Aurora: Unified Video Editing with a Tool-Using Agent

Yongsheng Yu[1*†] **Ziyun Zeng**[1*] **Zhiyuan Xiao**[1] **Zhenghong Zhou**[1]
Hang Hua[2] **Wei Xiong**[3] **Jiebo Luo**[1]

[1]University of Rochester [2]MIT-IBM Watson AI Lab [3]NVIDIA

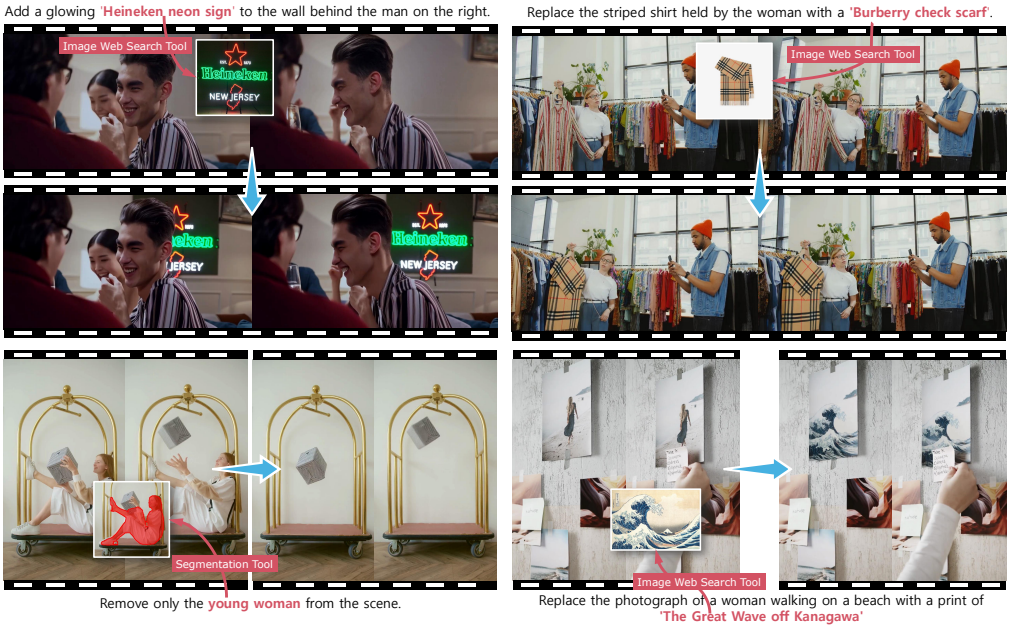


Figure 1: **Aurora** converts underspecified instructions into model-ready editing conditions. The VLM agent recovers missing reference images via web image search and missing masks via segmentation.

Abstract

Recent video editing models have converged on a unified conditioning design: a single diffusion transformer jointly consumes text, source video, and reference images, and one set of weights covers replacement, removal, style transfer, and reference-driven insertion. The design is flexible, but it assumes that the user provides model-ready text, reference images, and spatial grounding for local edits, which real requests often omit. We present Aurora, an agentic video editing framework that pairs a tool-augmented vision-language model (VLM) agent with a unified video diffusion transformer. The VLM agent maps a raw user request to a structured edit plan aligned with the transformer's conditioning channels, resolving textual and visual underspecification before generation. We train the VLM agent with supervised data for complete edit planning and reference-image selection, together with preference pairs for robust tool use and instruction refinement. We introduce AgentEdit-Bench to evaluate agent-enhanced video editing under textual and visual underspecification. Experiments on AgentEdit-Bench and two existing benchmarks show that Aurora improves over instruction-only baselines and that the VLM agent transfers to compatible frozen video editing models.

*Equal contribution. †Project lead. Code: github.com/yeates/Aurora Homepage: yeates.github.io/Aurora-Page

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

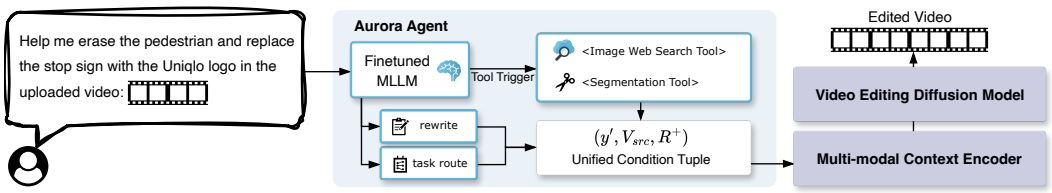


Figure 2: **Aurora pipeline.** The Aurora VLM agent (a LoRA-finetuned Qwen3-VL-8B-Instruct [24]) parses the raw user request, triggers tools, rewrites the instruction, and routes the task. The output is the unified conditioning tuple (y', V_{src}, R^+) that the Aurora video diffusion transformer consumes; the task label belongs to the agent plan and supports the rewrite but is not fed to the video model.

1 Introduction

Recent video editing models increasingly accept several conditioning inputs in one model: text specifies the edit, the source video supplies content to preserve, and optional reference images or masks specify appearance and location [36, 16, 19, 39]. This design lets one set of weights support replacement, removal, style transfer, and reference-driven insertion, but it also assumes that each request comes with the appropriate conditioning inputs: a precise text instruction, a reference image that shows a requested real-world entity or visual pattern, and a mask when the edit is spatially localized. In practice, however, user requests often leave these channels incomplete. We categorize this incompleteness along two axes. *Visual underspecification* occurs when the instruction specifies an edit target or region without providing the visual input needed to ground it. For example, *"Replace the striped shirt held by the woman with a Burberry check scarf"* names a specific visual pattern but provides no reference image. *Textual underspecification* occurs when the desired edit is described only indirectly, as in *"Change the blue item to a vehicle that runs much faster"*, where the model must infer the concrete editing goal. Unified video editing models thus provide channels for text, reference images, and masks, but do not fully resolve how missing conditions should be inferred, grounded, or filled from underspecified requests such as those in Figure 1.

This gap is best understood as a condition-construction problem. Unified video editing models already define the conditioning tuple they expect; what remains open is how to construct it from a raw user request. The missing piece may be a more complete instruction, a reference image for a requested real-world entity or visual pattern, or spatial grounding for a local edit. Each arises before video generation begins. Accordingly, we decouple condition construction from video generation.

We present Aurora, an agentic video editing framework built around this split, shown in Figure 2. Our design has two parts. First, a tool-augmented VLM agent maps a source video and a raw user request to a four-field edit plan: a rewritten instruction, a task label, an optional image-search query, and an optional mask phrase. The first two fields resolve textual underspecification through rewriting and task routing; the latter two recover missing visual conditions through web image search and grounded segmentation. When the source clip and the user's text already specify the edit, the tool fields remain empty and the framework reduces to ordinary instruction-conditioned editing.

Second, Aurora uses a unified video diffusion transformer (DiT) whose conditioning paths match those plan fields. Built on Wan2.2-TI2V-5B [34], the Aurora video DiT consumes the resulting conditions along two paths: a shared multimodal context for the rewritten instruction, sampled source frames, and reference images, and a latent token sequence where source and reference tokens accompany the noisy video tokens. The video DiT itself does not rewrite instructions or call tools. It consumes the conditioning tuple produced by the VLM agent.

We also introduce AgentEdit-Bench, a video editing benchmark that evaluates this two-part framework jointly rather than each part in isolation. Each case centers on a request underspecified along the textual or the visual axis, so the framework succeeds only if it fills in the missing visual input or instruction text with a concrete editing goal. This structure supports three evaluation protocols on the same source clips: our VLM agent with our video DiT, our VLM agent with any unified video editing model, and any video editing model under the raw user instruction alone. The first measures the full framework, the second isolates VLM agent transfer across models, and the third measures the VLM agent's gain over an instruction-only baseline. Our contributions are as follows.

- We present Aurora, an agentic video editing framework whose tool-augmented VLM agent maps raw user requests to more complete, model-ready conditions for a unified video DiT.
- We construct VLM agent training data for complete edit planning, reference-image selection, and DPO-based preference alignment.
- We introduce AgentEdit-Bench, a benchmark designed to evaluate video editing enhanced by a tool-augmented VLM agent under textual and visual underspecification.

2 Related work

2.1 Instruction-conditioned and unified video editing

Early instruction-conditioned video editing methods adapted image-editing techniques to video by sharing attention across frames [23, 41, 9]. Senorita-2M [48] scales supervision with task-specialist synthesizers, and OpenVE [11] and Ditto [1] place VLMs in the data-construction pipeline to generate instructions and filter low-quality edits. Recent methods move toward unified video editing models that accept richer conditioning inputs, including instructions, source videos, reference images, masks, in-context examples, or reasoning frames [15, 19, 36, 16, 39]. As conditioning becomes richer, the bottleneck shifts from what the model can ingest to what the user can reliably provide. Their conditioning interfaces accept references and masks as inputs, but the burden of producing those inputs, when the raw request omits them, is left to the user or to upstream pipelines. Aurora closes this gap with a trained tool-augmented VLM agent that reads the raw request, retrieves a reference image when needed, grounds a mask when a spatial region must be localized, and rewrites the instruction into a model-ready form, then hands a complete conditioning tuple to the video editing model.

2.2 Agentic visual generation and video agents

The closest related work is in image generation, where language-model agents retrieve external knowledge or reference images, decompose user requests, choose tools, and score intermediate results with visual feedback or learned rewards [7, 42, 20, 40, 45]. Their shared pattern is to keep planning outside the generator, so that the generator receives more complete conditions. Existing video-domain agents target complementary settings: generalist video generation, compositional text-to-video generation, model selection, or multi-step workflows over model libraries [44, 14, 33, 18, 13]. The Aurora VLM agent addresses a different problem: rather than coordinating a library of video models, it translates a raw instruction into the conditions that a single unified video DiT requires.

3 Method

Aurora pairs a tool-augmented VLM agent with a unified video DiT. Aurora uses two separate Qwen-family checkpoints: the VLM agent is a LoRA-finetuned Qwen3-VL-8B-Instruct [24] that emits the edit plan, and the multimodal context encoder inside the video DiT is a frozen Qwen3.5-4B [25] that turns the plan and source video into cross-attention tokens. The two share no weights and play distinct roles throughout the paper. We write π_ϕ for the VLM agent, where ϕ collects the trainable parameters (the LoRA adapter on top of Qwen3-VL-8B-Instruct described in Sec. 3.1). Given a source video $V_{\text{src}} \in \mathbb{R}^{F\times H\times W\times 3}$, a raw user instruction y, and an optional user reference-image set $R = \{r_1, \ldots, r_K\}$, the target edited video V_{edit} should realize the requested change while preserving the identity, motion, and background structure of V_{src} wherever the instruction does not ask for a change. The method has three parts. The VLM agent first maps (V_{src}, y, R) to a complete edit plan and constructs concrete conditions from optional tool calls. The resulting conditions are then assembled into a unified conditioning tuple used by the video DiT. Finally, the video DiT is trained with flow matching, and the VLM agent with supervised planning and preference alignment.

3.1 Agent planning and condition construction

The VLM agent first predicts a typed edit plan:

$$(y', c, q, m) = \pi_\phi(V_{\text{src}}, y, R), \tag{1}$$

where y' is the rewritten instruction used by the video DiT, $c \in \mathcal{C}$ is a task label, $q \in \Sigma^* \cup \{\emptyset\}$ is an optional image-search query, and $m \in \Sigma^* \cup \{\emptyset\}$ is an optional mask phrase. We constrain the

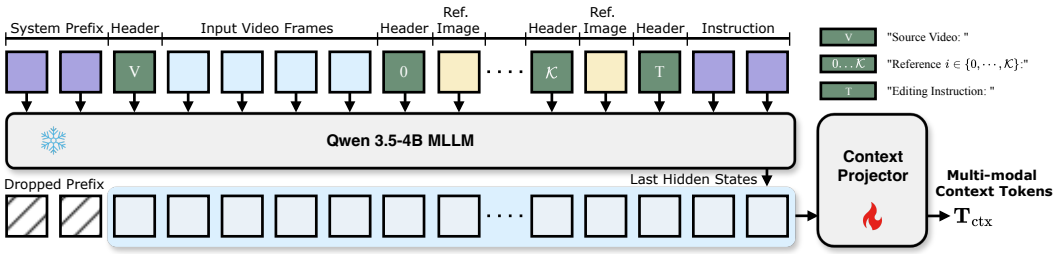


Figure 3: **Multimodal context encoder.** A frozen Qwen3.5-4B MLLM encodes the instruction, sampled source frames, and reference images into a shared context for DiT cross-attention.

output space so that user-provided references suppress web image search, and a non-empty search query is allowed only for search-eligible tasks such as addition, replacement, and background change. The plan fields separate the two forms of underspecification introduced in Section 1. The rewritten instruction y' turns the free-form user request into a model-ready form consistent with the task label c. The image-search query q retrieves a reference image for a requested real-world entity or visual pattern, and the mask phrase m localizes the regions named in the instruction.

Only q and m trigger external tools. We use $\mathcal{T}$ for these tool operators. Both produce reference images of the same type as those in R, so the downstream video DiT does not need a separate mask branch. $\mathcal{T}_{\text{search}}$ denotes web image retrieval (via the Serper API [29]) followed by reference-image selection, returning a single selected image. $\mathcal{T}_{\text{mask}}$ denotes grounded segmentation followed by mask compositing onto V_{src}, returning a masked image of the same resolution. A tool-execution stage converts non-empty plan fields into these images and merges them with the user references:

$$R^{+} = R \cup \mathcal{T}_{\text{search}}(q) \cup \mathcal{T}_{\text{mask}}(m, V_{\text{src}}). \tag{2}$$

An empty plan field denotes no tool call: we adopt the convention $\mathcal{T}_{\text{search}}(\emptyset) = \emptyset$ and $\mathcal{T}_{\text{mask}}(\emptyset, \cdot) = \emptyset$, so $q = \emptyset$ or $m = \emptyset$ contributes nothing to R^{+}. After condition construction, the inputs that the video DiT consumes are gathered into a conditioning tuple:

$$\mathbf{x} = (y', V_{\text{src}}, R^{+}). \tag{3}$$

We use $\mathbf{x}$ for this conditioning tuple throughout; the task label c is not fed to the video DiT.

