VIDEOFORGE: TOWARD RECURSIVE SELF-IMPROVEMENT ENVIRONMENTS FOR AGENTIC VIDEO EVALUATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Improving video evaluation agents requires identifying why they miss or misdescribe defects and determining which harness changes correct these failures. Existing methods typically optimize the evaluation harness with a fixed outer-loop procedure, leaving the improvement process itself unchanged. We introduce **VideoForge**, a recursive self-improvement environment that adapts both the video-evaluation harness and the procedure used to improve it. VideoForge exposes task execution, evidence inspection, harness revision, and self-reflection as interaction primitives, allowing improvement trajectories to adapt to observed failures and accumulated experience. To evaluate these capabilities, we introduce **VidDiag**, a benchmark comprising 1,087 real-world video generation and editing cases with 2,932 human-annotated defects spanning seven dimensions and 27 fine-grained failure types. The benchmark evaluates both defect-category identification and instance-level defect descriptions. Across two model–agent configurations, VideoForge improves overall category F1 by 3.6%–4.9% and strict semantic matching by 2.9%–4.6% over the corresponding initial harnesses, achieving the highest overall scores among the compared optimization methods, including SkillOpt and AHE. Ablations under the same task-execution budget further support the benefit of adapting the improvement procedure. We further conduct a separate human-evaluated video-debugging study on 100 cases to examine whether accurate diagnosis can support downstream repair. Providing ground-truth defect descriptions improves repair success by 11%–12%, yet the highest success rate remains only 32%. These results show that adapting the improvement process can strengthen fine-grained video diagnosis, while translating accurate diagnosis into effective repair remains challenging.

1 INTRODUCTION

Recent video generation models have demonstrated strong capabilities in producing high-quality videos (Wan et al., 2025; MiniMax, 2026; Seedance et al., 2026; Collins, 2025; HaCohen et al., 2026), supporting video generation from detailed instructions and complex multimodal references. However, these models still exhibit limitations in reference consistency and instruction following (Zhang et al., 2026c), controllable video editing (Wu et al., 2026), physical plausibility (Lin et al., 2026c), and audio-visual alignment (Zhou et al., 2026). To address these issues, recent work has explored prompt optimization (Long et al., 2026; Cheng et al., 2025; Song et al., 2026), test-time search (Liu et al., 2025), and localized video repair (Lee et al., 2026a).

Effective refinement relies on fine-grained video evaluation to identify specific errors. Recent work has explored using agents for video question answering (Wang et al., 2024; Zhang et al., 2025; Lin et al., 2026b) and generated-video evaluation (Song et al., 2026; Wei et al., 2026a). To improve long-video understanding, recent work has explored agent self-improvement through tool development and refinement (Ye et al., 2026), context-construction optimization (Xu & Chen, 2026), and revisions to agent modules based on execution feedback (Cui et al., 2026). However, existing methods improve video-agent harnesses while keeping the optimization loop fixed. They follow a predefined cycle of proposing harness revisions, evaluating candidates, and selecting updates. For fine-grained video evaluation, useful feedback should reveal missed errors, incorrect findings, and inaccuracies

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

in the agent's descriptions. These findings provide starting points for diagnosing evaluation failures, while experiments are needed to determine which harness changes actually help. Yet existing approaches place these activities inside a fixed optimization procedure. This leaves the process of improvement itself outside the space of adaptation. We instead ask how harness optimization can be formulated as an environment in which both the harness and the process used to improve it can evolve through interaction.

To address this challenge, we introduce **VideoForge**, a recursive self-improvement environment for harness evolution in video evaluation. VideoForge exposes task execution, evidence inspection, harness revision, and self-reflection as interaction primitives, allowing the improvement trajectory to adapt to observations acquired during interaction. Crucially, adaptation is not limited to the task harness: experience accumulated along the trajectory can also modify how subsequent improvement is organized. VideoForge therefore supports evolution at two levels—the harness being optimized and the process that drives its optimization. We study whether this form of recursive self-improvement can produce sustained gains in video evaluation.

To support harness evolution and systematically evaluate diagnostic capabilities, we construct **VidDiag**, a benchmark comprising 1,087 actual video generation and editing cases and 2,932 human-annotated defect instances across seven dimensions and 27 fine-grained failure types. Each case includes the generated video, the original instruction, and available multimodal reference materials, with human-annotated defect categories and descriptions providing the basis for diagnostic evaluation. Feedback from training cases supports harness improvement, while held-out test evaluation enables comparisons across multimodal models, agent configurations, and harness optimization methods. We further use 100 annotated cases as repair targets to study *video debugging*. Through this experiment, we explore the capability of current agents to repair video defects and assess how access to human-annotated defect descriptions affects repair performance. Human evaluation measures whether the target defects are successfully resolved. Our contributions are threefold:

- We introduce VideoForge, a recursive self-improvement environment for agentic video evaluation. To the best of our knowledge, VideoForge is the first framework to enable recursive evolution of both video evaluation capabilities and the process used to improve them.
- We provide human-annotated defects and a failure taxonomy grounded in observed generation and editing failures. By evaluating both defect category identification and instance-level descriptions, VidDiag provides a common basis for assessing and improving video defect diagnosis.
- We systematically benchmark current approaches to video evaluation. To the best of our knowledge, we further explore one of the first systematic evaluations of video debugging, examining both current agents' ability to repair video defects and the effect of descriptions on repair performance.

2 Related work

AI-Generated Video Evaluation and Refinement. Existing benchmarks evaluate generated videos across diverse dimensions, including visual quality and temporal consistency (Huang et al., 2024; 2025), physical plausibility (Bansal et al., 2025), conditioning and reference fidelity (Kang et al., 2026; Zhang et al., 2026c), and multimodal or audio-visual alignment (Zhu et al., 2026; Zhou et al., 2026). Table 1 compares representative video evaluation and refinement works in terms of conditioning modalities, fine-grained diagnostic capabilities, and video repair. Some dimensions can be evaluated with predefined metrics, while more complex ones require model-based assessment (Zhang et al., 2026c). Recent work moves toward more fine-grained and agentic evaluation, using active inspection, evidence gathering, error localization (Chinchure et al., 2025; Zeng et al., 2026), and reusable skills for structured video assessment (Wei et al., 2026a). Evaluation feedback is also used for video refinement through prompt optimization and inference-time search (Long et al., 2026; Cheng et al., 2025; Liu et al., 2025; Song et al., 2026), localized error correction (Lee et al., 2026a). As video evaluation becomes increasingly important and agentic, improving the quality of the evaluation process becomes an important problem.

Table 1: **Comparison of video evaluation and refinement works.** Conditions exclude generated outputs. Evaluation capabilities include associated evaluators.

Work	Text	Image	Audio	Video	Desc.	Loc.	Agentic	Repair
VBench++ (Huang et al., 2025)	✓	✓	—	—	—	—	—	—
Spotlight (Chinchure et al., 2025)	✓	—	—	—	✓	✓	✓*	—
UniVBench (Wei et al., 2026b)	✓	✓	—	✓	✓	—	✓	—
CLVG-Bench (Zhang et al., 2026c)	✓	✓	✓	✓	✓	—	—	—
VideoRepair (Lee et al., 2026a)	✓	—	—	—	✓	✓	—	✓
VQQA (Song et al., 2026)	✓	✓	—	—	✓	—	✓	✓
VideoArgus (Zeng et al., 2026)	✓	✓	—	✓	✓	—	✓	—
VideoForge	✓	✓	✓	✓	✓	—	✓	✓†

Note: Desc.: case-specific defect descriptions; Loc.: explicit error intervals or regions; Agentic: agentic or multi-agent evaluation; Repair: reported defect correction, including diagnosis-guided regeneration. —: not reported in the compared setting. *Category-wise multi-agent baseline. †Separate 100-case human study with and without ground-truth defect descriptions.

Self-Evolving Agents. Early work such as ADAS explores automatically designing agentic systems (Hu et al., 2025). Recent work improves agents by adapting their external harnesses from experience, including instructions and contextual knowledge (Agrawal et al., 2026; Zhang et al., 2026b), reusable skills (Yang et al., 2026; Shen et al., 2026), and executable or runtime structures (Lee et al., 2026b; Lin et al., 2026a; Xu et al., 2026; Karten et al., 2026; Zhang et al., 2026a). This direction has also been explored in long-video understanding, including autonomous tool evolution (Ye et al., 2026), evidence-grounded failure attribution and module updates (Cui et al., 2026), and video evidence organization (Xu & Chen, 2026). However, while these methods evolve the agent harness, the outer improvement loop remains fixed. This is limiting for video evaluation, where different failures may require different multimodal evidence, inspection tools, and validation procedures. We therefore study whether the improvement process itself can be autonomously organized and refined through experience.

3 Dataset & Preliminaries

3.1 VidDiag

VidDiag contains 1,087 video generation and editing cases collected from real-world generative AI applications, with 2,932 human-annotated defect instances spanning seven dimensions and 27 fine-grained failure types. Each case includes a generated video, its generation instructions, and available multimodal references. The cases cover creative production, commercial content creation, and educational and technical visualization.

Stage I: Failure Annotation. Each case is independently annotated by two annotators and subsequently reviewed by a third annotator. Annotators examine the generated video alongside its generation instructions and available references, and describe all observed defects in natural language. Each description specifies the affected content and the observed problem, including deviations from the instructions or reference assets where applicable. When multiple problems occur within the same video, annotators record them separately when possible. Figure **??** illustrates a representative case, including the generation conditions, an observed defect, and its use as feedback for subsequent video debugging.

Stage II: Taxonomy Construction and Structuring. We construct a two-level failure taxonomy by grouping similar human-annotated defects into fine-grained error types and broader dimensions. The resulting taxonomy comprises seven dimensions and 27 fine-grained types. The seven dimensions cover reference consistency, instruction following, text rendering, spatial and action correctness, visual quality, video editing, and speech and audio. Each fine-grained type is defined by its characteristic failure pattern and its boundaries with related types.

Further details on data collection, annotation, and the failure taxonomy are provided in Appendix C.

