# FRAMEWORKERS: A DYNAMIC MULTI-AGENT FRAMEWORK FOR AI VIDEO PRODUCTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Modern video generators produce high-quality individual clips, but complete video production requires coordinating many interdependent steps, from scripting and storyboarding to generation and editing, together with the assets they produce and consume. Existing automated systems rely on fixed pipelines that adapt poorly to diverse inputs, while general-purpose large language models (LLMs) remain unreliable for long-horizon orchestration and multimodal asset routing. We introduce FRAMEWORKERS, which replaces the fixed production pipeline with learned, descriptor-conditioned orchestration over modular capabilities, grounded in a persistent Workspace. The Director treats production as dynamic task management, editing a Dynamic Task Stack to decide what to do next, which capability to invoke, and how to replan the remaining work after feedback or failure. The Assistant executes each task against shared state that carries scripts, assets, and memory across steps. Capabilities are registered as sub-agents described only by natural-language descriptors, so adding one requires no change to the framework. We train the Director for this routing task with supervised fine-tuning followed by Group Relative Policy Optimization (GRPO). FRAMEWORKERS reaches 88.4% chain-level routing accuracy against 71.2% for the strongest LLM planner we evaluate, recovers from 96.7% of injected runtime failures with a general-purpose LLM as replanner and no recovery-specific training, correctly invokes sub-agents unseen in training, and is rated above single-agent and prior multi-agent video systems in automatic and human evaluation, with the largest human-rated gains on controllability and cross-shot consistency.

## 1 INTRODUCTION

Recent video generators synthesize visually rich clips from text prompts or reference images (Brooks et al., 2024; Kling Team, 2025; Gao et al., 2025; Wan Team, 2025). Real-world video making, however, rarely ends with a single clip: short-form videos, advertisements, and short films require a complete production process spanning script writing, character design, storyboarding, asset organization, video synthesis, and editing. The question is thus how to move from isolated clip synthesis to a complete, automated production system.

Such a system must handle diverse user requirements. Users may provide only a short idea, a detailed script, character portraits, product descriptions, or visual references, and each input calls for a different workflow: a given script should not be regenerated, user-supplied character images must be preserved across shots, and a product promotion must weave product information into the narrative, camera design, and visual style. A fixed pipeline struggles to accommodate this range, which motivates a dynamic, autonomous approach to AI-generated content (AIGC) video production.

Building such a system is challenging. First, production steps are interdependent: storyboards depend on the script, character consistency on both textual descriptions and visual assets, and video generation on shot descriptions, references, and style constraints; different tasks need different combinations of these steps, and intermediate steps may fail and require replanning. Designing a separate workflow for each task type leads to a combinatorial explosion of pipelines that is difficult to extend. Second, production requires extensive asset management: drafts, character profiles, storyboards, reference images, generated clips, and global context must be stored, retrieved, and passed correctly among sub-agents. Third, pretrained language models, despite strong general planning



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.

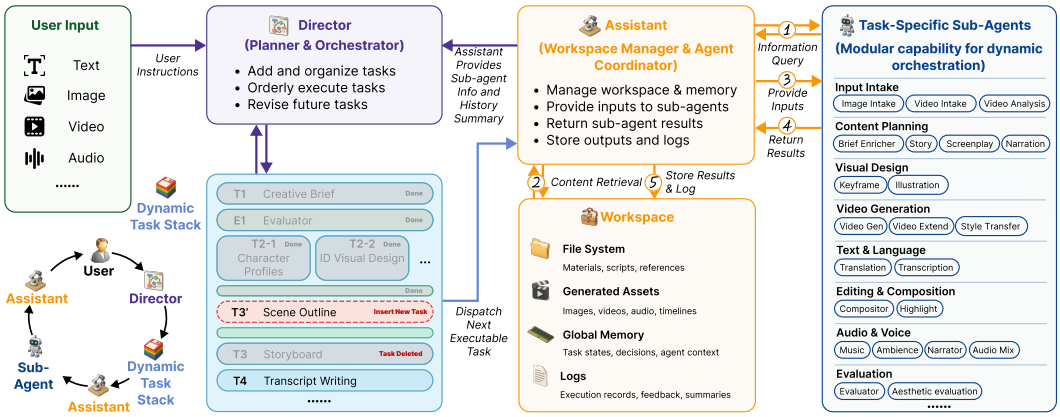


Figure 1: Closed-loop orchestration in FRAMEWORKERS. The Director edits the Dynamic Task Stack from the user input and the Assistant's summaries; the stack shows completed tasks, an inserted task (T3′), and a deleted pending task (T3). The Assistant executes the dispatched task (arrows 1–5): it queries the sub-agent's input requirements, retrieves the inputs from the Workspace, provides them to the sub-agent, receives the results, and stores results and logs in the Workspace. Right: the sub-agent inventory, grouped by capability.

ability, are unreliable as directors for complex task scheduling: they may misjudge task priorities, omit necessary steps, or invoke sub-agents at inappropriate moments. Existing long-form and agentic production systems (Long et al., 2024; Lin et al., 2024; Wu et al., 2025; Li et al., 2024; Zhao et al., 2025; Xie et al., 2026) and media-specific agents (Sandoval-Castañeda et al., 2025; Zhang et al., 2026a; Liang et al., 2025) address parts of this problem, but they assume fixed or task-specific stage orders and provide only partial support for asset management, scheduling, and failure recovery; we discuss related work in Appendix A.

We propose FRAMEWORKERS (Figure 1), which replaces the fixed pipeline with orchestration composed at runtime: a Director decides what to do next and which sub-agent should do it, and an Assistant executes each task against a persistent Workspace and reports back so that the Director can revise the remaining work.

**Contributions.**

- **Video production as dynamic task management.** Instead of executing a fixed pipeline, the Director edits a Dynamic Task Stack with add and delete primitives, so each workflow is composed at runtime and revised after feedback or failure; committed history is immutable (Section 2).
- **Workspace-grounded execution.** A persistent Workspace with a *strict-out, loose-in* interface lets heterogeneous sub-agents exchange artifacts through semantic retrieval rather than hard-wired schemas, so a new capability is added by registering its descriptor and execution wrapper, without modifying the orchestrator or the other sub-agents (Section 2).
- **Learned routing and benchmarks.** We train the Director for descriptor-conditioned routing with supervised fine-tuning followed by Group Relative Policy Optimization (GRPO) (Shao et al., 2024), and evaluate routing, runtime recovery, execution across LLM backends, extensibility to unseen sub-agents, and end-to-end production (Section 3); the code, benchmarks, and training data will be released.

## 2 METHOD

**Overview.** As shown in Figure 1, the *Director* reads the user brief, the sub-agent descriptors, and the current execution state and edits a *Dynamic Task Stack* of remaining work. The *Assistant* executes the next task: it retrieves artifacts from the shared *Workspace*, invokes the selected *sub-agent*, checks the output, and records the result in *Global Memory*, whose summary lets the Director continue, revise the stack, recover from failures, or take new user instructions. This separation between strategic planning and concrete execution enables complex multimodal workflows to be assembled dynamically from modular, replaceable sub-agents. Figure 2 traces one complete run, and Appendix B gives the per-task procedure with a further example (Figure 6).

**Multi-Agent Scheme.** FRAMEWORKERS organizes execution capabilities as a catalog of independent sub-agents, rather than encoding them in a hand-crafted workflow. Each sub-agent is a functionally defined capability module, implemented as a simple workflow, an LLM-driven agent, or a more complex agent with iterative self-refinement. It is exposed through a descriptor specifying its callable identifier, semantic role, required input artifacts, produced output artifacts, and intended trigger conditions. The Director uses these descriptors to select, order, and revise sub-agent calls, while the Assistant uses them to resolve concrete inputs, invoke the selected sub-agent, parse results, and store artifacts in the shared state, so adding or updating a sub-agent requires no change to the framework. Figure 1 (right) summarizes the sub-agent categories used in this study; each category may contain several variants, such as screenplay sub-agents specialized for different genres.

**Director and Dynamic Task Stack.** The Director runs no generation, analysis, or editing model itself; it only maintains the task-level plan. At each planning step it receives the user request and normalized brief, the full descriptor catalog, the Assistant's memory projection (completed tasks, artifacts, failures, and user-provided files), the current Task Stack with the status of every task, and any new user instructions. The Dynamic Task Stack is an ordered, editable record of remaining work and execution history. Each task entry records a unique identifier, assigned sub-agent, natural-language intent, prerequisites, expected output or state change, execution status, and runtime metadata.

Rather than producing a static plan once, the Director edits the Task Stack through two primitive operations: `ADD(task, position)`, which inserts a new task into the stack, and `DELETE(task_id)`, which removes a non-executed task. Task modification is implemented as deletion followed by insertion. This operation set supports initial planning, task refinement, failure recovery, and adaptation to new user instructions. The Director may revise the stack's remaining suffix but cannot erase execution history; this provides clear recovery semantics and auditability.

After each update, the earliest `pending` task whose prerequisites are satisfied by completed tasks or available Workspace artifacts is dispatched to the Assistant, and the workflow ends when no pending task or new instruction remains. If a task fails, the Assistant returns a failure summary, and the Director adds recovery tasks, deletes obsolete ones, or writes a new suffix. Planning and replanning go through the same Task Stack interface, so the Director's model is configurable: the fine-tuned model plans, and replanning can use it or a general-purpose LLM.

**Assistant and Workspace.** The Assistant maintains a persistent *Workspace* with four components: `file_system` (briefs, scripts, storyboards, and notes), `generated_assets` (images, videos, audio, and composited outputs), `global_memory` (high-level summaries of steps, decisions, outputs, failures, and user-provided files), and `logs` (detailed execution traces). The Director sees only a compact projection of `global_memory` and relevant entries, never raw files or logs.

The Assistant also mediates between independently implemented sub-agents under a *strict-out, loose-in* principle: inputs are retrieved semantically from Workspace records, assets, and memory summaries, while outputs pass schema validation, rule checks, and, for most text sub-agents, an LLM review; an output that fails these checks is regenerated with feedback. Sub-agents thus communicate through artifact semantics rather than synchronized schemas; in the run of Figure 2, every input a step consumed was recalled from the Workspace this way.

**Director Training.** Whereas sub-agents solve local tasks with explicit input and output requirements, the Director must map an underspecified goal and the full descriptor catalog to a dependency-consistent sub-agent order, which pretrained LLMs do not do reliably (Section 3). We train it in two stages: supervised fine-tuning (SFT) on structured rationales and plans, then GRPO (Shao et al., 2024) to reward valid format, complete sub-agent coverage, and consistent ordering.

*Training Task.* Each instance pairs a prompt $x$ with a target response $y^*$. The prompt contains the routing rules, the full sub-agent descriptor catalog, and the user goal; the target $y^* = (r^*, p^*)$ contains a structured routing rationale $r^*$ and an ordered, executable sub-agent plan $p^*$. Predicting a chain from an empty Task Stack is equivalent to adding a sequence of tasks, so the learning problem is to infer the required sub-agents and a valid order.

*Supervised Fine-Tuning.* We train on the pairs $(x, y^*)$, initializing the Director from Qwen3 (Yang et al., 2025a) (4B and 8B; 8B unless otherwise stated) with LoRA adapters (Hu et al., 2022). Responses are JSON with a `rationale` that decomposes the request and checks dependencies, and