Figure 4 summarizes this protocol, which turns an intermediate plan into complete conditions. When image search is triggered (the value `image_search` is not empty), a candidate set is retrieved via the Serper API [29] and a reference-image selection step keeps a single image. When mask generation is triggered (the value `mask` is not empty), GroundingDINO [21] and Segment Anything [27] localize the target and composite a masked image onto V_{src}. Both outputs are inserted into R^{+} as reference images. The VLM agent architecture itself is deliberately simple: Qwen3-VL-8B-Instruct with a lightweight LoRA adapter [24, 12], generating the four-field JSON output directly. The key design choice is the fixed four-field output, and agent training leaves the video DiT unchanged.

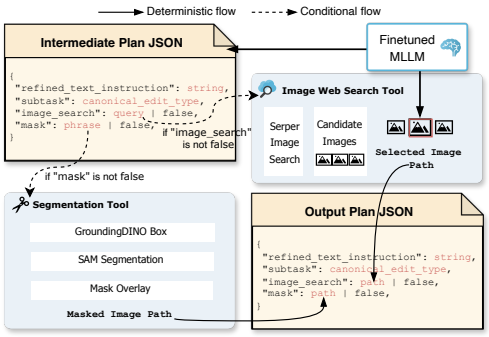


Figure 4: **Protocol for tool-augmented condition construction.**

3.2 Unified conditioning in the video DiT

Built on Wan2.2-TI2V-5B [34], the Aurora video DiT consumes the conditioning tuple $\mathbf{x}$ along two paths. A multimodal context sequence carries the rewritten instruction with sampled source frames and reference images into cross-attention, while a latent token sequence carries noisy video tokens with source and reference tokens into self-attention, as Figures 3 and 5 show.

Multimodal context encoder. We use a frozen Qwen3.5-4B multimodal encoder [25] to encode the rewritten instruction y', sampled source frames from V_{src}, and reference images in R^{+}. The final

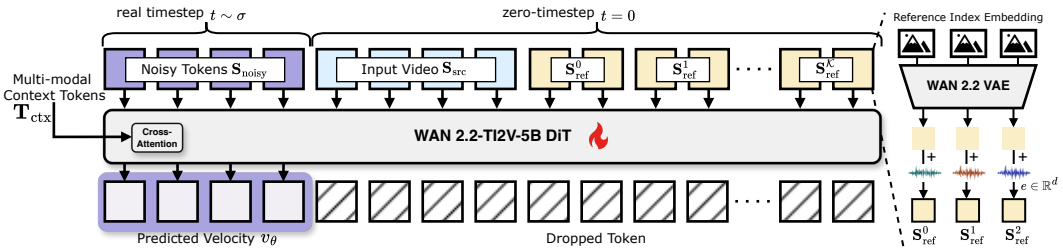


Figure 5: **Aurora video DiT.** It predicts velocity from noisy video tokens that attend to source and reference tokens concatenated along the sequence dimension under zero-timestep modulation.

hidden states are mapped to the DiT cross-attention dimension with a zero-initialized linear projector:

$$\mathbf{T}_{\text{ctx}} = \text{Linear}\big(\text{MLLM}(y', V_{\text{src}}, R^+)\big)\,. \tag{4}$$

The multimodal context $\mathbf{T}_{\text{ctx}}$ enters DiT cross-attention, so the instruction, source frames, and selected references share one multimodal context. This Qwen3.5-4B encoder is distinct from the VLM agent backbone of Sec. 3.1; only a linear projector is trained, and the encoder stays frozen.

Latent token sequence and zero-timestep modulation. Let $\mathcal{E}$ denote the frozen Wan VAE encoder [34], $\text{Patchify}(\cdot)$ the DiT patch embedding, and $\mathbf{z}_t$ the noisy video latent at flow-matching timestep t (defined in Eq. (7)). To preserve unedited content and the identity of reference images, these visual conditions are concatenated along the token-sequence dimension via the operator $[\,\cdot\,;\,\cdot\,]$:

$$\begin{aligned} \mathbf{S}_{\text{noisy}} = \text{Patchify}(\mathbf{z}_t), \mathbf{S}_{\text{src}} = \text{Patchify}(\mathcal{E}(V_{\text{src}})), \mathbf{S}_{\text{ref}} = \big[\,\text{Patchify}(\mathcal{E}(r_k)) + e_k\,\big]_{r_k \in R^+}, \\ \mathbf{S} = [\mathbf{S}_{\text{noisy}}; \mathbf{S}_{\text{src}}; \mathbf{S}_{\text{ref}}], \end{aligned} \tag{5}$$

where e_k is a learnable reference-index embedding following [37]. If $R^+ = \emptyset$, then $\mathbf{S}_{\text{ref}}$ is empty. Each reference image is VAE-encoded at its own aspect ratio, and only the noisy video latent is decoded at output time. Mask conditioning has no separate branch: a binary mask is first composited as a masked image, then inserted into R^+ like any other reference image.

Noisy, source, and reference tokens play different roles: noisy tokens move along the flow-matching trajectory, while source and reference tokens remain fixed across timesteps and provide conditioning. Wan2.2 DiT blocks receive timestep embeddings via AdaLN modulation [34]. We build real-timestep and zero-timestep modulations $a(t)$ and $a(0)$, then assign each token a selector η_i:

$$a_i = (1 - \eta_i)a(t) + \eta_i a(0), \qquad \eta_i = \begin{cases} 0, & i \in \text{noisy tokens}, \\ 1, & i \in \text{source or reference tokens}. \end{cases} \tag{6}$$

Conditioning tokens therefore do not change their modulation with the denoising noise level, while noisy tokens follow the normal flow-matching trajectory.

3.3 Training objectives

Video DiT training. Given the conditioning tuple $\mathbf{x}$, the video DiT predicts a velocity field $v_\theta(\mathbf{z}_t, t; \mathbf{x})$ over VAE latents. Let $\mathbf{z}_1 = \mathcal{E}(V_{\text{edit}})$ be the target-video latent, let $\mathbf{z}_0 \sim \mathcal{N}(0, I)$ be Gaussian noise, and set $\mathbf{z}_t = (1 - t)\mathbf{z}_0 + t\mathbf{z}_1$. Training uses the standard flow-matching objective

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t, \mathbf{z}_0, \mathbf{z}_1, \mathbf{x}} \Big[\big\| v_\theta(\mathbf{z}_t, t; \mathbf{x}) - (\mathbf{z}_1 - \mathbf{z}_0) \big\|_2^2 \Big]\,, \tag{7}$$

with samples from image editing, instruction-conditioned video editing, and reference-conditioned video editing mixed under the same objective. The model therefore sees empty-reference edits, user-reference edits, and masked-image edits as instances of one conditioning tuple. Prompt dropout and visual dropout train the unconditional and visual-negative branches that the inference-time guidance schedule uses (Appendix E); the training data mixture is described in Sec. 4.

VLM agent supervised finetuning (SFT). After training the DiT to use unified conditions, we freeze it and train the VLM agent to construct those conditions. Agent SFT has two supervised tasks: complete edit planning and reference-image selection. Let $\mathcal{D}_{\text{plan}}$ contain VLM agent context $s = (V_{\text{src}}, y, R)$ and target plan $p^\star = (y', c, q, m)$. The planning loss is

$$\mathcal{L}_{\text{SFT}}(\phi) = -\mathbb{E}_{(s, p^\star) \sim \mathcal{D}_{\text{plan}}} \log \pi_\phi(p^\star \mid s). \tag{8}$$

The supervision target is the complete edit plan $p^\star = (y', c, q, m)$ of Eq. (1), not a natural-language rationale; tool execution then turns this plan into the conditioning tuple of Eq. (3). Reference-image selection uses a separate image-choice set and the same token-level negative log likelihood, but outputs a single candidate index such as `Image 3`. SFT thus directly supervises the inference-time decisions to rewrite, route the task, issue a search query, select a reference, or request a mask.

VLM agent preference alignment. SFT yields valid plans but does not fully resolve borderline cases between similar plans, so we apply DPO [26] to plan-level preference pairs by minimizing

$$\mathcal{L}_{\mathrm{DPO}}(\phi) = -\mathbb{E}_{(s,p^+,p^-)} \log \sigma \left[\beta \log \frac{\pi_\phi(p^+ \mid s)\pi_{\mathrm{ref}}(p^- \mid s)}{\pi_{\mathrm{ref}}(p^+ \mid s)\pi_\phi(p^- \mid s)} \right], \tag{9}$$

where s is the VLM agent context, p^+ and p^- are the chosen and rejected plans, π_ϕ and π_{ref} are the current and frozen reference policies, σ is the logistic function, and β is the DPO temperature. Preference pairs are built around five plan-level boundary cases: source-entity false triggers (image search called for an entity already visible in the source video), ambiguous mask phrases (phrase too vague to ground cleanly), false image-search triggers (search called for edits that need no real-world reference, such as style transfer), constraint-losing rewrites (rewrite drops user-specified constraints from the original prompt), and task-routing confusions (a similar but wrong edit category is chosen). Because the video DiT is frozen throughout VLM agent training, the preference stage sharpens the VLM agent's decisions without mixing video sampling variance into the learning objective.

4 Data construction

Two interdependent datasets teach the video DiT to use source, text, reference, and masked-image conditions and teach the VLM agent which conditions to supply at inference time.

Video editing model data. The video DiT is trained entirely on open-source datasets. Table 1 reports the video DiT training mixture. Its three subsets correspond to the conditioning signals each sample teaches: image-edit pairs provide diverse instruction-following supervision, instruction-based video edits provide temporal source conditioning, and reference-guided video edits provide identity, and masked-image conditions. A subset of the open-source video-edit sources is filtered and relabeled with Gemini Flash-Lite [10]; identity references are synthesized with FLUX.2-klein [2]; and masked images are curated separately. Appendix G.1 gives source-by-source curation and filtering details.

Table 1: **Video DiT training data.** Counts are accepted pairs after curation. † marks filtered sources, $^{\mathrm{cap}}$ re-captioned sources, and $^{\mathrm{syn}}$ reference synthesis, captioning, or masked-image generation.

Training subset	Sources	Count
Image editing	CrispEdit-2M [5], UltraEdit [47], TextEdit [32]	2,388,440
Instruction video editing	ReCo [46]†, Ditto [1]†, OpenVE [11]†, EgoEdit [17]†, ROSE [22]$^{\mathrm{cap}}$, EffectErase [8]$^{\mathrm{cap}}$, Ditto combined-task data [1]†	1,672,008
Reference-guided video editing	OpenS2V [43]†$^{\mathrm{syn}}$, RefVIE [19], SpatialVID [35]†, ROSE [22]$^{\mathrm{syn}}$, EffectErase [8]$^{\mathrm{syn}}$, HuMoSet [4]†	609,988

VLM agent supervised data. The VLM agent is trained with three supervision types, with examples illustrated in Figures 14, 15, and 16. Planning SFT starts from accepted editing pairs and degrades clean instructions into shorter or more ambiguous user requests, with the four-field plan as supervision. Reference-image selection uses retrieved candidate sets and supervises which single image should occupy the reference slot. We use 25K planning examples and 10K reference-image selection examples; detailed construction appears in Appendix G.

Preference alignment. Preference pairs target boundary cases that the supervised plans alone do not separate, covering the same five categories defined in Sec. 3.3: source-entity false triggers, ambiguous mask phrases, false image-search triggers, constraint-losing rewrites, and task-routing confusions. Each pair keeps the source and most plan fields fixed while changing one decision axis, so DPO refines the VLM agent's decision boundaries. The final set contains 1.8K chosen/rejected pairs. The appendix provides illustrated examples.

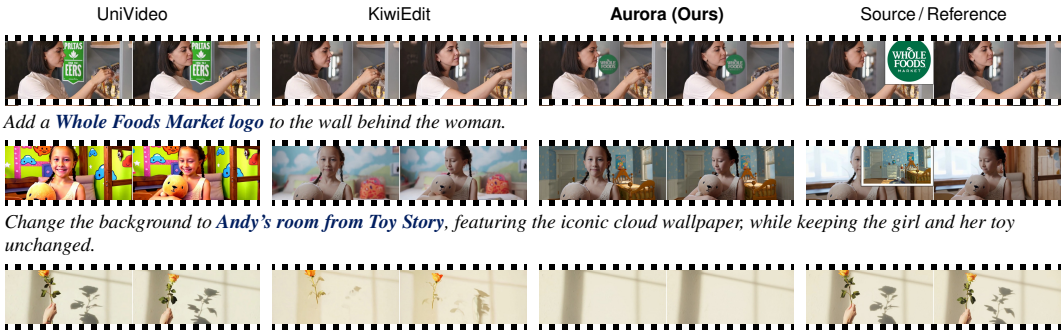


Figure 6: **Qualitative comparison on AgentEdit-Bench.** Rows show IP (named real-world entity) addition, IP background change, and localized removal. Target objects to edit are highlighted in (**bold blue**). Reference images come from Aurora's web image search tool and are not provided to the baselines in this figure, since this is their default deployment when the user request is underspecified. The complementary setup, baselines paired with the same Aurora VLM agent, is reported quantitatively in the ✓ rows of Table 2 and qualitatively in Figures 12 and 13.

5 Experiments

We organize the experiments around three questions: whether Aurora improves editing under underspecified user requests, which part of the framework is responsible for that gain, and how far the same VLM agent transfers across video editing models. The first question drives our main evaluation, in which AgentEdit-Bench is the primary benchmark for underspecified requests, while EditVerse-Bench [16] and OpenVE-Bench [11] act as sanity checks that the framework remains competitive when the instruction is already well specified. Ablation studies then address the other two questions, isolating the source of the gain and measuring its transfer across video editing models.

