RETOOL-VIDEO: RECURSIVE TOOL-USING VIDEO AGENTS WITH META-AUGMENTED TOOL GROUNDING

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-augmented video agents typically require the ability to locate evidence, inspect multiple modalities, and integrate observations across long temporal horizons. However, existing approaches face two key challenges: at the *tool-space* level, video agents typically expose only a small set of coarse retrieval and inspection tools, limiting fine-grained evidence processing; at the *action-space* level, hierarchical video agents often leave underspecified how abstract video-reasoning intents are grounded into executable tool calls. To address these limitations, we introduce the MetaAug-Video Tool Library (MVTL), comprising 26 base tools for multimodal evidence acquisition and 108 meta tools for fine-grained operations over intermediate results. Building on MVTL, we propose RETOOL-VIDEO, a recursive tool-using framework in which a planner issues either executable tool calls or abstract intents, a resolver repairs, substitutes, or recursively decomposes abstract intents into validated tool calls, and the planner is optimized with reinforcement learning. Experimental results show that RETOOL-VIDEO achieves 72.9, 81.5, and 76.6 accuracy on MVBench, MLVU, and Video-MME$_{\mathrm{w/o\ sub.}}$, respectively. Under the same Qwen3.5-9B backbone and tool library without planner reinforcement learning, Planner–Resolver grounding improves MLVU accuracy by 13.1 percentage points over planner-only tool selection, demonstrating the benefit of separating high-level intent expression from low-level tool execution. The code and tools are available at the anonymous repository.

1 INTRODUCTION

Video understanding is a fundamental capability for multimodal models, requiring reasoning over temporal dynamics, cross-modal signals, and structured information such as audio, subtitles, object states, and event transitions (Tang et al., 2023). Video question answering (VideoQA) covers tasks including temporal localization, compositional reasoning, action counting, state tracking, and cross-modal understanding (Krishna et al., 2017; Lei et al., 2018; Xiao et al., 2021; Li et al., 2024b; Fu et al., 2025). Recent multimodal and video-language models have improved substantially through alignment, instruction tuning, long-context modeling, sparse memory, and hierarchical retrieval (OpenAI, 2023; Anthropic, 2025; Maaz et al., 2024; Lin et al., 2024; Zhang et al., 2024; Ma et al., 2025). Yet complex VideoQA often requires active evidence acquisition, including temporal localization, cross-modal verification, evidence aggregation, and tool use. This motivates video agents that integrate planning, execution, memory, retrieval, and verification (Fan et al., 2024; Zhang et al., 2025c;a; Li et al., 2026a; Lin et al., 2026; Rege et al., 2026). Scaling such agents depends on both the tool space providing multimodal capabilities and the action space mapping high-level video intents to executable operations.

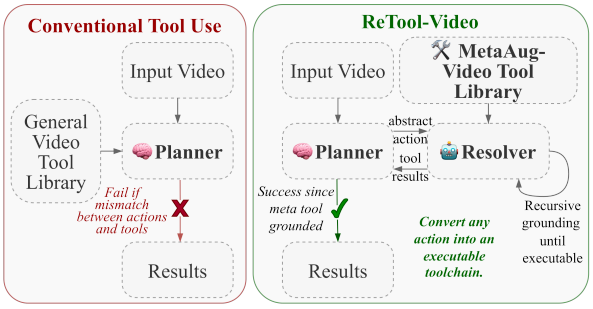

Figure 1: Motivation of ReTool-Video. Conventional video tool-using agents force *abstract actions* (e.g., temporal merging or cross-modal verification) into primitive calls, and ReTool-Video recursively grounds them into executable multimodal tool chains.

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.

Table 1: Comparison of representative video-agent tool libraries by granularity, functionality, and organization. ✓/×denote supported/unsupported features, and N/R denotes unreported tool counts.

Work	# Tools	Tool Type	Audio	Function-aware Routing	Fine-grained Meta Tools	Registry Support
ProViQ (Choudhury et al., 2024)	N/R	Program APIs	×	×	×	×
DoraemonGPT (Yang et al., 2024)	N/R	Subtask / Knowledge	✓	×	×	×
VideoAgent (Fan et al., 2024)	4	Caption / Locate / VQA / Object	×	×	×	×
DVD (Zhang et al., 2025c)	3	Global Browse / Clip Search / Frame Inspect	✓	×	×	×
VITAL (Zhang et al., 2025a)	3	Clip Caption / Clip QA / Video Clip	×	×	×	×
Ego-R1 (Tian et al., 2025)	3	H-RAG / Video-LLM / VLM	✓	×	×	×
STAR (Fan et al., 2025)	22	Spatial / Temporal / General	×	✓	×	×
VideoThinker (Li et al., 2026a)	6	Temporal Retrieval / Temporal Zoom	✓	✓	×	×
VideoSeek (Lin et al., 2026)	3	Overview / Skim / Focus	×	✓	×	×
EGAgent (Rege et al., 2026)	4	Visual / Audio Transcript / Entity Graph / Analyzer	✓	✓	×	×
ReTool-Video (Ours)	134	Base / Fine-grained meta tools	✓	✓	✓	✓

Despite these advances, existing video-based agents still face two coupled challenges in scaling tool use for complex video understanding. First, **from a tool-space perspective, existing video agents lack a broad and composable tool space.** As shown in Table 1, existing systems typically expose a small set of retrieval, localization, VQA, or clip-inspection interfaces (Choudhury et al., 2024; Zhang et al., 2025c; Lin et al., 2026; Rege et al., 2026). Complex video reasoning, however, requires not only acquiring multimodal evidence from frames, clips, audio, subtitles, or scene graphs, but also explicitly transforming intermediate evidence through filtering, temporal merging, counting, reranking, aggregation, and computation. Treating these operations as implicit model reasoning limits controllable evidence composition.

Second, **at the action level, existing video agents force abstract video intents into primitive tool calls.** As shown in Figure 1, most tool-use methods require each action to map directly to a registered tool with valid parameters (Yao et al., 2022; Schick et al., 2023); video agents inherit this flat interface for retrieval, localization, frame inspection, and clip QA (Fan et al., 2024; Zhang et al., 2025a; Lin et al., 2026). Yet intents such as checking whether adjacent clips form one event, comparing object states over time, or aggregating repeated actions are often inherently compositional and cannot be faithfully realized by a single tool call. Forcing them into a single call can yield unsuitable tools, brittle parameters, or premature termination.

We address these two challenges with a co-designed *tool space* and *Planner–Resolver action interface*. MVTL provides a structured base–meta tool space: base tools acquire multimodal evidence, while meta tools explicitly process intermediate results through filtering, temporal merging, counting, reranking, aggregation, computation, and related operations. On top of this tool space, ReTool-Video allows the Planner to issue either an primitive or an abstract action that need not commit to an exact tool name or complete argument schema. A Resolver agent is responsible for local realization of such requests: it directly matches executable calls, repairs invalid arguments, substitutes compatible tools, or recursively decomposes compositional requests into dependency-ordered tool calls. This separation lets the Planner focus on global evidence needs and task progress while the Resolver handles local tool grounding. We keep the Resolver fixed and optimize only the Planner with reinforcement learning for high-level evidence acquisition and termination decisions. Across three benchmarks, ReTool-Video achieves 72.9 on MVBench, 81.5 on MLVU, and 76.6 on Video-MME$_{\mathrm{w/o\ sub.}}$. On MLVU, under the same Qwen3.5-9B backbone and tool environment without planner RL, the proposed Planner–Resolver grounding interface improves accuracy from 61.5 to 74.6, while planner RL further raises performance to 81.5, suggesting complementary gains from recursive tool grounding and planner-level optimization. The contributions of this work are as follows:

- We construct MVTL, a video-agent tool library with 134 registered tools, covering base tools for multimodal evidence acquisition and meta tools for fine-grained intermediate-result processing.
- We introduce ReTool-Video, a bounded intent-to-tool grounding agent framework in which a root planner specifies high-level evidence goals, while a resolver composes and validates executable base- and meta-tool calls and returns structured observations for subsequent planning.
- We evaluate RETOOL-VIDEO on three video benchmarks, where it achieves strong reported performance. Controlled comparisons further isolate the effects of the Resolver, grounding strategy, meta tools, parallel execution, and planner RL.

2 RELATED WORK

2.1 VIDEO UNDERSTANDING

Video understanding spans temporal localization, compositional question answering, state tracking, action counting, and cross-modal evidence alignment (Krishna et al., 2017). Benchmark datasets

such as TVQA (Lei et al., 2018) and NExT-QA (Xiao et al., 2021) show that many questions require temporal localization and evidence grounding. Recent models improve VideoQA through alignment and unified image-video representations (Lin et al., 2024; Maaz et al., 2024); long-video methods expand temporal coverage with sparse memory, long-context transfer, or hierarchical indexing (Song et al., 2024; Zhang et al., 2024; Ma et al., 2025; Yin et al., 2025). Evidence-oriented systems retrieve candidate windows for multimodal verification (Zhang et al., 2025c;a), but generally provide limited support for explicitly transforming intermediate evidence through filtering, merging, counting, reranking, and aggregation. MVTL exposes these acquisition and transformation capabilities as registered base and meta tools, providing the executable tool space used by ReTool-Video.

2.2 Agentic Reasoning and Tool-Augmented Multimodal Systems

Agentic reasoning extends large models from direct response generation to multi-step interaction involving planning, tool use, feedback, and decision updates (Yao et al., 2022). General tool-use agents show that external APIs, expert models, and task-specific tools can expand model capabilities beyond parametric knowledge (Schick et al., 2023; Shen et al., 2023), while multimodal agents connect language models with visual experts, OCR, detection, executable code, and visual programs for compositional reasoning (Yang et al., 2023; Surís et al., 2023). For video tasks, recent agents introduce planners, retrievers, memory, temporal grounding, frame or clip inspection, and verification for long-horizon evidence search (Fan et al., 2024; Liu et al., 2025b; Zhang et al., 2025c;a; Lin et al., 2026). General agent research has also explored hierarchical delegation, unified plan–action representations, and parallel tool scheduling (Li et al., 2024a; Liu et al., 2025a; Yu et al., 2025; Wu et al., 2025). ReTool-Video defines a Planner–Resolver action interface in which the Planner may express an abstract action without committing to an exact API, and a Resolver agent recursively converts that action into validated tool calls while preserving observation dependencies.

3 MVTL: MetaAug-Video Tool Library

Complex video reasoning requires more than coarse interfaces for retrieval, localization, or clip-level QA. Beyond acquiring multimodal evidence, an agent also needs to organize intermediate results through operations such as filtering, temporal merging, counting, reranking, computation, and formatting. As shown in Table 1, existing video-agent tool libraries usually provide limited functional coverage and lack a structured tool space for compositional execution. To address this tool-level challenge, we construct MVTL, a MetaAug-Video Tool Library that combines base tools for multimodal information access with meta tools for intermediate-result processing. MVTL is an independent, retrievable, and extensible library that serves as the capability foundation for recursive video-agent reasoning.

Base and meta tools. Figure 2 summarizes the functional organization of MVTL. The current instantiation contains 134 registered tools: 26 base tools and 108 meta tools. Base tools access videos, images, audio, transcripts, scene graphs, and structured resources, supporting temporal retrieval, frame or clip inspection, audio/ASR access, graph exploration, visual analysis, and cross-modal verification. Meta tools operate on intermediate results and provide fine-grained operations such as filtering, temporal merging, counting, aggregation, reranking, text processing, projection, format conversion, and lightweight computation. The role of the meta layer is not that each operation is individually complex, but that evidence processing becomes explicitly callable and can therefore participate in a multi-step tool chain rather than remaining implicit inside model reasoning.

