ESTA-MEM: COMPILING MULTI-AGENT TRAJECTORIES INTO AUDITABLE MEMORY FOR HIGH-RESOLUTION VISUAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cognitive science shows that human visual memory binds observations to their sources for reliable decisions. In recent years, multimodal large language models (MLLMs) have approached human visual reasoning ability, but processing high-resolution images remains challenging. Existing agent methods enhance the visual perception ability of MLLMs, and multi-agent systems further improve complex reasoning by splitting tasks and assigning them to specialized roles. However, as multi-agent interactions increase, the shared history grows longer without clear links between visual findings and reasoning conclusions. Inspired by cognitive science, we propose ESTA-Mem, an ***E**vent-**S**ourced, **T**yped, and **A**uditable* Memory system that integrates interaction history into an evolving task state. Linked structures explicitly bind candidate answers to corresponding observations and verification records, while a memory agent proposes updates. Each agent receives these relations through a view tailored to its role, avoiding repeated reconstruction from mixed histories. This clarifies the current task state and next steps for each agent, making memory an active driver of multi-agent visual reasoning. We test our method on five high-resolution benchmarks. ESTA-Mem improves the baseline by 2.6 and 4.2 percentage points on V* and TreeBench, respectively, and outperforms the compared memory methods. Results show that memory organization itself matters, improving visual reasoning without stronger models or longer reasoning.

1 INTRODUCTION

Cognitive research suggests that people link new and previously seen details of an object, keeping them together in object files (Kahneman et al., 1992). Related research further suggests that event files link what we see to what we do (Hommel, 1998). Humans also try to distinguish what they saw from what they imagined, a process called source monitoring (Johnson et al., 1993). Consider seeing and using a cup, as illustrated in Figure 1. We recognize that its front and side views show the same cup, and seeing it again may remind us of holding it. We may also wonder whether a detail on the cup was seen or only imagined. These processes suggest that human visual memory does not passively accumulate what was seen. It organizes links among objects, experiences, and sources to provide a basis for later judgments.

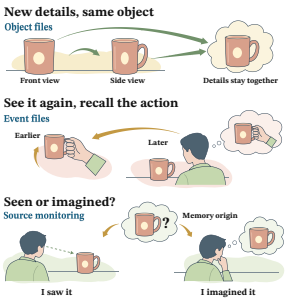

Figure 1: Cognitive basis.

How, then, do artificial systems organize and use visual information? In recent years, multimodal large language models (MLLMs) have demonstrated the ability to answer natural-language questions about images (Li et al., 2023; Liu et al., 2023; Bai et al., 2025a). However, many real-world applications, such as remote sensing and autonomous driving, require high-resolution images to capture fine details (Wang et al., 2025a; Ding et al., 2025; Zhang et al., 2025c). As illustrated in Figure 2 (left), MLLMs that answer directly from a global view may miss information required to answer correctly. To address this limitation, recent work equips MLLMs with active visual exploration. These systems use guided search and zooming to locate useful details (Wu & Xie, 2024; Shen et al., 2025), reinforcement learning to decide where to look (Zheng et al., 2026; Su et al., 2026), hierarchical scanning to examine an image at multiple scales (Li et al., 2026b), or adaptive scanning to change

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

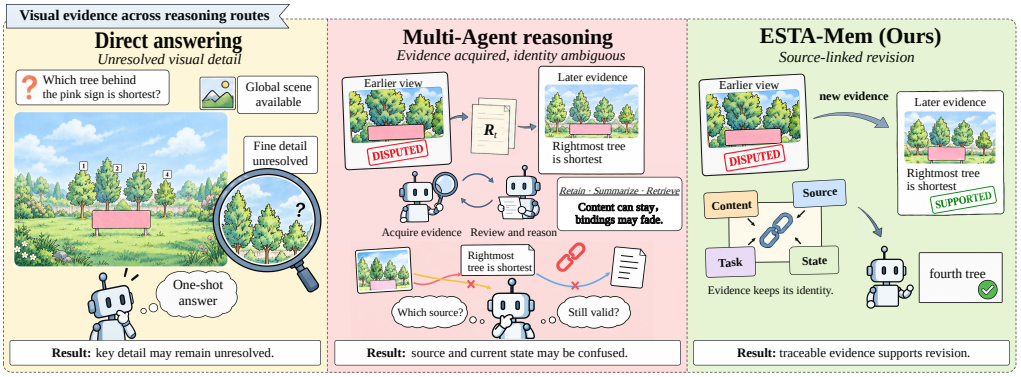

Figure 2: Left: direct answers may miss details. Middle: agent histories can obscure observation sources and review states. Right: ESTA-Mem preserves these links to guide answer revision.

the search when the current view is insufficient (Li et al., 2026a). However, jointly handling visual localization, cross-region reasoning, and verification can place an added burden on a single agent.

Unlike these single-agent systems, multi-agent systems distribute these responsibilities across specialized roles, allowing agents to focus on subtasks while exchanging findings and feedback. AutoGen (Wu et al., 2023) connects diverse agents through conversation, while MetaGPT (Hong et al., 2024) coordinates specialized roles through standardized development workflows. Subsequent work explores memory mechanisms that organize information from past interactions to support decisions. For example, MemGPT (Packer et al., 2023) manages memory tiers, A-MEM (Xu et al., 2026) links notes, and G-Memory (Zhang et al., 2026) organizes multi-agent experience. However, these methods primarily categorize and organize past records without explicitly preserving the relations among observations, candidate answers, and reviews within the current reasoning trajectory. *Later agents repeatedly reconstruct these relations independently from history and risk linking evidence or feedback to the wrong claim.* Figure 2 (middle) illustrates the consequence: although the observations and reasoning steps are retained, later agents struggle to reconstruct these relations from separate records, leaving them uncertain about which source produced a claim and whether it is still valid.

Inspired by the cognitive research above, we introduce ESTA-Mem, an ***E**vent-**S**ourced, **T**yped, and **A**uditable* Memory system that compiles histories into a task state that evolves during reasoning. To reduce the risk of linking evidence or reviews to the wrong judgment, ESTA-Mem links three structures: a state tree organizing evidence combinations, a provenance graph preserving source and task links, and a candidate ledger tracking each candidate's evidence and reviews. To keep these links up to date as reasoning proceeds, a memory agent proposes updates to evidence associations and review states. Role-specific views clarify these relations by providing relevant evidence and its updated status when agents consult memory, easing the burden of repeatedly reconstructing them from history. The memory system thus shifts from organizing history to explicitly maintaining reasoning state across roles, evolving from a passive participant into an active driver of visual reasoning.

