Crayotter: Artifact-Grounded Multimodal Agents for Long-Form Video Editing

Lecheng Yan*†, Yichong Zhang*†, Ben Pan*, Xiaoyu Zheng*, Jiawei Qian*, Anqi Wu*, Wenxi Li, Chenyang Lyu
Crayotter Project

Abstract

Recent progress in LLM agents has improved short-horizon tool use, but long-horizon multimodal editing remains difficult: decisions are weakly grounded, intermediate states are hard to audit, and failures are expensive to localize. We formulate long-form video editing as an *artifact-grounded agent trajectory* problem, where planning, execution, and revision are all conditioned on explicit external states. We present **Crayotter**, an open-source multimodal multi-agent framework that maps one natural-language intent to a complete edited video via three phases: material preparation, deep editing research, and tool-grounded agent execution. Our central mechanism is **environment-grounded reflection**: the agent updates its policy from observable artifacts instead of relying only on latent reasoning traces. We further propose a trajectory-level RLVR framework with a three-tier cascade reward that combines verifiable editing signals, LLM-as-judge multi-dimensional scoring, and human preference calibration, optimized via GRPO for sample-efficient policy improvement. Code and examples are publicly available at `https://github.com/idwts/Crayotter`.

1 Introduction

Long-form visual storytelling is rapidly shifting from single-model generation to production-oriented agent systems. Large language models now support multi-step reasoning, tool use, and role-based collaboration (Yao et al., 2022; Wu et al., 2024; Hong et al., 2023). In video creation, recent systems decompose production into explicit stages such as story planning, shot generation, review, cinematography, and post-production (Lin et al., 2023; Xie et al., 2024; Wu et al., 2025a; Xu et al., 2025a,b; Shi et al., 2025). FilmAgent, for example, makes virtual filmmaking concrete by simulating crew roles in 3D spaces, while AniMaker shows how multi-agent animation can benefit from candidate exploration and storytelling-aware evaluation. A common lesson is emerging: high-quality AI video depends not only on stronger generators, but also on controllable production workflows.

Long-form video *editing*, however, remains under-formalized from an agent-systems perspective. Editing is not simply a matter of prompting a video generator. It requires a sequence of concrete decisions over heterogeneous artifacts: user intent, local or retrieved footage, multimodal clip analyses, timeline structures, transitions, narration, subtitles, audio levels, tool logs, and rendered feedback. A small error in asset selection or clip timing can propagate into broken continuity, unnatural pacing, or narration that no longer matches the image. This makes editing a long-horizon, multimodal control problem where state grounding, trajectory reproducibility, and failure diagnosis are central.

Figure 1 illustrates the kind of workflow we target. A user starts from a high-level travel-video request, the system prepares or searches for visual material, and the agent exposes a ReAct-style editing trajectory rather than a black-box answer. Cuts, transitions, narration, subtitles, and rendered previews become inspectable units. When the result fails, the agent can revise the affected timeline segment or hand control to a human editor instead of restarting the entire production. This is the core intuition behind Crayotter: a video-editing agent should make its production state visible, reusable, and correctable.

Existing work only partially covers this setting. Generation-first pipelines are strong at narrative synthesis, and recent long-context models improve temporal consistency beyond single clips (Kim et al., 2024; Guo et al., 2025; Wu et al., 2025b; Dalal et al., 2025). Yet these methods usually focus on synthesizing shots rather than coordinating editing-time operations over real footage. Tooling-

*Core Contributor.
†Project Lead.

