REFACTOR: REFLECTION-AUGMENTED AGENTIC VIDEO GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Narrative video generation requires coordinating multiple tools and adapting subsequent operations to intermediate outputs. Existing approaches typically rely on fixed pipelines or one-shot plans, which often fail when generated outputs deviate from expectations and lack mechanisms for autonomous recovery. We propose ReFactor, a multi-turn agentic framework that formulates narrative video generation as a Markov decision process: At each step, the agent selects a tool call conditioned on the narrative prompt, execution history, and generated artifacts to maximize narrative completeness. Since video generation requires multiple sequential tool calls without informative feedback, the ability to reflection becomes crucial for agents to inspect generated artifacts, diagnose failures, and guide corrective actions. To enhance the agent's reflection ability, we construct 21K reflection-augmented trajectories grounded in actual tool executions and enriched with evaluator-accepted repair demonstrations for SFT. We further develop an agentic RL approach that introduces a reflection-anchored process reward to measure the relative quality improvements from intermediate repairs. This provides denser supervision for learning when and how to reflect across tasks of varying difficulties. On StoryEval, ReFactor achieves an 18.9% relative improvement in narrative completion over the strongest baseline. Beyond this gain, RL makes the policy learn selective reflection, cutting reflection frequency on easy prompts from 0.32 to 0.13 while raising it on hard prompts from 0.46 to 0.61 relative to the SFT policy, with scorer-defined repair success improving in both groups. Code is available at: https://anonymous.4open.science/r/Refactor-6BC2

1 INTRODUCTION

Recent large-scale video generation models have advanced short-clip fidelity and controllability He et al. (2025a); Zheng et al. (2024), but narrative video generation requires scene transitions and coherent action sequences that exceed the capability of a single model Guo et al. (2025b); Dalal et al. (2025); Song et al. (2026). Meeting these demands requires coordinating multiple specialized tools and real-time self-correction Long et al. (2024); Oh et al. (2024). Therefore, narrative video generation is not a one-shot generation problem, but a sequential decision-making process that requires selecting subsequent tools based on the actual outputs of prior executions.

This calls for a multi-turn agent with closed-loop decision-making: it observes generated visual artifacts (images or videos), adapts subsequent tool calls to realized outcomes, and learns from execution feedback. Realizing this goal raises three challenges. First, the tool set is diverse and task complexity varies dramatically, so adaptive control requires reasoning over a combinatorially large action space Ruan et al. (2023); Polyak et al. (2024) that rigid pipelines cannot cover Li et al. (2025). Second, stochastic generation can produce unexpected visual failures Yuan et al. (2025) that become observable only after execution, necessitating post-hoc evaluation and correction. Third, while reflection enables recovery from visual failures, it also incurs additional reasoning tokens and tool executions Chen et al. (2025); Yu et al. (2026). Effective agents must therefore learn to reflect selectively, adapting corrective effort to task difficulty and observed failures.

Existing approaches to narrative video generation can be broadly grouped into three categories. *End-to-end generation models* directly map prompts to videos, but provide no explicit mechanism for inspecting intermediate artifacts or correcting failures through subsequent tool calls Peebles

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

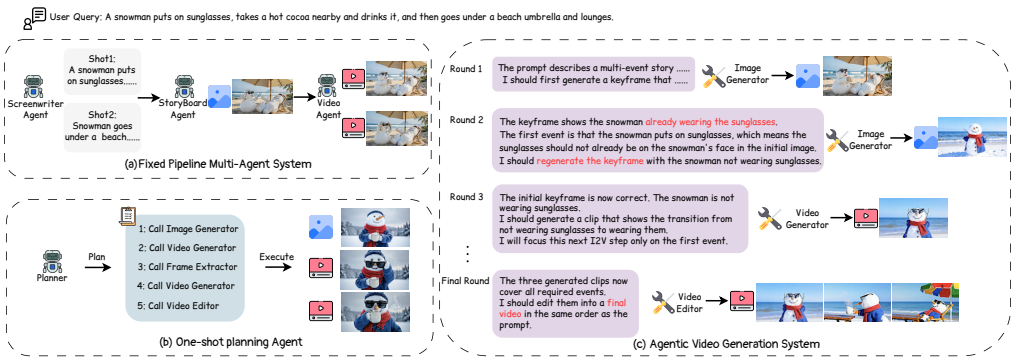


Figure 1: **Three agent paradigms for video generation**. (a) Fixed-pipeline multi-agent systems decompose the task into predetermined stages; some include rule-based quality checks, but the control logic is hand-designed. (b) One-shot planning agents generate the full tool-call sequence before observing any image or video, then execute blindly. (c) Agentic video generation (ReFactor): at each round the policy observes the latest image or video, produces a reasoning trace (purple boxes), and selects the next tool accordingly. When the generated keyframe contains a defect, the agent autonomously diagnoses the issue and invokes the appropriate tool to regenerate, continuing until all narrative events are covered and composed into the final video.

& Xie (2023); Brooks et al. (2024). *Fixed-pipeline multi-agent systems* Hu et al. (2024); Huang et al. (2026); Wang et al. (2026) decompose video creation into predefined stages. Although some incorporate critique-and-revision loops Long et al. (2025), their control logic and reflection criteria remain hand-designed, limiting their ability to learn when and how to correct failures from execution feedback. *Learning-driven planners* Zhang et al. (2026) optimize tool planning through supervised fine-tuning or preference optimization, but generate the full execution plan before observing any intermediate artifacts, preventing subsequent tool choices from adapting to realized outputs.

To this end, we propose **ReFactor**, a multi-turn, **ReF**lection-augmented **a**genti**c** video generat**i**on framew**or**k. ReFactor mimics human video creators by introducing tool use, visual memory, and post-hoc reflection. We formulate this process as an MDP, where the agent selects tool calls based on the prompt, execution history, and generated artifacts to maximize narrative completeness. Visual memory preserves narrative context and supports artifact inspection, enabling the agent to diagnose failures and guide corrective actions. We first train the agent in tool use and reflection through SFT on 21K reflection-augmented trajectories grounded in actual tool executions and enriched with evaluator-accepted repair demonstrations. While SFT teaches the agent how to correct failures, deciding when correction is worthwhile requires learning from its own execution outcomes. We therefore introduce an agentic RL approach that combines final rewards with a reflection-anchored process reward to measure relative quality improvements from repairs. This provides denser feedback on corrective actions, encouraging selective and effective reflection across tasks of varying difficulty. Our main contributions are summarized as follows:

1. **Agentic video generation framework.** Unlike fixed pipelines, ReFactor treats video generation as sequential decision-making, enabling an agent to inspect intermediate artifacts, adapt tool choices to execution outcomes, and autonomously initiate corrections.
2. **Reflection-augmented agentic video generation dataset.** We construct 21K narrative video generation trajectories grounded in actual tool executions and enriched with evaluator-accepted repair demonstrations. These trajectories can be used in SFT to provide basic tool use and reflection capabilities for narrative video generation agents.
3. **Agentic RL with reflection-anchored process reward.** We introduce a reflection-anchored process reward that measures relative quality improvements from intermediate repairs, providing denser supervision alongside final rewards. This reward mechanism avoids directly rewarding reflection count and encourages selective and adaptive reflection behaviors.
4. **Comprehensive evaluation and analysis.** We evaluate ReFactor through benchmark comparisons, ablation studies, and analyses of reflection behavior and generation efficiency. Results on StoryEval show an 18.9% relative improvement in narrative completion over the strongest baseline and demonstrate adaptive reflection across task difficulty levels.

2 Related Works

Model-Based Video Generation. End-to-end models, evolving from GAN-based Tulyakov et al. (2018); Saito & Saito (2018) and latent diffusion architectures Ho et al. (2022); He et al. (2022) to recent DiT-based systems Peebles & Xie (2023); Ma et al. (2024); Brooks et al. (2024) augmented with controllability plugins Chen et al. (2023); Ni et al. (2024); Henschel et al. (2025), have achieved strong short-clip fidelity. However, their stochastic single-pass nature precludes targeted error correction and the cross-scene narrative consistency required for long-form storytelling.

Agent-Based Video Generation. Agent-based systems have emerged as a promising paradigm for long-form narrative video generation, offering modularity and controllable workflows. Fixed pipeline multi-agent frameworks build hierarchical pipelines mimicking film production Hu et al. (2024); Wu et al. (2025b); Zheng et al.; Liu et al. (2025); Yuan et al. (2024); Huang et al. (2026); Wang et al. (2026), and some incorporate test-time critique loops for quality control Long et al. (2025). However, their control logic and reflection criteria are hand-designed and frozen; they cannot learn from execution feedback to improve over time. A separate line trains agents for tool planning via SFT or preference optimization Zhang et al. (2026), but these methods output the full execution in one forward pass without observing intermediate artifacts. In contrast, ReFactor treats reflection as a policy behavior learned from trajectories and execution feedback, rather than as a fixed, hand-designed checking rule.

RL for Tool-Use Agents. Reinforcement learning, from RLHF Ouyang et al. (2022) to recent verifiable-reward methods such as GRPO Shao et al. (2024); Guo et al. (2025a), has become a key paradigm for content generation agents. Outcome-reward training has been applied to search agents Jin et al. (2025); Wu et al. (2025a) and hierarchical multi-tool collaboration Dong et al. (2025), while process-reward modeling has shown that step-level supervision can improve long-horizon reasoning Lightman et al. (2024). For multimodal generation agents, intermediate tool decisions must instead be evaluated through the stochastic visual artifacts they produce. We bridge this gap by repurposing the agent's self-triggered reflection steps as process-level reward anchors, providing denser supervision in long-horizon multimodal trajectories.

3 Preliminaries

3.1 MLLM-Based Tool-Use Agents

Modern tool-use agents are built on multimodal large language models (MLLMs) that interact with external tools through a unified text generation interface Qin et al. (2023); Schick et al. (2023). We adopt the ReAct paradigm Yao et al. (2022), in which the agent interleaves explicit reasoning with acting at every decision step. Concretely, the agent operates on a multimodal context comprising the task instruction, prior dialogue turns, and visual observations. At each timestep, it first generates a reasoning trace z_t that analyzes the current situation and plans the next move. Based on z_t, the agent then produces a structured action a_t, which is either a tool call or a natural-language response. The environment executes the action and returns the result, which is concatenated with the existing context to form the updated observation for the next timestep. This cycle repeats until the task is complete. This interleaved reasoning-acting loop forms the basis of our agent architecture.