5.1 Experiment setup

Benchmarks. EditVerse-Bench [16] is built from 100 source videos and 200 instruction-video pairs spanning 20 edit categories, with horizontal and vertical formats evenly balanced. Following the standard source-video protocol used in current open-source comparisons, we report results on the 120-case subset, which removes tasks that require extra controls such as depth, pose, or sketch inputs. OpenVE-Bench [11] contains 431 video-edit pairs across eight categories. Our training data does not cover its creative-edit or camera-edit categories, so we report the other six categories.

Implementation details. Aurora video DiT uses a two-stage training recipe: a low-resolution warmup at 399,360 pixels followed by high-resolution finetuning at 921,600 pixels, both with 81 frames. Unless noted otherwise, all reported Aurora numbers use the Stage 2 checkpoint with 50 denoising steps. Benchmarking scores are judged by Gemini 2.5 Pro [6] at `temperature=0`, while video frame handling and scoring prompts follow the corresponding benchmark-specific evaluation protocol. In a human study with 15 annotators on 60 cases sampled from the three benchmarks, the judge's per-case rankings of the competing outputs agree with the human rankings at a mean Spearman correlation of 0.65 (Appendix F.1). Appendix E provides more details.

5.2 AgentEdit-Bench

Design goal. Existing editing benchmarks mostly measure prompt following under model-ready instructions. AgentEdit-Bench instead targets requests whose conditions are incomplete: it contains 150 cases spanning five task types, each one built around a request that is underspecified along the textual or the visual axis. It is a VLM-as-judge benchmark: Gemini 2.5 Pro [6] scores each case with a rubric that measures instruction following, edit region localization, source preservation, visual quality, and temporal consistency. The benchmark maps the two underspecification axes to five task types: IP replacement, IP addition, IP background change, localized removal, and reasoning edits.

Table 2: **Main results on AgentEdit-Bench**. Gemini 2.5 Pro [6] scores each case against a rubric; we report the percentage of the rubric maximum (higher is better), best per column in bold.

Method	Aurora Agent	IP Replace	IP Add	IP Background Change	Reasoning	Removal	Overall
UniVideo [36]	×	67.1	63.4	65.6	77.3	66.5	67.0
UniVideo [36]	✓	80.3	81.3	66.8	74.6	68.1	76.8
Kiwi-Edit [19]	×	66.7	65.1	86.6	79.5	69.0	69.7
Kiwi-Edit [19]	✓	71.6	73.1	78.3	84.3	57.8	71.7
Aurora (Ours)	×	73.0	68.0	71.2	86.4	83.0	74.7
Aurora (Ours)	✓	**89.6**	**85.3**	**90.1**	**88.1**	**86.8**	**87.9**

Here, IP refers to a named real-world entity, such as a brand, product, character, or artwork. For the three IP tasks, the rubric also scores IP presence and IP identity match. Appendix H provides the composition, full scoring rubric, judge prompts, and examples.

Quantitative results. Table 2 compares Aurora against UniVideo [36] and Kiwi-Edit [19], and also measures each model with and without the Aurora VLM agent on identical source clips. Even without the agent, the Aurora video DiT reaches 74.7 Overall and already outperforms UniVideo (67.0) and Kiwi-Edit (69.7) under the same raw-prompt input, so the gain on AgentEdit-Bench is not driven by the agent stage alone. Web image search is one way to supply a missing condition, not a requirement of the benchmark: at the video-model level, the IP tasks are subject-driven, text-conditioned editing, which all three models were trained for, and web search is handled by the agent rather than built into any video DiT. Adding the VLM agent raises Aurora from 74.7 to 87.9, and the gains are largest on the IP tasks, especially IP replacement and IP addition, where the raw request names an entity but omits the reference image the video DiT needs. The VLM agent closes that gap by selecting a concrete reference image before generation, so the model need not infer the appearance from text.

Qualitative comparison. Figure 6 shows that when the request omits the reference image needed for a requested real-world entity or visual pattern, or the spatial grounding needed for a local edit, raw-prompt baselines often synthesize a generic substitute or edit the wrong region, whereas Aurora resolves the missing condition before sampling. Appendix A adds more AgentEdit-Bench cases.

5.3 Results on existing video editing benchmarks

We next evaluate Aurora on existing video editing benchmarks whose inputs are already sufficiently specified. This section checks whether the agentic design that helps on AgentEdit-Bench also preserves strong general editing quality outside the underspecified setting. Table 3 reports Overall scores judged by Gemini 2.5 Pro [6] on the 120-case source-video subset of EditVerse-Bench [16] (0–10 scale) and on OpenVE-Bench [11] (1–5 scale), whose task distribution differs from EditVerse-Bench and includes both scene-level and localized edits. On both benchmarks, the prompts and any required reference images are already supplied by the benchmark itself. We disable the VLM agent's web image search and mask tools throughout this section; the VLM agent only rewrites the instruction. Aurora is the strongest open-source method on both benchmarks. The closed-source Runway Aleph [28] baseline reaches 3.51 Overall on OpenVE-Bench, 0.13 above Aurora, so Aurora closes most but not all of the open-vs.-closed gap on that benchmark. On OpenVE-Bench, our training data does not include subtitle removal, so the Subtitle column is reported as zero-shot generalization; even under this stricter setting, Aurora is the best in that column, and the 3.38 Overall is therefore a conservative estimate. For a like-for-like comparison against Kiwi-Edit [19], which does not natively support subtitle removal, Tab. 5(a)(b) reports a five-category

Table 3: **Overall scores on existing video editing benchmarks.** Gray rows denote closed-source baselines; "–" means the score is not reported.

Method	EditVerse-Bench	OpenVE-Bench
Runway Aleph [28]	7.17	3.51
EditVerse [16]	7.52	–
VACE [15]	4.60	1.55
InsViE [38]	–	1.58
LucyEditDev [31]	4.65	2.12
Ditto [1]	–	2.25
OpenVE-Edit [11]	–	2.57
UniVideo [36]	6.12	3.10
Senorita-2M [48]	6.54	–
Kiwi-Edit [19]	7.00	–
Aurora (Ours)	**7.61**	**3.38**

Table 5: **Agent transfer ablation** across video editing models on EditVerse-Bench [16] and OpenVE-Bench [11]. † denotes the five-category OpenVE-Bench average excluding Subtitle removal.

(a) OpenVE-Bench†

Setting	Overall
Kiwi-Edit	3.02
+ Aurora agent	**3.29**
Δ	+0.27

(b) OpenVE-Bench†

Setting	Overall
Aurora (Ours)	3.31
+ Aurora agent	**3.46**
Δ	+0.15

(c) EditVerse-Bench

Setting	Overall
UniVideo	6.12
+ Aurora agent	**6.48**
Δ	+0.36

(d) EditVerse-Bench

Setting	Overall
Aurora (Ours)	7.25
+ Aurora agent	**7.61**
Δ	+0.36

average (†) that excludes Subtitle on the same source clips. Per-category breakdowns and additional qualitative comparisons appear in Tables 6–7 and Appendices B–C.

5.4 Ablation study

VLM agent training stages and components. Table 4 ablates the VLM agent on Aurora's own video DiT. Without an agent, the video DiT scores 74.7 on AgentEdit-Bench; supervised planning raises this to 85.0, and DPO adds a further 2.9 points to 87.9. Supervised planning therefore carries most of the gain by teaching the VLM agent which conditioning channels each request needs, while DPO sharpens the boundary cases defined in Sec. 3.3 that SFT leaves ambiguous, such as false image-search triggers, ambiguous mask phrases, and task-routing confusions. The lower block of Table 4 ablates one part of the full agent at a time: without image search and reference selection, learned selection, instruction rewriting, or mask grounding, the score drops by 11.6 to 13.0 points. These drops indicate that the gain does not come from a single module and that instruction rewriting and visual-condition recovery both matter.

Table 4: **Ablation of agent training stages and components** on AgentEdit-Bench.

Configuration	Overall
Aurora, no agent	74.7
+ agent, SFT only	85.0
+ agent, SFT + DPO	**87.9**
Full agent with one part removed	
w/o search & selection	74.9
w/o selection	76.1
w/o rewriting	76.3
w/o mask grounding	75.9

Training-free VLM agents. Table 12 compares the trained VLM agent with training-free alternatives on the OpenVE-Bench subset (†) of Table 5(b), with the video DiT and judge fixed. Zero-shot planners with all tools enabled score 3.16 with Qwen3-VL-8B-Instruct [24], our agent's base model, and 3.18 with Qwen3.5-9B [25]; zero-shot prompt rewriting with the same two VLMs scores 3.19 and 3.09. All four fall below the raw prompt at 3.31, while the trained VLM agent reaches 3.46. The video DiT already handles diverse raw instructions, so changes to the input help only when they are needed: an untrained planner can add unnecessary or inaccurate tool calls, and an untrained rewriter can alter adequate prompts or drop constraints that the video DiT needs. Our agent, which only rewrites the instruction on OpenVE-Bench, is instead trained to decide when to preserve or rewrite a prompt, and constraint-losing rewrites are among the boundary cases that its preference pairs target (Sec. 3.3). Because OpenVE-Bench inputs are sufficiently specified, the +0.15 gain is small in absolute terms; the VLM agent's main gain appears on underspecified requests, where it raises the overall AgentEdit-Bench score by 13.2 points (Table 2).

Tool reliability and error propagation. Errors can enter at four stages: the planner can rewrite the instruction incorrectly, miss a needed tool call or make an unnecessary one, or give an inaccurate mask phrase; retrieval can return a reference with the wrong identity, variant, or viewpoint; segmentation can place the mask, and with it the edit, on the wrong region; and the video DiT can fail despite correct intermediate conditions. Disabling any one agent component in the ablation above costs more than 11 points, so failures in planning, retrieval, or segmentation can directly reduce editing quality. On AgentEdit-Bench, however, neither tool fails outright: every image search returns four or five candidates, from which the VLM agent picks the top-ranked one in 75.0% of searches and one of the top three in 96.3%, and every mask request returns a mask. Against human annotations, masks reach a mean IoU of 0.778 and a median of 0.960, and 15.4% fall below 0.5, mainly when the prompt does not uniquely specify the target. Planning takes 7.13 s per case on average and 4.01 s at the median on one H100 GPU, and each tool call typically adds less than 0.5 s (Appendix F.4).

VLM agent transfer across video editing models. Table 5 tests whether the fixed Aurora VLM agent still helps when only the video editing model changes. Each pair compares the same model with and without the VLM agent's outputs. On OpenVE-Bench, adding the VLM agent improves Kiwi-Edit [19] from 3.02 to 3.29 and improves our video DiT from 3.31 to 3.46. The two OpenVE-Bench panels (†) report a five-category average that excludes Subtitle removal, so each model is evaluated on a category set it natively supports. On EditVerse-Bench, the same setup improves UniVideo from 6.12 to 6.48 and improves our video DiT from 7.25 to 7.61. Both benchmarks already supply prompts and any required reference images or masks, so the VLM agent's image search and mask tools are disabled here and its only job is to rewrite the instruction. Together with the transfer rows in Table 2, these gains show that the Aurora VLM agent helps even as a pure instruction-rewriting module and that its benefit carries over to frozen video editing models it was never trained with. Each frozen model receives the agent's outputs through its native text and reference-image inputs (Appendix D). This evaluation is made possible by the open-source releases of UniVideo [36] and Kiwi-Edit [19], which we thank. Figures 12 and 13 illustrate this transfer qualitatively.

6 Limitations

Aurora has five main limitations. First, we align the VLM agent only with offline DPO on plan-level preferences (Sec. 4), and plan preference is an imperfect proxy for final video quality: on 200 preference pairs rendered by the same video DiT, preferred plans score 3.402 on average against 3.172 for rejected plans, yet the rejected plan yields the better video in 67 pairs (Appendix F.5). Online RL such as GRPO [30] over the full editing pipeline, where the reward would come from the edited clip rather than from the plan, is impractical under our compute budget. Few-step distillation of the video editing DiT, combined with joint online RL training [3] of the VLM agent and the distilled DiT, is a natural way to close this gap. Second, tool errors can propagate to the final video: a wrong reference can change the target identity, and an inaccurate mask can move the edit to the wrong region. Because the agent only predicts the mask phrase, a stronger segmentation backend can replace the current one without retraining. Third, the four-field plan is designed for a single primary edit; multi-object, multi-reference, or temporally staged requests would need list-valued fields or a sequence of plans, together with a video editing model that accepts several conditions. Fourth, all benchmark scores come from a single VLM judge whose pooled agreement with human ratings is moderate, so future evaluations would benefit from several judges. Finally, the 5B video DiT handles removal, stylization, and insertion of objects with limited motion reliably, yet adding a new subject with large, physically plausible motion remains difficult. We attribute this to the motion-modeling capacity of a 5B backbone and the coverage of large-motion insertion cases in our editing data, and plan to address it with a larger backbone and a broader training mixture.

Broader impacts. Like other video editing models, Aurora can be misused for non-consensual identity manipulation, and the image-search tool adds copyright and likeness risks from retrieved references. Aurora should therefore be treated as a creative tool, not an evidence-producing one.

7 Conclusion

This paper presents Aurora, an agentic video editing framework for user requests that leave out information the video model needs. Aurora places a tool-augmented VLM agent in front of a unified video editing model. The VLM agent rewrites the instruction, assigns a task label, and, when needed, retrieves a reference image or grounds a mask before generation. We also introduce AgentEdit-Bench, a benchmark for textual and visual underspecification that measures the full framework and isolates the contribution of the VLM agent stage on the same source clips. Experiments show that adding the VLM agent raises the overall AgentEdit-Bench score from 74.7 to 87.9 with the same Aurora video DiT. On EditVerse-Bench and OpenVE-Bench, where the inputs are already sufficiently specified, Aurora remains strong and the VLM agent still helps through instruction rewriting alone. The transfer results further show that this benefit is not tied to a single video editing model and carries over to other unified video editing models. Progress in video editing depends not only on stronger video DiTs, but also on reliably turning raw user requests into model-ready conditions.