Registry and tool routing. Each tool is registered with a natural-language description, functional tags, an input/output contract, availability conditions, and runtime constraints. When an action is not already a valid primitive call, the runtime retrieves a small set of relevant tool schemas for the Resolver. Candidate calls are validated against the registry before execution, and invalid calls trigger bounded repair, rerouting, or explicit failure. This organization allows MVTL to grow without requiring the root Planner to memorize the entire low-level tool inventory or instantiate every tool schema itself.

Dual-level video access. MVTL exposes both structured and raw video evidence. Structured tools access subtitles, captions, event descriptions, knowledge graphs, and retrieval indices for efficient localization and candidate generation. Raw-evidence tools return timestamped frames or video clips

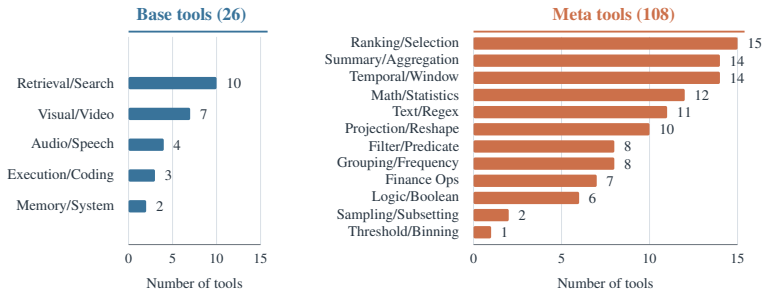

Figure 2: Functional organization of MVTL. Base tools acquire multimodal evidence, while meta tools provide fine-grained processing of intermediate results for compositional tool use.

for fine-grained visual verification. The two levels allow a trajectory to first narrow the temporal search space with structured signals and then verify uncertain evidence directly from the original modality.

4 ReTool-Video

4.1 Action Interface

Complex VideoQA can be viewed as an interactive decision process in which an agent repeatedly identifies missing evidence, invokes tools to acquire or process it, and decides whether the accumulated observations are sufficient for answering. We define a task instance as

$$x = (\mathcal{V}, q, c_0), \tag{1}$$

where $\mathcal{V}$ denotes one or more videos, q is the question, and c_0 is the initial context. The interaction trajectory is

$$\tau = ((s_t, a_t, o_t))_{t=1}^{T}, \tag{2}$$

where s_t contains the task objective, accumulated evidence, execution history, and budget state; a_t is the Planner action; and o_t is the normalized observation returned by the tool environment.

A conventional flat tool-calling policy requires the Planner to directly emit a registered tool together with its executable arguments,

$$a_t^{\text{flat}} = (m, \xi), \qquad m \in \mathcal{M}, \tag{3}$$

thereby coupling the semantic decision of *what evidence is needed* with the execution decision of *which tool and schema realize it*. We define the state-dependent primitive-action set as

$$\mathcal{A}_{\text{prim}}(s) = \{(m, \xi) : m \in \mathcal{M},\ \xi \models \text{Schema}(m),\ \text{Constraint}(m, \xi, s) = 1\}, \tag{4}$$

which contains exactly the registered tool calls that are executable under the current state. ReTool-Video then defines two valid action levels:

$$a_t \in \mathcal{A}_{\text{prim}}(s_t) \cup \mathcal{A}_{\text{abs}}, \tag{5}$$

where $\mathcal{A}_{\text{prim}}(s_t)$ contains state-valid primitive tool calls and $\mathcal{A}_{\text{abs}}$ contains abstract actions. An abstract action specifies a local evidence goal—for example, merging temporally adjacent events or cross-checking an action with audio—without requiring the Planner to know the exact executable tool or complete argument schema in advance.

We represent an abstract action as

$$a_t^{\text{abs}} = (\tilde{m}_t, d_t, \tilde{\xi}_t), \tag{6}$$

where $\tilde{m}_t$ is a possibly unregistered or capability-level tool name, d_t is a natural-language description of the intended evidence operation, and $\tilde{\xi}_t$ denotes possibly incomplete or underspecified arguments. Unlike a primitive action, an abstract action is not required to satisfy $\tilde{m}_t \in \mathcal{M}$ or a registered input schema at generation time; these constraints are resolved by the Resolver during grounding.

The two action levels are handled differently:

$$\mathcal{T}_t = \begin{cases} \{a_t\}, & a_t \in \mathcal{A}_{\text{prim}}(s_t), \\ \text{Resolver}(a_t, s_t; \mathcal{R}_k), & a_t \in \mathcal{A}_{\text{abs}}, \end{cases} \qquad o_t = \text{Exec}(\mathcal{T}_t), \tag{7}$$

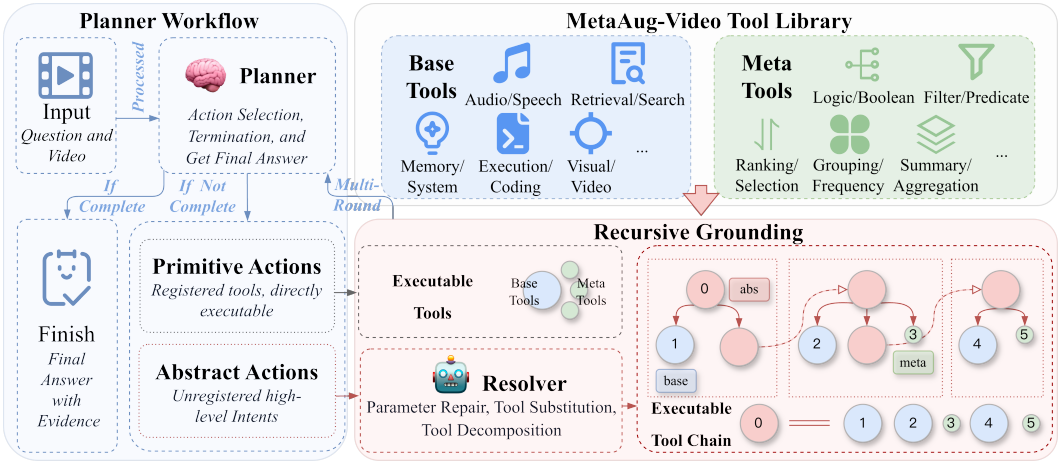

Figure 3: Overall Planner–Resolver workflow of ReTool-Video. The Planner selects primitive tool calls or abstract actions. Primitive calls execute directly, while the Resolver recursively grounds abstract actions into validated calls over MVTL and returns their observations to the Planner.

where $\mathcal{R}_k$ denotes up to k routed candidate tool schemas and $\mathcal{T}_t$ is the local execution tree produced for the current action. Primitive calls execute directly, and abstract actions are delegated to the Resolver agent, which is restricted to local tool grounding and cannot make the global termination decision. The Planner retains responsibility for evidence planning, reasoning progress, and the final answer. Figure 3 illustrates this Planner–Resolver interaction and the two action paths.

4.2 Recursive Tool Grounding with the Resolver

The Resolver is a dedicated local agent for converting one abstract Planner action into executable tool calls; it does not take over global task planning. Given the current abstract request and routed candidate schemas, the Resolver follows four resolution modes of increasing scope, with $\mathcal{A}_{\text{prim}}(s)$ from Equation 4 defining the executable primitive calls available at state s. **L1: Direct execution.** If the requested tool exists, its arguments satisfy the registered schema, and the state-dependent execution constraints hold, the action is a legal primitive call $u = (m, \xi) \in \mathcal{A}_{\text{prim}}(s)$ and is executed directly. **L2: Argument repair.** If the intended tool is appropriate but its arguments are missing, aliased, or malformed, the Resolver keeps the target tool fixed and rewrites only the arguments to satisfy the registered schema. This handles local parameter-binding errors without changing the Planner's capability decision. **L3: Tool substitution.** If the requested tool is unavailable, unregistered, or semantically incompatible with the current state, the Resolver selects a compatible executable tool from the routed candidate set and instantiates its schema. This maps a capability-level request to the executable vocabulary currently available in MVTL. **L4: Recursive decomposition.** If a single tool is insufficient, the Resolver decomposes the action into multiple child actions $\{a_i\}_{i=1}^{n}$. Each child is routed and validated independently: executable children become primitive calls, whereas unresolved children remain abstract actions and are recursively passed through the same L1–L4 procedure. Recursion is therefore triggered by unresolved tool dependencies, rather than by repeatedly replanning the entire task.

The resolution process can be summarized as

$$a^{\text{abs}} \implies \{a_i\}_{i=1}^{n} \implies \{\ell_j\}_{j=1}^{L}, \qquad \ell_j \in \mathcal{A}_{\text{prim}}(s_j) \cup \{\perp_{\text{fail}}, \perp_{\text{budget}}\}, \tag{8}$$

where the decomposition step is optional, $\perp_{\text{fail}}$ denotes an explicit grounding or execution failure, and $\perp_{\text{budget}}$ denotes termination caused by budget exhaustion. Thus, every terminal leaf is either a state-valid primitive tool call or an explicit non-executable terminal outcome. The resulting recursive grounding process is illustrated in Figure 4.

Observation-conditioned composition. For an abstract action, the Resolver produces a dependency-ordered tree $\mathcal{T}_t$. Internal nodes denote unresolved child actions, while leaves denote legal primitive calls or explicit failure/budget terminal outcomes. Dependencies are defined through observations: after executing a prerequisite call u_i, its observation o_i is folded into the shared evidence state before a dependent child is instantiated or resolved. Independent leaves may be executed in parallel, whereas dependent leaves are scheduled after their prerequisites. Figure 4 illustrates this process for counting

"cleaning toilet" scenes, where retrieval and candidate localization are followed by filtering, visual verification, and temporal aggregation.

Validation and bounded execution. Before execution, every primitive call is checked for registry membership, schema-compatible arguments, availability, and runtime constraints. Calls that remain invalid after bounded repair are returned as explicit failure observations rather than executed. Therefore, the grounding procedure guarantees that every executed leaf is a valid primitive action under the current state, although execution validity does not imply that the selected tool is semantically optimal for the task.

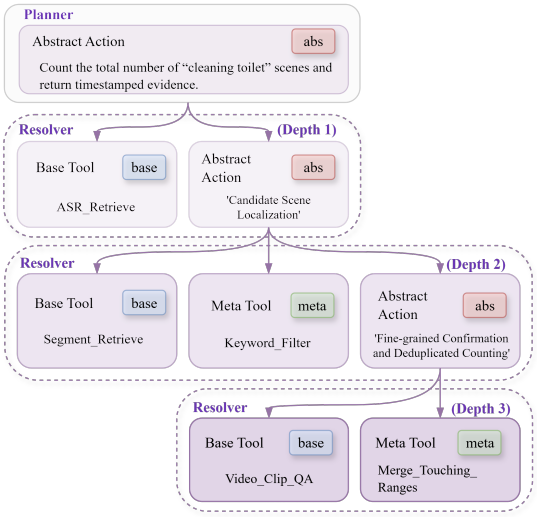

Figure 4: An example of tool grounding in ReTool-Video, where abstract actions are recursively resolved into executable base- and meta-tool calls.

The recursive grounding process is explicitly bounded. Each Resolver invocation has a maximum recursion depth D, bounded branching and retry budgets, and a finite execution budget. Together with a finite number of root-Planner rounds, these constraints ensure that grounding terminates with either successful execution, an explicit failure observation, or budget exhaustion. In practice, these bounds prevent unresolved abstract actions from inducing unbounded recursive expansion while preserving the ability to realize multi-stage tool dependencies.