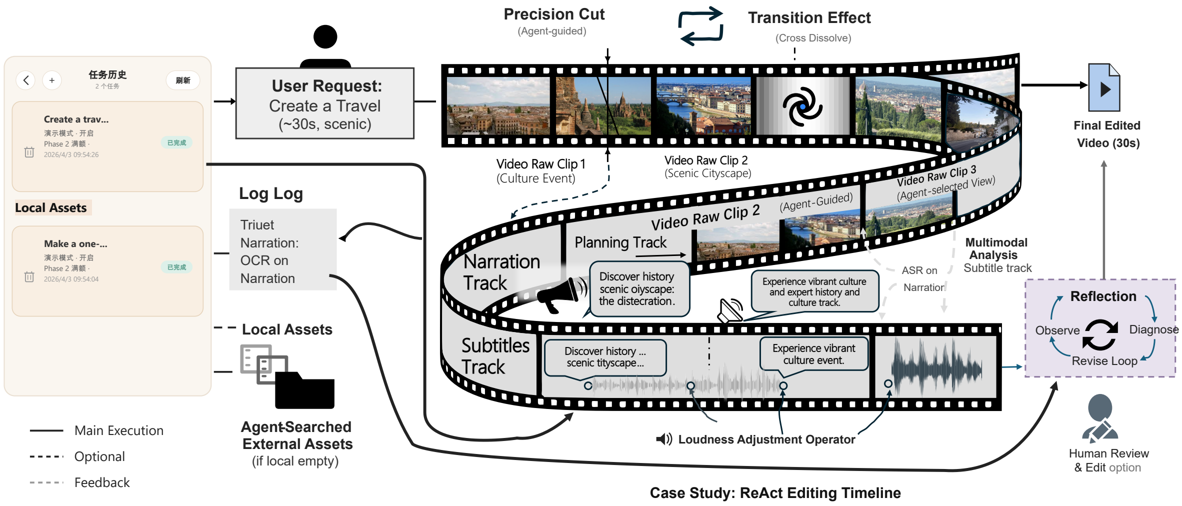

Figure 1: Crayotter client and case-study editing trajectory. Left: the workbench entry exposes task history, local assets, and the fallback path to agent-searched external assets. Center: a ReAct editing timeline for a travel-video request, where the agent performs precision cuts, arranges cross-dissolve transitions, and aligns narration and subtitle tracks with clip-level content. Right: multimodal analysis, reflection, and optional human review form an explicit revise loop before final export.

oriented projects improve practical usability, but they often lack scientific trajectory logs that allow a run to be replayed, diagnosed, or compared. Meanwhile, evaluation remains fragmented for editing-specific qualities such as cut continuity, transition naturalness, narration alignment, and localized error attribution (He et al., 2025; Mao et al., 2026; Zhuang et al., 2025; Liu et al., 2024).

We present **Crayotter**, a multimodal multi-agent framework for prompt-driven video editing over heterogeneous source material. Given a user intent, Crayotter executes three phases. In *material preparation*, it expands the request into coverage tags, retrieves or uses local videos, reranks candidate assets, and produces multimodal analysis artifacts. In *deep editing research*, it synthesizes those artifacts into an editing blueprint, including narrative structure, clip order, rhythm, transitions, and narration. In *tool-grounded agent execution*, it translates the blueprint into editing actions such as cutting, timeline construction, transition insertion, subtitle generation, loudness adjustment, and export.

The key mechanism is **environment-grounded reflection**. Rather than treating an LLM conversation as the only state, Crayotter externalizes state into artifacts that can be inspected by both agents and users: retrieval coverage reports, analysis JSON, timeline plans, transition plans, tool calls, intermediate renders, and final outputs. Inspired by process-oriented research agents and artifact-based reflection (Yang and Weng, 2025; Zheng et al., 2026), the system revises from observable environment feedback and selectively re-executes failed segments. This design turns long-form editing from a single opaque generation attempt into a reproducible trajectory of planning, execution, and repair.

Our contributions are four-fold.

1. We formulate long-form multimodal video editing as an artifact-grounded agent trajectory problem, where source assets, plans, timeline states, renders, and logs are explicit state variables for reflection and repair.

2. We introduce a coverage-aware multimodal footage retrieval loop that decomposes an editing request into visual, narrative, style, and action coverage tags, reranks candidate videos with sampled frames and temporal windows, and iteratively searches for missing semantic evidence.

3. We deliver Crayotter as an open-source three-phase editing system with structured editing research, tool-grounded execution, runtime observability, configurable phase routing, an experience memory system, and a workbench client for human-agent partnership.

4. We propose a trajectory-level RLVR framework using GRPO with a three-tier cascade reward that combines verifiable tool signals,

episode-level outcome scoring, and human preference calibration.

2 Related Work

2.1 Agentic Story and Film Generation

Multi-agent systems have recently been used to automate story and film production. VideoDirectorGPT introduces LLM-guided planning for multi-scene video generation with layout control (Lin et al., 2023). DreamFactory and MovieAgent further push multi-agent coordination for long-video production (Xie et al., 2024; Wu et al., 2025a). MM-StoryAgent extends the pipeline to cross-modal text-image-audio storybooks (Xu et al., 2025a), while FilmAgent targets end-to-end virtual 3D film automation (Xu et al., 2025b). AniMaker adds MCTS-guided candidate exploration and animation-specific evaluation (AniEval) for multi-shot narrative animation (Shi et al., 2025).

Compared to these systems, Crayotter focuses on practical editing over retrieved real videos, emphasizing artifact-observable execution, reproducible tool traces, and post-production integration rather than generation-first virtual production.

2.2 Long-Horizon Video Editing and Cinematic Compilation

Several recent systems are closer to Crayotter because they operate on existing or long-form video materials rather than only synthesizing new clips. DIRECT formulates video mashup creation as hierarchical multi-agent planning, with screenwriter, director, and editor roles coordinating structural anchoring, editing intent, and fine-grained shot-sequence optimization (Li et al., 2026). CineAgents targets instruction-driven cinematic video compilation, using CineBench for evaluation and a design-and-compose workflow with hierarchical narrative memory and iterative narrative planning (Zhang et al., 2026). CutClaw addresses hours-long video editing under music synchronization, combining multimodal decomposition, long-range planning, and fine-grained clip selection (Zhao et al., 2026).