3.2 Group Relative Policy Optimization

We adopt Group Relative Policy Optimization (GRPO) Shao et al. (2024) as the RL algorithm for policy optimization. Unlike PPO Schulman et al. (2017), which requires training a separate value network to estimate baselines, GRPO computes advantages from the relative performance within a group of rollouts sampled from the current policy, eliminating the need for a critic model. Let $R(\tau)$ denote the scalar return of trajectory τ (defined in Sec. 4.4). The advantage of τ_i is estimated by normalizing within the group:

$$\hat{A}_i = \frac{R(\tau_i) - \mathrm{mean}(\{R(\tau_j)\}_{j=1}^{G})}{\mathrm{std}(\{R(\tau_j)\}_{j=1}^{G})}. \tag{1}$$

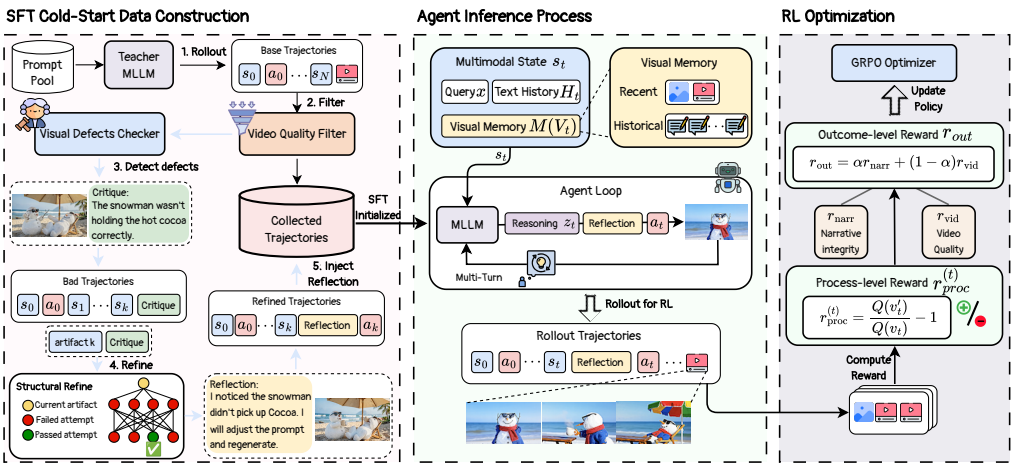


Figure 2: **Overview of the ReFactor**, consisting of reflection-augmented SFT data construction (left), multi-turn agent inference (middle), and RL optimization (right). **Left:** A teacher model generates candidate trajectories, which are filtered by visual quality and defect checkers; surviving trajectories are further augmented with reflection–repair episodes to provide cold-start supervision. **Middle:** Given a narrative prompt, the policy iteratively selects tools, inspects generated artifacts, and triggers post-hoc reflection to repair detected defects. **Right:** Outcome rewards score final narrative completeness and video quality. Process rewards, anchored at reflection steps, measure immediate artifact improvement. Together they provide denser credit assignment across long trajectories.

The policy is then updated by maximizing the clipped surrogate objective with a KL penalty:

$$\mathcal{L}_{\text{GRPO}}(\theta) = \mathbb{E}\big[\min\big(r_t \hat{A}_i,\, \bar{r}_t \hat{A}_i\big)\big] - \lambda D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}}), \tag{2}$$

4 Methodology

Motivation. Narrative video generation requires flexible tool use to coordinate generation steps and reflection to diagnose and correct visual failures. Fixed pipelines and one-shot plans limit the ability to adapt these decisions to intermediate outputs. We introduce an agentic framework that inspects generated artifacts and autonomously selects subsequent tool calls. To develop its tool-use and reflection capabilities, we train the agent through reflection-augmented SFT, followed by RL with feedback on repair quality to encourage selective and effective correction.

Overview. As illustrated in Figure 2, **ReFactor** combines multi-tool video generation, structured visual memory, and policy training. The policy inspects the latest artifact directly, uses captions of earlier outputs to retain narrative context, and decides whether to continue generation or repair an observed defect (Secs. 4.1–4.2). We initialize these behaviors with SFT trajectories containing evaluator-accepted repairs found through budgeted search (Sec. 4.3). During RL, each agent-triggered repair pairs the original and repaired image or clip; modality-specific scorers measure their relative quality change to provide local feedback alongside the final-video reward (Sec. 4.4).

4.1 Multi-Tool Video Generation as an MDP

In ReFactor, we formulate the multi-tool video generation process as a Markov Decision Process (MDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma)$.

Tool set and artifacts. The agent has access to five tool classes: (1) a *text-to-image* generator for keyframe synthesis; (2) a *text-to-video* generator for prompt-conditioned clip generation; (3) an *image-to-video* generator for reference-frame-conditioned clip generation; (4) a *video editor* for temporal stitching of clips; and (5) a *frame extractor* for sampling frames from existing clips. Each tool call produces a visual output (an image or video clip), which we refer to as an artifact. We denote the ordered set of artifacts up to step t as $\mathcal{V}_t = \{v_1, \ldots, v_t\}$.

State. At step t, the agent observes the user instruction, the textual execution history $\mathcal{H}_t$ (reasoning traces, tool calls, and tool responses), and the artifacts $\mathcal{V}_t$. The state is defined as $s_t = (x,\, \mathcal{H}_t,\, \mathcal{M}(\mathcal{V}_t))$, where x is the user query and $\mathcal{M}(\cdot)$ is the structured visual memory operator introduced in Section 4.2.

Action. Following the ReAct paradigm (Sec. 3.1), the policy generates a reasoning trace z_t followed by a structured tool call a_t at each step:

$$z_t \sim \pi_\theta(z \mid s_t), \qquad\qquad a_t \sim \pi_\theta(a \mid s_t, z_t).$$

The policy output at each step is therefore a reasoning–tool-call pair:

$$(z_t, a_t) = \begin{cases} \big(z_t^{\text{plan}},\ \textsc{ToolCall}(\textit{name}, \textit{args})\big), \\ \big(z_t^{\text{refl}},\ \textsc{ToolCall}(\textit{name}, \textit{args})\big). \end{cases} \tag{3}$$

A reflection step is a regular ReAct step where the reasoning trace z_t^{refl} identifies the visual issues, and the following tool call acts to correct them.

Transition. Executing a_t appends the resulting artifact to $\mathcal{V}_t$ and the tool response to $\mathcal{H}_t$, advancing the state stochastically due to the inherent randomness of multimodal generation.

Reward. The reward function combines outcome-level assessment of the final video with process-level rewards assigned at reflection steps. We describe the full reward design in Section 4.4.

4.2 Post-Hoc Reflection with Structured Visual Memory

Generic agent reflection is insufficient for video generation since effective correction requires reasoning over long multimodal histories while preserving fine-grained visual information.

Structured visual memory. Storing all artifacts as raw visual inputs exhausts the context window, yet text-only history loses the visual details needed for reflection. We introduce a structured visual memory operator $\mathcal{M}$ that maintains artifacts in two forms: the most recent artifact is preserved as a raw visual input so the agent can directly inspect fine-grained details such as identity drift or prompt mismatch; earlier artifacts are compressed into VLM-generated captions that compactly preserve narrative state: $\mathcal{M}(\mathcal{V}_t) = \{\mathrm{C}(v_i) \mid i \le t{-}1\} \cup \{v_i \mid i > t{-}1\}$, where $\mathrm{C}(\cdot)$ is a VLM-based captioning function. This two-tier design balances perceptual grounding for immediate decisions with context efficiency over long trajectories.

Reflection mechanism. This design separates post-hoc diagnosis from pre-execution planning: the agent first observes the realized artifact and then decides whether correction is necessary. Unlike pre-execution reflection that critiques a plan before observing its outcome, our post-hoc reflection is triggered after a visual artifact v_t has been generated and observed. The agent may either accept it and proceed, or enter a reflection step to correct defects. In a reflection step, the agent generates reflective reasoning z_t^{refl} that diagnoses the defect, then emits a corrective tool call a_t^{refl}:

$$z_t^{\text{refl}} \sim \pi_\theta(\cdot \mid s_t, v_t), \qquad\qquad a_t^{\text{refl}} \sim \pi_\theta(\cdot \mid s_t, v_t, z_t^{\text{refl}}). \tag{4}$$

Executing a_t^{refl} yields a corrected artifact v_t' that is appended to the visual history and becomes the newest artifact inspected in subsequent steps. The preceding failed v_t remains in both the visual-memory history and the trajectory record as grounded evidence for later decisions.

4.3 SFT with Reflection-Augmented Trajectories

Although the pretrained MLLM backbone possesses general reasoning and instruction-following abilities, it lacks task-specific knowledge of video generation tool use and reflection-based correction. Direct RL from scratch is costly due to the combinatorially large action space and the high cost of tool execution at each step. We therefore construct a reflection-augmented SFT dataset of approximately 21K agent-facing trajectories to initialize both sequential tool use and recoverable post-hoc correction, following the same agent architecture defined in Sec. 4.1–4.2.

Base trajectories. For each prompt in a curated pool covering single-scene to multi-event narratives, a strong MLLM teacher interacts with the same tool environment as the student to produce a multi-step trajectory $\tau = (s_0, z_0, a_0, \ldots, s_N)$, where each a_i is actually executed against the underlying tools and each s_{i+1} contains the resulting artifact. Full details are provided in Appendix I.

Reflection injection. Base trajectories rarely contain failure recovery. We restrict reflection injection to intermediate artifacts that exhibit high overall visual quality yet contain identifiable defects (e.g., prompt mismatch or subject inconsistency); videos that are either already satisfactory or severely degraded beyond recovery are excluded, ensuring the injected reflections teach the agent to correct realistic, recoverable errors. For every intermediate artifact v_t, a VLM-based evaluator outputs a binary failure label and a diagnostic critique $(y_t, \rho_t) = \textsc{Eval}(x, s_t, v_t)$, $y_t \in \{0, 1\}$. Steps with $y_t = 1$ become reflection injection points. For each failure, we run an iterative repair search that proposes multiple candidate revisions per round and accumulates unsuccessful critiques as feedback for the next round; the full procedure is described in Appendix I.2. Once a candidate is accepted by the evaluator ($\hat{y} = 0$), we splice the reflection block back into the trajectory as $\hat{\tau} = (\ldots, s_t, \hat{z}_t^{\text{refl}}, \hat{a}_t, \hat{v}_t, s'_{t+1})$. The agent only sees the reflective reasoning trace and revised tool call, not the external critique, so SFT remains consistent with the inference-time ReAct interface.

4.4 Agentic RL with Reflection-Anchored Process Reward

SFT teaches the agent how to reflect and repair visual failures, but it does not fully determine when reflection is necessary or which corrective action is most effective. Since these decisions depend on the stochastic artifacts encountered during execution, we use RL to learn selective and effective reflection from on-policy experience. We further introduce a reflection-anchored process reward that directly measures whether each repair improves the generated artifact, providing explicit feedback for learning better reflection behavior.

Outcome reward. At trajectory end, the final video is scored by external evaluators on narrative completeness r_{narr} and visual quality r_{vis}, combined as $r_{\text{out}} = \alpha r_{\text{narr}} + (1 - \alpha) r_{\text{vis}}$, with both components rescaled to $[0, 1]$.

Reflection-Anchored Process reward. For each reflection step, let v_t and v'_t denote the artifacts before and after the revised tool call, and $Q(\cdot)$ the artifact-specific quality scorer (an image-quality Huang et al. (2024) for image artifacts, VideoScore2.0 He et al. (2025b) for video artifacts; both rescaled to $(0, 0.2]$ and floored at $Q_{\min} = 0.05$). Thus, the scores used in the ratio lie in $[0.05, 0.2]$. The relative-improvement reward is

$$r_{\text{proc}}^{(t)} = \text{clip}\Big(\tfrac{Q(v'_t)}{Q(v_t)} - 1,\ -0.2,\ 0.2\Big). \tag{5}$$

The relative form is scale-invariant with respect to the floored score Q, expressing image- and video-quality changes as fractional improvements (Prop. 1, Appendix K.2).

