Searching Videos as Trees: Self-Correcting Agents for Grounded Long Video QA

Ce Zhang[1] **Ziyang Wang**[1] **Yulu Pan**[1] **Oluwatumininu Oguntola**[1] **Pranav Wagh**[1]
Qiyu Wu[2] **Hiromi Wakaki**[2] **Mohit Bansal**[1] **Gedas Bertasius**[1]
[1]University of North Carolina at Chapel Hill [2]Sony
{cezhang, ziyangw, yulupan, mohit, gedas}@cs.unc.edu,
{Qiyu.Wu, Hiromi.Wakaki}@sony.com

Abstract

Grounded long-video question answering (Grounded LVQA) requires answering a question about a long video while localizing the short evidence interval that supports the answer. Recent agentic methods frame this task as multi-turn exploration with a single `crop_video(start, end)` action, which supports coarse-to-fine narrowing but provides no primitive for fine-to-coarse backtracking. As a result, these agents typically converge prematurely and cannot recover from an early mistake. We propose **VideoTreeSearch (VTS)**, a framework that casts grounded LVQA as iterative self-correcting search over an adaptive temporal tree. VTS constructs a non-uniform tree through adaptive temporal segmentation so that each node corresponds to a semantically coherent segment, and trains an agent to navigate the tree through four discrete operations: `zoom_in`, `zoom_out`, `shift`, and `answer`. These operations expose backtracking and recovery as explicit, learnable primitives rather than implicit behaviors. To train this navigation, we introduce a trajectory synthesis pipeline that produces multi-step paths through the tree, including deliberate detours into incorrect branches followed by recovery. We use these trajectories for supervised fine-tuning, followed by reinforcement learning with grounding and answer-accuracy rewards. On three Grounded LVQA benchmarks (CG-Bench, Haystack-LVBench, Haystack-Ego4D), VTS outperforms the strongest prior multi-turn cropping agents by $+12.5$ mIoU on CG-Bench and $+7.4$ T-F1 on Haystack-Ego4D. The learned policy also transfers to general long-video QA, surpassing all prior multi-turn cropping agents on Video-MME, MLVU, and LVBench by up to $+7.1$ accuracy points. Ablations confirm that self-correcting hierarchical search is the central mechanism behind these gains: removing either adaptive descent or explicit backtracking substantially degrades performance. Code is available at `https://github.com/CeeZh/VTS`.

1 Introduction

Consider an hour-long cooking tutorial paired with the question, "What did the chef add to the dish right before placing it in the oven?" Answering it requires the model to first locate the brief moment when the dish is placed in the oven, and then identify what the chef added immediately before. Many real-world long videos share this structure: the evidence needed to answer a question occupies only a small portion of the video, while the surrounding content is largely irrelevant. This task, known as grounded long-video question answering (Grounded LVQA) [5, 50], requires a model to simultaneously localize the supporting evidence interval and produce a final answer. Uniform frame sampling is poorly suited to this setting because sampling too few frames likely skips past the brief evidence interval. In contrast, sampling too many frames exceeds the input budget that current

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

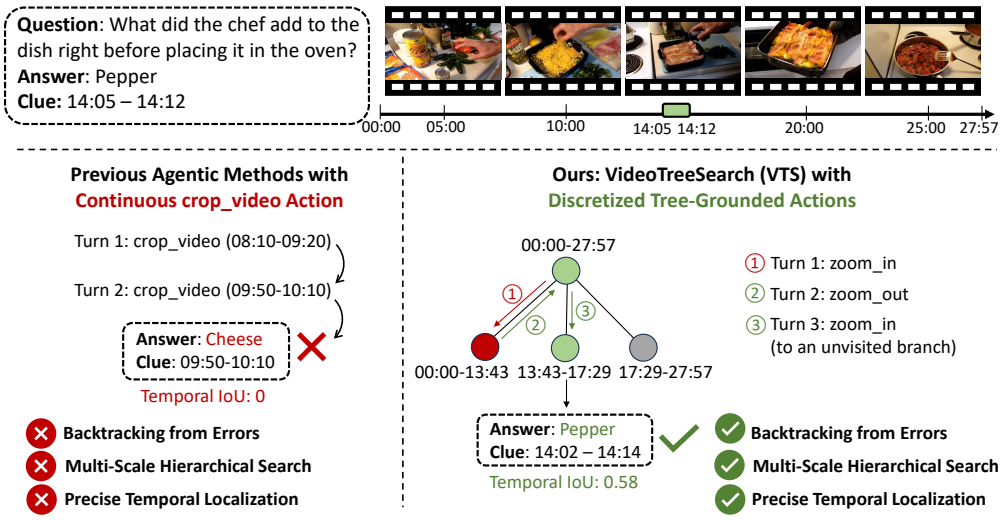


Figure 1: **Comparison between prior multi-turn cropping agents and our VideoTreeSearch (VTS) on grounded LVQA.** Prior multi-turn cropping agents (bottom left) rely on a continuous `crop_video` action that conflates exploration with correction. As a result, they often converge prematurely and cannot recover from early mistakes. Our proposed VideoTreeSearch (bottom right) instead organizes the video as a tree of semantically coherent segments and lets the agent navigate it through discrete actions (`zoom_in`, `zoom_out`, `shift`). This turns precise localization and backtracking from implicit capabilities into explicitly learnable behaviors.

video models can process. Grounded LVQA, therefore, calls for an iterative search, in which the model hypothesizes where the evidence lies and progressively narrows into promising regions.

Recent work formulates this search as agentic interaction with the long video [48, 26, 53, 10, 44]. Most of these methods equip a VLM with a single cropping tool, `crop_video(start, end)`, and let the agent repeatedly crop the video [48, 26, 53]. While this unified action is flexible in principle, it has two structural shortcomings. First, the action space is asymmetric. Cropping naturally narrows the search but provides no analog for fine-to-coarse backtracking when the agent commits to a wrong video region. Second, continuous timestamp regression provides no hierarchical decomposition of the video. The agent must localize evidence from raw pixels with no prior on where semantically coherent segments lie, leaving the search space flat and unorganized. As a result, these cropping agents typically converge prematurely and cannot recover from an early mistake.

We propose **VideoTreeSearch (VTS)**, a framework that treats the long video as an adaptive temporal tree and replaces continuous timestamp regression with discrete navigation over its nodes. Our key insight is that organizing the video as a tree of semantically coherent segments, and letting the agent navigate that tree through discrete actions, turns precise localization and backtracking from implicit capabilities into explicitly learnable behaviors. Concretely, VTS first constructs a non-uniform tree whose boundaries are derived from visual and semantic transitions using CLIP [28] embeddings, so that each tree node corresponds to a semantically coherent segment. The agent then navigates this tree through four discrete operations: `zoom_in` to descend into a child segment, `zoom_out` to ascend to the parent, `shift` to move laterally to a sibling, and `answer` to commit. This design offers two structural advantages over a flat continuous action space. Because tree nodes are pre-aligned with semantic content boundaries, the model no longer needs to regress precise timestamps from raw pixels, and localization reduces to selecting a node. Exploration and self-correction also become separate primitives. The `zoom_in` action supports adaptive coarse-to-fine narrowing, while `zoom_out` and `shift` provide dedicated mechanisms for fine-to-coarse backtracking and self-correction.

Training the agent to use these primitives effectively requires supervision not only for successful navigation but also for recovery from mistakes. Such supervision is not available in existing datasets [53, 48, 56]. We therefore develop a trajectory synthesis pipeline that produces both correct-path trajectories and trajectories that deliberately enter incorrect branches before recovering through ascending and lateral moves. We use these trajectories for supervised fine-tuning, followed by reinforcement learning with outcome-based rewards on grounding and answer accuracy.

We evaluate VTS on three Grounded LVQA benchmarks (CG-Bench [5], Haystack-LVBench [50], Haystack-Ego4D [50]) and three general long-video understanding benchmarks (Video-MME [12], MLVU [60], and LVBench [40]). VTS substantially outperforms prior multi-turn cropping agents across all three Grounded LVQA benchmarks. Although VTS is trained primarily for evidence localization, its navigation policy also transfers to general long-video QA. VTS surpasses the strongest prior multi-turn cropping agents on Video-MME, MLVU, and LVBench by up to $+7.1$ accuracy points, indicating that tree-grounded search is a useful inductive bias beyond temporal grounding alone. Inspecting the learned policy, we find that VTS averages 4.8 search turns per video on CG-Bench and invokes a backtracking action in roughly 60% of trajectories across all three benchmarks, indicating that the agent actively relies on its new primitives.

2 Related Work

Long video understanding and grounded long-video question answering. Long video understanding requires reasoning over videos that span minutes to hours, in which sparse query-relevant cues are buried in mostly redundant content [12, 40, 60, 45]. Most prior work focuses on the answer side. Video-specific MLLMs process long inputs end-to-end through context extension [8, 57, 30, 6], token compression [33, 41, 23, 35, 31], keyframe selection [37, 49, 27, 3], or state-space backbones [17, 18, 20] while captioner-LLM pipelines aggregate clip-level descriptions through a strong language model [54, 55, 21, 29]. These methods process the video in a single pass and do not localize the supporting evidence. We focus on Grounded LVQA [5, 50, 51], a more demanding setting in which the model must produce both an answer and the evidence interval that supports it. The grounding requirement makes predictions verifiable and prevents models from relying on language priors to answer correctly without attending to the relevant evidence. Recent benchmarks such as CG-Bench [5], Haystack-LVBench, and Haystack-Ego4D [50] couple grounding with QA, and recent methods including ReVisionLLM [15], TimeZero [42], VideoChat-R1 [22], VideoITG [39], and TimeChat [30] train VLMs with temporal awareness for fine-grained localization. These methods make a single forward pass or follow a fixed recursive pipeline. We instead cast Grounded LVQA as iterative hierarchical search through a tree of semantically coherent segments, where the agent learns to control its descent across temporal scales and to recover from incorrect choices.

Hierarchical representations of long videos. A complementary line of work organizes long videos into hierarchical structures for multi-scale reasoning. VideoTree [43] clusters dense video features and uses an LLM to organize frames into a query-relevant tree, and VideoMiner [4] groups frames into a tree and learns keyframe selection via reinforcement learning. Related approaches include Video ReCap [19], which produces captions at multiple temporal scales through recursive aggregation and ReVisionLLM [15], which performs coarse-to-fine localization on a fixed schedule. Our framework builds on this hierarchical inductive bias but differs in two key respects. Prior methods consume the hierarchy as a static representation in a single forward pass or fixed recursion schedule, whereas we treat the tree as an interactive environment that an agent navigates over multiple turns, learning when and how to descend. Our agent can also re-enter previously visited nodes through ascending and lateral moves, providing explicit self-correction that fixed recursive pipelines lack.

Temporal video segmentation. Temporal video segmentation aims to partition a video into coherent temporal units. One line of work focuses on shot boundary detection, which detects editing transitions such as hard cuts and gradual transitions between consecutive shots [36, 61]. Recently, OmniShotCut [38] formulates shot boundary detection as relational prediction with a shot-query Transformer. Another line of work groups shots into semantically coherent scenes [7, 46, 18], or detects generic event boundaries that humans naturally perceive [34, 59, 16]. However, these methods produce a single flat partition of the video at a fixed granularity and are optimized for boundary accuracy. Unlike these methods, our adaptive temporal segmentation recursively partitions any segment selected by the agent to build a multi-scale tree for search, while the final `answer` action predicts the precise evidence interval.

3 Method

We present VTS, a framework that frames Grounded LVQA as iterative hierarchical search over an adaptive temporal tree. An overview of the method is shown in Figure 2. Given a long video V and

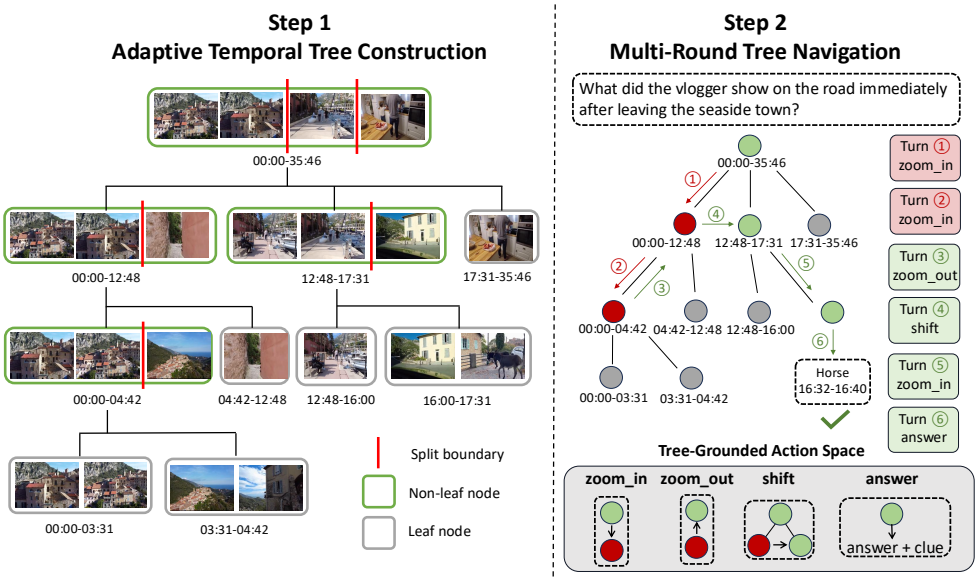


Figure 2: **Overview of VideoTreeSearch (VTS). Left**: Given a long video, VTS recursively partitions it into a non-uniform tree using adaptive temporal segmentation, so each node corresponds to a semantically coherent segment. **Right**: The agent navigates the tree through four discrete actions—`zoom_in` to descend into a child, `zoom_out` to ascend to the parent, `shift` to move laterally to a sibling, and `answer` to terminate with a final answer and evidence interval. In this example, the agent first descends into a wrong branch (red, turns 1–2), recovers via `zoom_out` and `shift` (turns 3–4), then zooms into the correct branch (turn 5) and answers with the localized clue at 16:32–16:40 (turn 6). Backtracking is thus an explicit, learnable primitive.

a question Q, VTS must produce both a final answer $\hat{y}$ and the evidence interval $[t_s, t_e] \subseteq [0, |V|]$ that supports it. The evidence interval is typically only seconds to tens of seconds wide, while the surrounding content is irrelevant. VTS constructs a non-uniform temporal tree whose nodes correspond to semantically coherent video segments (§3.1). An agent then navigates this tree through four discrete actions that expose hierarchical descent and self-correction as explicit moves (§3.2). A trajectory synthesis pipeline produces navigation paths with deliberate wrong-branch detours followed by recovery (§3.4), and these trajectories are used for supervised fine-tuning followed by reinforcement learning (§3.5).

