RLVU: RECURSIVE LONG VIDEO REASONING WITH COMPACT MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Using frontier models such as GPT-5 for video question answering can be prohibitively expensive at scale, making cost an important consideration alongside accuracy in applications such as surveillance, online videos, and recorded meetings. This motivates the use of compact reasoning backbones such as GPT-5 mini and Qwen3.5-27B. However, these models often struggle to locate and connect relevant evidence scattered throughout long videos. To address this challenge, we propose **RLVU**, which combines reusable memory and then recursively reasons over this memory to locate and integrate relevant evidence. Specifically, RLVU constructs video memory through top-down temporal refinement and bottom-up entity consolidation: event level context is propagated to shorter clips, while observations of recurring entities are merged into global records. Given a question, RLVU then performs recursive reasoning over the constructed memory through top-down question decomposition and bottom-up evidence integration. Compared with direct inference using the compact model like GPT-5 mini, the base RLVU configuration improves accuracy by 11.2%, 12.4%, and 4.3% on LongVideoBench, LVBench, and Video-MME, respectively. RLVU also consistently improves accuracy across all three benchmarks when using Qwen3.5-27B as the reasoning backbone, compared with direct inference using the same model. These results demonstrate that structured video memory and recursive reasoning enable compact reasoning models to more effectively locate, verify, and integrate evidence across long videos. The code is at `https://anonymous.4open.science/r/RLVU-4B58`.

1 INTRODUCTION

Video question answering over surveillance archives, online videos, and recorded meetings requires accurate answers at a reasonable cost. Using frontier multimodal models such as GPT-5 (OpenAI, 2025) can be expensive when large video collections receive repeated queries. We study compact reasoning backbones: GPT-5 mini, a lower cost GPT-5 variant, and Qwen3.5-27B (Qwen Team, 2026a). However, evaluations of long context reasoning show that compact models can lag behind frontier models even when they support large context windows (Bai et al., 2025b). A large window allows more evidence to be included, but using it effectively still requires selecting relevant details and combining them through several reasoning steps. In long videos, this becomes challenging when sparse events and recurring entities must be linked across distant clips. We thus aim to reduce both the context each call must process and the reasoning complexity.

Existing methods approach this problem through more efficient video processing and more flexible evidence access. Video models commonly process sampled frames (Lin et al., 2024; Bai et al., 2025a), but a limited frame budget can miss brief events, while denser sampling increases the input size. Frame selection (Zhu et al., 2025; Guo et al., 2025), adaptive visual compression (Wang et al., 2025a), and sparse attention (Song et al., 2025) offer different ways to control this processing cost. Efficient processing must also be paired with a way to locate and combine evidence for each question. Agentic methods address this need through adaptive retrieval and visual inspection (Wang et al., 2024b; Zhang et al., 2025b; Yuan et al., 2025), while structured video representations support access at different temporal scales (Wang et al., 2025c). We study how reusable memory that preserves event context and entity identity can guide local visual verification by compact reasoning models.

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.

To this end, we propose **RLVU** (Recursive Long Video Understanding), which couples reusable memory with local visual verification through two stages: recursive video memory construction and recursive video reasoning. **(1) Recursive video memory construction:** RLVU builds reusable video memory through top-down temporal refinement and bottom-up entity consolidation. Temporal refinement passes context from broad event descriptions to shorter clips, keeping local visual details connected to the surrounding events. Entity consolidation recursively merges local observations of the same person or object into global records with timestamps. This memory is built once per video and reused across questions. **(2) Recursive video reasoning:** RLVU combines top-down question decomposition with bottom-up evidence integration to reason recursively over this memory. Starting from the full question, the model generates focused subquestions and uses the temporal hierarchy and entity records to retrieve relevant clips. Subcalls verify local visual details in a focused context and can further decompose their subquestions when needed. Parent calls then combine the returned observations across events to answer the broader question. When more inference budget is available, optional Adaptive Test Time Computing (TTC) strengthens local verification through query diversification or multiple visual models.

Experiments on LongVideoBench, LVBench, and Video-MME show that the base RLVU configuration outperforms the evaluated baselines with both GPT-5 mini and Qwen3.5-27B, improving accuracy by 4.3 to 12.4 percentage points over direct inference. The GPT-5 mini configuration also outperforms our reproduced DVD baseline with 10.7% to 46.2% fewer online tokens, excluding memory construction and indexing.

Our contributions are threefold:

- We build reusable video memory that preserves event context and links recurring entities.
- We guide recursive visual verification with this memory for compact reasoning backbones.
- We show accuracy gains across three benchmarks and two backbones, supported by ablations.

2 Related Work

VLMs for Long Video Understanding. Video language models commonly process sampled frames (Lin et al., 2024; Bai et al., 2025a), while larger context windows support longer inputs (Gemini Team et al., 2024). To use the input budget more effectively, frame selection methods (Tan et al., 2026a; Zhu et al., 2025; Guo et al., 2025; Ye et al., 2025) prioritize relevant content, while visual token reduction methods (Wang et al., 2025a; Jiang et al., 2025) compress video representations. Sparse attention (Song et al., 2025) and hybrid Mamba architectures (Ren et al., 2025; Xu et al., 2025) provide complementary ways to reduce the cost of processing long sequences. RLVU focuses on organizing video evidence into reusable memory and accessing selected portions during reasoning.

Agentic Video Understanding. VideoAgent (Wang et al., 2024b) iteratively selects frames, while DVD (Zhang et al., 2025b), VideoDeepResearch (Yuan et al., 2025), and AVI (Gao et al., 2025) combine retrieval with visual inspection to gather evidence. LongVideoAgent (Liu et al., 2025) coordinates specialized agents, while VideoSeek (Lin et al., 2026) and LensWalk (Li et al., 2026) adapt visual exploration during reasoning. Other methods organize evidence for retrieval and integration: VideoRAG (Ren et al., 2026) combines textual knowledge graphs with visual representations, and MR.Video (Pang & Wang, 2025) aligns recurring entities across clip captions before combining local analyses. VideoTree (Wang et al., 2025c) builds a video hierarchy guided by the question, while VideoARM (Yin et al., 2025) builds and updates hierarchical memory during reasoning. RLVU builds memory independently of questions through context guided temporal refinement and entity consolidation, then guides local visual verification through recursive calls with compact reasoning backbones.

3 Methodology

Given a video V and a question q, RLVU constructs a reusable memory $\mathcal{M}(V)$ and answers the question through recursive evidence exploration. As shown in Figure 1, RLVU has two stages:

1. **Recursive Video Memory Construction:** top-down temporal refinement passes event context to shorter clips, while bottom-up entity consolidation merges local observations into global records.

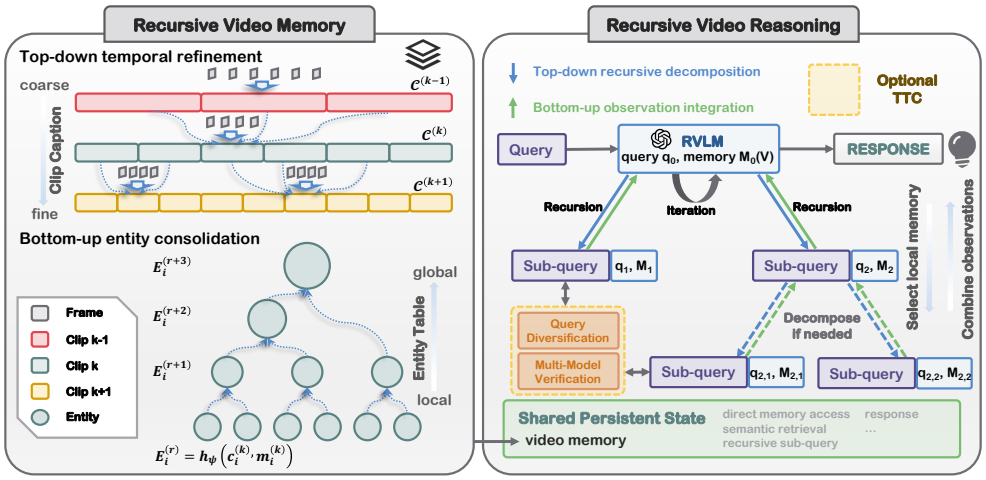


Figure 1: **Overview of the RLVU framework.** RLVU first builds reusable memory through top-down temporal refinement and bottom-up entity consolidation. During reasoning, the RVLM recursively decomposes the input query into localized sub-queries, selectively accesses the shared persistent video memory, and integrates returned observations bottom-up to support subsequent reasoning and the final response. Optional test-time computing (TTC) can further diversify or verify selected sub-query results before they are returned to parent calls.

Together, these processes organize events across temporal scales and connect recurring entities across clips.

2. **Recursive Video Reasoning:** top-down question decomposition creates focused subquestions for local verification, while bottom-up evidence integration combines the returned observations into an answer.