References

[1] 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.

[2] Black Forest Labs. FLUX.2 [klein]: Towards interactive visual intelligence. Blog post, January 2026. URL https://bfl.ai/blog/flux2-klein-towards-interactive-visual-intelligence. Model page: https://bfl.ai/models/flux-2-klein.

[3] Huanqia Cai, Sihan Cao, Ruoyi Du, Peng Gao, Steven Hoi, Zhaohui Hou, Shijie Huang, Dengyang Jiang, Xin Jin, Liangchen Li, et al. Z-image: An efficient image generation foundation model with single-stream diffusion transformer. *arXiv preprint arXiv:2511.22699*, 2025.

[4] Liyang Chen, Tianxiang Ma, Jiawei Liu, Bingchuan Li, Zhuowei Chen, Lijie Liu, Xu He, Gen Li, Qian He, and Zhiyong Wu. HuMo: Human-centric video generation via collaborative multi-modal conditioning. *arXiv preprint arXiv:2509.08519*, 2025.

[5] Wei Chow, Linfeng Li, Lingdong Kong, Zefeng Li, Qi Xu, Hang Song, Tian Ye, Xian Wang, Jinbin Bai, Shilin Xu, Xiangtai Li, Junting Pan, Shaoteng Liu, Ran Zhou, Tianshu Yang, and Songhua Liu. EditMGT: Unleashing potentials of masked generative transformers in image editing. *arXiv preprint arXiv:2512.11715*, 2025.

[6] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025.

[7] Kaituo Feng, Manyuan Zhang, Shuang Chen, Yunlong Lin, Kaixuan Fan, Yilei Jiang, Hongyu Li, Dian Zheng, Chenyang Wang, and Xiangyu Yue. Gen-Searcher: Reinforcing agentic search for image generation. *arXiv preprint arXiv:2603.28767*, 2026.

[8] Yang Fu, Yike Zheng, Ziyun Dai, and Henghui Ding. EffectErase: Joint video object removal and insertion for high-quality effect erasing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[9] Michal Geyer, Omer Bar Tal, Shai Bagon, and Tali Dekel. Tokenflow: Consistent diffusion features for consistent video editing. In *ICLR*, 2024.

[10] Google DeepMind. Gemini 3.1 Flash-Lite. Model card, 2026. URL https://deepmind.google/models/gemini/flash-lite/. Model card PDF: https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-1-Flash-Lite-Model-Card.pdf.

[11] 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.

[12] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

[13] Hang Hua, Ziyun Zeng, Yizhi Song, Yunlong Tang, Liu He, Daniel Aliaga, Wei Xiong, and Jiebo Luo. Mmig-bench: Towards comprehensive and explainable evaluation of multi-modal image generation models. *Advances in Neural Information Processing Systems*, 2026.

[14] Kaiyi Huang, Yukun Huang, Xuefei Ning, Zinan Lin, Yu Wang, and Xihui Liu. Genmac: compositional text-to-video generation with multi-agent collaboration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[15] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing. *arXiv preprint arXiv:2503.07598*, 2025.

[16] Xuan Ju, Tianyu Wang, Yuqian Zhou, He Zhang, Qing Liu, Nanxuan Zhao, Zhifei Zhang, Yijun Li, Yuanhao Cai, Shaoteng Liu, et al. Editverse: Unifying image and video editing and generation with in-context learning. *arXiv preprint arXiv:2509.20360*, 2025.

[17] Runjia Li, Moayed Haji-Ali, Ashkan Mirzaei, Chaoyang Wang, Arpit Sahni, Ivan Skorokhodov, Aliaksandr Siarohin, Tomas Jakab, Junlin Han, Sergey Tulyakov, Philip Torr, and Willi Menapace. EgoEdit: Dataset, real-time streaming model, and benchmark for egocentric video editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[18] Zhengyang Liang, Daoan Zhang, Huichi Zhou, Rui Huang, Bobo Li, Yuechen Zhang, Shengqiong Wu, Xiaohan Wang, Jiebo Luo, Lizi Liao, et al. Univa: Universal video agent towards open-source next-generation video generalist. *arXiv preprint arXiv:2511.08521*, 2025.

[19] 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.

[20] Yunlong Lin, Linqing Wang, Kunjie Lin, Zixu Lin, Kaixiong Gong, Wenbo Li, Bin Lin, Zhenxi Li, Shiyi Zhang, Yuyang Peng, Wenxun Dai, Xinghao Ding, Chunyu Wang, and Qinglin Lu. JarvisEvo: Towards a self-evolving photo editing agent with synergistic editor-evaluator optimization. *arXiv preprint arXiv:2511.23002*, 2025.

[21] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *European Conference on Computer Vision*, 2024.

[22] Chenxuan Miao, Yutong Feng, Jianshu Zeng, Zixiang Gao, Hantang Liu, Yunfeng Yan, Donglian Qi, Xi Chen, Bin Wang, and Hengshuang Zhao. ROSE: Remove objects with side effects in videos. *Advances in Neural Information Processing Systems*, 38, 2025.

[23] Chenyang Qi, Xiaodong Cun, Yong Zhang, Chenyang Lei, Xintao Wang, Ying Shan, and Qifeng Chen. Fatezero: Fusing attentions for zero-shot text-based video editing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15932–15942, 2023.

[24] Qwen Team. Qwen3-VL: A family of open multimodal large language models. Model card and blog post, 2025. URL https://qwenlm.github.io/blog/qwen3-vl/.

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

[26] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, 2023.

[27] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *ICLR*, 2025.

[28] Runway. Introducing runway aleph. https://runwayml.com/research/introducing-runway-aleph, 2025. Accessed: 2025-09-10.

[29] Serper. Serper: The world's fastest and cheapest Google search API. https://serper.dev/, 2024. Accessed: 2026-04-25.

[30] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[31] DecartAI Team. Lucy edit: Open-weight text-guided video editing. 2025.

[32] Changyao Tian, Danni Yang, Guanzhou Chen, Erfei Cui, Zhaokai Wang, Yuchen Duan, Penghao Yin, Sitao Chen, Ganlin Yang, Mingxin Liu, Zirun Zhu, Ziqian Fan, Leyao Gu, Haomin Wang, Qi Wei, Jinhui Yin, Xue Yang, Zhihang Zhong, Qi Qin, Yi Xin, Bin Fu, Yihao Liu, Jiaye Ge, Qipeng Guo, Gen Luo, Hongsheng Li, Yu Qiao, Kai Chen, and Hongjie Zhang. InternVL-U: Democratizing unified multimodal models for understanding, reasoning, generation and editing. *arXiv preprint arXiv:2603.09877*, 2026.

[33] Rong-Cheng Tu, Wenhao Sun, Zhao Jin, Jingyi Liao, Jiaxing Huang, and Dacheng Tao. Spagent: Adaptive task decomposition and model selection for general video generation and editing. *IEEE Transactions on Image Processing*, 2026.

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

[35] Jiahao Wang, Yufeng Yuan, Rujie Zheng, Youtian Lin, Jian Gao, Lin-Zhuo Chen, Yajie Bao, Yi Zhang, Chang Zeng, Yanxi Zhou, Xiao-Xiao Long, Hao Zhu, Zhaoxiang Zhang, Xun Cao, and Yao Yao. SpatialVID: A large-scale video dataset with spatial annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[36] 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. *arXiv preprint arXiv:2510.08377*, 2025.

[37] Chenyuan Wu, Pengfei Zheng, Ruiran Yan, Shitao Xiao, Xin Luo, Yueze Wang, Wanli Li, Xiyan Jiang, Yexin Liu, Junjie Zhou, Ze Liu, Ziyi Xia, Chaofan Li, Haoge Deng, Jiahao Wang, Kun Luo, Bo Zhang, Defu Lian, Xinlong Wang, Zhongyuan Wang, Tiejun Huang, and Zheng Liu. Omnigen2: Towards instruction-aligned multimodal generation. *arXiv preprint arXiv:2506.18871*, 2025.

[38] 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 IEEE/CVF International Conference on Computer Vision*, pages 16692–16701, 2025.

[39] Xiangpeng Yang, Ji Xie, Yiyuan Yang, Yan Huang, Min Xu, and Qiang Wu. VideoCoF: Unified video editing with temporal reasoner. *arXiv preprint arXiv:2512.07469*, 2025.

[40] Mingde Yao, Zhiyuan You, King-Man Tam, Menglu Wang, and Tianfan Xue. PhotoAgent: Agentic photo editing with exploratory visual aesthetic planning. *arXiv preprint arXiv:2602.22809*, 2026.

[41] Danah Yatim, Rafail Fridman, Omer Bar-Tal, Yoni Kasten, and Tali Dekel. Space-time diffusion features for zero-shot text-driven motion transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8466–8476, 2024.

[42] Ruijie Ye, Jiayi Zhang, Zhuoxin Liu, Zihao Zhu, Siyuan Yang, Li Li, Tianfu Fu, Franck Dernoncourt, Yue Zhao, Jiacheng Zhu, et al. Agent banana: High-fidelity image editing with agentic thinking and tooling. *arXiv preprint arXiv:2602.09084*, 2026.

[43] Shenghai Yuan, Xianyi He, Yufan Deng, Yang Ye, Jinfa Huang, Bin Lin, Jiebo Luo, and Li Yuan. OpenS2V-Nexus: A detailed benchmark and million-scale dataset for subject-to-video generation. *Advances in Neural Information Processing Systems*, 38, 2025.

[44] Zhengqing Yuan, Ruoxi Chen, Zhaoxu Li, Haolong Jia, Lifang He, Chi Wang, and Lichao Sun. Mora: Enabling generalist video generation via a multi-agent framework. *arXiv preprint arXiv:2403.13248*, 2024.

[45] Ziyun Zeng, Hang Hua, and Jiebo Luo. Mira: Multimodal iterative reasoning agent for image editing. *arXiv preprint arXiv:2511.21087*, 2025.

[46] Zhongwei Zhang, Fuchen Long, Wei Li, Zhaofan Qiu, Wu Liu, Ting Yao, and Tao Mei. Region-constraint in-context generation for instructional video editing. *arXiv preprint arXiv:2512.17650*, 2025.

[47] Haozhe Zhao, Xiaojian Ma, Liang Chen, Shuzheng Si, Rujie Wu, Kaikai An, Peiyu Yu, Minjia Zhang, Qing Li, and Baobao Chang. UltraEdit: Instruction-based fine-grained image editing at scale. *Advances in Neural Information Processing Systems*, 37, 2024.

[48] Bojia Zi, Penghui Ruan, Marco Chen, Xianbiao Qi, Shaozhe Hao, Shihao Zhao, Youze Huang, Bin Liang, Rong Xiao, and Kam-Fai Wong. Señorita-2m: A high-quality instruction-based dataset for general video editing by video specialists. *arXiv preprint arXiv:2502.06734*, 2025.

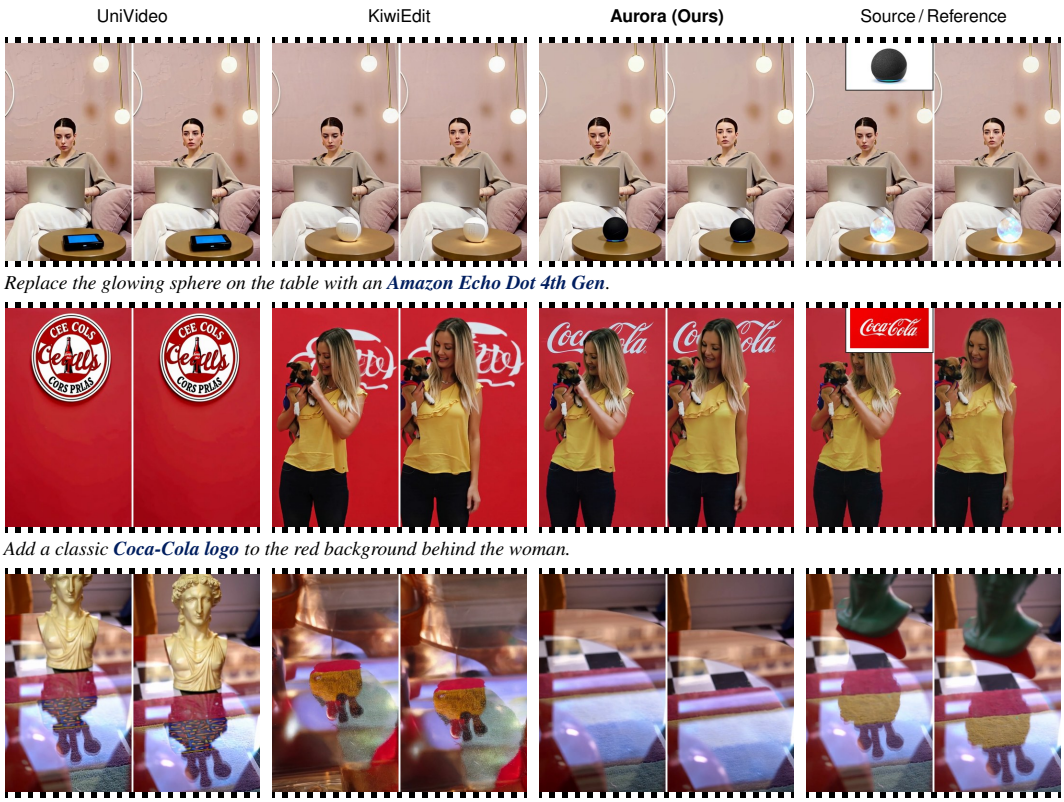


Figure 7: Extended qualitative comparison on AgentEdit-Bench. Reference images come from the Aurora VLM agent's web image search tool and are not provided to the baselines, which see only the original underspecified request.