Open-source editing tools also show strong practical momentum. FireRed-OpenStoryline supports conversational video creation with online media search, LLM-powered planning, precise tool orchestration, human-in-the-loop control, and reusable style skills.[1] NarratoAI focuses on one-click film commentary and automatic editing workflows, providing a product-oriented pipeline for script generation, narration, and video assembly.[2] These projects demonstrate usable interfaces and task-specific automation. Crayotter differs by making the editing process a research object: it externalizes retrieval evidence, analysis artifacts, timeline states, tool logs, and rendered feedback so that a run can be replayed, diagnosed, selectively repaired, and eventually optimized with trajectory-level signals.

Evaluation is increasingly central for story visualization and long-form video. DreamStory introduces multi-subject consistency modeling and DS-500 benchmark (He et al., 2025). StoryIter proposes iterative long-story refinement (Mao et al., 2026). ViStoryBench provides comprehensive story visualization metrics and human-verified scripts (Zhuang et al., 2025). EvalCrafter analyzes evaluation reliability for video generation models (Liu et al., 2024).

Crayotter can reuse such benchmarks while adding editing-specific axes such as cut continuity, transition naturalness, and narration timeline validity.

2.3 Long-Context and Multi-Event Video Modeling

Long context remains a bottleneck in video generation. FIFO-Diffusion explores infinite-length inference without retraining (Kim et al., 2024). Long Context Tuning expands context windows of pretrained single-shot diffusion models for coherent multi-shot scenes (Guo et al., 2025). MinT introduces temporal control by binding events to time windows (Wu et al., 2025b). One-Minute Video Generation with test-time training studies longer narrative generation via architectural adaptation (Dalal et al., 2025).

These approaches focus on generative backbones, while our system-level setting operates on editing-time composition and revision of heterogeneous source materials.

2.4 Deep Research Training and Reflection

DR Tulu proposes reinforcement learning with evolving rubrics for long-form deep research tasks

[1] https://github.com/FireRedTeam/FireRed-OpenStoryline

[2] https://github.com/linyqh/NarratoAI

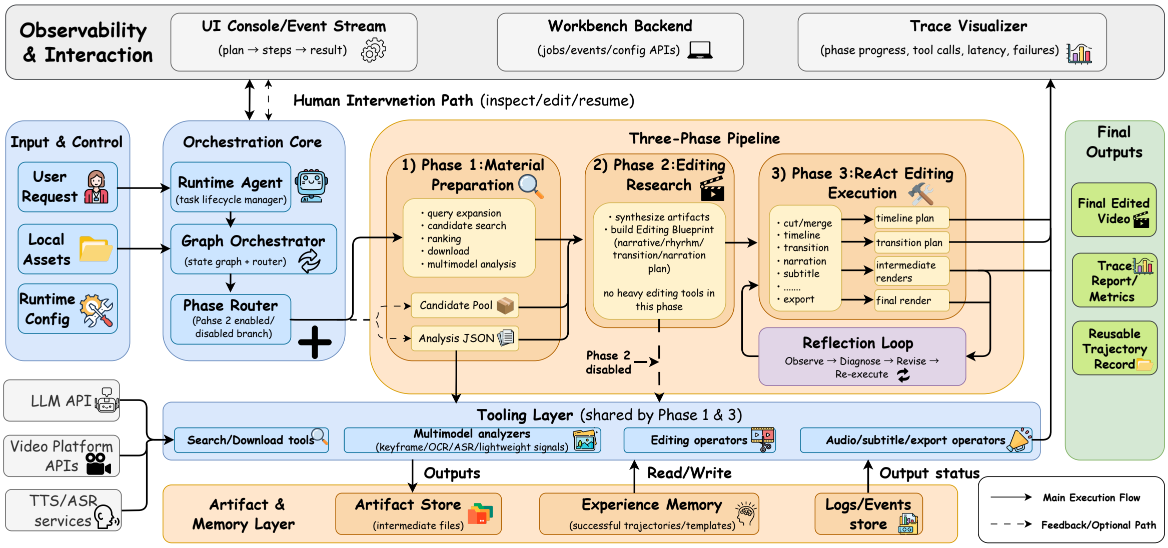

Figure 2: Overall architecture of Crayotter. The system follows a three-phase pipeline (material preparation, editing research, and tool-grounded agent execution) with artifact-grounded reflection, runtime observability, and reusable trajectory logging.

(Shao et al., 2025); SAGE diagnoses retrieval bottlenecks in deep research agents (Hu et al., 2026). DeepPresenter shows that environment-grounded reflection over rendered artifacts can outperform fixed template workflows in presentation generation (Zheng et al., 2026). This directly motivates our editing reflection loop over rendered video artifacts.

3 Method: Crayotter Architecture

Crayotter maps an open-ended editing request to a finished video through a sequence of inspectable intermediate representations rather than through a single latent generation step. Given a user request q and an optional local asset set $\mathcal{A}_0$, the system realizes the transformation

$$\mathcal{F} : (q, \mathcal{A}_0) \to y,$$

where y is the exported video. We factorize $\mathcal{F}$ into three production phases:

$$(q, \mathcal{A}_0) \xrightarrow{\text{retrieval}} \mathcal{P} \xrightarrow{\text{research}} \mathcal{B} \xrightarrow{\text{tools}} y.$$

Here $\mathcal{P}$ is a coverage-verified material pool, $\mathcal{B}$ is a time-grounded editing blueprint, and the final phase executes $\mathcal{B}$ in an external editing environment. This decomposition follows the production logic of recent agentic visual systems: every phase produces artifacts that are visible to later agents, to the tool environment, and to a human reviewer.