We denote this recursive reasoning procedure by $\mathcal{R}$:

$$a = \mathcal{R}(q, \mathcal{M}(V)). \tag{1}$$

Each call works with selected evidence and a focused reasoning task, supporting the use of compact reasoning backbones. When more inference budget is available, optional Adaptive Test Time Computing (TTC) strengthens local verification through query diversification or multiple visual models.

3.1 Recursive Video Memory Construction

Memory construction combines a temporal hierarchy with global entity records. Top-down temporal refinement passes broader event context to shorter clips, and bottom-up entity consolidation links observations of recurring people or objects across clips. The resulting memory $\mathcal{M}(V)$ is built independently of individual questions and reused across queries.

3.1.1 Top-Down Temporal Refinement

Temporal refinement builds clip descriptions from coarse to fine temporal scales. Descriptions of broader events guide those of shorter clips, helping local details retain their surrounding event context.

Formally, let $\mathcal{C}^{(\ell)}$ be the set of clips at temporal level ℓ, where larger ℓ corresponds to finer temporal granularity. Each clip $c_i^{(\ell)} \in \mathcal{C}^{(\ell)}$ is represented as

$$c_i^{(\ell)} = (I_i^{(\ell)}, F_i^{(\ell)}), \tag{2}$$

where $I_i^{(\ell)}$ denotes its temporal interval and $F_i^{(\ell)}$ denotes the frames sampled within that interval.

For each clip $c_i^{(\ell)}$ at level $\ell > 0$, let $\mathcal{P}_i^{(\ell)} \subseteq \mathcal{C}^{(\ell-1)}$ be the set of clips at the preceding coarser level that correspond to the same temporal region. Their previously constructed textual memories then provide coarse-grained context that recursively guides the construction of the current finer-level memory:

$$\Gamma_i^{(\ell)} = \{m_j^{(\ell-1)} \mid c_j^{(\ell-1)} \in \mathcal{P}_i^{(\ell)}\}, \quad m_i^{(\ell)} = f_\theta(\Gamma_i^{(\ell)}, F_i^{(\ell)}), \tag{3}$$

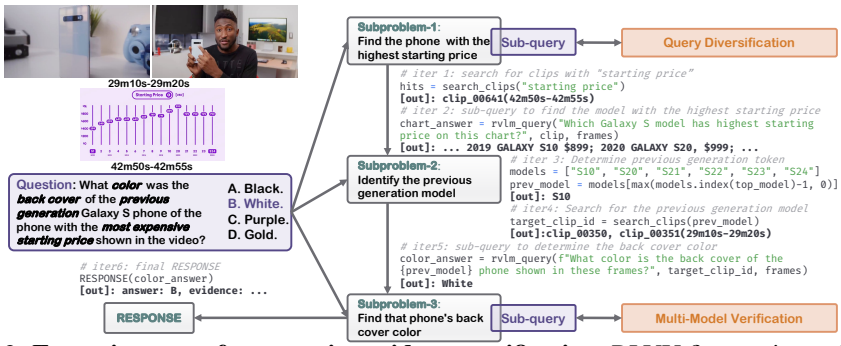


Figure 2: **Execution trace for recursive evidence verification.** RLVU first retrieves the price chart at 42m50s and uses query diversification to identify the Galaxy S20 as the model with the highest starting price shown. It then identifies the preceding generation, Galaxy S10, retrieves its appearance at 29m10s, and uses multiple visual models to verify its white back cover.

where $\Gamma_i^{(\ell)}$ represents the inherited temporal context from the preceding level and f_θ is a vision-language model prompted to generate the textual memory $m_i^{(\ell)}$.

This construction combines local visual details with the context inherited from broader events. During reasoning, the model can retrieve relevant clips at the appropriate temporal scale and inspect their evidence in a local context.

3.1.2 Bottom-Up Entity Consolidation

Temporal descriptions capture events within individual intervals, while many questions require linking the same person or object across distant clips. Entity consolidation addresses this need by recursively merging local observations into global records with timestamps.

Local entity extraction is first performed at an intermediate temporal level ℓ_e, which balances fine-grained visual evidence with sufficient temporal context. The resulting local entity tables are then recursively consolidated in a bottom-up manner:

$$E_i^{(0)} = h_\psi(c_i^{(\ell_e)}, m_i^{(\ell_e)}), \qquad E_j^{(r+1)} = g_\omega(E_{2j}^{(r)}, E_{2j+1}^{(r)}). \tag{4}$$

Here, $E_i^{(0)}$ denotes the local entity table extracted from clip $c_i^{(\ell_e)}$ and its associated temporal memory $m_i^{(\ell_e)}$, while r indexes the entity consolidation stages and is independent of the temporal level ℓ_e. The extractor h_ψ identifies local entity observations with their types, descriptions, and temporal anchors, whereas g_ω recursively consolidates neighboring tables according to semantic compatibility, type consistency, and temporal continuity. Records referring to the same entity are merged, while ambiguous observations remain separate to avoid erroneous associations. This process continues until a single global entity memory $E^{(\text{final})}$ is obtained.

The final video memory integrates the hierarchical temporal memories $\{m_i^{(\ell)}\}_{i,\ell}$, which capture multi-resolution temporal context, with the consolidated global entity memory $E^{(\text{final})}$, which maintains persistent, time-anchored entity representations:

$$\mathcal{M}(V) = \{\{m_i^{(\ell)}\}_{i,\ell}, E^{(\text{final})}\}. \tag{5}$$

During reasoning, temporal descriptions help locate relevant events, while entity records help trace recurring people or objects across clips.

3.2 Recursive Video Reasoning

Recursive video reasoning combines top-down question decomposition with bottom-up evidence integration over $\mathcal{M}(V)$. The model uses temporal descriptions to locate events and entity records to connect observations across clips, then verifies the visual details needed to answer the question. Figure 2 illustrates how recursive calls retrieve, verify, and combine evidence across events.

3.2.1 Top-Down Question Decomposition

The model divides a question into focused subquestions and retrieves a local memory $\mathcal{M}_i \subseteq \mathcal{M}(V)$ for each subquestion q_i. A recursive call processes this selected evidence and returns an observation:

$$o_i = \mathcal{R}(q_i, \mathcal{M}_i). \tag{6}$$

Each call has its own context containing the subquestion, selected memory information, and observations collected during reasoning. The call can answer from this evidence or inspect the corresponding frames to verify local visual details. When needed, it can further decompose its subquestion and invoke additional calls:

$$o_{i,j} = \mathcal{R}(q_{i,j}, \mathcal{M}_{i,j}), \quad j = 1, \ldots, n_i, \tag{7}$$

where $q_{i,j}$ and $\mathcal{M}_{i,j}$ are the child subquestion and its selected evidence, and n_i is the number of child calls. Decomposition is chosen as needed, so a call that can resolve its subquestion locally does not require further recursion.

3.2.2 Bottom-Up Evidence Integration

Each child call returns its observation to the parent, which adds it to its local reasoning state $\mathcal{H}_i$:

$$\mathcal{H}_i \leftarrow \mathcal{H}_i \oplus o_{i,j}, \tag{8}$$

where $\oplus$ denotes adding the returned observation to the state. The parent combines these observations with evidence it has already collected to relate events, check entity identities, and determine whether more evidence is needed. Once the current subquestion can be answered, its result is returned to the next parent; the root call produces the final answer. Detailed subcall histories remain in their local contexts, while their returned observations support reasoning across distant events. This keeps each call focused on selected evidence and a smaller reasoning task. We implement these calls as a Recursive Vision Language Model (RVLM) using the execution approach of Zhang et al. (2025a), with separate contexts for subcalls and persistent access to the temporal and entity memory. Appendix A.1 provides the execution interface and algorithm, with prompt templates in Appendix D.

3.2.3 Optional Adaptive Test Time Computing

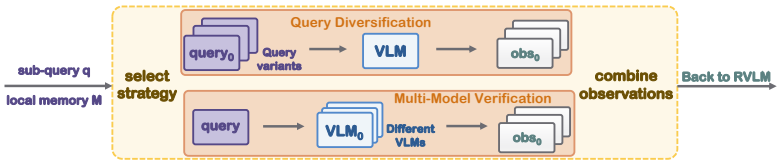


Figure 3: Optional TTC for local evidence verification.

Local verification can remain difficult when a subquestion is ambiguous or the evidence requires reading small text, recognizing small objects, or counting. With additional inference budget, optional Adaptive Test Time Computing (TTC) refines these observations before they return to the parent call.