A Extended qualitative comparison on AgentEdit-Bench

Figures 7, 8, and 9 extend Figure 6 with additional AgentEdit-Bench cases against the same two baselines (UniVideo [36] and Kiwi-Edit [19]) under their default deployment, in which they use the raw user prompt without a VLM agent. For IP cases, Aurora uses the retrieved reference image from our fine-tuned VLM agent, which is shown in the figure as a small inset overlaid on the first source frame. The cases are grouped by clip orientation: Figure 9 collects landscape clips covering replacement, removal, and addition; Figures 7 and 8 collect portrait clips covering the same five edit types. Across these cases, the two baselines either drop the named entity, substitute a generic one, or fail to remove the target cleanly. In contrast, Aurora more faithfully fulfills the raw user request via the VLM agent's rewritten instruction and, when needed, tool calls that retrieve a reference image. The gains are clearest on edits that require preserving the identity of a real-world entity.

B Qualitative comparison on EditVerse-Bench

Table 6 expands the EditVerse-Bench Overall column from Table 3 into the full per-category breakdown. Figure 10 compares Aurora against four open-source baselines (LucyEditDev [31], VACE [15], Kiwi-Edit [19], UniVideo [36]) on three EditVerse-Bench [16] cases that together cover reference-conditioned insertion, background change (i.e., Bkg.), and reasoning. The first case is a landscape clip shown as the two distant frames sampled at 20% and 80% of the clip to check temporal consistency.

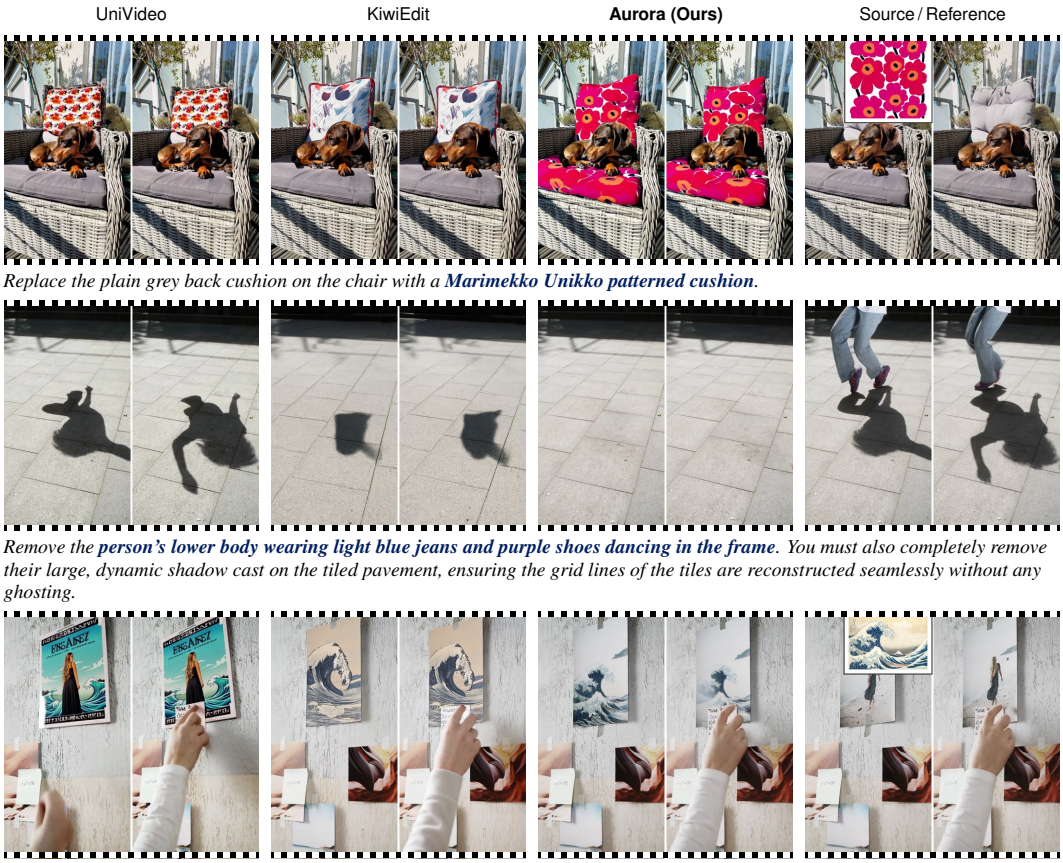


Figure 8: Extended qualitative comparison on AgentEdit-Bench.

Table 6: Per-category quantitative comparison on EditVerse-Bench [16]. Scores are obtained using Gemini 2.5 Pro [6]. Gray rows denote closed-source reference methods; bold marks the best open-source score per column. Overall scores are summarized in Table 3.

Method	Add	Remove	Change	Style	ID	Reason	Bkg.	Color	Material	Effect	Weather	Combined	Overall
Runway Aleph [28]	6.68	7.40	7.70	7.80	6.45	7.85	7.93	7.95	6.70	6.03	6.40	7.13	7.17
EditVerse [16]	6.93	8.10	7.40	8.73	7.50	7.73	7.68	7.90	7.30	6.25	7.53	7.25	7.52
VACE [15]	2.90	2.28	4.85	7.83	4.18	4.98	5.85	6.30	4.00	3.23	4.80	3.98	4.60
LucyEditDev [31]	4.67	3.70	4.00	4.40	5.30	3.27	4.63	7.50	5.27	4.33	4.30	4.47	4.65
UniVideo [36]	6.00	6.20	5.73	7.80	7.00	5.70	6.97	5.87	4.80	5.30	6.27	5.80	6.12
Senorita-2M [48]	5.83	7.13	6.53	8.33	6.63	6.63	6.63	7.20	6.07	5.17	6.40	5.93	6.54
Kiwi-Edit [19]	6.73	6.85	**7.53**	8.43	6.75	7.38	6.93	**8.10**	6.33	6.13	7.18	5.65	7.00
Aurora (Ours)	**7.35**	**8.00**	7.03	**8.63**	**7.68**	**7.70**	**7.93**	7.73	**7.15**	**6.58**	**7.93**	**7.55**	**7.61**

C Qualitative comparison on OpenVE-Bench

Table 7 expands the OpenVE-Bench Overall column from Table 3 into the full per-category breakdown. Figure 11 compares Aurora against UniVideo and Kiwi-Edit on four OpenVE-Bench [11] cases covering local removal, local addition, and local change. These cases are shown as the two distant frames (sampled at 25% and 75% of the clip) to check temporal consistency.

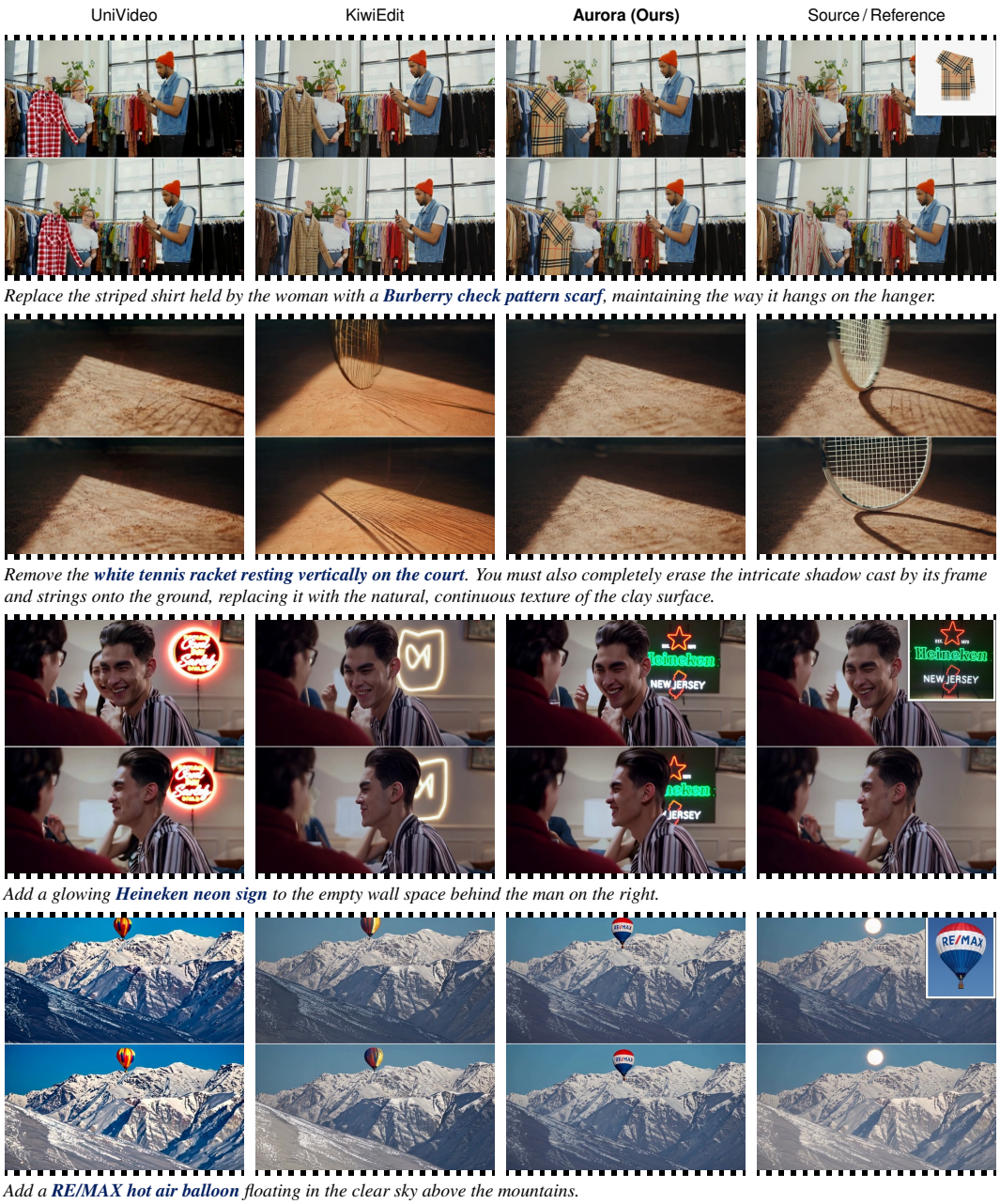


Figure 9: Extended qualitative comparison on AgentEdit-Bench.

D Qualitative VLM agent transfer across video DiTs

The frozen video editing models are neither modified nor finetuned. The VLM agent's outputs are mapped to each model's native inputs: the rewritten instruction goes to the text input, and a retrieved reference or masked image goes to the reference-image input together with the source video. The task label, search query, and mask phrase are used only for planning and tool execution and are not passed to the video editing model. This input-level mapping works because UniVideo and Kiwi-Edit both support text- and reference-conditioned video editing.

Figures 12 and 13 isolate the VLM agent's contribution by holding the receiving video DiT fixed and toggling the VLM agent stage on and off. UniVideo [36] and Kiwi-Edit [19] are run twice on the same source clip and the same user request: once on the raw user prompt (their default deployment,



Figure 10: Qualitative comparison on EditVerse-Bench [16].

Table 7: Per-category quantitative comparison on OpenVE-Bench [11], judged by Gemini 2.5 Pro [6] on a 1–5 scale. † marks zero-shot generalization: Aurora is not trained on subtitle removal data. Overall scores are summarized in Table 3.

Method	#Params.	Global Style	Bkg. Change	Local Change	Local Remove	Local Add	Subtitle	Overall
Runway Aleph [28]	–	3.72	2.62	4.18	4.16	2.78	3.62	3.51
VACE [15]	14B	1.49	1.55	2.07	1.46	1.26	1.48	1.55
InsViE [38]	2B	2.20	1.06	1.48	1.36	1.17	2.18	1.58
Lucy-Edit [31]	5B	2.27	1.57	3.20	1.75	2.30	1.61	2.12
Ditto [1]	14B	4.01	1.68	2.03	1.53	1.41	2.81	2.25
OpenVE-Edit [11]	5B	3.16	2.36	2.98	1.85	2.15	2.91	2.57
Kiwi-Edit [19]	5B	3.64	2.64	3.83	2.63	2.36	–	–
UniVideo [36]	13B	3.75	2.38	3.78	2.93	**2.87**	2.91	3.10
Aurora (Ours)	5B	**4.07**	**2.67**	**4.03**	**3.73**	2.78	**2.97**†	**3.38**

labeled *w/o agent*), and once on the VLM agent's rewritten prompt together with the VLM agent's retrieved reference image (labeled *w/ agent*). In each case, the third cell shows the first source frame with the retrieved reference image overlaid as a small inset. Across these examples, the *w/o agent* runs usually drop the named entity or replace it with a generic substitute, whereas the *w/ agent* runs follow the rewritten instruction and use the retrieved reference image to better preserve the identity of the requested real-world entity.