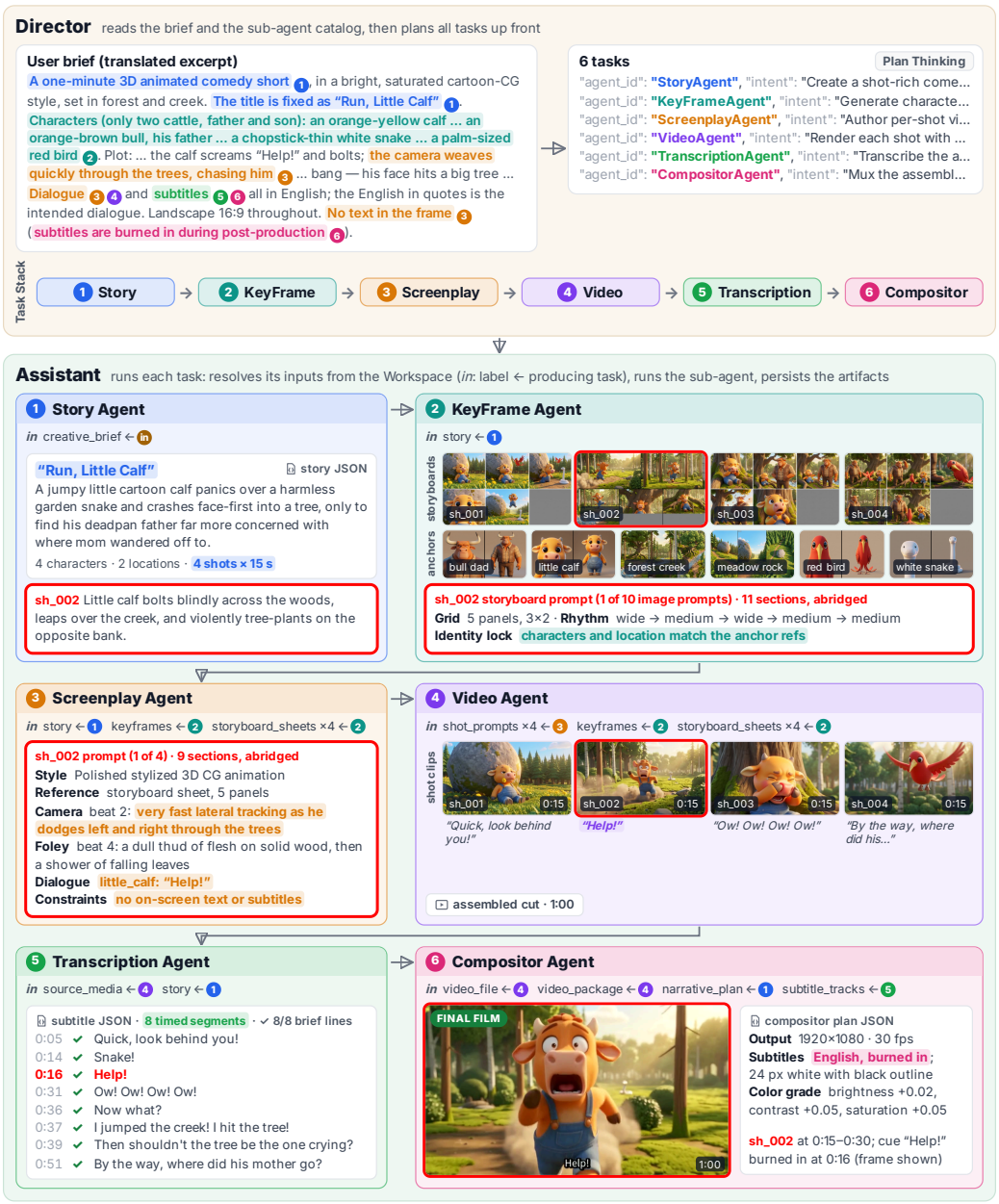


Figure 2: One complete run, from brief to film. Top: the Director plans six tasks on the Dynamic Task Stack; each highlighted requirement in the brief carries the color and number of the task that meets it, and the same highlight marks its evidence in that task's module (brief and prompt excerpts translated from Chinese). Bottom: one module per task, with the inputs it resolved from the Workspace and the artifacts it persisted; red traces shot `sh_002` to its burned-in subtitle. Plan and inputs are re-derived by re-running the planner and resolver on the archived run, and the re-derived plan matches the executed chain.

a `plan` of `(agent_id, intent)` entries (Appendix D.3). We minimize the causal language-modeling loss of the Director policy $\pi_\theta$ on response tokens only,

$$L_{\mathrm{SFT}} = -\,\mathbb{E}_{(x,y^*)}\,\frac{1}{|T(y^*)|}\sum_{t\in T(y^*)}\log \pi_\theta\big(y_t^* \mid x, y_{<t}^*\big),$$

where $T(y^*)$ is the set of response-token positions, so prompt tokens are masked out; this teaches the output format, sub-agent semantics, and basic task dependencies.

*GRPO Reward.* For each prompt $x$, the current policy samples $G$ candidate responses $\{\hat{y}_g\}_{g=1}^{G}$. Each response is parsed into a rationale and an executable plan $\hat{p}_g$. A response is valid only if it parses into the required schema with legal, non-duplicate sub-agent identifiers that form a chain; invalid responses receive zero reward. For valid responses, we score only the executable plan, using sub-agent coverage and dependency-consistent ordering; the rationale is treated as part of the output format rather than scored for semantic quality. The sequence-level reward for a response $\hat{y}$ with plan $\hat{p}$ is

$$R(\hat{y}) = R_{\text{val}}\left(\lambda_{\text{val}} + \lambda_{\text{cov}} R_{\text{cov}} + \lambda_{\text{ord}} R_{\text{ord}}\right), \qquad R_{\text{cov}} = \frac{2\,|S(\hat{p}) \cap S(p^*)|}{|S(\hat{p})| + |S(p^*)|},$$

where $R_{\text{val}} \in \{0, 1\}$ indicates validity, the $\lambda$'s weight validity, coverage, and ordering, and $S(p)$ denotes the sub-agents in plan $p$, so that $R_{\text{cov}}$ is their F1 overlap; unlike recall, F1 also penalizes extra sub-agents, so the policy cannot raise its reward by appending unneeded ones. The ordering reward $R_{\text{ord}}$ is the fraction of prerequisite edges $(a, b)$ of $p^*$ ($a$ must precede $b$) for which both agents appear in $\hat{p}$ with $a$ before $b$ (1 if $p^*$ has no edges), so missing sub-agents receive no ordering credit (Appendix D.3).

*GRPO Objective.* Following GRPO (Shao et al., 2024), rewards are normalized within each group into a sequence-level advantage, and each group drives one on-policy, group-normalized policy-gradient step with a KL penalty to the frozen SFT reference (full objective in Appendix D.3).

## 3 Experiments

We evaluate five aspects of FrameWorkers: Director routing, failure recovery, sub-agent execution across LLM backends, extensibility to unseen sub-agents, and end-to-end video quality.

**Data Construction and Task Setup.** Each routing sample pairs a user goal $g$ with a target sub-agent chain $\mathbf{c} = (c_1, \ldots, c_n)$ over the descriptor catalog. The goals cover diverse AIGC video-production scenarios, including storytelling, highlight generation, multilingual narration and subtitling, style transfer, image-conditioned generation, long-form narration, audio composition, and video extension. The inventory contains 20 sub-agents across text, image, video, audio, analysis, and composition. All reference plans are sequential, with no branching and no repeated sub-agent.

For training, we construct separate SFT and GRPO splits. The SFT split contains 1,404 instruction-plan pairs across eleven routing categories, with chain lengths typically ranging from 2 to 10 agents and long-story goals up to approximately 290 words. Ground-truth chains are produced by template-based generation; every test case is manually verified, and a random 20% of the training pairs are manually inspected (Appendix D.3). For each training chain, we distill a structured chain-of-thought (CoT) reasoning trace from Gemini 2.5 Flash (Comanici et al., 2025), conditioned on the user goal, descriptor catalog, and target chain. The GRPO split contains 1,383 prompts with a closely matched chain-shape distribution; data-quality checks are in Appendix D.3.

We evaluate on two held-out test sets: **Director-Chain-Easy** (DCE), with 522 cases over the 44 chain structures used in training, and **Director-Chain-Hard** (DCH), with 200 cases over 24 generally longer chain structures that never occur in training. Together they cover 68 chain structures, testing both routing accuracy and structural generalization.

**Director Planning Capability.** We evaluate the Director on DCE and DCH using chain accuracy (Chain↑; every step must fall in its reference slot of acceptable sub-agents), per-step routing accuracy (Step↑), and a set-aware edit distance to the target chain (Edit↓; Appendix D.3).

We compare with base Qwen3 checkpoints, production-grade non-reasoning chat models (DeepSeek-V3 (DeepSeek-AI, 2024), GPT-4o (OpenAI, 2024), and Qwen3.6-Plus (Qwen Team, 2026)), and strong reasoning models (Gemini 2.5 Pro (Comanici et al., 2025) and Gemini 3 Pro (Google DeepMind, 2026)). All systems receive the same core planning prompt (routing rules, full descriptor catalog, and user goal) without few-shot examples or policy hints.

As shown in Table 1, our model reaches 91.0% Chain accuracy on DCE and 81.5% on DCH (example plans in Figure 10, Appendix F). Scaling the untrained backbone from 4B to 8B raises overall Chain accuracy only from 12.5% to 24.5%, whereas descriptor-conditioned alignment raises it to 88.4%, suggesting that task-specific alignment matters far more than backbone scale here. Our Di-

Table 1: Director routing accuracy on DCE and DCH (number of cases in parentheses); Ours leads on every metric. C and S are Chain and Step accuracy (%), E is Edit distance, and Overall is the size-weighted average of the two datasets. Ours is the 8B SG-SC Director of Table 2.

| | DCE (522) | | | DCH (200) | | | Overall | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ |
| Qwen3-4B base | 13.4 | 36.7 | 2.99 | 10.0 | 40.5 | 2.62 | 12.5 | 37.8 | 2.89 |
| Qwen3-8B base | 29.5 | 52.9 | 1.93 | 11.5 | 42.6 | 1.93 | 24.5 | 50.0 | 1.93 |
| DeepSeek-V3 | 43.9 | 63.2 | 1.11 | 27.5 | 62.3 | 1.28 | 39.4 | 62.9 | 1.16 |
| GPT-4o | 58.0 | 80.6 | 0.62 | 31.0 | 65.7 | 1.07 | 50.5 | 76.5 | 0.74 |
| Qwen3.6-Plus | 60.5 | 78.5 | 0.85 | 44.0 | 70.2 | 1.21 | 55.9 | 76.2 | 0.95 |
| Gemini 2.5 Pro | 72.2 | 86.2 | 0.57 | 51.0 | 78.1 | 0.78 | 66.3 | 83.9 | 0.63 |
| Gemini 3 Pro | 80.3 | 90.6 | 0.32 | 47.5 | 76.4 | 0.84 | 71.2 | 86.7 | 0.46 |
| **Ours** | **91.0** | **94.4** | **0.15** | **81.5** | **90.7** | **0.50** | **88.4** | **93.4** | **0.25** |

Table 2: LoRA fine-tuning ablation for 4B and 8B Directors. SG is SFT+GRPO; D / FC / SC denote direct, free-form CoT, and structured-CoT supervision. Metrics as in Table 1; bold marks the best value per column within each model size.

| | *4B Models* | | | | | | | | | *8B Models* | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | DCE (522) | | | DCH (200) | | | Overall | | | DCE (522) | | | DCH (200) | | | Overall | | |
| **Setting** | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ | C↑ | S↑ | E↓ |
| SFT-D | 80.3 | 89.5 | 0.30 | 66.0 | 85.8 | 0.45 | 76.3 | 88.3 | 0.34 | 85.2 | 90.4 | 0.34 | 60.0 | 81.7 | 0.81 | 78.2 | 88.0 | 0.47 |
| SFT-FC | 66.9 | 83.4 | 0.63 | 63.0 | 85.4 | 0.52 | 65.8 | 84.0 | 0.60 | 63.0 | 80.3 | 0.72 | 62.0 | 80.6 | 0.78 | 62.7 | 80.4 | 0.74 |
| SFT-SC | 75.1 | 85.8 | 0.46 | 54.0 | 77.3 | 0.95 | 69.2 | 83.4 | 0.60 | 78.7 | 85.3 | 0.41 | 48.0 | 74.8 | 0.94 | 70.2 | 82.2 | 0.56 |
| SG-D | 82.8 | 89.7 | 0.30 | 67.0 | **87.0** | **0.42** | 78.4 | 89.0 | 0.33 | 88.7 | 92.4 | 0.23 | 75.5 | 88.8 | 0.54 | 85.0 | 91.3 | 0.32 |
| SG-FC | 78.9 | 88.8 | 0.38 | 65.5 | 85.2 | 0.62 | 75.2 | 87.7 | 0.45 | 88.5 | 93.1 | 0.22 | 76.0 | 89.5 | 0.55 | 85.1 | 92.0 | 0.32 |
| SG-SC | **92.0** | **94.7** | **0.16** | **68.0** | 86.3 | 0.70 | **85.4** | **92.4** | **0.30** | **91.0** | **94.4** | **0.15** | **81.5** | **90.7** | **0.50** | **88.4** | **93.4** | **0.25** |

rector also outperforms the strongest pretrained reasoning baseline overall, Gemini 3 Pro, by $+10.7$ Chain points on DCE, and the strongest baseline on DCH, Gemini 2.5 Pro, by $+30.5$ points. The larger DCH margin indicates stronger generalization to chain structures unseen in training. Low Edit scores of 0.15 on DCE and 0.50 on DCH suggest that the remaining errors stay close to the target chain rather than being severe structural failures.

Table 2 analyzes model scale, supervision format, and GRPO refinement. After GRPO, scaling from 4B to 8B raises DCH Chain accuracy in every format (e.g., 68.0 to 81.5 for SG-SC), while SG-SC is already near saturation on DCE at both sizes; after SFT alone the 8B model is not better on DCH, so the additional capacity becomes beneficial once GRPO optimizes the executable plan. Supervision format is also critical, but its effect depends on the training stage: after SFT alone, direct supervision is the strongest format, whereas after GRPO, structured CoT outperforms both direct supervision and free-form CoT, with the best structured-CoT setting at 88.4% Chain and 93.4% Step accuracy. GRPO improves every SFT checkpoint, most under distribution shift ($+33.5$ DCH Chain points for 8B structured CoT, 48.0 to 81.5).

**Failure Recovery and Replanning.** Because an initially valid plan can become infeasible after an intermediate failure, we also test replanning on 991 perturbed workflow states: *plan-level*, where a required producer task is removed while a downstream consumer remains, and *execution-level*, where a quality gate rejects the artifact of a completed task (Appendix C). In this recovery benchmark, the Director replans with Gemini 2.5 Flash as its backend, without recovery-specific training, conditioned on the current Task Stack, the failure summary, and the relevant descriptors; a case counts as recovered when the revised Task Stack yields a correct chain under the same slot criterion as Chain accuracy. The system recovers 494 of 495 plan-level cases (99.8%) and 464 of 496 execution-level cases (93.5%), 96.7% overall (Table 6, Appendix C).