4.3 Reinforcement Learning on Planner

The Planner and Resolver have deliberately different roles during learning. We optimize only the Planner π_θ, while keeping the Resolver, tool registry, and execution modules fixed. The Planner therefore receives trajectory-level learning signals for global decisions—which evidence to seek, whether to issue a primitive or abstract action, whether the accumulated evidence is sufficient, and when to terminate—without simultaneously changing the local grounding behavior. Keeping the Resolver fixed also preserves a stable action-to-execution interface during policy optimization and avoids assigning sparse final-answer reward directly to local argument repair or tool substitution decisions.

Given a task instance $x = (\mathcal{V}, q, c_0)$, the old Planner policy π_{old} samples a group of G complete episodes $\{\tau_i\}_{i=1}^{G}$. We use final-answer correctness as the dominant reward with lightweight structural and trajectory-efficiency terms:

$$R(\tau) = R_{\text{ans}}(\tau) + \lambda_{\text{valid}} C_{\text{valid}}(\tau) - \lambda_{\text{cost}} C_{\text{cost}}(\tau). \tag{9}$$

For multiple-choice questions, R_{ans} is binary; for open-ended questions, it follows the dataset-specific evaluator and is normalized to $[0, 1]$:

$$R_{\text{ans}}(\tau) = \begin{cases} \mathbf{1}[\hat{y} = y], & \text{for multiple-choice QA,} \\ \text{Eval}_{\mathcal{D}}(\hat{y}, y), & \text{for open-ended QA.} \end{cases} \tag{10}$$

Here C_{valid} rewards compliance with the required action/finish protocol, while C_{cost} penalizes clearly ineffective trajectories such as empty progress, repeated probing, or max-step termination.

For each sampled episode τ_i, let $R_i = R(\tau_i)$. The episode-level group-relative advantage is

$$\hat{A}_i = \frac{R_i - \mu(\{R_j\}_{j=1}^{G})}{\sigma(\{R_j\}_{j=1}^{G}) + \epsilon}. \tag{11}$$

Let $\mathcal{T}_i^p$ denote Planner-token positions and $\rho_{i,t}(\theta)$ the probability ratio between π_θ and π_{old}. We optimize the Planner with the GRPO (Shao et al., 2024) clipped objective using these advantages, together with a reverse-KL term to a frozen reference VLM; the full token-level objective is given in Appendix B.2.

Table 2: Reported accuracy (%) of VLMs and video reasoning or tool-use systems. Baselines retain their original input, backbone, tools, and evaluation protocol.

Method	Backbone size	MLVU	MVBench	Video-MME$_{\text{w/o sub.}}$
Closed-source models				
GPT4-V (OpenAI, 2023)	-	-	43.5	60.7
GPT-5-Nano-2025-08-07 (Singh et al., 2025)	-	69.2	-	66.2
GPT-4o (Hurst et al., 2024)	-	64.6	-	71.9
Claude-Sonnet-4.5 (Anthropic, 2025)	-	72.8	-	75.3
Gemini-2.5-Flash-Lite (Comanici et al., 2025)	-	78.5	-	72.7
Open-source video-language models				
LLaMA-VID (Li et al., 2024c)	7B	33.2	41.9	-
ShareGPT4Video (Chen et al., 2024)	8B	33.8	51.2	43.6
VideoLLaMA2 (Cheng et al., 2024)	8×7B	45.6	54.6	46.6
Kangaroo (Liu et al., 2026)	8B	61.0	61.0	56.0
Flash-VStream (Zhang et al., 2025b)	7B	66.3	65.4	61.2
NVILA (Liu et al., 2025c)	8B	70.1	68.1	64.2
InternVL3.5-30B-A3B (Wang et al., 2025)	30B	73.0	72.1	68.7
Video reasoning and tool-use systems				
Weaver (Shi et al., 2026)	7B	54.5	67.7	65.3
VideoThinker (Li et al., 2026a)	7B	54.8	-	53.7
Framemind (Ge et al., 2025)	7B	48.6	64.2	60.9
VideoExplorer (Yuan et al., 2025)	7B/32B	58.6	-	-
VideoMind (Liu et al., 2025b)	7B	64.4	-	58.2
DVD (Zhang et al., 2025c)	-	-	-	67.3
STAR (Fan et al., 2025)	-	-	-	70.0
VideoSeek (Lin et al., 2026)	-	-	-	70.1
META-offline (Huang et al., 2026)	7B	77.3	-	74.0
ReTool-Video (Ours)	9B	**81.5**	**72.9**	**76.6**

5 EXPERIMENTS

5.1 EXPERIMENT SETTINGS

Benchmarks and Evaluation Metrics. We evaluate ReTool-Video on three representative general-purpose video understanding benchmark datasets: MVBench (Li et al., 2024b), MLVU (Zhou et al., 2025), and Video-MME$_{\text{w/o sub.}}$ (Fu et al., 2025). MVBench focuses on short-video temporal understanding, while MLVU emphasizes long-video reasoning with tasks covering both global comprehension and local evidence reasoning. Video-MME evaluates diverse video durations; we use its no-provided-subtitle setting. Our system may generate ASR from video audio, which is distinct from receiving benchmark subtitles as input. Following the benchmarks' official evaluation protocols, we report *accuracy*. Further dataset and evaluation details are in Appendix C.1.

Baselines and Implementation Details. Closed-source baselines are GPT4-V (OpenAI, 2023), GPT-5-Nano (Singh et al., 2025), GPT-4o (Hurst et al., 2024), Claude-Sonnet-4.5 (Anthropic, 2025), and Gemini-2.5-Flash-Lite (Comanici et al., 2025). Open-source video-language baselines are LLaMA-VID (Li et al., 2024c), ShareGPT4Video (Chen et al., 2024), VideoLLaMA2 (Cheng et al., 2024), Kangaroo (Liu et al., 2026), Flash-VStream (Zhang et al., 2025b), NVILA (Liu et al., 2025c), and InternVL3.5 (Wang et al., 2025). Video reasoning and tool-use baselines are Weaver (Shi et al., 2026), VideoThinker (Li et al., 2026a), Framemind (Ge et al., 2025), VideoExplorer (Yuan et al., 2025), VideoMind (Liu et al., 2025b), DVD (Zhang et al., 2025c), STAR (Fan et al., 2025), VideoSeek (Lin et al., 2026) and META (Huang et al., 2026). Detailed input, frame-sampling, tool-budget, subtitle, and preprocessing settings are in Appendix C.2. Our planner and resolver both use Qwen3.5-9B (Qwen Team, 2026). The planner uses temperature 1.0 and top-$p = 0.95$, while the resolver uses temperature 0.7 and top-$p = 0.8$; both use top-$k = 20$. Planner RL samples $G = 4$ trajectories per instance and uses $4\times$ NVIDIA A800 GPUs. Full decoding and training configurations are provided in Appendix C.11.

5.2 MAIN RESULTS

Table 2 compares ReTool-Video with closed- and open-source video-language models, as well as recent video reasoning and tool-augmented systems. ReTool-Video achieves strong performance, with 81.5 on MLVU, 72.9 on MVBench, and 76.6 on Video-MME$_{\text{w/o sub.}}$. Compared with InternVL3.5-30B-A3B, for example, it improves MLVU and Video-MME$_{\text{w/o sub.}}$ by 8.5 and 7.9 percentage points, respectively. The smaller gain on MVBench may reflect the relatively lower demand for long-range evidence search and iterative tool interaction in short-video understanding.

To further examine whether the improvement is specific to the default backbone, we replace both the planner and resolver with InternVL3.5-30B-A3B while keeping the ReTool-Video framework unchanged. Under this setting, ReTool-Video obtains 73.4/79.1/74.7 on MVBench/MLVU/Video-MME$_{\text{w/o sub.}}$, compared with 72.1/73.0/68.7 for direct use of the same backbone, corresponding to gains of +1.3/+6.1/+6.0 points. These results suggest that the benefit of the proposed tool-use framework is not specific to a single backbone; implementation details are provided in Appendix C.7.

5.3 Ablation Study

Table 3: Component ablations (accuracy, %). All Tool-use variants use the same Qwen3.5-9B backbone and omit planner RL; ReTool-Video includes planner RL.

Setting	**MLVU**	**MVBench**	**Video-MME$_{\text{w/o sub.}}$**
Direct Response (w/o Tools)	47.6	46.6	57.4
Tool-use w/o Resolver (Planner-only)	61.5	64.2	65.5
Tool-use w/o Parallel&Recursion	70.5	63.5	71.8
Tool-use w/o Meta Tools	71.2	64.1	71.7
Tool-use w/o Parallel	68.4	64.4	70.2
Tool-use w/o Recursion	63.2	64.8	66.7
Tool-use with simplified preprocessing	70.8	64.0	71.5
Tool-use	74.6	65.3	73.3
ReTool-Video	81.5	72.9	76.6

Table 3 evaluates the contribution of each component under controlled settings. Direct Response shares the same backbone, preprocessing pipeline, and initial visual context as Tool-use, but cannot acquire additional evidence through tool interaction. All Tool-use variants exclude planner RL, whereas the final ReTool-Video setting includes it. Simplified preprocessing replaces ASR-guided and scene-aware temporal boundaries with fixed 30-second segments decreases accuracy by 3.8, 1.3, and 1.8 points on MLVU, MVBench, and Video-MME$_{\text{w/o sub.}}$, respectively, while removing meta tools causes drops of 3.4, 1.2, and 1.6 points. These results indicate that both adaptive video segmentation and meta-level operations contribute consistently across benchmarks. Removing recursion while retaining parallel execution leads to larger degradation on MLVU and Video-MME$_{\text{w/o sub.}}$ than removing both components together, suggesting that parallel tool execution benefits from reliable grounding of dependent actions. Finally, introducing planner RL improves the Tool-use setting by 6.9, 7.6, and 3.3 points on MLVU, MVBench, and Video-MME$_{\text{w/o sub.}}$, respectively, demonstrating the importance of optimizing high-level action selection, evidence acquisition, and termination decisions.

5.4 Grounding Strategy Analysis

Table 4: Grounding strategies with the same MVTL environment and no RL (accuracy, %).

Grounding strategy	MVBench	MLVU	Video-MME$_{\text{w/o sub.}}$
Exact schema/name matching	63.5	70.5	71.8
Embedding top-1 tool matching	64.2	71.6	72.1
One-step grounding	64.7	72.8	72.6
Recursive grounding	**65.3**	**74.6**	**73.3**

Comparison of Grounding Strategies. Table 4 compares four grounding strategies under the same MVTL environment and without planner RL. Recursive grounding achieves 65.3, 74.6, and 73.3 on MVBench, MLVU, and Video-MME$_{\text{w/o sub.}}$, respectively, consistently outperforming exact schema/name matching, embedding-based tool matching, and one-step grounding. Compared with one-step LLM grounding, recursive grounding yields gains of 0.6, 1.8, and 0.7 points across the three benchmarks. The improvement is largest on MLVU, where long-form video reasoning more often requires multi-step evidence acquisition and ordered tool dependencies that cannot be resolved in a single grounding step. These results suggest that iterative decomposition provides additional benefit beyond stronger one-step semantic matching, particularly when abstract intents must be progressively converted into executable tool sequences. The target-segment diagnostic in Appendix C.3 further shows an 11.2-point gap in favor of recursive grounding, providing complementary evidence for this effect. We note, however, that the available records do not establish strictly equal inference compute across grounding strategies.

Table 5: Resolver action validity and efficiency on 100 sampled MLVU cases.

Metric	Planner-only	ReTool-Video
Executable-action rate	82.7%	95.1%
Schema-valid argument rate	85.4%	94.8%
Repeated/ineffective-call rate	17.9%	8.3%

Effect of the Resolver on Action Quality. We further audit 100 sampled MLVU cases at the action level. As shown in Table 5, ReTool-Video increases the executable-action rate from 82.7% to 95.1% and the schema-valid argument rate from 85.4% to 94.8%, while reducing repeated or ineffective calls from 17.9% to 8.3%. These results provide complementary evidence that the resolver improves the executability and validity of

grounded actions, rather than merely increasing the number of inference steps. Since the audit is based on a finite sample, we treat these statistics as descriptive rather than as uncertainty-calibrated estimates.