Total return. We combine the outcome reward with the averaged reflection-anchored process reward to define the trajectory-level return:

$$R(\tau) = r_{\text{out}} + \frac{\beta}{\max(|\mathcal{T}_{\text{refl}}|, 1)} \sum_{t \in \mathcal{T}_{\text{refl}}} r_{\text{proc}}^{(t)}, \tag{6}$$

where $\mathcal{T}_{\text{refl}}$ is the set of reflection steps and β weights the averaged process term $P(\tau)$. Averaging keeps the process term bounded independently of reflection count. With the existing rewards held fixed, an additional reflection raises $P(\tau)$ only if its reward exceeds the current average (Prop. 2, Appendix K.3). The SFT-initialized policy is optimized with GRPO (Sec. 3.2). Under signal alignment and residual informativeness, Theorem 1 shows a local SNR improvement for sufficiently small positive process weight. We use $\alpha = 0.9$ and $\beta = 0.2$; Appendix B reports the evaluated reward weights and the corresponding scores on StoryEval.

5 Experiments

We evaluate ReFactor on StoryEval and VideoScore2.0, organized around four research questions.

- **RQ1 (Performance):** Does the complete ReFactor framework, combining agent architecture and policy training, outperform existing approaches in narrative completion?
- **RQ2 (Ablation):** How do the agent architecture and successive training stages contribute to ReFactor's performance?
- **RQ3 (Reflection):** Does RL lead to more selective and effective reflection across difficulty levels?
- **RQ4 (Efficiency):** How does ReFactor trade off narrative completion against generation cost?

Table 1: Main results on StoryEval. Event-level story completion scores on a 0–100 scale across seven categories and the overall average. Parentheses denote the controller (MLLM backbone). ReFactor Arc. denotes the training-free ReFactor architecture; ReFactor denotes the full framework. **Bold** indicates the best result and underlining the second best within each column.

Method	Human	Animal	Object	Retrieval	Creative	Easy	Hard	Avg.
Video Generation Models								
HunyuanVideo1.5	27.0	27.5	19.2	33.9	13.3	41.6	12.3	24.9
Closed-source Controller Models								
ReActAgent (Gemini-3-Flash)	50.4	53.6	43.3	54.6	44.1	66.6	30.3	48.5
StoryAgent (Gemini-3-Flash)	44.6	46.1	39.1	51.1	32.8	66.4	21.3	43.6
ReFactor Arc. (Gemini-3-Flash)	**64.7**	**62.6**	**60.8**	**58.8**	**58.4**	**72.6**	**47.6**	**62.8**
Rel. Improvement	+28.4%	+16.8%	+40.4%	+7.7%	+32.4%	+9.0%	+57.1%	+29.5%
Open-source Controller Models								
ReActAgent (Qwen3.5-9B)	32.3	37.2	30.1	42.2	26.4	51.9	18.0	33.6
ReActAgent (Qwen3.5-27B)	31.3	36.9	32.7	45.3	25.0	52.3	16.4	34.3
StoryAgent (Qwen3.5-9B)	41.0	43.4	37.9	47.1	33.0	61.6	20.6	41.1
StoryAgent (Qwen3.5-27B)	53.8	54.6	49.9	50.1	49.2	67.3	37.0	53.1
MultimediaAgent (Qwen3.5-9B)	58.1	55.4	53.1	57.2	48.2	72.4	37.0	54.5
ReFactor (Qwen3.5-9B)	**65.8**	**68.4**	**60.6**	**71.6**	**58.9**	**81.3**	**48.7**	**64.8**
Rel. Improvement	+13.3%	+23.5%	+14.1%	+25.2%	+22.2%	+12.3%	+31.6%	+18.9%

5.1 Experimental Setup

Benchmark. We evaluate on StoryEval Wang et al. (2025), a benchmark of 423 prompts across 7 story categories, each describing a 2–4 event narrative. The benchmark spans diverse subjects and interaction patterns, enabling controlled assessment of long-horizon narrative completion across both Easy and Hard prompts. All prompts are disjoint from our SFT and RL training data. Implementation details are provided in Appendix L.

Evaluation Metrics. StoryEval Wang et al. (2025) measures narrative completeness on a 0–100 scale. The training-time narrative reward is scored by Gemini-3.1-Pro, while the test-time StoryEval judge is GPT-5.4 following the official protocol, using different model families for reward and evaluation. Evaluation details, a consistency audit, and cross-validation with human preferences are provided in Appendix D. VideoScore2.0 He et al. (2025b) scores videos on visual quality (VQ), text alignment (TA), and physical consistency (PC), providing a complementary signal on generation quality.

Baselines. We compare four paradigms: (1) direct text-to-video generation (HunyuanVideo1.5); (2) generic ReAct tool-use agents (Qwen3.5-9B/27B, Gemini-3-Flash); (3) fixed multi-agent pipelines following StoryAgent Hu et al. (2024); and (4) learned tool planners (MultimediaAgent Zhang et al. (2026), trained on our base trajectories). All tool-using baselines share the same tool set and maximum step budget as ReFactor. Our StoryAgent-style implementation includes observer-triggered critique and regeneration and averages 4.00 generation calls with the Qwen3.5-27B controller (Table 5). It provides a comparable-call-count critique baseline, in the same broad family as VISTA Long et al. (2025). Appendix H.1 and H.2 detail implementations and fairness controls.

5.2 Main Results (**RQ1**)

Training-free architecture. Using Gemini-3-Flash as the backbone without any task-specific training, the ReFactor architecture achieves a StoryEval Avg. score of 62.8, surpassing the strongest closed-source baseline by +29.5%. This comparison evaluates the agent architecture—reflection mechanism and structured visual memory, applied on top of the same backbone and tool set. Gains are consistent across all seven story categories, with the largest improvements on Hard prompts (+57.1%). Results with a different video generator appear in Appendix C.

Complete trained framework. The trained 9B ReFactor agent outperforms all open-source baselines, exceeding the strongest by +18.9% on StoryEval, with gains across all seven categories and the largest improvement on Hard prompts (+31.6%; Table 1). Table 2 shows the largest gain over direct generation in text alignment, with visual quality and physical consistency preserved. Together,

Table 2: Video quality comparison using VideoScore2.0. VQ, TA, and PC denote visual quality, text alignment, and physical consistency, respectively. Relative-improvement rows compare each configuration with HunyuanVideo1.5.

Method	Visual Quality	Text Alignment	Physical Consistency	Overall
HunyuanVideo1.5	3.538	2.932	2.928	3.133
ReFactor Arc. (Gemini-3-Flash)	3.567	3.482	2.946	3.332
Rel. Improvement	+0.8%	+18.8%	+0.6%	+6.4%
ReFactor (Qwen3.5-9B)	**3.764**	**3.522**	**3.109**	**3.465**
Rel. Improvement	+6.4%	+20.1%	+6.2%	+10.6%

Table 3: Cumulative ablation on StoryEval and VideoScore2.0. Δ_{SE} and Δ_{VS} denote cumulative relative improvement over the base agent on StoryEval Avg. and VideoScore2.0 Overall, respectively.

#	Variant	StoryEval			Δ_{SE}	VideoScore2.0				Δ_{VS}
		Easy	Hard	Avg.		VQ	TA	PC	Overall	
1	Base agent	51.9	18.0	33.6	–	3.297	3.147	2.711	3.052	–
2	+ Reflection & visual memory (training-free)	62.3	33.1	50.2	+49.4%	3.463	3.356	2.837	3.219	+5.5%
3	+ SFT (no refl. injection)	72.7	36.4	52.9	+57.4%	3.611	3.502	3.020	3.378	+10.7%
4	+ Reflection injection in SFT	76.7	38.1	54.4	+61.9%	3.591	3.510	3.058	3.387	+11.0%
5	+ RL (outcome only)	80.1	42.8	60.0	+78.6%	3.652	3.526	2.993	3.390	+11.1%
6	+ RL (**Full**)	81.3	48.7	64.8	+92.9%	3.764	3.522	3.109	3.465	+13.5%

these results support combining multi-turn agentic control with reflection-augmented SFT and RL to improve narrative completion, particularly for complex prompts.

Human preferences favor ReFactor in both narrative coherence and visual quality (Appendix E). Paired bootstrap intervals for StoryEval are reported in Appendix G.

Qualitative examples. Appendix A and Figure 4 provide three prompt-level comparisons against MultimediaAgent and HunyuanVideo1.5. Across all three, ReFactor completes the full event sequence with higher visual quality, higher subject consistency and more natural transitions.

5.3 Ablation Studies (**RQ2**)

Table 3 reports cumulative ablation results. The agent architecture alone (row 2) yields +49.4% on StoryEval Avg. without any task-specific training. SFT (rows 3–4) teaches reliable tool use and reflection, raising the cumulative gain to +61.9%. RL adapts the policy to its stochastic outputs through on-policy experience (rows 5–6), reaching +78.6% with outcome rewards alone and +92.9% after adding the reflection-anchored process reward. The process reward's gain is concentrated on Hard prompts (42.8→48.7, +13.8%). Across the full stack, the gap between Easy and Hard categories narrows from 2.9× to 1.7×. The cumulative design measures gains in this ordering rather than independent component effects.

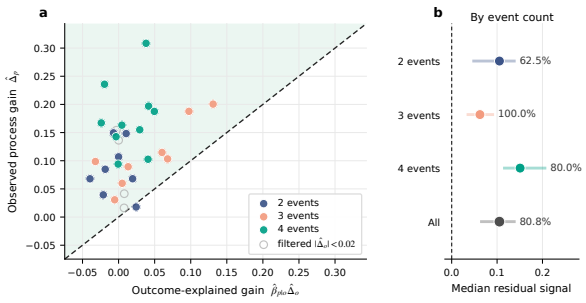


Figure 3: **The process reward carries information beyond the outcome reward.** Each prompt contributes a residual process-reward difference between RL and SFT after accounting for its outcome-reward difference (Appendix K). The largest median residual is observed on four-event prompts.

Why does the process reward help on Hard prompts? Hard prompts require more complex events, yielding longer trajectories with multiple stochastic generation steps between reflection and the final outcome. A later, unrelated failure at step $t+2$ can obscure a useful repair at step t in the noisy, delayed outcome reward, weakening the gradient signal for that repair. The process reward scores each reflection by comparing the artifact before and after repair, providing immediate local credit unaffected by downstream noise. We formalize this intuition and show (Thm. 1, Appendix K) that when the local improvement carries information beyond the outcome reward, adding a sufficiently small positive process term strictly improves the GRPO advantage SNR in the local comparison. Figure 3 reports a residual-signal diagnostic, with the largest median residual on 4-event prompts.

5.4 Reflection Behavior and Generation Cost (**RQ3–RQ4**)

Table 4: Reported reflection behavior before and after RL. Entries show SFT → RL. Success means $Q(v_t') > Q(v_t)$ under the artifact scorer used in training.

Metric	Easy (SFT → RL)	Hard (SFT → RL)
Avg. reflections	$0.32 \rightarrow 0.13$ (-59.4%)	$0.46 \rightarrow 0.61$ ($+33\%$)
Episodes with 0 reflections	$70\% \rightarrow 90\%$	$58\% \rightarrow 42\%$
Reflection success rate	$53\% \rightarrow 78\%$	$43\% \rightarrow 64\%$

Table 5: Generation calls and narrative completion. Calls count T2I and T2V/I2V invocations per prompt; calls/point divides this count by StoryEval.