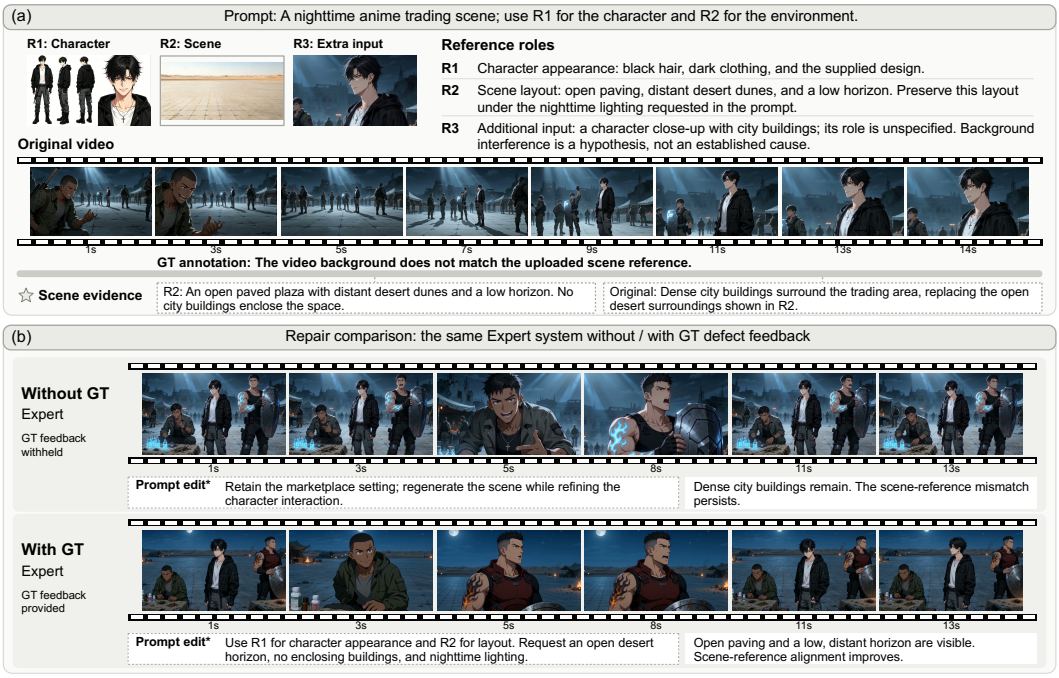


Figure 1: An illustrative example of the video debugging task. (a) Given a user prompt, reference images, and a generated video, the task is to identify generation defects and repair the video accordingly. Here, the character follows R1, but the urban background conflicts with the open desert scene specified by R2. (b) The same Expert system produces different repairs depending on whether ground-truth (GT) defect feedback is provided. Without feedback, the scene mismatch persists; with feedback, the revised prompt clarifies reference roles and restores an open layout that better matches R2.

3.2 PRELIMINARIES

We represent the harness as

$$H_t = (S_t, \mathcal{T}_t, \mathcal{K}_t, \mathcal{A}_t), \tag{1}$$

where S_t, $\mathcal{T}_t$, $\mathcal{K}_t$, and $\mathcal{A}_t$ denote the main agent's system prompt, tools, skills, and subagents, respectively. The harness specifies how these components are invoked and how their outputs inform the final prediction.

Given a generated video V and its generation conditions $\mathcal{R}$, the task is to identify and describe errors in V. Here, $\mathcal{R}$ comprises generation instructions and multimodal references. We denote the ground-truth error set by E and the agent's prediction by $\hat{E}$, where each element specifies an error category and its description. The task agent predicts $\hat{E}$ from $(V, \mathcal{R})$ using the harness without access to E. A fixed evaluator J scores $\hat{E}$ against E. With model parameters θ fixed, we optimize the harness to maximize the expected evaluation score:

$$H^\star \in \arg\max_{H \in \mathcal{H}} \mathbb{E}_{\substack{(V,\mathcal{R},E)\sim\mathcal{D} \\ \hat{E}\sim p_\theta(\cdot|V,\mathcal{R},H)}} \left[J(\hat{E}, E) \right], \tag{2}$$

where $\mathcal{H}$ is the harness search space and $\mathcal{D}$ is the task distribution. We pursue this objective through budgeted research on training cases and candidate selection under a fixed validation protocol.

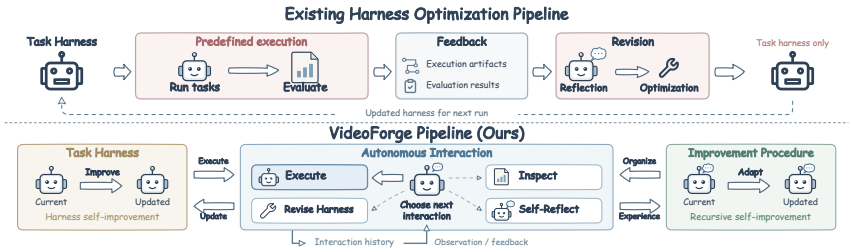


Figure 2: **Overview of VideoForge.** VideoForge enables feedback-driven interaction selection and recursive evolution of both the task harness and the improvement procedure.

We further study video debugging, where an agent B revises the video with optional defect feedback:

$$V' = B(V, \mathcal{R}; F), \qquad F \in \{\emptyset, E\}, \tag{3}$$

where $F = \emptyset$ and $F = E$ denote the settings without and with ground-truth defect feedback, respectively. Human evaluation assesses whether the target defects are resolved.

4 VideoForge

4.1 Interaction Primitives

VideoForge formulates harness optimization as interaction with an environment rather than a fixed outer loop. We denote the space of interactions available in VideoForge by Ω_{VF}, with

$$a_t \in \Omega_{\mathrm{VF}} \tag{4}$$

denoting the interaction performed at step t. An interaction may operate on task cases, execution records, evaluation results, or the editable harness itself, and returns an observation o_t.

Task execution is one such interaction. Given a harness H and a video-generation case $(V, \mathcal{R})$, we write

$$a_t = \mathcal{X}(H; V, \mathcal{R}), \qquad o_t = (\hat{E}, \tau, r), \tag{5}$$

where

$$(\hat{E}, \tau) \sim p_\theta(\cdot \mid V, \mathcal{R}, H), \qquad r = J(\hat{E}, E). \tag{6}$$

VideoForge supports evidence retrieval and inspection through

$$a_t = \mathcal{I}(x; q), \qquad o_t = z, \tag{7}$$

where x is an evidence source, q is an investigation request, and z is the resulting evidence, such as execution traces or evaluation results.

The harness itself is also an object of interaction. We represent a revision by

$$a_t = \mathcal{U}(H; \Delta), \qquad o_t = H', \tag{8}$$

where Δ specifies edits to one or more components of the harness.

Interactions and their observations are recorded in order. We represent the sequence of interactions observed up to step t by

$$\Gamma_t = [(a_0, o_0), (a_1, o_1), \ldots, (a_{t-1}, o_{t-1})]\,. \tag{9}$$

4.2 Harness Self-Improvement

At each step, the next interaction is selected according to

$$a_{t,k} \sim \pi_t\left(\cdot \mid H_t, \Gamma_{t,k}\right), \tag{10}$$

where π_t denotes the procedure that organizes improvement in session t. Executing $a_{t,k}$ produces an observation $o_{t,k}$, which extends the trajectory:

$$\Gamma_{t,k+1} = \Gamma_{t,k} \parallel (a_{t,k}, o_{t,k}). \tag{11}$$

The observation can immediately affect the next interaction. For example, an execution result may motivate further inspection, an inspection may motivate a harness revision, and an experimental result may trigger another revision or a return to earlier evidence. Thus, the interaction sequence is constructed during improvement rather than following a predefined ordering of stages.

Each improvement session operates under a fixed research budget. Within this budget, VideoForge may continue interacting with the environment, evaluate candidate harnesses under the fixed validation protocol, or conclude the session. Validation feedback, when available, is incorporated into the interaction trajectory and may inform subsequent interactions. Detailed budget and validation procedures are provided in App B.

4.3 Recursive Self-Improvement Environments

t

A fixed outer-loop optimizer keeps the procedure organizing improvement unchanged across sessions:

$$\pi_t = \pi, \qquad \forall t. \tag{12}$$

VideoForge instead allows the improvement procedure itself to evolve from accumulated research experience.

We introduce a self-reflection interaction

$$a_t = \mathcal{S}(\Gamma_t; q), \qquad o_t = c_t^{\text{self}}, \tag{13}$$

where Γ_t is the accumulated interaction trajectory and c_t^{self} captures experience about how improvement has been conducted and how it may be adapted.

After an improvement session, the accumulated experience can therefore drive evolution at two levels:

$$H_t \xrightarrow{\mathcal{U}} H_{t+1}, \qquad \pi_t \xrightarrow{c_t^{\text{self}}} \pi_{t+1}. \tag{14}$$

The first revises the task harness, while the second changes how subsequent improvement is organized. Either transition may be the identity.

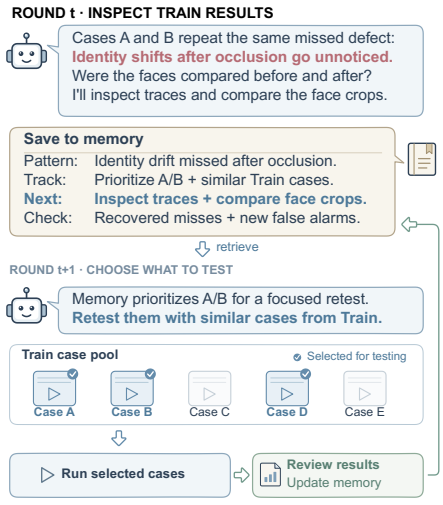


Figure 3: Memory-guided interaction selection in VideoForge. Experience from missed defects guides subsequent inspection and training-case selection, while new observations update memory for further improvement.

Beyond revising the task harness, experience accumulated through interaction can also update the improvement environment that governs subsequent sessions, enabling recursive evolution of both the harness and its improvement environment. Figure **??**

5 Experiments

5.1 Experimental Setup

Dataset and evaluation.

