit could effectively handle the infinite editing task.

2 Related Work

Video Generation and Editing

Text-to-image diffusion models have been proved to be a strong visual synthesis paradigm (Song et al., 2021; Rombach et al., 2022; Labs, 2024; Esser et al., 2024; Liu et al., 2025b; Zhao et al., 2026a; Tong et al., 2026). This progress has extended to video through DiT backbones (Peebles & Xie, 2022) and open generators such as CogVideoX, HunyuanVideo, and Wan (Yang et al., 2024; Kong et al., 2024; Wu et al., 2025a; Wan et al., 2025b). Recent systems further improve quality, motion, and length (OpenAI, 2024; Chen et al., 2026; Seedance et al., 2026). A related line edits existing videos with language, references, or effect transfer (Wu et al., 2025c; Bai et al., 2025; He et al., 2025; Bian et al., 2025; Lin et al., 2026; Jones et al., 2026). These methods edit a given source clip within the same temporal window, making them well suited to in-place editing applications.

Long Video Generation and Streaming Video Generation

Long video generation is commonly treated as autoregressive continuation. Each chunk is generated from previous frames (Yin et al., 2025), which makes drift a central issue when the model conditions on its own outputs. Prior work reduces this mismatch with history corruption, self-rollout, or error banks (Chen et al., 2025; Huang et al., 2025a; Li et al., 2025; Liu et al., 2025a). Streaming generation also targets low latency through step distillation and bounded history conditioning (Zhang & Agrawala, 2025; Lu et al., 2025; Zhu et al., 2026; Zhao et al., 2026b; Gu et al., 2026; Cai et al., 2026; Yang et al., 2026a). These techniques support interactive video systems (Feng et al., 2025; Huang et al., 2025b; Sun et al., 2026; Wang et al., 2026a).

Streaming Editing and Multi-shot Video Generation

Two settings are closest to our task. Streaming editors such as SANA-Streaming and LiveEdit edit live inputs with low latency (Zhao et al., 2026c; Wang et al., 2026b), but they remain in-place editing methods. Multi-shot generators such as LongLive, Anchor Forcing, and CausalCine continue a stream after prompt changes (Yang et al., 2026a;b; Meng et al., 2026), but each prompt directly describes a new target scene. Our task differs from both. It applies edit requests to an ongoing stream, and the edited result must continue into future segments. We define this setting in Section 3.

3 Preliminaries: Infinite Video Editing Task

We define infinite video editing by the temporal relation between input and output. Traditional **in-place** video editing operates on a fixed time span. Given a source clip $V_{\text{src}} \in \mathbb{R}^{T\times C\times H\times W}$ and an edit instruction, the edited clip $V_{\text{edit}} \in \mathbb{R}^{T\times C\times H\times W}$ is aligned to same temporal span T as V_{src}. Infinite video editing instead produces a temporally **subsequent** segment. Given a preceding segment $V_{\text{pcd}} \in \mathbb{R}^{T_1\times C\times H\times W}$ and an edit instruction c_{edit}, we seek a generator

$$V_{\text{tgt}} = \mathcal{G}(V_{\text{pcd}}, c_{\text{edit}}), \qquad V_{\text{tgt}} \in \mathbb{R}^{T_2\times C\times H\times W}, \tag{1}$$

where V_{tgt} continues V_{pcd} in time while satisfying c_{edit}. In general, $T_1 \neq T_2$, and target frames do not directly correspond to any input frame. We further consider the infinite regime, where a stream is edited by a sequence of edit instructions $\{c_{\text{edit}}^{(i)}\}_{i\geq 1}$:

$$V^{(i)} = \mathcal{G}\big(V^{(i-1)}[-T_1{:}], c_{\text{edit}}^{(i)}\big), \qquad V^{(0)} = V_{\text{init}}, \tag{2}$$

where $V^{(i-1)}[-T_1{:}]$ denotes the most recent T_1 frames generated so far. This task poses three challenges. First, edit alignment requires each $V^{(i)}$ to faithfully follow $c_{\text{edit}}^{(i)}$. Second, faithful continuation is harder than in-place editing because the target segment has no aligned source frames. The model must continue from history while preserving edit-dependent attributes. Style transfer should keep object motion and camera movement, while camera edits should keep subject and scene appearance. Third, repeated editing can feed errors back through the history, so quality must remain stable as edits accumulate. The latter two challenges are specific to infinite video editing. This task supports on-the-fly editing of open-ended streams, such as restyling game footage or adding camera moves to an ongoing shot.

4 Data Collection Pipeline

Training models for our task requires triplets $(V_{\text{pcd}}, c_{\text{edit}}, V_{\text{tgt}})$, where the target segment continues the preceding video while realizing a scene-grounded edit instruction. As shown in Figure 1, we construct these triplets by sampling source clips from UltraVideo (Xue et al., 2025), deriving concrete edit instructions from edit types used in VAP (Bian et al., 2025), and synthesizing type-aware target continuations. Detailed procedures are provided in Appendix C.

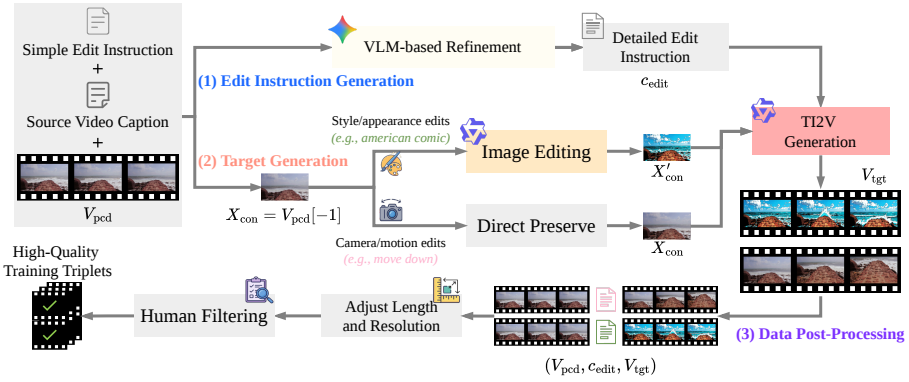


Figure 1: Our data collection pipeline.

5 METHODOLOGY

To achieve infinite video editing, we attach a lightweight edit adapter to a frozen streaming video generator. Our method includes a training recipe and an inference pipeline for repeated editing. Section 5.1 describes the architecture. Section 5.2 describes how we train the adapter to acquire editing capability. Section 5.3 describes how one trained adapter is used for infinite editing at inference.