Method	Gen. calls	StoryEval	Calls/point
ReActAgent (Qwen3.5-9B)	4.40	33.6	0.131
StoryAgent (Qwen3.5-27B)	4.00	53.1	0.075
MultimediaAgent (Qwen3.5-9B)	4.16	54.5	0.076
ReFactor (Qwen3.5-9B, SFT)	4.61	54.4	0.085
ReFactor (Qwen3.5-9B, RL)	4.53	64.8	**0.070**

Reflection selectivity (RQ3). Table 4 reports SFT-to-RL behavior on the 423 StoryEval prompts, grouped by the benchmark's Easy/Hard labels. RL reduces average reflection frequency on Easy prompts from 0.32 to 0.13, while increasing it on Hard prompts from 0.46 to 0.61. The reported fraction of episodes without reflection rises to 90% on Easy and falls to 42% on Hard. Scorer-defined success increases by 25 percentage points on Easy prompts and 21 points on Hard prompts. On Easy prompts, RL raises StoryEval from 76.7 to 81.3 while reducing reflection frequency. Together, these results suggest that RL enables the agent to reflect selectively, adapting corrective effort to narrative difficulty while increasing repair success rates in both groups. This direction is predicted by the return rather than imposed by the difficulty labels. Because the process term is averaged over reflection steps, an additional reflection raises $P(\tau)$ only if its improvement exceeds the running average (Prop. 2). On Easy prompts the first-pass artifact is usually adequate, so a marginal repair typically falls below that average and the gradient discourages it; on Hard prompts the first-pass artifact is more often defective, so the same term rewards another attempt. The policy therefore reallocates correction effort without any explicit difficulty signal.

Generation cost (RQ4). Table 5 counts T2I and T2V/I2V invocations over the same 423 prompts and reports average generation calls per prompt. ReFactor uses 4.53 calls versus 4.16 for MultimediaAgent, an 8.9% increase, while StoryEval rises by 18.9%. It has the lowest calls-per-point ratio among the listed systems. This ratio is a descriptive trade-off. Within ReFactor, RL lowers generation calls from 4.61 to 4.53 while increasing StoryEval from 54.4 to 64.8.

6 Conclusion

Narrative video generation is a long-horizon decision problem in which subsequent tool choices must depend on generated visual outputs. We present ReFactor, an agent framework trained through reflection-augmented SFT and reflection-aware RL, with reflection steps serving as process-level reward anchors. By inspecting outputs and adapting subsequent decisions, ReFactor improves narrative completeness and video quality, particularly on complex multi-event prompts, demonstrating the value of learned closed-loop decision-making for narrative video generation.

AI USE STATEMENT

Generative AI assistants were used for literature search and polish writing. The authors take responsibility for the final content of the paper. The method itself uses generative models for trajectory construction, artifact captioning, and reward evaluation, as described in the paper.

ETHICS STATEMENT

ReFactor may lower barriers to producing misleading synthetic videos and inherit biases or unsafe behaviors from its underlying models and data. We apply prompt safety filtering during data construction (Appendix I.1). External datasets, models, and tools are used under their applicable licenses or terms; generated artifacts are restricted to research use and will be released only where those terms permit.

REPRODUCIBILITY STATEMENT

Appendix I describes prompt curation and repair injection; Appendix J gives the inference algorithm; and Appendix L specifies models and training parameters. Evaluation protocols and bootstrap procedures appear in Appendices D–G, and reward assumptions and proofs in Appendix K. Baseline implementations and comparison controls are detailed in Appendix H. Source code is included in the supplementary material and will be publicly released upon acceptance.

REFERENCES

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Leo Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, et al. Video generation models as world simulators. *OpenAI Blog*, 1(8):1, 2024.

Weifeng Chen, Yatai Ji, Jie Wu, Hefeng Wu, Pan Xie, Jiashi Li, Xin Xia, Xuefeng Xiao, and Liang Lin. Control-a-video: Controllable text-to-video diffusion models with motion prior and reward feedback learning. *arXiv preprint arXiv:2305.13840*, 2023.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, et al. Do not think that much for 2+ 3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*, 2025.

Karan Dalal, Daniel Koceja, Jiarui Xu, Yue Zhao, Shihao Han, Ka Chun Cheung, Jan Kautz, Yejin Choi, Yu Sun, and Xiaolong Wang. One-minute video generation with test-time training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 17702–17711, 2025.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *arXiv preprint arXiv:2505.16410*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025a.

Yuwei Guo, Ceyuan Yang, Ziyan Yang, Zhibei Ma, Zhijie Lin, Zhenheng Yang, Dahua Lin, and Lu Jiang. Long context tuning for video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17281–17291, 2025b.

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. *arXiv preprint arXiv:2601.03233*, 2025.

Hao He, Yinghao Xu, Yuwei Guo, Gordon Wetzstein, Bo Dai, Hongsheng Li, and Ceyuan Yang. Cameractrl: Enabling camera control for video diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Xuan He, Dongfu Jiang, Ping Nie, Minghao Liu, Zhengxuan Jiang, Mingyi Su, Wentao Ma, Junru Lin, Chun Ye, Yi Lu, et al. Videoscore2: Think before you score in generative video evaluation. *arXiv preprint arXiv:2509.22799*, 2025b.

Yingqing He, Tianyu Yang, Yong Zhang, Ying Shan, and Qifeng Chen. Latent video diffusion models for high-fidelity long video generation. *arXiv preprint arXiv:2211.13221*, 2022.

Roberto Henschel, Levon Khachatryan, Hayk Poghosyan, Daniil Hayrapetyan, Vahram Tadevosyan, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Streamingt2v: Consistent, dynamic, and extendable long video generation from text. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 2568–2577, 2025.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

Panwen Hu, Jin Jiang, Jianqi Chen, Mingfei Han, Shengcai Liao, Xiaojun Chang, and Xiaodan Liang. Storyagent: Customized storytelling video generation via multi-agent collaboration. *arXiv preprint arXiv:2411.04925*, 2024.

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*, volume 40, pp. 5049–5057, 2026.

Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, Yaohui Wang, Xinyuan Chen, Limin Wang, Dahua Lin, Yu Qiao, and Ziwei Liu. VBench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Xuan Ju, Yiming Gao, Zhaoyang Zhang, Ziyang Yuan, Xintao Wang, Ailing Zeng, Yu Xiong, Qiang Xu, and Ying Shan. Miradata: A large-scale video dataset with long durations and structured captions. *Advances in Neural Information Processing Systems*, 37:48955–48970, 2024.

Black Forest Labs. FLUX.2: Frontier Visual Intelligence. `https://bfl.ai/blog/flux-2`, 2025.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. *arXiv preprint arXiv:2510.05592*, 2025.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Jingwei Liu, Ling Yang, Hao Luo, Fan Wang, Hongyan Li, and Mengdi Wang. Preacher: Paper-to-video agentic system. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 17129–17139, 2025.

Do Xuan Long, Xingchen Wan, Hootan Nakhost, Chen-Yu Lee, Tomas Pfister, and Sercan Ö Arık. Vista: A test-time self-improving video generation agent. *arXiv preprint arXiv:2510.15831*, 2025.

Fuchen Long, Zhaofan Qiu, Ting Yao, and Tao Mei. Videostudio: Generating consistent-content and multi-scene videos. In *European Conference on Computer Vision*, pp. 468–485. Springer, 2024.

Xin Ma, Yaohui Wang, Xinyuan Chen, Gengyun Jia, Ziwei Liu, Yuan-Fang Li, Cunjian Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024.

Kepan Nan, Rui Xie, Penghao Zhou, Tiehan Fan, Zhenheng Yang, Zhijie Chen, Xiang Li, Jian Yang, and Ying Tai. Openvid-1m: A large-scale high-quality dataset for text-to-video generation. In *The Thirteenth International Conference on Learning Representations*.

Haomiao Ni, Bernhard Egger, Suhas Lohit, Anoop Cherian, Ye Wang, Toshiaki Koike-Akino, Sharon X Huang, and Tim K Marks. Ti2v-zero: Zero-shot image conditioning for text-to-video diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9015–9025, 2024.

Gyeongrok Oh, Jaehwan Jeong, Sieun Kim, Wonmin Byeon, Jinkyu Kim, Sungwoong Kim, and Sangpil Kim. Mevg: Multi-event video generation with text-to-video models. In *European Conference on computer vision*, pp. 401–418. Springer, 2024.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Adam Polyak, Amit Zohar, Andrew Brown, Andros Tjandra, Animesh Sinha, Ann Lee, Apoorv Vyas, Bowen Shi, Chih-Yao Ma, Ching-Yao Chuang, et al. Movie gen: A cast of media foundation models. *arXiv preprint arXiv:2410.13720*, 2024.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*, 2023.

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

Jingqing Ruan, Yihong Chen, Bin Zhang, Zhiwei Xu, Tianpeng Bao, Hangyu Mao, Ziyue Li, Xingyu Zeng, Rui Zhao, et al. Tptu: Task planning and tool usage of large language model-based ai agents. In *NeurIPS 2023 foundation models for decision making workshop*, 2023.

Masaki Saito and Shunta Saito. Tganv2: Efficient training of large models for video generation with multiple subsampling layers. *arXiv preprint arXiv:1811.09245*, 2(6), 2018.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yale Song, Yiwen Song, Nick Losier, Nathan Hodson, Ye Jin, Rhyard Zhu, Yan Xu, Daniel Vlasic, Carina Claassen, Jasmine Leon, et al. Co-director: Agentic generative video storytelling. *arXiv preprint arXiv:2604.24842*, 2026.

Tencent Hunyuan Foundation Model Team. Hunyuanvideo 1.5 technical report, 2025. URL `https://arxiv.org/abs/2511.18870`.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535, 2018.

Qian Wang, Ziqi Huang, Ruoxi Jia, Paul Debevec, and Ning Yu. Mavis: A multi-agent framework for long-sequence video storytelling. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2273–2295, 2026.

Wenhao Wang and Yi Yang. Videoufo: A million-scale user-focused dataset for text-to-video generation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Wenhao Wang and Yi Yang. Vidprom: A million-scale real prompt-gallery dataset for text-to-video diffusion models. *Advances in Neural Information Processing Systems*, 37:65618–65642, 2024.

Yiping Wang, Xuehai He, Kuan Wang, Luyao Ma, Jianwei Yang, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Is your world simulator a good story presenter? a consecutive events-based benchmark for future long video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13629–13638, 2025.

Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. *arXiv preprint arXiv:2506.20670*, 2025a.

Weijia Wu, Zeyu Zhu, and Mike Zheng Shou. Automated movie generation via multi-agent cot planning. *arXiv preprint arXiv:2503.07314*, 2025b.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2022.

Zewei Yu, Lirong Gao, Yuke Zhu, Bo Zheng, Junbo Zhao, Sheng Guo, and Haobo Wang. Stop unnecessary reflection: Training lrms for efficient reasoning with adaptive reflection and length coordinated penalty. *arXiv preprint arXiv:2602.12113*, 2026.

Shenghai Yuan, Jinfa Huang, Xianyi He, Yunyang Ge, Yujun Shi, Liuhan Chen, Jiebo Luo, and Li Yuan. Identity-preserving text-to-video generation by frequency decomposition. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 12978–12988, 2025.

Zhengqing Yuan, Yixin Liu, Yihan Cao, Weixiang Sun, Haolong Jia, Ruoxi Chen, Zhaoxu Li, Bin Lin, Li Yuan, Lifang He, et al. Mora: Enabling generalist video generation via a multi-agent framework. *arXiv preprint arXiv:2403.13248*, 2024.

Daoan Zhang, Wenlin Yao, Xiaoyang Wang, Yebowen Hu, Jiebo Luo, and Dong Yu. A versatile multimodal agent for multimedia content generation. *arXiv preprint arXiv:2601.03250*, 2026.