Formally, as shown in Figure 3, given a subquestion q_i and its corresponding local memory $\mathcal{M}_i$, TTC first selects an inference strategy τ_i based on the characteristics of the query and available evidence:

$$\begin{aligned} &\tau_i = P(q_i, \mathcal{M}_i), \\ &\tau_i \in \{\text{query-diversify}, \text{multi-model}\}, \end{aligned} \tag{9}$$

and then aggregates the resulting candidate observations $\mathcal{O}_i$ into a final refined observation:

$$\mathcal{O}_i = \mathrm{TTC}_{\tau_i}(q_i, \mathcal{M}_i), \quad \tilde{o}_i = A(q_i, \mathcal{M}_i, \mathcal{O}_i), \tag{10}$$

where TTC_{τ_i} executes the selected strategy, and $A(\cdot)$ combines its outputs into a single observation returned to the parent call. When enabled, TTC selects one of two refinement strategies:

- **Query Diversification** checks the same local evidence with multiple formulations of the subquestion, including variants that distinguish plausible interpretations.
- **Multi-Model Verification** applies several visual models to the same subquestion and evidence, then combines their observations.

Both strategies preserve the scope of the original subquestion and return a single observation.

4 EXPERIMENTS

4.1 SETUP

Benchmarks. To rigorously assess RLVU's capabilities in long-video understanding, we evaluate it on three benchmarks specifically designed for extended temporal contexts. **(1) LongVideoBench (Wu et al., 2024)** is a video-language QA dataset with interleaved video and subtitle streams. The validation set comprises 753 videos and 1,337 questions, of which 188 videos and 564 questions belong to the Long subset. **(2) LVBench (Wang et al., 2024a)** contains 1,549 multiple-choice questions from 103 videos spanning 30 to 140 minutes, requiring evidence aggregation across distant timestamps and challenging long-range temporal reasoning. **(3) Video-MME (Fu et al., 2024)** is a multimodal benchmark organized by video duration. We evaluate its Long subset without subtitles, covering 300 videos and 900 questions, with an emphasis on visual memory rather than textual cues.

Implementation Details. We construct the structured video memory using Qwen3.5-27B (Qwen Team, 2026b) with a three-level temporal hierarchy, where coarse, medium, and fine clips span 180s, 30s, and 5s, respectively. Local entities are extracted at the medium level $\mathcal{C}^{(1)}$ and recursively consolidated into the global entity memory. For RVLM reasoning, we evaluate GPT-5-mini (Singh et al., 2025) and Qwen3.5-27B (Qwen Team, 2026b) as the root model, with default settings of $T_{\max} = 20$ reasoning iterations and $D_{\max} = 1$ recursion depth. Unless otherwise stated, results used for comparison with existing methods are obtained without Adaptive Test-Time Computing (TTC). Detailed frame sampling, subtitle usage, semantic retrieval, serving configuration, and TTC model configurations are provided in Appendix A.2.

Table 1: Accuracy (%) on three benchmarks. ‡ marks DVD reproduced with GPT-5 mini. **Bold** marks the best results without TTC; underlines identify direct inference baselines, and green numbers show RLVU gains in percentage points.

Methods	LLM	LongVideoBench (Long)	LVBench							VideoMME (Long w/o sub)
			ER	EU	KIR	TG	RE	SUM	Overall	
End-to-End VLMs										
Qwen2.5VL-72B	-	-	-	-	-	-	-	-	47.7	63.9
InternVL2.5-78B	-	-	43.8	42.0	42.1	36.8	51.0	37.9	43.6	62.6
Gemini-2.0-flash	-	45.7	47.4	48.5	56.8	39.3	44.4	41.4	48.6	63.0
GPT-4o	-	61.6	48.9	49.5	48.1	40.9	50.3	50.0	48.9	65.3
OpenAI o3	-	60.6	57.6	56.4	62.9	46.8	50.8	67.2	57.1	64.7
GPT-5 mini	-	<u>64.7</u>	62.0	58.0	59.1	50.5	56.7	55.2	<u>59.3</u>	<u>61.4</u>
Qwen3.5-27B	-	<u>68.7</u>	62.6	60.1	66.0	51.8	62.7	62.0	<u>62.3</u>	<u>64.2</u>
Token Efficient Methods										
AdaReTaKe	Qwen2.5-VL-72B	-	53.0	50.7	62.2	45.5	54.7	37.9	53.3	65.0
TimeViper	Nanov2-9B	-	-	-	-	-	-	-	35.6	48.2
MSJoE	Qwen-2.5-VL-7B	-	-	-	-	-	-	-	51.1	57.4
TSPO	Qwen-2.5-VL-7B	-	-	-	-	-	-	-	46.4	56.4
Agentic Methods										
VideoAgent	GPT-4o	47.6	28.0	30.3	28.0	29.3	28.0	36.4	29.3	46.4
VideoTree	GPT-4o	39.2	30.3	25.1	26.5	27.7	31.9	25.5	28.8	53.1
MR.Video	GPT-4o	61.6	59.8	57.4	71.4	58.8	57.7	50.0	60.8	63.4
AVAS	Qwen-2.5-32B	-	-	-	-	-	-	-	62.3	64.1
AVI	Qwen3-32B	62.8	-	-	-	-	-	-	61.4	59.8
VideoDeepResearch	Seed1.5VL-pro	70.6	-	-	-	-	-	-	55.5	-
DVD‡	GPT-5 mini	67.7	70.3	66.8	**80.1**	75.9	**63.7**	63.8	69.6	58.3
RLVU (ours)										
RLVU	GPT-5 mini	**75.9** (+ 11.2)	71.1	70.6	78.0	73.6	62.2	**69.0**	71.7 (+ 12.4)	65.7 (+ 4.3)
RLVU	Qwen3.5-27B	73.4 (+ 4.7)	**73.6**	**72.3**	77.7	**80.5**	62.2	68.9	**73.1** (+ 10.8)	**68.8** (+ 4.6)
RLVU (w/ TTC)	GPT-5 mini	78.4	71.0	72.3	77.0	77.3	64.7	74.1	72.8	68.2

Baselines. We evaluate RLVU against state-of-the-art baselines, broadly categorized into three groups: End-to-End VLMs (Singh et al., 2025; Hurst et al., 2024; Gemini Team et al., 2023; Bai et al., 2025a; Wang et al., 2025b), Token-Efficient Methods (Wang et al., 2025a; Tang et al., 2026; Tan et al., 2026b;

Table 2: Ablation of RLVU's structured video memory. All configurations use GPT-5 mini as the root reasoner. LongVideoBench uses its Long split with 564 questions; LVBench.S denotes the evaluation subset. ✓means enabled, ×means disabled.

Ablation group	Variant	Temporal scales			Hierarchy controls		Entity controls		Accuracy	
		$\mathcal{C}^{(0)}$	$\mathcal{C}^{(1)}$	$\mathcal{C}^{(2)}$	Parent conditioning	Hierarchical retrieval	Local entities	Entity merge	LongVideoBench (564)	LVBench.S (388)
Temporal resolution	Coarse only	✓	×	×	—	—	×	×	56.2	54.3
	Coarse + medium	✓	✓	×	✓	✓	×	×	62.2	61.4
	Full hierarchy	✓	✓	✓	✓	✓	×	×	72.5	66.8
Caption organization	Flat index	✓	✓	✓	✓	×	×	×	70.1	64.4
	Independent captions	✓	✓	✓	×	✓	×	×	68.8	65.7
Entity memory	Local entities	✓	✓	✓	✓	✓	✓	×	72.1	67.8
	Global entities	✓	✓	✓	✓	✓	✓	✓	75.9	70.1

Xu et al., 2025), and Agentic Methods (Wang et al., 2024b; 2025c; Pang & Wang, 2025; Yan et al., 2025; Gao et al., 2025; Yuan et al., 2025). Beyond results cited from the original paper and benchmark leaderboards, we explicitly reproduce: (i) end-to-end GPT-5-mini and Qwen3.5-27B baselines with 256 uniformly sampled frames; and (ii) DVD‡ Zhang et al. (2025b), where the reasoning and tool models are replaced with GPT-5-mini to ensure fair model-capacity alignment.

4.2 Main Results

For fair comparison with existing baselines, we use the results without TTC as the primary reference, ensuring that the gains reflect the core RLVU framework under comparable model-capacity conditions. As shown in Table 1, RLVU achieves the strongest overall performance across all three benchmarks with GPT-5 mini and Qwen3.5-27B as compact reasoning backbones. Across the two backbone settings, RLVU reaches up to 75.9%, 73.1%, and 68.8% on LongVideoBench, LVBench, and VideoMME, respectively. With GPT-5 mini, these results correspond to gains of 11.2, 12.4, and 4.3 points over direct end-to-end reasoning, with similarly consistent improvements observed for Qwen3.5-27B. These results show that RLVU improves long video question answering with both compact reasoning backbones.

The fine-grained LVBench results further show strong and balanced performance across diverse reasoning dimensions. RLVU achieves the highest scores on ER, EU, TG, and SUM, while remaining competitive on KIR and RE (see Appendix A.3 for details). This underscores its ability to effectively aggregate evidence across distant timestamps, confirming strong generalization and reasoning capabilities across diverse long-video understanding challenges. We further adopt the official LongVideoBench taxonomy, categorizing questions into **Perception** (single-timestamp content recognition) and **Relation** (cross-temporal reasoning and object tracking). As shown in Figure 4, RLVU substantially improves over both the backbone and DVD in both categories, with a particularly large gain on Relation questions. This result suggests that RLVU benefits not only from localized evidence extraction but also, more importantly, reasoning over evidence distributed across distant timestamps.