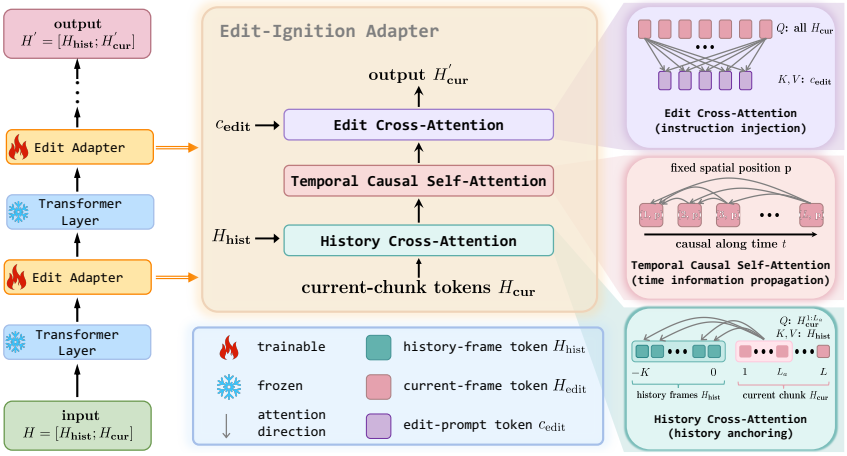


Figure 2: Our adapter's architecture. It is composed of three attention modules.

5.1 MODEL ARCHITECTURE

5.1.1 BACKBONE INTERFACE AND NOTATION

We build our editor on Helios-Distilled (Yuan et al., 2026), an autoregressive video diffusion transformer that generates videos chunk by chunk. Details are described in Appendix D.1. Here we restate only the parts used by the adapter and define the notation for this section. The backbone generates one latent video chunk at a time. At noise level σ, a diffusion transformer denoises a chunk of L frames. It is conditioned on a window of provided frames and an initial scene prompt. The history window has three temporal scales. It also starts with an anchor frame x_0, which stabilizes long-range appearance. Within each transformer layer, the hidden states concatenate the history tokens and current-chunk tokens as $H = [H_{\text{hist}}; H_{\text{cur}}]$. These per-layer hidden states, together with the multi-scale history and noise level σ, are the interfaces used by our edit adapter.

5.1.2 EDIT-IGNITION ADAPTER

The backbone already provides the generation prior needed for high-quality frames. Our goal is to add edit control without weakening this prior. We therefore freeze all base weights and train small adapter blocks that operate on the backbone features. We insert one adapter block after each transformer layer. These blocks are the only trainable components, together with a shared module that encodes the noise level σ. Given the per-layer hidden states $H = [H_{\text{hist}}; H_{\text{cur}}]$, an adapter block keeps the history tokens unchanged and refines the current-chunk tokens in three stages:

$$H^{1:L_a}_{\text{cur}} \leftarrow H^{1:L_a}_{\text{cur}} + m_\sigma\big(\textsc{HistCA}(H^{1:L_a}_{\text{cur}}, H_{\text{hist}})\big), \tag{3}$$

$$H_{\text{cur}} \leftarrow H_{\text{cur}} + \textsc{TempSA}(H_{\text{cur}}), \tag{4}$$

$$H_{\text{cur}} \leftarrow H_{\text{cur}} + m_\sigma\big(\textsc{EditCA}(H_{\text{cur}}, c_{\text{edit}})\big), \tag{5}$$

where the first stage updates only the leading L_a frames $H^{1:L_a}_{\text{cur}}$. The modulation $m_\sigma(\cdot) = (1 + s_\sigma) \odot (\cdot) + b_\sigma$ follows AdaLN, with scale and shift (s_σ, b_σ) predicted from σ by the shared sigma-embedding module. The output projection of each stage is zero-initialized. Thus, editing is learned as a residual update to the frozen prior. This keeps the original generation behavior intact.

History Cross-Attention for History Anchoring. The first stage anchors the denoising chunk to the provided history. We formulate a single attention head as $\textsc{Attn}(Q, K, V; M) =$

$\mathrm{softmax}\big(QK^\top/\sqrt{d}+M\big)\,V$, where d is the per-head key dimension. The mask is omitted ($M{=}0$) when attention is deployed. Here the queries come only from the L_a leading frames, while the keys and values come from the backbone-processed history tokens:

$$Q = H_{\text{cur}}^{1:L_a} W_Q^{\text{h}}, \quad K = H_{\text{hist}} W_K^{\text{h}}, \quad V = H_{\text{hist}} W_V^{\text{h}}, \qquad \textsc{HistCA} = \textsc{Attn}(Q, K, V; 0)\, W_O^{\text{h}}. \tag{6}$$

Restricting the queries to the first L_a frames keeps this attention inexpensive. It also lets these frames take in cues from the history, which propagates to the rest of the chunk in the next stage.

Temporal Causal Self-Attention for Forward Propagation. The second stage propagates the anchored signal forward along the temporal axis. We index the current tokens by frame $t \in \{1, \dots, L\}$ and spatial position $p \in \{1, \dots, P\}$, and denote the token at (t, p) by $(H_{\text{cur}})_{t,p}$. For each fixed spatial position p, we apply self-attention over its length-L temporal sequence:

$$\begin{gathered} Q = (H_{\text{cur}})_{:,p} W_Q^{\text{t}}, \quad K = (H_{\text{cur}})_{:,p} W_K^{\text{t}}, \quad V = (H_{\text{cur}})_{:,p} W_V^{\text{t}}, \\ \textsc{TempSA}(H_{\text{cur}})_{:,p} = \textsc{Attn}(Q, K, V; M_{\text{causal}})\, W_O^{\text{t}}, \end{gathered} \tag{7}$$

where $Q = (Q_1, \dots, Q_L)$. The causal mask sets $(M_{\text{causal}})_{t,t'} = 0$ for $t' \le t$ and $-\infty$ otherwise. We also apply a one-dimensional temporal rotary position encoding to Q and K, separate from the backbone's spatial encoding. This causal direction matches the autoregressive order of streaming generation and lets temporal cues flow only from earlier frames to later frames.

Edit Cross-Attention for Instruction Injection. The third stage provides the path for the edit instruction to enter generation. Each token from the current denoising chunk queries c_{edit}:

$$Q = H_{\text{cur}} W_Q^{\text{e}}, \quad K = c_{\text{edit}} W_K^{\text{e}}, \quad V = c_{\text{edit}} W_V^{\text{e}}, \qquad \textsc{EditCA} = \textsc{Attn}(Q, K, V; 0)\, W_O^{\text{e}}. \tag{8}$$

The output is added back to all denoising tokens, so c_{edit} can affect the whole chunk. This stage provides edit control, while the preceding two stages keep the edited chunk coherent along time.