WEI Zheng, Mingchen Li, Zeqian Zhang, Ruibin Yuan, Pan Hui, Huamin Qu, James Evans, Maneesh Agrawala, and Anyi Rao. Omniagent: Long-video generation via cross-modal multi-agent orchestration.

Zangwei Zheng, Xiangyu Peng, Tianji Yang, Chenhui Shen, Shenggui Li, Hongxin Liu, Yukun Zhou, Tianyi Li, and Yang You. Open-sora: Democratizing efficient video production for all. *arXiv preprint arXiv:2412.20404*, 2024.

A Qualitative Analysis

We compare ReFactor against MultimediaAgent (the strongest agent baseline) and HunyuanVideo1.5 (the strongest single-call baseline) on three representative narrative prompts. Figure 4 presents sampled frames for each system; we analyze event completion and visual quality below.



Figure 4: **Qualitative comparison on three narrative prompts.** Each column corresponds to one prompt; rows from top to bottom show outputs from ReFactor, MultimediaAgent, and HunyuanVideo1.5, respectively. ReFactor completes all specified events in every case, whereas the two baselines each miss at least one key event (see text for details).

Case 1. Prompt: "A person stretches out a tape measure, measures the distance from the table to the lamp, and then records it on the papers on the table." ReFactor completes all three events in the correct order: stretching the tape measure, measuring the table-to-lamp distance, and recording the result on paper. HunyuanVideo1.5 measures the table's length rather than the table-to-lamp distance, and the final recording event never occurs. MultimediaAgent does depict a measuring action, but the tape measure is placed between the table edge and the lamp base rather than spanning the correct distance; the recording event is present but imprecise.

Case 2. Prompt: "A painter dips a brush into water, paints a rainbow, and then the rainbow becomes real." ReFactor realizes all three events, including the causal transition from a painted rainbow to a physically present one in the scene. HunyuanVideo1.5 shows only the painting action on paper, missing both the brush-dipping event and the final transformation where the rainbow becomes real. MultimediaAgent includes the brush-dipping step, but the rainbow remains a painting and never transforms into a real one.

Case 3. Prompt: "A person writes in a diary, the words float off the page, and then form constellations in the sky." ReFactor completes the full sequence: writing, words lifting off the page, and the words transforming into constellations, with a smooth and visually coherent transition. HunyuanVideo1.5 generates only the diary-writing event; the words neither float nor form constellations. MultimediaAgent produces the writing and the final constellation, but omits the intermediate floating transition, making the words appear as constellations abruptly without the specified detachment from the page.

These three examples illustrate that ReFactor's advantage extends beyond isolated event completion. ReFactor achieves both higher event fidelity and more natural cross-event transitions than the baselines.

B Reward-Weight Sensitivity

Table 6 reports reward-weight sensitivity. At $\alpha = 0.9$, narrative completion receives most of the outcome weight. The $\alpha = 0.7$ setting has a lower StoryEval score (63.3 versus 64.8) and a higher VideoScore (3.52 versus 3.47), illustrating the trade-off between the two objectives.

Increasing β from 0 to 0.2 improves the Hard score by 5.9 points and the Easy score by 1.2 points. At $\beta = 0.5$, both scores are below their $\beta = 0.2$ values. The results favor 0.2 among the tested settings.

Table 6: Reward-weight sensitivity at the evaluated settings. Left: narrative versus visual outcome weight. Right: process reward weight.

α	StoryEval	VideoScore
0.7	63.3	3.52
0.9	64.8	3.47

β	Easy	Hard	Avg.
0.0	80.1	42.8	60.0
0.2	81.3	48.7	64.8
0.5	79.5	48.0	63.2

Table 7: Generalization to LTX2.3. StoryEval scores are reported on a 0–100 scale. All agent methods use Gemini-3-Flash as the controller and LTX2.3 as the generation tool. ReFactor Arc. denotes the training-free ReFactor architecture.

Method	Easy	Hard	Avg.
LTX2.3	55.5	14.2	34.8
ReActAgent (Gemini-3-Flash)	66.8	25.0	44.6
StoryAgent (Gemini-3-Flash)	76.8	41.3	58.6
ReFactor Arc. (Gemini-3-Flash)	**76.8**	**51.6**	**63.8**
Rel. Improvement	0.0%	+24.9%	+8.9%

C Tool Generalization

To verify that ReFactor's closed-loop control is not specific to a single video backbone, we replace HunyuanVideo1.5 with LTX2.3 HaCohen et al. (2025) as the underlying video generation tool while keeping the reflection mechanism and all other components unchanged. All agent methods use Gemini-3-Flash without task-specific SFT or RL. Table 7 reports StoryEval results for all three agent paradigms under LTX2.3. ReFactor achieves a StoryEval Avg. score of 63.8, outperforming the strongest non-ReFactor agent by +8.9% and direct LTX2.3 generation by +83.3%. Notably, ReFactor matches StoryAgent on Easy prompts (76.8) while improving the Hard score from 41.3 to 51.6 (+24.9%), indicating that its advantage is concentrated on more complex narratives. This pattern is consistent with our motivation: when subsequent tool choices must respond to intermediate generation outcomes, closed-loop control becomes increasingly valuable. These results confirm that ReFactor's training-free architecture generalizes across video generation backends without architecture-specific adaptation.

D StoryEval Consistency and Validity Verification

Implementation details. StoryEval evaluates whether each specified event in a narrative prompt is completed by the generated video. Following the official protocol Wang et al. (2025), we use GPT-5.4 as the automatic evaluator. For every prompt–video pair, we query the evaluator independently three times for each event and obtain three binary judgements. We use a conservative aggregation rule: an event is counted as completed only when all three judgements are positive. The reported StoryEval metric is then the average completion rate over all events. This all-pass rule is designed to reduce false positives, especially for ambiguous cases where an action is only partially visible or temporally incomplete.

Consistency audit. To verify the stability of the evaluator, we randomly sample 50 prompt–video pairs from the ReFactor-Qwen3.5-9B evaluation outputs, covering all seven reported StoryEval categories. For each event in these pairs, we run the same evaluator three independent times with the identical event-level evaluation prompt. This produces 142 event-level cases and 426 binary judgements in total. Table 8 summarises the agreement statistics. The three repeated judgements reach complete agreement on 90.1% of event-level cases, with an average pairwise agreement of 93.4% and Fleiss' $\kappa = 0.86$, indicating strong inter-run consistency. At the prompt level, the mean standard deviation of the repeated completion scores is 4.6 points, with a median of 0.0 points, confirming that score variance is low.

Table 8: StoryEval evaluator consistency statistics on 50 sampled prompt–video pairs.

Statistic	Value
Prompt–video pairs sampled	50
Event-level cases	142
Binary judgements	426
Complete agreement (event-level)	90.1%
Average pairwise agreement	93.4%
Fleiss' κ	0.86
Mean prompt-level std. dev.	4.6 points
Median prompt-level std. dev.	0.0 points

Consistency. Disagreements are concentrated in borderline cases, such as heavily occluded actions, short-lived object interactions, or videos that depict the correct objects but fail to show the requested temporal transition clearly. These are precisely the cases where the strict all-three-pass rule makes the metric conservative: uncertain events are treated as incomplete rather than credited as successes. The high repeated-judgement agreement, low prompt-level variance, and conservative aggregation together indicate that the VLM-as-judge protocol provides a repeatable automatic signal for event-level narrative completion on the sampled ReFactor outputs.

Cross-validation with human judgement. As an additional safeguard, we compare the automatic metric with human preferences. The human evaluation (Appendix E) independently confirms ReFactor's advantage over MultimediaAgent: on 100 StoryEval prompts, human annotators prefer ReFactor in 63% of narrative coherence judgements, consistent with the direction of the automatic StoryEval gap.

E Human Evaluation

To complement the automatic metrics, we conduct a human evaluation study comparing ReFactor-Qwen3.5-9B against MultimediaAgent-Qwen3.5-9B.

Protocol. We randomly sample 100 prompts from the StoryEval test set, stratified by difficulty (50 Easy, 50 Hard). For each prompt, three annotators are independently shown the output videos of both systems in randomized order and asked to select the best video along two dimensions: narrative coherence (whether the video faithfully depicts all specified events in the correct order) and visual quality (perceptual fidelity, motion plausibility, and cross-scene consistency). Annotators may also select "tie" if no system is clearly superior. All annotators are volunteer researchers familiar with video generation, participate without compensation, and are blind to system identities. The instructions ask annotators to compare two anonymized videos in randomized order on the two dimensions above, selecting either video or "tie." The task involves viewing synthetic videos only and presents no foreseeable risk beyond ordinary screen viewing.

Table 9: **Human preference rates (%) on 100 StoryEval prompts.** Each cell reports the percentage of annotations preferring the corresponding system.

Dimension	ReFactor	MultimediaAgent-9B	Tie
Narrative Coherence	**63**	19	18
Visual Quality	**41**	23	36
Overall Preference	**57**	25	18

Results. Table 9 reports the preference rates aggregated across annotators. ReFactor is preferred in 63% of narrative coherence judgements, against 19% for the baseline, demonstrating a clear advantage in story completion. Visual quality is less decisive—ReFactor leads 41% to 23%, but 36% of cases are judged as ties, consistent with both systems relying on the same underlying generation model. The narrative advantage is more pronounced on Hard prompts (68% preference) than Easy prompts (57%), consistent with the automatic metrics. Inter-annotator agreement (Fleiss' κ) is 0.61 for narrative coherence and 0.54 for visual quality, indicating moderate-to-substantial agreement.

These results confirm that ReFactor's gains on automatic metrics translate to perceptible quality improvements as judged by human evaluators.

F HUMAN–AUTOMATIC METRIC CORRELATION

To verify that StoryEval tracks human judgement of narrative completion, we recruit three annotators to independently judge event-level completion on all 423 StoryEval prompts for all compared systems, following the same per-event binary protocol as the automatic evaluator. We then compute the Pearson correlation between the per-prompt automatic completion rates and the averaged human completion rates. The Pearson correlation coefficient is $r = 0.9785$, indicating a near-linear relationship between the two evaluation signals. This high correlation supports StoryEval Avg. as a proxy for human judgement of narrative completion.

G PAIRED BOOTSTRAP CONFIDENCE INTERVALS

To assess whether ReFactor's gains are robust to prompt-level sampling variability, we report 95% paired bootstrap confidence intervals (CIs) for the difference in StoryEval Avg. between ReFactor and the strongest non-ReFactor baseline in each controller block. Absolute differences are reported in StoryEval points.

Procedure. For each of the 423 StoryEval prompts, we retain the per-prompt average event-completion rate for both systems. We then draw 10,000 bootstrap replicates: in each replicate, we sample 423 prompt indices with replacement, compute the mean completion rate of each system on the resampled set, and record the difference $\Delta^{(b)} = \bar{S}^{(b)}_{\text{ReFactor}} - \bar{S}^{(b)}_{\text{baseline}}$. The 95% CI is the 2.5th–97.5th percentile interval of the 10,000 $\Delta^{(b)}$ values. This procedure requires no additional inference, only the already-collected prompt-level scores are resampled.

Table 10: Paired bootstrap 95% CIs for the difference in StoryEval Avg. between ReFactor and the strongest non-ReFactor baseline in each block. Differences are reported in StoryEval points. A CI that excludes zero indicates the improvement is robust to prompt-level sampling uncertainty. ReFactor Arc. denotes the training-free ReFactor architecture.