Dataset and evaluation. We split VidDiag into 217 training, 109 validation, and 761 test cases (2:1:7). For video diagnosis, we report category-level F1 and an LLM-based semantic matching score. F1 is computed over deduplicated defect-label sets for each vide. For semantic evaluation, we use Kimi K3 to perform one-to-one matching between predicted and reference descriptions based on

whether they identify the same concrete defect. The score is the fraction of reference defects whose labels are detected and whose descriptions are semantically matched. Overall scores are weighted by the number of reference defects in each category. Video debugging is evaluated by human experts using repair success and pairwise preference.

Experimental settings. We benchmark a range of base models and agent harnesses on VidDiag to characterize their performance on fine-grained video evaluation. For harness optimization, we compare VideoForge with SkillOpt (Yang et al., 2026) and AHE (Lin et al., 2026a) under two model–agent configurations, Codex + Seed and DSH + DeepSeek. We additionally evaluate video debugging using the Expert-Designed and Baseline repair configurations, each with and without ground-truth defect descriptions, to assess whether fine-grained evaluation feedback is useful for video repair.

SkillOpt (Yang et al., 2026): We run four epochs over 217 training cases with a batch size of 40, yielding 24 update steps. At each step, trajectories are grouped by success using a reward threshold of 0.5 and analyzed in reflection minibatches of up to eight cases to propose instruction edits.

AHE (Lin et al., 2026a): We run four full-batch optimization rounds over 217 training cases. Each round uses Debugger diagnoses to guide harness revision, followed by reevaluation on the same cases. Including the initial baseline, this yields 1,085 case evaluations. The candidate with the highest training pass rate is selected for testing without a separate validation gate. VideoForge: We provide four APIs for task execution, evidence inspection, harness revision, and self-reflection, with an optimization budget of 1,000 case evaluations.

5.2 Experimental Results & Analysis

Table 2: **Diagnostic performance on VidDiag.** We compare direct inference with base models and coding-agent harnesses across seven defect dimensions. Each entry reports category-level F1 / LLM-based semantic matching score (%). RC, IF, TR, SA, VE, VQ, and AS denote reference consistency, instruction following, text rendering, spatial and action correctness, video editing, visual quality, and speech and audio, respectively.

Configuration		RC	IF	TR	SA	VE	VQ	AS	Overall
		F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)
Base LLMs									
DeepSeek 4.1 Flash		29.9 / 15.2	29.2 / 15.8	59.0 / 57.9	18.0 / 4.8	53.7 / 33.3	14.3 / 3.5	3.7 / 1.5	23.6 / 12.6
Kimi K3		**33.5** / 20.0	28.7 / 18.8	58.0 / 58.7	23.6 / 7.1	60.9 / 37.0	19.2 / 5.0	1.5 / 0.7	25.9 / 15.0
GLM 5.3		28.8 / 16.1	25.3 / 15.8	61.6 / 64.3	24.2 / 6.5	63.6 / 44.4	12.4 / 3.2	7.1 / 3.7	24.0 / 13.9
Seed 2.1 Pro		25.6 / 12.9	26.7 / 17.1	50.8 / 46.0	**29.3** / **10.4**	0.0 / 0.0	14.2 / 4.4	0.0 / 0.0	22.3 / 12.1
Coding Agents									
Seed 2.1 Pro	+ OpenCode	25.5 / 14.6	28.5 / 17.6	51.1 / 47.6	28.3 / 8.0	7.1 / 3.7	14.8 / 4.9	**39.5** / **26.6**	28.0 / 15.9
	+ CC	15.4 / 5.4	24.1 / 10.5	55.4 / 51.6	17.0 / 5.7	20.0 / 0.0	12.7 / 3.5	37.3 / 25.1	22.6 / 11.7
	+ DSH	24.7 / 14.6	25.4 / 18.2	47.4 / 51.1	**29.3** / 7.1	6.5 / 3.4	16.6 / 4.3	33.0 / 17.0	26.6 / 14.8
	+ Codex	28.4 / 17.3	28.0 / 22.0	47.0 / 57.5	26.8 / 10.2	11.1 / 3.4	**20.0** / **7.5**	33.0 / 17.0	28.1 / 17.7
	+ OC	31.6 / 19.9	27.6 / 21.7	**63.2** / **72.5**	16.5 / 5.0	**92.0** / **60.4**	16.6 / 6.1	37.4 / 25.6	29.1 / **19.9**
DeepSeek 4.1 Flash	+ CC	27.3 / 17.1	28.2 / 22.1	56.1 / 64.3	21.8 / 6.7	90.2 / 59.2	12.5 / 4.6	32.7 / 22.3	27.4 / 18.5
	+ DSH	32.7 / **20.6**	**29.9** / **23.5**	54.0 / 61.9	22.3 / 6.8	56.4 / 37.0	19.7 / 7.3	35.1 / 24.0	**30.0** / 19.8
	+ Codex	31.1 / 19.6	27.2 / 21.4	62.3 / 71.4	16.3 / 4.9	90.6 / 59.5	16.4 / 6.0	36.8 / 25.2	28.6 / 19.6

Diagnostic performance. Table 2 compares direct model inference with coding-agent configurations. Direct-inference models achieve overall F1 scores of 22.3–25.9 and strict semantic scores of 12.1–15.0, leaving substantial room for improvement. Coding-agent configurations improve performance in several settings, but the gains depend on the model–agent pairing. For Seed, Codex increases overall F1 from 22.3 to 28.1 and the strict semantic score from 12.1 to 17.7. In contrast, Claude Code with the same task model achieves 22.6 F1 and 11.7 on the semantic metric, providing little improvement in category identification and a slightly lower semantic score.

The differences also vary across failure dimensions. For example, Seed with Codex improves speech and audio diagnosis from 0.0/0.0 to 33.0/17.0, whereas spatial and action correctness does not improve over direct inference. These results show that gains from an agent configuration are not uniform across diagnostic capabilities. Choosing an agent backend alone does not resolve fine-grained diagnosis, motivating further optimization of the task harness.

Table 3: Optimization performance on VidDiag. **Bold** and underlined values indicate the best and second-best results, respectively, within each configuration.

Configuration	Method	RC	IF	TR	SA	VE	VQ	AS	Overall
		F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)	F1 / Sem. (%)
Codex + Seed	+Vanilla	28.4 / 17.3	28.0 / 22.0	47.0 / 57.5	26.8 / **10.2**	11.1 / 3.4	20.0 / 7.5	33.0 / 17.0	28.1 / 17.7
	+SkillOpt	34.0 / 18.6	**35.1** / 18.1	**59.6** / **65.1**	**29.2** / 8.0	30.3 / 14.8	21.6 / 7.3	31.7 / **27.4**	32.5 / 18.7
	+AHE	28.1 / 17.2	29.0 / 18.5	53.6 / **65.1**	26.9 / 9.5	13.8 / 7.4	23.5 / 8.3	35.7 / 24.8	29.8 / 18.6
	+VideoForge w/o Evo.	33.6 / 22.8	30.0 / 23.5	56.3 / 62.7	25.4 / 9.4	60.0 / 40.7	23.2 / 9.3	36.0 / 24.7	31.7 / 21.3
	+VideoForge	**36.3 / 23.9**	32.0 / **26.0**	55.6 / 61.9	24.9 / 9.1	**63.4 / 44.4**	**24.3 / 10.2**	**37.3** / 25.8	**33.0 / 22.3**
DSH + DeepSeek	+Vanilla	32.7 / 20.6	29.9 / 23.5	54.0 / 61.9	22.3 / 6.8	56.4 / 37.0	19.7 / 7.3	35.1 / 24.0	30.0 / 19.8
	+SkillOpt	36.0 / 23.0	**34.0** / 25.0	57.5 / 64.5	24.0 / 7.5	63.0 / 41.0	22.2 / 9.0	36.5 / 26.0	32.9 / 21.5
	+AHE	34.0 / 22.0	32.0 / 24.9	56.5 / 64.0	23.0 / 7.2	61.0 / 40.0	21.8 / 8.5	36.2 / 25.5	31.6 / 21.0
	+VideoForge w/o Evo.	35.4 / 23.3	31.2 / 23.5	57.5 / **65.9**	**25.1** / 8.0	61.2 / 40.7	**23.8** / 9.3	36.3 / 25.8	32.5 / 21.5
	+VideoForge	**37.3 / 24.5**	33.8 / **26.3**	**58.0** / 65.0	24.5 / **8.2**	**64.0 / 43.0**	23.5 / **10.0**	**37.8 / 27.5**	**33.6 / 22.7**

Effectiveness of harness optimization. Table 3 reports the results of harness optimization. VideoForge achieves the highest overall F1 and strict semantic score among the compared methods in both evaluated configurations. For Codex + Seed, it improves overall performance from 28.1/17.7 to 33.0/22.3, corresponding to gains of 4.9 F1 points and 4.6 semantic-score points. For DSH + DeepSeek, it improves performance from 30.0/19.8 to 33.6/22.7, yielding gains of 3.6 and 2.9 points, respectively.

The advantage over SkillOpt is more pronounced on the semantic metric than on category F1. For Codex + Seed, VideoForge exceeds SkillOpt by 0.5 F1 points and 3.6 semantic-score points; for DSH + DeepSeek, the corresponding margins are 0.7 and 1.2 points. These results suggest that the benefit is not limited to identifying failure types, but also includes better correspondence with the specific defects recorded in the annotations under the fixed semantic-matching protocol.

Dimension-level results show broad but nonuniform gains. Relative to Base, VideoForge improves both metrics in six of seven dimensions for Codex + Seed and all seven dimensions for DSH + DeepSeek. For Codex + Seed, video editing exhibits a particularly large increase, from 11.1/3.4 to 63.4/44.4. However, spatial and action correctness decreases from 26.8/10.2 to 24.9/9.1. Thus, the overall improvements do not imply that every diagnostic capability benefits equally, and some dimensions remain challenging after optimization.