5.2 Training Recipe

5.2.1 Flow-Matching Objective

We train the adapter with flow-matching objective. Given a clean target chunk z_0 and noise $\epsilon \sim \mathcal{N}(0, I)$, we form the interpolated state $z_\sigma = (1-\sigma)\, z_0 + \sigma\, \epsilon$ and regress the velocity $v = \epsilon - z_0$:

$$\mathcal{L} = \mathbb{E}_{z_0, \epsilon, \sigma}\big[w(\sigma)\, \|v_\theta(z_\sigma, \sigma, c) - v\|_2^2\big], \tag{9}$$

where v_θ is the adapter-augmented transformer, c includes the scene prompt, edit instruction, and history condition, and $w(\sigma)$ is a sigma-dependent loss weight.

5.2.2 History Corruption for Exposure-Bias Mitigation

At inference, the adapter conditions on previously generated frames, which may contain errors. Motivated by prior work (Chen et al., 2025; Yuan et al., 2026), we use history corruption during training to expose the adapter to such imperfect feedback. With probability p_{corrupt}, each clean history latent x is replaced by $\tilde{x} = \sigma_c\, \epsilon + (1 - \sigma_c)\, x$, where $\epsilon \sim \mathcal{N}(0, I)$ and σ_c is sampled independently per frame from a bounded range. With probability $1 - p_{\text{corrupt}}$, the history remains clean. We apply corruption independently to the long, middle, and short history scales. This design helps our adapter handle imperfect history while still working with clean history.

5.2.3 Noise Sampling and Two-Phase Curriculum

The backbone uses a pyramid denoising schedule (Jin et al., 2025; Yuan et al., 2026), where different σ ranges are coupled with different resolution stages. This is suitable for video generation, but not for our editing task. An edit may change layout or fine details. If σ and resolution change together, the adapter must learn edit control under two coupled factors. We therefore train the adapter with a single-stage schedule and sample the noise level from a Gaussian mixture centered at the denoising steps used during edit ignition:

$$\sigma \sim \sum_{k=1}^{N} \pi_k\, \mathcal{N}(\mu_k,\, \delta_k^2), \qquad \sum_{k=1}^{N} \pi_k = 1, \tag{10}$$

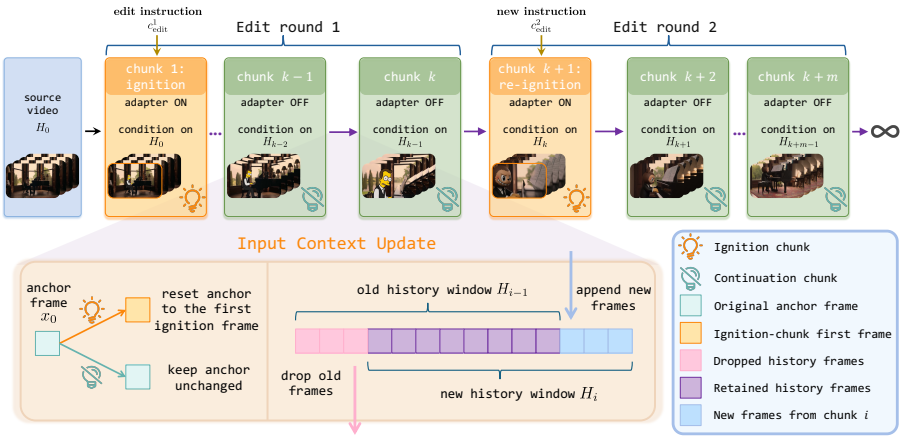


Figure 3: Inference pipeline of our method. Each edit round starts with an ignition chunk, after which subsequent chunks continue the stream.

where each center μ_k is one discrete σ sampled when applying an edit at inference. Here δ_k gives a small local spread, and π_k controls how often each center is sampled. This sampling puts more training examples near the σ values used by the adapter during inference.

Our training contains two phases and we use different mixture weights for them. The first phase uses nearly uniform π_k to cover all edit-denoising steps. The second phase shifts π_k toward middle- and low-σ components. This phase improves visual details and stabilizes frames far from the history anchor. Full mixture weights are provided in Appendix B.

5.3 Inference Pipeline

5.3.1 First-Chunk Ignition and Backbone Continuation

For each edit round at inference, our goal is to apply the edit once and then continue the edited stream. Therefore, the adapter is used only to "ignite" the edit on the first chunk after an edit instruction arrives. Later chunks are generated by the frozen backbone with the adapter disabled. The full inference pipeline is shown in Figure 3. Concretely, the first chunk is denoised by the adapter with the single-stage Euler schedule. The following chunks are denoised by the backbone with its pyramid scheduler. This switch works because the backbone conditions each new chunk on generated history. Once the first edited chunk enters the history window, later chunks inherit the edit from that history. Thus, the instruction-conditioned path is used only where the edit is introduced, while the backbone carries the edited content forward.

5.3.2 History-Window Sliding and Anchor Resetting for Infinite Editing

For long video generation, our backbone uses a sliding history window and an anchor frame x_0. The window provides the condition for future chunks, while x_0 and the initial scene prompt reduce drift. For repeated editing, we keep the sliding-window update. After each chunk, we append new frames and retain only the most recent window, so memory stays constant. To keep each edit round consistent, we update the anchor when a new edit arrives. After the adapter generates the ignition chunk for an edit, we reset x_0 to the first edited frame in that chunk. This anchor stays fixed during later backbone continuations until the next edit arrives. It helps later chunks follow the edited stream rather than revert to the source. The window provides recent edited content, and the moving anchor gives a stable reference for the current edit round. Figure 3 shows the update process.

6 Experiments

6.1 Experimental Settings

Implementation Details. We use the 14B Helios-Distilled model (Yuan et al., 2026) as our backbone and train only the edit adapter. By default, edited videos are generated at 384×640 resolution

and 16 fps. All training and evaluation are conducted on 32 NVIDIA H20 GPUs. The full training hyperparameters are provided in the Appendix A.

Benchmark. Existing video-editing benchmarks mainly evaluate in-place editing (He et al., 2025). They rely on source-to-target frame correspondence, which does not match our setting. We therefore build an OOD sequential-editing benchmark from UltraVideo (Xue et al., 2025). It contains 200 source videos from different entity categories, such as humans and scenery. For each source video, Gemini-3-Flash (Team et al., 2023) generates three scene-grounded edit instructions. The edit types are sampled from VAP (Bian et al., 2025). Each method produces three temporally continuous edit segments, with four chunks per segment.

Baselines. We compare with five baselines from three families. The *Pure Backbone* baseline uses Helios (Yuan et al., 2026) with prompt replacement. *In-Place Editing* baselines include Lucy-Edit (Team, 2025) and SANA-Streaming (Zhao et al., 2026c). *Prompt Switching* baselines include Anchor-Forcing (Yang et al., 2026b) and Infinity-RoPE (Yesiltepe et al., 2025). Methods marked with † do not take the source video as input. All source-conditioned methods use the same source video for fairness. Details on these baselines are provided in Appendix D.2.

6.2 Quantitative Comparison

We report quantitative results under two evaluation protocols: VBench metrics and a VLM-as-Judge study. We then analyze how each method behaves as edits accumulate.