Our evaluation spans V*, TreeBench, HRBench-4K, HRBench-8K, and XLRS-Bench Lite. ESTA-Mem improves the full-history multi-agent baseline on all five benchmarks, including gains of 2.6 points on V* and 4.2 points on TreeBench. Subsequent experiments show consistent gains across reasoning steps, strong late-stage error correction, and advantages over alternative memory methods. A grouped analysis shows larger gains on tasks requiring evidence integration across image regions and reasoning steps. Role-conditioned views reduce the memory text delivered to agents by 92.88%.

Our contributions in this work are threefold:

- We recast multi-agent memory as online task-state compilation and introduce ESTA-Mem, turning passive history storage into an active driver of multi-agent visual reasoning.
- We develop an event-sourced, typed, and auditable memory compiler with role-conditioned readouts, which resolves uncertainty over each claim's source and validity, eliminates repeated evidence relation reconstruction from mixed histories, and prevents misreading.
- We demonstrate gains on five benchmarks, supported by memory comparisons, component ablations, and reasoning-budget analyses, with larger gains on complex multi-step reasoning.

2 Related Work

Multimodal large language models. MLLMs connect visual representations with language models to support image understanding and instruction following. BLIP-2 (Li et al., 2023) bridges pretrained vision and language models, while LLaVA (Liu et al., 2023) develops multimodal instruction following through visual instruction tuning. LLaVA-OneVision (Li et al., 2024) extends this capability across single-image, multi-image, and video tasks. InternVL3 (Zhu et al., 2025) and the Qwen-VL series (Bai et al., 2025b;a) further advance visual perception and multimodal reasoning through model and training improvements. These models strengthen perception and reasoning but leave observations across regions and steps unorganized, which we address through online task-state compilation.

Visual agent reasoning. Visual agents extend direct answering with regional inspection, tool use, and feedback. V* (Wu & Xie, 2024), ZoomEye (Shen et al., 2025), and CVSearch (Li et al., 2026a) guide search toward fine-grained image details needed to answer the question. PixelReasoner (Su et al., 2026), DeepEyes (Zheng et al., 2026), and DeepEyesV2 (Hong et al., 2026) train models to integrate reasoning with visual operations and tool use. Evidence organization also supports this process: DeepScan (Li et al., 2026b) aggregates multiscale views in a hybrid evidence memory, while TreeVGR (Wang et al., 2026c) jointly trains localization and reasoning to produce traceable visual evidence. Complementing these visual agents, AutoGen (Wu et al., 2023) and MetaGPT (Hong et al., 2024) provide general frameworks for role-based collaboration. MATA (Cai et al., 2026) learns agent selection for visual reasoning, while AgentFlow (Li et al., 2026c) coordinates planning, execution, verification, and generation through evolving memory and optimizes the planner within the interaction loop. These approaches strengthen visual search and agent coordination, but do not preserve how observations relate to candidate answers and later reviews. Our method addresses this gap and complements them by maintaining these relations across steps and roles.

Memory for language, multimodal, and multi-agent systems. Agent memory supports reuse of past information through structured storage and selective retrieval. A-MEM (Xu et al., 2026), Mem0 (Chhikara et al., 2025), and AMA-Agent (Zhao et al., 2026b) organize linked notes, persistent facts, and causal relations, respectively. Beyond individual language agents, G-Memory (Zhang et al., 2026) supports cross-trial multi-agent collaboration, and MementoGUI (Zeng et al., 2026) learns multimodal memory control for long-horizon GUI tasks. Moving closer to evidence-aware memory, MemIR (Jin et al., 2026) separates raw evidence, retrieval cues, and factual claims through typed representations, while MESA (Zhao et al., 2026a) selects and fuses query-specific subsets of structured memory views. These methods organize records for storage and retrieval, whereas our method introduces a different unit of memory: the evolving decision state. It explicitly preserves how observations from particular sources support, conflict with, or revise candidate answers.

3 Method

In this section, we present details of our method, ESTA-Mem. It tracks evidence, its sources, candidate answers, and reviews across visual reasoning steps. As Figure 3 shows, it builds memory from the raw multi-agent execution record R, updates it through a memory agent, and supplies relevant content to task agents. A memory update followed by a role-specific read and decision can be expressed as

$$R \xrightarrow{\Phi} \tau \xrightarrow{\mathcal{C}_\Gamma} M \xrightarrow{\text{audit, append, replay}} M^+ \xrightarrow{\mathcal{P}_r} V^{(r)} \xrightarrow{F_0} d^{(r)}. \tag{1}$$

The mapper Φ converts records into the typed event sequence τ, and the compiler $\mathcal{C}_\Gamma$ constructs memory M. After validating and recording changes proposed by the memory agent, the system rebuilds memory as M^+. For task-agent role r (*e.g.,* planner or verifier), the role-specific projection $\mathcal{P}_r$ selects and formats relevant content from M^+ into a memory view $V^{(r)}$, which the frozen system F_0 uses to make decision $d^{(r)}$. The following subsections describe the multi-agent baseline, memory compilation, memory agent auditing, role-specific memory delivery, and terminal answering.