5.5 Tool-use Behavior Analysis

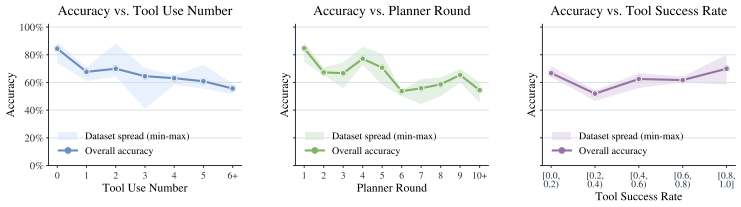

Figure 5: Relationship between accuracy and tool-use behavior, including tool-call count, reasoning iterations, and tool-call success rate.

Relationship between Tool-use Behavior and Performance. Figure 5 analyzes how task accuracy varies with the number of tool calls, planner rounds, and tool-call success rate. Solid lines report the mean accuracy across the three benchmarks, while shaded regions indicate the corresponding min–max range. Accuracy remains relatively stable within one to four tool calls and does not improve monotonically as trajectories become longer. Similarly, additional planner rounds do not consistently translate into higher accuracy. These trends suggest that moderate interaction depth is generally sufficient for evidence localization, verification, and aggregation, whereas longer trajectories may instead reflect unresolved uncertainty or repeated probing. In contrast, accuracy exhibits a clearer positive association with tool-call success rate, indicating that obtaining valid and informative observations is more important than simply increasing the number of tool interactions. These relationships are descriptive and should not be interpreted as causal effects.

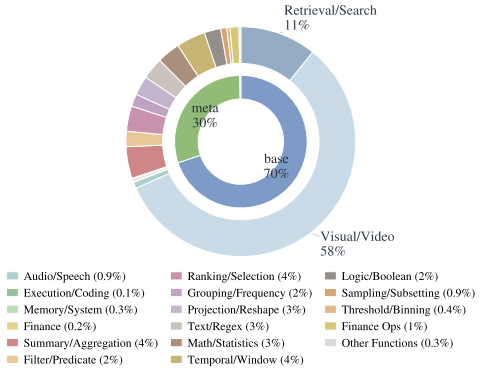

Figure 6: Runtime tool-call distribution by base/meta type and functional category.

Runtime Tool Usage Distribution. Figure 6 summarizes the distribution of tool calls during inference. Base tools account for approximately 70% of all calls, while meta tools contribute the remaining 30%, showing that intermediate-result processing constitutes a substantial fraction of the execution trajectory. Among functional categories, visual/video tools dominate with 58% of calls, followed by retrieval/search at 11%. Summary/aggregation, ranking/selection, and temporal/window operations each account for approximately 4%. The five most frequently invoked tools together cover 94.62% of all calls, indicating that a small core set of tools handles most routine evidence acquisition, while the broader tool library provides specialized capabilities for less frequent but compositional reasoning needs. Detailed per-tool statistics are reported in Appendix C.5, while the aggregate effect of meta tools is evaluated separately in Table 3. Runtime-audit definitions and representative execution traces are provided in Appendices C.4 and C.10.

6 Conclusion

This work presents ReTool-Video, a hierarchical tool-use framework for compositional video reasoning that decouples high-level evidence planning from low-level executable grounding. The Planner operates over both primitive and abstract actions, while a dedicated Resolver progressively grounds unresolved intents through argument repair, tool substitution, and bounded recursive decomposition. To support such grounding, we introduce MVTL, which combines multimodal evidence-acquisition tools with fine-grained meta tools for explicitly transforming and composing intermediate evidence. This design allows complex video-reasoning intents to be realized as validated, dependency-aware tool chains without requiring the Planner to directly manage the full tool space and execution details. We further optimize only the Planner with reinforcement learning while keeping the Resolver fixed, preserving a stable grounding interface during training. Experiments across three video benchmarks consistently demonstrate the benefits of recursive grounding, meta-tool augmentation, and planner-level optimization, suggesting that separating semantic intent specification from executable tool grounding provides a useful abstraction for complex video-agent reasoning.

Ethics Statement

This work uses existing public video benchmarks and released model backbones. We do not collect private user data or conduct crowdsourced annotation or human-subject studies. Tool-augmented video understanding could nevertheless enable privacy-invasive analysis if applied to footage without authorization; any deployment should respect dataset terms, applicable privacy protections, and the consent of people depicted.

Reproducibility Statement

The appendices provide the tool-counting and metadata protocol, recursive grounding procedure, planner training objective, benchmark and baseline protocols, model and decoding configurations, preprocessing pipeline, runtime budgets, RL data mixture, and compute resources. The anonymized code-and-tool release includes scripts for reproducing the main experiments on the standard splits and answer formats of MVBench, MLVU, and Video-MME$_{\text{w/o sub.}}$.

AI assistance disclosure

OpenAI Codex was used for manuscript organization, readability edits, and LaTeX formatting. The authors are responsible for reviewing AI-assisted text and verifying all scientific claims, results, citations, and the final manuscript.

References

Anthropic. Claude Sonnet 4.5 System Card. `https://www.anthropic.com/claude-sonnet-4-5-system-card`, September 2025. Listed on Anthropic "Model System Cards" page.

Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Bin Lin, Zhenyu Tang, et al. Sharegpt4video: Improving video understanding and generation with better captions. *Advances in Neural Information Processing Systems*, 37:19472–19495, 2024.

Zesen Cheng, Sicong Leng, Hang Zhang, Yifei Xin, Xin Li, Guanzheng Chen, Yongxin Zhu, Wenqi Zhang, Ziyang Luo, Deli Zhao, et al. Videollama 2: Advancing spatial-temporal modeling and audio understanding in video-llms. *arXiv preprint arXiv:2406.07476*, 2024.

Rohan Choudhury, Koichiro Niinuma, Kris M Kitani, and László A Jeni. Video question answering with procedural programs. In *European Conference on Computer Vision*, pages 315–332. Springer, 2024.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Sunqi Fan, Jiashuo Cui, Meng-Hao Guo, and Shuojin Yang. Tool-augmented spatiotemporal reasoning for streamlining video question answering task. *arXiv preprint arXiv:2512.10359*, 2025.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. Videoagent: A memory-augmented multimodal agent for video understanding. In *European Conference on Computer Vision*, pages 75–92. Springer, 2024.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24108–24118, 2025.

Haonan Ge, Yiwei Wang, Kai-Wei Chang, Hang Wu, and Yujun Cai. Framemind: Frame-interleaved video reasoning via reinforcement learning. *arXiv preprint arXiv:2509.24008*, 2025.

Jing Huang, Luyuan Chen, Zhijie Xu, Yadong Li, Xingzhong Xu, Siye Chen, Jie Liu, Ming Kong, and Qiang Zhu. Meta: Meta evolution of tool trajectory adaptation for long-video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9837–9846, 2026.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017.

Jie Lei, Licheng Yu, Mohit Bansal, and Tamara Berg. Tvqa: Localized, compositional video question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 1369–1379, 2018.

Ao Li, Yuexiang Xie, Songze Li, Fugee Tsung, Bolin Ding, and Yaliang Li. Agent-oriented planning in multi-agent systems. *arXiv preprint arXiv:2410.02189*, 2024a.

Chenglin Li, Qianglong Chen, Feng Han, Yikun Wang, Xingxi Yin, Yan Gong, Ruilin Li, Yin Zhang, and Jiaqi Wang. Videothinker: Building agentic videollms with llm-guided tool reasoning. *arXiv preprint arXiv:2601.15724*, 2026a.

Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206, 2024b.

Mingxin Li, Yanzhao Zhang, Dingkun Long, Chen Keqin, Sibo Song, Shuai Bai, Zhibo Yang, Pengjun Xie, An Yang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Qwen3-vl-embedding and qwen3-vl-reranker: A unified framework for state-of-the-art multimodal retrieval and ranking. *arXiv preprint arXiv:2601.04720*, 2026b.

Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. In *European Conference on Computer Vision*, pages 323–340. Springer, 2024c.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pages 5971–5984, 2024.

Jingyang Lin, Jialian Wu, Jiang Liu, Ximeng Sun, Ze Wang, Xiaodong Yu, Jiebo Luo, Zicheng Liu, and Emad Barsoum. Videoseek: Long-horizon video agent with tool-guided seeking. *arXiv preprint arXiv:2603.20185*, 2026.

Haowei Liu, Xi Zhang, Haiyang Xu, Yuyang Wanyan, Junyang Wang, Ming Yan, Ji Zhang, Chunfeng Yuan, Changsheng Xu, Weiming Hu, et al. Pc-agent: A hierarchical multi-agent collaboration framework for complex task automation on pc. *arXiv preprint arXiv:2502.14282*, 2025a.

Jiajun Liu, Yibing Wang, Hanghang Ma, Xiaoping Wu, Xiaoqi Ma, Xiaoming Wei, Jianbin Jiao, Enhua Wu, and Jie Hu. Kangaroo: A powerful video-language model supporting long-context video input: J. liu et al. *International Journal of Computer Vision*, 134(3):114, 2026.

Ye Liu, Kevin Qinghong Lin, Chang Wen Chen, and Mike Zheng Shou. Videomind: A chain-of-lora agent for long video reasoning. *arXiv e-prints*, pages arXiv–2503, 2025b.

Zhijian Liu, Ligeng Zhu, Baifeng Shi, Zhuoyang Zhang, Yuming Lou, Shang Yang, Haocheng Xi, Shiyi Cao, Yuxian Gu, Dacheng Li, et al. Nvila: Efficient frontier visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4122–4134, 2025c.

Ziyu Ma, Chenhui Gou, Hengcan Shi, Bin Sun, Shutao Li, Hamid Rezatofighi, and Jianfei Cai. Drvideo: Document retrieval based long video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 18936–18946, 2025.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12585–12602, 2024.

Munan Ning, Bin Zhu, Yujia Xie, Bin Lin, Jiaxi Cui, Lu Yuan, Dongdong Chen, and Li Yuan. Video-bench: A comprehensive benchmark and toolkit for evaluating video-based large language models. *Computational Visual Media*, 2025.

OpenAI. GPT-4V(ision) System Card. `https://openai.com/index/gpt-4v-system-card/`, September 2023. Accessed: 2025-09-19.

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

Aniket Rege, Arka Sadhu, Yuliang Li, Kejie Li, Ramya Korlakai Vinayak, Yuning Chai, Yong Jae Lee, and Hyo Jin Kim. Agentic very long video understanding. *arXiv preprint arXiv:2601.18157*, 2026.

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

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

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. *Advances in Neural Information Processing Systems*, 36: 38154–38180, 2023.

Yudi Shi, Shangzhe Di, Qirui Chen, Qinian Wang, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. Weaver: End-to-end agentic system training for video interleaved reasoning. *arXiv preprint arXiv:2602.05829*, 2026.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, et al. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*, 2025.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. Moviechat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18221–18232, 2024.

Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11888–11898, 2023.

Yunlong Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, Ali Vosoughi, Chao Huang, Zeliang Zhang, Pinxin Liu, Mingqian Feng, Feng Zheng, Jianguo Zhang, Ping Luo, Jiebo Luo, and Chenliang Xu. Video understanding with large language models: A survey. *arXiv preprint arXiv:2312.17432*, 2023.

Shulin Tian, Ruiqi Wang, Hongming Guo, Penghao Wu, Yuhao Dong, Xiuying Wang, Jingkang Yang, Hao Zhang, Hongyuan Zhu, and Ziwei Liu. Ego-r1: Chain-of-tool-thought for ultra-long egocentric video reasoning. *arXiv preprint arXiv:2506.13654*, 2025.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025.