6.2.1 Evaluation with VBench Metrics

We first use VBench (Huang et al., 2024) metrics for edit control, motion plausibility, and temporal stability. *Camera Motion* measures camera-edit accuracy with CoTracker2 (Karaev et al., 2024). *Motion Smoothness* uses AMT (Li et al., 2023) interpolation error. *Temporal Flickering* is measured on the final edit round to test accumulated degradation. We also report RAFT-based (Teed & Deng, 2020) *Dynamic Degree*, which is just used to reflect the motion rather than evaluating edit fidelity. Table 1 shows that our method performs best on the main metrics. It achieves the top Camera Motion and Temporal Flickering scores, while remaining close to the best Motion Smoothness. The Camera Motion gain is the largest, with about $+0.22$ over the second-best method. The final-round flickering result further shows that our method keeps temporal coherence late in the editing chain.

Table 1: Comparison with VBench metrics on our task. Best results are in **bold**.

Category	Method	Camera Motion ↑	Motion Smoothness ↑	Temporal Flickering ↑	Dynamic Degree
Pure Backbone	Helios-Base (w/o Adapter) (Yuan et al., 2026)	0.5494	**0.9869**	0.9641	0.6383
In-Place Editing	Lucy-Edit (Team, 2025)	0.5062	0.9808	0.9616	0.6067
	SANA-Streaming (Zhao et al., 2026c)	0.5432	0.9858	0.9631	0.4750
Prompt Switching	Anchor-Forcing (Yang et al., 2026b)	0.4815	0.9775	0.9536	0.7917
	Infinity-RoPE (Yesiltepe et al., 2025)	0.3580	0.9671	0.9344	0.8550
–	Ours	**0.7654**	0.9833	**0.9660**	0.7400

Table 2: VLM evaluation on our task (3 editing rounds per sample). Best results are in **bold**.

Category	Method	Edit Faithfulness ↑	Visual Quality ↑	Preservation ↑	Coherence ↑
Pure Backbone	Helios-Base (w/o Adapter) (Yuan et al., 2026)	2.637	2.850	2.790	2.885
In-Place Editing	Lucy-Edit (Team, 2025)	2.863	2.723	2.665	2.560
	SANA-Streaming (Zhao et al., 2026c)	3.303	3.093	3.060	3.030
Prompt Switching	Anchor-Forcing (Yang et al., 2026b)	2.545	2.620	1.830	3.480
	Infinity-RoPE (Yesiltepe et al., 2025)	2.658	2.675	1.865	3.265
–	**Ours**	**3.828**	**3.765**	**3.815**	**3.840**

6.2.2 VLM-as-Judge Evaluation

We also use Gemini-3.5-Flash (Team et al., 2023) as a VLM judge. It scores four dimensions on a 1–5 scale: *Edit Faithfulness*, *Visual Quality*, *Scene Identity Preservation*, and *Cross-Edit Coherence*. We use two calls per sample. The first scores faithfulness and quality **for each segment**. The second scores preservation and coherence **over the full edit chain**. Table 2 shows that our method ranks

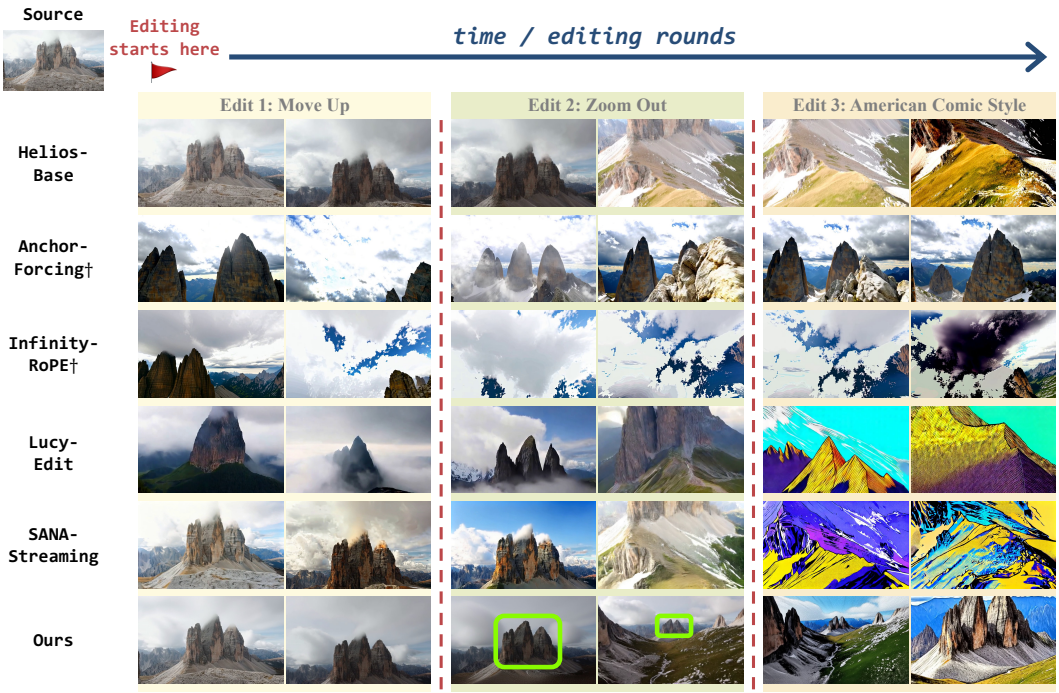


Figure 4: Qualitative comparison with various methods on the streaming video editing task, where a sequence of editing instructions (here camera-movement and style edits) is applied continuously to a scenery source video. All methods share the same source video and editing instructions. Methods marked with † take no source video as input, while all other conditions are kept identical for fairness.

first on all four dimensions. The largest margins appear in preservation and visual quality. *In-Place Editors* re-render given frames instead of continuing the stream, so they lag on cross-edit coherence. *Prompt Switching* methods show the opposite pattern. They keep relatively high coherence, but their preservation scores fall below 2.0, which denotes that they lack source-video grounding. Our method is strong on both preservation and coherence.

6.2.3 Stability across Editing Rounds

Our task requires quality and editing ability to remain stable over repeated rounds. Figure 5 tracks VBench *Aesthetic Quality* for source-conditioned methods. We exclude prompt switching because it does not take the source video as input. Among comparable methods, baselines drop by 0.02 to 0.04 after the first round. Our method stays nearly flat ($\Delta \approx 0$), showing stable visual quality under repeated editing.

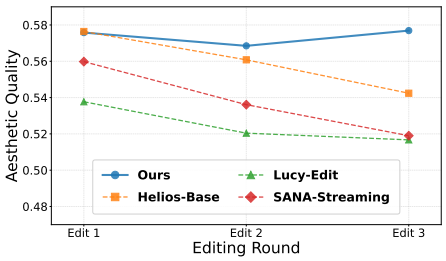


Figure 5: The aesthetic quality score recorded with different editing round.