The ablation further shows that the evolutionary component improves the effectiveness of the optimization process rather than simply increasing the optimization budget. For Codex + Seed, adding evolution raises the update acceptance rate from 30.0% to 60.0% and reduces the number of task executions per effective update from 333 to 167, while increasing optimizer input tokens only moderately from 154.3M to 163.2M. The resulting improvement also increases from $+3.6/+3.6$ to $+4.9/+4.6$. A similar, though smaller, pattern appears for DSH + DeepSeek: evolution increases the acceptance rate from 25.0% to 30.0%, reduces case executions per effective update from 500 to 333, and improves ΔOverall from $+2.5/+1.7$ to $+3.6/+2.9$. These results suggest that evolution primarily improves the yield of optimization: under the same 1,000-execution budget, a larger fraction of proposed updates become effective improvements.

Optimization efficiency. Table 4 reports case-level task executions and optimizer input tokens. VideoForge uses 1,000 task executions in each configuration, compared with 3,604 for SkillOpt and 1,085 for AHE. It therefore achieves higher final diagnostic scores with approximately 72.3% fewer task executions than SkillOpt.

Table 4: **Optimization efficiency on VidDiag.** ΔOverall denotes the improvement in Overall F1 / strict semantic score over the corresponding unoptimized Base. Accept. is the percentage of proposed harness updates that are accepted. Total denotes the overall optimization cost, while /Eff. denotes the average cost per effective harness update. Optimizer input tokens are reported in millions. **Bold** indicates the best result for each metric within each configuration.

Configuration	Method	ΔOverall ↑	Accept. (%) ↑	Case Efficiency		Token Efficiency	
				Total ↓	/ Eff. ↓	Total (M) ↓	/ Eff. (M) ↓
Codex + Seed	+SkillOpt	+4.4 / +1.0	25.0	3,604	601	**18.9**	**3.2**
	+AHE	+1.7 / +0.9	25.0	1,085	1,085	708.0	708.0
	+Ours w/o Evo.	+3.6 / +3.6	30.0	**1,000**	333	154.3	51.4
	+Ours	**+4.9 / +4.6**	**60.0**	**1,000**	**167**	163.2	27.2
DSH + DeepSeek	+SkillOpt	+2.9 / +1.7	29.2	3,604	515	**36.2**	**5.2**
	+AHE	+1.6 / +1.2	25.0	1,085	1,085	1,062.0	1,062.0
	+Ours w/o Evo.	+2.5 / +1.7	25.0	**1,000**	500	394.4	197.2
	+Ours	**+3.6 / +2.9**	**30.0**	**1,000**	**333**	407.9	136.0

Optimizer input tokens reveal a different trade-off. VideoForge consumes 163.2M tokens for Codex + Seed and 407.9M for DSH + DeepSeek, fewer than AHE's 708.0M and 1,062.0M, respectively. However, SkillOpt uses substantially fewer optimizer input tokens, at 18.9M and 36.2M. VideoForge thus combines fewer task executions with higher final performance, while requiring more optimizer input tokens than SkillOpt. These measurements characterize the trade-off between task execution counts and optimizer-side token usage, rather than establishing a uniform reduction in total computational cost.

5.3 Video Debugging

Table 5: **Video debugging with and without GT defect guidance.** Each configuration is evaluated on 100 cases. Repair success and pairwise preference are judged by human experts.

Method	Repair Success w/o / w/ GT	Preference w/o / Tie / w/ GT
Expert-Designed	21.0 / **32.0**	10.0 / 38.0 / **52.0**
Baseline	11.0 / **23.0**	15.0 / 30.0 / **55.0**

We further examine whether explicit defect feedback can assist video repair. Providing ground-truth defect descriptions improves repair success under both configurations, from 21.0% to 32.0% for Expert-Designed and from 11.0% to 23.0% for Baseline (Table 5). Pairwise evaluation shows the same trend: repairs with GT guidance are preferred in 52.0% and 55.0% of cases, whereas repairs without guidance are preferred in only 10.0% and 15.0%, respectively.

These results show that fine-grained defect descriptions provide useful and actionable guidance for video repair. However, even with ground-truth defects, the highest repair success is 32.0%, indicating that identifying what should be corrected and successfully performing the correction remain distinct challenges.

6 Conclusion

We introduced VideoForge, a recursive self-improvement environment that adapts both the video evaluation harness and its improvement procedure, and VidDiag, a benchmark for fine-grained video defect diagnosis. Across two model–agent configurations, VideoForge outperforms SkillOpt and AHE, while ablations support the benefit of adapting the improvement process itself. A separate video-debugging study further shows that ground-truth defect descriptions improve repair, although substantial failures remain even when defects are known. These results highlight the value of adaptive harness improvement and the remaining challenge of translating reliable diagnosis into effective correction.

AI USE STATEMENT

Generative AI tools were used to assist with code review and the writing and editing of the manuscript. All AI-assisted outputs were reviewed and validated by all authors. All authors take full responsibility for the final content of this work.

ETHICS STATEMENT

This work focuses on the evaluation and refinement of AI-generated video content. We follow the ICLR Code of Ethics throughout benchmark construction, human annotation, experimentation, and evaluation. The benchmark is intended solely for research purposes.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, et al. Gepa: Reflective prompt evolution can outperform reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 8479–8565, 2026.

Hritik Bansal, Zongyu Lin, Tianyi Xie, Zeshun Zong, Michal Yarom, Yonatan Bitton, Chenfanfu Jiang, Yizhou Sun, Kai-Wei Chang, and Aditya Grover. Videophy: Evaluating physical commonsense for video generation. In *International Conference on Learning Representations*, volume 2025, pp. 102075–102121, 2025.

Jiale Cheng, Ruiliang Lyu, Xiaotao Gu, Xiao Liu, Jiazheng Xu, Yida Lu, Jiayan Teng, Zhuoyi Yang, Yuxiao Dong, Jie Tang, et al. Vpo: Aligning text-to-video generation models with prompt optimization. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15636–15645. IEEE, 2025.

Aditya Chinchure, Sahithya Ravi, Pushkar Shukla, Vered Shwartz, and Leonid Sigal. Spotlight: Identifying and localizing video generation errors using vlms. *arXiv preprint arXiv:2511.18102*, 2025.

Eli Collins. Fuel your creativity with new generative media models and tools. Google Blog, May 2025. URL https://blog.google/innovation-and-ai/products/generative-media-models-io-2025/.

Benlei Cui, Ruize Wang, Junjie Li, Jinhao Chen, Longtao Huang, Yinghao Chen, Yuwen Zhai, Jingqun Tang, Ruijian Jia, Weiwei Wu, et al. Metavideoagent: Automated video-agent evolution for long-form video understanding. *arXiv preprint arXiv:2608.04587*, 2026.

Databricks, Inc. Omnigent: Declarative agent authoring and runtime framework. GitHub repository, 2026. URL https://github.com/omnigent-ai/omnigent. Accessed September 7, 2026.

Yoav HaCohen, Benny Brazowski, Nisan Chiprut, Yaki Bitterman, Andrew Kvochko, Avishai Berkowitz, Daniel Shalem, Daphna Lifschitz, Dudu Moshe, Eitan Porat, Eitan Richardson, Guy Shiran, Itay Chachy, Jonathan Chetboun, Michael Finkelson, Michael Kupchick, Nir Zabari, Nitzan Guetta, Noa Kotler, Ofir Bibi, Ori Gordon, Poriya Panet, Roi Benita, Shahar Armon, Victor Kulikov, Yaron Inger, Yonatan Shiftan, Zeev Melumian, and Zeev Farbman. Ltx-2: Efficient joint audio-visual foundation model, 2026. URL https://arxiv.org/abs/2601.03233.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems, 2025. URL https://arxiv.org/abs/2408.08435.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21807–21818. IEEE, 2024.

Ziqi Huang, Fan Zhang, Xiaojie Xu, Yinan He, Jiashuo Yu, Ziyue Dong, Qianli Ma, Nattapol Chanpaisit, Chenyang Si, Yuming Jiang, et al. Vbench++: Comprehensive and versatile benchmark suite for video generative models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

Jenna Kang, Maria Beatriz Silva, Patsorn Sangkloy, Kenneth Chen, Niall L Williams, and Qi Sun. Geneva: A dataset of human annotations for generative text to video artifacts. In *2026 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 6174–6183. IEEE, 2026.

Seth Karten, Joel Zhang, Tersoo Upaa Jr, Ruirong Feng, Wenzhe Li, Chengshuai Shi, Chi Jin, and Kiran Vodrahalli. Continual harness: Online adaptation for self-improving foundation agents. *arXiv preprint arXiv:2605.09998*, 2026.

Daeun Lee, Jaehong Yoon, Jaemin Cho, and Mohit Bansal. Self-correcting text-to-video generation with misalignment detection and localized refinement. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 36464–36489, 2026a.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026b.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, et al. Agentic harness engineering: Observability-driven automatic evolution of coding-agent harnesses. *arXiv preprint arXiv:2604.25850*, 2026a.

Jingyang Lin, Jialian Wu, Jiang Liu, Ximeng Sun, Ze Wang, Xiaodong Yu, Jiebo Luo, Zicheng Liu, and Emad Barsoum. Videoseek: Long-horizon video agent with tool-guided seeking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5465–5475, June 2026b.

Juyi Lin, Arash Akbari, Yumei He, Lin Zhao, Haichao Zhang, Arman Akbari, Xingchen Xu, Zoe Y. Lu, Enfu Nan, Hokin Deng, Edmund Yeh, Sarah Ostadabbas, Yun Fu, Jennifer Dy, Pu Zhao, and Yanzhi Wang. Phyground: Benchmarking physical reasoning in generative world models, 2026c. URL https://arxiv.org/abs/2605.10806.

Minhua Lin, Hanqing Lu, Zhan Shi, Bing He, Rui Mao, Zhiwei Zhang, Zongyu Wu, Xianfeng Tang, Hui Liu, Zhenwei Dai, et al. Position: Agentic evolution is the path to evolving llms. *arXiv preprint arXiv:2602.00359*, 2026d.