3.1 Adaptive Temporal Tree Construction

The temporal tree exposes the multi-scale structure of the video to the agent. The root corresponds to the entire video, intermediate nodes correspond to coarse-grained segments, and leaves correspond to fine-grained candidates for the evidence interval. Children of a node form a non-overlapping partition of their parent's interval, so descending the tree progressively narrows the agent's hypothesis. Rather than imposing a fixed branching factor or uniform splits, we construct each node's children adaptively, placing boundaries where the visual content changes most. This design ensures that descent through the tree corresponds to refinement along natural content boundaries rather than arbitrary partitions.

We refer to this procedure as adaptive temporal segmentation. Concretely, given a segment $S = [t_{\text{start}}, t_{\text{end}}]$, we sample up to 64 frames uniformly from S at 1 fps and compute a CLIP [28] embedding e_i for each sampled frame. We compute the cosine distance $\delta_i = 1 - \cos(e_i, e_{i+1})$ between consecutive embeddings as a measure of visual and semantic change, and place boundaries at frames where this distance exceeds an adaptive threshold $\tau = \text{mean}(\delta) + k \cdot \text{std}(\delta)$, using the later frame of each transition as the boundary. We constrain the number of children to between 3 and 8, retaining the top boundaries when the threshold yields too many splits and relaxing the threshold when it yields too few. We stop splitting when the current node is shorter than 64 seconds, so nodes below this duration become leaves of the tree. The tree is constructed lazily during inference: only the root and the children of nodes that the agent actually visits are materialized. This keeps the navigation space small and ensures that segmentation reflects the regions the agent has chosen to inspect.

3.2 Tree-Grounded Action Space

At each turn, the agent occupies a node of the tree and selects one of four discrete actions:

- `zoom_in(`c`)`: descend into the c-th child of the current node.
- `zoom_out()`: ascend to the parent of the current node.
- `shift(`s`)`: move laterally to the s-th sibling under the same parent.
- `answer(`$\hat{y}, [t_s, t_e]$`)`: terminate with a final answer and evidence interval.

A single `zoom_in` action descends one level rather than committing to a specific timestamp range, so reaching a fine-grained evidence interval naturally takes multiple turns of progressive narrowing. Exploration and self-correction are decoupled: `zoom_in` advances the search downward, while `zoom_out` and `shift` are dedicated to self-correction. The agent can learn to invoke backtracking explicitly, rather than approximating it through repeated continuous cropping [48, 26, 53].

3.3 Multi-Round Tree Navigation

At each turn, the agent materializes the children of the current node, receives an observation summarizing the current node and its children, generates a reasoning trace, and selects one of the four actions. The loop terminates when the agent emits `answer`, reaches the maximum number of turns, or falls into a navigation cycle (Appendix A). To support long search horizons within a fixed context budget, we maintain a compact memory comprising (i) the current state of the materialized tree, where each visited node is annotated with a short caption and a visited flag, and (ii) a chronological log of all actions taken. We use the same VTS model for both navigation and caption generation through two separate calls. The first call generates the reasoning trace and navigation action conditioned on the question, sampled frames, and memory. The second call generates short captions describing the child segments. These captions are cached, so previously processed segments do not need to be captioned again. The agent receives raw frames only for the current node, while all other nodes are represented by their corresponding textual descriptions. This separation of detailed perception from compressed history lets the agent reason over many turns without exhausting the context budget. Detailed memory structures, caption generation procedures, and prompts are provided in Appendix A.

3.4 Trajectory Synthesis for Hierarchical Search

Optimal-descent trajectories alone do not teach an agent to recognize and recover from mistakes. We therefore synthesize trajectories that combine correct hierarchical descents with deliberate wrong-branch detours followed by recovery. Given a video, question, and ground-truth evidence interval, a controller selects `zoom_in` actions toward the highest-scoring child of the current node (scored by Qwen3-VL-8B for question relevance). When the selected child does not contain the ground-truth interval, the controller may either recover, by applying the inverse of the action sequence that led into the wrong branch, or continue further into the wrong branch to produce a longer detour. Recovery is forced after two consecutive turns in a wrong branch. Each action is paired with a reasoning trace generated by DeepSeek-R1 [9], conditioned on the memory at that turn, the current node's frames, and the chosen action. The trajectories are derived from CG-Bench, Haystack-Ego4D, and our automatically generated LongClueQA corpus, which leverages long unlabeled YouTube videos. We focus on examples with a single ground-truth clue spanning at most 20% of the video duration. Full filtering and trajectory generation details are in the Appendix.

3.5 Training

Supervised fine-tuning. We first fine-tune the agent to imitate the generated trajectories. Each trajectory $\mathcal{T} = \{(O_0, M_0), (R_1, A_1, O_1, M_1), \ldots, (R_T, A_T)\}$ is decomposed into per-turn supervision examples, where at turn t the input is the previous observation and current memory (O_{t-1}, M_t) and the target is the reasoning trace and action (R_t, A_t). Let $\mathcal{D}_{\text{SFT}}$ denote the resulting set of supervised turns and θ the model parameters. We optimize the standard token-level negative log-likelihood $\mathcal{L}_{\text{SFT}} = -\sum_{\mathcal{D}_{\text{SFT}}} \log p_\theta(R_t, A_t \mid O_{t-1}, M_t, Q)$. Crucially, we apply the loss selectively: although every trajectory is generated as an end-to-end sequence (including the deliberate wrong-branch detours that set up subsequent recovery turns), we backpropagate only through turns whose actions are themselves desirable as supervision — correct `zoom_in`, `zoom_out` from a wrong node, and

`shift` from a wrong node to the correct sibling. Turns whose actions are intentional mistakes (e.g., the `zoom_in` into a wrong node that sets up a later recovery) remain in the model's input context as part of the trajectory history but are masked out of the loss. The agent, therefore, observes the full detour-and-recovery sequence as context but is supervised only on the moves we want it to imitate.

Reinforcement learning. We further refine the policy beyond the supervised checkpoint with reinforcement learning. Each rollout produces a complete trajectory τ that ends in an answer, and we score it with a weighted sum of three outcome-based rewards on grounding and answer correctness:

$$R(\tau) = \lambda_{\text{fmt}} R_{\text{fmt}}(\tau) + \lambda_{\text{IoU}} R_{\text{IoU}}(\tau) + \lambda_{\text{acc}} R_{\text{acc}}(\tau). \tag{1}$$

Here, R_{fmt} checks that all actions are syntactically well-formed; R_{IoU} measures the temporal overlap between the predicted and ground-truth evidence intervals; R_{acc} is the answer accuracy reward. We optimize $R(\tau)$ with GRPO [32] regularized by a KL penalty against the supervised checkpoint to prevent the policy from drifting too far from the supervised initialization. The exact reward weights and the full optimization setup are described in the Appendix.

3.6 Implementation Details

We instantiate VTS with Qwen3-VL-8B [1] as the base model. The adaptive segmentation procedure of §3.1 uses $k = 1.5$. The agent samples 64 frames per turn at 1 fps. The trajectory synthesis pipeline yields $5{,}966$ filtered training trajectories drawn from three sources: CG-Bench [5], Haystack-Ego4D [50], and **LongClueQA**, a corpus of timestamped question-clue pairs that we automatically generate from unlabeled long-form YouTube videos (described in the Appendix). Supervised fine-tuning and reinforcement learning are both performed on $4\times$H100 GPUs. We refer to the supervised checkpoint as VTS-SFT and the reinforcement-learning checkpoint as VTS-RL. VTS-RL is our primary model in the experiments. Full hyperparameters are reported in the appendix.

4 Experimental Setup

Benchmarks and metrics. We evaluate VTS on three Grounded LVQA benchmarks and three general long-video QA benchmarks. For Grounded LVQA, we use the mini set of CG-Bench [5] along with Haystack-LVBench [50] and Haystack-Ego4D [50]. We apply a filtering procedure for CG-Bench (described in the Appendix) to remove instances with unannotated or ambiguous evidence intervals. CG-Bench reports mIoU between predicted and ground-truth evidence intervals together with QA accuracy. The Haystack benchmarks report Temporal F1 (T-F1, the main grounding metric) along with QA accuracy. For general long-video QA, we evaluate on Video-MME [12], the test split of MLVU [60], and LVBench [40], all measured by multiple-choice answer accuracy.

Baselines. We compare VTS against three families of methods. *Uniform sampling* applies a vision-language model directly to uniformly sampled frames, with no agentic interaction. We evaluate Qwen3-VL-8B [1], the same base model as VTS, at 256 and 384 frames, and Video-R1 [11] at its optimal 64-frame setting. *Agents with proprietary LLMs* rely on proprietary or similarly large LLMs for reasoning. We compare against SiLVR [55], which uses DeepSeek-R1 [9] as the reasoner over densely extracted visual captions, DVD [58], which uses o3-mini [24] to search over a multi-granular video database, and AVP [44], which uses Gemini-3-Flash [13] to iteratively seek evidence from the video. *Multi-turn cropping agents* localize the evidence interval through continuous `crop_video(start, end)` actions, and include methods such as LongVT [48], Video-o3 [53], VideoZoomer [10], and TimeSearch-R [26].

5 Experimental Results

5.1 Main Results on Grounded Long-Video Question Answering

Table 1 reports results on the three Grounded LVQA benchmarks. VTS substantially outperforms every multi-turn cropping baseline. On CG-Bench, VTS reaches 16.8 mIoU and 36.4 accuracy, an absolute improvement of $+12.5$ mIoU and $+10.3$ accuracy over the strongest cropping baselines LongVT and Video-o3, respectively. On Haystack-LVBench, it reaches 15.2 T-F1, $+7.1$ over TimeSearch-R. On Haystack-Ego4D, it reaches 18.4 T-F1, $+7.4$ over the strongest multi-turn cropping

Table 1: **Grounded LVQA results.** VTS substantially improves both grounding metrics (mIoU, T-F1) and QA accuracy over prior multi-turn cropping agents across all benchmarks.

	CG-Bench		Haystack-LVBench		Haystack-Ego4D	
Method	mIoU	Acc	T-F1	Acc	T-F1	Acc
Uniform sampling						
Qwen3-VL-8B (256 frames) [1]	12.1	20.2	12.8	45.9	8.9	39.2
Qwen3-VL-8B (384 frames) [1]	12.1	23.6	9.2	39.5	7.0	33.4
Video-R1 (64 frames) [11]	5.5	15.8	4.3	40.6	2.9	33.1
Agents with proprietary LLMs						
SiLVR [55]	10.1	31.8	8.5	57.0	6.3	46.4
DVD [58]	13.5	35.0	–	–	–	–
AVP [44]	16.5	**54.2**	10.1	**76.1**	16.3	**70.0**
Multi-turn cropping agents						
LongVT [48]	4.3	17.4	5.4	46.2	3.7	42.1
Video-o3 [53]	2.9	26.1	–	–	–	–
VideoZoomer [10]	3.0	18.3	6.8	48.3	5.4	48.6
TimeSearch-R [26]	–	–	8.1	52.1	11.0	53.5
VTS (Ours)	**16.8**	36.4	**15.2**	58.3	**18.4**	66.1

baseline. VTS averages 4.8 turns per video on CG-Bench (Table 8 in the Appendix), whereas prior multi-turn cropping methods converge in only 1-2 turns, confirming that the cropping agents commit to an interval early while VTS continues to explore and self-correct over multiple turns. VTS also outperforms the agents with proprietary LLMs in grounding on all three benchmarks while using a smaller open-source backbone. On CG-Bench, VTS improves mIoU by $+6.7$, $+3.3$, and $+0.3$ over SiLVR, DVD, and AVP, respectively. AVP achieves higher QA accuracy than VTS, which we attribute to its larger proprietary backbone (Gemini-3-Flash).