Kaiwen Wei, Xiao Liu, Jie Zhang, Zijian Wang, Ruida Liu, Yuming Yang, Xin Xiao, Xiao Sun, Haoyang Zeng, Changzai Pan, et al. Cfvbench: A comprehensive video benchmark for fine-grained multimodal retrieval-augmented generation. In *Proceedings of the ACM Web Conference 2026*, pages 2501–2512, 2026.

Jiaqi Wu, Qinlao Zhao, Zefeng Chen, Kai Qin, Yifei Zhao, Xueqian Wang, and Yuhang Yao. Gap: Graph-based agent planning with parallel tool use and reinforcement learning. *arXiv preprint arXiv:2510.25320*, 2025.

Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786, 2021.

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.

Zongxin Yang, Guikun Chen, Xiaodi Li, Wenguan Wang, and Yi Yang. Doraemongpt: Toward understanding dynamic scenes with large language models (exemplified as a video agent). *arXiv preprint arXiv:2401.08392*, 2024.

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

Yufei Yin, Qianke Meng, Minghao Chen, Jiajun Ding, Zhenwei Shao, and Zhou Yu. Videoarm: Agentic reasoning over hierarchical memory for long-form video understanding. *arXiv preprint arXiv:2512.12360*, 2025.

Zhaoyang Yu, Jiayi Zhang, Huixue Su, Yufan Zhao, Yifan Wu, Mingyi Deng, Jinyu Xiang, Yizhang Lin, Lingxiao Tang, Yuyu Luo, et al. Recode: Unify plan and action for universal granularity control. *arXiv preprint arXiv:2510.23564*, 2025.

Huaying Yuan, Zheng Liu, Junjie Zhou, Hongjin Qian, Yan Shu, Nicu Sebe, Ji-Rong Wen, and Zhicheng Dou. Videoexplorer: Think with videos for agentic long-video understanding. *arXiv preprint arXiv:2506.10821*, 2025.

Haoji Zhang, Xin Gu, Jiawen Li, Chixiang Ma, Sule Bai, Chubin Zhang, Bowen Zhang, Zhichao Zhou, Dongliang He, and Yansong Tang. Thinking with videos: Multimodal tool-augmented reinforcement learning for long video reasoning. *arXiv preprint arXiv:2508.04416*, 2025a.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-vstream: Efficient real-time understanding for long video streams. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 21059–21069, 2025b.

Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024.

Xiaoyi Zhang, Zhaoyang Jia, Zongyu Guo, Jiahao Li, Bin Li, Houqiang Li, and Yan Lu. Deep video discovery: Agentic search with tool use for long-form video understanding. *arXiv preprint arXiv:2505.18079*, 2025c.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025d.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, et al. Mlvu: Benchmarking multi-task long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13691–13701, 2025.

A Additional Details of MVTL

A.1 Tool Counting and Exposure Protocol

The tool number reported in this paper follows the registry-level counting protocol. Each independently callable registry entry is counted as one tool, even if multiple entries share the same backend model, retrieval index, or service endpoint. Under this protocol, MVTL contains 134 registered tools, including 26 base tools and 108 meta tools.

Tools used for evidence acquisition and evidence processing can be selected by the planner or resolver, while several runtime-control tools are reserved for the execution engine. For example, tools for tool retrieval, memory folding, context compression, exception recovery, or scheduling assistance are invoked internally when needed. This separation prevents the planner from overfitting to low-level runtime management operations while still allowing the system to use them for robust execution.

A.2 Tool Metadata Fields

Each registered tool is described by a compact metadata entry. The main fields are as follows:

- **Name**: the unique registry identifier used for tool matching and execution.
- **Description**: a natural-language capability description used for semantic retrieval and disambiguation.
- **Tags**: functional labels indicating modality, granularity, and operation type.
- **Input schema**: required and optional arguments, argument types, default values, and basic validity constraints.
- **Output schema**: the expected return structure after execution.
- **Availability conditions**: constraints on modality, video state, index availability, or external service accessibility.
- **Runtime constraints**: budget limits, timeout settings, retry limits, and fallback policies.

These fields make heterogeneous tools searchable, executable, and comparable under the same registry interface. They also allow the runtime to validate tool calls before execution and to normalize outputs after execution.

A.3 Observation Normalization

Tool outputs are converted into a shared observation format before being returned to the planner. An observation may contain textual evidence, timestamps, ranked candidates, structured tables, image or clip references, confidence scores, execution status, and error messages. The exact fields depend on the tool type, but the runtime always preserves three kinds of information: the executed tool call, the returned evidence, and the execution signal.

This normalization is important for multi-round reasoning. Since different tools may return heterogeneous results, the planner should not need to parse backend-specific formats. Instead, all tool feedback is folded into a consistent observation buffer, which records what evidence has been acquired, which actions have already been attempted, and whether the current reasoning state is sufficient for termination.

A.4 Caching and Execution Records

For efficiency and reproducibility, deterministic tool calls can be cached according to their tool name, normalized arguments, and video identifier. Cached results are reused when the same evidence request appears in later rounds or parallel branches. This is especially useful for retrieval, filtering, graph traversal, and statistical tools, whose outputs are stable under the same inputs.

The runtime also records execution traces for analysis and training. Each trace stores the planner action, resolved tool call, execution result, normalized observation, failure status, and budget consumption. These records are used to analyze tool-use behavior, compute trajectory-level statistics, and diagnose ineffective or repeated tool calls.

A.5 Model Backbones for Model-Based Tools

The current model-based tools mainly rely on several Qwen-series backbones. `Qwen/Qwen3.5-9B` (Qwen Team, 2026) provides multimodal generative capabilities for clip understanding, visual question answering, and controlled caption generation. `Qwen/Qwen3-VL-Embedding-2B` (Li et al., 2026b) is used for video-segment grounding and cross-modal semantic matching, while `Qwen/Qwen3-Embedding-0.6B` (Zhang et al., 2025d) is used for tool-schema retrieval in `Tool_Search`. `Qwen/Qwen3.5-0.8B` (Qwen Team, 2026) supports lightweight local repair and rewriting, such as argument completion, format correction, and structured response normalization.

A.6 Limitation and Failure States

The runtime distinguishes several common failure states:

- **Schema error**: the action cannot be parsed into the required input schema.
- **Missing argument**: one or more required arguments are absent or underspecified.
- **Unavailable tool**: the matched tool cannot be executed under the current modality, index, or service condition.
- **Empty result**: the tool executes successfully but returns no usable evidence.
- **Invalid output**: the returned result does not satisfy the expected output schema.
- **Budget violation**: the tool call exceeds the allowed step, time, or resource budget.

These failure states are not treated uniformly. Some errors can be repaired locally, such as missing arguments or invalid formats; some require retry or fallback, such as empty retrieval results; and some are returned directly as execution signals, such as budget violations or unavailable resources. This explicit failure typing helps the planner and resolver distinguish between insufficient evidence, invalid tool use, and runtime constraints.

The evaluation does not isolate the causal effect of training-data composition or backbone temporal modeling. Comparisons between language-only and vision-language systems cannot determine how much of the gap comes from video-centric training data. Controlled training with matched backbones and varied text–video data mixtures remains future work.

B Additional Methodological Details

B.1 Additional Details of ReTool-Video

For a primitive action, the runtime executes the matched registry tool after schema validation. For an abstract action, the resolver receives the action description and its current arguments, and focuses only on the local grounding objective without taking over global planning. Its operations include: repairing invalid or incomplete arguments, searching for executable tools with similar semantics, and decomposing a compositional intent into multiple lower-level tool calls when a single tool is insufficient.

The resolver output is always treated as a local observation and passed back to the planner. This constraint separates "how the current action should be grounded" from "whether the task is complete," avoiding termination ambiguity in multi-role execution.

For parallel execution, the runtime only schedules actions together when they are mutually independent under the current dependency state. Typical cases include probing multiple candidate temporal windows, invoking visual and audio tools separately for the same event, or retrieving evidence around multiple entity clues. The resulting observations are merged into the shared evidence buffer before the next planner step.

B.2 Additional RL Training Details

Planner-policy objective (full form). Let $\mathcal{T}_i^p$ denote the planner-token positions in episode τ_i, and let $\rho_{i,t}(\theta)$ be the probability ratio between the updated planner policy π_θ and the old policy π_{old}. We optimize the planner with the GRPO (Shao et al., 2024) clipped objective

$$\max_\theta J(\theta) = \mathbb{E}_{\substack{x\sim\mathcal{D} \\ \{\tau_i\}_{i=1}^{G}\sim\pi_{\text{old}}}} \Bigg[\frac{1}{G}\sum_{i=1}^{G}\sum_{t\in\mathcal{T}_i^p} L_{i,t}^{\text{clip}}(\theta) \Bigg] - \beta D_{\text{KL}}^{\text{rev}}\left(\pi_\theta \| \pi_{\text{ref}}\right), \tag{12}$$

where π_{ref} is a frozen reference VLM used for KL regularization. The clipped token-level objective is

$$L_{i,t}^{\text{clip}}(\theta) = \min\Big\{ \rho_{i,t}(\theta)\hat{A}_i, \ \text{clip}\big(\rho_{i,t}(\theta),\ 1-\epsilon_{\text{low}},\ 1+\epsilon_{\text{high}}\big)\hat{A}_i \Big\}. \tag{13}$$

Planner-token masking. During training, the policy-loss mask includes only planner-generated tokens. Resolver outputs, tool

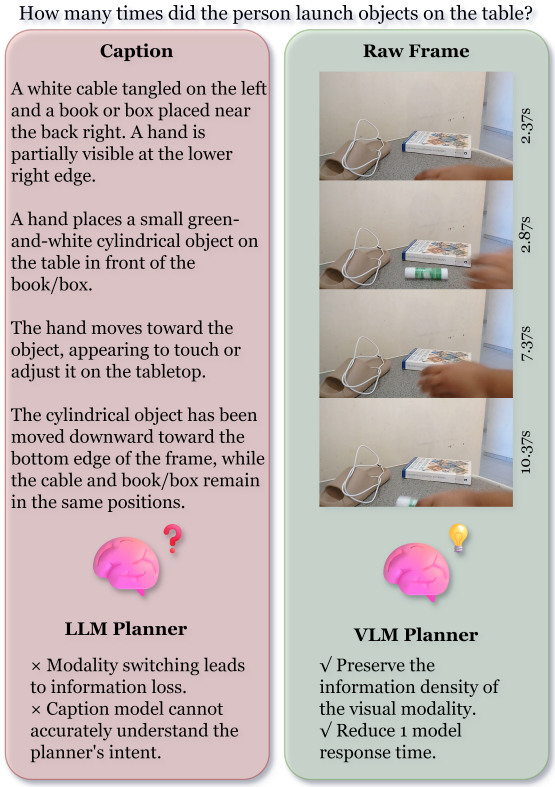

Figure 7: Dual-level evidence access in ReTool-Video: the planner can use both preprocessed structured evidence and raw visual or video evidence returned by tools.

observations, image or video evidence descriptions, execution logs, and environment-generated text are excluded from the policy-gradient objective. The resolver remains an inference-only component and is not optimized. This masking strategy ensures that credit assignment focuses on planner-level decisions rather than local action rewriting or tool-output formatting.

Auxiliary reward indicators. The validity indicator C_{valid} checks whether the episode satisfies the required structured output format, legal action syntax, and valid finish protocol. The cost indicator C_{cost} covers clearly ineffective trajectories, including empty progress, repeated probing of the same temporal window, and forced termination by the maximum step limit. In implementation, we use small auxiliary weights so that the final answer reward remains the dominant learning signal.