Block	**Comparison**	**Δ (points)**	**95% CI (points)**	$p<0.05$
Closed-source	ReFactor Arc. (Gemini-3-Flash) − ReActAgent(Gemini-3-Flash)	+14.3	[11.2, 17.4]	✓
Open-source	ReFactor(Qwen3.5-9B) − MultimediaAgent(Qwen3.5-9B)	+10.3	[7.4, 13.2]	✓

Interpretation. Both reported intervals exclude zero under prompt-level resampling. They quantify uncertainty from sampling these prompts, conditional on the collected model outputs and judge decisions; they do not cover training or generation-seed variability.

H BASELINE IMPLEMENTATIONS AND COMPARISON FAIRNESS

H.1 BASELINE IMPLEMENTATIONS

We instantiate baselines to cover the main alternatives to closed-loop learned tool control. All tool-using baselines operate over the same tool set as ReFactor and use the same unified JSON tool interface.

Direct text-to-video model. HunyuanVideo1.5 is evaluated as a direct generation baseline. Given the user query, it produces the final video with a single prompt-conditioned text-to-video call, without explicit keyframe generation, intermediate frame inspection, tool planning, or multi-step correction.

Generic ReAct tool-use agents. The generic agent baselines use the same ReAct-style reasoning–tool-call interface as ReFactor: at each step, the controller emits a reasoning trace followed by a structured tool call. These agents can invoke the text-to-image generator, text-to-video/image-to-video generator, video editor, and frame extractor. However, they are training-free and remove the two mechanisms introduced by ReFactor: they do not have an explicit post-hoc reflection action for diagnosing and correcting failed artifacts, and they do not use the structured visual memory that

preserves the latest artifact as visual input while compressing earlier artifacts into long-term captions. Thus, this baseline isolates whether simply giving an MLLM access to the same tools and ReAct interface is sufficient for narrative video generation.

StoryAgent-style multi-agent pipelines. StoryAgent Hu et al. (2024) is a representative training-free multi-agent framework for storytelling video generation. It decomposes production into five specialized roles: a story designer that expands the user prompt into per-shot descriptions, a storyboard generator that produces keyframe images aligned with each shot description, a *video creator* that animates each keyframe into a clip via I2V generation, an agent manager that orchestrates execution order, and an observer that evaluates intermediate results and requests re-generation if quality criteria are not met. We faithfully implement all five roles using our common tool environment, the same T2I generator, I2V generator, frame extractor, and video editor available to all baselines, so that the comparison isolates control strategy rather than generator capability. Concretely, for each prompt the pipeline executes: (1) story design via the LLM controller, producing shot-level descriptions; (2) storyboard generation by calling the T2I tool for each shot; (3) per-shot video generation via I2V; (4) observer-triggered quality review with at most one re-generation attempt per shot; and (5) final clip stitching via the video editor. Crucially, the execution order across these stages is fixed in advance: the pipeline does not autonomously decide mid-execution whether to revisit earlier stages, alter the shot decomposition, or skip steps based on the actual visual content of intermediate artifacts.

MultimediaAgent-style learned tool planner. MultimediaAgent Zhang et al. (2026) represents the learning-driven multimedia-agent paradigm: it trains an agent to select and compose generation tools from plan data rather than relying only on hand-written prompts. For a domain-matched and fair instantiation, we train a Qwen3.5-9B controller on the same base trajectory dataset used in our SFT construction (Appendix I) before reflection injection. These base trajectories contain executed tool calls and real tool outputs, giving MultimediaAgent the same task-domain tool-use supervision before ReFactor-specific reflection augmentation. Unlike ReFactor, this baseline does not receive reflection-injected repair examples, structured visual-memory supervision, or reflection-aware RL rewards. At inference time, it uses the same JSON tool interface and tool set to generate and execute tool-use actions, but it follows the learned planning policy without ReFactor's explicit post-hoc reflection action.

H.2 Comparison Fairness

We control generation tools and maximum action budget. Training exposure, memory, and actual resource use differ across configurations as detailed below.

Identical tool access. All tool-using baseline systems—generic ReAct agents, StoryAgent-style pipelines, and MultimediaAgent-style learned planners, have access to the exact same set of tools as ReFactor: the same T2I generator (FLUX.2[klein]4B), the same T2V/I2V generator (Hunyuan-Video1.5), the same video editor, and the same frame extractor. No baseline is restricted from using any tool that ReFactor can access. The tool APIs, input/output schemas, and underlying model weights are identical across all systems.

Matched maximum inference budget. All multi-step tool-using baselines operate under the same maximum step budget of $T = 12$ tool calls per episode, which is sufficient for the most complex prompts in StoryEval (4-event narratives typically require 6–8 generation steps). Importantly, baselines are not artificially constrained: ReAct agents, StoryAgent-style pipelines, and MultimediaAgent are free to use their full $T = 12$ budget in any way their control logic dictates, including retry or regeneration if their design supports it. Table 5 reports actual generation calls, excluding controller, captioner, and utility overhead.

The learning-based MultimediaAgent baseline is trained on our base trajectory dataset, the same pre-reflection data source from which ReFactor's SFT set is constructed. This controls for access to domain-specific tool-use demonstrations and prevents ReFactor's gains from being attributed merely to seeing narrative-video trajectories during training. Only ReFactor uses the additional reflection-injected trajectories and the subsequent RL stage, which are the training contributions evaluated in this paper.

Architecture and training differences. The training-free comparison evaluates explicit reflection and structured visual memory together. StoryAgent also inspects outputs through its observer, but

its production stages and retry rule are fixed. The complete ReFactor system additionally receives reflection-injected supervision and RL. Its comparison with MultimediaAgent therefore evaluates the full recipe rather than independently identifying the effect of architecture, data, or RL.

Additional backbone baselines. To further stress-test fairness, we instantiate the StoryAgent-style pipeline with Qwen3.5-27B and Gemini-3-Flash controllers. ReFactor remains competitive with or outperforms these baselines.

I SFT Dataset Construction Details

This appendix provides the full construction details of the SFT cold-start dataset summarized in Sec. 4.3. Section I.1 describes how the prompt pool is curated and how the teacher generates base trajectories. Section I.2 details the iterative repair algorithm used to inject reflection behaviors into these trajectories.

I.1 Prompt Pool and Teacher Configuration

Prompt sources. We construct the prompt pool from four public text-to-video prompt and caption datasets, chosen to balance realistic user queries with descriptive captions of varying length and detail: **VidProM** Wang & Yang (2024), the first million-scale dataset of real user prompts collected from text-to-video services, which best reflects the distribution of human-authored generation requests but contains substantial low-quality and unsafe content that requires aggressive cleaning; **MiraData** Ju et al. (2024), a 470k-prompt corpus where each entry includes a short caption, a dense caption, and structured background/object/style/camera captions; **VideoUFO** Wang & Yang, a topic-organized 1.09M-prompt corpus that pairs every entry with both a brief and a detailed caption; and **VIDGEN-1M** Nan et al., a million-scale caption dataset that provides additional diversity in scene composition and visual style. Of these, VidProM contributes the user-query-style backbone of our pool, while the remaining three contribute short and medium-length captions to broaden coverage of scene and detail descriptions.

Curation pipeline. The raw prompts are processed through a multi-stage curation pipeline designed around the heterogeneity of these sources. **(i) Safety filtering.** For VidProM, we apply both the dataset's own NSFW/toxicity flags and an additional LLM-based safety classifier to remove unsafe or borderline prompts. The other three datasets, being caption-derived, are subjected only to the LLM classifier. **(ii) Length filtering.** We retain prompts of 15–120 words, which excludes underspecified one-liners as well as overly long prompts that no current text-to-video model can faithfully render. **(iii) Text cleaning.** We strip residual HTML, repeated punctuation, encoding artifacts, and other noise that is particularly prevalent in scraped user-prompt data. **(iv) Temporal-cue preservation.** Prompts containing explicit temporal connectives (*then*, *after*, *while*, *before*, *finally*, etc.) are preferentially retained, since they are the most likely to specify the multi-event structure that closed-loop decision-making is designed for. **(v) Deduplication.** We embed each surviving prompt with a sentence encoder and remove near-duplicates with high cosine similarity. **(vi) Source-aware mixing.** The final pool is sampled from the four sources at fixed proportions: 75% from VidProM as the user-query backbone, 10% from VideoUFO brief captions and 10% from MiraData short captions for short-form diversity, and 5% from VIDGEN-1M for medium-length scene descriptions. The final prompt pool contains approximately 21K prompts.

Teacher configuration. We use Gemini-3-Flash as the teacher MLLM. Crucially, the teacher is not asked to imagine a tool-use trajectory in a single forward pass; instead, it interacts with the same tool environment as the student agent. At each step, the teacher observes the current state s_t, emits a reasoning trace and a structured tool call, the call is actually executed against the underlying generators, and the resulting visual artifact is fed back into the next state. This guarantees that every artifact v_i in a recorded trajectory is a real tool output rather than a hypothetical description, which is essential for two reasons: (1) downstream reflection injection can only be evaluated meaningfully on real artifacts; (2) the student agent, after SFT, generalizes to actual stochastic generators rather than to teacher-imagined idealized outputs. Teacher decoding uses a temperature of 0.7, a maximum trajectory length of $N_{\max} = 10$ steps. We discard trajectories that fail to produce a final video.

Base trajectory statistics. After teacher rollout and timeout filtering, we obtain approximately 21K base trajectories in total. The mean trajectory length is $\bar{N} = 7.1$ steps, with a per-prompt average of 1.5 T2I calls, 3.2 T2V/I2V calls, 1.0 video-edit/compose calls, and 1.4 frame-extraction calls.

Quality control on base trajectories. Not every teacher trajectory is suitable for SFT. We rank all base trajectories by the VideoScore2.0 overall score of their final video and discard the bottom 10%. This filter targets trajectories whose final outputs fall well below the typical quality of the teacher distribution—usually caused by catastrophic tool failures, prompt-incompatible execution choices, or unrecoverable error accumulation across multiple steps. Such trajectories provide weak supervision: their later steps condition on already-degraded artifacts, so the agent would learn planning behaviors that lead to unrecoverable videos regardless of subsequent reflection. Removing the bottom 10% filters out these cases while keeping trajectories with intermediate imperfections that reflection is designed to handle.

I.2 Iterative Repair Algorithm

The reflection injection stage augments base trajectories with synthetic correction events at every detected failure point. This subsection details the per-failure repair search.

Failure detection. For every intermediate artifact v_t in a (post-filter) base trajectory, we run a VLM-based evaluator that returns

$$(y_t, \rho_t) = \textsc{Eval}(x, s_t, v_t), \quad y_t \in \{0, 1\}. \tag{7}$$

$y_t = 1$ indicates that v_t is flawed, and ρ_t describes the observed problem in natural language. The evaluator is prompted to focus on four canonical failure modes common in stochastic video generation: identity drift, unnatural motion, prompt mismatch, and cross-scene inconsistency. It grounds its critique in specific frames or regions when possible. Steps with $y_t = 1$ are designated as repair points, and the corresponding critique initializes the repair context $\mathcal{P}_0 = \{\rho_t\}$.