Fangfu Liu, Hanyang Wang, Yimo Cai, Kaiyan Zhang, Xiaohang Zhan, and Yueqi Duan. Video-t1: Test-time scaling for video generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18671–18681. IEEE, 2025.

Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö Arik. Vista: A test-time self-improving video generation agent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6021–6032, 2026.

MiniMax. MiniMax H3: An open model breaking the boundaries between tasks and modalities. Official blog, July 2026. URL https://www.minimax.io/blog/minimax-h3.

Team Seedance, De Chen, Liyang Chen, Xin Chen, Ying Chen, Zhuo Chen, Zhuowei Chen, Feng Cheng, Tianheng Cheng, Yufeng Cheng, Mojie Chi, Xuyan Chi, Jian Cong, Qinpeng Cui, Fei Ding, Qide Dong, Yujiao Du, Haojie Duanmu, Junliang Fan, Jiarui Fang, Jing Fang, Zetao Fang, Chengjian Feng, Yu Gao, Diandian Gu, Dong Guo, Hanzhong Guo, Qiushan Guo, Boyang Hao, Hongxiang Hao, Haoxun He, Jiaao He, Qian He, Tuyen Hoang, Heng Hu, Ruoqing Hu, Yuxiang Hu, Jiancheng Huang, Weilin Huang, Zhaoyang Huang, Zhongyi Huang, Jishuo Jin, Ming Jing, Ashley Kim, Shanshan Lao, Yichong Leng, Bingchuan Li, Gen Li, Haifeng Li, Huixia Li, Jiashi Li, Ming Li, Xiaojie Li, Xingxing Li, Yameng Li, Yiying Li, Yu Li, Yueyan Li, Chao Liang, Han Liang, Jianzhong Liang, Ying Liang, Wang Liao, J. H. Lien, Shanchuan Lin, Xi Lin, Feng Ling, Yue Ling, Fangfang Liu, Jiawei Liu, Jihao Liu, Jingtuo Liu, Shu Liu, Sichao Liu, Wei Liu, Xue Liu, Zuxi Liu, Ruijie Lu, Lecheng Lyu, Jingting Ma, Tianxiang Ma, Xiaonan Nie, Jingzhe Ning, Junjie Pan, Xitong Pan, Ronggui Peng, Xueqiong Qu, Yuxi Ren, Yuchen Shen, Guang Shi, Lei Shi, Yinglong Song, Fan Sun, Li Sun, Renfei Sun, Wenjing Tang, Boyang Tao, Zirui Tao,

Dongliang Wang, Feng Wang, Hulin Wang, Ke Wang, Qingyi Wang, Rui Wang, Shuai Wang, Shulei Wang, Weichen Wang, Xuanda Wang, Yanhui Wang, Yue Wang, Yuping Wang, Yuxuan Wang, Zijie Wang, Ziyu Wang, Guoqiang Wei, Meng Wei, Di Wu, Guohong Wu, Hanjie Wu, Huachao Wu, Jian Wu, Jie Wu, Ruolan Wu, Shaojin Wu, Xiaohu Wu, Xinglong Wu, Yonghui Wu, Ruiqi Xia, Xin Xia, Xuefeng Xiao, Shuang Xu, Bangbang Yang, Jiaqi Yang, Runkai Yang, Tao Yang, Yihang Yang, Zhixian Yang, Ziyan Yang, Fulong Ye, Bingqian Yi, Xing Yin, Yongbin You, Linxiao Yuan, Weihong Zeng, Xuejiao Zeng, Yan Zeng, Siyu Zhai, Zhonghua Zhai, Bowen Zhang, Chenlin Zhang, Heng Zhang, Jun Zhang, Manlin Zhang, Peiyuan Zhang, Shuo Zhang, Xiaohe Zhang, Xiaoying Zhang, Xinyan Zhang, Xinyi Zhang, Yichi Zhang, Zixiang Zhang, Haiyu Zhao, Huating Zhao, Liming Zhao, Yian Zhao, Guangcong Zheng, Jianbin Zheng, Xiaozheng Zheng, Zerong Zheng, Kuan Zhu, and Feilong Zuo. Seedance 2.0: Advancing video generation for world complexity, 2026.

Yifei Shen, Bo Li, and Xinjie Zhang. Skillopt-lite: Better and faster agent self-evolution via one line of vibe. *arXiv preprint arXiv:2607.03451*, 2026.

Yiwen Song, Tomas Pfister, and Yale Song. Vqqa: An agentic approach for video evaluation and quality improvement, 2026. URL https://arxiv.org/abs/2603.12310.

Harbor Framework Team. Harbor: A framework for evaluating and optimizing agents and models in container environments. *Zenodo*, 2026.

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

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent, 2024. URL https://arxiv.org/abs/2403.10517.

Jianhui Wei, Jie Tan, Hengchuan Zhu, Xiaotian Zhang, Yan Zhang, Ziyi Chen, Daoan Zhang, Wei Xu, and Zuozhu Liu. Videoweaver: Evaluating and evolving skills for agentic long video generation, 2026a. URL https://arxiv.org/abs/2606.08091.

Jianhui Wei, Xiaotian Zhang, Yichen Li, Yuan Wang, Yan Zhang, Ziyi Chen, Zhihang Tang, Wei Xu, and Zuozhu Liu. UniVBench: Towards unified evaluation for video foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026b.

Jiangtao Wu, Jiaming Wang, Yiwen He, Yuanxing Zhang, Shihao Li, Dunyuan Liu, Xuedong Zhao, Jialu Chen, Zekun Moore Wang, and Jiaheng Liu. Covebench: Can video editing models handle complex instructions?, 2026. URL https://arxiv.org/abs/2606.08415.

Guoyang Xu and Hao Chen. Videoharness-rsi: Recursive harness self-improvement for long-video understanding with frozen vision-language models, 2026. URL https://arxiv.org/abs/2608.24302.

Tianshi Xu, Huifeng Wen, and Meng Li. Adapting the interface, not the model: Runtime harness adaptation for deterministic llm agents. *arXiv preprint arXiv:2605.22166*, 2026.

Yifan Yang, Ziyang Gong, Weiquan Huang, Qihao Yang, Ziwei Zhou, Zisu Huang, Yan Li, Xuemei Gao, Qi Dai, Bei Liu, et al. Skillopt: Executive strategy for self-evolving agent skills. *arXiv preprint arXiv:2605.23904*, 2026.

Dingqiang Ye, Dongdi Zhao, Kaishen Wang, Qingqiao Hu, Jingchen Sun, Yijun Liang, Yuqi Jia, Yiqiao Huang, Yunjie Tian, Jiaxing Zhang, Chuanyang Jin, Ke Zhang, Vishal M. Patel, and Di Fu. Videoresearcher: Self-improving tool design for long-video understanding, 2026. URL https://arxiv.org/abs/2609.19664.

Ziyun Zeng, Zixuan Wang, Yongsheng Yu, Hang Hua, and Jiebo Luo. Videoargus: Agentic rubric-grounded unified evaluation for video generation and editing, 2026. URL https://arxiv.org/abs/2608.05485.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-harness: Harnesses that improve themselves. *arXiv preprint arXiv:2606.09498*, 2026a.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, et al. Agentic context engineering: Evolving contexts for self-improving language models. In *International Conference on Learning Representations*, volume 2026, pp. 86069–86100, 2026b.

Xiaotian Zhang, Jianhui Wei, Yuan Wang, Jie Tan, Yichen Li, Yan Zhang, Ziyi Chen, Daoan Zhang, Dezhi Yu, Wei Xu, et al. How far are video models from true multimodal reasoning? In *European Conference on Computer Vision*, pp. 112–129. Springer, 2026c.

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding, 2025.

Yanghao Zhou, Haitian Li, Rexar Lin, He-Yan Huang, Jinxing Zhou, Changsen Yuan, Tian Lan, Ziqin Zhou, Yudong Li, Jiajun Xu, et al. Mtavg-bench: A diagnostic benchmark for multi-talker dialogue-centric audio-video generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 19906–19942, 2026.

Chen Zhu, Jiashu Zhu, Yanxun Li, Meiqi Wu, Bingze Song, Chubin Chen, Jiahong Wu, Xiangxiang Chu, and Yangang Wang. Artifact-aware evaluation for high-quality video generation, 2026.

Algorithm 1 VideoForge Self-Evolving Harness Optimization

Require: Training set $\mathcal{D}_{\text{tr}}$, validation set $\mathcal{D}_{\text{val}}$, held-out test set $\mathcal{D}_{\text{te}}$, interaction space Ω_{VF}, initial task harness H_0, initial improvement procedure π_0, fixed evaluator J, per-session research budget B, maximum number of sessions T

for $t = 0, \ldots, T-1$ **do**
  $\Gamma_{t,0} \leftarrow [\,]$, $k \leftarrow 0$, $\mathcal{C}_t \leftarrow$
  while $C(\Gamma_{t,k}) < B$ **do**
    Select the next interaction:

$$a_{t,k} \sim \pi_t(\cdot \mid H_t, \Gamma_{t,k}), \qquad a_{t,k} \in \Omega_{\text{VF}} \cup \{\textsc{Stop}\}$$

    if $a_{t,k} = \textsc{Stop}$ **then**
      break
    end if
    Execute $a_{t,k}$ in the permitted training environment using $\mathcal{D}_{\text{tr}}$ and J as applicable; receive observation $o_{t,k}$
    $\Gamma_{t,k+1} \leftarrow \Gamma_{t,k} \parallel (a_{t,k}, o_{t,k})$
    if $o_{t,k}$ contains a revised harness H' **then**
      $\mathcal{C}_t \leftarrow \mathcal{C}_t \cup \{H'\}$
    end if
    $k \leftarrow k + 1$
  end while
  $\Gamma_{t,\text{end}} \leftarrow \Gamma_{t,k}$
  $H_{t+1} \leftarrow H_t$
  Select a candidate $H_t^{\text{cand}} \in \mathcal{C}_t$ for submission, if available
  if a candidate is submitted **then**
    Evaluate H_t^{cand} using J under the fixed validation protocol on $\mathcal{D}_{\text{val}}$
    Receive permitted validation feedback r_t^{val} and acceptance decision $d_t \in \{\textsc{Accept}, \textsc{Reject}\}$
    Append validation feedback and the decision:

$$\Gamma_{t,\text{end}} \leftarrow \Gamma_{t,\text{end}} \parallel \big(\textsc{Validate}(H_t^{\text{cand}}), (r_t^{\text{val}}, d_t)\big)$$

    if $d_t = \textsc{Accept}$ **then**
      $H_{t+1} \leftarrow H_t^{\text{cand}}$
    end if
  end if
  Update the improvement procedure using the session trajectory, including validation feedback when available:

$$\pi_{t+1} \leftarrow \text{Evolve}(\pi_t, \Gamma_{t,\text{end}})$$

  ▷ The update may be the identity
end for
Evaluate H_T once on $\mathcal{D}_{\text{te}}$ using J
return H_T

...

