VideoPro: Adaptive Program Reasoning for Long Video Understanding

Anonymous ACL submission

Abstract

Understanding long videos remains challenging due to the sparsity of visual evidence relevant to a given query. Prior work has explored program-based visual grounding, typically relying on executable programs generated by auxiliary large language models. However, when scaling to long videos, existing approaches face several critical limitations: (1) frame-centric vision modules are often insufficient for long video processing; (2) naively applying program-based reasoning to all queries incurs considerable computational overhead; and (3) errors arising from low-confidence predictions and imperfect program execution are difficult to recover from. To address these challenges, we propose VideoPro, a unified framework that enables VideoLLMs to adaptively reason over long videos and refine their predictions through executable programs. VideoPro first performs adaptive reasoning, dynamically determining whether a query can be resolved directly by the native VideoLLM or requires explicit multi-step program reasoning. For complex queries, the model decomposes the task into executable programs that invoke specialized vision modules for precise temporal and semantic grounding. To further improve robustness, VideoPro incorporates a self-refinement mechanism that leverages execution feedback and confidence signals to correct erroneous executions and refine low-confidence reasoning programs. By tightly integrating adaptive reasoning with self-refinement, VideoPro consistently outperforms prior methods across multiple long-video understanding benchmarks, yielding an average 6.7% improvement for Qwen3-VL-8B.

1 Introduction

Long-video understanding is fundamentally challenged by the sparsity and long-range dispersion of query-relevant visual evidence. Effective reasoning, therefore, requires precise temporal grounding and multi-step integration across distant segments, rendering dense frame-level processing computationally intractable. Program-based visual grounding has been proposed to support explicit multi-step reasoning by executing LLM-generated programs over modular vision components (Gupta and Kembhavi, 2023; Surís et al., 2023; Mahmood et al., 2024; Gao et al., 2024) (Figure 1). However, when scaled to long videos, existing methods exhibit three critical limitations. (1) Frame-centric vision modules are insufficient for long-range temporal and semantic grounding. (2) Uniform program invocation is inefficient: many queries can be answered directly by native VideoLLMs, which already achieve nearly 90% accuracy in high-confidence cases without program execution (Figure 2). (3) Error recovery is fragile: low-confidence predictions and imperfect executions can cascade under static, one-shot program pipelines, with limited mechanisms for correction.

To address these challenges, we propose VideoPro, a unified framework that synergizes adaptive reasoning with refinement for efficient long-form video understanding. VideoPro functions as a dynamic planner, determining on-the-fly whether a query can be resolved by the native VideoLLMs or requires multi-step visual programming. For complex queries, it decomposes the task into structured sub-tasks and selectively invokes vision modules to retrieve, localize, and inspect relevant information. To support the latter, VideoPro integrates a library of general video modules, including multimodal retrieval, temporal localization, and fine-grained visual extraction. To further improve robustness, VideoPro incorporates a self-refinement mechanism, which revises failed executions and low-confidence reasoning programs, ensuring robust reasoning against static programs.

To instantiate this framework, we leverage advanced LLMs to synthesize high-quality visual programs and use the native VideoLLM's initial pre-

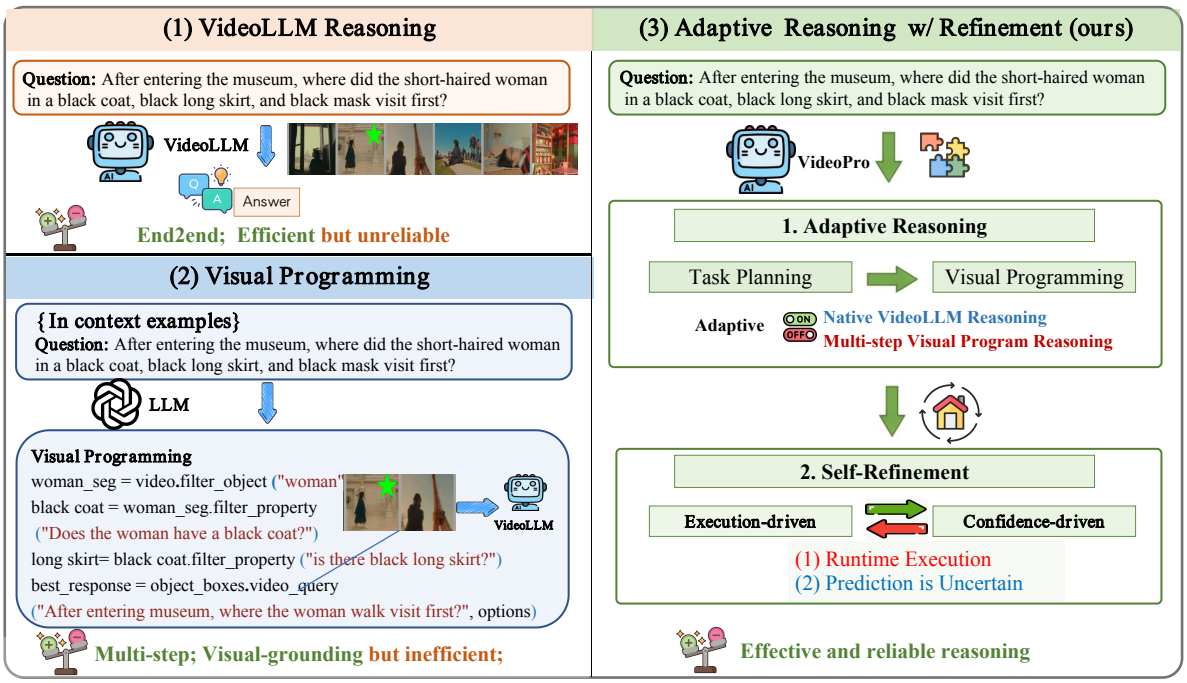

Figure 1: Comparison of prior methods and VideoPro: effective and reliable adaptive reasoning with refinement.

diction to decide whether a query can be solved directly or should be routed to explicit visual program reasoning. We categorize execution programs into failures, successful executions with incorrect predictions, and correct predictions to construct a *reason-and-refine* supervision dataset. We then train a unified VideoLLM to internalize the ability of adaptive reasoning and self-refinement, further optimized using Group Relative Policy Optimization (GRPO). Extensive experiments demonstrate that VideoPro achieves superior stability and performance on long-form video benchmarks, surpassing GPT-4o on LVBench with 49.7% accuracy and boosting Qwen3-VL-8B's performance by 6.7% on average. Our main contributions are:

- Effective Adaptive Reasoning. We introduce a query-level adaptive mechanism that dynamically selects between native VideoLLM reasoning and multi-step visual program reasoning, achieving an optimal balance between performance and efficiency.
- Reliable Refinement. We propose a self-refinement mechanism guided by execution and confidence signals, enhancing the reliability of program-based reasoning.
- Promising Performance on Long-Form Video. We design a suite of general video modules for retrieval, localization, and inspection, enabling our VideoPro to achieve consistent improvements across multiple challenging long-video benchmarks.