Table 3 reports edit faithfulness for each round. Our method ranks first at every step and stays near 3.8 with low variance (std = 0.023). The *Pure Backbone* remains weak across rounds. *Prompt Switching* methods vary strongly because the first segment has no source video to rely on. *In-Place Editors* appear to improve, but this trend is misleading. They repeatedly re-edit their own outputs, which causes source drift and makes later instructions easier to impose.

6.3 Qualitative Comparison

Figure 4 shows a scenery case with camera-movement and style edits. All methods use the same source video and the same three instructions. Our method applies each edit at the intended point



Figure 6: Long-video streaming editing. We lengthen each editing segment so that the full sequence exceeds 1000 frames. Our method stays visually stable and keeps realizing each instruction. The edited attributes also persist across segments.

and carries the result forward. The green boxes track the same mountain peak. Under the *zoom-out* instruction, our method pulls the camera back so that the peak takes a smaller part of the frame. Baselines either miss the edit, drift from the source scene, or break continuity across rounds. More qualitative cases are provided in Appendix E.1.

Table 3: Edit faithfulness across sequential editing rounds. Our method maintains stable performance ($\Delta < 0.06$ between rounds), demonstrating robust sequential editing capability.

Method	Edit 1	Edit 2	Edit 3	Std ↓
Helios-Base (w/o Adapter) (Yuan et al., 2026)	2.590	2.740	2.580	0.072
Lucy-Edit (Team, 2025)	2.490	2.980	3.120	0.268
SANA-Streaming (Zhao et al., 2026c)	3.260	3.295	3.355	0.039
Anchor-Forcing† (Yang et al., 2026b)	1.530	2.955	3.150	0.715
Infinity-RoPE† (Yesiltepe et al., 2025)	1.655	3.215	3.105	0.694
Ours	**3.860**	**3.805**	**3.820**	**0.023**

6.4 Long-Video Editing

We further test long-horizon editing by extending the full sequence beyond 1000 frames, as shown in Figure 6. Our method remains stable and follows each instruction without quality collapse. The frozen Helios prior preserves long-video generation, while the adapter adds edit control. Edited attributes also persist across segments. For example, the Ladudu appearance from the second segment remains visible when the third instruction (*move up*) is applied. More cases are provided in Appendix E.2.

7 Conclusion

In this paper, we introduced **infinite video editing**, a task where edits are applied to an ongoing video stream beyond a fixed input clip. Unlike in-place editing, the model must generate the next segment while applying the requested edit. This setting requires faithful continuation and stable quality as edits accumulate. To address these challenges, we proposed **InfinityEdit**, a lightweight edit adapter for a frozen streaming video generator. The adapter has three modules: history cross-attention for grounding the new segment in the input history, temporal causal self-attention for forward temporal propagation, and edit cross-attention for injecting the edit instruction. At inference, the adapter is activated only when an edit request arrives, while the frozen generator carries the edited stream forward with a sliding history window and a reset anchor frame. Experiments show that InfinityEdit outperforms baseline methods. It follows edit requests more faithfully, continues the stream without falling back to frame-wise rewriting, and remains stable as edits accumulate over long sequences.

AI USE STATEMENT

In this work, we used generative AI tools for *generating synthetic data sets, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting dataset* . We have not used generative AI tools for *supporting qualitative and thematic data analysis, interpreting results* , and *helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, and proposing or refining hypotheses* are not applicable to this work. Additionally, we used generative AI tools for *sourcing/searching for information, editing our paper to improve readability* . We have reviewed all AI-assisted work. Two main authors check the code and examine all the texts in this paper. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not raise any ethical issues. We did not collect sensitive personal data or release datasets that could potentially cause harm. All experiments were conducted in compliance with relevant institutional guidelines and legal regulations.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we include the implementation code in the supplementary materials. Upon acceptance of this paper, subject to internal review approval and feasible conditions, we plan to further open-source the model weights and datasets to support future research.

REFERENCES

Qingyan Bai, Qiuyu Wang, Hao Ouyang, Yue Yu, Hanlin Wang, Wen Wang, Ka Leong Cheng, Shuailei Ma, Yanhong Zeng, Zichen Liu, et al. Scaling instruction-based video editing with a high-quality synthetic dataset. *arXiv preprint arXiv:2510.15742*, 2025.

Yuxuan Bian, Xin Chen, Zenan Li, Tiancheng Zhi, Shen Sang, Linjie Luo, and Qiang Xu. Video-as-prompt: Unified semantic control for video generation. *arXiv preprint arXiv:2510.20888*, 2025. URL https://arxiv.org/abs/2510.20888.

Shengqu Cai, Ceyuan Yang, Lvmin Zhang, Yuwei Guo, Junfei Xiao, Ziyan Yang, Yinghao Xu, Zhenheng Yang, Alan Yuille, Leonidas Guibas, Maneesh Agrawala, Lu Jiang, and Gordon Wetzstein. Mixture of contexts for long video generation. In *ICLR*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2025.

Guibin Chen, Dixuan Lin, Jiangping Yang, Youqiang Zhang, Zhengcong Fei, Debang Li, Sheng Chen, Chaofeng Ao, Nuo Pang, Yiming Wang, et al. Skyreels-v4: Multi-modal video-audio generation, inpainting and editing model. *arXiv preprint arXiv:2602.21818*, 2026.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *ICML*, 2024.

Tianrui Feng, Zhi Li, Shuo Yang, Haocheng Xi, Muyang Li, Xiuyu Li, Lvmin Zhang, Keting Yang, Kelly Peng, Song Han, et al. Streamdiffusionv2: A streaming system for dynamic and interactive video generation. *arXiv preprint arXiv:2511.07399*, 2025.

Yuchao Gu, Guian Fang, Yuxin Jiang, Weijia Mao, Song Han, Han Cai, and Mike Zheng Shou. Anyflow: Any-step video diffusion model with on-policy flow map distillation. *arXiv preprint arXiv:2605.13724*, 2026.

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. *arXiv preprint arXiv:2512.07826*, 2025.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. *arXiv preprint arXiv:2506.08009*, 2025a.

Yubo Huang, Hailong Guo, Fangtai Wu, Shifeng Zhang, Shijie Huang, Qijun Gan, Lin Liu, Sirui Zhao, Enhong Chen, Jiaming Liu, and Steven Hoi. Live avatar: Streaming real-time audio-driven avatar generation with infinite length. *arXiv preprint arXiv:2512.04677*, 2025b.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *CVPR*, 2024.

Yang Jin, Zhicheng Sun, Ningyuan Li, Kun Xu, Hao Jiang, Nan Zhuang, Quzhe Huang, Yang Song, Yadong Mu, and Zhouchen Lin. Pyramidal flow matching for efficient video generative modeling. In *International Conference on Learning Representations*, volume 2025, pp. 23378–23402, 2025.