3.1 Task Setup and Multi-Agent Baseline

Given an instance $x = (q, I, \Omega)$ comprising a question, an image, and an answer space, the multi-agent system F_0 runs for at most T steps. It includes the task agents *planner*, *executor*, *verifier*, and *answerer*, together with a *manager*, *tool agents*, a *visual reflector*, and a *reasoner*. The *manager* first identifies the visual target instances to inspect. At step t, the *planner* selects the next action or submits

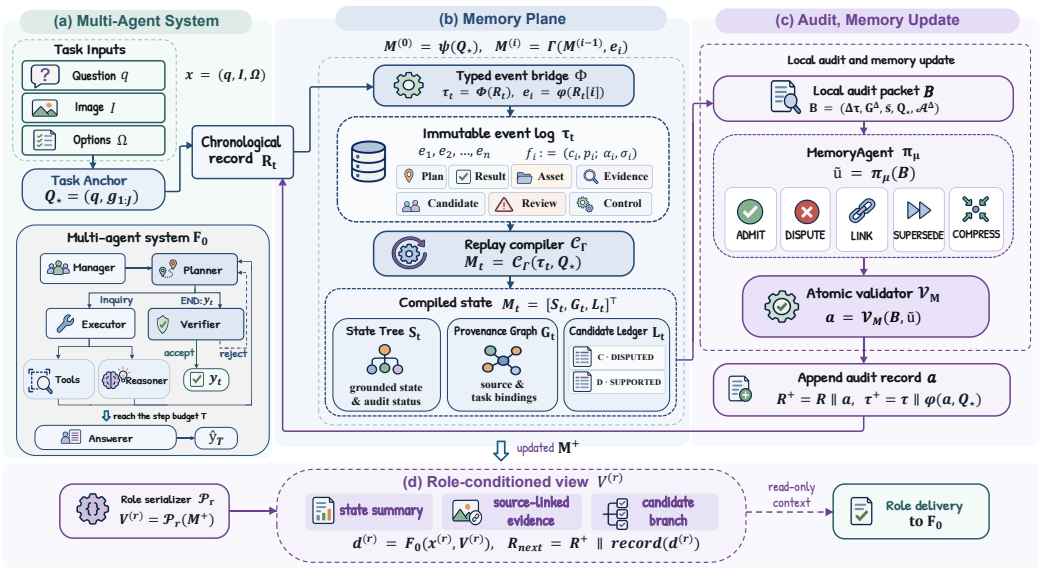

Figure 3: Overview of ESTA-Mem. (a) Agents acquire evidence. (b) Events form linked memory structures. (c) Audits update evidence relations. (d) Role-specific views inform task agents.

a candidate answer. An action instruction enters the inquiry branch, where the *executor* translates it into calls to *tool agents* for specific operations or to the *reasoner* for cross-region reasoning. For crop actions, low-confidence proposals trigger the *visual reflector* to suggest the next region to inspect. The system then begins step $t + 1$ with the *planner*. When the *planner* instead submits a candidate answer, the system enters the end branch, where the *verifier* checks it against the available evidence. Acceptance ends the process and returns the candidate, while rejection returns control to the *planner* for further reasoning. If the step budget is exhausted without an accepted candidate, the *answerer* produces the final answer from the available memory. Throughout this process, the agent outputs of F_0 are recorded chronologically in R and shared with the baseline task agents. ESTA-Mem further organizes R into a shared task state and selectively delivers role-relevant content to the task agents.

3.2 Compiling History into Task State

Memory uses task anchors $Q_\star = (q, g_{1:J})$, where q is the question without answer options and $g_{1:J}$ contains the manager's ordered target instances. After control step t, the accumulated records R_t are mapped to the typed event log $\tau_t = \Phi(R_t)$. Since a step may produce multiple records, i indexes individual events, with $e_i = \phi(R_t[i])$ for each record. Each event has one of seven types: plan, result, asset, evidence, candidate, review, and control. The mapping preserves event order, provenance, creation state, and references to parent events and visual assets, and recorded events remain immutable. Evidence events, in particular, store semantic observations for subsequent review, association, and reuse. For each such event e_i, replay constructs the corresponding observation frame

$$f_i := (c_i,\ p_i\ ;\ \alpha_i,\ \sigma_i), \qquad e_i \in \tau_t. \tag{2}$$

Here, c_i stores the observation content and p_i records its source event, producing role, step, and visual-asset references, with both fields fixed after recording. The task links α_i and review state σ_i are computed from the current log τ_t. New memory operations or reviews can change which targets the observation is linked to and how it is judged, without altering its content or source. The state tree S_t organizes combinations of available visual assets and observations, while the provenance graph G_t links observations to sources and task targets. The candidate ledger L_t records the observations and memory state available when each candidate was proposed, and links the candidate to its later reviews. We use ψ to initialize memory from the task anchors, Γ to update it with each event, and $\mathcal{C}_\Gamma$ to denote the result of replaying the complete log in order:

$$M^{(0)} = \psi(Q_\star), \quad M^{(i)} = \Gamma\left(M^{(i-1)}, e_i\right), \quad M_t = \mathcal{C}_\Gamma(\tau_t, Q_\star) = M^{(|\tau_t|)} = [S_t, G_t, L_t]^\top. \tag{3}$$

State Tree. The state tree S_t records the usable information available at the end of each control step. Let $\mathcal{A}_t^+$ contain the visual assets acquired through step t that remain available, and $\mathcal{O}_t^+$

contain observations currently admitted by the memory agent or accepted by the verifier. Sorting the identifiers in both sets produces the state signature $\kappa_t = \mathrm{canon}(\mathcal{A}_t^+, \mathcal{O}_t^+)$. This signature identifies whether the current combination of available visual assets and effective observations has appeared before. If κ_t already corresponds to a node, the tree reuses that node. Otherwise, it creates a child of the current node. Disputed observations remain recorded with the node but are excluded from the signature. Repeating an action without obtaining new assets or observations therefore does not count as new state progress, so repeated attempts are tracked within the same evidence state for later review.

Provenance Graph. The provenance graph G_t expands p_i into relations to source events, producing roles, and visual assets, and α_i into links to the corresponding targets in $Q_\star$. Ordinary events supply existing parent-event and visual-asset references, while validated audits add permitted task, support, conflict, or review relations. All relations retain their original event identifiers for later source tracing.

Candidate Ledger. The candidate ledger L_t records submitted candidates and uses observation identifiers to reference the observation frames visible at candidate production, the corresponding state nodes, and review results. Identical answer strings from different events are stored separately to preserve their source and state contexts for subsequent review and comparison with later candidates.

Let $\Delta\tau_t$ contain the new events produced during control step t. The following equation describes how the state tree, provenance graph, and candidate ledger update together at the end of the step:

$$\begin{bmatrix} S_t \\ G_t \\ L_t \end{bmatrix} = \begin{bmatrix} \Gamma_S(S_{t-1}, \kappa_t, \Delta\tau_t) \\ \mathcal{D}_G(G_{t-1} \cup \Delta G(\Delta\tau_t)) \\ \Gamma_L(L_{t-1}, \Delta\tau_t, S_t) \end{bmatrix}. \tag{4}$$

$\Delta G(\Delta\tau_t)$ contains the relations introduced by this step's events, $\mathcal{D}_G$ removes exact duplicate edges without merging sources, and Γ_L associates the step's candidate records with the updated state S_t.