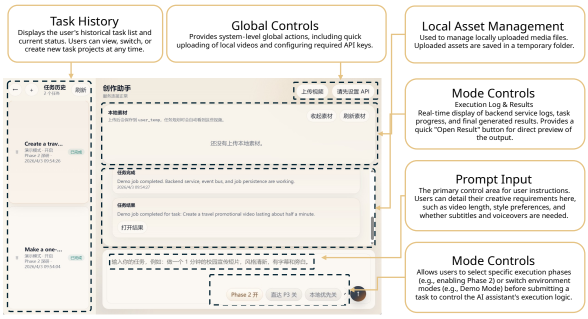

Figure 3: Crayotter workbench interface. The client exposes task history, local asset slots, execution status, and agent-generated intermediate artifacts in one workspace, supporting human-agent inspection during long-horizon editing.

Figure 3 shows the corresponding workbench interface through which users inspect tasks, assets, and intermediate agent outputs during execution.

3.1 Task Formulation and Intermediate Artifacts

We define a source video as $v_j = (x_j, a_j, L_j)$, with visual stream x_j, optional audio stream a_j, and duration L_j. An edited video is represented as a timeline

$$y = \mathrm{Render}(\mathcal{E}), \quad \mathcal{E} = \{e_\ell\}_{\ell=1}^{N},$$

where each event $e_\ell = (v_{j_\ell}, \tau_\ell, o_\ell)$ selects a source video, a temporal interval $\tau_\ell = (t_\ell^{\text{in}}, t_\ell^{\text{out}})$, and an editing operator o_ℓ such as cut, transition, narration

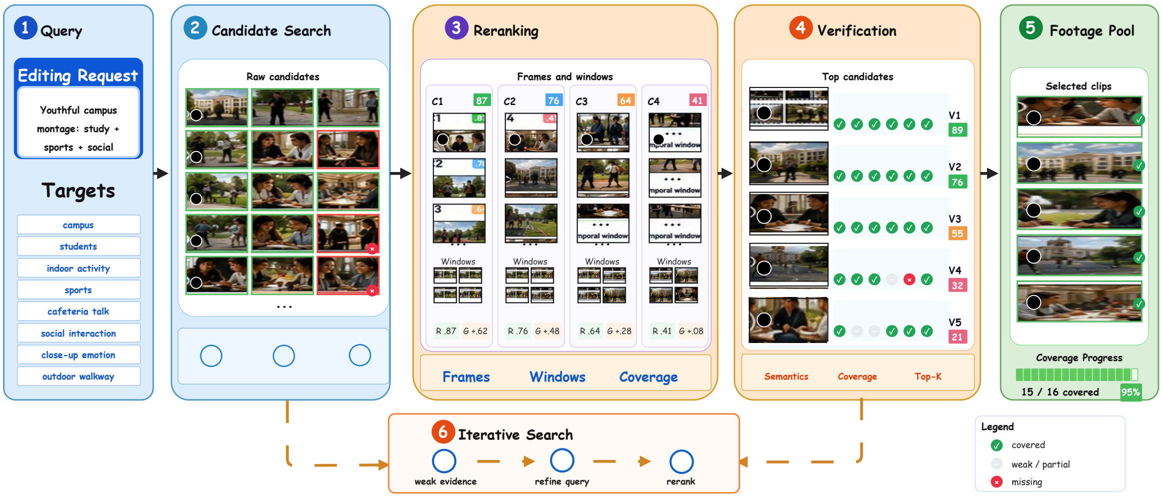

Figure 4: Coverage-aware multimodal footage retrieval. Crayotter converts an abstract editing request into concrete coverage tags, retrieves a high-recall video pool, reranks candidates through frame and temporal-window analysis, verifies top-K videos with full-video understanding, and turns uncovered tags into follow-up searches until the material pool is sufficient or the retrieval budget is exhausted.

alignment, subtitle placement, loudness normalization, or export. The core difficulty is that τ_ℓ and o_ℓ must be chosen from imperfectly retrieved material while remaining faithful to the global request q.

Crayotter therefore introduces three explicit artifact types:

$$\mathcal{P} = (\mathcal{S}, \Gamma, \mathcal{M}),$$
$$\mathcal{B} = \{b_\ell\}_{\ell=1}^{N},$$
$$\mathcal{H} = \{(s_k, a_k, s_{k+1}, \rho_k)\}_{k=1}^{K}.$$

$\mathcal{S}$ is the selected source pool, Γ stores coverage evidence, $\mathcal{M}$ stores multimodal analysis records, $\mathcal{B}$ is a shot-level editing blueprint, and $\mathcal{H}$ is the tool execution history with observable rewards or diagnostics ρ_k. The following subsections describe how each artifact is constructed.