Maxwell Jones, Rameen Abdal, Or Patashnik, Ruslan Salakhutdinov, Sergey Tulyakov, Jun-Yan Zhu, and Kuan-Chieh Jackson Wang. Tuning-free visual effect transfer across videos. *arXiv preprint arXiv:2601.07833*, 2026.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European conference on computer vision*, pp. 18–35. Springer, 2024.

Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv preprint arXiv:2412.03603*, 2024.

Black Forest Labs. Flux. https://github.com/black-forest-labs/flux, 2024.

Wuyang Li, Wentao Pan, Po-Chien Luan, Yang Gao, and Alexandre Alahi. Stable video infinity: Infinite-length video generation with error recycling. *arXiv preprint arXiv:2510.09212*, 2025.

Zhen Li, Zuo-Liang Zhu, Ling-Hao Han, Qibin Hou, Chun-Le Guo, and Ming-Ming Cheng. Amt: All-pairs multi-field transforms for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9801–9810, 2023.

Yiqi Lin, Guoqiang Liang, Ziyun Zeng, Zechen Bai, Yanzhe Chen, and Mike Zheng Shou. Kiwi-edit: Versatile video editing via instruction and reference guidance. *arXiv preprint arXiv:2603.02175*, 2026.

Kunhao Liu, Wenbo Hu, Jiale Xu, Ying Shan, and Shijian Lu. Rolling forcing: Autoregressive long video diffusion in real time. *arXiv preprint arXiv:2509.25161*, 2025a.

Mushui Liu, Dong She, Jingxuan Pang, Qihan Huang, Jiacheng Ying, Wanggui He, Yuanlei Hou, and Siming Fu. Tfcustom: Customized image generation with time-aware frequency feature guidance. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2714–2723, 2025b.

Yunhong Lu, Yanhong Zeng, Haobo Li, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Jiapeng Zhu, Hengyuan Cao, Zhipeng Zhang, Xing Zhu, et al. Reward forcing: Efficient streaming video generation with rewarded distribution matching distillation. *arXiv preprint arXiv:2512.04678*, 2025.

Yihao Meng, Zichen Liu, Hao Ouyang, Qiuyu Wang, Ka Leong Cheng, Yue Yu, Hanlin Wang, Haobo Li, Jiapeng Zhu, Yanhong Zeng, Xing Zhu, Yujun Shen, Qifeng Chen, and Huamin Qu. Causalcine: Real-time autoregressive generation for multi-shot video narratives. *arXiv preprint arXiv:2605.12496*, 2026.

OpenAI. Sora 2. https://openai.com/sora, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. *arXiv preprint arXiv:2212.09748*, 2022.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022.

Team Seedance, De Chen, Liyang Chen, Xin Chen, Ying Chen, Zhuo Chen, Zhuowei Chen, Feng Cheng, Tianheng Cheng, Yufeng Cheng, et al. Seedance 2.0: Advancing video generation for world complexity. *arXiv preprint arXiv:2604.14148*, 2026.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *ICLR*, 2021.

Zhiyao Sun, Ziqiao Peng, Yifeng Ma, Yi Chen, Zhengguang Zhou, Zixiang Zhou, Guozhen Zhang, Youliang Zhang, Yuan Zhou, Qinglin Lu, et al. Streamavatar: Streaming diffusion models for real-time interactive human avatars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10887–10897, 2026.

DecartAI Team. Lucy edit: Open-weight text-guided video editing. 2025. URL https://d2drjpuinn46lb.cloudfront.net/Lucy_Edit__High_Fidelity_Text_Guided_Video_Editing.pdf.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pp. 402–419. Springer, 2020.

Yunze Tong, Mushui Liu, Canyu Zhao, Wanggui He, Shiyi Zhang, Hongwei Zhang, Peng Zhang, Jinlong Liu, Ju Huang, Jiamang Wang, Hao Jiang, and Pipei Huang. Alleviating sparse rewards by modeling step-wise and long-term sampling effects in flow-based grpo. *arXiv preprint arXiv:2602.06422*, 2026.

Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wente Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025a.

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*, 2025b.

Lizhen Wang, Yongming Zhu, Zhipeng Ge, Youwei Zheng, Longhao Zhang, Tianshu Hu, Shiyang Qin, Mingshuang Luo, Jiaxu Zhang, Xin Chen, et al. Flowact-r1: Towards interactive humanoid video generation. *arXiv preprint arXiv:2601.10103*, 2026a.

Xinyu Wang, Chongbo Zhao, Fangneng Zhan, and Yue Ma. Liveedit: Towards real-time diffusion-based streaming video editing. In *European Conference on Computer Vision*. Springer, 2026b.

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.

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

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 *ICCV*, 2025c.

Zhucun Xue, Jiangning Zhang, Teng Hu, Haoyang He, Yinan Chen, Yuxuan Cai, Yabiao Wang, Chengjie Wang, Yong Liu, Xiangtai Li, and Dacheng Tao. Ultravideo: High-quality uhd video dataset with comprehensive captions. *arXiv preprint arXiv:2506.13691*, 2025.

Shuai Yang, Wei Huang, Ruihang Chu, Yicheng Xiao, Yuyang Zhao, Xianbang Wang, Muyang Li, Enze Xie, Yingcong Chen, Yao Lu, et al. Longlive: Real-time interactive long video generation. In *ICLR*, 2026a.

Yang Yang, Tianyi Zhang, Wei Huang, Jinwei Chen, Boxi Wu, Xiaofei He, Deng Cai, Bo Li, and Peng-Tao Jiang. Anchor forcing: Anchor memory and tri-region rope for interactive streaming video diffusion. *arXiv preprint arXiv:2603.13405*, 2026b.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024.

Hidir Yesiltepe, Tuna Han Salih Meral, Adil Kaan Akan, Kaan Oktay, and Pinar Yanardag. Infinity-rope: Action-controllable infinite video generation emerges from autoregressive self-rollout. *arXiv preprint arXiv:2511.20649*, 2025.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From slow bidirectional to fast autoregressive video diffusion models. In *CVPR*, 2025.

Shenghai Yuan, Yuanyang Yin, Zongjian Li, Xinwei Huang, Xiao Yang, and Li Yuan. Helios: Real real-time long video generation model. *arXiv preprint arXiv:2603.04379*, 2026.

Lvmin Zhang and Maneesh Agrawala. Packing input frame contexts in next-frame prediction models for video generation. *Arxiv*, 2025.

Canyu Zhao, Hao Chen, Yunze Tong, Yu Qiao, Jiacheng Li, and Chunhua Shen. Marble: Multi-aspect reward balance for diffusion rl. *arXiv preprint arXiv:2605.06507*, 2026a.