3.3 Auditing and State Updates

A frozen memory agent π_μ audits new semantic observations, submitted candidates, rejection feedback, or repeated actions in the same evidence state. Its input includes new events $\Delta\tau$, the portion of the provenance graph directly connected to these events (G^Δ), the frontier node $\bar{s}$, the task anchors $Q_\star$, and referenced cached visual assets $\mathcal{A}^\Delta$. These inputs together form the audit context:

$$B = \big(\Delta\tau, G^\Delta, \bar{s}, Q_\star, \mathcal{A}^\Delta\big), \qquad \widetilde{u} = \pi_\mu(B). \tag{5}$$

The proposal $\widetilde{u}$ can use five operations to update observation states, evidence relations, and summaries of repeated attempts. Specifically, admit marks observations as visually supported and eligible for the active set, dispute flags unresolved concerns, and link adds permitted source, task, support, or conflict relations between existing items. The supersede operation connects a better-grounded observation to an older one about the same object and property, admitting the former and marking the latter as superseded. The compress operation summarizes repeated attempts within a state while retaining event references. Admission and supersession remain subject to verifier precedence. The validator checks identifiers, types, scopes, required links, and asset availability against M. It keeps the full proposal when every operation passes these checks and records an empty transaction when any operation fails. Let $\mathcal{V}_M(B, \widetilde{u})$ be the audit record created by this check, storing the proposal, retained operations, audited range, trigger, and status. It is appended before the next replay of the event log:

$$a = \mathcal{V}_M(B, \widetilde{u}), \quad R^+ = R \| a, \quad \tau^+ = \tau \| \phi(a, Q_\star), \quad M^+ = \mathcal{C}_\Gamma(\tau^+, Q_\star). \tag{6}$$

Here $\|$ denotes appending. Invalid or failed proposals produce an audit record with an empty transaction and error status, leaving S, G, and L unchanged. Authority over visual correctness remains with F_0 and the verifier. For observation i, let σ_i^0, σ_i^μ, and σ_i^v denote its creation, memory agent, and verifier judgments, respectively. Replay then determines the current review state as

$$\sigma_i = \xi\big(\sigma_i^v,\, \sigma_i^\mu,\, \sigma_i^0\big). \tag{7}$$

The operator ξ selects the first available judgment from the verifier, memory agent, or creation state, in that order. Disputed or rejected judgments are not skipped, and observation content and provenance remain unchanged. Admitting evidence to memory does not imply accepting a candidate answer.

3.4 Role-Conditioned Delivery

At each designated memory read, a serializer selects content from the updated memory M^+ relevant to role r, forming a read-only view $V^{(r)}$. The role reads this view alongside its original task inputs

$x^{(r)}$ and produces an output, which is then added to the record for subsequent memory updates:

$$V^{(r)} = \mathcal{P}_r(M^+), \quad d^{(r)} = F_0\big(x^{(r)}, V^{(r)}\big), \quad R_{\text{next}} = R^+ \| \operatorname{record}\big(d^{(r)}\big). \tag{8}$$

Here $\mathcal{P}_r$ is a deterministic projection for role r. The planner view V^P contains the current frontier, uncovered task targets, source-linked observations, and repeated attempts. The executor view V^E contains the current action's dependencies, task associations, and available visual assets. The verifier view V^V adds observation sources, task associations, and effective review states without candidate preferences or answer recommendations. The answerer state view $V^{A,s}$ collects the current state and accepted observations with their sources and task associations for answer generation. New role outputs enter the same event-mapping and replay process, updating the shared task state.

3.5 Source-Aware Terminal Answering

ESTA-Mem preserves the original controller's answer-return rules. If the candidate y_t at step t passes verification, the system returns it directly. If no candidate is accepted within the step budget T, the answerer performs terminal answering. The system constructs a terminal view V_T^A from the compiled memory, retaining currently available observations, their visual sources and task associations, and the evolution of candidate answers across memory states. The answerer then produces the final answer

$$\widehat{y}_T = \pi_A\big(q, I, \Omega, V_T^A\big). \tag{9}$$

Here π_A is the frozen answerer and V_T^A its read-only terminal view. The view treats historical candidates as observations tied to their production states and sources, not as votes accumulated by frequency. The answerer reassesses the question from the current, traceable memory state rather than reusing the last candidate. If the terminal output cannot be parsed, the system falls back to the original answerer interface. Algorithm 2 in the appendix details view serialization and execution.

4 Experiments

4.1 Experimental Setup

To assess the effectiveness of our method, we conduct experiments across the general high-resolution benchmarks V*Bench (Wu & Xie, 2024), TreeBench (Wang et al., 2026c), HRBench-4K, and HRBench-8K (Wang et al., 2025b), as well as the ultra-high-resolution remote-sensing benchmark XLRS-Bench Lite (Wang et al., 2025a). By default, all VLM-based roles use the same frozen Qwen3-VL-8B backbone (Bai et al., 2025a). The additional backbone setting in Table 1 uses Qwen2.5-VL-7B. The mixed-size systems in Section 4.3 assign Qwen3-VL-2B, 4B, 8B, or 32B to different roles. The main experiments use at most $T = 8$ reasoning steps. Full details are provided in Appendix B.1.

4.2 RQ1: Does ESTA-Mem Improve the Multi-Agent System?

ESTA-Mem consistently improves the multi-agent system. In Table 1, we compare our method with general-purpose MLLMs, training-free visual-search systems, and trained visual reasoning and tool-use agents. ESTA-Mem achieves the highest four-benchmark macro average of 73.37 among all the reported methods. Then, we compare it with the full-history multi-agent baseline, as shown in the bottom block of Table 1. The gains are 2.6 points on V*, 4.2 on TreeBench, and 1.1 on each HRBench resolution. The larger gains on V* and TreeBench are consistent with their greater need to integrate evidence acquired from different image regions across multiple reasoning steps, while the positive HRBench results show that the benefit persists across image resolutions. These competitive results show that ESTA-Mem can strengthen the underlying visual-reasoning system without parameter updates. Appendix B.2 reports the same positive trend on XLRS-Bench Lite.