3.2 Phase 1: Coverage-Aware Multimodal Footage Retrieval

Source-footage preparation is a central bottleneck in practical video editing: the editor cannot plan a convincing timeline unless the material pool contains visual evidence for the requested story. We formulate retrieval as weighted tag coverage. The request q is expanded into production-oriented tags

$$T = T^{\text{scene}} \cup T^{\text{act}} \cup T^{\text{style}} \cup T^{\text{story}} \cup T^{\text{shot}},$$

where the dimensions encode scene requirements, people/actions, affective style, narrative purpose, and expected shot evidence. Each tag t_i is assigned an importance weight w_i.

At retrieval round r, a query generator produces q_r from the original request and the currently missing tags. A coarse retriever returns a high-recall candidate list

$$\mathcal{C}_r = \mathrm{TopM}_{v \in \mathcal{D}} \big[\alpha \, s_{\text{text}}(q_r, v) + (1-\alpha) \, s_{\text{meta}}(q_r, v) \big],$$

where $\mathcal{D}$ is an external video index. For each candidate v, the system samples frames $F(v)$ and temporal windows $W(v)$, then estimates a tag-support vector

$$g(v, t_i) = \max_{u \in F(v) \cup W(v)} p_\psi(t_i \mid u, q),$$

where p_ψ denotes the multimodal understanding module. Candidates are reranked by request relevance and marginal coverage gain:

$$D_i(v, \mathcal{S}) = \max\big(0, g(v, t_i) - G_{\mathcal{S}}(t_i)\big),$$
$$\mathrm{score}(v \mid \mathcal{S}, T) = \lambda \, s_{\text{rel}}(q, v) + (1-\lambda) \sum_i w_i \, D_i(v, \mathcal{S}),$$

with $G_{\mathcal{S}}(t_i) = \max_{v \in \mathcal{S}} g(v, t_i)$. The top-$K$ videos are then downloaded for full-video verification. The selected pool $\mathcal{S}$ is evaluated by the normalized coverage objective

$$\mathrm{Cov}(\mathcal{S}, T) = \frac{\sum_i w_i \, \mathbb{I}[G_{\mathcal{S}}(t_i) \geq \eta_i]}{\sum_i w_i}.$$

Uncovered tags form $\Delta_r = \{t_i \mid G_{\mathcal{S}}(t_i) < \eta_i\}$ and are converted into the next query q_{r+1}. The loop stops when $\mathrm{Cov}(\mathcal{S}, T)$ reaches a target threshold or the retrieval budget is exhausted. Phase 1 outputs the material pool $\mathcal{P} = (\mathcal{S}, \Gamma, \mathcal{M})$, where Γ records tag-level evidence and $\mathcal{M}$ stores per-video multimodal analyses.

3.3 Phase 2: Deep Editing Research

Phase 2 transforms the material pool into an editing blueprint without executing editing tools. Its key problem is temporal localization: the agent must decide not only which clip to use, but which in/out point supports a narrative beat. We borrow the interface principle from our timestamp-watermark technical report: many frozen MLLMs localize events more reliably when temporal coordinates are rendered directly on the perceptual evidence. For each selected video v_j, Crayotter constructs a time-marked view

$$\tilde{v}_j = R_\theta(v_j),$$

where R_θ overlays human-readable time coordinates on sampled frames or analysis windows. The model parameters are unchanged; only the input interface is modified. Because the timestamp is drawn on the same frame that contains the event evidence, the research agent can bind semantic observations to absolute editing coordinates.

Let $\mathcal{Z} = \{z_m\}$ be narrative beats derived from q and the retrieval report, such as establishing shot, activity montage, close-up interaction, or closing scene. For each beat, the research agent predicts a source and interval

$$(\hat{j}_m, \hat{\tau}_m) = \mathrm{parse}\big(f_\phi(\{\tilde{v}_j\}_{v_j \in \mathcal{S}}, z_m, \Gamma)\big),$$

then assembles a blueprint

$$b_m = (v_{\hat{j}_m}, \hat{\tau}_m, c_m, r_m, n_m),$$

where c_m is the intended cinematic role, r_m is the transition or rhythm rationale, and n_m is narration or subtitle intent. The result $\mathcal{B} = \{b_m\}$ is therefore not free-form prose: it is a temporally addressable production plan whose intervals can be inspected, revised, and executed by tools. This design localizes later failures to a specific source clip, timestamp span, or planning rationale.

3.4 Phase 3: Tool-Grounded Timeline Execution

Phase 3 realizes the blueprint through external editing tools. We model the editing environment as an artifact transition system. At step k, the state s_k contains the current timeline, source assets, subtitle tracks, narration files, rendered previews, and tool logs. The editor chooses an action $a_k \in \mathcal{U}$ from the tool set:

$$s_{k+1} = E(s_k, a_k; \mathcal{B}),$$

where E is the editing environment. Typical actions include trimming, placing a clip on the timeline, inserting a transition, aligning narration, generating subtitles, normalizing loudness, and exporting a preview.