In Table 2, we analyze the inference efficiency of VTS and compare it against Qwen3-VL-8B and four agentic methods (SiLVR, LongVT, DVD, and AVP). We also report a more efficient variant (VTS-fast), which uses at most five turns and generates each action and the corresponding captions in a single VLM call. From the table, we can see that VTS achieves the best mIoU while processing fewer frames than all baseline methods. With a comparable runtime to SiLVR (61.6s vs. 62.3s), VTS improves mIoU by 6.7 and accuracy by 4.6. Compared to DVD and AVP, VTS achieves higher mIoU while running $7.2\times$ and $1.6\times$ faster, respectively. Notably, VTS-fast outperforms Qwen3-VL-8B by 1.0 mIoU and 5.5 accuracy with a comparable runtime (26.5s vs. 25.1s). We also provide the runtime breakdown of VTS in Table 13 in the Appendix. These results show that VTS is both efficient and effective.

Table 2: **Efficiency analysis on CG-Bench.** VTS achieves the best mIoU and the best balance between performance and runtime. Runtime is measured per video on a single H100 GPU. †Uses a proprietary LLM.

Method	Frames	Time (s)	mIoU	Acc
Qwen3-VL-8B [1]	384	**25.1**	12.1	23.6
SiLVR [55]	450	62.3	10.1	31.8
LongVT [48]	512	28.4	4.3	17.4
DVD† [58]	1 fps	442.0	13.5	35.0
AVP† [44]	512	99.5	16.5	**54.2**
VTS-fast (Ours)	**192**	26.5	13.1	29.1
VTS (Ours)	328	61.6	**16.8**	36.4

5.2 Performance Gains Analysis

We attribute VTS's gains to three key components: interactive search over the tree, hierarchical search across multiple temporal scales, and discrete navigation actions that make recovery from wrong branches explicit. Table 3 investigates the contributions of these components.

Interactive search outperforms static tree representations. To examine the benefits of interactive tree navigation, we evaluate a static full-tree baseline inspired by VideoTree [43]. The baseline provides Qwen3-VL-8B with captions of all tree nodes generated by Qwen3-VL-8B itself, and predicts the answer and evidence in a single pass, without iterative search. As shown in Table 3, VTS outperforms this baseline by 6.1–7.1 points on temporal grounding metrics and 10.1–22.4 points on QA accuracy across all three benchmarks, demonstrating the effectiveness of interactive tree navigation over static hierarchical representations.

Table 3: **Performance gains analysis on Grounded LVQA.** Interactive search, the tree-grounded action space, the backtracking primitives, and hierarchical descent each contribute to the strong performance of VTS.

	CG-Bench		Haystack-LVBench		Haystack-Ego4D	
Variant	mIoU	Acc	T-F1	Acc	T-F1	Acc
Static full tree (no interactive search)	9.7	18.1	9.1	48.2	12.1	43.7
Continuous `crop_video` action	15.5	32.5	13.2	49.1	14.2	56.3
Tree-grounded, no backtracking	14.4	35.0	13.2	52.9	15.5	64.6
Tree-grounded, flat tree (no hierarchy)	15.3	34.5	13.2	53.2	**18.9**	62.9
VTS (full design)	**16.8**	**36.4**	**15.2**	**58.3**	18.4	**66.1**

Tree-grounded actions outperform continuous cropping. Replacing the tree-grounded actions with a continuous `crop_video` action under the same data and recipe degrades performance by 1.3–4.2 points on temporal grounding metrics and by up to 9.8 points on QA accuracy, indicating that the tree-grounded action space is more effective than continuous cropping.

Self-correction is critical. Removing the backtracking primitives `zoom_out` and `shift` degrades performance on every benchmark. Specifically, temporal grounding drops by 2.0–2.9 points on all datasets while QA accuracy drops by 1.4–5.4 points, indicating that explicit backtracking lets the agent recover from wrong descents, thereby improving both temporal grounding and answer accuracy. Additionally, we count any `zoom_out` or `shift` in a VTS trajectory as a self-correction event, and any continuous-cropping turn whose predicted interval has IoU < 0.2 with the previous turn's interval as the analogous event. VTS self-corrects in roughly 60% of trajectories on all three benchmarks, while the continuous-cropping baseline trained on identical data self-corrects in only 7-15$\%$. Among self-correcting trajectories, 42.7% subsequently reach a node containing the ground-truth evidence, indicating that the agent recovers on a substantial fraction of these harder cases.

Hierarchical search supports a wide range of clue durations. Replacing the hierarchical tree with a flat tree of adaptive short segments degrades QA accuracy by 1.9-5.1 points on all datasets. The flat tree achieves higher temporal grounding than the hierarchical tree only on Haystack-Ego4D (18.9 vs. 18.4 T-F1), which has the shortest average clue duration (9.5s). On the other two benchmarks, clue durations vary widely across queries, from a few seconds to minutes (mean 19.2s on CG-Bench and 96.2s on Haystack-LVBench), so no single flat partition matches them all. Hierarchical search lets the agent navigate to the appropriate scale for each query.

5.3 Ablation Studies

We isolate the contribution of VTS's design choices and examine how the framework generalizes across training data sources and backbones. More ablation studies are provided in the Appendix.

Tree construction. In Table 4, we compare our adaptive temporal segmentation with two alternative tree construction strategies: uniform splitting, which divides each segment into a fixed number of equal-length children, and OmniShotCut [38], a recent shot boundary detection method. From the table, we can see that adaptive temporal segmentation outperforms uniform splitting with 4 and 16 children by 1.2 and 2.9 mIoU, and OmniShotCut by 2.8 mIoU and 3.9 accuracy. These results demonstrate that adaptive temporal segmentation constructs a more effective search tree than both uniform splitting and shot boundary detection.

Training stages. VTS is trained in two stages, and Table 4 shows that each stage helps. The zero-shot variant, using only the action space

Table 4: **Ablation studies on CG-Bench.** Each ablation varies one component and keeps the rest at their default setting.

Variant	mIoU	Acc
Tree construction		
Uniform splitting, 4 children	15.6	34.9
Uniform splitting, 16 children	13.9	33.0
OmniShotCut [38] shot boundaries	14.0	32.5
Adaptive temporal segmentation (Ours)	**16.8**	**36.4**
Training stages		
Zero-shot (action space + prompting)	12.8	20.9
VTS-SFT (+ synthesized trajectories)	14.9	30.7
VTS-RL (+ reinforcement learning)	**16.8**	**36.4**
Trajectory synthesis		
Optimal-path-only trajectories	10.4	31.3
Detour-and-recovery trajectories (default)	**16.8**	**36.4**

Table 5: **Effect of training data source and backbone.** We compare training VTS on LongClueQA alone, our corpus of synthetic timestamped QA from unlabeled long-form videos, against training on all sources, which adds in-domain videos from CG-Bench and Haystack-Ego4D. We also train VTS on Qwen2.5-VL-7B, the backbone used by prior multi-turn cropping agents. The reference row reports the strongest prior multi-turn cropping baseline per metric (LongVT for CG-Bench mIoU, Video-o3 for CG-Bench accuracy, TimeSearch-R for Haystack-LVBench and Haystack-Ego4D).

		CG-Bench		Haystack-LVBench		Haystack-Ego4D	
Training data source	Backbone	mIoU	Acc	T-F1	Acc	T-F1	Acc
LongClueQA only	Qwen3-VL-8B	15.3	28.7	14.5	57.5	16.8	53.7
All sources	Qwen2.5-VL-7B	14.5	30.7	12.3	53.1	13.1	55.3
All sources	Qwen3-VL-8B	**16.8**	**36.4**	**15.2**	**58.3**	**18.4**	**66.1**
Reference: strongest cropping baseline	Qwen2.5-VL-7B	4.3	26.1	8.1	52.1	11.0	53.5

and prompting with no training, reaches 12.8 mIoU. SFT on synthesized trajectories adds $+2.1$ mIoU, and RL adds a further $+1.9$.

Trajectory synthesis. Table 4 shows that training on trajectories with deliberate detours and recovery outperforms training on optimal-path-only trajectories by $+6.4$ mIoU and $+5.1$ accuracy. This shows that the agent learns to search effectively from exposure to recovery in its training trajectories.

Effect of training data source. We compare training on LongClueQA alone, generated entirely from unlabeled long-form videos, against training on all sources. Note that LongClueQA has no video overlap with any evaluation benchmark. As shown in Table 5, VTS trained on LongClueQA alone already surpasses the strongest cropping baseline on each benchmark, by $+11.0$ mIoU on CG-Bench, $+6.4$ T-F1 on Haystack-LVBench, and $+5.8$ T-F1 on Haystack-Ego4D. Adding in-domain training videos from CG-Bench and Haystack-Ego4D (the all-sources variant) improves grounding further by 1.5, 0.7, and 1.6 points, respectively. The results suggest that synthetic timestamped QA from unlabeled videos is a strong training signal on its own.

Backbone generalization. To check that VTS's gains do not depend on the Qwen3-VL-8B backbone, we also train VTS on Qwen2.5-VL-7B [2], the backbone used by prior multi-turn cropping agents. Under this matched-backbone setting (Table 5), VTS improves over the strongest Qwen2.5-VL-7B cropping baseline by $+10.2$ mIoU and $+4.6$ accuracy on CG-Bench, $+4.2$ T-F1 and $+1.0$ accuracy on Haystack-LVBench, and $+2.1$ T-F1 and $+1.8$ accuracy on Haystack-Ego4D. The gains therefore stem from the framework rather than the base model, and a stronger backbone adds further gains.

5.4 General Long-Video Understanding

We test whether VTS's tree-grounded navigation policy generalizes beyond Grounded LVQA on three general long-video QA benchmarks: Video-MME, MLVU, and LVBench. These benchmarks evaluate QA accuracy alone, with no temporal grounding, so when the agent answers, VTS pools the frames it observed across all visited nodes and conditions the answer on this pooled set rather than on a single node. Table 6 reports the results. VTS outperforms every multi-turn cropping baseline on all three benchmarks: $+0.5$ on Video-MME over LongVT, $+2.4$ on MLVU over VideoZoomer, and $+7.1$ on LVBench over Video-o3. The tree-grounded navigation policy thus generalizes beyond evidence localization, despite being trained primarily for Grounded LVQA.

Table 6: **General long-video understanding.** Accuracy (%) on Video-MME, MLVU, and LVBench, with frame counts for VTS averaged across the three. All MLVU results are on the test split. *Reproduced on the MLVU test split using the official Video-o3 checkpoint.

Method	Frames	V-MME	MLVU	LVB
LongVT [48]	512–768	67.0	–	41.3
Video-o3 [53]	$\leq$768	66.5	53.1*	47.6
TimeSearch-R [26]	768	66.6	–	–
VideoZoomer [10]	128	65.2	55.8	41.5
VTS (Ours)	274	**67.5**	**58.2**	**54.7**

5.5 Qualitative Analysis

Figure 3 shows a six-turn trajectory on a question asking what is visible through the 22nd-floor glass window. On the third turn the agent commits to a wrong branch, descending into a segment showing the building's ground-floor ticket booth. Recognizing that the ticket booth is not the queried