2 Related Work

2.1 Long-form Video Understanding

Understanding long-form videos requires identifying and connecting sparse evidence distributed across extensive temporal horizons (Wu et al., 2024; Fu et al., 2024). Current Video-LLMs typically extend image-based multimodal frameworks (Liu et al., 2023) by sampling multiple frames to capture temporal dynamics (Li et al., 2023; Zhang et al., 2023b; Lin et al., 2023; Li et al., 2024). These models generally align visual features with the language space using a frame-wise encoder followed by a projection module. Despite recent advancements in spatio-temporal representations (Li et al., 2023; Bai et al., 2025), dense processing of long videos remains computationally prohibitive. To mitigate this, alternative strategies leverage textual summaries or keyframe captioning (Zhang et al., 2023a; Wang et al., 2024b, 2025); however, while these methods enhance scalability, they often sacrifice fine-grained visual details and involve complex, multi-step inference. To address this, some strategies use captioning or keyframe

summarization to create textual representations for LLMs (Zhang et al., 2023a; Wang et al., 2024b, 2025), which improve scalability but can lose fine-grained temporal details and require multiple inference steps. VideoPro addresses this trade-off by leveraging native VideoLLMs for video understanding while selectively invoking targeted modules to capture key visual evidence when necessary.

2.2 Visual Program Reasoning

Visual program reasoning empowers LLMs to decompose queries into executable programs that orchestrate perception tools (Gupta and Kembhavi, 2023; Choudhury et al., 2023). However, existing frameworks are predominantly tailored for images or short clips and often suffer from brittleness in complex environments. For instance, ViperGPT (Surís et al., 2023) integrates visual modules for image and short-video QA (Choudhury et al., 2023), while VURF (Mahmood et al., 2024) enhances program reliability. These efforts position LLMs as general-purpose planners for decomposing complex tasks into interpretable steps. We extend this paradigm by introducing Adaptive Reasoning to determine when to invoke programs for long-form videos dynamically.

2.3 Adaptive Reasoning and Refinement

Adaptive computation allocates resources based on problem complexity, akin to the "System 1 vs. System 2" duality (Evans, 2003; Xiao et al., 2025; Sun et al., 2025; Zhang et al., 2025; Sun et al., 2024). Complementary to this, self-refinement enhances reliability by revising outputs based on feedback (Madaan et al., 2023). While prior works often treat routing and refinement as separate prompting strategies, VideoPro unifies them into a learnable framework. We train a unified VideoLLM to jointly perform Adaptive Reasoning and Iterative Refinement, ensuring robust performance for long-form video understanding.

3 Method

3.1 Overview

Given a long-form video $V = \{f_1, \ldots, f_T\}$ and a natural language query Q, our goal is to generate an accurate answer A with low computational cost. In long-form videos, task-relevant evidence is often sparse and temporally dispersed, making dense frame-wise processing inefficient and largely redundant. We thus cast long-form video understanding as a cost-aware dynamic inference problem centered around two core mechanisms: *adaptation* and *refinement*. As illustrated in Figure 3, VideoPro implements a unified adaptive reason-refine process with two coupled components. (1) **Adaptive Reasoning:** VideoPro adaptively routes each query to either *native reasoning* (direct answer generation with native VideoLLM reasoning) when it is confident, or *program reasoning* that explicitly composes and executes multi-step video modules when tool use is necessary. (2) **Self-Refinement:** conditioned on runtime feedback (e.g., execution failures) and low-confidence predictions, VideoPro revises the generated program and re-executes it to recover from errors and improve answer reliability.

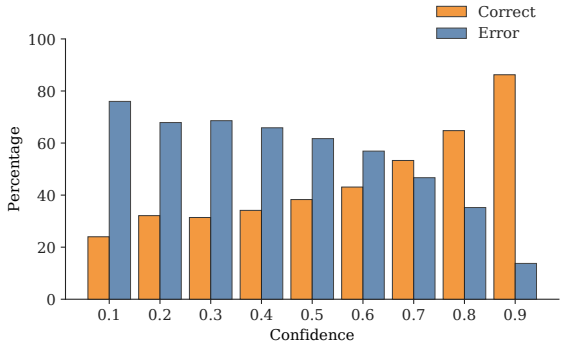

Figure 2: Distribution of correct vs. error predictions across confidence on LongVideoBench. The proportion of correct predictions exceeds errors in $[0.7, 0.8)$ interval, and exceeds 90% when confidence is above 0.9.

3.2 Video Module Library

Prior approaches are mostly frame-centric, often running VQA models (e.g., BLIP-2) on every single frame to find key visual information (e.g., checking for the presence of a text object) (Choudhury et al., 2023; Surís et al., 2023). While this works for short clips, it is extremely slow and inefficient for long videos. In contrast, VideoPro adopts a general video module library tailored for long contexts. Rather than checking every frame blindly, we adopt a coarse-to-fine pipeline: progressing from global semantic retrieval to precise temporal localization, and finally to fine-grained visual information. We structure the capabilities of VideoPro into five core modules: Multimodal Retrieval, Temporal Localization, Fine-grained Visual Extraction, Global Context Summarization, and Reasoning and Answer Generation. In addition to these core modules, VideoPro utilizes basic Python

operations as underlying utilities. More details for each vision module are provided in Appendix A.1.