The key mechanism is environment-grounded reflection. After each tool call, a verifier computes artifact-observable diagnostics

$$\rho_k = R_{\mathrm{tool}}(s_k, a_k, s_{k+1}, \mathcal{B}, q),$$

including tool validity, timestamp accuracy, coverage preservation, narration alignment, transition smoothness, and render quality. If a diagnostic fails, the agent observes the concrete artifact that failed and repairs only the affected segment rather than restarting the full edit.

3.5 Agent Roles and Artifact Contracts

The implementation follows an explicit planner-researcher-executor separation. The planner converts user intent into phase goals, the researcher synthesizes multimodal evidence into an editable blueprint, and the executor translates the blueprint into concrete editing tool calls. Across all phases, state is externalized as artifacts rather than hidden in latent traces. A mandatory *composite re-analysis* step occurs before narration: the merged video is re-analyzed via the multimodal vision model to ensure narration scripts align with actual rendered content rather than only source-material analysis.

We maintain a stable artifact contract:

- input-side artifacts: request, retrieval candidates, and per-video analysis outputs;
- planning artifacts: narrative blueprint, clip ordering, rhythm targets, transition and narration plans;
- execution artifacts: timeline files, intermediate renders, and tool logs.

This contract allows deterministic replay, phase-level diagnosis, and selective re-execution after failures.

Table 1: Tool taxonomy with functional grouping.

Category	Tools
Search	`search_video`
Ranking	`rank_video_candidates`
Download	`download_video`
Analysis	`analyze_video`, `recall_semantic_segments`
Cutting	`cut_video`, `batch_cut_video`
Merging	`merge_videos`
Inspection	`inspect_video_duration`
Transitions	`add_transition`, `plan_transition_timeline`, `list_transition_presets`
Continuity	`score_cut_continuity`, `recommend_transition_for_cut`
Timeline	`build_edit_timeline_from_segments`, `align_narration_to_timeline`, `validate_timeline_constraints`
Narration	`add_narration_segments`, `validate_narration_timeline`
Audio Post	`duck_background_audio`, `normalize_loudness`
Subtitles	`add_subtitles`
Export	`export_video`

3.6 Modular Tool Taxonomy

Crayotter exposes 21 registered tools partitioned into two disjoint sets: Phase 1 preparation tools and Phase 3 editing tools. Table 1 summarizes the full taxonomy.

The transition subsystem supports 17 preset effects organized into three categories—basic (cross-fade, fade-through-black/white), motion (wipe and slide in four directions), and cinematic (zoom-in, smooth-left/right, distance)—each with configurable duration and per-cut-point assignment via a `transition_plan` schema. Continuity tools sample frame signatures around each cut, estimate visual gaps from brightness, saturation, and motion, and recommend hard cuts, dissolves, or fade-through-black transitions. These modules are deliberately lightweight: they provide observable diagnostics for the reflection loop without turning the method into a collection of unrelated engineering features.

Several runtime utilities support the same artifact contract. Phase routing can run the full pipeline, skip the research phase, operate directly on local materials, or perform a local-first sufficiency check before online retrieval. A bounded reference-only memory stores reusable tool patterns, failure guards, and quality checklists while stripping task-specific creative content. Together with an agent stall watchdog, these utilities improve replayability and robustness but do not change the core three-phase formulation.

3.7 Trajectory-Level RLVR

The artifact transition view naturally defines a reinforcement learning with verifiable rewards (RLVR) environment for Phase 3. We optimize a policy π_θ over editing-tool actions with GRPO:

$$\max_{\theta} \mathbb{E}_{a_k \sim \pi_\theta(\cdot|s_k,\mathcal{B})} \left[\sum_{k=1}^{K} R_{\text{tool}}(s_k, a_k, s_{k+1}, \mathcal{B}, q) \right].$$

Unlike preference-only rewards, R_{tool} is grounded in observable editing artifacts: parsable timelines, valid file outputs, measured durations, coverage records, preview renders, and execution logs. The reward is organized as a three-tier cascade: step-level verifiable signals for tool execution and artifact production, episode-level outcome scoring over export success, duration adherence and efficiency, and human preference calibration. The concrete shaping constants are implementation details and are listed in Appendix A.

Training episodes are specified by fixtures containing the user request, target duration, allowed tools, seeded workspace files, and scripted warm-up turns when needed. The implementation exports these fixtures to a `verl`-compatible JSONL format and uses an sglang multi-turn rollout loop with subprocess-backed tool calls. This setup lets us validate reward plumbing and trajectory replay locally before scaling the policy-improvement run.

4 Competitive Evaluation

We evaluate Crayotter on a fixed pack of 23 editing themes against two open-source or practical baselines: CapCut-Mate and CutClaw. Each generated video is rated on a five-point scale along five dimensions: theme alignment, content richness, narrative coherence, editing smoothness, and visual quality. The overall score is the mean of the five dimension scores.