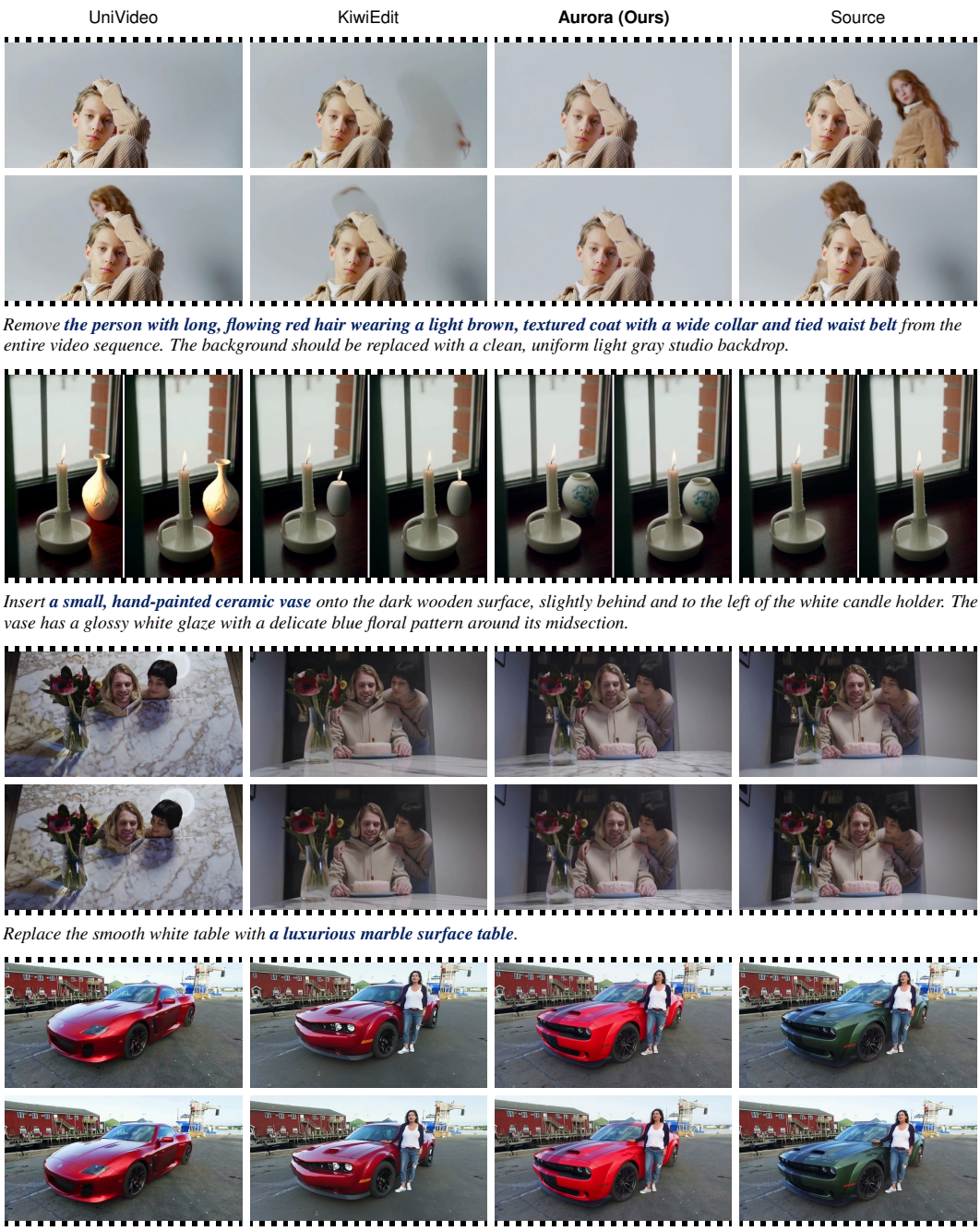


Figure 11: Qualitative comparison on OpenVE-Bench [11].

E Experimental details

Tables 8–9 summarize the Aurora video model and training recipe used for the reported results. Table 10 summarizes the per-benchmark inference settings and the VLM agent tool toggles for each evaluation.

Output frames per benchmark. The video DiT always denoises 81 frames internally, matching Wan's $4k+1$ temporal constraint, but the number of frames written to disk differs across benchmarks so that each saved clip matches the protocol of its evaluator. On AgentEdit-Bench, the benchmark

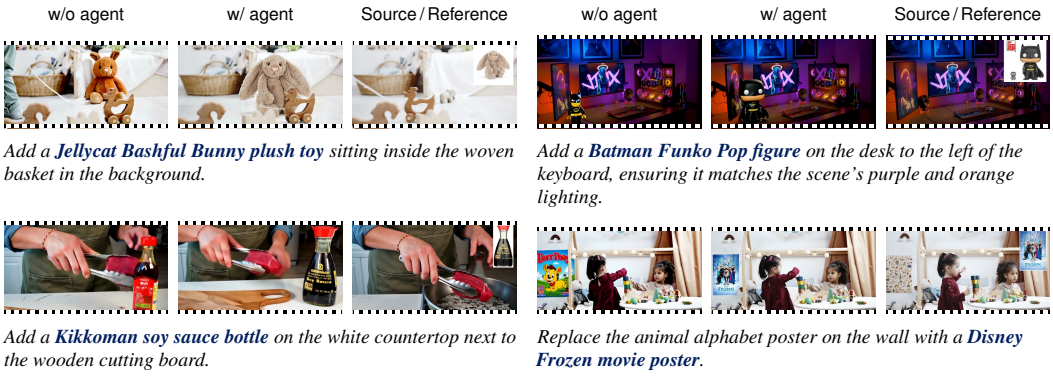


Figure 12: Qualitative VLM agent transfer on UniVideo [36].



Figure 13: Qualitative VLM agent transfer on Kiwi-Edit [19].

Table 8: Aurora video DiT architecture hyperparameters.

Component	Item	Setting
Video DiT	Backbone	Wan2.2-TI2V-5B with patch size $(1, 2, 2)$.
VAE	Model	Wan2.2 VAE, frozen, with $16\times$ spatial downsampling.
MLLM encoder	Model	Frozen Qwen3.5-4B in bf16.
Context projector	Architecture	Zero-initialized linear layer.
Source conditioning	Latent path	VAE-encoded source video tokens are concatenated with noisy video tokens along the token-sequence dimension.
Source conditioning	MLLM path	Images are resized with maximum pixels 147456 before feeding into Qwen. Frames are sampled at 1 fps with at least 2 frames.

consumes the full 81-frame output, so we save all 81 frames. On EditVerse-Bench, the official evaluator expects 64-frame clips, so we keep the first 64 frames of the 81-frame output and discard the rest. On OpenVE-Bench, source clips have variable frame counts, so we temporal-resize each source to 81 frames before denoising and then temporal-resize the 81-frame output back to the source's original frame count when saving, so that the saved edited clip has the same length as the input.

Inference-time guidance. At inference time, we evaluate three predictions from the same velocity field v_θ trained in Sec. 3.3: v_+ with full text and visual conditions, v_{vneg} with empty or negative text and visual conditions, and $v_\emptyset$ with no text and no visual conditions. The guided velocity is

$$\hat{v} = v_\emptyset + \lambda_{\text{img}}(v_{\text{vneg}} - v_\emptyset) + \lambda_{\text{txt}}(v_+ - v_{\text{vneg}}). \tag{10}$$

When $\lambda_{\text{img}} = 1$, Eq. (10) algebraically reduces to the standard two-pass text CFG path. The unconditional and visual-negative branches that this formula evaluates are exactly the two dropout branches trained by the prompt dropout and visual dropout in Sec. 3.3.

Table 9: Aurora video DiT training parameters. Stage 1 is the low-resolution warmup stage; Stage 2 resumes from the Stage 1 checkpoint stream at the higher pixel budget.

Parameter	Stage 1	Stage 2	Notes
Frames	81	81	Wan's 4k+1 temporal constraint.
Video pixel budget	399,360	921,600	Stage 1 is about 448×832; Stage 2 is about 708×1280
Reference VAE budget	921,600	921,600	Per reference image and source video frame.
MLLM source budget	147,456	147,456	Per sampled source video frame.
MLLM reference budget	147,456	147,456	Per reference image.
MLLM video sampling	1 fps, min 2 frames	1 fps, min 2 frames	Qwen video-processor override.
Optimizer	AdamW	AdamW	-
Learning rate and schedule	1×10^{-5}, constant	1×10^{-5}, constant	-
Weight decay and grad clip	0.01 / 1.0	0.01 / 1.0	-
Precision and distributed recipe	bf16, ZeRO-2	bf16, ZeRO-2	-
Effective global batch	128	128	-
Source conditioning mode	token-sequence concat	token-sequence concat	-
Prompt dropout	0.1	0.1	Trains the empty-text branch.
Visual dropout given prompt drop	0.5	0.5	Produces the visual-negative and unconditional subsets.

Table 10: Aurora per-benchmark inference settings. `cfg_scale` and `image_cfg_scale` correspond to λ_{txt} and λ_{img} in Eq. (10); when $\lambda_{\text{img}} = 1.0$ the pipeline falls back to the two-pass CFG path defined in Appendix E. The agent VLM tools (web image search and mask overlay) are disabled on the two already-specified benchmarks, so the agent only rewrites the instruction there.

Setting	AgentEdit-Bench	EditVerse-Bench	OpenVE-Bench
Checkpoint	Stage 2	Stage 2	Stage 2
Cases	150 (5 edit types)	120 (12 edit types)	358 (6 edit types)
`cfg_scale` (λ_{txt})	2.0	1.5	2.0
`image_cfg_scale` (λ_{img})	1.25	1.0	1.0
Two-pass CFG fallback when λ_{img}=1	disabled	enabled	enabled
Denoising steps	50	50	50
Output frames	81 (save 81)	81 (save 64)	81, temporal-resize
Max video pixels	921,600	921,600	921,600
Negative prompt	empty	empty	empty
VLM agent base	Qwen3-VL-8B-Instruct [24] + Aurora LoRA adapter		
Agent rewrite (y') used	yes	yes	yes
Agent web image search	enabled	disabled	disabled
Agent mask overlay	enabled	disabled	disabled
Judge model	Gemini 2.5 Pro [6] at `temperature`=0		

F Additional analyses of the VLM agent

F.1 Human evaluation of the judge

We sample 60 cases at random across AgentEdit-Bench, EditVerse-Bench, and OpenVE-Bench. Fifteen human annotators rate the generated videos of each case, and we compare the aggregated human ratings with the scores of each benchmark's official evaluation protocol. Table 11 reports two forms of agreement. Pooled over all cases, human and automatic scores correlate only moderately, with a Pearson correlation of 0.534. Within a case, the human and automatic rankings of the competing outputs agree more closely, with mean Pearson, Spearman, and Kendall correlations of 0.692, 0.645, and 0.545; this per-case agreement is the one most relevant to the method comparisons in our tables.

Table 11: **Agreement between human ratings and the benchmark judges** on 60 cases sampled from the three benchmarks. Pooled scores correlate all human and automatic scores; per-case ranking correlates the two rankings of the competing outputs within each case and averages over cases.

Agreement	Pearson	Spearman	Kendall
Pooled scores	0.534	0.470	0.352
Per-case ranking (mean)	0.692	0.645	0.545

Table 12: **Training-free alternatives to the Aurora VLM agent** on the OpenVE-Bench subset (†) of Table 5(b). All rows use our video DiT and the Gemini 2.5 Pro judge; Δ is relative to the raw prompt.

Condition	VLM	Overall	Δ
Raw user instruction	–	3.31	–
Zero-shot planning, all tools enabled	Qwen3-VL-8B-Instruct	3.16	-0.15
Zero-shot planning, all tools enabled	Qwen3.5-9B	3.18	-0.13
Zero-shot prompt rewriting	Qwen3-VL-8B-Instruct	3.19	-0.12
Zero-shot prompt rewriting	Qwen3.5-9B	3.09	-0.22
Aurora VLM agent, rewriting only	Qwen3-VL-8B-Instruct + LoRA	3.33	$+0.02$
Aurora VLM agent	Qwen3-VL-8B-Instruct + LoRA	**3.46**	$+0.15$

Table 13: **System-level statistics of the Aurora VLM agent** on AgentEdit-Bench. Planning latency is measured on one NVIDIA H100 80GB GPU, and mask IoU against human annotations.

Metric	Result
Planning latency per case, mean / median	7.13 s / 4.01 s
Latency per tool call	typically < 0.5 s
Image-search cases	108
Queries returning no candidates	0/108 (0.0%)
Reference selected from the top-1 result	81/108 (75.0%)
Reference selected within the top-3 results	104/108 (96.3%)
Mean selected rank	1.40
Mask-generation cases	26
Cases returning no mask	0/26 (0.0%)
Mask IoU, mean / median	0.778 / 0.960
Cases with IoU < 0.5	4/26 (15.4%)

Because every reported benchmark score still comes from one VLM judge, we recommend that future evaluations combine several judges to reduce judge-specific bias.

F.2 Training-free VLM agents

Table 12 lists the training-free alternatives discussed in Sec. 5.4. The zero-shot planners follow a rule-based system prompt with all tools enabled, whereas the zero-shot rewriting pipelines only rewrite the instruction. All rows share the Aurora video DiT and the Gemini 2.5 Pro judge.

F.3 Component ablation

Each row in the lower block of Table 4 removes one part of the full VLM agent and keeps the video DiT unchanged. "w/o search & selection" disables image retrieval and reference selection, "w/o selection" replaces learned reference selection with a fixed retrieval strategy, "w/o rewriting" disables instruction rewriting, and "w/o mask grounding" disables the segmentation tool. Every setting runs on the full AgentEdit-Bench and is scored by Gemini 2.5 Pro.

F.4 Tool reliability, latency, and failure modes

Table 13 measures planning latency over the full AgentEdit-Bench; image search and mask generation run only when the plan requests them. Sec. 5.4 discusses these statistics together with the four stages

Table 14: **Plan-level preference versus final video quality** on 200 prompts. Per prompt, the plan-level judge of our DPO data procedure prefers one of two plans, and the same video DiT renders both. Top: mean video score on the 1–5 scale. Middle: paired outcomes. Bottom: videos per score.

Plan used for generation	n	Mean video score
Preferred plan	200	3.402
Rejected plan	200	3.172
Δ (preferred $-$ rejected)	–	$+0.230$

Preferred-plan wins	Ties	Preferred-plan losses	Total
76 (38.0%)	57 (28.5%)	67 (33.5%)	200

Video score	1.00	1.33	1.67	2.00	2.33	2.67	3.00	3.33	3.67	4.00	4.33	4.67	5.00
Preferred plan	6	2	5	32	11	19	20	15	12	12	13	1	52
Rejected plan	10	3	7	36	16	18	29	14	5	7	4	3	48

at which errors can enter the pipeline and how they propagate to the final video. The agent only predicts the mask phrase, and the segmentation backend returns a masked image in a fixed format, so a stronger backend can replace it without changing the agent or the video DiT.