**Sub-Agent Execution across LLM Backends.** We run the Assistant and sub-agents on GPT-4o, GPT-4o-mini, Gemini 2.0 Flash, and Gemini 2.5 Flash over a stratified execution suite of 204 cases sampled from DCE and DCH; each case executes its reference chain, so the suite isolates execution from planning. We stratify by chain shape and sample up to three cases per structure, giving a dedu-

Table 3: Case-level execution robustness and failure attribution across LLM backends of the Assistant and sub-agents (204 cases; metrics defined in the text; Lat. is mean end-to-end latency). Pass = First-shot + Rework, and the three failure columns sum to $100\% - \text{Pass}$.

| | Execution | | | | Failure attribution | | |
|---|---|---|---|---|---|---|---|
| Backend | Pass↑ | First-shot↑ | Rework | Lat.↓ | Resolver↓ | Missing↓ | Rework Fail↓ |
| GPT-4o | 61.76% | 55.39% | 6.37% | **55.1s** | 32.35% | 5.39% | 0.49% |
| GPT-4o-mini | 81.37% | 78.43% | 2.94% | 81.1s | 11.76% | 6.86% | **0.00%** |
| Gemini 2.0 Flash | 91.67% | 87.25% | 4.41% | 57.9s | 5.39% | **0.00%** | 2.94% |
| Gemini 2.5 Flash | **100.00%** | **97.55%** | 2.45% | 137.0s | **0.00%** | **0.00%** | **0.00%** |

Table 4: System-level extensibility to previously unseen sub-agents with the fine-tuned Qwen3-8B Director frozen (number of cases in parentheses). Selection denotes correct invocation of the new sub-agent; the remaining metrics follow Table 3.

| New Sub-agent | Selection↑ | Pass↑ | First-shot↑ | Rework | Lat.↓ |
|---|---|---|---|---|---|
| ComedyAgent (200) | 93.0% | 99.5% | 88.0% | 11.5% | 276.3s |
| ScreenwriterAgent (240) | 99.2% | 88.8% | 84.2% | 4.6% | 221.7s |

plicated benchmark covering 68 distinct chain structures. All metrics are computed under mock materialization, where external generation APIs are replaced with mock artifacts while preserving the full scheduling and execution logic. In Table 3, Pass measures whether the full chain completes without input-resolution, evaluator, or Workspace-level failures; First-shot measures success without rework; Rework measures evaluator-guided recovery; and Latency is the average end-to-end runtime. Failed cases are attributed to exactly one of three types: Resolver, where input resolution fails to bind upstream artifacts to the sub-agent's input slots; Missing, where a required upstream artifact is absent or semantically insufficient; and Rework Fail, where rework does not fix the output within the retry budget.

As shown in Table 3, execution robustness varies widely across backends, and a larger model is not necessarily more robust: GPT-4o-mini passes more cases than GPT-4o (81.37% vs. 61.76%), because GPT-4o fails far more often at input resolution (see below), while the two Gemini backends are the most robust (91.67% and 100%). Gemini 2.5 Flash completes all 204 cases successfully; 97.55% succeed on the first attempt, while the remaining 2.45% are recovered through evaluator-guided rework. The gap between First-shot and Pass shows that evaluator feedback repairs local formatting or structural errors before they propagate.

The OpenAI backends mainly fail at input resolution, before any sub-agent runs. GPT-4o and GPT-4o-mini have high Resolver failure rates (32.35% and 11.76%). Because these errors occur before the sub-agent is invoked, evaluator-guided rework cannot repair them, making cross-agent semantic grounding the dominant bottleneck. Their higher Missing failure rates, 5.39% and 6.86%, are consistent with weaker long-range dependency tracking and asset propagation. Gemini 2.0 Flash reduces Resolver failures to 5.39%, but still has 2.94% Rework Fail cases, so a larger share of its remaining failures occurs after input resolution succeeds. Across the four backends, execution stability tracks cross-agent semantic grounding and long-horizon dependency management, and evaluator-guided orchestration keeps pass rates high once these are reliable.

**Extensibility to Unseen Sub-Agents.** To test plug-and-play extensibility, we register two sub-agents unseen in training, ComedyAgent and ScreenwriterAgent, through their descriptors and execution wrappers alone, keeping the fine-tuned Qwen3-8B Director frozen so that it must infer each new sub-agent's role and inputs from its descriptor. As shown in Table 4, ComedyAgent and ScreenwriterAgent are correctly selected in $93.0\%$ and $99.2\%$ of cases, respectively, while the resulting workflows achieve $99.5\%$ and $88.8\%$ pass rates. These results suggest that the frozen Director infers a new sub-agent's role and applicability from its descriptor alone.

**End-to-End Video Evaluation.** We evaluate end-to-end video generation under matched prompts on two real-generation benchmarks, within each of which all systems share the generation backends and use Gemini 2.5 Flash as their LLM (Appendix D.3). The main benchmark has 20 representative cases (10 text-to-video and 10 image-conditioned) and compares a single-agent baseline (S-Agent, Appendix D.1), UniVA (Liang et al., 2025), and FRAMEWORKERS, whose plans come from our

Table 5: End-to-end evaluation on both benchmarks; FRAMEWORKERS has the best judge average and the highest human ratings on both. MLLM-as-a-judge scores C1–C6 (defined in the text) and their average (Avg.) on a 1–5 scale, with the prompt in Appendix D.4; human ratings (1–5, mean over raters) for instruction following (Instr.), temporal consistency (Temp.), and realism (Real.). Per-sub-question differences are in Figure 3, the study protocol in Appendix G.

| | MLLM-as-a-judge | | | | | | | Human | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | Avg.↑ | C1 | C2 | C3 | C4 | C5 | C6 | Instr.↑ | Temp.↑ | Real.↑ |
| *Main benchmark (20 cases)* | | | | | | | | | | |
| S-Agent | 4.08 | 4.50 | 4.55 | 4.50 | 4.60 | 2.70 | 3.65 | 2.61 | 2.70 | 2.69 |
| UniVA | 4.27 | 4.60 | 4.60 | 4.65 | 4.75 | **3.25** | **3.75** | 2.81 | 2.86 | 2.82 |
| Ours | **4.32** | **4.80** | **4.90** | **4.70** | **4.85** | 3.15 | 3.50 | **3.03** | **3.20** | **3.02** |
| *Story-level benchmark (20 cases)* | | | | | | | | | | |
| Anim-Director | 3.10 | 2.95 | 3.35 | 2.70 | 3.50 | 3.55 | 2.55 | 2.91 | 3.05 | 3.29 |
| MovieAgent | 2.16 | 2.00 | 1.90 | 1.25 | 2.35 | 4.30 | 1.15 | 2.24 | 2.80 | 3.18 |
| Ours | **4.17** | **4.25** | **4.30** | **3.75** | **4.40** | **4.55** | **3.75** | **3.61** | **3.45** | **3.58** |

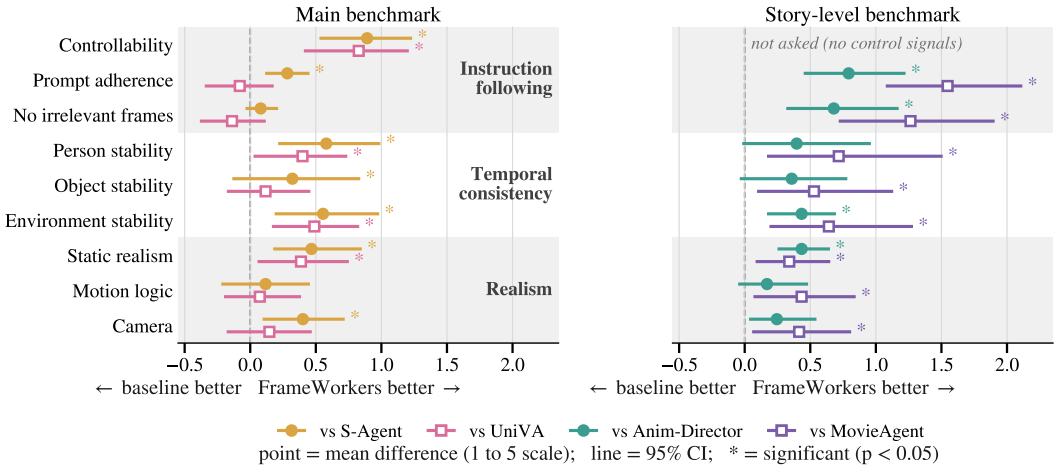


Figure 3: Human evaluation per sub-question: how much higher raters scored FRAMEWORKERS than each baseline on the same case (right of the dashed line = FRAMEWORKERS rated higher); it is rated higher on most sub-questions, most clearly on controllability and throughout the story-level benchmark. Lines are 95% confidence intervals from resampling raters; $*$ marks a significant difference (Holm-corrected Wilcoxon test on the (rater, case) pairs, $p < 0.05$), which can disagree with the interval for items near zero. Controllability is asked only on the main benchmark, whose prompts contain explicit control signals.

fine-tuned 8B Director rather than from Gemini 2.5 Flash. The story-level benchmark has 20 cases (five adapted from MovieAgent (Wu et al., 2025), five from Anim-Director (Li et al., 2024), and ten newly created) and compares both story-focused systems.

*Quantitative Evaluation.* We use Gemini 2.5 Pro as a multimodal LLM (MLLM) judge (Chen et al., 2024a) over 1-FPS sampled frames and the instruction, which reports six 1–5 scores (C1–C6: semantic content, spatial relations, actions, story richness, style consistency, and overall video-text consistency; prompt in Appendix D.4).

As shown in Table 5, FRAMEWORKERS obtains the best average judge score on both benchmarks. On the main benchmark it scores highest on semantic content, spatial relations, actions, and story richness (C1–C4), while UniVA is slightly higher on style consistency (C5) and both baselines on overall video-text consistency (C6); FRAMEWORKERS has the best average (Avg. 4.32 against 4.27 and 4.08). On the story-level benchmark it leads on every dimension (Avg. 4.17 against 3.10 for Anim-Director and 2.16 for MovieAgent). Its main-benchmark margin over UniVA is thus small and concentrated in C1–C4, whereas its story-level margins are large.

*Human Evaluation.* Each rater was assigned 10 cases of one benchmark and scored all systems side by side in random order on nine 5-point sub-questions (eight on the story-level benchmark) grouped into instruction following, temporal consistency, and realism (protocol in Appendix G). Against S-

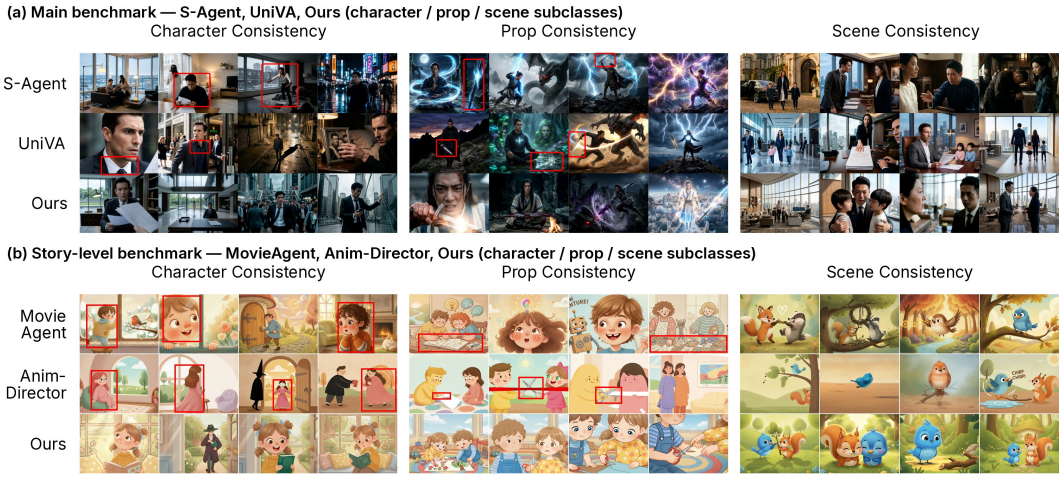


Figure 4: Qualitative comparison of entity consistency. (a) Main benchmark against S-Agent and UniVA: clothing changes, the sword morphs, the office layout shifts. (b) Story-level benchmark against MovieAgent and Anim-Director, one story each for character, prop, and scene: three-year-old Sofia becomes a boy or a grown woman; the children draw instead of cutting, or the scissors change size and shape; the bird's forest turns to autumn or desert. Red boxes mark drift.

Agent, FRAMEWORKERS is rated higher on all three dimensions (paired differences $+0.42$, $+0.49$, $+0.33$; Wilcoxon signed-rank over (rater, case) pairs, Holm-corrected $p < 10^{-4}$) and wins 66% of the pairs in which raters expressed a preference. Figure 3 breaks each human dimension of Table 5 down into its sub-questions as paired differences. Controllability, the one item that asks directly whether requested control signals were produced, shows the largest gap ($+0.89$ over S-Agent and $+0.83$ over UniVA, $p < 10^{-6}$), environment and person stability gain $+0.40$ to $+0.58$, and static realism gains $+0.47$ and $+0.39$, while motion logic is on par with both main-benchmark baselines. Against UniVA the paired dimension-level margins are smaller ($+0.20$ to $+0.33$); temporal consistency is significant on (rater, case) pairs ($p = 0.003$) while content-level prompt adherence is on par, so the gap is concentrated in controllability, cross-shot consistency, and static realism.