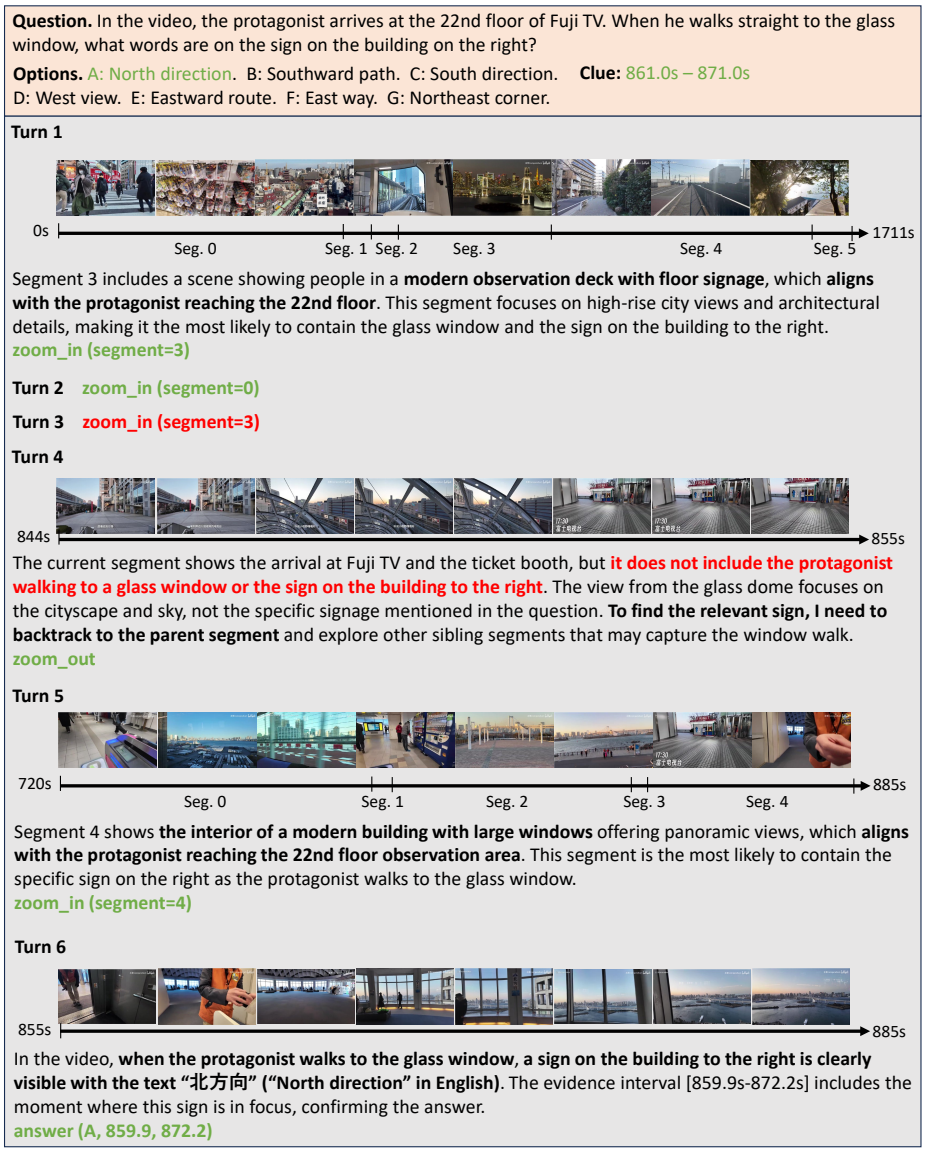


Figure 3: **Qualitative example: recovery via `zoom_out`.** For a question about what is visible through a 22nd-floor window of the Fuji TV building, the agent first descends into a wrong branch showing the building's ground-floor ticket booth (turns 1–3). It detects the error, issues `zoom_out` to return to the parent (turn 4), and explores a different child to reach the correct clue (turns 5–6). The example shows `zoom_out` as an explicit recovery action after the agent commits to a wrong branch.

22nd-floor window, the agent issues `zoom_out` to return to the parent, then re-descends through a different child to reach the correct clue. Additional qualitative examples are included in the Appendix.

6 Conclusion

We presented VTS, a tree-grounded agentic framework for grounded long-video question-answering. VTS organizes the video as a tree of semantically coherent segments and navigates it with discrete actions, which turns precise localization and backtracking from implicit capabilities into explicitly learnable behaviors. On three grounded long-video QA benchmarks, VTS substantially outperforms prior multi-turn cropping agents and achieves better grounding than agents with proprietary LLMs while processing fewer frames, and it transfers to three general long-video understanding benchmarks. More broadly, our results suggest that structuring the search over a long video is a useful inductive bias for long-horizon video reasoning.

Acknowledgements

This work was supported by Laboratory for Analytic Sciences via NC State University, ONR Award N00014-23-1-2356, Sony Focused Research award, and NSF CAREER Award 2541848.

References

[1] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[2] 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*, 2025.

[3] Shyamal Buch, Cristobal Eyzaguirre, Adrien Gaidon, Jiajun Wu, Li Fei-Fei, and Juan Carlos Niebles. Revisiting the "video" in video-language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] Xinye Cao, Hongcan Guo, Jiawen Qian, Guoshun Nan, Chao Wang, Yuqi Pan, Tianhao Hou, Xiaojuan Wang, and Yutong Gao. VideoMiner: Iteratively grounding key frames of hour-long videos via tree-based group relative policy optimization. *arXiv preprint arXiv:2510.06040*, 2025.

[5] Guo Chen et al. CG-Bench: Clue-grounded question answering benchmark for long video understanding. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2025.

[6] Lin Chen, Xilin Wei, Jinsong Li, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Zehui Chen, Haodong Duan, Bin Lin, Zhenyu Tang, et al. ShareGPT4Video: Improving video understanding and generation with better captions. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[7] Shixing Chen, Xiaohan Nie, David Fan, Dongqing Zhang, Vimal Bhat, and Raffay Hamid. Shot contrastive self-supervised learning for scene boundary detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9796–9805, 2021.

[8] Yukang Chen, Fuzhao Xue, Dacheng Li, Qinghao Hu, Ligeng Zhu, Xiuyu Li, Yunhao Fang, Haotian Tang, Shang Yang, Zhijian Liu, et al. LongVILA: Scaling long-context visual language models for long videos. *arXiv preprint arXiv:2408.10188*, 2024.

[9] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[10] Yang Ding, Yizhen Zhang, Xin Lai, Ruihang Chu, and Yujiu Yang. VideoZoomer: Reinforcement-learned temporal focusing for long video reasoning. *arXiv preprint arXiv:2512.22315*, 2025.

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

[12] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal LLMs in video analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[13] Google DeepMind. Gemini 3 Flash model card. `https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Flash-Model-Card.pdf`, December 2025.

[14] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18995–19012, 2022.

[15] Tanveer Hannan, Md Mohaiminul Islam, Jindong Gu, Thomas Seidl, and Gedas Bertasius. ReVisionLLM: Recursive vision-language model for temporal grounding in hour-long videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[16] Jaejun Hwang, Dayoung Gong, Manjin Kim, and Minsu Cho. Generic event boundary detection via denoising diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14084–14094, 2025.

[17] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. In *European Conference on Computer Vision (ECCV)*, 2022.

[18] Md Mohaiminul Islam, Mahmudul Hasan, Kishan Shamsundar Athrey, Tony Braskich, and Gedas Bertasius. Efficient movie scene detection using state-space transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[19] Md Mohaiminul Islam, Ngan Ho, Xitong Yang, Tushar Nagarajan, Lorenzo Torresani, and Gedas Bertasius. Video ReCap: Recursive captioning of hour-long videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[20] Md Mohaiminul Islam, Tushar Nagarajan, Huiyu Wang, Gedas Bertasius, and Lorenzo Torresani. BIMBA: Selective-scan compression for long-range video question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[21] Kumara Kahatapitiya, Kanchana Ranasinghe, Jongwoo Park, and Michael S. Ryoo. Language repository for long video understanding. In *Findings of the Association for Computational Linguistics (ACL Findings)*, 2025.

[22] Xinhao Li, Ziang Yan, Desen Meng, Lu Dong, Xiangyu Zeng, Yinan He, Yali Wang, Yu Qiao, Yi Wang, and Limin Wang. VideoChat-R1: Enhancing spatio-temporal perception via reinforcement fine-tuning. *arXiv preprint arXiv:2504.06958*, 2025.

[23] Yanwei Li, Chengyao Wang, and Jiaya Jia. LLaMA-VID: An image is worth 2 tokens in large language models. *European Conference on Computer Vision (ECCV)*, 2024.

[24] OpenAI. OpenAI o3-mini system card. `https://cdn.openai.com/o3-mini-system-card-feb10.pdf`, 2025.

[25] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

[26] Junwen Pan, Qizhe Zhang, Rui Zhang, Ming Lu, Xin Wan, Yuan Zhang, Chang Liu, and Qi She. TimeSearch-R: Adaptive temporal search for long-form video understanding via self-verification reinforcement learning. *arXiv preprint arXiv:2511.05489*, 2025.

[27] Jongwoo Park, Kanchana Ranasinghe, Kumara Kahatapitiya, Wonjeong Ryu, Donghyun Kim, and Michael S. Ryoo. Too many frames, not all useful: Efficient strategies for long-form video QA. *arXiv preprint arXiv:2406.09396*, 2025.

[28] Alec Radford et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[29] Kanchana Ranasinghe, Xiang Li, Kumara Kahatapitiya, and Michael Ryoo. Understanding long videos in one multimodal language model pass. In *International Conference on Learning Representations (ICLR)*, 2025.

[30] Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. TimeChat: A time-sensitive multimodal large language model for long video understanding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[31] Michael S. Ryoo, Honglu Zhou, Shrikant Kendre, Can Qin, Le Xue, Manli Shu, Jongwoo Park, Kanchana Ranasinghe, Silvio Savarese, Ran Xu, Caiming Xiong, and Juan Carlos Niebles. xgen-MM-vid (BLIP-3-video): You only need 32 tokens to represent a video even in VLMs. *arXiv preprint arXiv:2410.16267*, 2025.

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

[33] Xiaoqian Shen, Yunyang Xiong, Changsheng Zhao, Lemeng Wu, Jun Chen, Chenchen Zhu, Zechun Liu, Fanyi Xiao, Balakrishnan Varadarajan, Florian Bordes, et al. LongVU: Spatiotemporal adaptive compression for long video-language understanding. *arXiv preprint arXiv:2410.17434*, 2024.

[34] Mike Zheng Shou, Stan Weixian Lei, Weiyao Wang, Deepti Ghadiyaram, and Matt Feiszli. Generic event boundary detection: A benchmark for event segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8075–8084, 2021.

[35] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, et al. MovieChat: From dense token to sparse memory for long video understanding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[36] Tomáš Souček and Jakub Lokoč. Transnet v2: An effective deep network architecture for fast shot transition detection. *arXiv preprint arXiv:2008.04838*, 2020.

[37] Xi Tang, Jihao Qiu, Lingxi Xie, Yunjie Tian, Jianbin Jiao, and Qixiang Ye. Adaptive keyframe sampling for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[38] Boyang Wang, Guangyi Xu, Jiahui Zhang, Zhipeng Tang, and Zezhou Cheng. Omnishotcut: Holistic relational shot boundary detection with shot-query transformer. *arXiv preprint arXiv:2604.24762*, 2026.

[39] Shihao Wang, Guo Chen, De-an Huang, Zhiqi Li, Minghan Li, Guilin Li, Jose M. Alvarez, Lei Zhang, and Zhiding Yu. VideoITG: Multimodal video understanding with instructed temporal grounding. *arXiv preprint arXiv:2507.13353*, 2025.

[40] 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. *arXiv preprint arXiv:2406.08035*, 2024.

[41] Xiao Wang, Qingyi Si, Jianlong Wu, Shiyu Zhu, Li Cao, and Liqiang Nie. AdaReTaKe: Adaptive redundancy reduction to perceive longer for video-language understanding. *arXiv preprint arXiv:2503.12559*, 2025.

[42] Ye Wang, Boshen Xu, Zihao Yue, Zihan Xiao, Ziheng Wang, Liang Zhang, Dingyi Yang, Wenxuan Wang, and Qin Jin. TimeZero: Temporal video grounding with reasoning-guided LVLM. *arXiv preprint arXiv:2503.13377*, 2025.

[43] 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[44] Ziyang Wang, Honglu Zhou, Shijie Wang, Junnan Li, Caiming Xiong, Silvio Savarese, Mohit Bansal, Michael S. Ryoo, and Juan Carlos Niebles. Active video perception: Iterative evidence seeking for agentic long video understanding. *arXiv preprint arXiv:2512.05774*, 2025.

[45] Haoning Wu, Dongxu Li, Bei Chen, and Junnan Li. LongVideoBench: A benchmark for long-context interleaved video-language understanding. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[46] Haoqian Wu, Keyu Chen, Yanan Luo, Ruizhi Qiao, Bo Ren, Haozhe Liu, Weicheng Xie, and Linlin Shen. Scene consistency representation learning for video scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14021–14030, 2022.

[47] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[48] Zuhao Yang, Sudong Wang, Kaichen Zhang, Keming Wu, Sicong Leng, Yifan Zhang, Bo Li, Chengwei Qin, Shijian Lu, Xingxuan Li, and Lidong Bing. LongVT: Incentivizing "thinking with long videos" via native tool calling. *arXiv preprint arXiv:2511.20785*, 2025.

[49] Linli Yao, Haoning Wu, Kun Ouyang, Yuanxing Zhang, Caiming Xiong, Bei Chen, Xu Sun, and Junnan Li. Generative frame sampler for long video understanding. *arXiv preprint arXiv:2503.09146*, 2025.

[50] Jinhui Ye, Zihan Wang, Haosen Sun, Keshigeyan Chandrasegaran, Zane Durante, Cristobal Eyzaguirre, Yonatan Bisk, Juan Carlos Niebles, Ehsan Adeli, Li Fei-Fei, Jiajun Wu, and Manling Li. Re-thinking temporal search for long-form video understanding. *arXiv preprint arXiv:2504.02259*, 2025.

[51] Huaying Yuan, Jian Ni, Zheng Liu, Yueze Wang, Junjie Zhou, Zhengyang Liang, Bo Zhao, Zhao Cao, Zhicheng Dou, and Ji-Rong Wen. MomentSeeker: A benchmark for long-video moment retrieval. *arXiv preprint arXiv:2502.12558*, 2025.

[52] Rowan Zellers, Jiasen Lu, Ximing Lu, Youngjae Yu, Yanpeng Zhao, Mohammadreza Salehi, Aditya Kusupati, Jack Hessel, Ali Farhadi, and Yejin Choi. Merlot reserve: Neural script knowledge through vision and language and sound. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16375–16387, 2022.