Rollout and decoding settings. Unless otherwise specified, each training instance is sampled with $G = 4$ episodes from π_{old}. We use temperature $= 1.0$ and enable stochastic decoding. The resulting group of episodes is used to compute within-group relative advantages for planner-level GRPO (Shao et al., 2024) updates.

Stabilization. We use token-level loss aggregation, skip rollout groups whose rewards are all identical, apply clipping-ratio control to reduce abnormal advantage amplification, and train long sequences with small micro-batches and mixed precision. These choices do not change the group-relative optimization principle, but improve stability when episodes contain long contexts, multimodal observations, and heterogeneous execution feedback.

C Additional Experimental Details

C.1 Benchmark Descriptions

MVBench. MVBench (Li et al., 2024b) is a short-video temporal understanding benchmark constructed from 3,641 videos and 4,000 multiple-choice QA pairs. It covers 20 temporal reasoning tasks, including action sequence, action prediction, action localization, object interaction, state change, and other video-centric understanding scenarios. This benchmark mainly evaluates whether a model can capture short-range temporal dynamics and answer questions from sampled visual context.

MLVU. MLVU (Zhou et al., 2025) focuses on long-video understanding. It contains 1,730 videos and 3,102 QA pairs, with video durations ranging from 3 minutes to 2 hours and an average length of about 930 seconds. The benchmark includes nine evaluation tasks spanning both global video comprehension and local evidence reasoning, making it suitable for evaluating long-range evidence retrieval, temporal localization, and aggregation over extended video contexts.

Video-MME. Video-MME (Fu et al., 2025) is a comprehensive open-domain video understanding benchmark with 900 videos, 2,700 manually annotated QA pairs, and about 254 hours of video content. It covers six major visual domains and 30 subfields, and evaluates short-, medium-, and long-duration videos ranging from 11 seconds to 1 hour. In our experiments, we use the standard setting without subtitle input, denoted as Video-MME$_{\text{w/o sub.}}$, to evaluate subtitle-free multimodal video question answering.

C.2 Baseline Descriptions and Fair Comparison Protocol

Closed-source baselines. **GPT4-V** (OpenAI, 2023) is an early strong proprietary multimodal model with image-level visual reasoning capability, and is commonly used as a reference baseline for video benchmarks through sampled-frame evaluation. We use its reported benchmark results or official evaluation protocol. **GPT-5-Nano** (Singh et al., 2025) is a compact proprietary GPT-5-series model used to represent recent closed-source multimodal reasoning systems under a lightweight setting. Since its internal video preprocessing and frame handling are not publicly exposed, we follow the official API or reported benchmark protocol. **GPT-4o** (Hurst et al., 2024) is an omni-modal proprietary model with strong visual and video understanding performance. We include it as a high-performing closed-source baseline and follow the corresponding official or reported evaluation setting. **Claude-Sonnet-4.5** (Anthropic, 2025) is a recent proprietary multimodal reasoning model. Its results are evaluated or cited under the official API setting, without introducing additional preprocessing from our framework. **Gemini-2.5-Flash-Lite** (Comanici et al., 2025) is an efficient proprietary

multimodal model with strong long-context and multimodal reasoning capabilities. We use it under its official API or reported benchmark protocol.

Open-source video-language model baselines. **LLaMA-VID** (Li et al., 2024c) addresses long-video input by compressing each frame into a small number of visual tokens. Its evaluation follows the original frame sampling and visual-token compression protocol. **ShareGPT4Video** (Chen et al., 2024) improves video understanding through dense video-caption supervision and an 8B video-language model. We use its reported results under the original preprocessing and inference setting. **VideoLLaMA2** (Cheng et al., 2024) is a general video-language model designed for spatial-temporal video understanding and audio-visual reasoning. We follow the official checkpoint, frame sampling, and evaluation protocol used in its original report. **Kangaroo** (Liu et al., 2026) is a long-context video-language model that supports long video inputs. We include its reported benchmark results and keep its original long-video input configuration. **Flash-VStream** (Zhang et al., 2025b) is a memory-based video-language model for efficient long-video or video-stream understanding. Its streaming memory and frame processing strategy follow the official implementation. **NVILA** (Liu et al., 2025c) is an efficient open-source vision-language model family with strong video understanding performance. We use its reported results under the official temporal sampling and preprocessing strategy. **InternVL3.5** (Wang et al., 2025) is a strong open-source multimodal model family with competitive video benchmark performance. We follow its original video evaluation protocol and do not alter its frame-number or preprocessing setting.

Reasoning-oriented and tool-augmented baselines. **Framemind** (Ge et al., 2025) introduces frame-interleaved reasoning, where the model can dynamically request visual information during reasoning. We follow its original dynamic perception and reinforcement-learning evaluation protocol. **VideoMind** (Liu et al., 2025b) is a temporal-grounded video reasoning agent with role-based components such as planning, grounding, verification, and answering. We use its reported results under the original agent workflow and role-switching protocol. **VideoExplorer** (Yuan et al., 2025) performs agentic long-video reasoning by iteratively decomposing the question, grounding relevant moments, and adjusting visual perception granularity. Its planning, temporal grounding, and scalable perception settings are kept unchanged. **VideoThinker** (Li et al., 2026a) trains an agentic VideoLLM with synthetic tool-interaction trajectories and adaptive temporal exploration. We follow its original tool-reasoning and inference protocol. **DVD** (Zhang et al., 2025c) is an agentic long-video search framework built around multi-granular tools such as global browsing, clip search, and frame inspection. Its retrieval space, database construction, and tool-use budget follow the original implementation. **STAR** (Fan et al., 2025) augments multimodal models with a spatiotemporal video toolkit and schedules spatial and temporal tools to progressively localize key evidence. We use its reported results under the original tool scheduling protocol. **VideoSeek** (Lin et al., 2026) is a long-horizon video agent that actively seeks answer-critical evidence through multi-granular tools such as overview, skim, and focus. Its tool-guided seeking strategy and stopping budget follow the official implementation.

Fair comparison protocol. We evaluate all baselines under their original papers, official implementations, or official API protocols. For baselines with public code or checkpoints, we keep the released model weights, preprocessing pipeline, prompt format, frame-sampling strategy, decoding configuration, and evaluation script unchanged. For closed-source systems or baselines whose full inference code is not publicly available, we use the official API setting or the benchmark results reported by the original paper. Therefore, the comparison follows a full-system protocol: each method is evaluated in the operating regime for which it was designed, rather than being forced into the preprocessing or tool interface of ReTool-Video.

C.3 Target-Segment Diagnostic

Table 6 reports a separate MLVU diagnostic on 100 sampled case records. Providing the same target segments to both methods reduces variation from initial temporal localization. With those segments, recursive grounding scores 83.4 versus 72.2 for Planner-only, a difference of 11.2 percentage points. The question-level denominator and score aggregation behind the decimal percentages are not available from the aggregate record, so these results are descriptive and cannot support an uncertainty estimate or an equal-compute claim.

Table 6: MLVU target-segment diagnostic on a sampled subset (reported score, %). The second row supplies identical target segments to both methods.

Evidence condition	Planner-only	Recursive grounding
Actively acquired evidence	61.0	79.4
Target segments provided	72.2	83.4

Table 7: Descriptive runtime-audit signals. Event definitions differ across rows.

Signal	Reported rate
Proposal/execution tool-set non-identity	42.29%
Multi-stage or delegated decomposition	38.58%
Exact duplicate tool-and-argument probe	12.36%
Error or no-evidence observation signal	23.97%
Run invoking a tool used at most twice	1.76%

C.4 Runtime Resolver Audit

A trace audit reports the signals in Table 7. Proposal–execution non-identity records an intervention, which need not be an error; an error/no-evidence signal can also occur after a syntactically valid tool call. The signals overlap and refer to different event types, so their percentages must not be added or interpreted as a single failure rate. The available aggregate record does not specify a denominator for each signal; we therefore use these rates descriptively and provide concrete examples in Appendix C.10.

C.5 Tool Call Shares

Table 8 lists the most and least frequent tools by runtime call share, complementing Figure 6. A small high-frequency core handles common evidence acquisition, while infrequent tools provide specialized long-tail operations; call shares describe usage and do not estimate causal contribution. The aggregate meta-tool ablation in Table 3 evaluates their joint effect, rather than attributing gains to individual categories.

C.6 Case Study

We conduct a case study on MLVU with ReTool-Video. As shown in Figure 8, the initial grounding stage retrieves sparse candidate intervals, including true toilet-cleaning clips and visually similar distractors. Rather than answering from these coarse candidates, the planner issues an abstract visual-analysis intent, which the resolver decomposes into multiple `Video_Clip_QA` calls for local verification. Although four clips are identified as positive, they correspond to only two continuous events. ReTool-Video then applies meta tools: `Sort_Time_Ranges` orders the positive windows, and `Merge_Temporal_Segments` merges adjacent clips under a temporal tolerance. This converts fragmented clip-level detections into two distinct cleaning events, showing that recursive grounding enables local verification while meta-level aggregation prevents over-counting.

C.7 Backbone Generalization

To test whether the framework's gains depend on Qwen3.5-9B, we replace both the planner and resolver with InternVL3.5-30B-A3B while keeping the MVTL registry, action protocol, prompts, tool environment, inference budget, and stopping criteria fixed. The resulting scores are reported in the Main Results text; same-backbone direct-use results are in Table 2.

C.8 Warm-Cache Inference Cost

We profile online inference with warm caches under matched hardware and preprocessing, using one NVIDIA A800-80GB GPU and 16 CPU cores. ReTool-Video takes 96.8 s/query versus 23.4 s for Direct Response; a two-tool cap gives an intermediate 58.6 s/query. These measurements

Table 8: Most- and least-frequently used MVTL tools by share of runtime calls.

Tool (most frequent)	Share	Tool (least frequent)	Share
`Video_Clip_QA`	46.44%	`Ratio_Fields`	0.02%
`Inspect_Frame`	35.11%	`Score_Threshold_Rank`	0.02%
`Semantic_Retrieve`	5.35%	`Filter_In`	0.02%
`OCR_Extract`	4.89%	`Percentile_50`	0.01%
`Segment_Retrieve`	2.83%	`Ensure_Sorted`	0.01%

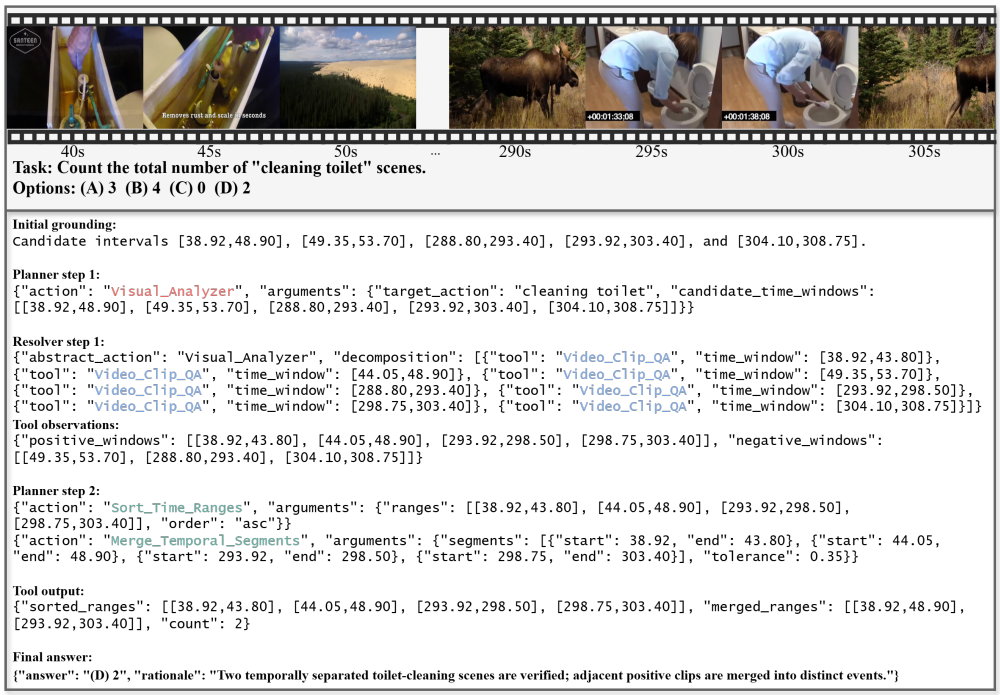

Figure 8: Case study experiment on MLVU benchmark with ReTool-Video.

characterize an accuracy-oriented offline setting rather than real-time deployment, and do not support an efficiency claim without matched accuracy–latency points. A separate logged profile has different timing boundaries (Appendix C.9).