On the story-level benchmark, FRAMEWORKERS is rated highest on all eight sub-questions; on (rater, case) pairs (rater-level tests in Appendix G) it is significantly above MovieAgent on every one and above Anim-Director on all three dimensions, with the largest paired margins on instruction following ($+1.41$ over MovieAgent and $+0.74$ over Anim-Director).

Across both evaluations, the gap is most visible on entity consistency (Figure 4): the baselines let characters, props, and scenes drift between shots, while FRAMEWORKERS keeps them stable across distant shots, in line with the person- and environment-stability gains above. Since the generation backend is shared, this is consistent with the design: keyframes are rendered against persistent identity anchors held in the Workspace, and on the main benchmark the evaluator also sends drifting frames back for rework before clip generation.

## 4 CONCLUSION

We presented FRAMEWORKERS, which replaces fixed video-production pipelines with orchestration composed at runtime: a Director edits a Dynamic Task Stack of sub-agent tasks, and an Assistant executes them against a shared Workspace. Trained with SFT and GRPO, the Director routes requests more accurately than general-purpose LLMs, most clearly on chain structures unseen in training; replanning over the Task Stack recovers from most injected failures even when run by a general-purpose LLM; Workspace-grounded execution completes every reference chain with the strongest backend, and its remaining failures concentrate in cross-agent input resolution; unseen sub-agents are invoked from their descriptors alone; and the resulting videos are rated above single- and multi-agent baselines, most clearly on controllability and cross-shot consistency. Scaling the real-generation evaluation and the sub-agent inventory is the main next step; Appendix H discusses the limitations of this work.

### AI USE STATEMENT

Generative AI is part of the system studied in this paper and is disclosed as such: Gemini 2.5 Flash is the internal backend of the Assistant and sub-agents and the Director's replanning backend in the recovery benchmark (Section 3), and Gemini 2.5 Pro is the MLLM judge for automatic video evaluation (Section 3, Appendix D.4); the end-to-end videos of all compared systems are produced with the generative backends listed in Appendix D.3. Among the uses that require disclosure, we used generative AI tools to *generate synthetic data*: the structured chain-of-thought rationales in the SFT split are distilled from Gemini 2.5 Flash conditioned on template-generated target chains (Section 3), and during GRPO the candidate plans are sampled from the Director itself and scored against the reference chains of the GRPO split (Section 2). We also used them to *implement methods*: LLM-based coding assistants helped write parts of the framework code. We have not used generative AI tools to develop the conceptual framework, propose or refine hypotheses, design the methodology or experiments, or interpret results; formulating or proving mathematical claims, translation, dataset cleaning, and qualitative data analysis are not applicable to this work. Among the tasks with recommended disclosure, we used LLM-based assistants to edit the manuscript for readability and length. We have reviewed all AI-assisted work: a random 20% sample of the training instruction-plan pairs and every test case were manually inspected for capability coverage and dependency-consistent ordering (Appendix D.3), all AI-assisted code was reviewed and tested by the authors, and all AI-edited text was checked by the authors for accuracy. We take responsibility for the content of this work, including text, claims, and artifacts produced with generative AI.

### ETHICS STATEMENT

Our human evaluation (Section 3, Appendix G) collected only Likert ratings of generated videos from volunteer raters; no personal or demographic information was recorded, raters could stop at any time, and the rating instrument is reproduced in full. The training and evaluation data consist of template-generated production requests, descriptors of our own sub-agents, and publicly available or newly written story prompts (Section 3); they contain no personal data. FRAMEWORKERS is an orchestration framework and does not introduce a new generative backbone; it inherits the content-safety filters, biases, and potential for misuse (e.g., misleading or infringing video) of the underlying image, video, and language models, which we discuss in Appendix H. We see no conflicts of interest beyond the affiliations listed on the camera-ready version.

### REPRODUCIBILITY STATEMENT

The framework is specified in Section 2, with the per-task execution procedure and a complete sample run in Appendix B, and a sub-agent descriptor exactly as rendered into the Director prompt in Appendix D.2. Appendix D.3 lists the model, LoRA, SFT, and GRPO hyper-parameters, the SFT schema, the full GRPO objective and ordering reward, the set-aware edit distance, the data-quality procedure, and the exact generation backends and versions; Appendix C details the construction of the 991 perturbed workflow states; Appendix D.4 gives the complete MLLM-judge prompt; Appendix G gives the human-study protocol and the full rating instrument. We will release the code, sub-agent descriptors, the SFT and GRPO training data, the DCE and DCH benchmarks, the 991-case perturbation suite, and the two real-generation benchmarks upon publication.

## REFERENCES

Omer Bar-Tal, Hila Chefer, Omer Tov, Charles Herrmann, Roni Paiss, Shiran Zada, Ariel Ephrat, Junhwa Hur, Guanghui Liu, Amit Raj, et al. Lumiere: A space-time diffusion model for video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Tim Brooks, Bill Peebles, Connor Holmes, Will DePue, Yufei Guo, Li Jing, David Schnurr, Joe Taylor, Troy Luhman, Eric Luhman, Clarence Ng, Ricky Wang, and Aditya Ramesh. Video generation models as world simulators. https://openai.com/index/video-generation-models-as-world-simulators/, February 2024. Technical report introducing Sora.

Dongping Chen, Ruoxi Chen, Shilin Zhang, Yaochen Wang, Yinuo Liu, et al. MLLM-as-a-Judge: Assessing multimodal LLM-as-a-judge with vision-language benchmark. In *International Conference on Machine Learning (ICML)*, 2024a.

Haoyu Chen, Wenbo Li, Jinjin Gu, Jingjing Ren, Sixiang Chen, Tian Ye, Renjing Pei, Kaiwen Zhou, Fenglong Song, and Lei Zhu. RestoreAgent: Autonomous image restoration agent via multimodal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Haoyu Chen, Keda Tao, Yizao Wang, Xinlei Wang, Lei Zhu, and Jinjin Gu. Intelligent photo retouching with language model-based artist agents. In *Findings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Zhixun Chen, Ming Li, Yuxuan Huang, Yali Du, Meng Fang, and Tianyi Zhou. ATLaS: Agent tuning via learning critical steps. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25334–25349, 2025.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Zhiqing Cui, Haotong Xie, Jiahao Yuan, Cheng Yang, Hanqing Wang, Yuxin Wu, Yifan Wu, Siru Zhong, Tao Yu, Yifu Guo, Siyu Zhang, Xinlei Yu, Qibing Ren, and Usman Naseem. Uno-Orchestra: Parsimonious agent routing via selective delegation. *arXiv preprint arXiv:2605.05007*, 2026.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. Plan-and-Act: Improving planning of agents for long-horizon tasks. In *International Conference on Machine Learning (ICML)*, 2025.

Yu Gao, Haoyuan Guo, Tuyen Hoang, Weilin Huang, Lu Jiang, Fangyuan Kong, Huixia Li, Jiashi Li, Liang Li, Xiaojie Li, Xunsong Li, Yifu Li, Shanchuan Lin, Zhijie Lin, Jiawei Liu, Shu Liu, Xiaonan Nie, Zhiwu Qing, Yuxi Ren, Li Sun, Zhi Tian, Rui Wang, Sen Wang, Guoqiang Wei, Guohong Wu, Jie Wu, Ruiqi Xia, Fei Xiao, Xuefeng Xiao, Jiangqiao Yan, Ceyuan Yang, Jianchao Yang, Runkai Yang, Tao Yang, Yihang Yang, Zilyu Ye, Xuejiao Zeng, Yan Zeng, Heng Zhang, Yang Zhao, Xiaozheng Zheng, Peihao Zhu, Jiaxin Zou, and Feilong Zuo. Seedance 1.0: Exploring the boundaries of video generation models. *arXiv preprint arXiv:2506.09113*, 2025.

Google DeepMind. Gemini 3 Pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf, 2026.

Jinjin Gu. Position: Agentic systems constitute a key component of next-generation intelligent image processing. *arXiv preprint arXiv:2505.16007*, 2025.

Yingqing He, Menghan Xia, Haoxin Chen, Xiaodong Cun, Yuan Gong, Jinbo Xing, Yong Zhang, Xintao Wang, Chao Weng, Ying Shan, et al. Animate-A-Story: Storytelling with retrieval-augmented video generation. *arXiv preprint arXiv:2307.06940*, 2023.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations (ICLR)*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Haobo Hu, Qi Mao, Yuanhang Li, and Libiao Jin. Camera Artist: A multi-agent framework for cinematic language storytelling video generation. *arXiv preprint arXiv:2604.09195*, 2026.

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, Xintao Wang, Zinan Lin, Xuefei Ning, Pengfei Wan, Di Zhang, Yu Wang, and Xihui Liu. FilMaster: Bridging cinematic principles and generative AI for automated film generation. In *International Conference on Learning Representations (ICLR)*, 2026a.

Lingxuan Huang, Sizhe He, Hengji Zhou, Liqiang Nie, Lianghao Xia, and Chao Huang. ViMax: Agentic video generation. *arXiv preprint arXiv:2606.07649*, 2026b.

Kling Team. Kling-Omni technical report. *arXiv preprint arXiv:2512.16776*, 2025.

Dan Kondratyuk, Lijun Yu, Xiuye Gu, José Lezama, Jonathan Huang, Grant Schindler, Rachel Hornung, Vighnesh Birodkar, Jimmy Yan, Ming-Chang Chiu, et al. VideoPoet: A large language model for zero-shot video generation. In *International Conference on Machine Learning (ICML)*, pp. 25105–25124, 2024.

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

Yunxin Li, Haoyuan Shi, Baotian Hu, Longyue Wang, Jiashun Zhu, Jinyi Xu, Zhen Zhao, and Min Zhang. Anim-Director: A large multimodal model powered agent for controllable animation video generation. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

Zhendong Li, Lei Sun, Ruibo Ming, He Zhang, Danda Pani Paudel, Luc Van Gool, and Jinjin Gu. Knowledge-centric agents for workflow generation in ComfyUI. In *European Conference on Computer Vision (ECCV)*, 2026.

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

Han Lin, Abhay Zala, Jaemin Cho, and Mohit Bansal. VideoDirectorGPT: Consistent multi-scene video generation via LLM-guided planning. In *Conference on Language Modeling (COLM)*, 2024.

Junxiang Liu, Lin Wang, Haiyu Shi, Hongxu Ma, Xiaoyu Yang, Chunjie Chen, Xiaoxiao Xu, Kaiqiao Zhan, Boao Wang, Shuizhou Shi, Tianyun Zhu, Jie Li, and Jiangtong Li. CineForge: Self-improving agents for long-horizon video generation. *arXiv preprint arXiv:2608.29621*, 2026.

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

Chenyu Mu, Xin He, Qu Yang, Wanshun Chen, Jiadi Yao, Huang Liu, Zihao Yi, Bo Zhao, Xingyu Chen, Ruotian Ma, Fanghua Ye, Erkun Yang, Cheng Deng, Zhaopeng Tu, Xiaolong Li, and Linus. The script is all you need: An agentic framework for long-horizon dialogue-to-cinematic video generation. *arXiv preprint arXiv:2601.17737*, 2026.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

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, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Qwen Team. Qwen3.6-Plus: Towards real world agents. https://qwen.ai/blog?id=qwen3.6, April 2026.

Marcelo Sandoval-Castañeda, Bryan Russell, Josef Sivic, Gregory Shakhnarovich, and Fabian Caba Heilbron. EditDuet: A multi-agent system for video non-linear editing. In *SIGGRAPH 2025 Conference Papers*, pp. 1–11, 2025.

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

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Haoyuan Shi, Yunxin Li, Xinyu Chen, Longyue Wang, Baotian Hu, and Min Zhang. AniMaker: Multi-agent animated storytelling with MCTS-driven clip generation. In *SIGGRAPH Asia 2025 Conference Papers*, pp. 1–11, 2025.

Yale Song, Yiwen Song, Nick Losier, Nathan Hodson, Ye Jin, Rhyard Zhu, Yan Xu, Daniel Vlasic, Carina Claassen, Jasmine Leon, Khanh G. LeViet, Zack Chomyn, Joe Timmons, Brett Slatkin, Scott Penberthy, and Tomas Pfister. Co-Director: Agentic generative video storytelling. *arXiv preprint arXiv:2604.24842*, 2026.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via Python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11888–11898, 2023.

Wan Team. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

Jiuniu Wang, Zehua Du, Yuyuan Zhao, Bo Yuan, Kexiang Wang, Jian Liang, Yaxi Zhao, Yihen Lu, Gengliang Li, Junlong Gao, et al. AesopAgent: Agent-driven evolutionary system on story-to-video production. *arXiv preprint arXiv:2403.07952*, 2024.

Qian Wang, Ziqi Huang, Ruoxi Jia, Paul Debevec, and Ning Yu. MAViS: A multi-agent framework for long-sequence video storytelling. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 2026.

Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual ChatGPT: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversations. In *Conference on Language Modeling (COLM)*, 2024.

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

Tianyidan Xie, Mingjie Wang, Qiang Tang, Feixuan Liu, Rui Ma, Lanjun Wang, and Zili Yi. CineAGI: Character-consistent movie creation through LLM-orchestrated multi-modal generation and cross-scene integration. In *IEEE International Conference on Multimedia and Expo (ICME)*, 2026.