Min Zhao, Hongzhou Zhu, Kaiwen Zheng, Zihan Zhou, Bokai Yan, Xinyuan Li, Xiao Yang, Chongxuan Li, and Jun Zhu. Causal forcing++: Scalable few-step autoregressive diffusion distillation for real-time interactive video generation. *arXiv preprint arXiv:2605.15141*, 2026b.

Yuyang Zhao, Yicheng Pan, Qiyuan He, Jincheng Yu, Junsong Chen, Tian Ye, Haozhe Liu, Enze Xie, and Song Han. Sana-streaming: Real-time streaming video editing with hybrid diffusion transformer. *arXiv preprint arXiv:2605.30409*, 2026c.

Hongzhou Zhu, Min Zhao, Guande He, Hang Su, Chongxuan Li, and Jun Zhu. Causal forcing: Autoregressive diffusion distillation done right for high-quality real-time interactive video generation. *arXiv preprint arXiv:2602.02214*, 2026.

A TRAINING DETAILS

We keep the 14B Helios-Distilled backbone frozen and train only the edit adapter. We optimize with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-4}, $\epsilon = 10^{-8}$) in `bf16` mixed precision with gradient checkpointing. We use a per-GPU batch size of 1 with 2 gradient-accumulation steps across 32 NVIDIA H20 GPUs. Training clips are generated at 384×640 resolution, and we adopt a 16-step single-stage schedule for the validation inference path.

To mitigate exposure bias, we corrupt the conditioning history during training (frame-independent noise corruption applied with probability 0.9) and sample the noise level with a mixture-of-Gaussians sigma distribution centered on the raw sigmas seen at inference. Training proceeds in two phases: Phase 1 uses uniform sigma sampling with uniform per-frame loss weighting, and Phase 2 shifts the mixture toward the low-sigma tail and adds temporal loss reweighting to emphasize the weakly anchored chunk-tail frames.

Detailed hyperparameters and configuration is listed in Table 4.

Table 4: Detailed hyperparameters and configuration used to train our edit adapter.

Hyperparameter	**Stage 1**	**Stage 2**
Stage-specific		
Initialization	random (zero-init)	Stage 1 checkpoint
Learning rate	3×10^{-5}	1×10^{-5}
LR warmup steps	1000	0
Training steps	50,000	10,000
Sigma mixture weights	uniform	bell-shaped (Table 5)
Temporal loss reweighting	✗	✓
max reweight ratio	–	4.0
Shared across stages		
Optimization		
Optimizer	AdamW (β_1=0.9, β_2=0.999)	
LR scheduler	cosine	
Weight decay	1×10^{-4}	
Max gradient norm	1.0	
Schedule & resources		
Batch size (per GPU)	1	
Gradient accumulation	2	
EMA decay	0.999	
History conditioning		
Input history window (latent frames)	19	
History sizes (short/mid/long)	16 / 2 / 1	
Output Chunk (latent frames)	9	
History corruption	$p_{\text{corruption}} = 0.9$	
Edit Adapter architecture		
History cross-attn (dim / heads)	512 / 4	
Temporal self-attn (dim / heads)	1024 / 8	
Edit cross-attn (dim / heads)	1024 / 8	
Text embedding dim	4096	

B SAMPLED NOISE DISTRIBUTION IN TWO-PHASE TRAINING

As introduced in Section 5.2.3, we use a Gaussian mixture to control how training noises are sampled across different noise levels. The two training phases in Section 5.2.3 use different sampling distributions. We show them in Figure 7 and report the mixture settings in Table 5. In the first phase, all mixture components use uniform weights. This gives equal coverage to the σ values used during inference. In the second phase, the weights shift toward middle- and low-σ centers. This places more training samples on later denoising steps and helps refine details.

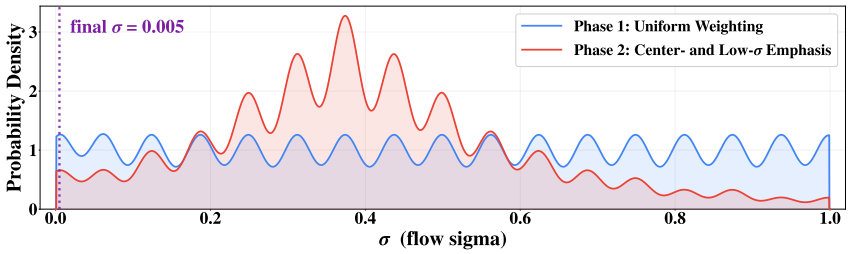


Figure 7: σ's distribution deployed for each training phase.

Table 5: The 16 Gaussian-mixture centers and their per-stage sampling weights.

Idx	Center σ	Weight (Stage 1)	Weight (Stage 2)
1	0.999	1.0	0.3
2	0.937	1.0	0.3
3	0.874	1.0	0.5
4	0.812	1.0	0.5
5	0.749	1.0	0.8
6	0.687	1.0	1.0
7	0.624	1.0	1.5
8	0.562	1.0	2.0
9	0.499	1.0	3.0
10	0.437	1.0	4.0
11	0.374	1.0	5.0
12	0.312	1.0	4.0
13	0.249	1.0	3.0
14	0.187	1.0	2.0
15	0.124	1.0	1.5
16	0.062	1.0	1.0
Per-component standard deviation δ_k		0.02	0.02
Final σ for extra sampling		0.005	0.005

C Data Collection Pipeline

As described in Section 3, training a model for our task requires triplets $(V_{\text{pcd}}, c_{\text{edit}}, V_{\text{tgt}})$. V_{pcd} is the preceding video. c_{edit} is the edit instruction. The target video V_{tgt} continues from V_{pcd} while applying c_{edit}. We construct the three parts in separate steps. Source videos V_{pcd} are sampled from UltraVideo (Xue et al., 2025) dataset and resized to a fixed resolution and frame count. Basic edit types are drawn from representative cases used by VAP (Bian et al., 2025). They are then used to construct concrete edit instructions c_{edit}. Target videos V_{tgt} are synthesized with the image-to-video model Wan2.2-I2V-A14B (Wan et al., 2025a). The following paragraphs describe edit instruction generation, edit-type-aware target video generation, and data post-processing. The full data generation pipeline is shown in Figure 1.

Edit Instruction Generation. Our basic edit types are sampled from the instruction sets used in VAP (Bian et al., 2025). These edit types are abstract phrases and are not tied to a specific source video. Thus, they cannot be used directly as edit instructions. To generate usable c_{edit}, we first group source videos V_{pcd} by the main entities in their captions. We then pair each group with suitable edit types. Given the source caption and a simple edit type, we use Gemini 3 Flash (Team et al., 2023) to expand it into a detailed edit instruction c_{edit}. In this case, the generated instruction effectively describes both the edit behavior and the specific change to apply.