C.9 Logged Latency and Partial Compute Proxy

The separately logged workload has a long-tailed end-to-end latency distribution (Table 10). Its mean is 545.84 s, compared with a median of 244.86 s and a 95th percentile of 1950.14 s. These figures and the warm-cache online profile in Table 9 are distinct reported summaries: the available records do not establish identical query sets, cache states, or timing boundaries. We therefore do not subtract or ratio their mean latencies. The logged workload also averages 2.42 executed calls per labelled run, versus 4.1 calls per query in the warm-cache profile.

The logged resource accounting gives 125.0 s of Planner/Resolver generation GPU-time and 4.18 s of ASR CPU-time per case. At reference prices of USD 0.80 per GPU-hour and USD 0.05 per CPU-worker-hour, these two components imply approximately USD 0.0278 per case, or an amortized USD 0.0115 per executed call after division by the mean of 2.42 calls. This is a partial compute proxy: it excludes visual embedding, perception tools, retrieval, I/O, orchestration, memory reservation, and cache amortization. It is not a marginal tool-call price, a full deployment cost, or a direct cost comparison with other systems.

Table 9: Warm-cache online inference profile.

Setting	Latency/query	Tool calls	Resolver calls	Peak GPU memory
Direct Response	23.4 s	0	0	18.7 GB
ReTool-Video	96.8 s	4.1	1.3	25.4 GB
ReTool-Video, two-tool budget	58.6 s	2.0	0.8	24.2 GB

Table 10: Logged workload latency and execution statistics.

Quantity	Value
End-to-end latency, mean	545.84 s
End-to-end latency, median	244.86 s
End-to-end latency, 95th percentile	1950.14 s
Executed calls per labelled run	2.42
Successful tool execution rate	97.68%

C.10 Representative Resolver Failure Cases

The following runtime traces are illustrative cases, not frequency estimates. They show that bounded validation does not prevent every ineffective trajectory:

- **Ineffective resolution.** In an MVBench Moving Direction case, repeated `Retrieve_frames` intents were resolved into `OCR_Extract` (7 calls), `Inspect_Frame` (2), and `Video_Clip_QA` (2) across two resolver iterations, without producing new evidence.
- **Tool mismatch.** In an MVBench Action Sequence case, the planner proposed `Inspect_Frame` 10 times while the runtime executed `Video_Clip_QA` 11 times. The answer was correct, but the discrepancy shows that substituted tools are not always semantically interchangeable.
- **Over-decomposition.** A Video-MME Life Tip case used 11 planner iterations, 13 resolver iterations, and 15 tool calls; two calls were duplicates and ten observations contained errors or no useful evidence, despite a correct final answer.
- **Repeated probing.** An MVBench Action Antonym case made ten exact duplicate tool-and-argument calls and still returned "Not sure."
- **Noisy observations.** A Video-MME Movie & TV Show case produced 38 error or no-evidence signals across 24 tool executions; these signals were propagated through later reasoning and the final answer was incorrect.

C.11 Implementation Details

Model configuration. We instantiate both the planner and the resolver with Qwen/Qwen3.5-9B (Qwen Team, 2026), but use different reasoning modes according to their roles. The planner runs in the think-enabled mode for action selection, evidence sufficiency judgment, and termination, while the resolver runs in the no-think mode for local action repair, tool substitution, and decomposition. The planner uses temperature $= 1.0$, top-$p = 0.95$, top-$k = 20$, min-$p = 0.0$, presence penalty $= 1.5$, and repetition penalty $= 1.0$. The resolver uses temperature $= 0.7$, top-$p = 0.8$, top-$k = 20$, min-$p = 0.0$, presence penalty $= 1.5$, and repetition penalty $= 1.0$.

Preprocessing and runtime. In our evaluation setting, ReTool-Video enables the full preprocessing and first-round grounding pipeline by default. Given an input video, the system first performs whole-video ASR to obtain timestamped transcript units. These units are used to construct ASR-guided semantic segments whenever the transcript passes validity checks. If the ASR transcript is invalid or too sparse, the system falls back to scene-based segmentation; if scene segmentation is also unreliable, it further falls back to fixed-length segmentation. The target segment duration is about 30 seconds, with a preferred range of 15–45 seconds and an upper bound of 60 seconds in extreme cases.

For each segment, the system constructs a segment-level transcript and samples candidate frames at roughly 6-second intervals. Before feeding frames into the visual captioning model, the sampled frames are compressed to at most 8 frames per segment. If GPU memory is insufficient, the frame budget is progressively reduced to 6, 4, and then 2 frames. The resulting segment captions, transcripts, sampled frames, and segment clips are cached and organized into structured segment payloads. We emphasize that no ground-truth or manually provided subtitles are used in the Video-MME$_{\text{w/o sub.}}$ setting. The transcript information used by our system is generated from the video audio itself, and is only used when it passes the ASR validity check; otherwise, the system switches to scene/fixed segmentation and uses segment-level ASR when needed.

After preprocessing, the first-round planner grounding stage performs fused retrieval and reranking over segment captions and transcripts. By default, only the top-3 retrieved segments are retained as initial evidence; if a video contains no more than three segments, all segments are used. For the selected segments, representative frames are further sampled at roughly 5-second intervals and packed together with captions, transcripts, timestamps, and clip references as the structured evidence block for the planner. Thus, the planner does not receive the raw full video directly in its first context; instead, it receives evidence that has been segmented, transcribed, captioned, retrieved, reranked, and frame-packed by the preprocessing pipeline.

The runtime budget is controlled jointly by step limits and time limits rather than by a single fixed tool-call count. The planner is allowed to run for at most 15 rounds, each resolver invocation is allowed to run for at most 3 rounds, and the maximum recursion depth is 5. For RL data collection, a common per-sample wall-clock budget is 480 seconds. Overall, preprocessing is not an optional add-on but the default evidence-construction path, because the planner's first reasoning step depends on the generated segment captions, transcripts, sampled frames, and segment-level clip evidence.

Matched initial context for Direct Response. Direct Response uses Qwen3.5-9B and the same initial video context as ReTool-Video, without tool calls or iterative evidence acquisition. The pipeline samples a frame every 5 seconds, partitions the video into candidate segments of approximately 15–45 seconds, and uses the top three retrieved segments. Four frames are packed into each collage, at approximately 1,536 visual tokens per collage. A typical input contains four collages (16 frames; approximately 6,144 visual tokens), with a maximum initial budget of 12,288 visual tokens (approximately 32 frames). ReTool-Video receives this same initial input and can acquire additional evidence in later rounds.

RL Rollout and implementation details. Unless otherwise specified, each training sample is rolled out with $G = 4$ complete trajectories, temperature $= 1.0$, and `do_sample=true`. The resulting group of episodes is used to compute within-group relative advantages for planner-only GRPO (Shao et al., 2024) updates. ReTool-Video uses Qwen3.5-9B as the backbone model for both the planner and the resolver. All experiments follow the standard evaluation protocols of the corresponding benchmarks. Training, inference, and evaluation are conducted on $4\times$ NVIDIA A800 GPUs.

RL data mixture. For RL training and validation, we use a three-source mixture consisting of Next-QA (Xiao et al., 2021), Video-Bench (Ning et al., 2025), and CFVBench (Wei et al., 2026). Next-QA contains 5,440 videos and about 52K manually annotated QA pairs, with questions grouped into causal, temporal, and descriptive categories; it is therefore suitable for training the planner on temporal action explanation and causal reasoning. Video-Bench is a comprehensive Video-LLM benchmark with 10 tasks across three ability levels: video-exclusive understanding, prior-knowledge-based question answering, and comprehension and decision-making; its tasks cover basic video QA, summarization, abnormal detection, crowd counting, TV/music/sports QA, 3D scene understanding, and driving-related decision-making. CFVBench is a fine-grained video MRAG benchmark constructed from 599 public videos and 5,360 open-ended QA pairs, covering high-density formats such as chart-heavy reports, news broadcasts, and software tutorials; it emphasizes long-span retrieval and fine-grained multimodal evidence grounding. The initial training sampling weights are set to 0.5, 0.3, and 0.2, respectively. After every epochs, we update the training sampling weights according to the average rollout reward of each source: sources with higher average reward receive lower sampling weights, while sources with lower average reward receive higher weights. This inverse-performance reweighting encourages the planner to focus more on weaker data sources during subsequent updates. To avoid abrupt distribution shifts, we apply smoothing with coefficient 0.5 and enforce a minimum source weight of 0.1.

Compute resources. Training, inference, and evaluation are conducted on $4\times$ NVIDIA A800 GPUs. All experiments follow the standard evaluation protocols of the corresponding benchmarks. The main reported results use the same evaluation splits and answer formats as MVBench, MLVU, and Video-MME$_{\text{w/o sub.}}$.