A VideoForge Optimization Procedure

B Harness Framework and Configuration

B.1 Unified Harness Framework

We implement all methods in a unified harness framework. A harness consists of four editable components: instructions, tools, skills, and subagents. Optimization methods operate on the same harness representation while retaining their own strategies for constructing, revising, and selecting candidates. Candidate harnesses are executed through a shared trial interface that standardizes task inputs, runtime execution, tracing, output extraction, and evaluation. Within each compari-

son group, task definitions, model settings, execution budgets, and evaluators are kept fixed. The framework design follows ideas from A-Evolve (Lin et al., 2026d), Harbor (Team, 2026), and OmniGent (Databricks, Inc., 2026) for editable harness artifacts, execution–evaluation separation, and declarative agent configuration, respectively.

B.2 Task-Specific Environments

The initial video-evaluation harness H_0 provides nine media-inspection skills (Table 6), all of which are also available in the video-debugging environment. Video debugging extends this shared set with fourteen additional skills for defect localization, repair, generation, assembly, and validation (Table 7), yielding 23 skills in total.

Table 6: Nine shared media-inspection skills and their main dependencies, available to both video-evaluation H_0 and video debugging.

Skill	**Capability**	**Dependencies**
`get-video-metadata`	Read duration, resolution, frame rate, and frame count.	ffprobe.
`get-image-metadata`	Read dimensions, format, size, and available metadata.	ExifTool.
`extract-video-frame`	Extract individual frames or sample a time interval.	OpenCV for individual frames; FFmpeg for dense sampling.
`grid-generate`	Build timestamped grids of sampled frames.	OpenCV and Pillow; PySceneDetect for shot-based sampling.
`video-shot-split`	Detect shots and optionally export video segments.	PySceneDetect; FFmpeg for segment export.
`ocr`	Recognize visible text with confidence and locations.	PaddleOCR (local inference).
`asr`	Transcribe speech with optional timestamps and speakers.	Volcengine BigASR API.
`audio-understanding`	Query a multimodal model about audio content.	Seed 2.0 Lite
`audio-vocal-separate`	Separate vocal and non-vocal estimates.	Demucs `htdemucs` and PyTorch (local inference).

Table 7: Fourteen additional skills and their main dependencies available only to video debugging.

Skill	Capability	Dependencies
`media-inspection`	Inspect metadata, verify complete decoding, and identify defects from evidence.	FFmpeg and ffprobe.
`frame-analysis`	Extract frames and analyze scene boundaries, visual anomalies, and motion evidence.	FFmpeg, ffprobe, OpenCV, and NumPy.
`visual-repair`	Correct local color, brightness, noise, flicker, and geometric defects.	FFmpeg, OpenCV, Pillow, and ImageMagick.
`motion-repair`	Reduce camera shake and geometric jitter using motion tracking and measured transforms.	FFmpeg, OpenCV, and NumPy.
`temporal-repair`	Repair frame timing, cadence, black or frozen frames, and short gaps using source evidence.	FFmpeg, ffprobe, PyAV, OpenCV, and NumPy.
`audio-repair`	Repair noise, hum, clicks, clipping, silence, gain, and channel defects.	FFmpeg, NumPy, SciPy, and SoundFile.
`av-sync-repair`	Correct audio–video offsets, drift, and replacement-track misalignment.	FFmpeg, ffprobe, NumPy, SciPy, SoundFile, and librosa.
`container-recovery`	Recover readable streams from damaged containers through remuxing and stream salvage.	FFmpeg and ffprobe.
`format-conversion`	Convert media to specified containers, codecs, resolutions, frame rates, and audio profiles.	FFmpeg and ffprobe.
`subtitle-repair`	Retime, convert, restyle, or remux existing subtitles while preserving their text.	FFmpeg, ffprobe, and pysubs2.
`video-assembly`	Rebuild video from edited frames or prepared streams with explicit timing and stream mapping.	FFmpeg and ffprobe.
`image-gen`	Generate or edit reference images for semantic visual defects beyond local repair.	Seedream 5.0.
`video-gen`	Regenerate bounded intervals for semantic visual defects unrecoverable from source pixels.	Seedance 2.5
`repair-validation`	Check decoding, defect resolution, content preservation, new defects, and evidence completeness.	FFmpeg, ffprobe, OpenCV, and NumPy.

C Additional Dataset Details

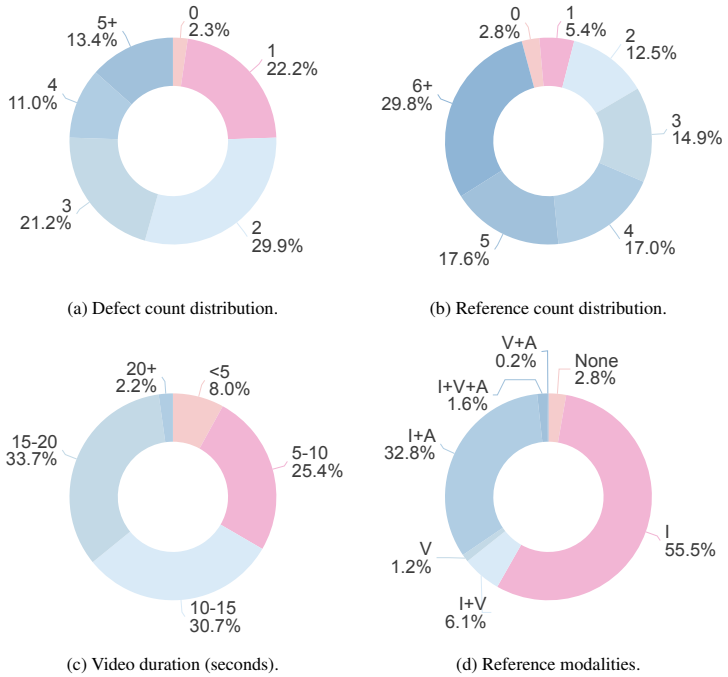
C.1 Annotation Pipeline

We adopt a two-stage annotation pipeline. Annotators first identify and describe defects in the collected videos. We then derive a failure taxonomy from these annotations and organize the observed defects into structured error instances.

Given the human-written defect descriptions and the taxonomy, we use a language model to separate independent problems into individual defect instances and assign each instance a single fine-grained category. Each instance is represented as a category–description pair, preserving the case-specific observations in the original annotation. Human reviewers verify that each structured description is supported by the original annotation and that its category assignment follows the taxonomy.

Stage I: Failure Annotation. Each case is independently annotated by two annotators and subsequently reviewed by a third annotator. Annotators examine the generated video alongside its generation instructions and available references, and describe all observed defects in natural language. Each description specifies the affected content and the observed problem, including deviations from the instructions or reference assets where applicable. When multiple problems occur within the same video, annotators record them separately when possible.

Stage II: Taxonomy Construction and Structuring. We construct a two-level failure taxonomy by grouping similar human-annotated defects into fine-grained error types and broader dimensions. The resulting taxonomy comprises seven dimensions and 27 fine-grained types. The seven dimensions cover reference consistency, instruction following, text rendering, spatial and action correctness, visual quality, video editing, and speech and audio. Each fine-grained type is defined by its characteristic failure pattern and its boundaries with related types.



(a) Defect count distribution.

(b) Reference count distribution.

(c) Video duration (seconds).

(d) Reference modalities.

Figure 4: **Additional VidDiag statistics.** All percentages use 1,087 cases. Reference counts refer to input-file entries and exclude the generated output. Duration bins include their lower bounds and exclude their upper bounds. In (d), I, V, and A denote image, video, and audio references; None denotes no reference files.

C.2 Annotation and Input Statistics

VidDiag contains 2,932 annotated failure instances, with an average of 2.70 instances per case (median: 2; maximum: 12). Among all cases, 821 contain at least two failure instances, and 725 contain failures spanning multiple dimensions. Multi-failure cases therefore make up a substantial portion of the benchmark, including both repeated instances within a failure category and combinations of failures across different dimensions.

The benchmark also includes heterogeneous multimodal conditioning inputs. The public input lists contain 4,614 reference-file entries, corresponding to an average of 4.3 references per case (median: 4). Image references are available for 1,042 cases, video references for 98 cases, and audio references for 375 cases; these modality groups are not mutually exclusive. Thirty cases contain no reference files and instead retain the original textual generation instructions as conditioning information.

Generated-video durations range from 4 to 30 seconds, Figure 4 summarizes the distributions of failure counts, reference counts, video durations, and reference-modality combinations, providing a more detailed view of the benchmark composition.

C.3 Failure Tag Statistics

Table 8 reports case coverage and instance counts for each failure tag; cases can carry multiple tags, percentages use all 1,087 cases (including the 25 cases with empty annotation sets), and instance counts sum to 2,932.

Table 8: Case coverage and instance counts for the 27 fine-grained failure tags, grouped by dimension.