[53] Xiangyu Zeng, Zhiqiu Zhang, Yuhan Zhu, Xinhao Li, Zikang Wang, Changlian Ma, Qingyu Zhang, Zizheng Huang, Kun Ouyang, Tianxiang Jiang, et al. Video-o3: Native interleaved clue seeking for long video multi-hop reasoning. *arXiv preprint arXiv:2601.23224*, 2026.

[54] Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. A simple LLM framework for long-range video question-answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[55] Ce Zhang, Yan-Bo Lin, Ziyang Wang, Mohit Bansal, and Gedas Bertasius. SiLVR: A simple language-based video reasoning framework. *arXiv preprint arXiv:2505.24869*, 2025.

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

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

[58] 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. In *Advances in Neural Information Processing Systems*, 2025.

[59] Ziwei Zheng, Lijun He, Le Yang, and Fan Li. Fine-grained dynamic network for generic event boundary detection. In *European Conference on Computer Vision (ECCV)*, 2024.

[60] Junjie Zhou, Yan Shu, Bo Zhao, Boya Wu, Shitao Xiao, Xi Yang, Yongping Xiong, Bo Zhang, Tiejun Huang, and Zheng Liu. MLVU: A comprehensive benchmark for multi-task long video understanding. *arXiv preprint arXiv:2406.04264*, 2024.

[61] Wentao Zhu, Yufang Huang, Xiufeng Xie, Wenxian Liu, Jincan Deng, Debing Zhang, Zhangyang Wang, and Ji Liu. Autoshot: A short video dataset and state-of-the-art shot boundary detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2238–2247, 2023.

Appendix

Our appendix consists of Additional Implementation Details (Section A), Evaluation Benchmarks (Section B), Data Construction (Section C), Additional Analyses (Section D), More Qualitative Results (Section E), and Limitations and Future Work (Section F).

A Additional Implementation Details

Training Details. We use full fine-tuning of the LLM for SFT and LoRA fine-tuning for RL. SFT is conducted with the LLaMA-Factory framework and RL is conducted with the ms-swift framework. For SFT, we set the learning rate to `1e-5`, the batch size to 16, and train for 2 epochs. For RL, we set the learning rate to `5e-5`, the batch size to 96, and train for 1 epoch with GRPO using a group of 8 trajectories per prompt. For each video segment, we sample frames at 1 fps with a maximum of 64 frames, and each frame is constrained to a maximum resolution of 224×224 pixels. Both SFT and RL are conducted on $4\times$ H100 GPUs, taking approximately 4 and 32 hours, respectively. Full hyperparameters are reported in Table 7.

Table 7: **Key training hyperparameters** for the SFT and RL stages of VTS.

(a) SFT stage

Hyperparameter	Value
Train epochs	`2`
Train batch size	`16`
Learning rate	`1e-5`
Learning rate scheduler	`cosine`
Warmup ratio	`0.1`
Freeze vision encoder	`true`

(b) RL stage

Hyperparameter	Value
Format reward weight	`0.5`
Accuracy reward weight	`0.5`
IoU reward weight	`1.0`
Max interaction turns	`10`
LoRA rank	`32`
LoRA alpha	`64`
LoRA dropout	`0`
Train batch size	`96`
Rollout temperature	`1.0`
Rollouts per prompt (n)	`8`
KL coefficient (β)	`0.04`
Learning rate	`5e-5`
Learning rate scheduler	`constant`

GRPO Objective. For each training question, we sample a group of $G = 8$ trajectories $\{\tau_i\}_{i=1}^{G}$ from the old policy $\pi_{\theta_{\text{old}}}$, each with at most 10 turns. Each trajectory receives the reward $R_i = 0.5\, R_{\text{fmt}}(\tau_i) + 1.0\, R_{\text{IoU}}(\tau_i) + 0.5\, R_{\text{acc}}(\tau_i)$, and its advantage is computed within the group as $\hat{A}_i = (R_i - \text{mean}(R_{1:G}))/\text{std}(R_{1:G})$. Let $a_{i,t}$ denote the t-th token generated in τ_i, $s_{i,t}$ its context, and $\rho_{i,t} = \pi_\theta(a_{i,t} \mid s_{i,t})/\pi_{\theta_{\text{old}}}(a_{i,t} \mid s_{i,t})$ the importance ratio. The loss is

$$\mathcal{L}_{\text{GRPO}} = -\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|\tau_i|}\sum_{t=1}^{|\tau_i|}\Big[\min\big(\rho_{i,t}\hat{A}_i,\ \text{clip}(\rho_{i,t}, 1-\varepsilon, 1+\varepsilon)\,\hat{A}_i\big) - \beta\, D_{\text{KL}}\big(\pi_\theta(\cdot \mid s_{i,t}) \,\|\, \pi_{\text{ref}}(\cdot \mid s_{i,t})\big)\Big], \tag{2}$$

where π_{ref} is the SFT policy, $\varepsilon = 0.2$, and $\beta = 0.04$.

Training Data. The training data is sourced from the CG-Bench full set (excluding questions from the mini set), the Haystack-Ego4D training set, and LongClueQA. In total, we have 5,966 QA pairs, and 5,966 trajectories from all data sources (1,144 from CG-Bench, 1,130 from Haystack-Ego4D, and 3,692 from LongClueQA). We randomly pick 40% of trajectories for SFT training and 60% of the QA pairs for RL training.

Adaptive Temporal Segmentation Algorithm. At each node of the search tree, we partition the current segment into child segments aligned with visual and semantic transitions. Algorithm 1

summarizes the procedure. We uniformly sample frames from the current segment, encode each with a CLIP image encoder, and compute the cosine distance between consecutive frame embeddings to obtain a per-position boundary signal δ. We then form an adaptive threshold $\tau = \text{mean}(\delta) + k \cdot \text{std}(\delta)$ from the local statistics of δ, and treat positions with $\delta_i > \tau$ as boundaries. We clip the boundary set so that the number of children falls within $[N_{\text{min}}, N_{\text{max}}]$: if the threshold yields too many boundaries we keep those with the largest δ_i, and if it yields too few we add the next-largest until the minimum is met. We set $N_{\text{min}} = 3$ and $N_{\text{max}} = 8$. For each selected transition δ_i, we place the boundary at the timestamp of the later frame $i{+}1$.

Algorithm 1 Adaptive Temporal Segmentation

Require: Segment $S = [t_{\text{start}}, t_{\text{end}}]$, CLIP encoder E, threshold coefficient k, min/max children $N_{\text{min}}, N_{\text{max}}$

$F \leftarrow \texttt{sampleFrames}(S)$ ▷ ≤ 64 frames at 1 fps
$\{e_i\}_{i=1}^{n} \leftarrow \texttt{encode}(F, E)$
$\delta_i \leftarrow 1 - \cos(e_i, e_{i+1}),$ for $i = 1, \ldots, n-1$
$\tau \leftarrow \text{mean}(\delta) + k \cdot \text{std}(\delta)$
$B \leftarrow \{\, i : \delta_i > \tau \,\}$
if $|B| + 1 > N_{\text{max}}$ **then**
    $B \leftarrow$ indices of the top $(N_{\text{max}} - 1)$ values in δ
else if $|B| + 1 < N_{\text{min}}$ **then**
    $B \leftarrow$ indices of the top $(N_{\text{min}} - 1)$ values in δ
end if
return $\texttt{splitByBoundaries}(S, B)$ ▷ cut S at the timestamp of frame $i{+}1$ for each $i \in B$

Hierarchical Tree Search Algorithm. Algorithm 2 describes the inference-time hierarchical search. The agent starts at the root segment covering the full video and maintains a memory $\mathcal{M}$ containing (i) the partial search tree expanded so far, with a caption for each visited node, and (ii) the interaction history of prior actions and their outcomes. At each turn, the agent splits the current segment into children via Algorithm 1 (or retrieves them from the cached tree), is shown frames sampled from those children together with $\mathcal{M}$, and outputs an action in $\{\texttt{zoom_in}, \texttt{zoom_out}, \texttt{shift}, \texttt{answer}\}$. `zoom_in` descends into a selected child, `zoom_out` returns to the parent, and `shift` moves to a sibling under the same parent. After the action is executed, the same model π generates a caption for each child segment in a separate call, and these captions are written into $\mathcal{M}$ together with the reasoning trace and the action. The loop terminates when the agent emits `answer`, returning the predicted answer $\hat{y}$ and evidence interval $[t_s, t_e]$. In practice, the algorithm also terminates when a navigation cycle is detected, i.e., when the agent returns to a previously visited node and selects the same action that it took at its previous visit to that node. Revisiting a node alone does not count as a cycle, so backtracking through `zoom_out` and `shift` is unaffected. When a cycle is detected, the agent answers via `forceAnswer` using the current memory.

Prompt Structure. We show the prompt structure used at every decision turn in the block below. The system prompt establishes the agent's role and enumerates the four discrete actions, while the per-turn user prompt provides frames from the current node's children, a textual memory block (tree structure, interaction history, and navigation context), and the question with its multiple-choice options.

Algorithm 2 Hierarchical Tree Search

Require: Video V, question Q, agent policy π, captioning prompt P_{cap}, max turns $T_{\max}$

$S \leftarrow [0, \text{len}(V)]$ ▷ current node
$\mathcal{M} \leftarrow \texttt{initMemory}()$ ▷ tree structure + interaction history
for $t = 1, \ldots, T_{\max}$ **do**
  $\{S_1, \ldots, S_m\} \leftarrow \texttt{AdaptiveTemporalSegmentation}(S)$ ▷ Algorithm 1
  $O \leftarrow \texttt{sampleFrames}(\{S_1, \ldots, S_m\})$
  $(R, A) \leftarrow \pi(O, \mathcal{M}, Q)$ ▷ reasoning and action
  if $A \neq \texttt{answer}$ and A was taken at S in a previous turn **then**
    return $\texttt{forceAnswer}(O, \mathcal{M}, Q)$ ▷ navigation cycle
  end if
  if $A = \texttt{zoom_in}(j)$ **then**
    $S \leftarrow S_j$
  else if $A = \texttt{zoom_out}$ **then**
    $S \leftarrow \texttt{parent}(S)$
  else if $A = \texttt{shift}(j)$ **then**
    $S \leftarrow$ sibling of S indexed by j
  else if $A = \texttt{answer}(\hat{y}, [t_s, t_e])$ **then**
    return $\hat{y}, [t_s, t_e]$
  end if
  $C \leftarrow \pi(O, P_{\text{cap}})$ ▷ caption each child segment with π
  $\mathcal{M} \leftarrow \texttt{updateMemory}(\mathcal{M}, C, R, A, S)$
end for
return $\texttt{forceAnswer}(O, \mathcal{M}, Q)$ ▷ if turn budget exhausted

VTS Agent Prompt

System Prompt: You are a video question-answering agent that navigates a long video through hierarchical temporal search. At each step, the current video segment is divided into non-overlapping child segments. You observe frames from each child segment and decide one of four actions:

1. `zoom_in <segment_id>`: Examine a child segment at finer granularity.
2. `zoom_out`: Backtrack to the parent segment to explore a different region.
3. `shift <segment_id>`: Move to a sibling segment under the same parent.
4. `answer <letter> <evidence_start> <evidence_end>`: Provide the answer to the question along with the time interval (in seconds) that contains the supporting evidence.

User:

Current Segment [t_s-t_e]:
...
Segment i [t_s^i-t_e^i]: <frame_timestamp><image> ...
...

Memory:
Tree Structure:
<materialized tree with [visited]/[current] flags and short node captions>

Interaction History:
<chronological log of past turns and actions>

Navigation Context:
<current position at the tree, parent, valid zoom_in children, valid shift siblings>

Question: ...
Choices: A. ... B. ... C. ... ...

Memory Structure. The memory is a textual block serialized into the prompt at every turn. It contains three components. The *tree structure* records the materialized part of the tree, where each node contains its temporal interval, a short caption, and a status flag (not visited, visited, or current).

The *interaction history* records the actions taken in previous turns. The *navigation context* lists the current node, its parent, and the valid `zoom_in` and `shift` targets, which prevents invalid actions. An example of the memory at an intermediate turn is shown below.

Memory Example

```
Tree Structure:
[0s-1531s] [visited], Caption:  [...]
|- Segment 0:  [0s-12s] [not visited], Caption:  [...]
|- Segment 1:  [12s-328s] [visited], Caption:  [...]
|    |- Segment 1.0:  [12s-20s] [not visited], Caption:  [...]
|    |- ...
|    `- Segment 1.3:  [90s-230s] [visited], Caption:  [...]
|        |- Segment 1.3.0:  [90s-113s] [visited] [current], Caption:  [...]
|        |- Segment 1.3.1:  [113s-149s] [not visited], Caption:  [...]
|        `- ...
`- ...
Interaction History:
- Turn 1:  ZOOM_IN(segment=1, [12s-328s])
- Turn 2:  ZOOM_IN(segment=3, [90s-230s])
- Turn 3:  ZOOM_IN(segment=0, [90s-113s])
Navigation Context:
- Current Position:  [90s-113s], Parent:  [90s-230s]
- Valid ZOOM_IN targets:  [...]
- Valid SHIFT targets:  Segment 1.3.1 [113s-149s], Segment 1.3.2 [149s-153s],
Segment 1.3.3 [153s-225s], Segment 1.3.4 [225s-230s]
```

Caption Generation. The captions in the memory are generated by VTS-8B itself. At each turn, the model is called twice. The first call takes the frames, the question, and the memory as input and generates the reasoning and the action. The second call generates a short caption for each child segment. The captions are cached in the memory, so each segment is captioned only once.