3.3 Visual Programming

To facilitate multi-step program reasoning with vision modules, VideoPro frames long-video understanding as the execution of a synthesized program P over a module library $\mathcal{M}$. We formally define the execution interface as:

$$(\hat{A}, c) = \text{Exec}(P \mid V, Q, \mathcal{M}), \tag{1}$$

where $\hat{A}$ denotes the predicted answer, $c \in [0, 1]$ is the associated confidence score. This explicit formalism uncovers intermediate reasoning steps, providing essential diagnostic signals for the subsequent refinement stage. Specifically, the confidence score c is derived from the distribution as:

$$c = \exp\left(\frac{1}{L}\sum_{i=1}^{L} \log p_\theta(\hat{a}_i \mid V, Q, \hat{a}_{<i})\right), \tag{2}$$

where $\hat{A} = \{\hat{a}_1, \ldots, \hat{a}_L\}$ denotes the generated response of length L.

3.4 Adaptive Reasoning

To balance reasoning effectiveness with computational efficiency, VideoPro introduces Adaptive Reasoning via query-conditioned task planning. Given a query Q, the model acts as a planner that adaptively selects one of two reasoning modes:

- **Native VideoLLM reasoning:** If the VideoPro determines that the query can be answered using holistic understanding with native VideoLLM, it will generate programs that only use the query_native module as a single-step program call. Without multi-step reasoning, the VideoLLM will generate the prediction directly from the video frames, avoiding unnecessary vision module invocations.
- **Multi-step visual program reasoning:** If VideoPro identifies a need for multi-step and video modules, it will explicitly collect multi-step evidence before deriving the answer.

In both modes, the model outputs an answer $\hat{A}$ along with a confidence score c, which serves as a critical quality indicator for the refinement stage.

3.5 Self-Refinement

To improve robustness against the rigidity of static visual programs that are produced in a single pass, VideoPro incorporates a self-refinement mechanism designed to correct invalid executions and revise low-confidence reasoning programs:

- **Refinement for failed executions:** When a visual program encounters a runtime failure (e.g., empty retrieval results or invalid arguments), the model inspects the execution log to diagnose the issue and refine a corrected program.
- **Refinement for Low-Confidence Reasoning:** Even if execution completes successfully, the prediction may have low confidence. If $c < \tau$ the model modifies the reasoning program, such as broadening the retrieval scope or adjusting invoked modules, and re-executes the refined program to produce a more reliable answer.

3.6 Training Pipeline

We denote our trained VideoLLMs to execute the framework end-to-end using a two-stage pipeline: Supervised Fine-Tuning (SFT), followed by Group Relative Policy Optimization (GRPO) to further enhance reasoning quality and efficiency. To facilitate reproducibility, we provide the detailed prompts in our tasks in Appendix A.2.

Supervised Fine-Tuning. We construct a *reason-and-refine* dataset using a teacher model, covering three execution trajectories: runtime failures, successful executions with incorrect predictions, and correct predictions. The supervision is organized into two types:

- **Adaptive Reasoning Supervision:** To train VideoLLMs to select appropriate reasoning modes, we categorize each query into different reasoning regimes. Queries that can be correctly answered by the native VideoLLM with high confidence ($c > 0.75$) are supervised with `Native Reasoning` (R1); Otherwise, they are supervised with `Multi-step Visual Program Reasoning` (R2), where ground-truth correct predictions of visual programs are provided.
- **Refinement Supervision:** To enable reasoning refinement, we construct supervision signals: (i) Execution Failure Refinement (R3), where failed programs and their runtime logs are paired with correct predictions of visual programs; and (ii) Low-Confidence / Incorrect Reasoning Refinement (R4), where programs yielding incorrect or low-confidence predictions are revised into corrected prediction programs. We provide relevant prompts in Appendix A.2.