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

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

Zhenran Xu, Jifang Wang, Longyue Wang, Zhouyi Li, Senbao Shi, Baotian Hu, and Min Zhang. FilmAgent: Automating virtual film production through a multi-agent collaborative framework. In *SIGGRAPH Asia 2024 Technical Communications*, pp. 1–4, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. MM-REACT: Prompting ChatGPT for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. CogVideoX: Text-to-video diffusion models with an expert transformer. In *International Conference on Learning Representations (ICLR)*, 2025b.

Zhiyuan You, Ke Wang, He Zhang, Xin Cai, Jinjin Gu, Tianfan Xue, Chao Dong, and Zhoutong Zhang. PhotoFramer: Multi-modal image composition instruction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

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.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. Agent-Tuning: Enabling generalized agent abilities for LLMs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3053–3077, 2024.

Chenchen Zhang. Reinforcement learning for LLM-based multi-agent systems through orchestration traces. *arXiv preprint arXiv:2605.02801*, 2026.

Peixuan Zhang, Chang Zhou, Ziyuan Zhang, Hualuo Liu, Chunjie Zhang, Jingqi Liu, Xiaohui Zhou, Xi Chen, Shuchen Weng, Si Li, and Boxin Shi. A benchmark and multi-agent system for instruction-driven cinematic video compilation. In *European Conference on Computer Vision (ECCV)*, pp. 213–230, 2026a.

Xuanyu Zhang, Weiqi Li, Qunliang Xing, Jingfen Xie, Bin Chen, Junlin Li, Li Zhang, Jian Zhang, and Shijie Zhao. VQ-Jarvis: Retrieval-augmented video restoration agent with sharp vision and fast thought. *arXiv preprint arXiv:2603.22998*, 2026b.

Canyu Zhao, Mingyu Liu, Wen Wang, Weihua Chen, Fan Wang, Hao Chen, Bo Zhang, and Chunhua Shen. MovieDreamer: Hierarchical generation for coherent long visual sequences. In *International Conference on Learning Representations (ICLR)*, 2025.

Mingzhe Zheng, Yongqi Xu, Haojian Huang, Xuran Ma, Yexin Liu, Wenjie Shu, Yatian Pang, Feilong Tang, Qifeng Chen, Harry Yang, and Ser-Nam Lim. VideoGen-of-Thought: Step-by-step generating multi-shot video with minimal manual intervention. *arXiv preprint arXiv:2412.02259*, 2024.

Hengji Zhou, Lingxuan Huang, Jian Wang, Bing Zhou, Si Wu, Lianghao Xia, and Chao Huang. VideoAgent: All-in-one framework for video understanding and editing. *arXiv preprint arXiv:2606.23327*, 2026a.

Jinsong Zhou, Yihua Du, Xinli Xu, Luozhou Wang, Zijie Zhuang, Yehang Zhang, Shuaibo Li, Xiaojun Hu, Bolan Su, and Ying-cong Chen. VideoMemory: Toward consistent video generation via memory integration. *arXiv preprint arXiv:2601.03655*, 2026b.

Kaiwen Zhu, Jinjin Gu, Zhiyuan You, Yu Qiao, and Chao Dong. An intelligent agentic system for complex image restoration problems. In *International Conference on Learning Representations (ICLR)*, 2025.

Shaobin Zhuang, Kunchang Li, Xinyuan Chen, Yaohui Wang, Ziwei Liu, Yu Qiao, and Yali Wang. Vlogger: Make your dream a vlog. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8806–8817, 2024.

# APPENDIX

This appendix provides extended implementation details, expanded experimental analysis, and a full description of our evaluation protocols to complement the main paper. Generated videos, their prompts, and the task-level process of each run are available on an anonymous project page: https://papaya-froyo-9745ae.netlify.app.

Appendix A discusses related work. Appendix B gives a procedural walkthrough of the FRAMEWORKERS workflow, illustrating the closed-loop orchestration between the Director, Assistant, and Workspace. Appendix C details the runtime-recovery evaluation and gives a per-category breakdown of failure modes across the four evaluated backends, with emphasis on the two Gemini versions. Appendix D covers implementation details: the single-agent baseline (S-Agent), the sub-agent descriptors, the training and reproducibility settings, and the complete prompt of the MLLM judge. Appendix E compares the input and output coverage of FRAMEWORKERS with prior systems, and Appendix F shows example Director plans. Appendix G describes the user studies and the rating instrument, and Appendix H discusses limitations.

## A EXTENDED RELATED WORK

**Video Generation Backbones.** Recent video generation models have made substantial progress in generating high-quality video clips. Lumiere (Bar-Tal et al., 2024), VideoPoet (Kondratyuk et al., 2024), CogVideoX (Yang et al., 2025b), and Movie Gen (Polyak et al., 2024) advance space-time generation, multimodal token prediction, diffusion-transformer scaling, and high-resolution multi-task media generation. These backbones improve motion coherence, fidelity, editing, personalization, and audio support, but remain clip-centric and do not handle story decomposition, persistent assets, conditional step skipping, or recovery across a production workflow.

**Long-Form and Agentic Production Systems.** Long-form systems add explicit narrative structure. VideoStudio (Long et al., 2024), Vlogger (Zhuang et al., 2024), Animate-A-Story (He et al., 2023), and VideoGen-of-Thought (Zheng et al., 2024) improve multi-scene coherence through scripting, role decomposition, reference conditioning, retrieval-guided motion, or step-by-step shot generation. Agentic systems further automate production planning. VideoDirectorGPT (Lin et al., 2024) expands prompts into structured scene plans, while FilmAgent (Xu et al., 2024), DreamFactory (Xie et al., 2024), AesopAgent (Wang et al., 2024), Mora (Yuan et al., 2024), StoryAgent (Hu et al., 2024), MM-StoryAgent (Xu et al., 2025), MAViS (Wang et al., 2026), AniMaker (Shi et al., 2025), and Anim-Director (Li et al., 2024) coordinate specialized roles across scripting, storyboarding, asset and clip generation, and evaluation. These methods improve coherence and automation, but most still assume fixed or task-specific stage orders. This limitation persists in more recent production systems. ScriptAgent (Mu et al., 2026), MovieAgent (Wu et al., 2025), MovieDreamer (Zhao et al., 2025), VideoMemory (Zhou et al., 2026b), Camera Artist (Hu et al., 2026), CineAGI (Xie et al., 2026), FilMaster (Huang et al., 2026a), and Co-Director (Song et al., 2026) introduce executable acting scripts, director-like roles, hierarchical keyframe prediction, dynamic memory, cinematic language, recursive refinement, audio-visual synchronization, film-corpus-guided camera design and editing, and bandit-guided search over creative directions. ViMax (Huang et al., 2026b) adds retrieval-augmented narrative planning with dependency-aware visual consistency, and CineForge (Liu et al., 2026) evolves stage-targeted production policies across stories. However, they typically target a specific production setting or workflow formulation, or provide only partial support for asset management and scheduling. Existing work thus advances long-form generation but leaves dynamic scheduling over optional steps, heterogeneous assets, and failure recovery open.

**Agentic Visual Media.** General visual-agent frameworks show how LLMs can compose tools and expert modules. Visual ChatGPT (Wu et al., 2023), MM-REACT (Yang et al., 2023), ViperGPT (Surís et al., 2023), HuggingGPT (Shen et al., 2023), AutoGen (Wu et al., 2024), MetaGPT (Hong et al., 2024), and ToolLLM (Qin et al., 2024) establish tool invocation, code and model composition, multi-agent communication, role specialization, and API search. They are important precursors to modular media production, but are domain-agnostic and lack a persistent media asset layer spanning scripts, storyboards, references, clips, and editable files. Media-specific agents begin to add such structure. EditDuet (Sandoval-Castañeda et al., 2025), CineAgents (Zhang et al.,

2026a), UniVA (Liang et al., 2025), VideoAgent (Zhou et al., 2026a), and VQ-Jarvis (Zhang et al., 2026b) use critique loops, hierarchical memory, Plan-and-Act tool use, multi-agent orchestration of editing tools, or operator scheduling for editing, cinematic compilation, general video assistance, video understanding and remaking, and restoration. Agentic designs have also been applied to image-centric media tasks, including autonomous image restoration (Chen et al., 2024b; Zhu et al., 2025), photo retouching and composition (Chen et al., 2026; You et al., 2026), and workflow generation for node-based generation interfaces such as ComfyUI (Li et al., 2026); a recent position paper further argues that agentic systems are a key component of next-generation intelligent image processing (Gu, 2025). These systems demonstrate the value of modular tools, shared state, and iterative planning, but remain tied to narrower downstream regimes rather than open-ended AIGC production from diverse specifications to finished video.

**Agent Training.** Agent-training work motivates fine-tuning a director. AgentTuning (Zeng et al., 2024), Plan-and-Act (Erdogan et al., 2025), and ATLaS (Chen et al., 2025) show that agents benefit from interaction trajectories, planner-executor separation, and learning critical planning steps. Uno-Orchestra (Cui et al., 2026) learns task decomposition and subtask routing jointly from reinforcement-learning trajectories, and Zhang (2026) studies reinforcement learning over the orchestration traces of LLM-based multi-agent systems. For video production, where scheduling errors propagate across story, storyboard, generation, and editing, this motivates training the Director rather than relying on prompting alone.

FRAMEWORKERS differs in combining dynamic scheduling over optional steps, a Workspace-grounded asset layer, and a trained routing policy; Appendix E tabulates the resulting input and output coverage against prior systems.

## B MORE DETAILS ABOUT THE FRAMEWORKERS WORKFLOW

As shown in Figure 5 (and, for a complete sample run, Figure 6), FRAMEWORKERS operates as a closed-loop, autonomous multi-agent framework for open-ended video creation, continuously cycling through planning, execution, evaluation, and replanning. The workflow begins when a user provides instructions through the system interface, which are then forwarded to the central Director. The Director evaluates this new user message alongside the current status of the Task Stack and the overall execution state. Using the available sub-agent catalog, the Director determines the required operations and updates the Task Stack accordingly.

Once the Task Stack is updated, the Director identifies the next executable task and dispatches it to the Assistant. Serving as the execution runtime, the Assistant inspects the input requirements of the sub-agent assigned to the job and retrieves the necessary content and artifacts from the shared Workspace. It then provides these formatted inputs to the sub-agent, which executes its generation or processing task. After the sub-agent completes its operation, the Assistant retrieves the results, stores them, and logs the detailed execution traces back into the Workspace.

After execution, the Assistant runs the evaluator: it gathers the necessary context from the Workspace, applies the evaluator's checks to the produced artifacts (schema validation, rule checks, and, for most text sub-agents, an LLM review; an output that fails a check, or a failed media generation, is regenerated with feedback, up to three times), and stores the evaluation results in the shared Workspace. Finally, the Assistant compiles a compact task and evaluation summary and returns it to the Director. Given the updated state and feedback, the Director decides whether to continue, revise the plan, or recover from failure, and updates the Task Stack and status messages accordingly. The interface periodically polls for these updates, fetches new user information if available, and displays all generated changes to the user, continuing this closed-loop cycle until the user's goal is achieved.

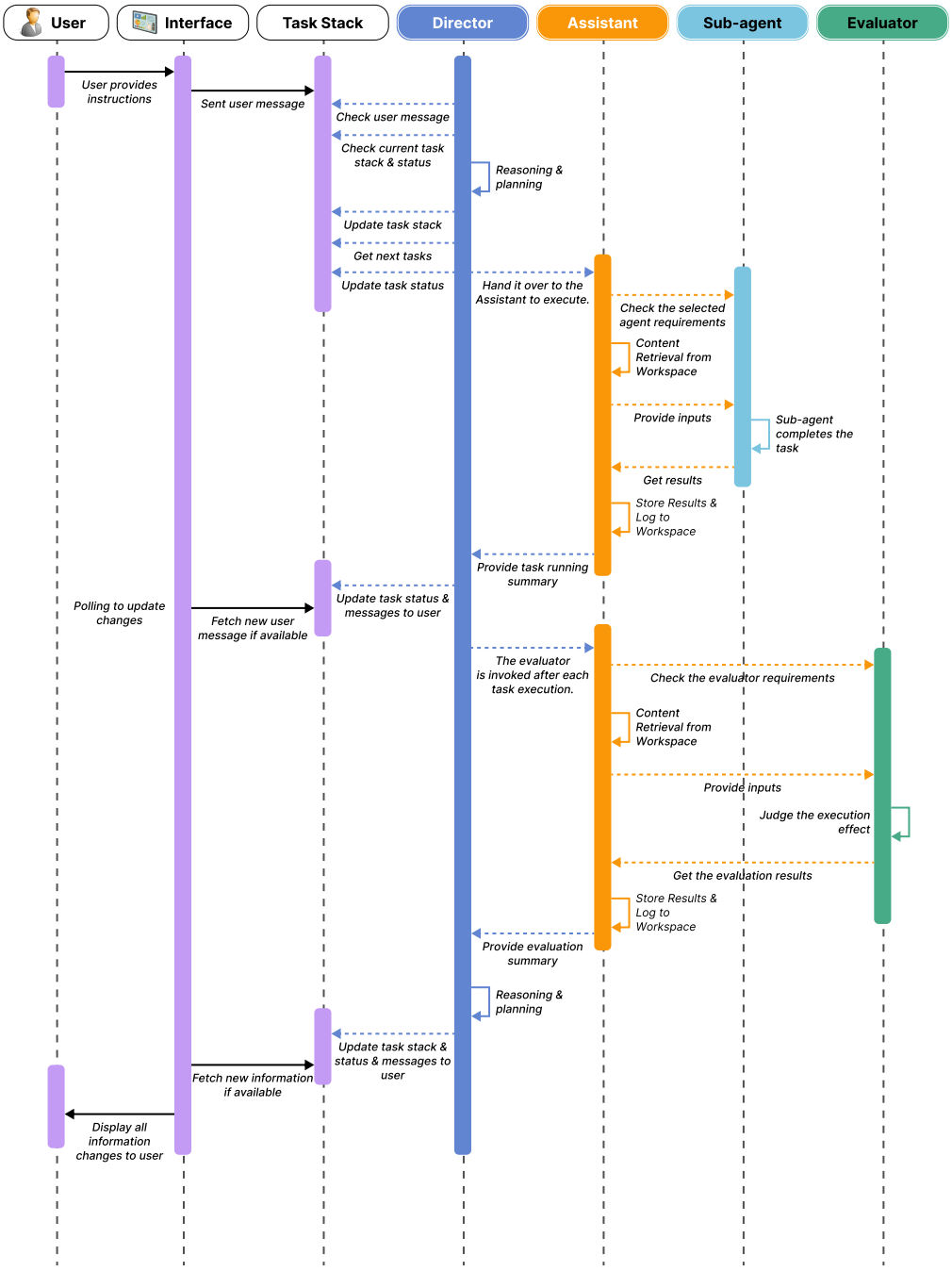


Figure 5: Sequence diagram of one closed-loop iteration: the Director updates the Task Stack, the Assistant runs the next task's sub-agent and then the evaluator, and the Director replans from the returned summaries; the text below walks through each step.