4.3 RQ2: How Do Model Scale and Max Steps Affect Performance?

Model scale is most effective when allocated according to role responsibility. To investigate whether all VLM-based roles should use the same backbone size, we test multiple mixed-size configurations on V*, with different agent roles using Qwen3-VL-2B, 4B, 8B, or 32B models. In the left panels of Figure 4, the blue line represents mixed-size systems, whereas the orange line represents

Table 1: Main benchmark accuracy. Avg is the macro average over V*, TreeBench, HRBench-4K, and HRBench-8K. Green number boxes denote gains, with deeper shades marking larger gains.

Method	Backbone	Params	V*			TreeBench			HRBench-4K			HRBench-8K			Avg
			Attr.	Rel.	All	Perc.	Reas.	All	FSP	FCP	All	FSP	FCP	All	
General-purpose MLLMs															
GPT-4o (Hurst et al., 2024)	Proprietary	–	72.2	60.5	67.5	61.7	38.3	46.9	66.8	63.3	65.0	60.8	58.5	59.6	59.8
LLaVA-OneVision (Li et al., 2024)	Qwen2	7B	75.7	75.0	75.4	44.3	33.2	37.3	72.0	54.0	63.0	67.3	52.3	59.8	58.9
InternVL3 (Zhu et al., 2025)	InternVL	8B	67.8	73.7	70.2	46.3	34.4	38.8	78.8	61.3	70.0	78.8	59.8	69.3	62.1
Training-free visual search and test-time reasoning															
ZoomRefine (Yu et al., 2025)	Qwen2.5-VL	7B	85.3	77.6	82.2	48.3	32.0	38.0	88.5	55.3	71.9	83.9	54.0	69.0	65.3
DyFo (Li et al., 2025)	Qwen2.5-VL	7B	82.6	86.8	84.3	49.0	33.6	39.3	89.2	53.5	71.3	86.5	53.2	69.8	66.2
LazyMCoT (Wang et al., 2026d)	Qwen3-VL	8B	92.2	89.5	91.1	55.0	36.7	43.5	94.3	66.5	80.4	90.8	64.0	77.4	73.1
Trained visual reasoning and tool-use models															
PixelReasoner (Su et al., 2026)	Qwen2.5-VL	7B	83.5	76.3	80.6	54.3	30.1	39.0	86.0	60.3	73.2	80.0	54.3	67.2	65.0
DeepEyes (Zheng et al., 2026)	Qwen2.5-VL	7B	84.3	88.2	85.9	57.7	29.3	39.8	90.5	55.2	72.9	84.3	54.7	69.5	67.0
Mini-o3 (Lai et al., 2026)	Qwen2.5-VL	7B	87.8	88.2	88.0	–	–	40.3	84.0	54.0	69.0	82.0	49.2	65.6	65.7
Chain-of-Focus (Zhang et al., 2025a)	Qwen2.5-VL	7B	90.4	85.5	88.5	–	–	39.8	92.2	53.7	73.0	87.7	51.0	69.4	67.7
TreeVGR (Wang et al., 2026c)	Qwen2.5-VL	7B	86.1	85.5	85.9	59.7	30.1	41.0	–	–	72.7	–	–	69.8	67.4
Thyme (Zhang et al., 2025b)	Qwen2.5-VL	7B	83.5	80.3	82.2	57.0	28.9	39.3	91.0	63.0	77.0	86.5	57.5	72.0	67.6
Our Method															
Direct	Qwen2.5-VL	7B	73.9	67.1	71.2	43.6	33.2	37.0	85.2	52.2	68.8	78.8	51.8	65.3	60.6
Multi-Agent Baseline	Qwen2.5-VL	7B	81.7	78.9	80.6	57.0	30.5	40.2	88.5	58.8	73.6	80.5	55.0	67.8	65.6
ESTA-Mem (Ours)	**Qwen2.5-VL**	**7B**	**86.1**	**84.2**	**85.3**	**59.7**	**32.8**	**42.7**	**89.0**	**60.8**	**74.9**	**81.5**	**56.8**	**69.1**	**68.0**
Direct	Qwen3-VL	8B	82.6	80.3	81.7	64.4	34.0	45.2	89.5	61.0	75.3	80.8	55.5	68.1	67.6
Multi-Agent Baseline	Qwen3-VL	8B	87.8	86.8	87.4	65.8	42.6	51.1	90.5	62.3	76.4	80.5	58.5	69.5	71.1
ESTA-Mem (Ours)	**Qwen3-VL**	**8B**	**88.7**	**92.1**	**90.1**	**68.5**	**47.7**	**55.3**	**90.8**	**64.3**	**77.5**	**81.5**	**59.8**	**70.6**	**73.4**
Δ *vs. Direct*	Qwen3-VL	8B	↑ 6.1	↑ 11.8	↑ 8.4	↑ 4.0	↑ 13.7	↑ 10.1	↑ 1.3	↑ 3.3	↑ 2.3	↑ 0.8	↑ 4.3	↑ 2.5	↑ 5.8
Δ *vs. Multi-Agent*	Qwen3-VL	8B	↑ 0.9	↑ 5.3	↑ 2.6	↑ 2.7	↑ 5.1	↑ 4.2	↑ 0.3	↑ 2.0	↑ 1.1	↑ 1.0	↑ 1.3	↑ 1.1	↑ 2.3

uniform systems in which every role uses the same model size. The upper-left panel shows that the blue frontier reaches and slightly exceeds the uniform-8B accuracy with less compute and overall outperforms all uniform systems. The lower-left panel shows that uniform systems can incur many more calls with small models or substantially more compute with large models, whereas mixed-size systems reduce both compute and the number of calls. In particular, assigning 4B models to the manager and memory agent reduces the parameter-call budget by 15.8% relative to uniform-8B while improving V* accuracy by 0.52%. The middle panels replace one role at a time in the uniform 8B system and show that sensitivity to backbone scale differs substantially across roles. This shows that different roles benefit from different model sizes rather than a single model size for all roles. Full configurations and TreeBench results showing the same trend are reported in Appendices G.2–G.3.