Training Dynamics. Figure 4 shows the training dynamics of our best GRPO model over the first 150 updates. Both the accuracy reward and temporal-IoU reward improve over training. The IoU reward is noticeably noisier, suggesting that temporal grounding is harder to optimize.

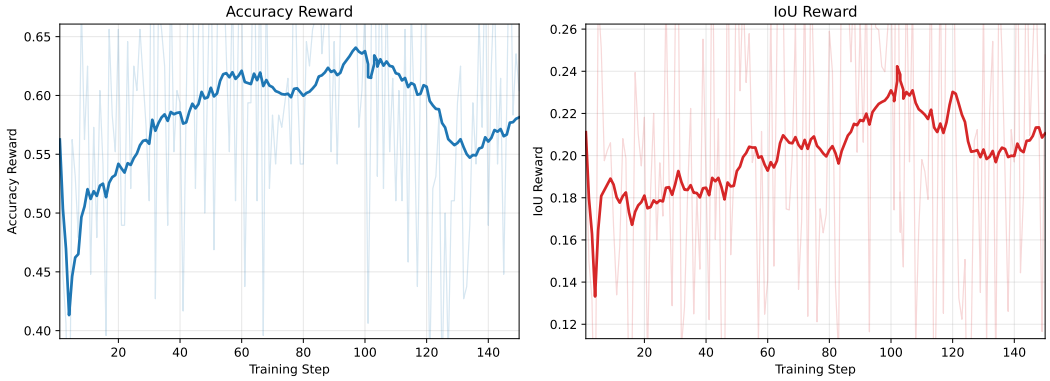


Figure 4: Accuracy (left) and temporal-IoU (right) rewards over the first 150 GRPO updates of our best model variant.

Inference Configuration. We employ the vLLM framework for inference, setting the temperature to 0 and using greedy decoding to obtain deterministic results. The maximum number of interaction turns is set to 10, matching the training setting. We use the official evaluation code for all three Grounded LVQA datasets: CG-Bench, Haystack-LVBench, and Haystack-Ego4D. Haystack-LVBench and Haystack-Ego4D evaluate predictions as a set of relevant frame indices rather than a temporal interval.

To adapt VTS's segment-level predictions to this frame-level evaluation, we uniformly sample 8 frames from the predicted segment as the frame-level prediction.

B Evaluation Benchmarks

CG-Bench-mini (filtered). CG-Bench provides multiple-choice questions paired with annotated clue intervals. The mini split contains 3,000 questions over videos averaging 28.5 min. In practice, we identify three issues with the original mini split: (1) some questions are answerable without video input; (2) the annotated clue is not always the only evidence, since the information needed to answer the question often appears outside the annotated segment as well, so the question can be answered without localizing that clue; and (3) some clue annotations are inaccurate. We address these with a four-stage filtering pipeline applied in the order below, where after each stage only the surviving questions are passed to the next. *Filter 1 (Rule-based)* applies filters that require no model inference: (i) *multi-segment removal*, dropping questions whose GT consists of more than one disjoint clue segment, since our hierarchical search formulation assumes a single contiguous target interval; (ii) *high-coverage removal*, dropping questions for which the GT segment covers more than 20% of the full video duration, because such questions admit trivial trajectories and provide little supervision for temporal search; and (iii) *validity check*, removing examples with non-positive video duration or missing clue intervals. *Filter 2 (Video dependency)* removes questions answerable from the question text alone: we prompt the VLM with only the question text and the multiple-choice options (no frames) and run direct inference, discarding the question if the predicted answer matches the ground-truth letter. *Filter 3 (Clue sufficiency)* verifies that the annotated GT clue is informative enough to answer the question: we sample frames at 1 fps from the GT segment (capped at 256 frames per question), feed them to the VLM with the question and choices, and drop questions for which the VLM fails to predict the correct answer. *Filter 4 (Clue uniqueness)* ensures the GT clue is the unique evidence required: we sample 1 fps frames from the entire video excluding the GT segment (capped at 256 frames) and prompt the VLM with these out-of-clue frames, discarding the question if the predicted answer matches the ground-truth letter, since such questions provide weak supervision for temporal grounding. All VLM-based filters use Qwen3-VL-8B-Instruct served via vLLM with temperature 0. The final filtered set contains 1,176 questions, averaging 28.7 min per video with a mean clue duration of 19.2 s, roughly 1.2% of the full video.

Haystack-LVBench is built on LongVideoBench. We use the official 342-question validation split over 114 videos, averaging 22.9 min with 1.84 keyframes per question.

Haystack-Ego4D is derived from Ego4D. We use the validation split, which contains 1,000 questions over 71 videos, averaging 26.2 min with 2.17 keyframes per question.

C Data Construction

C.1 LongClueQA Dataset

To train our model, we construct LongClueQA, a temporally grounded multiple-choice QA dataset constructed from unlabeled long YouTube videos. The dataset serves as a training resource for evidence-seeking behavior required by VTS: the model must not only answer a question, but also identify the short temporal interval that supports the answer.

Data Collection. We firstly collect videos from YouTube for a large and diverse pool of real-world long-form videos. Specifically, we collect candidate videos from YT-Temporal-1B [52], a large-scale corpus of YouTube videos. We then perform a filtering and select videos that are between 10 to 90 minutes in length with content types such as instructional, narrative, procedural, sports, and vlog-style videos that include sufficient metadata. We also only select videos that have Creative Commons or fair-use licensing requirements. In addition, we only select videos that do not appear in CG-Bench [5], Ego4D [14], and LongVideoBench [45], so that there is no video overlap between LongClueQA and the evaluation benchmarks.

To ensure that the final videos contain meaningful temporal structure, we apply a visual diversity filter based on CLIP embeddings. For each candidate video, we sparsely sample frames and compute normalized CLIP features. We then measure the average pairwise cosine similarity between sampled

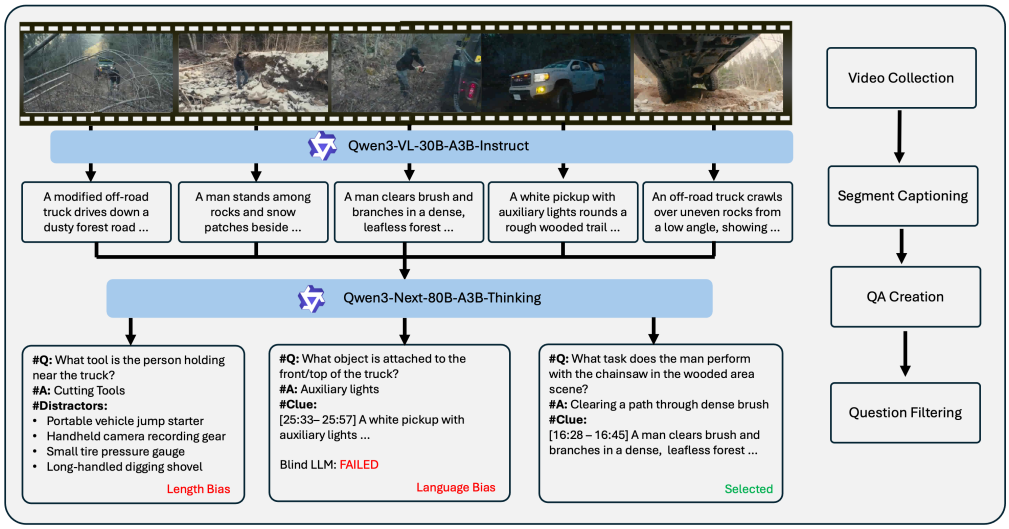


Figure 5: **Overview of the LongClueQA construction pipeline.** Collected YouTube videos are segmented into 10-second clips and captioned with Qwen3-VL. Multiple-choice QA pairs with temporal clues are then generated from the captions and filtered to retain only high-quality questions.

frames. We filter out videos with low diversity scores as such videos are more likely to contain static camera setups or repeated visual content, and are therefore less useful for training temporal grounding models.

QA Generation. We generate timestamp-anchored multiple-choice questions through a multi-stage pipeline. First, each video is segmented into 10-second clips. We then use Qwen3-VL-30B-A3B-Instruct to produce a free-form caption describing the visual content of that temporal segment. The caption is augmented with its start and end timestamp. We then use Qwen3-Next-80B-A3B-Thinking [47] as question generation model and input the previously extracted captions prompted to create three questions for each video with an answer and corresponding timestamp as the supporting clue. We then generate four distractors in a separate pass with the question, answer, and ±30-second caption around the supporting clue as input. The distractors are required to be plausible, semantically similar to the correct answer, and unambiguously incorrect with respect to the video. This yields a five-way multiple-choice question that challenges model to distinguish the correct answer from plausible alternatives. In total, LongClueQA contains 3,692 questions from 1,636 unique videos.

C.2 Trajectory Synthesis

Data Filtering. We use three data sources for generating trajectories: the CG-Bench full set (excluding questions from the mini set), the Haystack-Ego4D training set, and LongClueQA. CG-Bench and LongClueQA provide ground-truth (GT) clue-segment annotations. The annotations of Haystack-Ego4D are GT frame indices. We convert these to a single segment by taking the minimum and maximum of the annotated frame timestamps. We then apply a four-stage filtering pipeline to all data sources as described in Section B.

Trajectory Generation. Given a long video, a question, and the ground-truth evidence interval $[t_s^{\mathrm{GT}}, t_e^{\mathrm{GT}}]$, the controller produces a navigation trajectory through the tree. At any point during this process, the trajectory is in one of two states: a *correct* state, in which the current node overlaps the ground-truth interval, or a *wrong* state, in which it does not. The current state determines how the next action is chosen. In the correct state, we score each child of the current node for question relevance using an external scorer (Qwen3-VL-8B in our main experiments) and select the highest-scoring child as the next `zoom_in` target. If this child also contains the ground-truth interval, the trajectory remains in the correct state; otherwise it transitions into the wrong state. In the wrong state, we randomly choose between two options: (i) *recover*, in which we apply the inverse of the action sequence that led into the wrong branch (e.g., a `zoom_in` into a wrong sibling is recovered by `zoom_out` followed

by an optional `shift` to the correct sibling), or (ii) *continue*, in which we deliberately take another step deeper into the wrong branch to produce a longer detour. To prevent unbounded detours, we force recovery after the trajectory has spent two consecutive turns in the wrong state. The procedure terminates when the model can produce a correct answer with sufficiently accurate evidence (mIoU > 0.3) at the current node, or when further descent is no longer possible.

Trajectory Statistics Figure 6 reports the distribution of trajectory lengths across the three data sources. CG-Bench and LongClueQA trajectories are typically short (median 4 turns), while Haystack-Ego4D requires deeper search (median 6). The gap reflects the difficulty of each dataset's temporal-grounding queries: shorter trajectories correspond to questions answerable after a few coarse zooms, while longer ones require deeper traversal of the segment tree.

Figure 7 shows the breakdown of action types issued during trajectory generation. ZOOM_IN dominates on every dataset, reflecting the top-down nature of hierarchical search. ANSWER appears once per trajectory and therefore tracks dataset size. ZOOM_OUT and SHIFT act as corrective moves, and their relative frequency indicates how often the initial descent path needs to be revised before the agent commits to an answer.

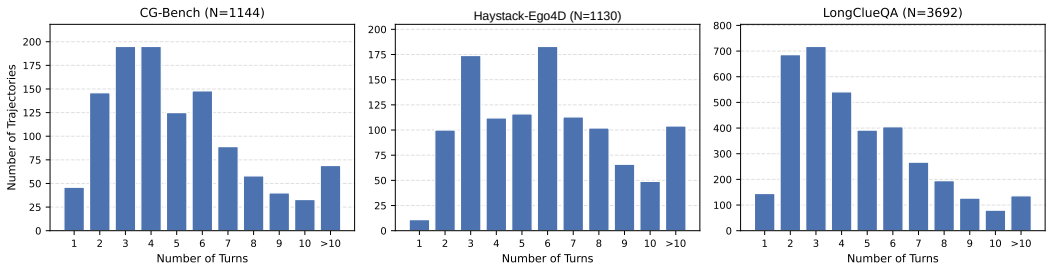


Figure 6: **Distribution of trajectory lengths across the three data sources.** For each dataset we plot the number of trajectories that terminate after a given number of turns, where one turn corresponds to a single agent action (ZOOM_IN, ZOOM_OUT, SHIFT, or ANSWER). Trajectories with more than 15 turns are discarded as outliers, and lengths beyond 10 are aggregated into a single "> 10" bin.