F.5 Plan-level preference and final video quality

For each of 200 sampled prompts, we generate two plans with the procedure used for our DPO data (Sec. 4): one preferred and one rejected by the plan-level judge. The same video DiT renders both plans, and each video is scored on the 1–5 scale. Videos from preferred plans average 3.402, against 3.172 for rejected plans. Per pair, however, the preferred plan yields the better video in 76 cases, ties in 57, and yields the worse video in 67 (Table 14). Plan preference is therefore an imperfect proxy for final video quality, which motivates online RL with video-level rewards (Sec. 6).

G Additional VLM agent data details

G.1 Data-construction details

Video editing model data. The mixture of training data in Table 1 is built so that instructions, source videos, and reference images all appear under the same conditioning format. Large open-source video-edit sources such as ReCo [46], Ditto [1], EgoEdit [17], and OpenVE [11] are judged by Gemini Flash-Lite [10] on source quality, target quality, edit localization, identity preservation, motion consistency, edit authenticity, and prompt alignment. Accepted samples keep the original instruction when it is faithful; otherwise the judge-provided rewrite is used. The "Ditto combined-task data" row of Table 1 is mined from same-source pairs in Ditto [1]. For each source video S with two edited variants T_i and T_j produced by instructions A and B, the model is trained on the pair whose input is T_i and whose target is T_j, so that S is never seen and the new instruction must describe the change from T_i to T_j using only what is currently visible. Three rule-based filters first discard candidates in which A and B are both style transfers, both global scene changes, or both manipulate the same object, since those collapse into a single edit. Gemini Flash-Lite [10] then inspects the three frames and accepts a candidate only when the two edits are visually distinct and the change from input to target can be written as one short instruction describable from the input alone. The final subset contains 64,782 pairs.

Planning SFT and reference-image selection. Planning SFT starts from accepted editing data with a clean instruction. We do not ask the teacher to write an explanation; instead, we ask what a real user would most likely omit and how the missing information should be restored into the four-field format. The clean instruction is degraded into a shorter, more ambiguous, more colloquial, or identity-hiding raw request, while the supervision target remains the plan defined in Eq. (1). Sampling is organized by task control rather than raw frequency. Reference-image selection uses another 10K supervised examples adapted from [7]. Each example contains a retrieved entity, web context, and several

candidate images; the target is a discrete candidate index such as `Image 3`, so the VLM agent learns which image should populate the reference slot after retrieval.

Figure 14 shows the additional planning SFT examples. Each cell pairs two source video frames with the colloquial user request and the four-field edit plan that the VLM agent is supervised to produce. The fifteen cells together illustrate when each plan field carries information versus when it remains `false`: an identity-bearing target or a named real-world entity triggers `image_search`; a localized removal triggers `mask`; a global style change and a combined edit leave both tool fields empty.

Figure 15 shows the additional reference-image selection SFT subset summarized in Section 4. Once the planner has emitted a non-empty `image_search` query, web image search returns multiple candidates. The VLM agent's task here is simple: pick the single image that is visually grounded and factually consistent with the named-entity constraints in the search prompt.

Preference alignment. Preference data targets cases that SFT does not isolate well. Each chosen/rejected pair keeps the source, raw request, and most plan fields fixed while varying one decision axis. The 1.8K-pair set covers the same five plan-level boundary cases defined in Sec. 3.3: source-entity false triggers, ambiguous mask phrases, false image-search triggers, constraint-losing rewrites, and task-routing confusions.

Figures 16 and 17 show the preference alignment data. Each panel keeps the user prompt and shared plan fields fixed and varies one preference axis at a time. The chosen and rejected plan JSONs are shown in full so the reader can see exactly which field flips between the two; that field is marked with a green bar and bold green text in the chosen panel and a red bar and bold red text in the rejected panel.

H AgentEdit-Bench

AgentEdit-Bench is a 150-case video editing benchmark over five edit types. All source videos of benchmark are drawn from the Pexels stock library. Figure 18 shows nine cases covering all five edit types, with the named target or counterfactual cue and the edit region or physical effect highlighted in the instruction. Figure 19 reports the instruction word cloud and four count distributions: cases per edit type, instruction length, reasoning sub-style, and the physical effects that removal targets carry.

H.1 Scoring rubric and judge prompt

Each case is scored on the seven axes in Table 15; IP edits use all seven, and `reasoning` and `removal` use only axes 1–5. The removal subset adds an anti-replacement clause to axis 1: an output that substitutes a new object for the target is capped at 1, which prevents hallucinated replacements from scoring as clean inpainting. Separating *IP Presence* from *IP Identity Match* distinguishes an edit that places *something* in the right region from one that misses the region entirely. Per-edit-type numbers use the native axis count; the cross-type aggregate normalizes every case to a 0–21 scale and we report it as a percentage.

The judge runs with `temperature=0`, and scores each case on three frames sampled at indices 0, $n/3$, and $2n/3$. Per-axis scores are averaged across the three frames and the raw judge text is persisted next to the parsed scores. Figure 20 shows the verbatim prompts.

Table 15: AgentEdit-Bench scoring rubric. IP edits use all seven axes (max 21); reasoning and removal use axes 1–5 only (max 15).

#	Axis	What it measures
1	Instruction Following	Whether the requested edit happened in the named region with the named target. The removal subset adds an anti-replacement clause: an output that replaces the target with a new object is capped at 1.
2	Edit Region Localization	Whether the change stays inside the specified region.
3	Source Preservation	Whether unedited regions, motion, geometry, and lighting are preserved.
4	Visual Quality	Realism, lighting and shadow match, scale and perspective, absence of paste-on artifacts.
5	Temporal Consistency	Whether the edited entity would remain stable across the clip.
6	IP Presence	Whether the named entity is visible and recognizable in roughly the right region.
7	IP Identity Match	Whether the visible entity matches the specific real-world identity (brand colors, logo, signature shape).



SOURCE VIDEO FRAMES

User prompt: "hey, i was just looking at this and wondering if you could maybe take that lifebuoy thing off the wall so it looks plain?"

Output plan JSON: {
  refined_text_instruction: "Remove the red and white striped lifebuoy from the center of the tiled wall. Replace the space it occupied with the continuous pattern of the light gray square tiles…"
  subtask: remove_object
  image_search: false
  mask: "red and white striped lifebuoy from the center of the tiled wall"
}



SOURCE VIDEO FRAMES

User prompt: "hey could you maybe change the background to 10 downing street and make it look like a still daylight shot instead of the stormy cliff?"

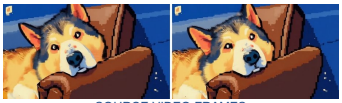
Output plan JSON: {
  refined_text_instruction: "Replace the dynamic stormy coastal cliff with a static outdoor scene of 10 Downing Street under consistent daylight with no background motion."
  subtask: change_background
  image_search: "10 Downing Street"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "swap duck joker"

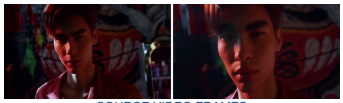
Output plan JSON: {
  refined_text_instruction: "Replace the yellow duck mascot on the right side of the frame with a Joker in a white lab coat, plaid shirt, and orange vest."
  subtask: replace_object
  image_search: "Joker"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "can u make this look real and not like pixel art anymore thx"

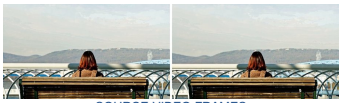
Output plan JSON: {
  refined_text_instruction: "Convert this pixel art video back to a live-action, photorealistic format by systematically removing all pixel art characteristics. Restore real-world textures, natural lighting…"
  subtask: global_style
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "make music video add flares change colors"

Output plan JSON: {
  refined_text_instruction: "Transform the entire video into a cinematic music video aesthetic with slow-motion effects, lens flares, and a color grade that blends deep reds and electric blues into a moody…"
  subtask: combined_tasks
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "Change the hair to long and light colored with that accessory on the end."

Output plan JSON: {
  refined_text_instruction: "Change the shoulder-length brown hair into long, silvery-white hair with a single red ribbon tied at the end."
  subtask: change_color
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "hey i was just thinking it would be super cool if you could maybe make those toy blocks look like thomas the tank engine."

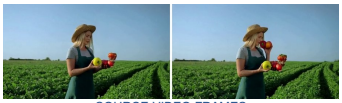
Output plan JSON: {
  refined_text_instruction: "Transform the colorful toy blocks into a Thomas the Tank Engine toy."
  subtask: replace_object
  image_search: "Thomas the Tank Engine toy"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "Put a kid in that shirt on the bench between the two people."

Output plan JSON: {
  refined_text_instruction: "Add a young boy with dark hair, wearing a white t-shirt with two black stripes on the sleeves, looking down, sitting on the bench between the adult and the young girl."
  subtask: add_object
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "put dwight schrute in a blue shirt and green apron holding red stuff standin next to the lady lookin down pls"

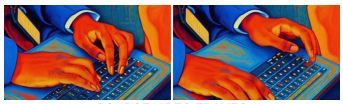
Output plan JSON: {
  refined_text_instruction: "Add Dwight Schrute wearing a blue shirt and a green apron, holding red produce, looking down, standing to the right of the woman in the field."
  subtask: add_object
  image_search: "Dwight Schrute"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "swap bowl for nutella jar"

Output plan JSON: {
  refined_text_instruction: "Replace the white bowl filled with hazelnuts held by hands in the center of the frame with a jar of Nutella."
  subtask: replace_object
  image_search: "jar of Nutella"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "make photorealistic"

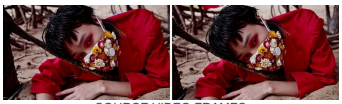
Output plan JSON: {
  refined_text_instruction: "Convert this Abstract Art-animated video to a live-action, photorealistic format. Meticulously strip away the abstract shapes, stylized color fields, and gestural animations…"
  subtask: global_style
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "swich the diner background to monica gellers kitchen from friends w the purple walls and teal cabinets pls thx"

Output plan JSON: {
  refined_text_instruction: "Replace the dynamic 1950s diner background with Monica Geller's kitchen from the TV show Friends, making sure to include its iconic purple walls and teal cabinets."
  subtask: change_background
  image_search: "Monica Geller's kitchen from Friends"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "change mask pattern"

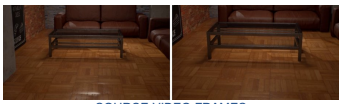
Output plan JSON: {
  refined_text_instruction: "Transform the floral face mask into a geometric patterned mask in gold and black."
  subtask: change_color
  image_search: false
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "put darth vader on the right chair thx"

Output plan JSON: {
  refined_text_instruction: "Add Darth Vader sitting on the rightmost chair."
  subtask: add_object
  image_search: "Darth Vader"
  mask: false
}



SOURCE VIDEO FRAMES

User prompt: "hey, the room feels a bit cramped so could you maybe take out that table and just fill in the floor so it looks empty?"

Output plan JSON: {
  refined_text_instruction: "Remove the dark brown rectangular wooden table frame located in the center of the room. Replace it by continuing the pattern of the polished parquet flooring…"
  subtask: remove_object
  image_search: false
  mask: "dark brown rectangular wooden table frame located in the center of the room"
}

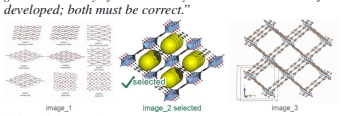
Figure 14: Planning SFT examples.

Search prompt: "A macro photo of the orchid mantis (Hymenopus coronatus) on a flower. A nearby informational card lists the scientific family name and the specific country where the holotype was collected; both must be correct."



Selected: image_2.

Search prompt: "A detailed 3D visualization of the metal-organic framework MIL-53. A data sidebar on the screen lists its specific BET surface area in square meters per gram and the city of the university in France where it was first developed; both must be correct."



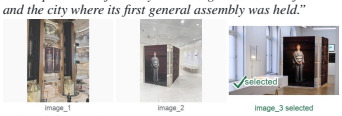
Selected: image_2.

Search prompt: "A macro photograph of a 'Chuvash Khushpu' headpiece resting on a pedestal. A small archival tag identifies the village where this specific piece was acquired and lists the precise number of silver coins sewn onto it."



Selected: image_1.

Search prompt: "A photo of an exhibit at the Nobel Peace Center focusing on the organization 'Nihon Hidankyo'. A wall caption identifies the year the organization was founded and the city where its first general assembly was held."



Selected: image_3.

Search prompt: "A cinematic shot of the 'Dungeon Meshi' character 'Izutsumi' in her transformed beast-girl form. A side panel lists her archetype and her listed height; both must be correct."



Selected: image_3.

Search prompt: "A detailed shot of the 'Jingdezhen Imperial Kiln Museum' in China, showcasing its brick-vaulted halls. A signage panel lists the architect of record and the year construction was completed."

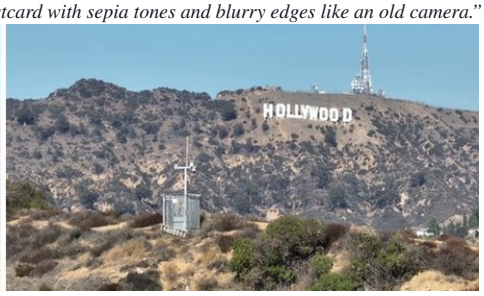


Selected: image_2.

Figure 15: Reference-image selection data examples.

User prompt: "make that place look like an old vintage postcard with sepia tones and blurry edges like an old camera."