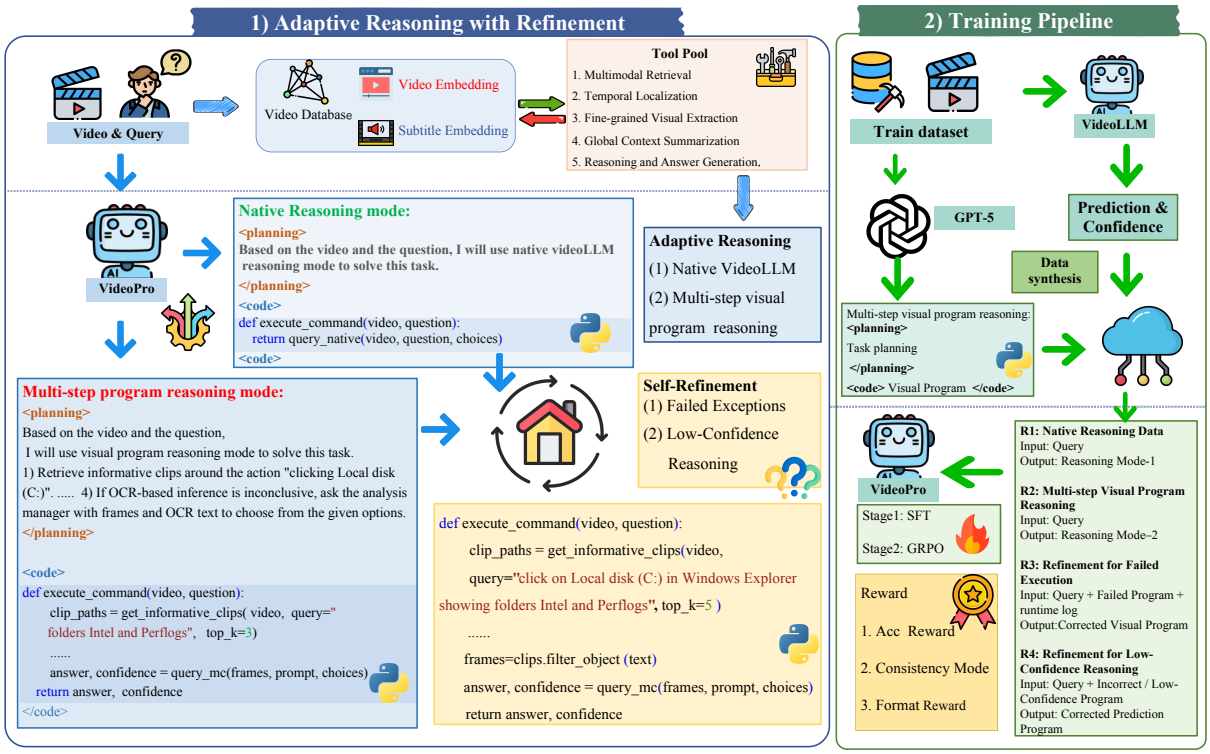

Figure 3: **(a) Adaptive Reasoning & Self-Refinement:** VideoPro dynamically selects between Native VideoLLM and Multi-step visual program reasoning based on query complexity. Self-refinement is employed to correct failed executions and low-confidence reasoning programs. **(b) Training Pipeline:** The process involves (1) SFT on the *reason-and-refine* dataset, and (2) GRPO to optimize rewards for correctness, format validity, and consistency.

Group Relative Policy Optimization. To better align generation with the desired behavior, we further optimize VideoLLMs with GRPO (Guo et al., 2025). For each query, we sample a group of outputs and maximize a composite reward:

$$\mathcal{R} = 0.5 \cdot r_{\text{acc}} + 0.2 \cdot r_{\text{fmt}} + 0.3 \cdot r_{\text{mode}}, \quad (3)$$

where r_{acc} rewards answer correctness, r_{fmt} penalizes invalid program syntax, and r_{mode} (Mode Consistency Reward, more details in Appendix A.3.) encourages selecting the reasoning mode consistent with the labels (where native VideoLLM can solve them with high confidence).

4 Experiments

Benchmarks We evaluate VideoPro on four benchmarks: (1) LongVideoBench (Wu et al., 2024), using the validation set across diverse durations; (2) VideoMME (Fu et al., 2024), where we focus on the long subset ($>$ 600s) to assess long-range reasoning; (3) LVBench (Wang et al., 2024a), featuring extremely long videos (up to 2h) with complex temporal logic; and (4) MLVU (Zhou et al., 2025), a multi-task benchmark where we report results on the test set.

Implementation Details We employ Qwen3-VL-8B (Yang et al., 2025) and Qwen2.5-VL-7B (Bai et al., 2025) as the backbone VideoLLM, utilizing a strong proprietary LLM, GPT5 (OpenAI, 2025) for data synthesis (which only inputs the query and few-shot examples). For the video database, we process long-duration videos by dividing them into 10-second clips. These segments are then encoded using LanguageBind_Video (Zhu et al., 2023) and paired with subtitles extracted by FFmpeg and Whisper (Radford et al., 2023). The framework integrates DEVA (Cheng et al., 2023) and EasyOCR to support object and text grounding. Implemented within the MS-SWIFT framework (Zhao et al., 2024), our two-stage training consists of SFT on 5k trajectories from CG-Bench (Chen et al., 2024a) for 1 epoch, followed by Group Relative Policy Optimization (GRPO) on 10k samples for 1 epoch. During inference, the model processes up to 64 video frames, automatically triggering a refinement process if the confidence score falls below $\tau = 0.75$. We set $\tau = 0.75$ since it already yields strong accuracy shown in Figure 2; while higher confidence (e.g., 0.9) is even more reliable, refining all such cases would be unnecessary. The

Model	Frames	LVBench	VideoMME$_L$	LongVideoBench	MLVU	Avg.
Closed-source Models						
GPT-4o (Hurst et al., 2024)	384	48.9	72.1	66.7	54.9	60.7
OpenAI o3 (OpenAI, 2025)	256	57.1	64.7	67.5	–	–
Gemini-1.5-pro (Team et al., 2024)	256	33.1	67.4	58.6	–	–
Seed1.5VL-pro (Team, 2025)	32	46.1	63.3	63.7	54.9	57.0
Open-source Models						
Qwen2.5-VL-72B (Bai et al., 2025)	128	47.4	64.6	60.3	53.8	56.5
LongVILA-7B (Chen et al., 2024b)	256	–	52.1	57.7	49.0	–
VideoMind-7B (Liu et al., 2025)	2/FPS	40.8	49.2	–	–	–
Video-R1-7B (Feng et al., 2025)	64	36.2	48.4	53.9	–	–
Video-XL-7B (Shu et al., 2025)	256	–	54.9	50.7	45.5	–
Qwen2.5-VL-7B (Bai et al., 2025)	64	38.3	50.0	58.6	48.0	48.7
Qwen3-VL-8B (Yang et al., 2025)	64	40.2	56.3	61.5	53.6	52.9
Agentic LLMs						
VideoAgent (GPT-4) (Wang et al., 2024b)	–	–	46.2	–	52.2	–
VideoAgent (GPT-4) (Fan et al., 2024)	–	–	48.1	–	55.4	–
VideoTree (Qwen-Plus)	–	–	39.3	–	51.6	–
Ours (Reason + Refine)						
Qwen2.5-VL-7B + VideoPro	64	47.2	56.7	60.9	49.6	53.6 ↑4.9
Qwen3-VL-8B + VideoPro	64	**49.7**	**68.8**	**64.5**	**55.2**	**59.6** ↑6.7

Table 1: **Quantitative results on long-video benchmarks.** We report performance across four benchmarks. Notably, the trained VideoLLM within our VideoPro framework outperforms the native VideoLLM, achieving performance gains of 4.9% and 6.7%, respectively.

sampling temperature is set to 0.7 when generates visual programs.

Baselines Our comparative study involves three distinct categories of state-of-the-art models: (i) *Closed-source models*, represented by GPT-4o (Hurst et al., 2024), Gemini-1.5 Pro (Team et al., 2024), and Seed-1.5VL-Pro (Team, 2025); (ii) *Open-source models*, including Qwen2.5-VL (Bai et al., 2025), LongVILA-7B (Chen et al., 2024b), Video-XL-7B (Shu et al., 2025); (iii) *Agentic frameworks*: VideoAgent (Wang et al., 2024b) and VideoTree (Wang et al., 2025). All models are evaluated following their official decoding configurations to show the advantages of our framework.