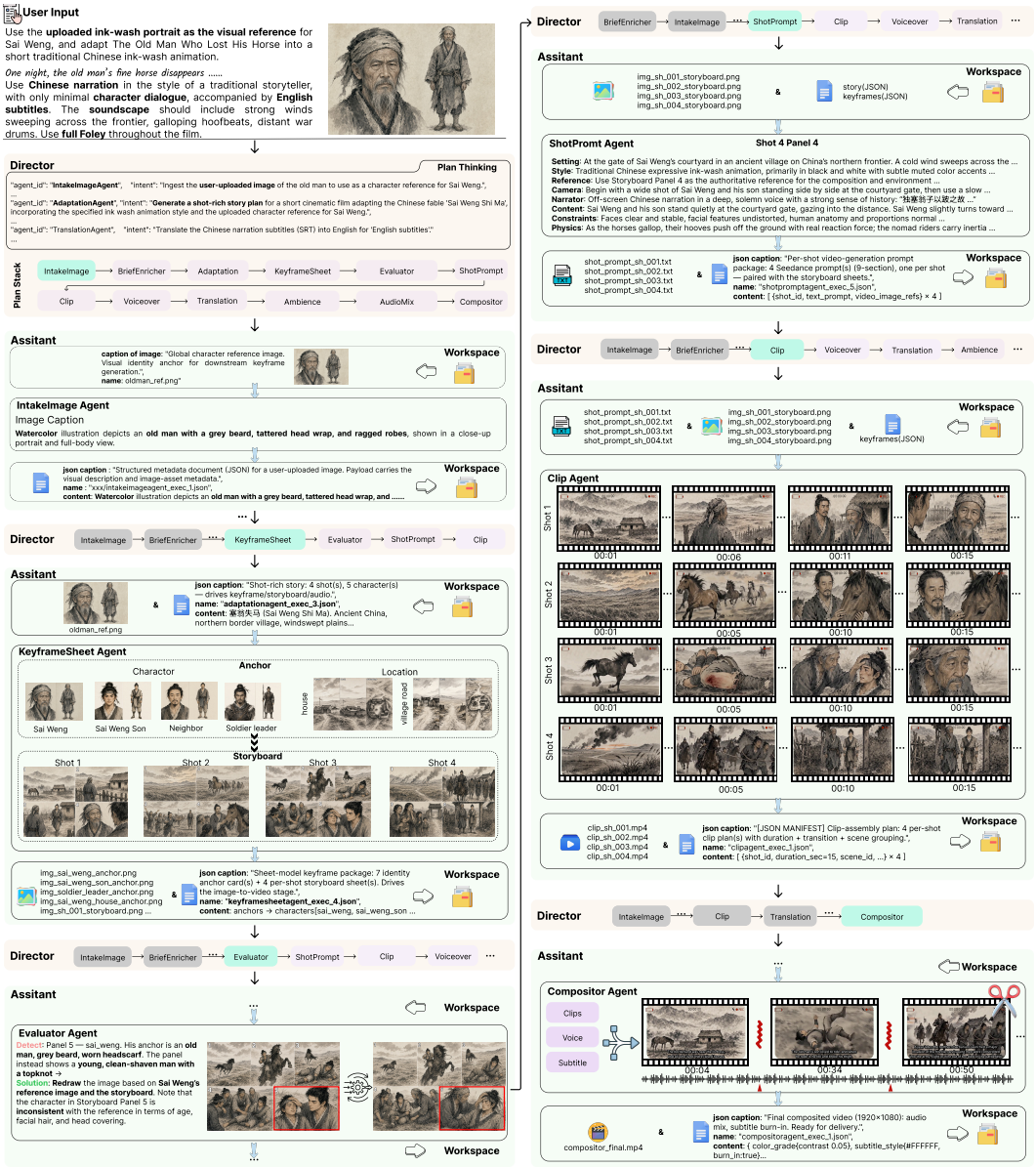


Figure 6: Sample run for the instruction "The Old Man Who Lost His Horse" with a reference portrait of the protagonist. Each Director row shows the Task Stack (*Plan Stack*) with the current task highlighted; each Assistant block shows the inputs retrieved from the Workspace, the sub-agent's output, and what is written back. When the evaluator detects drift, here a keyframe whose protagonist no longer matches the reference, the responsible sub-agent regenerates that output. Agents appear under their implementation names (KeyframeSheet, ShotPrompt, and Clip correspond to KeyFrame, Screenplay, and Video in the inventory).

# C Robustness and Failure Modes

Table 6: Runtime recovery evaluation on 991 perturbed cases. The Director replans with Gemini 2.5 Flash as its backend, without recovery-specific training.

| Error Type | N | Recovery Success↑ |
|---|---|---|
| Missing producer (plan-level) | 495 | 99.8% (494) |
| Quality-gate failure (execution-level) | 496 | 93.5% (464) |
| Overall | 991 | 96.7% (958) |

**Perturbation Construction (Section 3).** We construct 991 perturbed workflow states using two representative failure types. For *plan-level perturbations*, we remove a required producer task while retaining a downstream task that depends on its output. The resulting Task Stack is structurally incomplete, and successful recovery requires the replanning mechanism to identify the missing dependency and insert an appropriate producer before the affected consumer. For *execution-level perturbations*, a quality gate rejects an artifact produced by a completed task. The Assistant records the failure type and returns a structured failure summary, which is combined with the current Task Stack and the available sub-agent descriptors to determine a corrective action, such as regeneration or refinement, before downstream execution continues. In both settings, completed execution history is preserved, and only the unexecuted portion of the Task Stack can be revised through the standard ADD and DELETE operations.

Table 7 decomposes failure distributions across the four backends of Section 3; we focus on the two Gemini variants. For Gemini 2.0 Flash, failures are spread over several categories, especially *audio overlay* (66.7% pass), *style transfer* (85.7%), *storytelling* (86.1%), *bilingual subtitle* (88.9%), *creative* (91.7%), and *complex* (92.9%), suggesting that long-horizon multimodal coordination remains challenging under deeper generation chains. Step-level failures are likewise distributed across multiple agents, particularly BriefEnricherAgent (88.1%), AudioMixAgent (93.8%), and MusicAgent (96.9%), indicating instability in long-context planning and audio-scene synchronization.

In contrast, Gemini 2.5 Flash passes all 204 cases and completes every planned step: the five cases that required rework (Table 3) are all recovered within the rework loop, so no category or agent falls below 100%. In this controlled execution suite, Gemini 2.5 Flash thus shows none of the unrecovered cross-agent coordination failures observed with Gemini 2.0 Flash.

# D More Implementation Details

## D.1 Single-agent Baseline (S-Agent)

To establish a solid baseline for comparison, we implement a single-agent system, denoted as S-Agent. Unlike our proposed framework, the S-Agent operates on a straightforward, sequential pipeline without multi-agent collaboration. Given a user instruction, the S-Agent uses a single MLLM call for the planning phase. Specifically, the MLLM is prompted to decompose the high-level instruction into a sequential list of individual shots and simultaneously generate detailed textual prompts for the keyframe of each shot. Following this planning stage, the system executes a two-step visual generation process. First, a Text-to-Image (T2I) model is employed to synthesize the keyframes based on the generated shot prompts. Subsequently, an Image-to-Video (I2V) model takes these synthesized keyframes as starting points to render the final video segments. The I2V model keeps each shot internally consistent, but nothing links different shots.

## D.2 Sub-agent Descriptors

Each sub-agent is exposed to the Director through a plain-text descriptor containing three fields. `Inputs` specifies the artifact labels consumed by the sub-agent. Each input entry includes its cardinality (`single` or `collection`), whether it is `required` or `optional`, and a natural-language description that specifies the expected artifact and, when needed, provides guidance for resolving it from the Workspace. The Assistant uses this information to retrieve and select concrete inputs from the Workspace. `Output` specifies the artifact produced by the sub-agent, while `Purpose / Trigger` describes when the sub-agent should be invoked.

Table 7: Per-category breakdown of the execution suite of Table 3, sorted worst-first by GPT-4o; categories are those of Figure 8. We report case-level pass rate and step-level completion rate (completed over attempted steps); $N$ is the number of cases or of planned steps. Agents at 100% on all backends are grouped as *Other*.

| Category | Metric | N | GPT-4o | GPT-4o-mini | Gemini 2.0 Flash | Gemini 2.5 Flash |
|---|---|---|---|---|---|---|
| highlight reel | case | 24 | 0.00 | 91.67 | 95.83 | 100.00 |
| creative | case | 12 | 41.67 | 75.00 | 91.67 | 100.00 |
| bilingual subtitle | case | 9 | 55.56 | 88.89 | 88.89 | 100.00 |
| complex | case | 42 | 61.90 | 88.10 | 92.86 | 100.00 |
| video extend | case | 21 | 66.67 | 85.71 | 100.00 | 100.00 |
| creative subtitle | case | 6 | 66.67 | 66.67 | 100.00 | 100.00 |
| video subtitle | case | 6 | 66.67 | 100.00 | 100.00 | 100.00 |
| storytelling | case | 36 | 75.00 | 55.56 | 86.11 | 100.00 |
| image-ref creative | case | 21 | 80.95 | 80.95 | 95.24 | 100.00 |
| style transfer | case | 21 | 85.71 | 90.48 | 85.71 | 100.00 |
| audio overlay | case | 6 | 100.00 | 100.00 | 66.67 | 100.00 |
| HighlightAgent | step | 33 | 0.00 | 96.97 | 100.00 | 100.00 |
| TranslationAgent | step | 54 | 74.19 | 100.00 | 100.00 | 100.00 |
| NarrationAgent | step | 36 | 80.00 | 100.00 | 100.00 | 100.00 |
| StoryAgent | step | 66 | 83.87 | 100.00 | 100.00 | 100.00 |
| BriefEnricherAgent | step | 42 | 88.10 | 100.00 | 88.10 | 100.00 |
| VideoExtendAgent | step | 45 | 93.02 | 100.00 | 100.00 | 100.00 |
| KeyFrameAgent | step | 66 | 94.00 | 100.00 | 100.00 | 100.00 |
| MusicAgent | step | 102 | 95.59 | 100.00 | 96.94 | 100.00 |
| VideoAgent | step | 66 | 95.74 | 100.00 | 98.46 | 100.00 |
| ScreenplayAgent | step | 66 | 96.15 | 100.00 | 100.00 | 100.00 |
| AudioMixAgent | step | 120 | 98.75 | 76.27 | 93.75 | 100.00 |
| CompositorAgent | step | 192 | 99.15 | 95.65 | 100.00 | 100.00 |
| AmbienceAgent | step | 63 | 100.00 | 96.83 | 98.33 | 100.00 |
| Other (step) | step | 396 | 100.00 | 100.00 | 100.00 | 100.00 |

Importantly, each descriptor describes only its own sub-agent and does not explicitly refer to other sub-agents. Instead, dependencies between sub-agents are implicit in their input and output artifact labels: an artifact produced by one sub-agent may satisfy an input required by another. The Director is responsible for inferring these dependencies during planning.

At planning time, the descriptors of all registered sub-agents are rendered verbatim into the Director prompt as a capability catalog, preceded by the routing rules and the list of allowed identifiers. The same descriptor representation is used during both training and inference. In the configuration of Section 3, individual descriptors range from approximately 0.6K to 3.8K characters, while the full catalog contains about 30K characters. Thus, the prompt is dominated by sub-agent descriptors rather than by handcrafted examples, and it contains no example plans. Figure 7 shows one descriptor exactly as rendered into the Director prompt. The descriptor schema and the sub-agent implementations will be released.