Structured reasoning states keep additional steps useful. On the full TreeBench test set, we compare the multi-agent baseline and the complete system at maximum reasoning budgets $T \in \{4, 8, 12, 16\}$. The upper-right panel of Figure 4 reports the full-system gains of 3.46–4.20% over the multi-agent baseline at every budget. Here, late recovery captures cases whose earliest prediction is incorrect but whose final answer is correct, whereas retention measures whether an initially correct prediction remains correct. The lower-right panel shows that this advantage is driven mainly by stronger late recovery, which remains 6.9–7.9 points above the baseline. Memory supports these revisions by retaining the connection between newly acquired visual evidence and the unresolved state. Metric definitions and complete paired statistics are provided in Appendix C and Table 18.

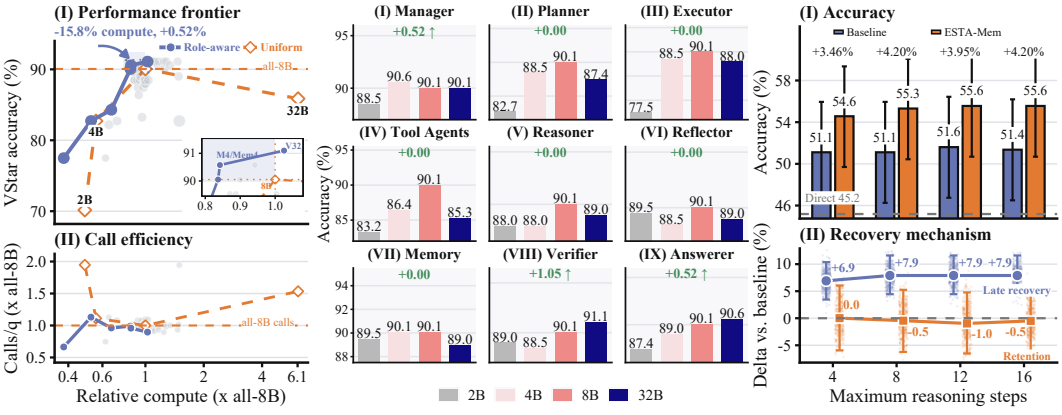

Figure 4: Model scaling and maximum-step analysis. Left: model-allocation efficiency. Middle: role-wise scale sensitivity. Right: step-budget accuracy and recovery.

4.4 RQ3: Does ESTA-Mem Outperform Other Memory Systems?

ESTA-Mem outperforms the alternative memory systems. Table 2 compares our method with four strategies for organizing context and three structured memory methods: A-MEM, Online AMA-Agent, and Mem0-Graph. All memory systems are integrated into our multi-agent baseline, with the same role architecture and base prompts. For the structured memory methods, update schedules and computation remain method-specific. The Representation column in Table 2 highlights how these methods organize past information. Context organization strategies retain raw trajectories, add reflections, or compress history into summaries. Rolling Summary performs best within this group, but remains below ESTA-Mem on both benchmarks. The three structured memory methods instead use linked notes, adaptive memory aggregation, or entity-relation graphs. Each improves on at most one benchmark over the full-context baseline. In contrast, ESTA-Mem explicitly links evidence to candidate answers and delivers role-conditioned views of the resulting state. This allows an agent to access the evidence relevant to its current decision without processing the entire history. The results suggest that effective memory requires not only organizing information, but also making its relevance to current decisions explicit. Appendix B.3 gives the budget-matching rules and adaptation details.

Table 2: Comparison with context organization strategies and structured memory methods.

Memory organization			Matched accuracy		
Category	**Method**	**Representation**	**V⋆ (Δ)**	**TreeBench (Δ)**	**Average (Δ)**
Baseline	Full Context	Raw complete trajectory	87.43 (+0.00)	51.11 (+0.00)	69.27 (+0.00)
Context organization	Generic Reflection	Raw trajectory with generic reflection	85.86 (−1.57)	51.36 (+0.25)	68.61 (−0.66)
Context organization	Delta Summary	Summary with state deltas	86.39 (−1.05)	50.86 (−0.25)	68.63 (−0.65)
Context organization	Flat History	Token-budgeted flat history	87.43 (+0.00)	51.85 (+0.74)	69.64 (+0.37)
Context organization	Rolling Summary	Incremental natural-language summary	88.48 (+1.05)	53.58 (+2.47)	71.03 (+1.76)
Structured memory	A-MEM	Atomic notes with dynamic links	86.39 (−1.05)	52.84 (+1.73)	69.61 (+0.34)
Structured memory	Online AMA-Agent	Adaptive memory aggregation	87.43 (+0.00)	50.37 (−0.74)	68.90 (−0.37)
Structured memory	Mem0-Graph	Graph-structured entity relations	85.86 (−1.57)	54.32 (+3.21)	70.09 (+0.82)
Task-state memory	**ESTA-Mem**	**Typed, role-conditioned state views**	**90.05** (+2.62)	**55.31** (+4.20)	**72.68** (+3.41)

4.5 RQ4: How Do the Memory Components Work Together?

Every component makes a measurable contribution to the complete system. We ablate each component, removing online audit, the state tree, the provenance graph, role-conditioned views, or the candidate ledger once at a time. Table 3 shows that every removal lowers the mean accuracy across V⋆ and TreeBench by 1.05–3.27%. The state tree contributes most clearly on V⋆ by separating recurring observations according to their evidence configurations, while online audit and the candidate ledger contribute most on TreeBench by keeping candidates revisable as new evidence arrives. The provenance graph and role-conditioned views support both benchmarks by preserving evidence identity and delivering the appropriate compiled state to each receiving role. Together, these components maintain what is known, where it came from, and how the answer has evolved, while supporting online updates and selective delivery. Details are in Appendix E.2 and Table 19.

Table 3: Component ablation results.