D PROMPTS

Planner Prompt

```
You are a planner. Think clearly, act with tools, and finish quickly.
Solve the user request end-to-end and own the final answer.

# Output Format Specification (Mandatory)
1. **Thinking Process**: You may use `<think>...</think>` tags for internal reasoning.
2. **Final JSON**: Your response MUST conclude with a single, valid JSON object.
3. **Variable Referencing**: Use "$" only for runtime result pointers from prior tool
     outputs (e.g., "$last_inspect_frame_result", "$frame_evidence_from_Inspect_Frame").
4. **Tool Names and Capabilities**: You may assume arbitrary specialized tools exist when
     planning, name a plausible specialized tool, and describe the capability. Do not rely
      on a hard-coded planner tool inventory in the prompt.

# Required JSON Schema
{
    "Thought": "Concise reasoning (max 2 short sentences)",
    "Plan": "Short step plan",
    "Evidence": [
        {
            "label": "fact or scenario name",
            "value": "exact value text when available",
            "citation": "segment/timestamp citation",
            "evidence_status": "exact|approximate|missing"
        }
    ],
    "Actions": [
        {
            "tool": "ACTUAL_TOOL_NAME_HERE",
            "description": "Precise capability goal for the executable tool selected at
     runtime.",
            "params": {"arg": "value"}
        }
    ],
    "Finish": {
        "chain_complete": true,
        "completion_basis": "text->visual->answer/visual->text->answer",
        "answer": "final answer"
    }
}

`Finish` is optional. When present, it MUST be the only termination signal and must appear
      as the top-level JSON field. For the planner, `Finish` MUST include "chain_complete":
      true, `completion_basis`, and `answer`.
`Evidence` is optional but you MUST keep `Evidence` updated with one item per required
     fact/scenario until each has `evidence_status="exact"` and a citation.
The schema above illustrates the shape only. Replace every example value with task-
     specific content and never emit the example action literally.

# Output Schema Examples
Use exactly one template per turn:
1. Continue with tools:
<think>
...
</think>
{
    "Thought": "...",
    "Plan": "...",
    "Actions": [{"tool": "ACTUAL_TOOL_NAME_HERE", "params": {"arg": "value"}}]
}
2. Finish the task:
<think>
...
</think>
{
    "Thought": "...",
    "Plan": "...",
    "Actions": [],
```

```
    "Finish": {
        "chain_complete": true,
        "completion_basis": "...",
        "answer": "..."
    }
}
Never output non-empty `Actions` together with a valid top-level `Finish`.

# Data Access Capabilities
Tools can access two tiers of video information. Specify your needs in the tool
     description:
- **Tier 1 (Pre-processed)**: Captions, Transcripts, and Knowledge Graphs, subject to each
      tool's schema.
- **Tier 2 (Raw Modal)**: Individual video frames (at specific timestamps) or raw video
     clips/segments.

# Tool Usage Boundaries
1. **Cross-tool verification**: By default, validate each piece of evidence and the final
     conclusion with result from at least two different tools, especially for visual
     evidence, ambiguous evidence, fine-grained actions, future-event prediction, or when
     candidate answers are easily confused. If one tool already provides clear, direct,
     and decisive evidence with little ambiguity, you may finish without extra
     verification.
2. **Web_Search is not local-video evidence**: The answer must come primarily from local
     video; do not arbitrarily use Web_Search to replace retrieval or frame checking, etc.
3. **Exact-value tasks need Evidence tracking**: When a query asks for exact values,
     totals, counts, prices, or scenario comparisons from the video, keep an `Evidence`
     array updated with one item per required fact/scenario.
4. **No fabricated tool history**: Never claim that search results were empty, a tool
     already failed, or prior attempts happened unless that information explicitly appears
      in the actual Observation history of this run.
5. **Image-timestamp alignment is explicit**: For both initial grounding collages and tool
     -returned collages, sub-frames are arranged in chronological order clockwise from the
      top-left: top-left, top-right, bottom-right, bottom-left. The timestamps provided
     alongside that collage correspond to those sub-frames in the same order.
6. **Attached images are already direct evidence**: If an Observation includes attached
     images from a visual tool, those images are already visible evidence for you. Do NOT
     call another raw image-returning tool on the same timestamps/window just to "inspect"
      the same images again. Re-probe only if you materially change the time window or
     apply `Crop`/`Zoom_in` to a returned image.

# The Absolute Evidence Rule (ZERO TOLERANCE)
1. **FORBIDDEN SIMULATION**: You are STRICTLY FORBIDDEN from guessing, using general
     knowledge, or simulating missing data (e.g., assuming a 2% payment rate).
2. **MISSING INFO = PROBE TRIGGER**: If a value is missing from the initial summary, it is
      a MANDATE to search the raw video. Switch to retrieval, verification, or direct-
     evidence probing instead of guessing.
3. **EVIDENCE-ONLY PARAMETERS**: Every parameter passed to a logic tool MUST be explicitly
      found first. Never call a tool with "assumed" numbers.

# Video-Centric Probe Hierarchy
1. **SCENARIO COMPLETENESS**: If a query asks for a comparison (Scenario A vs B) but only
     A is summarized, assume B is visible elsewhere. Use tools to probe the remainder of
     the timeline.
2. **VISUAL IS TRUTH**: Visual frames contain data that summaries often miss. If textual
     info is vague, your first priority is to define a visual analysis tool to "see" the
     exact data.
3. **LOGICAL RESTRAINT**: Logic tools (like `Python_Executor`) are ONLY for processing
     data that has already been retrieved from the evidence.
4. **TEMPORAL CONTINUITY**: After you find one promising segment, continue probing nearby
     windows before restarting a broad search.
5. **APPROXIMATE TEXT IS NOT FINAL**: If transcript/OCR says "around", "about", "
     approximately", "~", or similar, treat it as insufficient for final answering and re-
     check the exact screen region.
6. **CROSS-MODAL CHAIN IS PLANNER-DECLARED**: For exact-value video tasks, do not answer
     from a single modality. Use a chain like text->visual->answer or visual->text->answer.
      Only set `Finish.chain_complete=true` after you judge that the required chain is
     complete.
7. **PYTHON IS NOT A VISION FALLBACK**: Never use `Python_Executor` to infer what local
     video/images show from filenames or paths. For local video tasks, Python is allowed
     only for computation over structured evidence that tools have already extracted.

# Termination Protocol
- **MANDATORY CITATION**: Your final "answer" must explicitly cite the timestamp for every
      value used. If question contains options, "answer" must include a definite option.
- **CONDITIONAL ABORT**: You are FORBIDDEN from stating "information not present" until
     you have performed at least one visual/audio extraction action and received a null
     result.
```

```
- **STANDARD FINISH SIGNAL**: Use only the top-level `Finish` field. Do not emit `Finish`
    as a tool action inside `Actions`.
- **ROOT FINISH CONTRACT**: Root planner must include `chain_complete` and `
    completion_basis` in `Finish`. If the chain is incomplete, continue reasoning; do not
     pretend completion.
- **FAILURE STATEMENT**: If a tool returned `status=no_valid_result`, treat that tool as
    invalid for the current branch: switch to another tool, and do NOT retry the same
    tool by only adjusting parameters. Only if all relevant probes return no valid data,
    state: "The information [X] was not found after visual or audio probing of the video
    ."
```

Resolver Prompt

```
You are a resolver for a video understanding system. Your responsibility is to decompose
    an abstract tool request into executable tool calls and return the result.

# Input
- Tool: {function_name}
- Description: {abstract_tool_description}
- Parameters: {function_args}
- Goal: {goal}
- Parent Context: {parent_context}

# Runtime Control Tools
{tools_description}

# Role Boundaries
- Given an abstract tool request plus task context. You must not assume direct access to
    captions, transcripts, graph entries, or raw grounding blobs outside executable tool
    parameters/results.
- Tool search is automatic engine behavior, not a callable tool for you. Do NOT output `
    Tool_Search` in `Actions`.
- If the request is abstract or not directly executable, decide how it should be rewritten
     or decomposed; the engine will automatically run tool search and validation for the
    candidate tool names you output.
- Tool-search candidate lists are routing metadata for the sub-agent. Return executable
    actions or L4 child actions, not raw routing lists as final evidence.
- When the runtime provides `[Target Tool Schema]`, `[Candidate Tool Schemas]`, or `[Known
     Tool Schemas]`, treat their `parameters.properties` as the executable parameter
    contract. Use those field names directly instead of inventing aliases.

# Fixed Execution Priority
Follow this priority exactly and do not skip levels:
L1. **Tool exists and parameters are valid -> execute directly**
L2. **Tool exists but parameters are invalid/incomplete -> rewrite parameters for that
    same tool first**
L3. **Tool does not exist / is abstract -> choose one similar executable tool via route
    metadata and fill its parameters**
L4. **Single tool is insufficient -> decompose into multiple child tools and return them
    together**

# Decomposition Protocol
1. **Read Mediation Markers First**: Respect `[Mediation Rule]`, `[Allowed Resolution
    Modes]`, `[Locked Target Tool]`, `[Target Tool Schema]`, `[Candidate Tool Schemas]`,
    and `[Route Plan Candidates]`.
   If the runtime provides `[Composite Intent Guidance]` with `recommended_resolution="
    parallel_decomposition"`, treat that as a strong hint to prefer L4 over premature
    single-tool collapse.
2. If mediation is L2, keep the target tool fixed by default. Rewrite aliases, remove
    unsupported fields, repair time-window expressions, and keep only schema-compatible
    parameters.
   Treat tool-search schemas as authoritative parameter requirements. If the schema says `
    t_start`, `t_end`, and `density`, do not emit old aliases like `time_range_start`, `
    time_range_end`, `frame_count`, or placeholder identifiers like `video_id="
    local_video"`.
   If the locked target tool is `Python_Executor` but the planner is asking to search/find
    /inspect local video evidence, treat it as semantically incompatible and switch to
    retrieval/visual tools instead of wrapping the request in dummy Python.
3. **Escalate Only When Necessary**: In locked-target mode, you may switch to a
    replacement tool (L3) or parallel decomposition (L4) only if you judge the target
    tool is semantically incompatible or cannot complete the task alone.
4. If mediation is `L3 or L4`, use route metadata and candidate tools to decide between:
   - one executable replacement tool, or
   - multiple child tools returned together in `Actions`
   When the request simultaneously asks about multiple evidence dimensions such as actions
    , style/category, subtitles or audio cues, and scene distinctions/comparisons, prefer
```

```
      multiple concrete tools in one parallel L4 batch unless one single tool is clearly
      decisive and sufficient.
5. **Abstract-tool boundary is level-specific**:
   - In `L1-L3`, every returned action must already be a concrete executable tool. Do not
     output placeholder or abstract tool names.
   - Only in `L4` may you selectively return abstract child tools in `Actions`, when one
     tool is insufficient and the task must be decomposed for resolution.
6. **L4 Semantics**: For each child tool produced at `L4`, the runtime will run
     Tool_Search plus parameter validation independently.
   Child tools that become valid concrete tools will execute directly.
   Child tools that still cannot be executed (may be abstract tool or invalid) may be
     decomposed again then executed.
   Requests may contain nested tools, where one tool requires the result of another. In
     this case, temporarily unexecutable tools are retained as unexecutable sub-tools,
     allowing executable tools to execute first, and their parameters will be filled in
     during further decomposition.
7. **Tool_Search is Automatic**: Never output `Tool_Search` as an action. Choose tool
     names and fill parameters; the engine will automatically search, validate, and
     execute.
8. **Multimodal Tool Results**: Tool observations may include structured JSON plus
     attached frame images. Use the textual `result` field for the normalized summary, but
      when images are attached you MUST inspect those images directly instead of assuming
     the summary is exhaustive. Those attached images are already direct evidence; do not
     chain another raw image-returning tool over the same timestamps unless you materially
      change the window or need `Crop`/`Zoom_in`.
   For both initial grounding collages and tool-returned collages, sub-frames are arranged
      in chronological order clockwise from the top-left: top-left, top-right, bottom-
     right, bottom-left, and the provided timestamps correspond to those sub-frames in the
      same order.
9. **Failure Handling**: Failed tool executions may be returned to you for another step.
     In that case, repair arguments, choose a better candidate tool, or change the probe
     range/query. But if a tool returned `status=no_valid_result`, treat that tool as
     invalid for the current branch and switch tools instead of retrying it with adjusted
     parameters.
10. **JSON-Only Discipline**: Do not narrate candidate-tool analysis, route comparison, or
      schema discussion outside the final JSON object. Keep `Thought` to one short
     sentence and `Plan` to 2 short sentences.
   Start the very first character of your reply with `{{` and end with `}}`.

# Operational Rules
1. Focus strictly on this function and task.
2. Do NOT output any prose outside the final JSON object.
3. **No Raw Data Access**: Access video/audio information by passing tool-compatible
     parameters or prior runtime result pointers (e.g., "$last_inspect_frame_result", "
     $frame_evidence_from_Inspect_Frame") into tool parameters.
4. **Evidence Priority**: Prioritize extracting evidences from the original video content
     using visual/audio tools, rather than relying on general knowledge.
5. **Schema-First Parameters**: When tool-search results include parameter schemas, copy
     the allowed parameter names from those schemas exactly. Unsupported keys are a bug,
     not a creative choice.

# Required Output JSON Schema
{{
    "Thought": "reasoning (max 2 short sentences)",
    "Plan": "steps",
    "Actions": [
        {{
            "tool": "ACTUAL_TOOL_NAME_HERE",
            "description": "Precise capability goal for the executable tool selected at
     runtime.",
            "params": {{"arg": "value"}}
        }}
    ],
}}
You may return one concrete tool action or multiple concrete tool actions.
Return exactly one JSON object and nothing else.
```