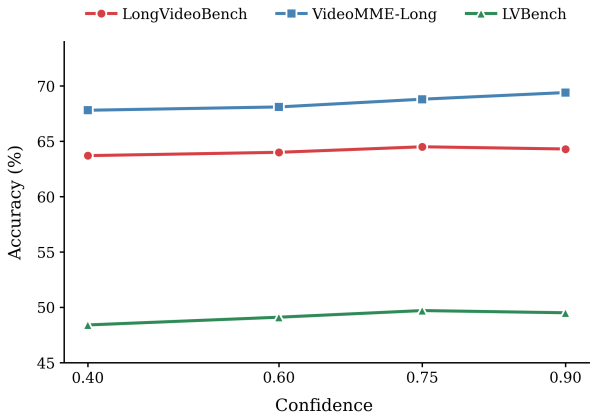

Figure 4: **Performance at varying confidence thresholds.** VideoPro exhibits robust performance on the Long Video Benchmark across the wide interval of $[0.4, 0.9]$.

4.1 Main Results

Table 1 summarizes the quantitative results on long-form video benchmarks. By adaptively switching between *native* direct answering and *program-based* reasoning, VideoPro consistently improves over the corresponding native VideoLLM backbones. The largest gains appear on benchmarks that demand long-range temporal aggregation. In particular, on LVBench and VideoMME$_L$, VideoPro improves Qwen3-VL by more than 10% relative. On LongVideoBench and MLVU, the gains remain consistent but are smaller (+3.0 and +1.6 points, respectively), which we attribute to their broader mix of short- and long-horizon questions. As native VideoLLMs are already highly capable of handling the short-video subset, the overall gain is naturally averaged down.

4.2 Ablation Studies

Reasoning Paradigms. We evaluate VideoPro against three baseline paradigms: Native VideoLLM, Multi-step Visual Program, and the base Adaptive Reasoning without refinement. As reported in Table 2, Native VideoLLM exhibits the lowest latency but suffers from inferior accuracy due to its limited reasoning depth. In contrast,

Dataset	Method	Avg. Acc. (%) ↑	Output Len. ↓	Avg. Runtime ↓
VideoMME$_L$	Native VideoLLM	56.3	255	2.1s
	Multi-step Visual Program	66.7	1496	8.2s
	Adaptive Reasoning	66.8	825	5.2s
	+ Refine	**68.8**	1326	7.1s
LVBench	Native VideoLLM	40.2	255	2.3s
	Multi-step Visual Program	46.4	1594	8.2s
	Adaptive Reasoning	48.1	1227	5.9s
	+ Refine	**49.7**	1435	7.8s

Table 2: Performance and Efficiency Comparison. We evaluate efficiency-accuracy trade-offs across different reasoning paradigms. VideoPro achieves the best balance by combining adaptive reasoning with refinement.

Method (w/o)	VideoMME$_L$		LVBench	
	Acc.	Δ	Acc.	Δ
Full Model	**61.2**		**44.5**	
w/o Retrieval	57.3	−3.9	42.8	−1.7
w/o Temp. Loc.	57.5	−3.7	42.8	−1.7
w/o Fine-grained Vis.	58.1	−3.1	43.9	−0.6
w/o Global Context	56.6	−4.6	43.8	−0.7

Table 3: Ablation studies on different modules using Qwen3-VL (SFT-only). We report the accuracy (%) and the performance drop (Δ).

Base Model	Strategy	VideoMME$_L$		LVBench	
		Reasoning	+Refine	Reasoning	+Refine
Qwen3-VL	SFT	61.2	65.0	44.5	49.1
	SFT+GRPO	66.8	**68.8**	48.1	**49.7**
Qwen2.5-VL	SFT	52.2	58.1	42.1	46.7
	SFT+GRPO	53.2	**56.7**	45.4	**47.2**

Table 4: Ablation of training strategies and refinement mechanisms across models.

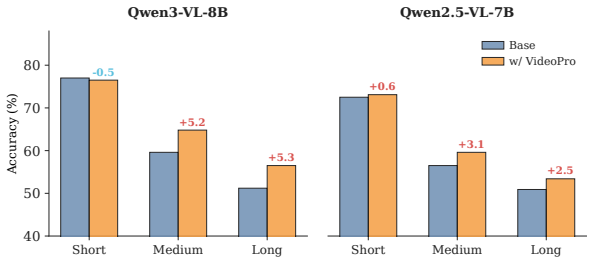

Figure 5: Accuracy on LongVideoBench and VideoMME across different video durations.

Multi-step Visual Program improves performance at the expense of a significant increase in output length and runtime. The base Adaptive Reasoning achieves a more favorable trade-off by reducing computational overhead while maintaining high accuracy. By incorporating the refinement stage, VideoPro achieves the best performance, reaching 68.8% on VideoMME$_L$ and 49.7% on LVBench. These results demonstrate that the self-refinement mechanism effectively rectifies execution failures and low-confidence reasoning programs, striking an optimal balance between reasoning quality and execution efficiency.

Confidence Threshold We examine the effect of the confidence threshold τ, which serves as the decision boundary for our model's reasoning process. As shown in Figure 4, increasing τ from 0.40 to 0.75 consistently improves performance across all benchmarks. Specifically, LVBench and LongVideoBench achieve their peak accuracy of 49.7% and 64.5%, respectively at $\tau = 0.75$. However, further raising the threshold to 0.90 leads to a slight performance degradation on these datasets, implying that an excessively strict threshold might filter out valid cues or reasoning paths.

Video Modules. Table 3 presents an ablation study on the Qwen3-VL (SFT) backbone to evaluate the contribution of individual components within our library $\mathcal{M}$. Global Context Summarization is indispensable for long-form reasoning, as its removal causes the most significant performance drop on VideoMME$_L$, where the accuracy decreases from 61.2% to 56.6%. Regarding precise grounding, omitting either Multimodal Retrieval or Temporal Localization leads to a marked regression in LVBench accuracy, dropping to 42.8%. Furthermore, the absence of Fine-grained Visual Extraction consistently impairs performance across all benchmarks. Collectively, these results empirically validate the necessity of our coarse-to-fine design, as the full pipeline achieves the optimal balance between efficiency and accuracy.