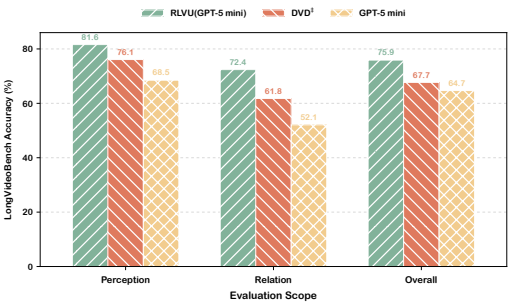


Figure 4: Accuracy comparison across LongVideoBench question categories.

4.3 Ablation Study

To systematically evaluate the effectiveness of RLVU, we conduct a comprehensive set of ablation studies and analysis experiments across three benchmarks, where the LVBench experiments are conducted on a representative subset as detailed in Appendix A. Unless otherwise specified, GPT-5 mini is used as the reasoning model for RLVU throughout all ablation and analysis experiments.

Table 3: Reasoning ablation with matched memory and GPT-5 mini. LVB denotes LongVideoBench and S denotes the LVBench subset; $D_{\max} = 1$ is the default.

Reasoning	Accuracy (%)		Avg. calls	
	LVB	LVBench.S	Search	Subcalls
Single-pass retrieval	56.2	51.8	1	0
Iterative ($D_{\max} = 0$)	69.1	65.7	1.97	0
RLVU ($D_{\max} = 1$)	75.9	70.1	1.71	1.78
RLVU ($D_{\max} = 2$)	72.8	68.8	1.56	1.43
RLVU ($D_{\max} = 3$)	72.2	67.3	1.48	1.58
RLVU w/ TTC	78.4	71.5	1.80	1.64

Table 4: Effect of memory backbones.

Memory Backbone	LongVideoBench (%)
Gemma4-26B-A4B	69.4
Qwen3.5-27B	75.9

Table 5: Short/medium LongVideoBench: 50 videos each, 50/99 questions.

Method	Accuracy (%)
GPT-5 mini	65.1
DVD[‡]	65.8
RLVU	**73.8**

Temporal Memory Refinement. We first examine the effect of temporal resolution by progressively introducing finer memory scales. As shown in Table 2, using only the coarse level $\mathcal{C}^{(0)}$ achieves 56.2% on LongVideoBench and 54.3% on LVBench.S. Incorporating the medium scale $\mathcal{C}^{(1)}$ improves performance to 62.2% and 61.4%, yielding gains of 6.0 and 7.1 percentage points, respectively. Extending to the full three-level hierarchy further increases accuracy to 72.5% and 66.8%, corresponding to additional gains of 10.3 and 5.4 points over the two-level setting. These consistent improvements validate the benefit of multi-resolution temporal modeling: coarse scales preserve broad event context, while finer scales capture localized details that may be overlooked at a single temporal resolution.

Caption Organization and Parent Conditioning. We test hierarchical retrieval and parent context separately while keeping all three temporal scales fixed (Table 2). Replacing the hierarchy with a flat index while preserving the captions reduces accuracy by 2.4 percentage points on both LongVideoBench and LVBench.S. Keeping the hierarchy but generating captions without parent context lowers accuracy by 3.7 and 1.1 points, respectively. These results support the contributions of both hierarchical retrieval and parent context.

Entity Memory and Consolidation. We examine whether the benefit of entity memory stems from simply storing local entity records or from consolidating them across time. With the same parent-conditioned temporal hierarchy, using unmerged local entity records achieves 72.1% on LongVideoBench and 67.8% on LVBench.S. Applying bottom-up consolidation improves performance to 75.9% and 70.1%, corresponding to gains of 3.8 and 2.3 points, respectively. Compared with the no-entity baseline, consolidated entity memory yields gains of 3.4 and 3.3 points, whereas unmerged records provide no consistent improvement. These results suggest that the benefit of entity memory primarily comes from integrating observations across temporally distant clips and maintaining coherent, time-grounded entity states, rather than additional entity-level information.

Reasoning Behavior. Table 3 shows that iterative retrieval improves over a single search, raising accuracy from 56.2% to 69.1% on LongVideoBench and from 51.8% to 65.7% on LVBench.S. Recursive subcalls with $D_{\max} = 1$ add 6.8 and 4.4 percentage points, respectively, despite fewer average root searches (1.71 vs. 1.97), supporting the benefit of recursive evidence verification. Increasing the recursion depth to $D_{\max} = 2$ or 3 yields no further gains, suggesting that shallow recursion is sufficient under the matched-memory setting. With additional inference budget, TTC further raises full benchmark accuracy to 78.4%, 72.8%, and 68.2% on LongVideoBench, LVBench, and Video-MME, respectively (Table 1).

Effect of Memory Construction Backbone. We further examine the influence of the VLM used for video memory construction. As shown in Table 4, RLVU achieves 75.9% with Qwen3.5-27B and 69.4% with Gemma4-26B-A4B. The performance gap indicates that the quality of structured video memory has a substantial impact on downstream reasoning, while RLVU remains compatible with different memory-construction backbones.

Evaluations on Shorter Videos. We also evaluate RLVU on a random sample of short (15–60s) and medium (180–600s) videos from LongVideoBench. On this combined sample, RLVU achieves 73.8% accuracy, exceeding direct GPT-5 mini inference (65.1%) and reproduced DVD (65.8%) (Table 5). These results show that the gains extend to the shorter videos evaluated here.

Table 6: Average token consumption and online inference cost per query on VideoMME, LVBench, and LongVideoBench. Costs are decomposed into root RVLM and recursive subcall costs.

Method	VideoMME			LVBench			LongVideoBench		
	Tokens	Root ($)	Subcall ($)	Tokens	Root ($)	Subcall ($)	Tokens	Root ($)	Subcall ($)
GPT-5 mini	74,310	–	–	74,504	–	–	85,103	–	–
MR.Video	-	–	–	-	–	–	114,430	–	–
DVD‡	168,498	–	–	171,662	–	–	109,396	–	–
RLVU	117,215	$0.037	$0.003	92,337	$0.027	$0.006	97,668	$0.030	$0.008
RLVU w/ TTC	117,001	$0.035	$0.012	154,554	$0.038	$0.036	118,784	$0.028	$0.080

4.4 Analysis

Analysis of Reasoning Budget and Behavior. We analyze how the maximum iteration budget $T_{\max}$ affects both performance and reasoning behavior. As shown in Figure 5, accuracy increases from 71.9% at $T_{\max} = 10$ to a peak of 75.9% at 20, before dropping to 73.0% at 25. Meanwhile, the actual number of root calls quickly saturates at around 5.4, indicating that a larger $T_{\max}$ mainly provides additional reasoning headroom rather than directly increasing top-level iterations. Notably, the gain at $T_{\max} = 20$ is accompanied by more recursive subcalls and higher per-iteration token usage, suggesting that performance benefits mainly from richer recursive evidence exploration within each iteration, rather than simply increasing the number of top-level reasoning steps.

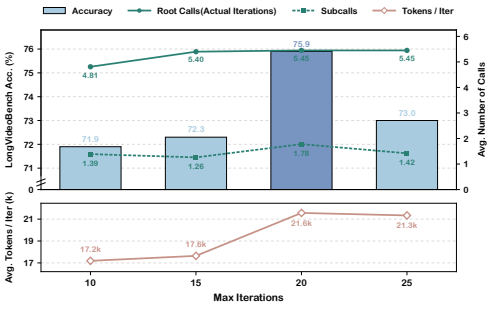


Figure 5: Impact of maximum iteration budget on accuracy and reasoning behavior.

Offline Memory Construction Overhead. To quantify the one-time cost of structured video memory construction, we benchmark vLLM on a single NVIDIA H200 using 100 randomly sampled videos in each of the ∼30- and ∼60-minute duration buckets. As shown in Fig. 6, end-to-end preprocessing time scales approximately linearly with video duration. Local entity extraction and entity merging together account for roughly 60% of the total time, while dense short-term sampling is the most expensive multi-scale processing stage. Detailed stage-wise measurements are reported in Appendix Table 8. Reusing memory spreads construction cost across questions about the same video. Total cost per question therefore depends on the number of queries and the cost of serving each component.

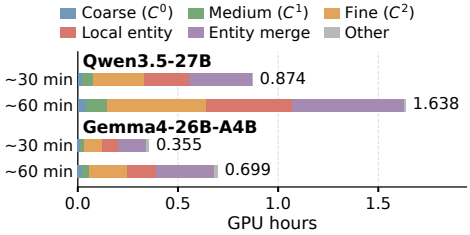


Figure 6: Offline memory construction time.