Component setting	Matched accuracy		
Variant	**V⋆ (Δ)**	**TreeBench (Δ)**	**Average (Δ)**
Full ESTA-Mem	**90.05** (+0.00)	**55.31** (+0.00)	**72.68** (+0.00)
w/o Online Audit	89.53 (−0.52)	53.09 (−2.22)	71.31 (−1.37)
w/o State Tree	87.96 (−2.09)	55.31 (+0.00)	71.63 (−1.05)
w/o Provenance Graph	87.43 (−2.62)	52.35 (−2.96)	69.89 (−2.79)
w/o Role-conditioned Views	87.96 (−2.09)	50.86 (−4.44)	69.41 (−3.27)
w/o Candidate Ledger	90.05 (+0.00)	53.09 (−2.22)	71.57 (−1.11)

4.6 RQ5: Which Tasks and Reasoning Conditions Benefit Most from ESTA-Mem?

ESTA-Mem helps most when decisions require evidence integration across long trajectories. To identify when and how the gain emerges, we conduct three matched analyses. First, we divide all 596 V⋆ and TreeBench samples into direct readout (answerable from one observation) and evidence integration (which requires multiple steps or image regions). The left panel of Figure 5 shows gains of 5.12% on the 332 evidence-integration samples and 1.89% on direct readout. Second, we use baseline reasoning steps, model calls, and input tokens to measure each sample's trajectory-state pressure. The center panel shows a 9.82% gain under high pressure and a positive continuous association of $\beta = +11.9\%$ per unit with $p = 0.006$. Third, on the same 332 evidence-integration samples, the right panel compares corrections and regressions across memory representations. Our method makes 28

corrections and gains 5.12%, while Reflection, Summary, and Flat History remain near zero. Together, these analyses show that ESTA-Mem helps most when demanding trajectories require multiple pieces of evidence. Online audit, state tree, provenance graph, candidate ledger, and role-conditioned views route new evidence to the affected candidate. Appendices C and B.3 provide more details.

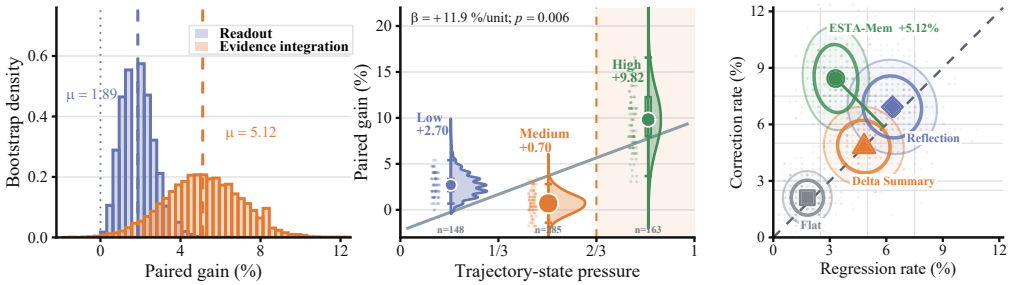

Figure 5: Mechanism localization. Gains are largest on evidence-integration and high-pressure trajectories, while basic memory methods remain near zero. Analyses are post-hoc and non-causal.

4.7 RQ6: How Does Compilation Improve Delivery, Utility, and Timeliness?

Compilation reduces the delivered context while preserving decision-relevant structure. On V⋆ and TreeBench, we compare the full compiled memory text with the text delivered to agents and compare the number of states before and after merging redundant states. We also compare accuracy according to whether explicit support, accepted observations, and verified candidates are present. Tables 4 and 5 show that role-conditioned views expose only 7.12% of the compiled text and state folding reduces the state count by 43.71%, while the three retained signals are associated with accuracy gains of 11.00–15.75 points. Compilation therefore reduces redundant delivery while preserving the evidence relations needed for grounded and reliable decisions.

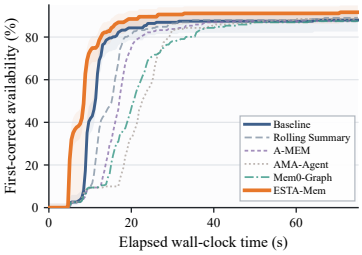

Figure 6: First-correct availability on V⋆. Our method rises earlier and attains the highest coverage.

The compiled state also leads to the earlier appearance of correct answers. On V⋆, we reconstruct the sequence of candidate answers and their wall-clock timestamps from trajectories for the Multi-Agent Baseline, Rolling Summary, A-MEM, AMA-Agent, Mem0-Graph, and ESTA-Mem. Figure 6 shows that ESTA-Mem produces an earlier rise in first-correct availability and reaches the highest coverage. The corresponding TreeBench analysis shows the same trend. Together, these compelling results show that compilation delivers less text, retains informative evidence structure, and forms a correct state earlier. More details and further experiments are reported in Appendices D, H, and G.

Table 4: Compiled-memory compression and state folding.

Dataset	Delivered text			Folded state		
Dataset	**Full tok.**	**View tok.**	**Tok. red.**	**State pre**	**State post**	**State red.**
V⋆	6,226.0	372.6	**94.02%**	772	485	**37.18%**
TreeBench	8,237.0	621.0	**92.46%**	1,914	1,027	**46.34%**
Overall	**7,575.0**	**539.2**	**92.88%**	**2,686**	**1,512**	**43.71%**

Table 5: Terminal-signal reliability.

Terminal signal			Conditional accuracy		
Signal	n_+	n_-	**Acc.+**	**Acc.−**	**Gap**
Supports	339	257	72.86	57.98	**+14.88**
Accepted evidence	368	228	70.65	59.65	**+11.00**
Verified candidate	419	177	71.12	55.37	**+15.75**

5 Conclusion

We introduced ESTA-Mem, an ***Event-Sourced, Typed, and Auditable*** memory system for multi-agent visual reasoning. It reframes agent memory from storing past interactions to maintaining the evolving state of a shared decision process. By preserving how observations, candidate answers, and reviews relate across agents and reasoning steps, ESTA-Mem offers a general way for memory to guide later reasoning. The results in various high-resolution tasks demonstrate the practical value of this direction. More broadly, our work establishes task-state compilation as a new design principle and provides a foundation for building more reliable, interpretable, and scalable multimodal agent systems.

Reproducibility statement

Our experiments use publicly available pretrained models with frozen parameters. We will soon release our code with implementation details, including agent prompts and inference settings. Appendix A describes the implementation and inference settings. Appendix B details the benchmark protocols and memory comparisons, while Appendix C defines the evaluation metrics and statistical procedures.

AI use statement

Generative AI tools were used to assist with manuscript organization, English translation and rewriting of author-provided technical drafts, LaTeX equation and table formatting, figure-layout code, and consistency checks across the paper. They were not used to generate experimental measurements or replace the authors' execution and verification of the reported experiments. The authors reviewed all AI-assisted text, mathematical notation, tables, figures, and claims against the frozen implementation and result files, and take responsibility for the complete content of the submission.

References

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*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025b.