Figure 6: Cases of adaptive program reasoning: native VideoLLM reasoning, multi-step visual program reasoning, and reasoning with refinement where it explicitly invokes vision modules to capture fine-grained details.

Training Strategies. We study the effectiveness of our two-stage learning framework by comparing models trained with SFT against our full SFT+GRPO pipeline. As shown in Table 4, while SFT provides the basic reasoning and tool-use capabilities, adding GRPO leads to a clear performance boost. For example, Qwen3-VL with base reasoning improves from 61.2% to 66.8% after GRPO training. This improvement shows that reinforcement learning helps the model better handle complex queries through reward alignment. Furthermore, the Refinement module consistently improves results under both training settings. For instance, it increases the SFT-only score of Qwen3-VL from 61.2% to 65.0%. These results prove that encouraging the model to self-correct its reasoning paths effectively reduces errors in difficult cases. Overall, the combination of GRPO training and the Refinement module achieves the best performance.

Video Durations. We evaluate VideoPro across three duration groups: Short ($<$ 2 min), Medium (2–15 min), and Long ($>$ 15 min). As shown in Figure 5, while performance on Short videos remains comparable to the baselines, VideoPro significantly improves accuracy as duration increases. Specifically, for Qwen3-VL-8B, we observe absolute gains of 5.2% and 5.3% on the Medium and Long splits, respectively. This trend, where the performance gap widens with video length, shows VideoPro's efficacy in handling long videos.

4.3 Case Study

We present representative cases in Figure 6 to show the adaptive reasoning and refinement process of VideoPro. For simple actions, the model utilizes **Native VideoLLM reasoning** to directly predict the action, ensuring efficiency. For complex long-video queries (middle), the planner invokes **Multi-step visual program reasoning** to ground specific subtitles and locate relevant intervals. To further enhance precision, VideoPro performs **self-refinement** by employing `detect_object` and `crop` to isolate the target and filter out noise and retrieve relevant clips using scene-level textual descriptions. Crucially, VideoPro integrates the native ability to reason over uniformly sampled frames with the program's capability to provide localized visual evidence.

5 Conclusion

We propose VideoPro, a unified and adaptive reason-refine framework for long-form video understanding that explicitly balances accuracy and efficiency. To overcome the rigidity of static inference, VideoPro performs adaptive planning to select between *native* mode (direct answering with the native VideoLLM) and *program* mode (multi-step, executable visual programs) based on the query's reasoning demands. To improve robustness on challenging grounding and temporal reasoning, we introduce a dual-signal self-refinement mechanism that uses execution feedback and confidence-aware triggers to detect and correct failures. Finally, we integrate these components into a GRPO-based training pipeline, enabling VideoPro to achieve strong empirical performance while producing multi-step reasoning steps through executable programs.

Limitations

VideoPro works well for long-form video understanding but has some limitations: (1) its temporal grounding depends on a manually curated vision-module library, reducing zero-shot flexibility and requiring manual integration or task-specific training for new categories/domains; (2) it can be overconfident (e.g., confidence > 0.9 on wrong reasoning/answers), so self-refinement may fail when the VideoLLM is confidently wrong, potentially misleading users.

Ethics and Potential Risks

Deploying VideoPro raises ethical concerns about reliability in high-stakes settings: its overconfidence can cause "silent failures" (e.g., forensics or autonomous monitoring) where wrong outputs with high certainty bias human decisions. Because it uses LLMs for program synthesis and summarization, it is also vulnerable to hallucinations. Finally, since training data is teacher-synthesized and optimized via GRPO, societal or cultural biases in teacher models may be absorbed and amplified.

References

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, and 1 others. 2025. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*.

Guo Chen, Yicheng Liu, Yifei Huang, Yuping He, Baoqi Pei, Jilan Xu, Yali Wang, Tong Lu, and Limin Wang. 2024a. Cg-bench: Clue-grounded question answering benchmark for long video understanding. *arXiv preprint arXiv:2412.12075*.

Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, and 1 others. 2024b. Longvila: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*.

Ho Kei Cheng, Seoung Wug Oh, Brian Price, Alexander Schwing, and Joon-Young Lee. 2023. Tracking anything with decoupled video segmentation. In *ICCV*.

Rohan Choudhury, Koichiro Niinuma, Kris M Kitani, and László A Jeni. 2023. Zero-shot video question answering with procedural programs. *arXiv preprint arXiv:2312.00937*.

Jonathan St BT Evans. 2003. In two minds: dual-process accounts of reasoning. *Trends in cognitive sciences*, 7(10):454–459.

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

Kaituo Feng, Kaixiong Gong, Bohao Li, Zonghao Guo, Yibing Wang, Tianshuo Peng, Junfei Wu, Xiaoying Zhang, Benyou Wang, and Xiangyu Yue. 2025. Video-r1: Reinforcing video reasoning in mllms. *arXiv preprint arXiv:2503.21776*.

Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, and 1 others. 2024. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*.

Minghe Gao, Juncheng Li, Hao Fei, Liang Pang, Wei Ji, Guoming Wang, Zheqi Lv, Wenqiao Zhang, Siliang Tang, and Yueting Zhuang. 2024. De-fine: De composing and re fin ing visual programs with auto-feedback. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 7649–7657.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14953–14962.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and 1 others. 2024. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*.

KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. 2023. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*.

Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. 2023. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Ye Liu, Kevin Qinghong Lin, Chang Wen Chen, and Mike Zheng Shou. 2025. Videomind: A chain-of-lora agent for long video reasoning. *arXiv preprint arXiv:2503.13444*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Ahmad Mahmood, Ashmal Vayani, Muzammal Naseer, Salman Khan, and Fahad Shahbaz Khan. 2024. Vurf: A general-purpose reasoning and self-refinement framework for video understanding. *arXiv preprint arXiv:2403.14743*.

OpenAI. 2025. Gpt-5. https://openai.com/gpt-5/. Accessed: 2025-12-25.

OpenAI. 2025. Introducing OpenAI o3 and o4-mini. https://openai.com/index/introducing-o3-and-o4-mini/. Accessed: 2025-05-15.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *International conference on machine learning*, pages 28492–28518. PMLR.