Failure tag	Cases	Cases (%)	Instances
Reference consistency			
Character appearance mismatch	158	14.5	177
Facial identity mismatch	67	6.2	68
Scene inconsistency	141	13.0	153
Scene binding failure	65	6.0	66
Multi-character identity confusion	116	10.7	128
Prop binding failure	42	3.9	44
Instruction following			
Missing plot steps	232	21.3	301
Unrequested actions	46	4.2	47
Unrequested visual elements	143	13.2	149
Negative instruction violation	18	1.7	18
Camera / timing noncompliance	135	12.4	149
Text rendering			
Subtitle noncompliance	94	8.6	104
Incorrect rendered text	84	7.7	94
Spatial and action correctness			
Multi-character action mismatch	228	21.0	298
Spatial positioning errors	192	17.7	216
Visual quality			
Artificial-looking appearance	117	10.8	141
Anatomical abnormalities	81	7.5	83
Physical / motion errors	176	16.2	206
Poor fine-detail reproduction	62	5.7	68
Video editing			
Editing target failure	33	3.0	40
Speech and audio			
Pronunciation / delivery errors	164	15.1	180
Lip-sync errors	28	2.6	28
Dialogue speaker mismatch	28	2.6	28
Narration rendered as dialogue	15	1.4	15
Noise / unwanted audio	61	5.6	61
Unrequested dialogue	44	4.0	44
Audio reference mismatch	26	2.4	26

C.4 FINE-GRAINED FAILURE DEFINITIONS

Table 9: Definitions of the 27 fine-grained failure types, grouped into seven dimensions.

Failure type	Definition
Reference consistency	
Character appearance mismatch	Nonfacial appearance differs from references, specified character attributes, or the same character across shots.
Facial identity mismatch	Facial identity differs from its reference or changes, drifts, or merges across shots.
Scene inconsistency	Scene attributes, layout, or atmosphere conflict with the instructions or an established scene.
Scene binding failure	A specified scene reference, boundary frame, or named scene asset is omitted, replaced, or incorrectly inherited.
Multi-character identity confusion	Characters exchange identities or attributes, share faces, become duplicated, or are incorrectly replaced.
Prop binding failure	A required prop is missing, replaced, of the wrong type, or assigned to the wrong character.
Instruction following	
Missing plot steps	Required events, actions, appearances, dialogue, sounds, effects, or state changes are missing, incomplete, reordered, or mistimed.
Unrequested actions	Characters, animals, or objects perform additional actions that the generation instructions do not request.
Unrequested visual elements	Unrequested characters, props, animals, graphics, layouts, logos, or other visible entities appear.
Negative instruction violation	Explicitly prohibited content, states, or behavior appear without a more specific applicable failure type.
Camera / timing noncompliance	Duration, framing, viewpoint, camera movement, focus, cuts, transitions, or boundary shots violate requirements.
Text rendering	
Subtitle noncompliance	Subtitle presence, quantity, placement, style, timing, or readability does not follow the stated requirements.
Incorrect rendered text	Required or permitted text contains incorrect, missing, garbled, unreadable, or temporally unstable characters or content.
Spatial and action correctness	
Multi-character action mismatch	An action has the wrong performer, recipient, target, body part, direction, or coordination among characters.
Spatial positioning errors	Posture, relative position, distance, orientation, gaze, or entry and exit directions conflict with requirements.
Visual quality	
Artificial-looking appearance	Rendering artifacts include synthetic appearance, excessive smoothing, blur, flicker, dropped frames, noise, ghosting, or corruption.
Anatomical abnormalities	Bodies exhibit abnormal shape, topology, or proportions, including malformed fingers, missing limbs, duplication, or fusion.
Physical / motion errors	Motion or evolving visual states violate physics or continuity through interpenetration, teleportation, unsupported floating, or abrupt changes.
Poor fine-detail reproduction	The correct object is present, but its shape, count, size, material, structure, or fine details are incorrect.
Video editing	
Editing target failure	A requested video or image editing operation fails or takes effect only in some segments.
Speech and audio	
Pronunciation / delivery errors	Speech deviates from scripted wording or specified pronunciation, language, pacing, pauses, emphasis, tone, or emotion.

Continued on next page

Table 9 (continued)

Failure type	Definition
Lip-sync errors	Visible mouth movements do not match the spoken content or audio timing.
Dialogue speaker mismatch	Dialogue has the wrong speaker, turn, sequence, or assignment among multiple characters.
Narration rendered as dialogue	Narration, internal monologue, scene descriptions, or interface instructions are spoken aloud by an onscreen character.
Noise / unwanted audio	The soundtrack contains unintended nonverbal noise, distortion, interruptions, abrupt music, or incorrect sound effects.
Unrequested dialogue	Additional speaking events, complete utterances, narration, or full-line repetitions appear beyond the script.
Audio reference mismatch	Generated voices fail to match specified audio references or character voices, including incorrect voice assignment.

C.5 Counting Conventions

All statistics use the released case list and its current public reference inputs. Reference-file counts describe within-case file entries, not globally unique assets or independent sources of information. Reference modalities exclude the generated video and its soundtrack. Video durations are obtained from media metadata.

The 25 cases with empty annotation sets (2.3%) are included in every dataset-level denominator and in the zero-failure bin. An empty annotation set denotes no identified failure under the annotation protocol.

D Semantic Matching Stability

Table 10: **Overall semantic matching performance on VidDiag.** Results report the mean ± standard deviation (%) across 3 runs.

Configuration	Method	Overall Sem. (%)
Codex + Seed	Vanilla	17.7 ± 0.3
	SkillOpt	18.7 ± 0.2
	AHE	18.6 ± 0.1
	VideoForge w/o Evo.	21.3 ± 0.2
	VideoForge	$\mathbf{22.3 \pm 0.3}$
DSH + DeepSeek	Vanilla	19.8 ± 0.2
	SkillOpt	21.5 ± 0.3
	AHE	21.0 ± 0.1
	VideoForge w/o Evo.	21.5 ± 0.1
	VideoForge	$\mathbf{22.7 \pm 0.2}$

E Prompts

E.1 Evaluation Task prefix

Evaluation Task prefix

Analyze the current video-generation case using the following public inputs:

- Generation requirements: {`input_text_path`}.
- Generated video: {`output_video_path`}.
- Reference materials: {`reference_materials`}.

- Output file: {`output_path`}.

Determine which failures are supported by the generated video, the generation requirements, and the reference materials.

Create the specified output file when the analysis is complete. For each distinct failure, write one line in the following format:

`level1:level2:feedback`

Use the canonical labels from the supplied taxonomy. Describe the observed failure and provide relevant temporal or spatial details when available.

After successfully writing the final output, stop processing the case and reply only with `DONE`.

E.2 Initial Task Prompt

Initial Task Prompt (H_0)

You are a video failure detection agent.

Assess the generated video in relation to its generation requirements, reference materials, and the supplied failure taxonomy. Ground your conclusions in observable evidence. Distinguish an observed discrepancy from an assumption about what may have happened.

Follow the task prefix for input locations, workspace restrictions, output format, and completion behavior.

Taxonomy: {`taxonomy`}

The taxonomy contains the canonical category labels and their definitions. Use those labels exactly in the final output.

E.3 Main Improvement Harness Prompt

Improvement Harness (R_0)

You are an autonomous improvement harness.

Your objective is to develop and improve a task harness H for the specified task and improvement objective. If no task harness is provided, construct an initial one. You may also improve the improvement harness R when doing so can improve subsequent iterations.

You are given access to an authorized task environment containing the available task data, execution interfaces, evaluation feedback, tools, and computational resources. Decide autonomously how to use them to pursue the objective. You may inspect available information, run experiments, compare alternatives, modify harnesses, or stop when appropriate. No particular sequence of actions is prescribed.

The task harness may contain instructions, tools, skills, subagents, and their organization. You may add, remove, modify, or reorganize these components as needed. The improvement harness may be changed within the same editable scope when improving future improvement behavior.

Use observed task executions and evaluation results to support decisions. Do not use held-out test data or test results to guide improvement. Do not change fixed model parameters, data-access boundaries, the external evaluation protocol, or resource constraints.

Maintain a concise persistent state containing the current harnesses, established findings, unresolved issues, unsuccessful attempts, and information needed for subsequent improvement.

E.4 SEMANTIC EVALUATOR

Fixed Semantic Matching Judge

You are a text-only semantic matcher for video-failure descriptions.

Compare the meanings expressed by the prediction descriptions and reference descriptions. Use only the supplied description text. Do not infer additional facts from category labels, media, generation requirements, method identities, or outside knowledge.

Match two descriptions only when they refer to the same concrete failure. A shared topic, object, or similar wording is insufficient. Differences in wording alone do not prevent a match when the described failure is equivalent.

Produce a one-to-one matching. Each prediction index and each reference index may appear in at most one matched pair. Leave unsupported matches unassigned.

Use the reference descriptions as the matching targets for this metric. Determine textual correspondence without adding new defects or revising the supplied annotations.

Predictions: {`predictions_json`}

References: {`gold_json`}

Return only a JSON object with the following structure. Use the indices supplied in the input. If no pairs match, return an empty `matches` array.

```
{
  "matches": [
    {
      "prediction_index": 0,
      "gold_index": 0,
      "reason": "Brief explanation of equivalence."
    }
  ],
  "overall": "Brief summary."
}
```

F REPRESENTATIVE CASES

Figures 5–9 present twelve representative cases, one per page. Each includes the available reference assets, the complete substantive instruction in English, chronological full-frame video samples, and the annotated failure dimensions, fine-grained tags, and descriptions. Repeated instruction requirements are consolidated in translation.

Red boxes locate observed errors, and magnified insets show the same image regions in greater detail; blue boxes identify reference comparisons. Audio icons denote supplied audio references. Audio-related descriptions follow the original annotations: still frames alone do not establish audio-reference mismatches, pronunciation errors, or speaking assignments.

Replacing a fox while preserving its scene

Reference images





Instruction

Use Reference image 1 as the scene and composition reference and Reference image 2 as the replacement fox. Preserve the lighting and color settings of image 1. Use an overhead view and replace the fox in the initial scene with the fox from image 2. The replacement fox looks up at the camera in fear, then recoils backward and curls up to hide.

Keep the environment and composition unchanged. Use a low-saturation cinematic look with 70 mm IMAX film character, a wide aperture and lighting motivated by the scene. Retain only natural ambience and on-location sound effects: no music or subtitles. Keep any dialogue clear and background sound unobtrusive. Produce a 4K image with smooth, natural movement, realistic friction and physical behavior, and no distortion or structural artifacts.