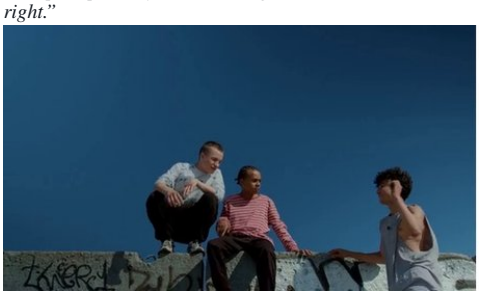




✓ Chosen
`refined_text_instruction`: "Transform the video into a vintage Hollywood postcard style with soft sepia tones, film grain, and a slightly blurred edge, as if seen through an old camera lens."
`subtask`: **global_style**
`image_search`: false
`mask`: false

x Rejected
`refined_text_instruction`: "Transform the video into a vintage Hollywood postcard style with soft sepia tones, film grain, and a slightly blurred edge, as if seen through an old camera lens."
`subtask`: **combined_tasks**
`image_search`: false
`mask`: false

User prompt: "hey i was looking at this and i think it would be cool if you could maybe take out the guy sitting on the right."

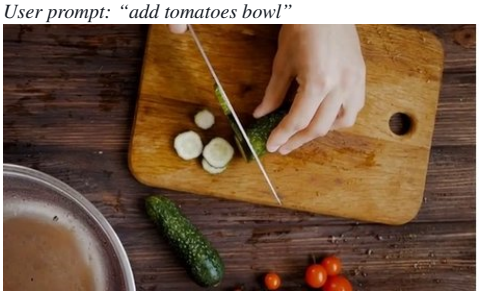




✓ Chosen
`refined_text_instruction`: "Remove the person on the right."
`subtask`: remove_object
`image_search`: false
`mask`: **"person on the right"**

x Rejected
`refined_text_instruction`: "Remove the person on the right."
`subtask`: remove_object
`image_search`: false
`mask`: **"person"**

User prompt: "add tomatoes bowl"





✓ Chosen
`refined_text_instruction`: "Add small red cherry tomatoes into the clear bowl."
`subtask`: add_object
`image_search`: **false**
`mask`: false

x Rejected
`refined_text_instruction`: "Add small red cherry tomatoes into the clear bowl."
`subtask`: add_object
`image_search`: **"NatureSweet Cherubs tomatoes"**
`mask`: false

Figure 16: DPO preference examples (1/2).

User prompt: "change that guy in the center to an old man wearing different clothes."



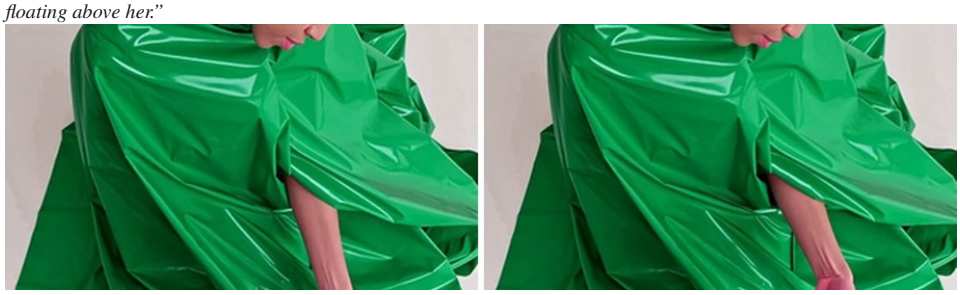
✓ Chosen

`refined_text_instruction`: "Replace the man wearing a blue baseball cap and a grey sweater in the center of the frame with an elderly man with a long white beard, a black hat, and a black robe."
`subtask`: replace_object
`image_search`: **false**
`mask`: false

x Rejected

`refined_text_instruction`: "Replace the man wearing a blue baseball cap and a grey sweater in the center of the frame with an elderly man with a long white beard, a black hat, and a black robe."
`subtask`: replace_object
`image_search`: **"the man wearing a blue baseball cap and a grey sweater"**
`mask`: false

User prompt: "hey i was just thinking it would be cool if you could maybe make that dress pink and put a red flower floating above her."



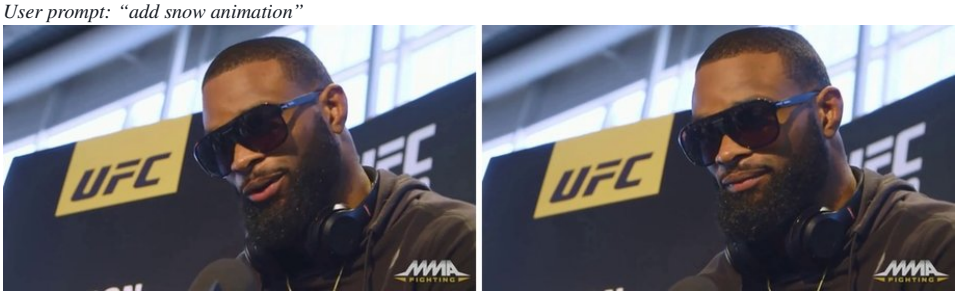
✓ Chosen

`refined_text_instruction`: **"Change the green garment to pink and add a single floating red rose near the top of the scene."**
`subtask`: combined_tasks
`image_search`: false
`mask`: false

x Rejected

`refined_text_instruction`: **"Change the green outfit to pink and add a rose near the top."**
`subtask`: combined_tasks
`image_search`: false
`mask`: false

User prompt: "add snow animation"



✓ Chosen

`refined_text_instruction`: **"Apply a dynamic, snowy animation style to this video, ensuring smooth, frame-by-frame consistency. The final output should feature realistic snowfall, icy textures, and a wintery atmosphere…"**
`subtask`: global_style
`image_search`: false
`mask`: false

x Rejected

`refined_text_instruction`: **"Apply a snowy animation style."**
`subtask`: global_style
`image_search`: false
`mask`: false

Figure 17: DPO preference examples (2/2).



`ip_object_replace`

Replace the *glass bowl of salad on the left of the counter* with a **red Le Creuset Dutch oven**.



`ip_object_replace`

Replace the *white armchair in the background* with a **classic Eames Lounge Chair**.



`ip_object_add`

Add a bottle of **Hendrick's Gin** on the *white counter to the left of the tray*.



`ip_object_add`

Add a **red KitchenAid stand mixer** on the *counter in the background*.



`ip_object_add`

Add a **Goodyear blimp** flying in the *empty blue sky to the left of the kite*.



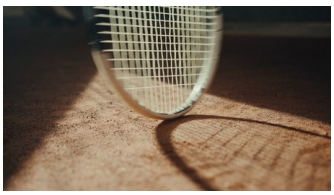
`ip_background_change`

Change the *view through the window* to a **New York City skyline at sunset**, keeping the interior unchanged.



`reasoning`

Turn the lake into a **perfect mirror for the passing clouds** above.



`removal`

Remove the **white tennis racket** on the court, including the *intricate shadow cast by its frame and strings* on the clay surface.



`removal`

Remove the **ballerina in the middle of the room** and reconstruct the *bright window, wall, and barre behind her*.

Figure 18: AgentEdit-Bench examples. Each case is one source clip plus one user request; we highlight the named target or counterfactual cue (**bold blue**) and the edit region or physical effect (*italic green*).



(a) Instruction word cloud, generic edit verbs and stop words removed.

(b) Cases per edit type, instruction length, reasoning sub-style, and removal physical effects.

Figure 19: AgentEdit-Bench distribution. The vocabulary in (a) is dominated by scene and entity nouns, and the four count plots in (b) show the case mix and the substructure inside the reasoning and removal subsets.

Judge prompt: 7-axis IP rubric (IP edit types)

```
You are a meticulous video editing quality evaluator.  Compare the BEFORE frame and the
AFTER frame against the editing instruction and a named target entity, and score the edit
on seven axes.

Editing instruction:  {prompt}
Target entity:  {target_entity}
Edit region:  {edit_region}

For each axis, return an integer from 0 (worst) to 3 (best) with a brief justification.
Do not exceed 3.

1.  Instruction Following (0-3).  Was the requested edit performed in the named region
with the named target?
  3:  requested edit performed correctly in the right region.
  2:  edit performed but with a minor inaccuracy or omission.
  1:  edit partially performed or applied to the wrong region.
  0:  instruction ignored, or the opposite was done.
2.  Edit Region Localization (0-3).  Was the change confined to the specified region?
3.  Source Preservation (0-3).  Are subject motion, geometry and lighting outside the edit
region preserved?
4.  Visual Quality (0-3).  Realism, seamless integration, lighting and shadow match, scale
and perspective.
5.  Temporal Consistency (0-3).  Inferred from the AFTER frame, would the edited entity
remain stable across the clip without ghosting or pop-in?
6.  IP Presence (0-3).  Is the named entity actually visible and recognizable in roughly
the right region?
7.  IP Identity Match (0-3).  Does the visible entity match the specific real-world
identity, brand colors, logo, and signature shape?  You may rely only on your internal
world knowledge of the brand or product.

Return your evaluation in exactly this format:
Instruction Following:  [score] - [one-sentence justification]
Edit Region Localization:  [score] - [one-sentence justification]
Source Preservation:  [score] - [one-sentence justification]
Visual Quality:  [score] - [one-sentence justification]
Temporal Consistency:  [score] - [one-sentence justification]
IP Presence:  [score] - [one-sentence justification]
IP Identity Match:  [score] - [one-sentence justification]
Total:  [sum of the seven scores]
```

Judge prompt: 5-axis non-IP rubric (reasoning, removal)

```
You are a meticulous video editing quality evaluator.  Compare the BEFORE frame and the
AFTER frame against the editing instruction, and score the edit on five axes.
Editing instruction:  {prompt}
For each axis, return an integer from 0 (worst) to 3 (best) with a brief justification.
Do not exceed 3.
1.  Instruction Following (0-3).  Was the requested edit performed?
  3:  edit performed correctly.
  2:  edit performed with a minor inaccuracy.
  1:  edit partially performed or applied to the wrong region.
  0:  instruction ignored, or the opposite was done.
  Removal-only clause:  if the model replaced the target with a new object instead of
removing it, give at most 1.
2.  Edit Region Localization (0-3).  Was the change confined to the specified region or,
for global edits, to the implied scope?
3.  Source Preservation (0-3).  Are subject motion, geometry and lighting outside the edit
region preserved?
4.  Visual Quality (0-3).  Realism, seamless integration, lighting and shadow match, scale
and perspective.
5.  Temporal Consistency (0-3).  Inferred from the AFTER frame, would the result remain
stable across the clip?
Return your evaluation in exactly this format:
Instruction Following:  [score] - [one-sentence justification]
Edit Region Localization:  [score] - [one-sentence justification]
Source Preservation:  [score] - [one-sentence justification]
Visual Quality:  [score] - [one-sentence justification]
Temporal Consistency:  [score] - [one-sentence justification]
Total:  [sum of the five scores]
```

Figure 20: VLM-as-judge prompts. The two variants share the body and differ in axis count and the removal-only anti-replacement clause.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state the three contributions, namely the agentic framework, the VLM agent training data, and AgentEdit-Bench; Sections 3–5 support them with method details and experiments, and Section 6 states their scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 6 discusses the offline, plan-level preference alignment of the VLM agent and its imperfect agreement with final video quality, tool-error propagation, the single-edit scope of the four-field plan, the reliance on a single VLM judge, and the difficulty of inserting new subjects with large motion using a 5B video DiT.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper contains no theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 3–5 describe the architecture, training objectives, data construction, and evaluation protocol; Appendix E lists the training and inference settings, Appendix G the VLM agent data, and Appendix H the benchmark rubric and judge prompts.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, model weights, and the self-curated training subsets are released at https://github.com/yeates/Aurora with instructions for installation, inference, training, and evaluation; the AgentEdit-Bench evaluation code is included there, and the benchmark data will be released in the same repository.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 5 describes the benchmarks, evaluation subsets, and judge, and Appendix E lists the optimizer, learning rate, batch size, resolution schedule, dropout rates, and per-benchmark inference settings, including guidance scales and denoising steps.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: The reported numbers come from single evaluation runs without error bars, because each run generates every benchmark clip with a 5B video DiT at 81 frames, which makes repeated runs expensive; the VLM judge runs at temperature 0.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Appendix E reports the training configuration, including precision, distributed recipe, batch size, learning rate, and resolution schedule, but the paper does not state the GPU type, GPU count, or training time, so the total compute cannot be derived from it.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We reviewed the NeurIPS Code of Ethics and the work conforms to it; the only human participation is a small rating study of the VLM judge (Appendix F.1), in which annotators rate generated videos, and no sensitive personal data are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The broader-impacts paragraph in Section 6 discusses misuse for non-consensual identity manipulation and the copyright and likeness risks of retrieved references, and positions Aurora as a creative editing tool rather than an evidence-producing one.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [No]

Justification: The code and model weights are released openly under the MIT license without access gating; the paper states the misuse risks in Section 6, and the AgentEdit-Bench media come from the Pexels stock library under the Pexels license.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All models, datasets, and benchmarks we use, including Wan2.2-TI2V-5B, Qwen3-VL-8B-Instruct, Qwen3.5-4B, Qwen3.5-9B, GroundingDINO, SAM 2, the training

sources in Table 1, EditVerse-Bench, and OpenVE-Bench, are cited at first use and used under their original licenses; the code release lists every training source and points to its license terms, and the AgentEdit-Bench media follow the Pexels license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The released code, weights, and training subsets are documented in the repository README, which covers installation, inference, training, evaluation, and data attribution; AgentEdit-Bench is documented in Section 5 and Appendix H, and its data will be released in the same repository with the same documentation.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [No]

Justification: Appendix F.1 reports a human rating study of the VLM judge with 15 annotators on 60 cases and its agreement analysis, but the paper does not include the annotator instructions, screenshots, or compensation details.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: The paper does not describe participant risks, whether they were disclosed, or IRB review for the human rating study in Appendix F.1.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are core components: a LoRA-finetuned Qwen3-VL-8B-Instruct is the VLM agent, a frozen Qwen3.5-4B is the multimodal context encoder (Section 3), Gemini Flash-Lite filters and relabels data (Section 4), Gemini 2.5 Pro is the judge (Section 5), and zero-shot Qwen3-VL-8B-Instruct and Qwen3.5-9B are training-free baselines (Section 5.4).

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.