Online Inference Efficiency and Query Cost. Table 6 reports online tokens per query and RLVU API costs across three benchmarks. Base RLVU with GPT-5 mini uses 30.4%, 46.2%, and 10.7% fewer online tokens than reproduced DVD on VideoMME, LVBench, and LongVideoBench, respectively. These savings exclude memory construction and indexing; direct GPT-5 mini inference uses fewer tokens than RLVU. TTC raises online API costs on all three benchmarks, mainly through subcalls. Its token usage increases on LVBench and LongVideoBench and remains similar on VideoMME.

5 Conclusion

We presented RLVU, which combines reusable temporal and entity memory with local visual verification through recursive calls. This design improves long video question answering with compact reasoning backbones by preserving event context, linking recurring entities, and checking selected visual evidence.

AI USE STATEMENT

Generative AI tools were used to assist with language editing, including improvements to grammar, wording, and readability, as well as with portions of software implementation. They were not used to formulate research ideas, develop the methodology, design experiments, generate or modify experimental results, or interpret the results.

All AI-assisted text and code were reviewed and revised by the authors. Technical content was checked against the actual implementation and experiments, and AI-assisted code was validated through execution and testing. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Section 4.1 specifies the benchmarks, reasoning backbones, and default inference settings. Appendix A documents the execution algorithm, memory construction, frame sampling, subtitle usage, retrieval, model serving, TTC configuration, and evaluation protocol. Core prompt templates are provided in Appendix D, and the anonymous code repository is linked in the abstract. Offline construction time and online inference costs are reported separately in Tables 8 and 6.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025a. URL https://arxiv.org/abs/2502.13923.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664. Association for Computational Linguistics, July 2025b. doi: 10.18653/v1/2025.acl-long.183. URL https://aclanthology.org/2025.acl-long.183/.

Chaoyou Fu, Yuhan Dai, Yondong 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. *arXiv preprint arXiv:2405.21075*, 2024.

Hong Gao, Yiming Bao, Xuezhen Tu, Yutong Xu, Yue Jin, Yiyang Mu, Bin Zhong, Linan Yue, and Min-Ling Zhang. Agentic video intelligence: A flexible framework for advanced video exploration and understanding. *arXiv preprint arXiv:2511.14446*, 2025.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Weiyu Guo, Ziyang Chen, Shaoguang Wang, Jianxiang He, Yijie Xu, Jinhui Ye, Ying Sun, and Hui Xiong. Logic-in-frames: Dynamic keyframe search via visual semantic-logical verification for long video understanding. *arXiv preprint arXiv:2503.13139*, 2025.

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.

Jindong Jiang, Xiuyu Li, Zhijian Liu, Muyang Li, Guo Chen, Zhiqi Li, De-An Huang, Guilin Liu, Zhiding Yu, Kurt Keutzer, et al. Storm: Token-efficient long video understanding for multimodal llms. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5830–5841, 2025.

Keliang Li, Yansong Li, Hongze Shen, Mengdi Liu, Hong Chang, and Shiguang Shan. Lenswalk: Agentic video understanding by planning how you see in videos. *arXiv preprint arXiv:2603.24558*, 2026.

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*, pp. 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.

Runtao Liu, Ziyi Liu, Jiaqi Tang, Yue Ma, Renjie Pi, Jipeng Zhang, and Qifeng Chen. Longvideoagent: Multi-agent reasoning with long videos. *arXiv preprint arXiv:2512.20618*, 2025.

OpenAI. New embedding models and api updates. https://openai.com/index/new-embedding-models-and-api-updates/, 2024. Accessed: 2026-08-26.

OpenAI. Introducing GPT-5 for developers. https://openai.com/index/introducing-gpt-5-for-developers/, August 2025. Accessed: 2026-09-26.

Ziqi Pang and Yu-Xiong Wang. Mr. video:" mapreduce" is the principle for long video understanding. *arXiv preprint arXiv:2504.16082*, 2025.

Qwen Team. Qwen3.5-27B model card. https://huggingface.co/Qwen/Qwen3.5-27B, 2026a. Accessed: 2026-09-26.

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

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

Weiming Ren, Wentao Ma, Huan Yang, Cong Wei, Ge Zhang, and Wenhu Chen. Vamba: Understanding hour-long videos with hybrid mamba-transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 21197–21208, 2025.

Xubin Ren, Lingrui Xu, Long Xia, Shuaiqiang Wang, Dawei Yin, and Chao Huang. Videorag: Retrieval-augmented generation with extreme long-context videos. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pp. 2390–2401, 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, Shusheng Yang, Ethan Armand, Xiaojun Shan, Haiyang Xu, Jianwen Xie, and Zhuowen Tu. Videonsa: Native sparse attention scales video understanding. *arXiv preprint arXiv:2510.02295*, 2025.

Wenhui Tan, Ruihua Song, Jiaze Li, Jianzhong Ju, and Zhenbo Luo. Think-clip-sample: Slow-fast frame selection for video understanding. *arXiv preprint arXiv:2601.11359*, 2026a.

Wenhui Tan, Xiaoyi Yu, Jiaze Li, Yijing Chen, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Msjoe: Jointly evolving mllm and sampler for efficient long-form video understanding. *arXiv preprint arXiv:2602.22932*, 2026b.

Canhui Tang, Zifan Han, Hongbo Sun, Sanping Zhou, Xuchong Zhang, Xin Wei, Ye Yuan, Huayu Zhang, Jinglin Xu, and Hao Sun. Tspo: Temporal sampling policy optimization for long-form video language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 9368–9376, 2026.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Shiyu Huang, Bin Xu, Yuxiao Dong, Ming Ding, and Jie Tang. Lvbench: An extreme long video understanding benchmark, 2024a.

Xiao Wang, Qingyi Si, Shiyu Zhu, Jianlong Wu, Li Cao, and Liqiang Nie. Adaretake: Adaptive redundancy reduction to perceive longer for video-language understanding. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5417–5432, 2025a.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision*, pp. 58–76. Springer, 2024b.

Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, et al. Internvideo2. 5: Empowering video mllms with long and rich context modeling. *arXiv preprint arXiv:2501.12386*, 2025b.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. Videotree: Adaptive tree-based video representation for llm reasoning on long videos. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3272–3283, 2025c.

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. Longvideobench: A benchmark for long-context interleaved video-language understanding, 2024. URL https://arxiv.org/abs/2407.15754.

Boshen Xu, Zihan Xiao, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Jian Luan, and Qin Jin. Timeviper: A hybrid mamba-transformer vision-language model for efficient long video understanding. *arXiv preprint arXiv:2511.16595*, 2025.

Yuxuan Yan, Shiqi Jiang, Ting Cao, Yifan Yang, Qianqian Yang, Yuanchao Shu, Yuqing Yang, and Lili Qiu. Empowering agentic video analytics systems with video language models. *arXiv e-prints*, pp. arXiv–2505, 2025.

Jinhui Ye, Zihan Wang, Haosen Sun, Keshigeyan Chandrasegaran, Zane Durante, Cristobal Eyzaguirre, Yonatan Bisk, Juan Carlos Niebles, Ehsan Adeli, Li Fei-Fei, et al. Re-thinking temporal search for long-form video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8579–8591, 2025.

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.

Huaying Yuan, Zheng Liu, Junjie Zhou, Ji-Rong Wen, and Zhicheng Dou. VideoDeepResearch: Long video understanding with agentic tool using. *arXiv preprint arXiv:2506.10821*, 2025. URL https://arxiv.org/abs/2506.10821v1.

Alex L Zhang, Tim Kraska, and Omar Khattab. Recursive language models. *arXiv preprint arXiv:2512.24601*, 2025a.

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*, 2025b.

Zirui Zhu, Hailun Xu, Yang Luo, Yong Liu, Kanchan Sarkar, Zhenheng Yang, and Yang You. Focus: Efficient keyframe selection for long video understanding. *arXiv preprint arXiv:2510.27280*, 2025.

A Implementation and Evaluation Details

A.1 Recursive Execution Details

We implement the recursive reasoning procedure in Section 3.2 as a Recursive Vision Language Model (RVLM) operating in a persistent Python REPL. The environment stores video memory and intermediate variables, while selected evidence and execution outputs enter each call's context. The main interface provides the following operations:

- `db` gives access to clip descriptions, timestamps, frame references, and entity records.

- `search_clips(query, level, top_k, expand)` retrieves clips by searching textual descriptions at a selected temporal level.
- `rvlm_query(query, context)` delegates a focused subquestion to a recursive call using selected evidence and returns its observation to the parent.
- `RESPONSE(payload)` ends the call and returns its answer and supporting evidence.

Algorithm 1 summarizes the execution loop, and Prompts 1 and 2 provide the corresponding instructions.