4.1 Scoring Protocol

We report both AI and human judgments. For AI scoring, we use the same multidimensional rubric across all 23 themes and all methods. For human scoring, three annotators score the same outputs; we first average annotator scores for each theme-method pair, then average across the 23 themes. This case-level aggregation avoids giving extra weight to any single annotator file.

Table 2: Human evaluation by dimension. Each cell is averaged over three annotators and 23 themes.

Method	Theme	Richness	Narrative	Smoothness	Visual	Overall
Crayotter (ours)	**3.59**	**3.35**	**3.22**	**3.29**	**3.54**	**3.40**
CapCut-Mate	2.59	2.71	2.01	2.13	2.74	2.44
CutClaw	1.59	1.64	1.72	1.86	1.67	1.70

Table 3: Overall comparison on 23 themes. Scores are mean $\pm$ standard deviation across themes on a 1–5 scale.

Method	Human score	AI score
Crayotter (ours)	$\mathbf{3.40} \pm 0.59$	$\mathbf{2.39} \pm 0.62$
CapCut-Mate	2.44 ± 0.72	2.10 ± 0.64
CutClaw	1.70 ± 0.62	1.57 ± 0.40

The results show that Crayotter substantially outperforms both baselines under human and AI evaluation. Human raters give Crayotter the strongest scores across all five dimensions, which is consistent with the system's explicit material preparation and artifact-grounded planning stages.

5 Discussion

5.1 Process-Level Diagnostics

Our analysis focuses on whether the system remains controllable during long-horizon execution: where plans drift, which artifacts expose failure early, and which revisions yield the highest recovery gain. This process-level view is critical for transforming case anecdotes into actionable engineering signals.

5.2 System Visualization and Case Showcase

Figure 1 is a concrete case-study view. It combines the client-side entry panel, runtime trace surfaces, and an editing-timeline walkthrough for a travel-video request. The left side shows the user-facing workbench: task history, local asset slots, and the fallback path to agent-searched external assets when local material is insufficient. This grounds the system in an observable execution environment instead of a prompt-only interaction.

The center panel makes the editable units explicit: raw clips, agent-guided precision cuts, transition operators, narration planning, subtitle alignment, and loudness adjustment all appear as first-class timeline artifacts. This is important for our claim of environment-grounded reflection, because the agent does not revise from hidden chain-of-thought alone; it revises from rendered timeline states, multimodal analysis outputs, and tool-produced intermediate artifacts that can be inspected by both the system and the user.

The right side of Figure 1 highlights the control loop we care about most in practice: observe, diagnose, revise, and optionally hand control to a human editor. In our setting, debugging a failed run rarely means starting over from scratch. Instead, users can trace failures back to specific asset choices, narration mismatches, or timeline decisions, then revise only the affected stage. This figure therefore serves as both a UI snapshot and a process diagram for controllable long-horizon video editing.

5.3 Limitations and Scope

The 23-theme evaluation provides a competitive picture of end-to-end editing quality across representative long-form requests. Its scope is deliberately system-level: it measures theme alignment, content richness, narrative coherence, editing smoothness, and visual quality, but it does not isolate the individual contribution of every retrieval, reflection, or tool-control module. The RLVR formulation is implemented with verifiable reward traces and a replayable trajectory schema; large-scale policy optimization and broader human preference studies remain outside the scope of this report.

6 Conclusion

This paper presented Crayotter, an open-source multimodal multi-agent framework for long-form video editing. We formulated editing as an artifact-grounded agent trajectory problem and showed how material preparation, deep editing research, tool-grounded execution, and environment-grounded reflection provide explicit states for planning, auditing, and revision. The report also introduced a trajectory-level RLVR formulation and a 23-theme evaluation with both AI and human scoring against CapCut-Mate and CutClaw baselines. Together, these components show that long-form editing agents can be evaluated and improved through observable artifacts rather than latent reasoning traces alone.

References

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

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

Huiguo He, Huan Yang, Zixi Tuo, Yuan Zhou, Qiuyue Wang, Yuhang Zhang, Zeyu Liu, Wenhao Huang, Hongyang Chao, and Jian Yin. 2025. Dreamstory: Open-domain story visualization by llm-guided multi-subject consistent diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, and 1 others. 2023. Metagpt: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*.

Tiansheng Hu, Yilun Zhao, Canyu Zhang, Arman Cohan, and Chen Zhao. 2026. Sage: Benchmarking and improving retrieval for deep research agents. *arXiv preprint arXiv:2602.05975*.

Jihwan Kim, Junoh Kang, Jinyoung Choi, and Bohyung Han. 2024. Fifo-diffusion: Generating infinite videos from text without training. *Advances in Neural Information Processing Systems*, 37:89834–89868.

Ke Li, Maoliang Li, Jialiang Chen, Jiayu Chen, Zihao Zheng, Shaoqi Wang, and Xiang Chen. 2026. Direct: Video mashup creation via hierarchical multi-agent planning and intent-guided editing. *arXiv preprint arXiv:2604.04875*.

Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. 2023. Videodirectorgpt: Consistent multi-scene video generation via llm-guided planning. *arXiv preprint arXiv:2309.15091*.

Yaofang Liu, Xiaodong Cun, Xuebo Liu, Xintao Wang, Yong Zhang, Haoxin Chen, Yang Liu, Tieyong Zeng, Raymond Chan, and Ying Shan. 2024. Evalcrafter: Benchmarking and evaluating large video generation models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22139–22149.

Jiawei Mao, Xiaoke Huang, Yunfei Xie, Yuanqi Chang, Mude Hui, Bingjie Xu, Zeyu Zheng, Zirui Wang, Cihang Xie, and Yuyin Zhou. 2026. Story-iter: A training-free iterative paradigm for long story visualization. In *The Fourteenth International Conference on Learning Representations*.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G Finlayson, David Sontag, and 1 others. 2025. Dr tulu: Reinforcement learning with evolving rubrics for deep research. *arXiv preprint arXiv:2511.19399*.

Haoyuan Shi, Yunxin Li, Xinyu Chen, Longyue Wang, Baotian Hu, and Min Zhang. 2025. Animaker: Multi-agent animated storytelling with mcts-driven clip generation. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–11.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, and 1 others. 2024. Autogen: Enabling next-gen llm applications via multi-agent conversations. In *First Conference on Language Modeling*.

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

Ziyi Wu, Aliaksandr Siarohin, Willi Menapace, Ivan Skorokhodov, Yuwei Fang, Varnith Chordia, Igor Gilitschenski, and Sergey Tulyakov. 2025b. Mind the time: Temporally-controlled multi-event video generation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 23989–24000.

Zhifei Xie, Daniel Tang, Dingwei Tan, Jacques Klein, Tegawend F Bissyand, and Saad Ezzini. 2024. Dreamfactory: Pioneering multi-scene long video generation with a multi-agent framework. *arXiv preprint arXiv:2408.11788*.

Xuenan Xu, Jiahao Mei, Chenliang Li, Yuning Wu, Ming Yan, Shaopeng Lai, Ji Zhang, and Mengyue Wu. 2025a. Mm-storyagent: Immersive narrated storybook video generation with a multi-agent paradigm across text, image and audio. *arXiv preprint arXiv:2503.05242*.

Zhenran Xu, Longyue Wang, Jifang Wang, Zhouyi Li, Senbao Shi, Xue Yang, Yiyu Wang, Baotian Hu, Jun Yu, and Min Zhang. 2025b. Filmagent: A multi-agent framework for end-to-end film automation in virtual 3d spaces. *arXiv preprint arXiv:2501.12909*.

Linyi Yang and Yixuan Weng. 2025. Researstudio: A human-intervenable framework for building controllable deep research agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 896–905.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*.

Peixuan Zhang, Chang Zhou, Ziyuan Zhang, Hualuo Liu, Chunjie Zhang, Jingqi Liu, Xiaohui Zhou, Xi Chen, Shuchen Weng, Si Li, and 1 others. 2026. A benchmark and multi-agent system for instruction-driven cinematic video compilation. *arXiv preprint arXiv:2604.10456*.

Shifang Zhao, Yihan Hu, Ying Shan, Yunchao Wei, and Xiaodong Cun. 2026. Cutclaw: Agentic hours-long video editing via music synchronization. *Preprint*, arXiv:2603.29664.

Hao Zheng, Guozhao Mo, Xinru Yan, Qianhao Yuan, Wenkai Zhang, Xuanang Chen, Yaojie Lu, Hongyu Lin, Xianpei Han, and Le Sun. 2026. Deeppresenter: Environment-grounded reflection for agentic presentation generation. *arXiv preprint arXiv:2602.22839*.

Cailin Zhuang, Ailin Huang, Yaoqi Hu, Jingwei Wu, Wei Cheng, Jiaqi Liao, Hongyuan Wang, Xinyao Liao, Weiwei Cai, Hengyuan Xu, and 1 others. 2025. Vistorybench: Comprehensive benchmark suite for story visualization. *arXiv preprint arXiv:2505.24862*.

A RLVR Reward Details

The trajectory reward used in our implemented RLVR module combines step-level verifiable signals and episode-level outcome scoring. At each tool call, the agent receives a tool-success reward of $+0.6$ or a failure penalty of -0.8, an artifact-production bonus of $+0.1$ when valid files are created, a non-zero return-code penalty of -0.1, a repeat-call penalty of -0.2, and an ordering bonus or penalty in the range ±0.1–0.4 for procedure-sensitive actions.

At episode end, the reward adds an export bonus of $+1.0$, a duration-accuracy term clipped between -0.5 and $+0.8$ using $0.8 - |d_{\text{final}} - d_{\text{target}}|/d_{\text{target}}$, a completion bonus of $+0.3$ for non-empty final output, and an efficiency penalty of -0.02 per tool call beyond six calls. The third tier supports human preference calibration; in the experiments reported here, Tiers 1–2 provide the active training signal while human judgments are reported as evaluation results.