**StoryAgent descriptor**

```
StoryAgent
  - Inputs:
      [creative_brief] (single, required): A natural-language brief describing what kind
      ↪  of story / video to produce. May be a short prompt ('a film about a cat
      ↪  chasing a butterfly') or a longer detailed outline / draft story text. The
      ↪  caption describes it as a creative brief / project intent description. If an
      ↪  enriched brief (one that integrates image reference descriptions) is
      ↪  available, prefer it over the raw text-intake brief. Pick the single most
      ↪  recent / most authoritative such brief.
      [reference_analysis] (single, optional): Optional scene-level video-analysis
      ↪  report of an inspiration / reference video. Use cases include 'analyse this
      ↪  hit drama and write me a same-genre sequel / similar new story' -- the LLM
      ↪  reads its genre, mood, scene summaries, and entities as creative seeds to
      ↪  shape tone and pacing of the NEW blueprint, without copying plot verbatim.
      ↪  Skip this label when there is no reference video being analysed.
  - Output: story_blueprint (logline, cast, locations, story_arc, scene_outline).
```

```
- Purpose / Trigger: First creative step for any task that produces a NEW film from
↪  scratch. Trigger: requests to create new film content (mini-drama, manhua,
↪  trailer, vertical short, etc.) -- user provides a creative brief and is not asking
↪  to transform / edit an existing source video. A reference video used purely as
↪  creative inspiration (for genre / mood seeding via an upstream video analysis
↪  report) is allowed and does not disqualify this trigger.
```

Figure 7: The complete StoryAgent descriptor, exactly as rendered into the Director prompt (`Inputs`, `Output`, `Purpose / Trigger`).

### D.3 Training and Reproducibility Details

**Director Training.** The Director is initialized from Qwen3-4B or Qwen3-8B and fine-tuned with LoRA adapters (rank 32, $\alpha = 64$, dropout 0.05; applied to all attention and MLP projections), with reasoning ("thinking") disabled. For SFT we use a learning rate of $2 \times 10^{-5}$, an effective batch size of 8 (per-device batch 1, gradient accumulation 4, 2 GPUs), 4 epochs, and a maximum sequence length of 12,288 tokens; the loss is computed on assistant response tokens only, with the prompt masked out. For GRPO we use group size $G = 4$, reward weights $\lambda_{\text{val}} = 0.1$, $\lambda_{\text{cov}} = 0.5$, $\lambda_{\text{ord}} = 0.4$, KL coefficient $\beta = 0.04$ (K3 estimator), sampling temperature 0.9 (top-$p = 1.0$, top-$k$ disabled), maximum 1,024 new tokens, and a learning rate of $1 \times 10^{-6}$, optimizing against a frozen SFT reference policy for 4 epochs. Updates are single-step and on-policy (one step per group).

**Data Quality.** The training data spans eleven routing categories, as shown in Figure 8, and the effect of different reasoning supervision formats is evaluated in Table 2. To assess data quality, we manually inspect a random 20% sample of the training instruction-plan pairs, judging each pair by required capability coverage and dependency-consistent ordering rather than by exact sequence match, since a production request may admit multiple valid sub-agent orderings; every DCE and DCH test case is manually verified. Nevertheless, the generated data may still inherit biases from the predefined routing categories, generation rules, and the sub-agent inventory.

**SFT Objective.** Each SFT target follows the schema

```
{"rationale":"...", "plan":[
  {"agent_id":"...", "intent":"..."}]}
```

and is trained with the SFT loss of Section 2.

**Ordering Reward.** With $E(p^*)$ the prerequisite edges of the target plan and $\text{pos}_{\hat{p}}(\cdot)$ the position in the predicted plan, the ordering reward of Section 2 is

$$R_{\text{ord}}(\hat{p}, p^*) = \begin{cases} 1, & |E(p^*)| = 0, \\ \frac{1}{|E(p^*)|} \sum_{(a,b) \in E(p^*)} \mathbf{1}\left[ \begin{array}{c} a, b \in S(\hat{p}) \\ \text{pos}_{\hat{p}}(a) < \text{pos}_{\hat{p}}(b) \end{array} \right], & |E(p^*)| > 0. \end{cases}$$

**GRPO Objective.** We give the full objective summarized in Section 2. For each prompt group we normalize rewards among the $G$ sampled responses. Let $R_g = R(\hat{y}_g)$. The relative advantage is

$$A_g = \frac{R_g - \text{mean}(R_{1:G})}{\text{std}(R_{1:G}) + \epsilon_{\text{adv}}}.$$

Here $\epsilon_{\text{adv}}$ is a small constant for numerical stability. This sequence-level advantage is broadcast to all completion tokens. Because each sampled group is used for a single on-policy gradient step, the importance ratio between the current and sampling policies is identically one, and no clipping is needed. The objective is a group-normalized policy gradient with a KL anchor to the frozen SFT reference $\pi_{\text{ref}}$:

$$L_{\text{GRPO}} = -\frac{1}{|T|} \sum_g \sum_{t \in T_g} A_g \log \pi_\theta(\hat{y}_{g,t} \mid x, \hat{y}_{g,<t}) + \beta \frac{1}{|T|} \sum_g \sum_{t \in T_g} \widehat{\text{KL}}_{\text{K3},g,t},$$

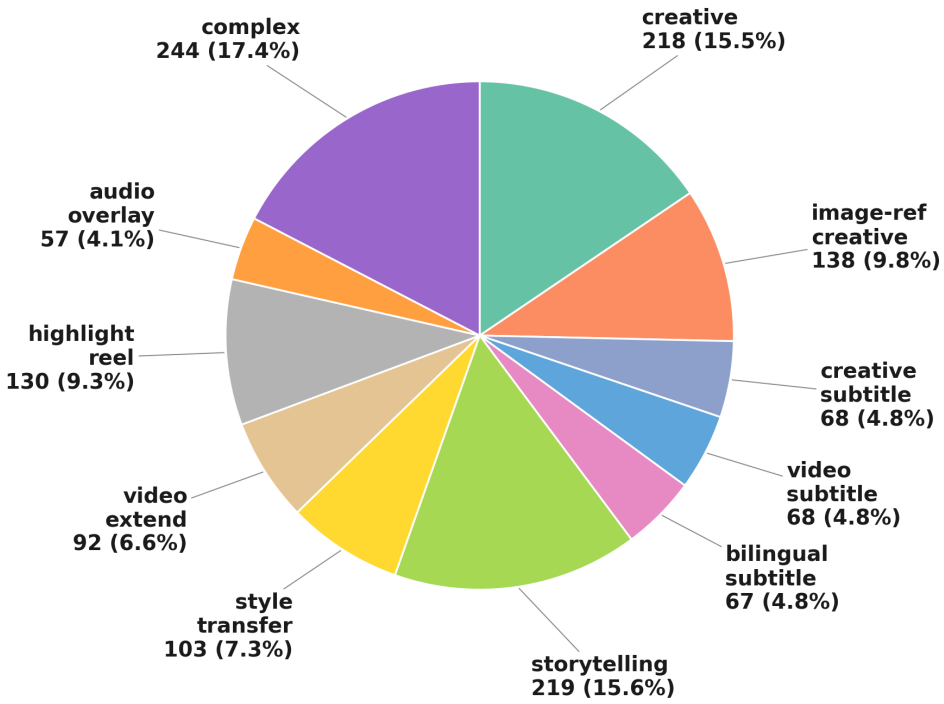


Figure 8: Category distribution of the SFT split (1,404 instruction-plan pairs).

where $T_g$ denotes the completion-token positions of $\hat{y}_g$, $T = \bigcup_g T_g$ ranges over all completion tokens in the batch, and $\beta$ weights the KL penalty, for which we use the non-negative K3 estimator:

$$\widehat{\mathrm{KL}}_{\mathrm{K3},g,t} = \exp(\Delta_{g,t}) - \Delta_{g,t} - 1,$$
$$\Delta_{g,t} = \log \pi_{\mathrm{ref}}(\hat{y}_{g,t} \mid x, \hat{y}_{g,<t}) - \log \pi_\theta(\hat{y}_{g,t} \mid x, \hat{y}_{g,<t}).$$

When all candidates in a group receive identical rewards, the normalized advantage carries no learning signal, and the group is excluded from the update.

**Edit Distance (Section 3).** Specifically, let $a = (a_1, \ldots, a_n)$ denote the predicted agent plan and $e = (E_1, \ldots, E_m)$ the ground-truth chain of *slot sets*, where $\mathcal{A}$ is the sub-agent inventory and each $E_j \subseteq \mathcal{A}$ contains all acceptable agents for step $j$. We compute Edit as the standard Levenshtein distance, except that a predicted agent matches a reference slot whenever $a_i \in E_j$:

$$d_{i,j} = \min\Big\{ d_{i-1,j} + 1,\ d_{i,j-1} + 1,$$
$$d_{i-1,j-1} + \mathbb{1}[a_i \notin E_j] \Big\}, \qquad d_{i,0} = i,\ d_{0,j} = j.$$

The final Edit score is $d(a, e) = d_{n,m}$, where lower is better, and $d(a, e) = 0$ if and only if the predicted plan has the same length as the reference and every predicted agent belongs to the corresponding reference slot.

**Backends and Versions.** Unless otherwise stated, Gemini 2.5 Flash is the LLM of the Assistant, the sub-agents, and all end-to-end baselines; it does not produce FRAMEWORKERS' plans, which come from the fine-tuned Director, except that it is the Director's replanning backend in the recovery benchmark (Section 3). The generation backends of the main benchmark are `gemini-3-pro-image-preview` (2K resolution) for text-to-image and a

self-hosted HunyuanVideo-I2V (Kong et al., 2024) server (25 denoising steps) for image-to-video. The MLLM judge is Gemini 2.5 Pro. On the story-level benchmark all systems use `gemini-2.5-flash-image` and `veo-3.1-fast-generate-preview` (native audio, clips $\leq$8 s), and every film is 48 s long. Table 1 queries `gpt-4o`, `deepseek-v3`, `qwen3.6-plus`, `gemini-2.5-pro`, and `gemini-3-pro-preview`.

### D.4 Prompt for the MLLM Judge

The judge reports six 1–5 scores: semantic content accuracy (C1), multi-object and spatial relation consistency (C2), action accuracy (C3), story richness and scene diversity (C4), style and cinematic consistency (C5), and overall video-text consistency (C6). We use the prompt in Figure 9:

**Prompt for MLLM-as-a-judge**

```
You are a rigorous multi-modal video evaluation expert (MLLM as a judge). Based only on
↪  the provided frames/timestamps and text/control information, evaluate a single video
↪  with structured scoring and traceable evidence. Do not hallucinate unseen content.

C1. Semantic Content Accuracy (Objects & Scene)
- What to check:
  - Are the specified object categories present and correct?
  - Is the overall scene type (nature/city/indoor/outdoor/weather/terrain) correct and
↪  stable?
  - Are major scene elements consistent with the text prompt and visual context?
- Typical evidence: timestamps where required objects/scenes appear (or fail), brief
↪  notes on correctness.
- Anchors:
1: Objects/scenes largely wrong or missing; persistent mismatch in most segments.
2: Frequent mismatches; objects or scene type often incorrect or unstable.
3: Mostly correct but with noticeable lapses (e.g., brief wrong class or scene drift).
4: Correct and stable with only minor slips in a few moments.
5: Fully correct and stable throughout; no contradictory frames observed.

C2. Multi-Object & Spatial Relations
- What to check:
  - Are object counts correct and stable?
  - Are spatial relations (above/below, inside/outside, left/right, front/back) correct
↪  and perspective-consistent?
  - Are occlusions and object interactions physically plausible?
- Typical evidence: frames showing relation satisfaction/violation (e.g., "cup above
↪  plate").
- Anchors:
1: Major errors in count/placement; relations frequently wrong or contradictory.
2: Multiple wrong relations or unstable layouts; occlusion frequently implausible.
3: Largely correct with occasional conflicts or transient misplacements.
4: Almost entirely correct; rare, minor inconsistencies.
5: Fully correct and stable; relations clear and consistently maintained.

C3. Action / Behavior Accuracy (Human or Specified Agent)
- What to check:
  - If the prompt specifies actions or poses (e.g., "running", "waving"), are they
↪  recognizable and temporally continuous?
  - Are motion trajectories and behavior transitions coherent?
  - Is the intended action maintained throughout the relevant duration?
  - If no action is specified, set null.
- Typical evidence: timestamps covering onset, continuity, and completion of the action.
- Anchors:
1: Action absent or clearly wrong most of the time.
2: Frequent mismatches or fragmentation; hard to recognize the intended action.
3: Generally matches, but with noticeable distortions or brief interruptions.
4: Clear and continuous match, with minor imperfections only.
5: Strong, consistent match; clear start-to-end execution with no ambiguity.

C4. Story Richness & Scene Diversity
- What to check:
  - Does the video present diverse scenes, locations, or environments rather than
↪  repetitive content?
  - Is there meaningful narrative or event progression across the video?
  - Are transitions between scenes, events, or cinematic moments coherent and visually
↪  engaging?
  - Does the video maintain viewer interest through evolving visual storytelling rather
↪  than static repetition?
- Typical evidence: timestamps showing scene transitions, new environments, evolving
↪  events, or repeated/redundant content.
- Anchors:
```

```
1: Extremely repetitive or static; little to no scene variation or narrative
↪  progression.
2: Limited diversity; scenes or events repeat frequently with weak progression.
3: Moderate scene diversity and storytelling progression, but with noticeable repetition
↪  or underdeveloped transitions.
4: Rich and coherent storytelling with diverse scenes and only minor repetition or
↪  transition issues.
5: Highly engaging and diverse visual storytelling with coherent progression, varied
↪  scenes, and strong cinematic development throughout.

C5. Style Consistency (Appearance & Cinematic Movement)
- What to check:
  - Does the visual appearance style (e.g., oil painting, cyberpunk, monochrome) match
↪  the prompt and remain temporally consistent?
  - Do camera movements (zoom, pan, dolly, tilt, handheld, etc.) match the intended
↪  cinematic style?
  - Are visual aesthetics and motion grammar stable throughout the video?
- Typical evidence: timestamps showing style adoption, camera behavior, or temporal
↪  drift.
- Anchors:
1: Style severely mismatched or mostly absent; camera grammar opposite or missing.
2: Frequent mismatches or drift in either appearance or camera style.
3: Generally matches with occasional drift or brief instability.
4: Clear and consistent match with only slight, rare issues.
5: Fully consistent in both appearance and camera grammar throughout.

C6. Overall Video-Text Consistency (set null if no text prompt)
- What to check:
  - Does the video holistically satisfy the text prompt?
  - Are scene, actions, style, objects, and narrative progression mutually coherent?
  - Does the overall semantic impression align with the intended theme and user
↪  instruction?
  - Avoid double-counting fine-grained issues already covered above.
- Typical evidence: timestamps representing core theme fulfillment or contradictions.
- Anchors:
1: Largely mismatched; core theme or requirements not met.
2: Many inconsistencies across key elements (theme/scene/action/style).
3: Mostly correct with noticeable errors in secondary aspects.
4: Overall consistent with small mismatches that do not change the theme.
5: Highly consistent; strong semantic agreement with the text prompt.



[Reference text for evaluation]
{target_text}

[Media context]
{media_description}

Respond with a JSON object containing at minimum a numeric ``score`` and a
string ``reasoning``. Optional ``subscores`` may carry per-dimension C1..C6
scores. Do not wrap the JSON in extra commentary.
```