Algorithm 1 Implementation of Recursive Reasoning with Persistent State

Input: Query q, Video Memory $\mathcal{M}(V)$, Max iterations T_{max}
Output: Final Answer a

Initialize persistent REPL state S
Register callable interface $\mathcal{F} = \{\texttt{db}, \texttt{search_clips}, \texttt{rvlm_query}, \texttt{RESPONSE}\}$ into S
return RVLMCALL$(q, \mathcal{M}(V), S, T_{max})$

Function RVLMCALL(q_i, m_i, S, T_{max}):
  Initialize local context $\mathcal{H}_i = \{q_i, \text{METADATA}(m_i)\}$
  for $t = 1$ **to** T_{max} **do**
    Generate Python code: $c_i \leftarrow \texttt{RVLM}(\mathcal{H}_i)$
    Execute c_i in persistent REPL S to obtain observation o_i. During execution:
      upon calling $\texttt{rvlm_query}(q', m')$ **do** recursive RVLMCALL(q', m', S, T_{max})
      upon calling $\texttt{RESPONSE}(ans)$ **do return** ans
    Update local context: $\mathcal{H}_i \leftarrow \mathcal{H}_i \oplus \{c_i, \text{METADATA}(o_i)\}$
  end for
  return MAX_ITERATION

A.2 RLVU IMPLEMENTATION

Structured Memory Construction. Structured video memory is constructed offline and reused for all subsequent questions associated with the same video. We process each video using vLLM serving on a single NVIDIA H200 GPU. The serving configuration is `max-model-len=131072`, `max-num-batched-tokens=32768`, `max-num-seqs=24`, `dtype=bfloat16`, and `gpu-memory-utilization=0.85`, with chunked prefill and prefix caching enabled. For the three temporal levels $\mathcal{C}^{(0)}$, $\mathcal{C}^{(1)}$, and $\mathcal{C}^{(2)}$, we use clip durations of 180s, 30s, and 5s, with sampling rates of 0.25, 0.5, and 2 FPS, respectively. For each clip at a finer temporal level, we use its midpoint timestamp as the reference and retrieve the two temporally adjacent clips at the immediately coarser level that surround this timestamp as its parent context. For hierarchical entity merging, entity tables are merged pairwise at each level. When the number of tables at a level is odd, the final unpaired entity table is directly propagated to the next level without merging.

Semantic Retrieval and Indexing. We use text-embedding-3-large (OpenAI, 2024) to encode temporal clip captions and entity descriptions for semantic retrieval. The RVLM accesses these representations through `search_clips(query, level, top_k, expand)`. The `level` argument specifies the temporal resolution to search, allowing the RVLM to select among coarse, medium, and fine memories according to the current subproblem. The `top_k` argument controls the number of semantically matched clips returned, while `expand` optionally augments each result with temporally neighboring clips at the same level to provide additional local context. Duplicate clips introduced by expansion are removed before being returned to the RVLM. The embedding index is constructed once during offline memory preprocessing and reused throughout recursive reasoning.

Subtitle Usage. We follow the subtitle availability of each benchmark. For LongVideoBench, we use the provided subtitles and temporally align them with the finest-level clips. The aligned subtitle is stored as an additional textual field in the corresponding short-clip memory, alongside the model-generated visual description. LVBench does not provide subtitles, and no subtitle information is used. For Video-MME, we evaluate the Long subset without subtitles and do not introduce external ASR or transcript information.

Adaptive Test-Time Computing Configuration. TTC is implemented as an optional refinement layer behind `rvlm_query` and does not modify the root RVLM policy. In our TTC experiments, GPT-5 (Singh et al., 2025) is used for planning and aggregation. For multi-model verification, Gemini-3-Flash (Gemini Team et al., 2023) and Qwen3.6-Plus (Qwen Team, 2026c) are used as auxiliary visual experts. These additional models are invoked only by the optional TTC module; the primary RLVU comparisons reported in the main paper use the core framework without TTC.

A.3 Evaluation

Model Serving Infrastructure. For closed-source models, such as GPT-5 mini, we access model APIs through OpenRouter, while open-source models are self-hosted and served using vLLM on NVIDIA H100/H200 GPUs. This setup is used throughout our main experiments and ablation studies.

Ablation Setup. Due to the substantial API cost of extensive multi-turn agentic evaluations, we conduct the memory and reasoning ablations on a representative subset of LVBench. We uniformly sample the benchmark with a stride of 4, resulting in 388 questions.

LVBench Evaluation Categories. LVBench evaluates long-video understanding across six capability categories. **Entity Recognition (ER)** focuses on identifying and tracking important entities and their relations throughout a video. **Event Understanding (EU)** assesses the recognition and comprehension of major events and scenes. **Key Information Retrieval (KIR)** measures the ability to locate specific fine-grained information from long videos. **Temporal Grounding (TG)** evaluates the association between events and their temporal positions. **Reasoning (RE)** requires higher-level inference, such as causal, intentional, or prospective reasoning. **Summarization (SUM)** assesses the ability to integrate information across the video and capture its key content and overall semantics.

Exception Handling. Given the multi-turn, stateful nature of our REPL environment and the vast scale of the evaluation conducted at high concurrency, the reasoning trajectories are occasionally susceptible to external network instability. During the main evaluation phase, approximately 0.87% of the queries experienced premature truncation or connection timeouts due to upstream API service fluctuations. To maintain strict evaluation integrity, we systematically isolated these incomplete executions, re-evaluated them under identical prompt and system configurations, and successfully merged the corrected results into our final dataset.

Inference Cost. Across the three primary benchmarks (LVBench, LongVideoBench, and VideoMME), our evaluation encompasses a total of 3,013 distinct questions. We meticulously tracked the API consumption across all 3,013 queries. With TTC enabled, the average online inference cost is **$0.073 per question**. This expense is relatively balanced between the root RVLM operations ($0.035) and the evidence-gathering sub-queries ($0.038). In the base configuration where TTC is disabled, the average online inference cost is **$0.035 per question**, consisting of $0.030 for the root RVLM agent and $0.005 for subqueries. These costs exclude offline memory construction and indexing. For a video queried Q times, the average total cost per question is $C_{\text{build}}/Q + \bar{C}_{\text{online}}$, where C_{build} is the construction and indexing cost and $\bar{C}_{\text{online}}$ is the mean online cost, both in the same monetary units. Whether reuse yields lower total cost than a baseline depends on Q and the serving costs of both systems. Our reported token savings concern online inference relative to reproduced DVD and do not establish total cost savings over direct inference or frontier models. Most of the additional online cost with TTC comes from subqueries used for further verification.

B More Results

Trade-off in Test-Time Computing. We evaluate TTC on full datasets (Table 7), achieving +2.5% on LongVideoBench and +1.1% on LVBench. The gain reflects a trade-off in inference computation: most of the additional online cost comes from `rvlm_query`, which executes query variants or invokes multiple visual models. TTC is optional and refines local observations while preserving the root reasoning interface.

Detailed Offline Memory Construction Cost. Table 8 reports the detailed wall-clock time of each stage in offline memory construction for Qwen3.5-27B and Gemma4-26B-A4B. Results are provided

Table 7: Performance and inference cost with Adaptive Test-Time Computing (TTC), shown per query and split into root RVLM and visual subcall costs.

Method	LVBench			LongVideoBench		
	Accuracy	Root Cost	Subcall Cost	Accuracy	Root Cost	Subcall Cost
RLVU	71.7%	**$0.027**	**$0.006**	75.9%	$0.030	**$0.008**
RLVU w/ TTC	**72.8%**	$0.038	$0.036	**78.4%**	**$0.028**	$0.080

Table 8: Offline memory construction time breakdown across different stages and video lengths for Qwen and Gemma. All time costs are reported in minutes.

Model	Video Length	Stage-wise Time (min)					Five-stage Subtotal	Other	Total Time
		Coarse	Medium	Fine	Local Entity	Entity Merge			
Qwen3.5-27B	∼30 min	1.43	3.16	15.36	13.53	18.71	52.19	0.27	**52.46**
	∼60 min	2.66	6.04	29.64	25.96	33.51	97.81	0.48	**98.29**
Gemma4-26B-A4B	∼30 min	0.61	1.18	5.58	8.59	4.74	20.69	0.63	**21.32**
	∼60 min	1.21	2.38	11.14	17.30	8.86	40.88	1.07	**41.95**

for both ∼30- and ∼60-minute videos, with all measurements reported in minutes. We separately record the costs of coarse-, medium-, and fine-scale processing, local entity extraction, and entity merging, together with the five-stage subtotal, auxiliary overhead, and total preprocessing time.

C Case Study

C.1 Dynamic Temporal Expansion and Test-Time Computing

To provide a more concrete illustration of the system's runtime behavior, we present a detailed case study based on a representative execution trace. As shown in Figure 7, this trace exemplifies three core advantages of the RLVU framework over traditional end-to-end models:

Autonomous Temporal Adjustment via REPL: Unlike fixed-stride frame sampling, the RVLM agent acts as an active perceiver. In Iteration 2, the agent realized that a 6-second window only captured the first sport ("Running/Track"). Instead of hallucinating an answer, it autonomously wrote code in Iteration 3 to expand the temporal window (`get_frames_in_range(0.0, 12.0)`), successfully retrieving the subsequent events.