Edit-type-aware Target Video Generation. Then we begin to synthesize target videos V_{tgt} from V_{pcd} and c_{edit}. Since V_{tgt} should continue from V_{pcd}, its first frame should match the end of the source video. We take the last source frame as the connection frame, $X_{\text{con}} = V_{\text{pcd}}[-1]$, and use it

to start target generation. Different edit types require different modification of this frame. A style change should show the new appearance immediately, while a camera-motion edit should keep the scene consistent at the boundary and only alter the viewpoint. We therefore group edit types into four main categories and process X_{con} according to the group. For appearance edits such as style changes, we first use Qwen-Image-Edit-2511 (Wu et al., 2025b) to obtain an edited connection frame $X'_{\text{con}} = \text{Edit}(X_{\text{con}}, c')$, where c' describes the intended change. We then use X'_{con} as the first frame of V_{tgt}. For edits that mainly change the camera view, we keep X_{con} unchanged and use it directly as the first frame. We then obtain V_{tgt} through image-to-video generation with Wan2.2-I2V-A14B.

Data Post-processing After collecting the triplets $(V_{\text{pcd}}, c_{\text{edit}}, V_{\text{tgt}})$, we first adjust them to match the input format of our backbone. As introduced in Section D.1, our base model is Helios-Distilled. We therefore resize all videos to its default training resolution. We also clip the frame counts so that the ratio $\frac{T_1}{T_2}$ matches the lengths of the full input window and the output window of the model. We then filter the triplets by manual scoring along four criteria: the alignment between V_{tgt} and c_{edit}, the consistency between V_{pcd} and V_{tgt}, the rationality of V_{tgt}, and the visual quality of V_{pcd}. Alignment measures how well the target video fulfills the instruction, *i.e.*, whether V_{tgt} conveys the intended edit type and realizes the other semantics described in c_{edit}. Consistency measures how well V_{tgt} continues from V_{pcd}, based on whether the main subjects remain stable and the scene agrees with the final source frame. Rationality checks whether the generated video contains implausible artifacts, such as a person with extra hands or an unnatural change in a subject's appearance. Visual quality measures the quality of the selected source clip from UltraVideo. Each criterion is rated on a scale from 1 to 4, with 4 as the best and 1 as the worst, and the scoring is carried out by 20 human labelers.

D Background

D.1 Base Model: Helios

Helios (Yuan et al., 2026) is a 14B autoregressive video diffusion transformer for real-time long-video generation. Helios-Distilled is its few-step distilled variant. Instead of generating a fixed-length clip in a single pass, Helios generates video chunk by chunk. It concatenates the clean history with the noisy target as input and denoises only the target. The history is kept clean and serves as the conditioning context. Each generated chunk is then appended to the history, allowing generation to continue. To keep the cost bounded as the video grows, Helios compresses the history into a hierarchical multi-scale memory with a constant token budget. Its distilled variant supports few-step, real-time inference at the 14B scale. The model is conditioned on the initial input, without an explicit design for injecting new edit instructions during intermediate generation. In this paper, we adopt Helios-Distilled as the backbone of our infinite video editor.

D.2 Baseline Descriptions

As discussed in Section 2, no prior method directly addresses infinite video editing. We therefore compare with five representative baselines from three families: *Pure Backbone*, *In-Place Editing*, and *Prompt Switching*. The *Pure Backbone* baseline uses the frozen Helios (Yuan et al., 2026) backbone. It replaces the source prompt with the edit instruction at each boundary and uses the same history input as our method. This baseline helps measure the gain from adding the edit adapter beyond prompt replacement alone. *In-Place Editing* is represented by Lucy-Edit (Team, 2025) and SANA-Streaming (Zhao et al., 2026c). These methods rewrite the source clip while preserving frame-level correspondence. Comparing with them shows the difference between forward continuation and in-place rewriting. *Prompt Switching* is represented by Anchor-Forcing (Yang et al., 2026b) and Infinity-RoPE (Yesiltepe et al., 2025). These methods continue an autoregressive stream after switching to a self-contained prompt, following a process similar to multi-shot video generation. They are architecturally close to our setting, but perform free generation rather than source-grounded editing. Because these methods do not take the source video as input, we mark them with † for distinction in the tables. All other source-conditioned methods, including pure backbone and in-place editing methods, receive the same source video for fairness.



Figure 8: Qualitative comparison with various methods on the streaming video editing task, where a sequence of editing instructions (here entity-transformation edits) is applied continuously to a human-centric source video. All methods share the same source video and editing instructions. Methods marked with † take no source video as input, while all other conditions are kept identical.

E More Experimental Results

E.1 Additional Qualitative Comparison

Figure 8 shows a human-centric case with entity-transformation edits. The green boxes mark the last frame of the first segment and an early frame of the second segment. Our method changes the character from a *Simpsons-comic* style to *Ladudu*, while keeping the background continuous. The three edits therefore read as one evolving video. In contrast, competing methods either fail to realize the target entity, lose source identity, or introduce visible discontinuities between segments.

E.2 Additional Long-Video Cases

We provide an additional long-video infinite-editing case in Figure 9 and 10, complementing the example in the main text. Consistent with our observations there, our method sustains visual stability and continues to realize each successive instruction as the sequence is extended beyond 1000 frames.



Figure 9: Additional long-video streaming editing case. This example takes animals as the main entities.



Figure 10: Additional long-video streaming editing case. This example takes scene as the main background.

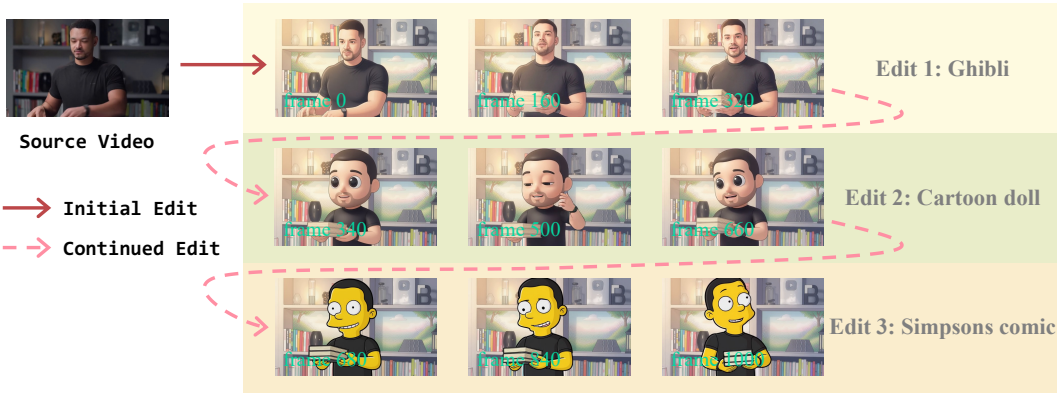


Figure 11: Additional long-video streaming editing case. This example takes a human as the main entity.