Figure 9: Prompt given to the MLLM judge (Gemini 2.5 Pro) to score C1–C6 in Table 5.

Table 8: Property-level capability comparison of representative academic video-production systems. A checkmark means the paper explicitly implements or demonstrates the input configuration or output property (for an input combination, all listed modalities are consumed within a single end-to-end workflow); a cross means no such support is demonstrated.

| Dimension | Property | FrameWorkers (Ours) | UniVA | Co-Director | Movie Agent | Anim-Director | Ani Maker | MM-Story Agent |
|---|---|---|---|---|---|---|---|---|
| Supported Inputs | Instruction Only | ✓ | ✓ | ✓ | ✗ | ✓ | ✓ | ✓ |
| | Instruction + raw Source Text | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| | Instruction + Script | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| | Instruction + Reference Image(s) | ✓ | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ |
| | Instruction + Video Asset(s) | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| | Instruction + Audio Asset(s) | ✓ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| | Instruction + Script + Reference Image(s) | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| | Instruction + Script + Audio Asset(s) | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| | Instruction + Reference Image(s) + Video Asset(s) | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| | Instruction + Script + Reference Image(s) + Audio Asset(s) | ✓ | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Output Properties | Multi-shot Dynamic Video | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| | Image-sequence / Storybook Video | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |
| | Refined Existing Video | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ |
| | Speech / Voice-over Audio | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Music / Foley / Sound Effects | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| | Subtitles | ✓ | ✓ | ✗ | ✓ | ✗ | ✓ | ✗ |
| | Final Audiovisual Composition | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

*Notes.* "Reference Image(s)" denotes user-provided character, product, logo, style, scene, or keyframe images that serve as visual anchors the generated footage must remain consistent with. "Video Asset(s)" denotes user-provided videos consumed either as editable source footage or as motion, camera, temporal, or visual references. "Audio Asset(s)" denotes user-provided voice, narration, dialogue, music, or sound-effect recordings consumed as content- or timing-defining input. "Refined Existing Video" denotes instruction-guided editing, temporal extension, recomposition, or quality enhancement of a user-supplied video (e.g., highlight recomposition, resolution upscaling, or tail continuation). For grouped audio properties, a checkmark indicates explicit support for at least one listed non-speech audio type (e.g., natively generated foley/ambience, or a muxed background-music track). "Final Audiovisual Composition" denotes the automatic assembly and synchronization of visual clips, voice-over, Foley or music, and subtitles into a single finished video.

## E Capability Comparison with Prior Systems

A key design goal of FrameWorkers is to decouple video-production capabilities from a fixed pipeline and a predefined input-output regime. Each capability is encapsulated as a self-contained sub-agent and registered through a semantic descriptor that specifies its functional role, required inputs, produced outputs, and trigger conditions, so new capabilities can be incorporated without redesigning the existing framework. At runtime, the Director dynamically selects and organizes sub-agents according to the user request, available multimodal assets, and current execution state, while the Assistant resolves and transfers heterogeneous artifacts through the shared Workspace. Table 8 compares the resulting input combinations and output properties supported by FrameWorkers with those of representative academic video-production systems: UniVA (Liang et al., 2025), Co-Director (Song et al., 2026), MovieAgent (Wu et al., 2025), Anim-Director (Li et al., 2024), AniMaker (Shi et al., 2025), and MM-StoryAgent (Xu et al., 2025).

# F QUALITATIVE RESULTS

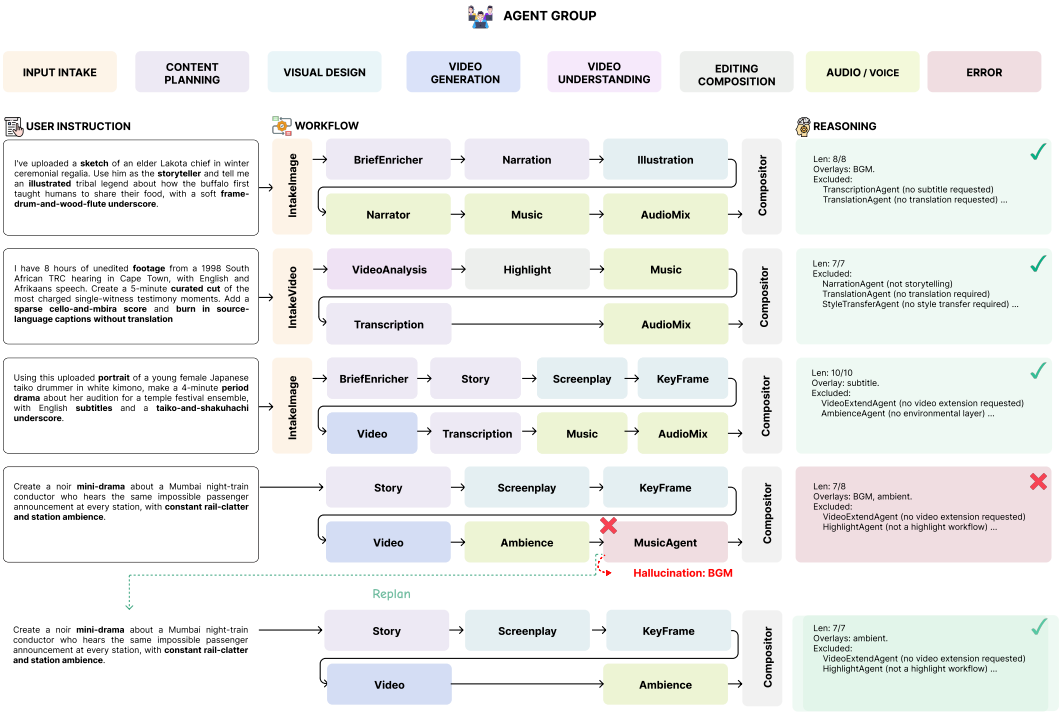


Figure 10: Example Director plans for four requests. Each row shows the instruction, the planned workflow colored by agent group (legend at top), and the Director's reasoning: chain length, requested overlays, and excluded agents. The first three plans are correct; the fourth adds background music to an ambience-only request, and after evaluation and replanning the corrected plan drops it.

# G USER STUDY

**Participants and Procedure.** Two studies follow the same protocol. The main benchmark contains 20 cases, each produced by S-Agent, UniVA, and FRAMEWORKERS; the story-level benchmark contains a further 20 cases produced by FRAMEWORKERS, MovieAgent, and Anim-Director. Raters were unpaid volunteers, and no personal or demographic data was recorded. Each rater was assigned a deterministic, identity-seeded subset of 10 of the 20 cases, which kept a session within about 30 minutes; for every assigned case the three system outputs were shown side by side in random order together with the prompt, and the rater scored each on the nine sub-questions below (eight on the story-level benchmark, whose prompts contain no explicit control signals). Raters could save their progress at any time, and each rater's latest saved ratings were used. The main benchmark collected 3,715 ratings, giving 137–139 (rater, case) pairs per comparison, and the story-level benchmark 1,276 ratings, giving 53 pairs per comparison. The Wilcoxon tests use (rater, case) pairs as units; because one rater contributes up to 10 pairs, which are not independent, the confidence intervals resample raters instead of pairs. As a rater-level check, a Wilcoxon test over per-rater mean differences keeps all three dimension-level gains over S-Agent significant (Holm-corrected $p \leq 0.013$) but not those over UniVA ($p = 0.065$ to $0.13$), and on the story-level benchmark no dimension remains significant after Holm correction (smallest uncorrected $p = 0.031$, instruction following against both baselines). One main-benchmark rater is an author; excluding that session leaves the ordering of the three systems unchanged.

**Rating Instrument.** Each *case–system* tuple was rated on a 1–5 Likert scale across three dimensions, each decomposed into three sub-questions, nine in total. For each dimension, raters first saw a short hint and then answered each sub-question independently. All definitions were displayed in the rater's selected language. We show the English version below.

*Instruction following.* This dimension measures adherence to the input prompt. Controllability-related phrases in the prompt, including BGM, subtitles, ambient sound, style tags, and curated keywords, were highlighted in red to anchor the rater's attention.

- **Controllability.** Does the video respond to explicit control signals in the prompt, such as BGM, subtitles, ambient sound, and style tags? A score of 5 indicates that every requested control signal is clearly produced, whereas a score of 1 indicates that none of them appear.
- **Prompt adherence.** Does the video content match the prompt, including the described subjects, actions, setting, time, and story arc? When both a text prompt and a reference image were provided, raters were told to prioritize the text prompt over the reference image.
- **No irrelevant frames.** Does the video remain on-prompt throughout its entire duration? Higher scores indicate fewer frames or sequences that were not requested by the prompt.

*Temporal consistency.* This dimension measures the stability of the depicted world across frames. Raters were asked to focus on how characters, objects, and the environment evolve over time.

- **Person stability.** Faces, identities, and body proportions of characters remain consistent across frames.
- **Object stability.** Props, clothing, and objects keep their shape, appearance, and position over time.
- **Environment stability.** The background, lighting, and scene layout remain coherent across frames, without brightness, color, or texture flicker.

*Realism.* This dimension measures how natural the video looks and moves. Raters were instructed to penalize obvious out-of-place or unnatural moments.

- **Static realism.** Per-frame photorealism or stylistic fidelity. When the prompt specifies a style, such as animated short drama or wuxia/xianxia (Chinese martial-arts / immortality-fantasy genres), the video is judged against that style rather than against photorealism. Uncanny or broken artifacts lower the score.
- **Motion logic.** Body and object motion should flow smoothly and follow physical and logical patterns, including gravity, collisions, and cloth or fluid dynamics. Warping, teleportation, and impossible movements lower the score. Videos with little or no motion receive a low score; this was stated explicitly so that raters do not reward static outputs.
- **Camera.** Camera movement, such as pans, dollies, or focal changes, should feel natural and intentional. As with motion logic, little or no camera movement receives a low score; this was stated explicitly to avoid rewarding locked-off cameras.

Beyond the rubric, raters received three procedural reminders through an in-app help popup, which opened automatically once per device and could be reopened from a "?" button:

- For instruction-following questions, read the prompt carefully, with special attention to the red-highlighted phrases, before scoring.
- For temporal-consistency questions, watch how persons, objects, and the environment change across the video.
- For realism questions, listen to the audio track when BGM, subtitles, or ambient sound are mentioned, and note any moment that feels obviously out of place.

## H Limitations and Discussion

The main contribution of FrameWorkers is not a new generation backbone, but a modular coordination mechanism that transforms a fixed video-production pipeline into a dynamically constructed workflow. Our experiments support this design: the fine-tuned Director improves dependency-aware routing and incorporates newly registered sub-agents without retraining, and task-stack replanning recovers from most injected failures even when run by a general-purpose LLM. Together, these results suggest that descriptor-based sub-agent integration and dynamic task-stack editing form an effective basis for flexible video production.

At the same time, the current evaluation uses a controlled system configuration and a predefined sub-agent inventory. Although the routing, recovery, execution, and extensibility experiments are conducted at a much larger scale, the real-generation evaluation remains limited in scale by the cost

of generating complete videos. Therefore, our results support the effectiveness of our orchestration design, but do not yet establish robustness across the full diversity of open-world production requests, sub-agent combinations, and generation backends. FRAMEWORKERS also inherits the quality, safety, and reliability limitations of its generative models.