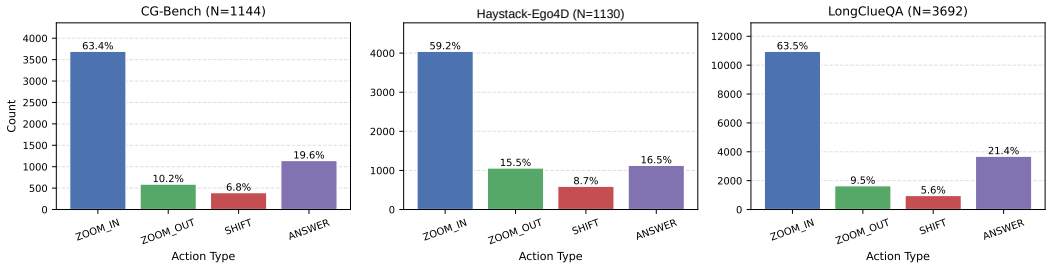


Figure 7: **Action type distribution across the three data sources.** Counts of the four action types issued by the agent across all trajectories from the three data sources.

D Additional Analyses

Controlled Comparison with Continuous Cropping. To disentangle the contribution of our training data from that of the tree-grounded action space, we train a continuous `crop_video` agent (Table 3) that is identical to VTS except for the action space. Specifically, this agent uses the same Qwen3-VL-8B backbone, training data, training budget, caption memory, and turn limit as VTS. The agent is also trained on the same detour-and-recovery trajectories, where each recovery step is expressed as a crop back to the parent or sibling interval. Trained on our data, this agent achieves 15.5 mIoU on CG-Bench, compared to 4.3 mIoU of the published LongVT. On top of the identical data and captions, the tree-grounded action space further improves temporal grounding by 1.3–4.2 points and QA accuracy by up to 9.8 points. These results indicate that our training data contributes

a large part of the improvement, while the tree-grounded action space brings consistent additional gains that do not come from caption quality.

Turn-Budget-Matched Comparison. To study whether the gains of VTS come from a longer search horizon, we construct LongVT-5, which calls `crop_video` in the first four turns and answers in the fifth turn. As shown in Table 8, LongVT-5 improves over LongVT by only 0.2 mIoU, while VTS outperforms LongVT-5 by 12.3 mIoU with a similar number of turns. These results indicate that the gains of VTS mainly come from the tree-grounded action space rather than a longer search horizon.

Table 8: **Turn-budget-matched comparison on CG-Bench.** Extending LongVT to five turns brings only marginal gains, while VTS outperforms it by a large margin with a similar number of turns.

Method	Turns	mIoU	Acc
LongVT [48]	2.0	4.3	17.4
LongVT-5	5.0	4.5	18.3
VTS (Ours)	4.8	**16.8**	**36.4**

Flat vs. Hierarchical Tree. In Table 3, the flat tree achieves slightly higher T-F1 than the hierarchical tree on Haystack-Ego4D (18.9 vs. 18.4). We note that the flat tree adaptively divides the video into short segments that are directly connected to the root. Since the clues in Haystack-Ego4D are uniformly short (9.5s on average), short segments are the right scale for almost every question, so direct access to these segments is well matched to this benchmark. In contrast, CG-Bench and Haystack-LVBench have more variable clue durations, and the hierarchical tree performs better on all their metrics.

Oracle Duration-Adaptive Tree. To study whether adapting the tree to the clue duration can bring further gains, we construct an oracle tree using the ground-truth clue duration, where nodes shorter than twice the clue duration are removed together with their descendants. As shown in Table 9, the oracle tree achieves 16.2 mIoU and 35.5 accuracy, slightly lower than our default tree. We hypothesize that the agent already decides at each node whether to descend further or answer, so it can stop at the natural scale of the clue on its own.

Table 9: **Comparison with an oracle duration-adaptive tree on CG-Bench.** Pruning the tree with the ground-truth clue duration does not improve performance.

Tree structure	mIoU	Acc
Oracle duration-adaptive tree	16.2	35.5
Adaptive temporal segmentation (default)	**16.8**	**36.4**

Incorporating Shot Boundary Detection. To study whether shot boundary detection can further improve adaptive temporal segmentation, we incorporate OmniShotCut [38] into our tree construction. Specifically, we keep our hierarchical tree but snap each boundary to the nearest shot boundary detected by OmniShotCut. As shown in Table 10, this variant slightly decreases mIoU (16.3 vs. 16.8) with the same accuracy. These results indicate that aligning our boundaries with editing cuts provides no additional benefit.

Table 10: **Incorporating shot boundary detection on CG-Bench.** Snapping our boundaries to the nearest OmniShotCut shot boundary does not improve performance.

Tree construction	mIoU	Acc
Snapped to OmniShotCut [38] boundaries	16.3	**36.4**
Adaptive temporal segmentation (default)	**16.8**	**36.4**

Visual Encoder for Segmentation. To study which visual encoder is more suitable for adaptive temporal segmentation, we conduct an ablation study with two widely used visual encoders, CLIP [28] and DINOv2 [25], while keeping all other components unchanged. The results are shown in Table 11. From the table, we can see that CLIP outperforms DINOv2 by 0.6 mIoU and 1.3 accuracy, possibly because its vision-language pretraining helps capture semantic transitions in the video. Meanwhile, DINOv2 still achieves comparable results, showing that VTS is robust to the choice of visual encoder.

Table 11: **Effect of the visual encoder for adaptive temporal segmentation on CG-Bench.** CLIP slightly outperforms DINOv2, and VTS is robust to the choice of visual encoder.

Visual encoder	mIoU	Acc
DINOv2 [25]	16.2	35.1
CLIP (default)	**16.8**	**36.4**

Captioning Model. To study how the quality of node captions affects VTS, we conduct an ablation study with three captioning models of different sizes: Qwen3-VL-2B, Qwen3-VL-4B, and VTS-8B (default). For all variants, we keep the VTS-8B action model and the search configuration fixed. The results are shown in Table 12. Stronger captioning models lead to better performance, improving mIoU by 1.2 and accuracy by 3.9 from 2B to 8B.

Table 12: **Effect of the captioning model on CG-Bench.** The action model is fixed to VTS-8B. Stronger captioning models lead to better performance.

Captioning model	mIoU	Acc
Qwen3-VL-2B	15.6	32.5
Qwen3-VL-4B	15.9	33.1
VTS-8B (default)	**16.8**	**36.4**

Runtime Analysis. Table 13 reports the runtime of each component of VTS on CG-Bench. Most of the runtime is spent on action generation and memory construction (mostly caption generation), while CLIP segmentation accounts for only 14% of the total runtime. We further compare the peak GPU memory in Table 14. VTS uses 20.0 GB of peak GPU memory, 34% less than Qwen3-VL-8B with 384 uniformly sampled frames (30.5 GB), since VTS processes at most 64 frames at each turn. VTS-fast has the same peak GPU memory as VTS.

Table 13: **Runtime breakdown of VTS on CG-Bench.** We report the average runtime per video on a single H100 GPU.

Component	Runtime (s)
CLIP segmentation	8.8
Action generation	24.7
Memory construction	28.1
Total	61.6

E More Qualitative Results

Qualitative Examples of Tree Navigation. Figures 8–9 present two additional qualitative results produced by VTS. Figure 8 illustrates a deeper nine-turn trajectory that uses both backtracking primitives. The question asks how many turkeys are placed on the table behind the man. The agent first follows several `zoom_in` actions into an incorrect branch, where the man is pouring liquid into a pot but no turkeys are visible on the table behind him. After recognizing this error, the agent issues `zoom_out` to return to the parent node, followed by `shift` to move laterally to an unvisited sibling. This sibling contains both the turkeys on the table and the man, allowing the agent to terminate with `answer`. Figure 9 shows a short two-turn trajectory for a question asking about the types of fruits placed on the cabinet. In the first `zoom_in` turn, the agent directly reaches the segment containing

Table 14: **Peak GPU memory on CG-Bench.** We report the peak GPU memory per video on a single H100 GPU.

Method	Peak GPU memory (GB)
Qwen3-VL-8B (384 frames) [1]	30.5
VTS-fast (Ours)	**20.0**
VTS (Ours)	**20.0**

both the fruits and the cabinet, and then terminates with `answer` in the next turn. This represents the common case for questions whose evidence is visually salient at a coarse level, showing that the agent avoids unnecessary exploration when the initial descent is already correct.

Comparison between VTS and SiLVR. Figure 10 compares VTS with SiLVR [55] on a question that requires counting objects at a specific moment. SiLVR's caption-based reasoning returns a moment that matches the textual description of the question (a man in a blue beanie cooking outdoors beside a table with a turkey) but precedes the queried moment by 146s. Because SiLVR reasons only over captions, it cannot verify the count and commits to "One". VTS instead reasons over the frames of the selected node and confirms the count visually before answering.

Comparison between VTS-RL and VTS-SFT. Figure 11 compares the trajectories of VTS-RL and VTS-SFT on the same question. Both models take the same first `zoom_in`, which already reaches a segment containing the evidence. They differ in when they stop. VTS-RL recognizes that the evidence is sufficient and answers correctly at turn 2. VTS-SFT keeps searching. It zooms into a node that lies entirely inside the ground-truth clue (724.2s–732.5s), yet backtracks with two `zoom_out` actions and `shifts` to a sibling segment that does not show the boy choosing the chips. There it misidentifies the second brand as "Potato Chip Workshop" and answers incorrectly. Both models use the same navigation actions, so the difference lies in deciding when the evidence is sufficient. SFT imitates demonstrated trajectories, whereas RL optimizes the whole trajectory for the answer and grounding outcome, which teaches the agent to terminate once it has found the evidence.

Qualitative Examples of Temporal Segmentation. Figure 12 shows adaptive temporal segmentation on two 64-second clips, one that changes slowly and one that changes rapidly. When a video changes gradually, few consecutive-frame distances exceed the adaptive threshold. In this case we add the next-largest distances until the node has at least 3 children, so every node is split along its most salient changes. The fixed-camera traffic clip in Figure 12(a) has no editing cuts and is split into only three segments, with boundaries where a white shuttle finishes crossing the intersection and where a red car passes. When a video changes frequently, many distances exceed the threshold. We keep only the 7 strongest boundaries, which caps each node at 8 children and keeps the branching factor bounded. The vlog clip in Figure 12(b) spans the same 64 seconds but is split into seven segments, each corresponding to a distinct activity, such as picking clothes, tying hair, or loading laundry. Moreover, visually subtle evidence does not need to produce a segment boundary, since the `answer` action predicts the evidence interval from the frames of the current segment.

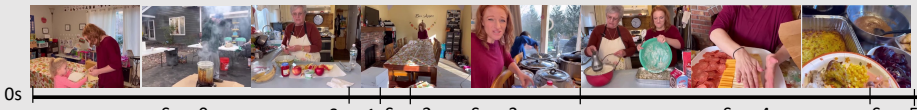
F Limitations and Future Work

VTS has several limitations. The current training data provides a single evidence interval per question, so VTS is trained to output one interval. Extending VTS to questions whose evidence is distributed across multiple disjoint temporal regions would require multi-interval annotations and corresponding training trajectories. The adaptive temporal segmentation relies on CLIP embeddings to detect visual and semantic transitions. Boundary detectors designed for long-form video could yield finer trees and additional gains. Finally, trajectory synthesis depends on external vision-language and language models for captioning and reasoning, so the quality of the training signal is bounded by these models. We expect it to improve as stronger open-source models become available.

Question. In the video, when the man in the blue hat is preparing to cook soup, how many turkeys are placed on the table behind him?

Options. A: One. B: Six. C: Four. D: Two. E: Three. F: Five. **Clue:** 691.0s – 694.0s

Turn 1



Segment 1 includes scenes of **outdoor cooking with a person in a blue beanie stirring a pot**, which matches the description of the man preparing to cook soup. This segment also contains later kitchen activities involving turkey preparation, making it the most relevant for finding turkeys on the table behind him.

zoom_in (segment=1)

Turn 2 zoom_in (segment=1) **Turn 3** zoom_in (segment=0) **Turn 4** zoom_in (segment=2)

Turn 5 zoom_in (segment=0) **Turn 6** zoom_in (segment=1)

Turn 7



The current segment shows the man pouring liquid into a pot, but the view is focused on the cooking activity with **no turkeys visible on the table behind him**. The question asks about turkeys placed on the table during soup preparation, which likely occurs in a different part of the parent segment. **Zooming out allows exploration of sibling segments that may show the relevant scene**.

zoom_out

Turn 8



The current segments show the man pouring liquids into pots or cooking with a deep fryer, but **turkeys are not clearly visible on tables behind him**. The **sibling segment 1** [642s-705s] continues the outdoor cooking scene and is the next unvisited region that **may reveal the turkeys behind him**.

shift (sibling_segment=1)

Turn 9



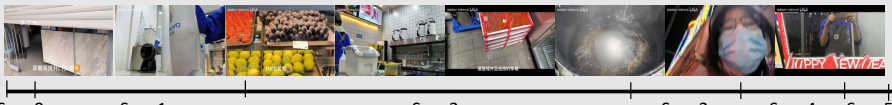
In the video from 689.7s to 696.4s, I observe **the man in the blue hat stirring a large pot outdoors**. Behind him, on a white table, **three turkeys are clearly visible**, confirming the answer.

answer (E, 689.7, 696.4)

Figure 8: **Qualitative example: recovery via `zoom_out` and `shift`.** The question asks for the number of turkeys on the table behind the man. The agent first follows several `zoom_in` steps into an incorrect branch that shows the man but no turkeys (turns 1–6). It then recovers by using `zoom_out` to ascend to the parent node (turn 7), followed by `shift` to move laterally to an unvisited sibling (turn 8). This sibling contains the relevant evidence—turkeys on the table behind the man—allowing the agent to answer correctly (turn 9). This trajectory illustrates how `zoom_out` and `shift` work together to recover from incorrect exploration.