Yan Shu, Zheng Liu, Peitian Zhang, Minghao Qin, Junjie Zhou, Zhengyang Liang, Tiejun Huang, and Bo Zhao. 2025. Video-xl: Extra-long vision language model for hour-scale video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26160–26169.

Guangyan Sun, Mingyu Jin, Zhenting Wang, Cheng-Long Wang, Siqi Ma, Qifan Wang, Tong Geng, Ying Nian Wu, Yongfeng Zhang, and Dongfang Liu. 2024. Visual agents as fast and slow thinkers. *arXiv preprint arXiv:2408.08862*.

Yiliu Sun, Yanfang Zhang, Zicheng Zhao, Sheng Wan, Dacheng Tao, and Chen Gong. 2025. Fast-slow-thinking: Complex task solving with large language models. *arXiv preprint arXiv:2504.08690*.

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

ByteDance Seed Team. 2025. Seed1.5-vl technical report. *arXiv preprint arXiv:2505.07062*.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Weihan Wang, Zehai He, Wenyi Hong, Yean Cheng, Xiaohan Zhang, Ji Qi, Xiaotao Gu, Shiyu Huang, Bin Xu, Yuxiao Dong, and 1 others. 2024a. Lvbench: An extreme long video understanding benchmark. *arXiv preprint arXiv:2406.08035*.

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

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

Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. 2024. Longvideobench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems*, 37:28828–28857.

Wenyi Xiao, Leilei Gan, Weilong Dai, Wanggui He, Ziwei Huang, Haoyuan Li, Fangxun Shu, Zhelun Yu, Peng Zhang, Hao Jiang, and 1 others. 2025. Fast-slow thinking for large vision-language model reasoning. *arXiv preprint arXiv:2504.18458*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. 2023a. A simple llm framework for long-range video question-answering. *arXiv preprint arXiv:2312.17235*.

Hang Zhang, Xin Li, and Lidong Bing. 2023b. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*.

Shengjia Zhang, Junjie Wu, Jiawei Chen, Changwang Zhang, Xingyu Lou, Wangchunshu Zhou, Sheng Zhou, Can Wang, and Jun Wang. 2025. Othink-r1: Intrinsic fast/slow thinking mode switching for over-reasoning mitigation. *arXiv preprint arXiv:2506.02397*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Zhengyang Liang, Shitao Xiao, Minghao Qin, Xi Yang, Yongping Xiong, Bo Zhang, and 1 others. 2025. Mlvu: Benchmarking multi-task long video understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 13691–13701.

Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, Wang HongFa, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, Cai Wan Zhang, Zhifeng Li, Wei Liu, and Li Yuan. 2023. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. *Preprint*, arXiv:2310.01852.

A Appendix

A.1 Vision modules

We implement a general video module library to facilitate coarse-to-fine video reasoning. This library streamlines the transition from global context retrieval to fine-grained visual probing. We summarize the API reference, where each module is specialized for a distinct sub-task to ensure efficient and accurate program execution:

- **Multimodal Retrieval.** Retrieves semantically relevant clips and transcripts with offline embeddings efficiently, avoiding exhaustive frame-by-frame scanning and significantly reducing the search space from the frame level to the clip level.
- **Temporal Localization.** Taking the identified temporal intervals as input, the module retrieves the corresponding video frames and subtitles. This filtering reduces irrelevant context and focuses the model on query-relevant visual and textual evidence.
- **Fine-grained Visual Extraction.** Conducts detailed analysis on selected frames, such as object detection and optical character recognition (OCR), to extract fine visual details.
- **Global Context Summarization.** Aggregates key thematic and narrative information from the full transcript, providing essential context for queries requiring understanding of the video's overall storyline.
- **Reasoning and Answer Generation.** Integrates gathered evidence to generate final answers. VideoPro is prompted to answer the question with frames by outputting a single uppercase letter.

A.2 Prompt Details

We prompt GPT-5 with the vision-module specifications (Fig. 7) and few-shot examples (Fig. 8) to synthesize a multi-step visual program. Concurrently, the native VideoLLM processes the query, yielding a prediction with a confidence score via `query_native`; high-confidence outputs serve as labels for the native reasoning mode. VideoPro initiates with Adaptive Reasoning (Fig. 9), where native outputs adhere to the format in Fig. 10 and multi-step outputs follow Fig. 8. Subsequently, VideoPro applies a refinement mechanism when program execution fails or confidence is insufficient: execution failures trigger `prompt_refine1` (Fig. 11), while low-confidence outcomes in native and multi-step reasoning utilize Fig. 12 and Fig. 13, respectively.

Dataset	Native (%)	Program (%)	Refinement (%)
LongVideoBench	29.4	70.6	35.9
LVBench	17.7	82.3	55.5

Table 5: Distribution of execution modes and refinement ratios.

A.3 Mode Consistency Reward

We design the mode consistency reward to encourage the model to select the native reasoning mode only when the backbone VideoLLM can solve the query correctly and confidently without invoking auxiliary modules. For each training sample, we first run `query_native` to obtain a native prediction and its confidence score $(\hat{A}_{\text{nat}}, c_{\text{nat}})$, where c_{nat} is computed from the output token probabilities. We then define an oracle mode label $m^\star$ as:

$$m^\star = \begin{cases} \text{native}, & \text{if } \hat{A}_{\text{nat}} = y \text{ and } c_{\text{nat}} \geq \tau, \\ \text{program}, & \text{otherwise}, \end{cases}$$

where y is the ground-truth answer and τ is a confidence threshold (0.75). Finally, we define

$$r_{\text{mode}} = \mathbb{I}[m = m^\star],$$

So the model is rewarded for selecting the native reasoning mode on simple queries and the program reasoning mode otherwise.