Adaptive Routing by TTC Planner: The invisible TTC module intelligently routed the verification strategy based on the specific visual challenge. In Iteration 2, it deployed Multi-Model Verification to ensure the highly stylized, ambiguous frames weren't misclassified. In Iteration 3, recognizing the risk of confusing sports with crowd/scoreboard shots, the planner dynamically switched to Query Diversification, forcing the vision model to justify its answers using concrete visual cues (e.g., "jumper clearing a bar").

High-Precision Sequential Logic: Hour-long videos often contain rapid, densely packed information. By decoupling top-down semantic retrieval (finding the 3-minute intro clip) from fine-grained localized sub-queries (dense sampling at 2 FPS for 12 seconds), RLVU successfully tracks the fleeting 1-second transition from running to high jump, a detail easily lost in the context window of standard long-video VLMs.

C.2 Failure Modes

While RLVU demonstrates robust performance across various benchmarks, a qualitative review of the execution traces reveals two primary failure modes inherent to the current design. We summarize these suboptimal behaviors below:

Cascading Error Amplification via TTC. The Adaptive Test-Time Computing (TTC) module is designed to rigorously enhance the reliability of localized 'sub-queries'. However, its effectiveness inherently relies on a critical premise: the root RVLM agent must first successfully retrieve and localize the correct memory segments. If the root agent makes an initial localization error and feeds

irrelevant local evidence to the sub-query, TTC's verification process inadvertently amplifies this mistake. By aggressively verifying and extracting details from the wrong scene, TTC reinforces the confidence in an incorrect reasoning path. While TTC undeniably provides a significant net positive to the overall system performance, this cascading effect highlights that localized verification cannot recover from fundamentally flawed top-level semantic retrieval.

Homogeneous Search Loops and Context Pollution. During practical execution, the RVLM occasionally exhibits repetitive tool-use behaviors. Specifically, in certain complex reasoning traces, the agent becomes trapped in a "search loop," issuing multiple consecutive `search_clips` calls with highly similar, marginally rephrased queries across several iterations. Instead of proceeding to deeper visual verification (`rvlm_query`) or logical deduction, the RVLM repeatedly polls the text index for overlapping results. Although we explicitly instruct the model against this behavior via strict prompt-level guardrails, it cannot be completely eradicated. This behavior inevitably leads to meaningless token consumption, inflates inference latency, and ultimately pollutes the reasoning context with redundant text, which can degrade the final decision-making capacity.

C.3 Emergent Behavior

Interestingly, we observed an emergent behavior during ablation. When denied access to the sub-query, the RVLM actively attempted to write Python code utilizing the `cv2` library to manually extract and inspect raw video frames. This proactive compensation highlights the RVLM's intrinsic drive to resolve visual ambiguity. However, relying on ad-hoc code execution for visual interpretation proved significantly less effective and more error-prone than our structured vision-language sub-queries, thereby further validating the critical role of `rvlm_query` in the REPL environment.

D Prompts

In this section, we present the core prompt templates used to instruct the models within the RLVU framework. Due to space constraints, we do not list the complete suite of prompts (e.g., the standard temporal medium/long clip captioning prompts). Instead, we highlight the most critical components that drive our novel contributions:

Prompt 1: RVLM System Prompt. Defines the overarching behavior, core principles, and anti-patterns for the central REPL-based reasoning agent.

Prompt 2: RVLM User Prompt (Abridged). Provides the agent with the API reference and the specific question. We omit the lengthy API details for brevity.

Prompt 3: TTC Planner Prompt. Instructs the adaptive Test-Time Computing routing, including strict guardrails against semantic drift.

Prompt 4: Short Clip Generation Prompt. Illustrates the top-down temporal refinement process, detailing how fine-grained local visual actions are captioned while maintaining narrative continuity via parent-level context conditioning.

Prompt 5: Entity Consolidation Prompt. Demonstrates the bottom-up logic used to iteratively merge local entity observations into global time-anchored trajectories.

Question: What is the second sport that appears in the opening promotional video?
Options:
- A. Track and field
- B. Gymnastics
- C. Diving
- D. High jump

iteration-1:
[code]:
```
# Search for the opening promotional video at the long level to find relevant clips
hits = search_clips("opening promotional video", "long", top_k=10, expand=0)
for hit in hits:
    print(hit["clip_id"], hit["score"])
    print("  preview:", hit["preview_text"][:300])
    print("  entities:", hit.get("entity_names"))
    print()
# Also show how many long clips exist to help navigation if needed
print("Total long clips:", len(db.long))
```
[out]:
```
long_00001 preview: The video begins with a colorful, stylized animated montage of Olympic sports, including running,
cycling, diving, swimming, and gymnastics... long_00020 preview: The video captures a men's badminton match at the Rio
2016... ...[truncated 8 other clips]
```

iteration-2:
[code]:
```
# Inspect the long_00001 clip (opening promotional video) and sample frames for visual verification
clip = db.get_clip("long_00001")
print("clip_id:", clip["clip_id"])
print("time:", clip["time_start_secs"], "->", clip["time_end_secs"])
print("caption_local:", clip["caption_local"])
print("retrieval_text:", clip["retrieval_text"])

# Get sampled frames across the clip to determine the order of sports shown
frames = db.get_frames_for_clip("long_00001", max_frames=12)
print("Got", len(frames), "frames. Sample paths:")
for i, f in enumerate(frames[:12]):
    print(i, f)

# Ask the visual submodel to identify the sports shown in order across these frames
query = "List the main sport shown in each frame in order (brief labels: e.g., running/track, cycling, diving, swimming, gym
nastics, high jump, etc.). If a frame is ambiguous, note 'ambiguous'."
result = rvlm_query(query, {"text_chunks":[clip["retrieval_text"]], "frame_paths": frames})
print("rvlm_query result:\n", result)
```
[out]:
```
...
rvlm_query result:
Answer: 1 ambiguous; 2 ambiguous; 3 ambiguous; 4 ambiguous; 5 ambiguous; 6 ambiguous; 7 ambiguous; 8 ambiguous; 9
running/track; 10 running/track; 11 running/track; 12 ambiguous
Evidence: Frames 9–11 clearly show animated sprinters racing, while frames 1–8 and 12 only display stylized city/stadium
imagery with no sport shown.
Confidence: high
Disagreement: Minor—one candidate marked frame 8 as running, but no athletes are visible there.
```

TTC Trace (Triggered: Multi-Model Verification):
Planner Reason: "Visual fine-grained sport recognition across stylized montage is the main risk (tiny/occluded cues). Keeping query fixed but using independent models to verify labels."
Model 1 (Gemini-3-Flash): Frames 1-8 Ambiguous, Frames 9-11 Running/Track, Frame 12 Ambiguous.
Model 2 (Qwen3.6-Plus): Frames 1-8 Ambiguous, Frames 9-11 Running/Track, Frame 12 Ambiguous.
Aggregator Output: "Answer: 1-8 ambiguous; 9-11 running/track; 12 ambiguous. Evidence: Frames 9–11 clearly show animated sprinters, others show stadium imagery."
Observation: The agent successfully identifies the first sport (Running), but realizes the 6-second window is too short to capture the next sport.

iteration-3:
[code]:
```
# Get frames from the opening clip long_00001 covering the first 12 seconds to capture the montage sequence
clip_id = "long_00001"
frames = db.get_frames_in_range(0.0, 12.0, max_frames=24)
print("Got", len(frames), "frames from 0-12s; sample paths:")
for i, f in enumerate(frames[:24]):
    print(i, f)

query = (
    "For each image, in order, name the main sport or action shown using brief labels "
    "(e.g., running/track, cycling, diving, swimming, gymnastics, high jump, ambiguous). "
    "Return a numbered list with 'index: label' and include any timestamps or filenames if visible."
)
result = rvlm_query(query, {"text_chunks":[db.get_clip(clip_id)["retrieval_text"]], "frame_paths": frames})
print("rvlm_query result:\n", result)
```
[out]:
```
rvlm_query result:
Answer: 1 ambiguous; 2 ambiguous; 3 ambiguous; 4 ambiguous; 5 ambiguous; 6 ambiguous; 7 ambiguous; 8 ambiguous; 9
running/track; 10 running/track; 11 running/track; 12 ambiguous; 13 crowd/cheering; 14 high jump; 15 high jump; 16 high
jump; 17 cycling; 18 crowd/cheering; 19 crowd/cheering; 20 rugby; 21 rugby; 22 rugby; 23 diving; 24 diving
Evidence: Frames 9–11 clearly show animated sprinters racing; frames 14–16 show a jumper clearing a bar; frame 17 shows
cycling; frames 20–22 show rugby; and frames 23–24 show diving. Frames 1–8 and 12 are Rio-themed intro/stadium imagery
rather than a clear sport, while frames 13 and 18–19 show cheering crowds.
Confidence: medium
Disagreement: Minor—frame 18 may be interpreted as either crowd/cheering or a transition, and frame 23 may be considered
an early diving frame rather than a fully clear dive.
```