Iterative repair search. At repair round $m \in \{1, \dots, M\}$, the teacher conditions on $\mathcal{P}_{m-1}$ and samples N parallel candidate revisions

$$\{(z_{t,m}^{(n)},\ a_{t,m}^{(n)},\ \hat{v}_{t,m}^{(n)})\}_{n=1}^{N}, \tag{8}$$

each consisting of a reflective reasoning trace, a revised tool call, and the regenerated artifact obtained by actually executing that call. Each candidate is judged by the same evaluator:

$$(y_{t,m}^{(n)},\ \rho_{t,m}^{(n)}) = \textsc{Eval}(x, s_t, \hat{v}_{t,m}^{(n)}). \tag{9}$$

If any $y_{t,m}^{(n)} = 0$, the corresponding candidate is accepted and the search terminates immediately. Otherwise, all N critiques from round m are appended:

$$\mathcal{P}_m = \mathcal{P}_{m-1} \cup \{(\hat{v}_{t,m}^{(n)},\ \rho_{t,m}^{(n)})\}_{n=1}^{N}, \tag{10}$$

so that the next round is conditioned on a richer description of which revision strategies have already failed and why. This failure-mode accumulation is important. With it, later rounds are pushed away from previously failed strategies.

We use $M = 3$ rounds and $N = 4$ candidates per round, yielding up to 12 candidates per failure point. If no candidate succeeds within this budget, the failure point is discarded rather than spliced with a non-improving repair. By discarding them, we ensure that every reflection block in the SFT data corresponds to a repair accepted by the defect evaluator.

Algorithm 1 summarizes the full procedure.

Trajectory splicing. For each accepted repair, we splice the reflection block back into the original trajectory:

$$\hat{\tau} = (\dots, s_t,\ \hat{z}_t^{\text{refl}},\ \hat{a}_t,\ \hat{v}_t,\ s'_{t+1}, \dots), \tag{11}$$

where $\hat{z}_t^{\text{refl}}$ is the accepted reflective reasoning trace, $\hat{a}_t$ is the revised tool call, and $\hat{v}_t$ is the repaired artifact. The subsequent state s'_{t+1} is recomputed from the corrected artifact.

Agent-facing format. Although the external evaluator is used during data construction, the final SFT trajectory only exposes the agent-facing trajectory segment: the reflective reasoning trace and

Algorithm 1 Reflection-Augmented SFT Data Construction

Require: Prompt pool $\mathcal{X}$, teacher π_T, tools $\mathcal{E}$, evaluator EVAL
Require: VIDEOSCORE, rollout limit $N_{\max}$, rounds M, candidates N
Ensure: Agent-facing SFT dataset $\mathcal{D}_{\mathrm{SFT}}$

$\mathcal{B} \leftarrow \emptyset$, $\mathcal{D}_{\mathrm{SFT}} \leftarrow \emptyset$
for all $x \in \mathcal{X}$ **do**
  Roll out π_T in $\mathcal{E}$ for at most $N_{\max}$ steps to obtain τ
  if τ produces a final video **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{(x, \tau)\}$
  end if
end for
Remove the bottom 10% of $\mathcal{B}$ by final-video VIDEOSCORE
for all $(x, \tau) \in \mathcal{B}$ **do**
  $\hat{\tau} \leftarrow \tau$
  for all intermediate artifacts v_t in τ **do**
    $(y_t, \rho_t) \leftarrow \textsc{Eval}(x, s_t, v_t)$
    if $y_t = 0$ **then**
      continue
    end if
    $\mathcal{P} \leftarrow \{\rho_t\}$, accepted $\leftarrow$ FALSE
    for $m = 1$ to M **do**
      for $n = 1$ to N **do**
        Sample $(z, a) \sim \pi_T(\cdot \mid s_t, v_t, \mathcal{P})$
        Execute a in $\mathcal{E}$ to obtain $\hat{v}$
        $(\hat{y}, \hat{\rho}) \leftarrow \textsc{Eval}(x, s_t, \hat{v})$
        if $\hat{y} = 0$ **then**
          Splice $(z, a, \hat{v})$ into $\hat{\tau}$ at step t
          Recompute the next state as s'_{t+1}
          accepted $\leftarrow$ TRUE
          break
        else
          $\mathcal{P} \leftarrow \mathcal{P} \cup \{(\hat{v}, \hat{\rho})\}$
        end if
      end for
      if accepted **then**
        break
      end if
    end for
  end for
  Convert $\hat{\tau}$ to the agent-facing format
  $\mathcal{D}_{\mathrm{SFT}} \leftarrow \mathcal{D}_{\mathrm{SFT}} \cup \{\hat{\tau}\}$
end for
return $\mathcal{D}_{\mathrm{SFT}}$

the revised tool call. The evaluator critiques ρ_t and any failed candidate critiques $\rho_{t,m}^{(n)}$ are used purely as conditioning inputs for the teacher and never appear in the data the student is trained on. Consequently, the student learns reasoning-mediated correction grounded in the visible artifact v_t, without dependence on any external feedback signal at inference time.

Dataset statistics. The repair search succeeds for 39.3% of detected failure points within the $M \times N = 12$ budget; the remaining 60.7% are dropped under our "improvement-or-nothing" policy. The mean accepted repair round is $\bar{m} = 1.48$: 70.8% of successful repairs are found in the first round and 86.7% within the first two. The final dataset contains approximately 21K trajectories.

Why discarding unrepaired failures is appropriate. Manual inspection of a random sample of 50 discarded failure points reveals that the vast majority (82%) stem from limitations of the underlying generation tool. SFT retains accepted repair demonstrations, while RL exposes the policy to outcomes from its own decisions. The Easy/Hard behavior comparison in Section 5.4 characterizes the resulting selectivity.

Algorithm 2 ReFactor Inference with Post-Hoc Reflection

Require: Prompt x, trained policy π_θ, tools $\mathcal{E}$, memory operator $\mathcal{M}$, step budget T
Ensure: Final video $v^\star$, or failure if no video is produced

$\mathcal{H} \leftarrow \emptyset, \mathcal{V} \leftarrow \emptyset, v^\star \leftarrow \varnothing$
for $t = 0$ to $T - 1$ **do**
  $s_t \leftarrow (x, \mathcal{H}, \mathcal{M}(\mathcal{V}))$
  Sample $o_t \sim \pi_\theta(\cdot \mid s_t)$
  if $o_t = \text{FINISH}(v)$ and v is a video artifact in $\mathcal{V}$ **then**
    return v
  end if
  Parse $o_t = (z_t, a_t)$ and execute a_t in $\mathcal{E}$ to obtain v_{new}
  Append $(z_t, a_t, v_{\text{new}})$ to the trajectory record and update $\mathcal{H}$
  $\mathcal{V} \leftarrow \mathcal{V} \,\|\, v_{\text{new}}$
  if v_{new} is a candidate final video **then**
    $v^\star \leftarrow v_{\text{new}}$
  end if
end for
if $v^\star \neq \varnothing$ **then**
  return $v^\star$
else
  return failure
end if

J Inference-Time Closed-Loop Generation

At inference time, ReFactor executes a single policy in a closed loop with the tool environment. Unlike reflection injection during SFT construction, inference uses neither an external defect evaluator nor a quality scorer: after each tool output, the trained policy inspects the newest artifact through structured visual memory and chooses whether to continue generation, issue a reflective repair call, or terminate with a final video. We use $\text{FINISH}(v)$ below to denote a terminal decision that returns an already generated video artifact without executing another tool. Algorithm 2 summarizes the procedure.

The memory operator retains the most recently generated artifact as raw visual input and captions all earlier artifacts ($K = 1$ in our implementation), including artifacts that motivate a reflective correction. A repair result is appended as the newest observation; no generated artifact is overwritten. Under a fixed step budget T, a reflection call therefore incurs a direct inference cost and must trade off local correction against completing remaining narrative events.

K Formal Properties of Reflection-Anchored Rewards

This appendix justifies the three reward-design choices used in Sec. 4.4: relative improvement, averaging across reflection steps, and process anchors for credit assignment. We give notation and then state the corresponding scale, incentive, and SNR properties.

K.1 Setup and Assumptions

Let $\tau = (s_0, z_0, a_0, \ldots, s_N)$ be a trajectory generated by a policy π_θ in the MDP defined in Sec. 4.1, and let $\mathcal{T}_{\text{refl}} \subseteq \{0, \ldots, N-1\}$ index its reflection steps, with $k := |\mathcal{T}_{\text{refl}}|$. For $t \in \mathcal{T}_{\text{refl}}$, write v_t and v'_t for the artifacts before and after the reflective tool call, and denote the artifact-specific scorer by $Q(\cdot)$.

The trajectory return takes the form

$$R(\tau) = r_{\text{out}}(\tau) + \beta\, P(\tau), \qquad P(\tau) = \frac{1}{\max(k, 1)} \sum_{t \in \mathcal{T}_{\text{refl}}} r^{(t)}_{\text{proc}}, \tag{12}$$

with $r^{(t)}_{\text{proc}}$ as in Eq. (5).

Assumptions.

A1 Image scores and averaged VideoScore2.0 scores are rescaled to $(0, 0.2]$ and floored at $Q_{\min} = 0.05$; Q denotes the resulting positive score in $[0.05, 0.2]$.

A2 Process rewards are clipped to $[-0.2, 0.2]$, hence $|r^{(t)}_{\text{proc}}| \le 0.2$.

A3 Within a GRPO group, rollouts share the same prompt and sampling policy, giving well-defined within-group means, variances, and covariances for the SNR analysis.

A1–A3 are properties of the implemented scorer, clipping rule, and GRPO sampling protocol. The only substantive conditions used below are signal alignment and residual informativeness.

K.2 Property 1: Scale-Invariance and Distortion of the Relative-Improvement Reward

A common alternative to Eq. (5) is the additive form $r^{\text{abs}}_t = Q(v'_t) - Q(v_t)$. The relative form has three useful properties:

Proposition 1 (Properties of the relative form)**.** *Under A1–A2, the unclipped relative reward $\tilde{r}_t := Q(v'_t)/Q(v_t) - 1$ satisfies:*

(i) *(*Scale invariance.*) For any constant $c > 0$, the rescaled scorer $\tilde{Q} := cQ$ yields the same reward $\tilde{r}_t$. In contrast, r^{abs}_t scales linearly with c.*

(ii) *(*Cross-modal commensurability.*) Regardless of the absolute range of each scorer, $\tilde{r}_t$ is dimensionless and admits a uniform "fractional improvement" interpretation: for our image-quality scorer and VideoScore2.0 (both rescaled to $(0, 0.2]$ and floored at 0.05), $\tilde{r}_t = 0.5$ means a 50% relative gain in either modality. The additive form r^{abs}_t inherits the scorer's unit and scale.*

(iii) *(*First-order equivalence to a log-quality potential.*) Define $\Phi(v) := \log Q(v)$. Then $\tilde{r}_t = \exp(\Phi(v'_t) - \Phi(v_t)) - 1$, so $\operatorname{sign}(\tilde{r}_t) = \operatorname{sign}(\Phi(v'_t) - \Phi(v_t))$ and, for $|\tilde{r}_t| \to 0$, $\tilde{r}_t = \Phi(v'_t) - \Phi(v_t) + O(\tilde{r}_t^2)$.*

Proof sketch. (i)–(ii) follow because multiplicative scorer constants cancel in the ratio. (iii) follows from $\tilde{r}_t = \exp(\Phi(v'_t) - \Phi(v_t)) - 1$ and the expansion $\log(1 + x) = x + O(x^2)$. □

The log-potential equivalence is first-order, not exact for finite quality changes. The score floor bounds the log-quality difference, while clipping limits each process reward to $[-0.2, 0.2]$.