A.4 Execution Mode Analysis

Table 5 summarizes the distribution of execution modes in VideoPro. The majority of queries across both benchmarks are processed via multi-step program reasoning. This is particularly evident on LVBench, where 82.3% of tasks utilize the program mode. Such results highlight the necessity of multi-step reasoning for complex long-video understanding. Additionally, LVBench triggers the refinement mechanism more frequently than LongVideoBench, reaching a ratio of 55.5%.

```python
# --- 1. Multimodal Retrieval ---
def get_informative_clips(video_path, query, top_k=3, total_duration=None):
    """Retrieves visual intervals based on semantic text descriptions."""
    pass

def get_informative_subtitles(video_path, query, top_k=1, total_duration=None):
    """Retrieves intervals using audio subtitle text matching."""
    pass

# --- 2. Temporal Localization ---
def trim_around(video_path, timestamp, intervals=30, num_frames=64):
    """Extracts frames centered at a specific timestamp."""
    pass

def trim_frames(video_path, start, end, num_frames=64):
    """Retrieves frames within a specified [start, end] interval."""
    pass

# --- 3. Fine-grained Visual Extraction ---
def detect_object(frame, text, box_threshold=0.5, text_threshold=0.25):
    """Locates specific objects within a frame."""
    pass

def run_ocr(frame):
    """Extracts visual text (OCR) from the frame."""
    pass

# --- 4. Global Context Summarization ---
def get_subtitle_hints(video_path, question, choices, duration, word_number=300):
    """Summarizes narrative context from the full transcript."""
    pass

# --- 5. Reasoning and Answer Generation ---
def query_native(video_path, question, choices):
    """
    Native VideoLLM reasoning.
    Returns: (prediction, confidence_score)
    """
    pass

def query_mc(frames, question, choices):
    """
    Multiple-choice QA based on localized frames.
    Returns: (prediction, confidence_score)
    """
    pass
```

Listing 1: Some API Reference for Video Module Library

Vision Modules API Definitions

1. Multimodal Retrieval (Offline semantic search)

- `get_informative_clips(video, query, k=3)`: Returns relevant visual intervals.
- `get_informative_subtitles(video, query, k=1)`: Returns intervals based on audio text.

2. Temporal Localization (Context filtering)

- `trim_{after, before, around, frames}(...)`: Slices video into target temporal segments.

3. Fine-grained Visual Extraction (Local detail analysis)

- `detect_object(frame, text)`: Returns bounding boxes for specified objects.
- `run_ocr(frame)`: Extracts visual text; `crop(frame, box)`: Zooms into regions.

4. Global Context Summarization (Narrative aggregation)

- `get_subtitle_hints(video, question, choices)`: Summarizes full-transcript evidence.

5. Reasoning and Answer Generation (VideoLLM inference)

- `query_native(video, q, c)`: Native VideoLLM reasoning for direct option selection.
- `query_mc(frames, q, c)` / `query_yn(frames, q)`: VQA over localized frames.

Figure 7: The predefined API library for our vision modules. The system decomposes complex queries into sub-tasks using these five functional blocks to bridge the gap between raw video data and program reasoning.

Few-shot Example for Visual Program Reasoning

Question: In the video, a woman wearing glasses and a brown top folded the lower right corner of the book. What vegetables were on the book?
Choices:

- A. pumpkins B. beans C. corn D. carrots E. squash F. peas G. potatoes

<planning>
Based on the video and the question, I will use the visual program reasoning mode to solve this task. First, retrieve clips showing the woman in glasses and a brown top, folding the book. Then, filter frames containing the book. If no such frames are found, use all frames. Finally, query the model to identify the vegetables on the book from the given choices.
</planning>

<code>
```
def execute_command(video, question):
    intervals, clip_paths = get_informative_clips(
        video,
        "woman wearing glasses and a brown top folding the book",
        top_k=1,
    )
    frames = []
    for clip in clip_paths:
        frames.extend(extract_frames(clip))
    book_frames = []
    for frame in frames:
        if detect_object(frame, "book"):
            book_frames.append(frame)
    if not book_frames:
        book_frames = frames
    prompt = "What vegetables were on the book?"
    return query_mc(book_frames, prompt, choices)
```
</code>

Figure 8: **Few-shot Example of Visual Program reasoning.**

Adaptive Reasoning Prompt

You will receive a multiple-choice question about a video. Your output must define a Python function in the following format:

<planning>
Briefly judge whether the question can be direct solved by videoLLM and plan the main API calls and reasoning steps you will use.

- If it can be answered, use native reasoning mode with a single `query_native` call.
- If it needs long-range or multi-step reasoning, use more detailed visual program mode with other modules (retrieval, subtitles, frame analysis, etc.).

</planning>

<code>
Write a Python function in the following format.

```
def execute_command(video, question):
    # Visual program code (no comments needed inside the code body).
    ...
    return answer
```
</code>

Figure 9: **Adaptive Reasoning Prompt Template. (Input for R1 & R2)**

Native VideoLLM Reasoning Output

<planning>
Based on the video and the question, I will use native videoLLM reasoning mode to solve this task.
</planning>

<code>

```
def execute_command(video, question):
    return query_native(video, question, choices)
```
</code>

Figure 10: **Example Output for Native VideoLLM Reasoning Mode. (Output for R1)**

Refinement for Failed Program

You will receive a multiple-choice question about a video and a Python visual program in the `execute_command` format, and a runtime error log from running this program.

`{question_with_choices}`

Buggy visual program: {}

Runtime error log: {}

Refine this visual program by fixing the bugs.

Figure 11: **Refinement Prompt for Failed Program. (Input for R3)**

Refinement for Low-Confidence Native Reasoning

You will receive a multiple-choice question about a video and an existing visual program that only uses the native-mode helper API `query_native`.

`{question_with_choices}`

Current native visual program:
<code>
```
def execute_command(video, question):
    return query_native(video, question, choices)
```
</code>

Refine this visual program.

Figure 12: **Refinement Prompt for Low-Confidence Native Reasoning. (Input for R4)**

Refinement for Low-Confidence Program Reasoning

You will receive a multiple-choice question about a video and an existing visual program.

`{question_with_choices}`

Current visual program: {}

Refine this visual program to improve its reasoning and correctness.

Figure 13: **Refinement Prompt for Low-Confidence Program Reasoning. (Input for R4)**

Prompt for Answer Generation

Select the best answer to the following multiple-choice question based on the video. Respond with only the letter (A, B, C, or D or other letter) of the correct option.

`{question_with_choices}`

Output a single letter. The best answer is:

Figure 14: Prompt for Answer Generation. This prompt guides the model to output a distinct uppercase letter prediction consistent with the vision module interface.