Generated video



Error details



Reference fox



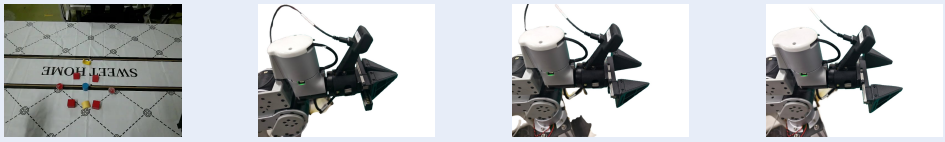
Original fox retained

Video editing • *Editing target failure*
The generated sequence retains the original fox rather than replacing it with the white fox shown in Reference image 2.

Figure 5: **Fox replacement.** The original fox remains despite the requested change of subject.

Grasping and holding a geometric block

Reference images



Instruction

Generate a 5 s video from the fixed overhead face camera in Reference image 1. Use a single continuous shot with no cuts, transitions, viewpoint changes, camera translation, rotation or zoom, and do not switch to either wrist camera. Preserve the tabletop layout, lighting, materials and background. Images 2–4 show the same robot gripper closed, half-open and open: two parallel black wedge-shaped fingers with diamond-pattern grip surfaces. Use these images only for gripper geometry, without importing their backgrounds, viewpoints or scale.

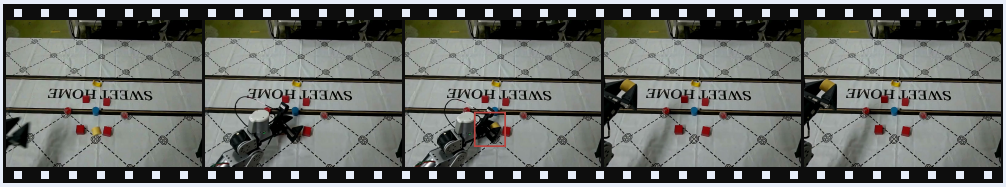
The scene contains a white tablecloth and triangular, hexagonal and square blocks. The two black wedges at the lower-left and lower-right corners are the ends of the left and right grippers; these are the only two grippers. Do not generate additional grippers or arms elsewhere. Both arms begin in their observation positions. Only the left arm may act. The right arm remains completely stationary at its initial lower-right position and must not descend, close, assist or move toward the center.

The target is a yellow triangular-prism block. Grasp its two sloping rectangular faces, not either triangular end or the bottom. Open the fingers wider than the target, lower them on either side and close fully around it. The block must remain stationary until firmly grasped; it must not stick to a fingertip or follow without contact. Keep it rigid with no deformation or sagging, show contact shadows, and move it as a single object with the gripper.

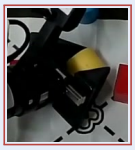
Show this failed operation: the left gripper rapidly descends over the two sloping faces, closes around the block and lifts it vertically by about half a body height. It then stops abruptly as if its control signal has been interrupted. The arm, gripper and block remain completely frozen in midair for most of the remaining video, without displacement, pose drift or shaking. Do not move toward the destination, lower or release the block, adjust the grasp, retract the arm or reset.

Depict only this failure, with no correction or completed placement. Keep the other objects, including the red square and blue circle, stationary and unchanged. Use natural, physically plausible motion before the halt, with photorealistic lighting and materials matching image 1. End with the block still held motionless in midair and not delivered to the target.

Generated video



Error details



Wrong grasped block

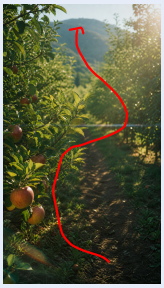
Spatial and action correctness • *Multi-character action mismatch*
The gripper lifts a semicylindrical block instead of the requested triangular prism.

Instruction following • *Missing plot steps*
Only the left arm appears; the required stationary right arm is absent.

Figure 6: **Robot grasping.** The gripper selects the wrong block and the required stationary arm is absent.

Following an FPV camera path

Reference image



Instruction

The red line and arrows in Reference image 1 indicate only the intended camera trajectory; remove these markings completely from the final video. Use a first-person FPV view for an extremely fast, cinematic, continuous shot that follows the marked route precisely, without deviating, jumping or simplifying the path. Show clear scene structure, coherent contours and pronounced surface textures. Use a realistic visual style with smooth, stable motion, a strong sense of speed and continuous, unambiguous space. Do not repeat the scenery. No red lines or arrow drawings may remain in the final video.

Generated video



Error details



Retained route marking

Instruction following • *Unrequested visual elements*
The red trajectory guide remains visible in the generated video despite the instruction to remove all route markings.

Figure 7: **FPV trajectory guidance.** The route marking remains visible in the generated scene.

Transferring a vehicle emblem

Instruction

Copy the Segway brand emblem from the front of the all-terrain vehicle in Reference video 2 onto the empty emblem position on the front of the vehicle in Source video 1. Match the reference emblem and leave all other content unchanged.

Source video



Reference video



Generated video



Error details



Reference emblem



Added lettering

Instruction following • *Unrequested visual elements*
The output adds lettering beneath the emblem even though the requested change was limited to transferring the reference logo.

Instruction following • *Missing plot steps*
The original video's audio is omitted from the edited result.

Figure 8: **Vehicle emblem transfer.** The edit adds unrequested lettering and omits the original audio.

Assigning dialogue to animal characters

Reference images



Reference audios

Instruction

Reference image 1 depicts the little gray mouse; Reference image 2, the little yellow mouse; Reference image 3, the little wild rabbit; Reference image 4, the red-haired squirrel; and Reference image 5, the dense forest behind the mountain in daylight. Use Reference audio 1 for the yellow mouse's voice and Reference audio 2 for the rabbit's voice.

Create the third shot as a 15-second sequence with six seconds of dialogue. Begin with a two-character medium shot of the gray and yellow mice for seven seconds, then track sideways and cut to a close-up of the rabbit and squirrel for eight seconds. Include self-satisfied mouse squeaks from 1 to 6 seconds and mischievous rabbit laughter from 8 to 11 seconds.

The gray and yellow mice stand side by side in the grass. The gray mouse waves its short forepaws while the yellow mouse raises its head, puts its paws on its hips, and speaks smugly: "If they dare come, they cannot blame Sister Qingsui for showing them no mercy." Cut to the rabbit and red-haired squirrel sitting together on a nearby rock and looking happily at each other. The rabbit grins with its split upper lip and gleefully blurts out: "Someone is going to be in trouble!"

Use fluid, continuous animation with detailed, photorealistic cinematic rendering, 8K resolution, natural soft lighting, balanced colors, clear light-and-shadow contrast, and carefully composed, expressive performances. Give every character rim lighting on its fur, fine non-oily surface detail, and soft facial-lighting transitions. Use a softly diffused, delicate image with foreground blur and shallow depth of field. Do not add subtitles or background music.

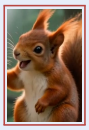
Generated video



Error details



Expected speaker: rabbit



Annotated speaker: squirrel

"Someone is going to be in trouble!"

Speech and audio • *Pronunciation / delivery errors*
The dialogue contains pronunciation errors.

Speech and audio • *Dialogue speaker mismatch*
The red-haired squirrel delivers "Someone is going to be in trouble!", although the instruction assigns this line to the little wild rabbit.

Figure 9: **Dialogue assignment.** The squirrel delivers the rabbit's line; the annotation also records pronunciation errors.

G VIDEO DEBUG CASE ANALYSIS

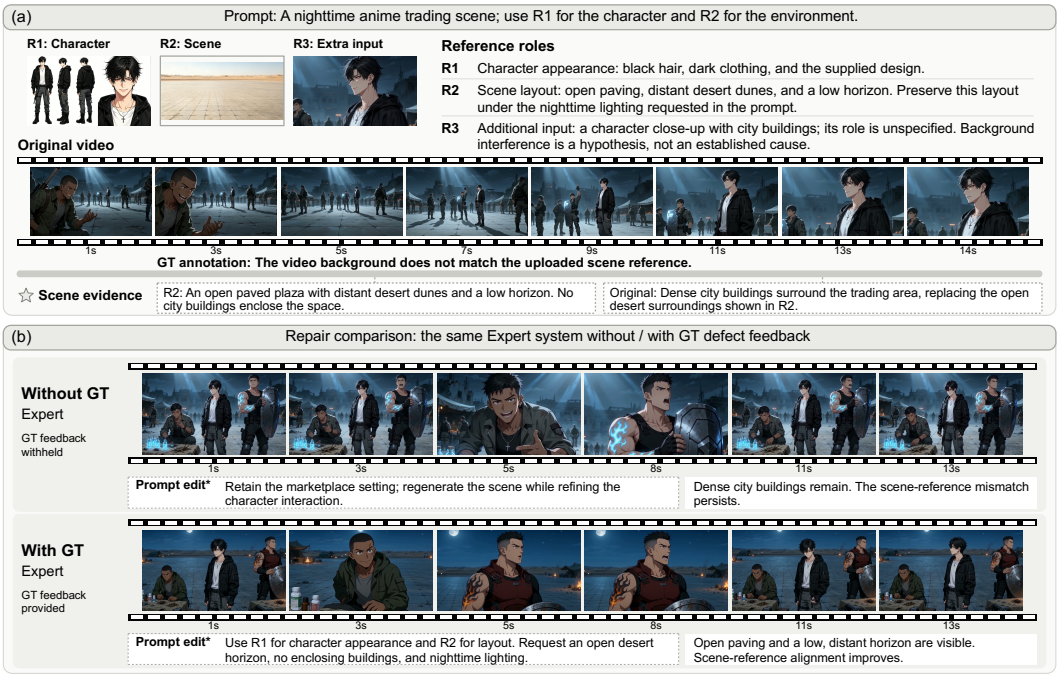


Figure 10: **Case study of diagnosis-guided video debugging.** Without GT defect feedback, the scene-reference mismatch persists. With GT feedback, the repair better follows R1 for character appearance and R2 for scene layout.