Question
What fruits are there from left to right on the plate on the second layer of the fruit display cabinet?
Options
A: Strawberry, lemon, lemon. B: Grapes, strawberry, strawberry. C: Grapes, lemon, lemon.
D: Lemon, grapes, lemon. E: Lemon, lemon, grapes. F: Grapes, lemon, strawberry.
Clue
230.4s – 240.3s

Turn 1



Segment 2 includes **a scene with a refrigerated display case containing strawberries, grapes, and lemons**, which directly relates to the question about fruits on a display cabinet. This segment is the only one showing a fruit display, making it the most relevant for examining the arrangement of fruits. **Zooming in will allow closer inspection to identify the specific fruits from left to right**.
zoom_in (segment=2)

Turn 2

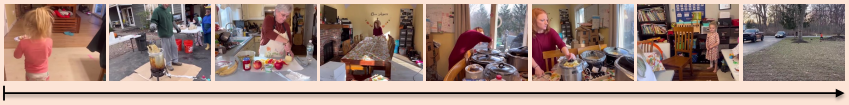


The video shows **the second layer of the fruit display cabinet** from 227.3s to 242.1s, where **grapes are visible on the left, followed by lemons in the middle, and more lemons on the right**. This arrangement is consistently observed across multiple frames within this interval, confirming the order from left to right.
answer (C, 227.3, 242.1)

Figure 9: **Qualitative example: hierarchical `zoom_in` without backtracking.** The question asks about the types of fruits placed on the cabinet. The agent directly reaches the relevant segment containing both the fruits and the cabinet after a single `zoom_in` step (turn 1), and then terminates with `answer` in the next turn (turn 2).

Question. In the video, when the man in the blue hat is preparing to cook soup, how many turkeys are placed on the table behind him?

Video.



Options. A: One. B: Six. C: Four. D: Two. E: Three. F: Five.
Ground truth. E: Three. **Clue:** 691.0s – 694.0s

SiLVR ✗ wrong

Output clue: 544.8s – 546.8s
Answer: A: One.

Retrieved caption (545.8s):

"A man in a green jacket and blue beanie prepares food outdoors, standing beside a table with a turkey, oranges, and red buckets, near a house with an American flag."

Why it fails: the retrieved moment is a **visually similar outdoor-cooking scene** 146s before the clue. Its caption mentions **only one turkey**, so SiLVR answers "One".

VTS (Ours) ✓ correct

Output clue: 689.7s – 696.4s
Answer: E: Three.

Final node 642s – 705s:



Reasoning: In the video from 689.7s to 696.4s, I observe the man in the blue hat stirring a large pot outdoors. Behind him, on a white table, **three turkeys are clearly visible**, confirming the answer.

Figure 10: **Comparison between VTS and SiLVR.** The question asks how many turkeys are on the table behind the man in the blue hat; the ground-truth clue is 691.0s–694.0s in a 4163s video. SiLVR retrieves a visually similar outdoor-cooking moment 146s earlier (544.8s–546.8s) whose caption mentions a single turkey, and answers "One". VTS navigates to the node 642s–705s, observes three turkeys on the table behind the man, and answers "Three" with evidence 689.7s–696.4s.

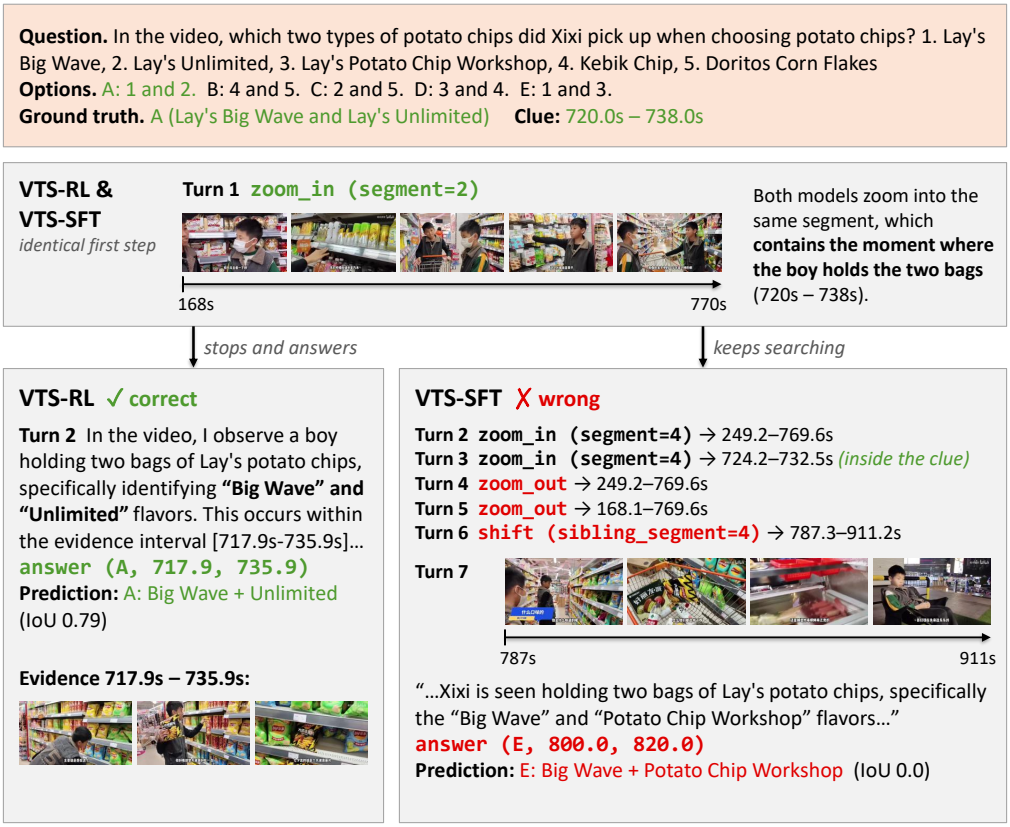


Figure 11: **Comparison between VTS-RL and VTS-SFT.** The question asks which two brands of potato chips the boy picks up; the ground-truth clue is 720.0s–738.0s. Both models first `zoom_in` to the segment 168.1s–769.6s, which contains the evidence. VTS-RL recognizes that the evidence is sufficient and answers correctly at turn 2 with evidence 717.9s–735.9s (IoU 0.79). VTS-SFT keeps searching: it reaches a node inside the clue (724.2s–732.5s), but then issues `zoom_out` twice, `shifts` to the sibling segment 787.3s–911.2s, and answers incorrectly from it (IoU 0.0).

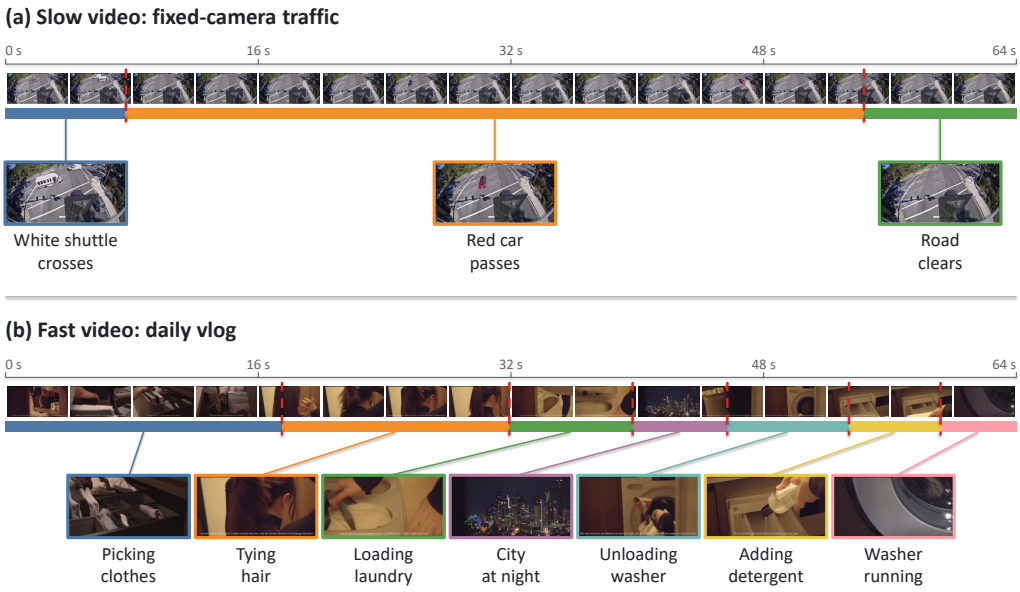


Figure 12: **Qualitative examples of adaptive temporal segmentation.** Each panel shows a 64-second clip with uniformly sampled frames (top), the resulting segments (colored bar) and boundaries (red dashed lines), and one representative frame per segment (bottom). (a) A fixed-camera traffic video with no editing cuts is split into three segments, the minimum allowed, with boundaries where a white shuttle finishes crossing the intersection and where a red car passes. (b) A daily-life vlog with frequent scene changes is split into seven segments, each corresponding to a distinct activity, such as picking clothes, tying hair, and loading laundry.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state three claims: (i) VTS's tree-grounded action space outperforms continuous cropping on long-video temporal grounding; (ii) explicit backtracking primitives improve self-correction; (iii) the navigation policy generalizes to general long-video QA. Each claim is supported by experiments in Sections 5.1, 5.4, and 5.2.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Appendix F, covering the single evidence interval per question in the current training data, the reliance on CLIP embeddings for adaptive temporal segmentation, and the dependence on external models for trajectory synthesis.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results; the contributions are an action-space design, a training pipeline, and empirical evaluation.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 describes benchmarks and baselines, and Section 3.6 describes the training data sources. The adaptive segmentation algorithm, action space, trajectory synthesis algorithm, and training procedure are specified in Section 3 with hyperparameters in the appendix. We additionally release code and the LongClueQA dataset.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code, model checkpoints (VTS-SFT and VTS-RL), and the LongClueQA dataset are released with documentation and reproduction instructions at the repository whose URL is provided in the abstract.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training data sources, frame counts, segmentation hyperparameters ($k = 1.5$, min/max children 3/8), trajectory synthesis configuration, and the SFT and RL training procedures (full fine-tuning for SFT and LoRA for RL) are described in Sections 3 and 4 and Appendix A, with full hyperparameters in Table 7.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not report error bars across multiple training runs because each VTS training run requires substantial compute (SFT and RL on $4\times$H100 GPUs), making multiple-seed evaluation prohibitive. The reported gains over baselines are large in absolute terms (e.g., $+12.5$ mIoU over LongVT on CG-Bench), substantially exceeding the run-to-run variance typically observed in similar settings.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Training uses $4\times$NVIDIA H100 GPUs. Supervised fine-tuning takes approximately 4 hours, and LoRA-based reinforcement learning takes approximately 32 hours (Appendix A). Inference runtime is measured on a single NVIDIA H100 GPU. The frame budgets, runtime, and peak GPU memory are reported in Table 2 and the appendix.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics?

   Answer: [Yes]

   Justification: The research uses publicly available video datasets and pretrained models in accordance with their licenses. The LongClueQA corpus is built from publicly available YouTube videos in YT-Temporal-1B with Creative Commons or fair-use licenses; we do not collect new data from human subjects.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Positive impacts include accessibility applications (assistive video search and summarization) and reduced computational cost relative to dense-frame methods. Potential negative impacts include misuse for surveillance footage analysis or unauthorized content moderation.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: VTS is a video understanding model that produces multiple-choice or short-form answers grounded in video evidence; it does not generate images, video, or open-ended text content with substantial misuse potential beyond what is already present in the underlying Qwen3-VL backbone.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets used in the paper properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets (CG-Bench, LV-Haystack, Video-MME, MLVU, LVBench, LongVideoBench, Ego4D, YT-Temporal-1B) and models (Qwen3-VL, Qwen2.5-VL, Qwen3-Next, DeepSeek-R1, CLIP, DINOv2, OmniShotCut) are cited in the paper with their original publications, and we use them in accordance with their licenses and terms of use.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the VTS-SFT and VTS-RL checkpoints and the LongClueQA dataset. The appendix documents the data generation pipeline, filtering procedure, dataset statistics, and training hyperparameters, and the release includes usage instructions.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human-subject research. LongClueQA annotations are generated automatically from long-form videos using vision-language and language models.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve human-subject research and therefore does not require IRB approval.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research?

    Answer: [Yes]

    Justification: External vision-language and language models are used as components of the trajectory synthesis pipeline (Section 3.4): Qwen3-VL-8B for scoring the question relevance of child segments and DeepSeek-R1 for reasoning content generation. LongClueQA is generated with Qwen3-VL-30B-A3B-Instruct for clip captioning and Qwen3-Next-80B-A3B-Thinking for question generation. These are documented in Section 3 and the appendix. No external LLM is used as part of VTS at inference time.