K.3 Property 2: Marginal-Incentive Structure of Averaging

Let $\bar{r}_k := \frac{1}{k}\sum_{i=1}^{k} r^{(i)}_{\text{proc}}$ be the running average over the first k reflection steps along τ. Adding a $(k{+}1)$-th reflection with reward r_{k+1} produces

$$\bar{r}_{k+1} = \bar{r}_k + \frac{r_{k+1} - \bar{r}_k}{k+1}. \tag{13}$$

Proposition 2 (Marginal incentive of an additional reflection)**.** *Under A2, for any $k \ge 1$:*

(i) *(*Strict improvement gate.*) Adding a $(k{+}1)$-th reflection strictly increases $P(\tau)$ if and only if $r_{k+1} > \bar{r}_k$.*

(ii) *(*Length invariance.*) For every τ, $|P(\tau)| \le 1$ regardless of k. In particular, $P(\tau)$ does not grow with the reflection count and remains commensurate in scale with $r_{\text{out}} \in [0, 1]$.*

(iii) *(*Comparison to summation.*) The unnormalized alternative $P_{\text{sum}}(\tau) := \sum_{i=1}^{k} r^{(i)}_{\text{proc}}$ satisfies $|P_{\text{sum}}(\tau)| \le k$ and is strictly increased by any $r_{k+1} > 0$, so it admits an inflation incentive that* (12) *does not.*

Proof. (i) follows from (13); (ii) from the triangle inequality combined with A2; (iii) is direct. □

Averaging therefore prevents reward inflation by reflection count: a new reflection helps only when it beats the running average, and the process term remains bounded independently of k.

K.4 Property 3: SNR Improvement of the GRPO Advantage

Property 3 formalizes the credit-assignment claim: when the process reward contains residual signal beyond the outcome reward, adding it improves the SNR of the GRPO action comparison.

Two-policy setup. Fix a prompt x and consider two policies π^+ and π^- that agree everywhere except at a single reflection step t^*. Under either policy, draw rollouts and compute, at t^*:

$$\Delta_{\mathrm{o}} = \mathbb{E}_{\pi^+}[r_{\mathrm{out}}] - \mathbb{E}_{\pi^-}[r_{\mathrm{out}}], \quad \Delta_{\mathrm{p}} = \mathbb{E}_{\pi^+}[r_{\mathrm{proc}}^{(t^*)}] - \mathbb{E}_{\pi^-}[r_{\mathrm{proc}}^{(t^*)}],$$

$$\sigma_{\mathrm{o}}^2 = \mathrm{Var}(r_{\mathrm{out}}), \quad \sigma_{\mathrm{p}}^2 = \mathrm{Var}(r_{\mathrm{proc}}^{(t^*)}), \quad c = \mathrm{Cov}(r_{\mathrm{out}}, r_{\mathrm{proc}}^{(t^*)}),$$

where variances and covariance are computed under the local comparison distribution. For a single reflection ($k = 1$), the augmented return is $R_\beta = r_{\mathrm{out}} + \beta\, r_{\mathrm{proc}}^{(t^*)}$, and the per-sample SNR is

$$\rho_{\mathrm{aug}}^2(\beta) := \frac{(\Delta_{\mathrm{o}} + \beta\Delta_{\mathrm{p}})^2}{\sigma_{\mathrm{o}}^2 + \beta^2\sigma_{\mathrm{p}}^2 + 2\beta c}, \qquad \rho_{\mathrm{out}}^2 := \rho_{\mathrm{aug}}^2(0) = \Delta_{\mathrm{o}}^2/\sigma_{\mathrm{o}}^2. \tag{14}$$

Larger ρ_{aug}^2 means the two policies are easier to separate under group-relative normalization.

The informativeness condition. Let $\widehat{\beta}_{\mathrm{p|o}} := c/\sigma_{\mathrm{o}}^2$ be the regression coefficient of $r_{\mathrm{proc}}^{(t^*)}$ on r_{out}.

Theorem 1 (Strict SNR improvement)**.** *Suppose $\Delta_{\mathrm{o}}\Delta_{\mathrm{p}} \geq 0$ (signal alignment), and that the residual-signal condition*

$$\frac{\Delta_{\mathrm{p}}}{\Delta_{\mathrm{o}}} > \widehat{\beta}_{\mathrm{p|o}} \tag{15}$$

holds (with the convention that the inequality is satisfied when $\Delta_{\mathrm{o}} = 0$ and $\Delta_{\mathrm{p}} \neq 0$). Then there exists $\beta_0 > 0$ such that $\rho_{\mathrm{aug}}^2(\beta) > \rho_{\mathrm{out}}^2$ for every $\beta \in (0, \beta_0]$.

Proof. If $\Delta_{\mathrm{o}} = 0$ and $\Delta_{\mathrm{p}} \neq 0$, then $\rho_{\mathrm{out}}^2 = 0$ while

$$\rho_{\mathrm{aug}}^2(\beta) = \frac{\beta^2\Delta_{\mathrm{p}}^2}{\sigma_{\mathrm{o}}^2 + \beta^2\sigma_{\mathrm{p}}^2 + 2\beta c} > 0$$

for every sufficiently small $\beta > 0$, proving the claim. Assume henceforth that $\Delta_{\mathrm{o}} \neq 0$. By the quotient rule,

$$\frac{\mathrm{d}}{\mathrm{d}\beta}\rho_{\mathrm{aug}}^2(\beta)\bigg|_{\beta=0} = \frac{2\Delta_{\mathrm{o}}^2}{\sigma_{\mathrm{o}}^2}\left(\frac{\Delta_{\mathrm{p}}}{\Delta_{\mathrm{o}}} - \widehat{\beta}_{\mathrm{p|o}}\right),$$

which is strictly positive under (15). By continuity, $\rho_{\mathrm{aug}}^2(\beta) > \rho_{\mathrm{aug}}^2(0) = \rho_{\mathrm{out}}^2$ on some interval $(0, \beta_0]$. □

Interpretation. Condition (15) says that the process reward contains residual information about the policy difference beyond what is linearly recoverable from the outcome reward. The theorem analyzes a single reflection comparison and sufficiently small positive β. With multiple reflections, Eq. (12) bounds the process term independently of k, but this alone does not establish the same SNR guarantee for variable reflection counts or the chosen $\beta = 0.2$.

Empirical residual-signal diagnostic. We examine a residual-signal diagnostic motivated by (15) on the SFT-only versus full-RL comparison. From validation prompts whose $G = 4$ rollouts contain at least one reflection step, we sample 30 prompts and record r_{out} and $P(\tau)$ for both checkpoints. For each prompt x, we compute RL-minus-SFT differences $\widehat{\Delta}_{\mathrm{o}}(x)$ and $\widehat{\Delta}_{\mathrm{p}}(x)$, estimate $\widehat{\beta}_{\mathrm{p|o}}(x) = \widehat{c}(x)/\widehat{\sigma}_{\mathrm{o}}^2(x)$ from pooled within-checkpoint covariance, and define

$$\widehat{\Delta}_{\mathrm{p}}^{\mathrm{resid}}(x) = \widehat{\Delta}_{\mathrm{p}}(x) - \widehat{\beta}_{\mathrm{p|o}}(x)\widehat{\Delta}_{\mathrm{o}}(x). \tag{16}$$

Positive values mean that the process-reward gain is not exhausted by the component linearly explained by outcome-reward gain. When computing headline rates, we filter prompts with $|\widehat{\Delta}_{\mathrm{o}}| < 0.02$. Figure 3 shows that positive residual values occur for 80.8% of retained prompts, with the largest median residual signal on 4-event prompts.

L Implementation Details

Backbone and tool set. Our final model ReFactor-Qwen3.5-9B uses Qwen3.5-9B Qwen Team (2026) as the policy backbone. The agent has access to five tool classes: (1) a text-to-image generator (FLUX.2[klein]4B Labs (2025)) for keyframe synthesis; (2) a text-to-video generator (HunyuanVideo1.5 Team (2025)) for prompt-conditioned clip generation; (3) an image-to-video generator (HunyuanVideo1.5 Team (2025)) for reference-frame-conditioned clip generation; (4) a video editor that supports temporal stitching; (5) a frame extractor that returns a fixed-size set of frames from any video artifact for inspection or image-based video generation. All generation backends—the policy backbone, the T2I generator, and the T2V/I2V generator—are fully open-source with publicly available model weights. The video editor and frame extractor are lightweight deterministic utilities implemented with standard open-source libraries (FFmpeg and OpenCV). The controller and generation tools use open weights; artifact captioning uses Gemini-3-Flash as specified below. All tools are wrapped behind a unified JSON schema and exposed to the agent through the same ReAct interface.

SFT training. We fine-tune Qwen3.5-9B for 1 epoch on the constructed cold-start dataset (Sec. 4.3, Appendix I). We use AdamW with learning rate 5×10^{-5}, cosine decay, and global batch size 128. We mask the loss on tool-execution outputs so that gradients only flow through the agent's reasoning traces and tool-call tokens. Sequence length is capped at 16k tokens; visual artifacts in the short-term buffer are encoded by the native vision tower of the backbone.

RL training. We optimize the SFT-initialized policy with GRPO. Each rollout has a step budget of $T = 12$ per episode. We use a learning rate of 1×10^{-6}, KL coefficient $\lambda = 0.01$, clipping $\epsilon = 0.2$, sampling temperature 0.5, and reward weights $\alpha = 0.9$, $\beta = 0.2$. The group size is $G = 8$ with a mini-batch size of 4, yielding 32 responses per batch across 16 GPUs (one per GPU), with gradient accumulation steps set to 2. The outcome reward r_{narr} is computed by a frozen Gemini-3.1-Pro evaluator outputting in $[0, 1]$, and r_{vis} by a frozen VideoScore2.0 model whose three axes (each in $[0, 5]$) are averaged and rescaled to $[0, 1]$. Process rewards use an image-quality scorer for image artifacts and averaged VideoScore2.0 scores for video artifacts. Both are rescaled to $(0, 0.2]$ and floored at $Q_{\min} = 0.05$ before computing the quality ratio, so the effective score range is $[0.05, 0.2]$. The process reward is clipped to $[-0.2, 0.2]$ per Eq. (5).

Memory. The structured visual memory keeps the most recent $K = 1$ artifact as a raw visual input; older artifacts are captioned by a Gemini-3-Flash captioner.

Discussion on the visual memory window size. Setting $K = 1$ is a pragmatic trade-off between context efficiency and visual detail retention under the current backbone's context budget. A larger K would allow the agent to directly compare non-adjacent artifacts—for instance, verifying character appearance consistency between the first and fourth scenes—without relying on the fidelity of intermediate captions. We expect that as MLLM backbones scale to support longer multimodal contexts and achieve stronger fine-grained visual understanding, increasing K may yield further gains, particularly for identity-sensitive multi-character narratives where caption-based representations inevitably lose subtle visual details such as clothing texture and facial features.

Hardware. SFT is performed on 32 GPUs for approximately 8 hours. RL training takes 16 GPUs for approximately 30 hours, with tool execution distributed on a separate worker pool.

Code. Source code is included in the supplementary material and will be publicly released upon acceptance.

M Limitations

ReFactor introduces an autonomous agentic video generation framework, which greatly advances current model-based or workflow-based video generation frameworks. However, as an early attempt of agentic video generation, ReFactor faces two main limitations. First, the use of memory in ReFactor is relatively simple, future work could develop more efficient memory management methods. Second, we consider five commonly used video generation tools in ReFactor, it is beneficial to extend the pool of tools, which may further improve performance.