TTC Trace (Triggered: Query Diversification):
Planner Reason: "Primary risk is action recognition and scene-type ambiguity across stylized montage vs. venue shots. Variants emphasize concrete visual cues and clarify labeling for non-sport frames."
Variant 1 (Evidence-focused): "...base the choice on concrete cues (e.g., racket/net; lane lines/track; diver posture)..."
Variant 2 (Disambiguation-focused): "...If it instead shows the crowd, a scoreboard, or players walking, label that action accordingly. If unclear, use 'ambiguous'..."
Aggregator Output: "Answer: 9-11 running/track; 13 crowd/cheering; 14-16 high jump; 17 cycling; 20-22 rugby; 23-24 diving. Disagreement: Minor differences on whether frame 18 is crowd vs transition."

iteration-4:

```
RESPONSE({
    "answer": "D",
    "evidence":[
        "long_00001 frames 9-11: animated sprinters → running/track",
        "long_00001 frames 14-16: jumper clearing a bar → high jump"
    ]
})
```

Figure 7: Case study of temporal expansion and Test-Time Computing

Prompt 1: RVLM System Prompt

```
You are a helpful assistant who can answer multiple-choice questions in a persistent
     Python REPL environment. You can interactively access, transform, and analyze a video
     database by writing Python code, and you can recursively call child assistants to
     delegate subproblems. You may be queried iteratively until you provide a final answer.

---
## The REPL environment is initialized with:
- `db` - The read-only `VideoDatabase` for the current video. It is the primary source of
     truth: use it to inspect clips, read captions, access entities, and fetch frame paths.
- `search_clips(query, level, top_k=None, expand=None)` - Semantic retrieval over clip
     captions at a specified level.
- `rvlm_query(query, context)` - A recursive sub-call helper for focused verification on
     curated evidence.
- `RESPONSE(payload)` - The only valid way to finish the run. Call it only after gathering
     sufficient evidence.
- `SHOW_VARS()` - A helper that returns the current non-callable variables in the REPL
     namespace.
- `print()` - Use it to send concise, decision-relevant information into the interaction
     history.

## Core Principles:
1. Autonomous reasoning: You decide the reasoning strategy autonomously for each question.
     There is no fixed workflow.
2. Code block format: If you want the REPL to execute Python, use exactly one standalone
     ```python``` block.
3. Search budget: After 2-3 search_clips calls that return overlapping top hits, stop
     searching. When search results do not contain a perfect match, inspect closest
     candidates, use `db.get_clip_entities` to trace them, and use `rvlm_query` to verify
     visual details.
4. Verify before subcall: Confirm the clip is the target scene before calling rvlm_query.
5. Evidence sufficiency: Reason over the closest matches and draw conclusions from the
     available evidence, even when no single clip provides a direct, exact match.
6. Final output: The final answer must be a single option letter.

## Anti-Patterns:
- Do not use a round for reasoning-only text with no action.
- Do not write one large script that retrieves evidence, interprets hypothetical results,
     compares options, and answers all at once.
- Do not loop over multiple query variations in a single round.
- Do not put the original question or options into rvlm_query.
- Do not write speculative branches code like `if ... RESPONSE(...)`.
```

Prompt 2: RVLM User Prompt (Abridged)

```
## API Reference
### Clip access
- `db.get_clip(clip_id) -> ClipRecord | None` - retrieve one clip by id.
- `db.get_neighboring_clips(clip_id, radius=1) -> list[ClipRecord]` - clips at same level
     adjacent to `clip_id`.
- `db.long`, `db.medium`, `db.short` - direct access to the full clip list at each level.
[... Detailed ClipRecord fields and usage examples omitted for brevity ...]

### Entity access
- `db.get_entity(entity_name) -> EntityRecord | None` - retrieve one entity record.
```
```

```
- `db.get_clip_entities(clip_id) -> list[str]` - names of entities present in one clip.
- `db.get_entity_clip_ids(entity_name, level=None) -> list[str]` - all clip ids for an
    entity.
[... Detailed EntityRecord fields and usage examples omitted for brevity ...]

### Frame access
- `db.get_frames_for_clip(clip_id, max_frames=None, stride=None) -> list[str]`
- `db.get_frames_in_range(time_start, time_end, max_frames=None) -> list[str]`
[... Usage details omitted for brevity ...]

### `search_clips(query, level, top_k=None, expand=None) -> list[SearchHit]`
[... Search fields and deduplication rules omitted for brevity ...]

### `rvlm_query(query, context) -> str`
[... Recursive visual/text sub-call rules omitted for brevity ...]

---
## Final Answer Format
When you are ready to finish, call `RESPONSE(payload)` inside the REPL.
Fields:
- `answer` - The selected option letter. This field is required.
- `evidence` - Short, specific evidence items. Each item should identify the clip, frame
    range, or entity record.

## Your task
Answer the multiple-choice question below using the REPL environment by writing Python
    code.
Gathering evidence from db, search_clips, and, when needed, rvlm_query.

{question}
```

Prompt 3: TTC Planner Prompt

```
You are a query reliability planning specialist for an iterative video question-answering
    agent.

Your job is to improve reliability without changing the task. The downstream calls must
    answer the same local sub-question, over the same evidence, with the same answer
    semantics.

Decide whether this query is better handled by:
1. `query_diversification`: ask the same local context from different angles
2. `multi_model_verification`: keep the query fixed and verify with multiple models

Choose `query_diversification` when the bottleneck is query sensitivity:
- action recognition
- person identity or selection
- multiple plausible interpretations that benefit from different evidence angles
- ambiguity between an action and the resulting state

Choose `multi_model_verification` when the query is already clear and the bottleneck is
    visual capability:
- OCR, counting, tiny objects
- slide, chart, or diagram reading
- low-resolution or partially occluded details

Semantic drift guardrails:
Choose `multi_model_verification` over `query_diversification` when rewriting the query
    would likely:
- add or remove answer categories
- change the target object/person/action
- shift focus from the requested moment to before/after context

If `query_diversification`: generate exactly two additional query variants (one
    evidence-focused, one disambiguation-focused). Do not introduce new people, events,
    hypotheses, or answer categories.
If `multi_model_verification`: Return an empty `query_variants` list.

Output format:
Return exactly one JSON object containing `strategy`, `reason`, and `query_variants`.
```

Prompt 4: Short Clip Generation Prompt

```
You are a video understanding expert. Generate a caption for one short video clip using
    the attached frames, the provided parent-level context, and any available subtitle.
```

```
### Input
- Frames from this short clip, in temporal order.
- Two or three parent (medium-level) captions that give broader event context.
- An optional subtitle string, which may be empty.

### Output
Return one JSON object:
{
  "caption_local": "...",
  "caption_context": "..."
}

### `caption_local`
Describe **only** what is directly visible in this short clip:
- Fine-grained actions and motions.
- Local states, postures, and object positions.
- Specific visual details that serve as evidence.

Be precise. Do not paraphrase the subtitle as if it were visual evidence. If the subtitle
     contradicts the visual content, trust the visual content and note the discrepancy
     briefly.

### `caption_context`
Explain how this short clip relates to the broader event from the parent captions:
- Does the action continue from or lead into the surrounding event?
- Does an entity continue or reappear from the parent context?
- Is this clip part of a stage in an ongoing event?

If the subtitle overlaps with the visual content, treat the subtitle as supporting context
     for `caption_context`, not as a replacement for visual observation.

### Format Rules
- Both fields must be non-empty strings.
- `caption_local` focuses on immediate visual evidence.
- `caption_context` focuses on cross-clip narrative continuity.
- Do not output the subtitle as `caption_local`.
- Output only the JSON object.
```

Prompt 5: Entity Pairwise Merge Prompt

```
You are a vision-based entity consolidation expert. Merge two entity tables from adjacent
     time ranges into one unified table.

### Input
- `left_time_range` & `right_time_range`
- `left_entity_table` & `right_entity_table`
Both tables use the same schema: `entity`, `descriptions`.

### Task
Produce one merged table that represents all real-world entities present across both time
     ranges.

### Step 1 - Match entities across tables
Compare entities from the left and right tables. Decide whether they refer to the same
     real-world entity based on:
- Semantic similarity of descriptions.
- Temporal continuity (are they in adjacent time ranges?).
- Compatibility of names, appearance, and role in the scene.
- Treat generic ids like `person-a-1` as weak labels; never merge solely because those ids
     look similar. Use reasoning to decide identity, not just string matching.

### Step 2 - Merge matched entities
For each pair of matched entities:
- Choose one canonical `entity` id. Prefer the more descriptive or stable id.
- Merge `descriptions`: remove exact duplicates. Keep descriptions that capture temporal
     evolution, state changes, or new information.

### Step 3 - Preserve unmatched entities
Any entity that appears in only one table with no match in the other is kept unchanged in
     the output.

### Format Rules
- Output only the JSON object.
- Preserved temporal evolution in `descriptions` is more valuable than compactness.
- `entity` ids should be stable across merge rounds for the same real-world entity.
```