Zhixi Cai, Fucai Ke, Kevin Leo, Sukai Huang, Maria de la Banda, Peter Stuckey, and Hamid Rezatofighi. Mata: A trainable hierarchical automaton system for multi-agent visual reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 139116–139132, 2026.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*, 2025.

Xinpeng Ding, Jianhua Han, Hang Xu, Wei Zhang, and Xiaomeng Li. Hilm-d: Enhancing mllms with multi-scale high-resolution details for autonomous driving. *International Journal of Computer Vision*, 133(8):5379–5395, 2025.

Bernhard Hommel. Event files: Evidence for automatic integration of stimulus-response episodes. *Visual cognition*, 5(1-2):183–216, 1998.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, and Guohai Xu. Deepeyesv2: Toward agentic multimodal model. In *International Conference on Learning Representations*, volume 2026, pp. 114851–114872, 2026.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Steven Yau, Zijuan Lin, Liyang Zhou, et al. Metagpt: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, volume 2024, pp. 23247–23275, 2024.

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

Zhengda Jin, Bingbing Wang, Jing Li, Ruifeng Xu, and Min Zhang. Mitigating provenance-role collapse in long-term agents via typed memory representation. *arXiv preprint arXiv:2605.25869*, 2026.

Marcia K Johnson, Shahin Hashtroudi, and D Stephen Lindsay. Source monitoring. *Psychological bulletin*, 114(1):3, 1993.

Daniel Kahneman, Anne Treisman, and Brian J Gibbs. The reviewing of object files: Object-specific integration of information. *Cognitive psychology*, 24(2):175–219, 1992.

Kartik Kuckreja, Muhammad Sohail Danish, Muzammal Naseer, Abhijit Das, Salman Khan, and Fahad Shahbaz Khan. Geochat: Grounded large vision-language model for remote sensing. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27831–27840. IEEE, 2024.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, volume 2026, pp. 76722–76746, 2026.

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

Geng Li, Jinglin Xu, Yunzhen Zhao, and Yuxin Peng. Dyfo: A training-free dynamic focus visual search for enhancing lmms in fine-grained visual understanding. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9098–9108. IEEE, 2025.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pp. 19730–19742. PmLR, 2023.

Liupeng Li, Haoqian Kang, Zhenyu Lu, Jinpeng Wang, Bin Chen, Ke Chen, and Yaowei Wang. Cvsearch: Empowering multimodal llms with cognitive visual search for high-resolution image perception. *arXiv preprint arXiv:2605.23655*, 2026a.

Yangfu Li, Hongjian Zhan, Jiawei Chen, Yuning Gong, Qi Liu, and Yue Lu. Deepscan: A training-free framework for visually grounded reasoning in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19155–19164, 2026b.

Zhuofeng Li, Haoxiang Zhang, Seungju Han, Sheng Liu, Jianwen Xie, Yu Zhang, Yejin Choi, James Y Zou, and Pan Lu. In-the-flow agentic system optimization for effective planning and tool use. In *International Conference on Learning Representations*, volume 2026, pp. 50524–50570, 2026c.

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

Ruixun Liu, Bowen Fu, Jiayi Song, Kaiyu Li, Wanchen Li, Lanxuan Xue, Hui Qiao, Weizhan Zhang, Deyu Meng, and Xiangyong Cao. Zoomearth: Active perception for ultra-high-resolution geospatial vision-language tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 34877–34888, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

Haozhan Shen, Kangjia Zhao, Tiancheng Zhao, Ruochen Xu, Zilun Zhang, Mingwei Zhu, and Jianwei Yin. Zoomeye: Enhancing multimodal llms with human-like zooming capabilities through tree-based image exploration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6613–6629, 2025.

Sagar Soni, Akshay Dudhane, Hiyam Debary, Mustansar Fiaz, Muhammad Akhtar Munir, Muhammad Sohail Danish, Paolo Fraccaro, Campbell D Watson, Levente J Klein, Fahad Shahbaz Khan, et al. Earthdial: Turning multi-sensory earth observations to interactive dialogues. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14303–14313. IEEE, 2025.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. *Advances in Neural Information Processing Systems*, 38:8222–8251, 2026.

Fengxiang Wang, Hongzhen Wang, Zonghao Guo, Di Wang, Yulin Wang, Mingshuo Chen, Qiang Ma, Long Lan, Wenjing Yang, Jing Zhang, et al. Xlrs-bench: Could your multimodal llms understand extremely large ultra-high-resolution remote sensing imagery? In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14325–14336. IEEE, 2025a.

Fengxiang Wang, Mingshuo Chen, Yueying Li, Di Wang, Haotian Wang, Zonghao Guo, Zefan Wang, Shan Boqi, Long Lan, Yulin Wang, et al. Geollava-8k: Scaling remote-sensing multimodal large language models to 8k resolution. *Advances in Neural Information Processing Systems*, 38: 159185–159218, 2026a.

Fengxiang Wang, Mingshuo Chen, Yueying Li, Yajie Yang, Yifan Zhang, Long Lan, Xue Yang, Hongda Sun, Yulin Wang, Di Wang, et al. Geoeyes: On-demand visual focusing for evidence-grounded understanding of ultra-high-resolution remote sensing imagery. *arXiv preprint arXiv:2602.14201*, 2026b.

Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Sule Bai, Zijian Kang, Jiashi Feng, Zhuochen Wang, et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and method. In *International Conference on Learning Representations*, volume 2026, pp. 148769–148794, 2026c.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 7907–7915, 2025b.

Yifan Wang, Peiming Li, Shiyu Li, Zhiyuan Hu, Xiaochen Yang, Wenming Yang, Yang Tang, and Zheng Wei. Focus when necessary: Adaptive routing and collaborative grounding for training-free visual grounding. *arXiv preprint arXiv:2606.16158*, 2026d.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13084–13094. IEEE, 2024.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, et al. Autogen: Enabling next-gen llm applications via multi-agent conversation. *arXiv preprint arXiv:2308.08155*, 2023.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Xuan Yu, Dayan Guan, and Yanfeng Gu. Zoom-refine: Boosting high-resolution multimodal understanding via localized zoom and self-refinement. *arXiv preprint arXiv:2506.01663*, 2025.

Ziyun Zeng, Hang Hua, Bocheng Zou, Mu Cai, Rogerio Feris, and Jiebo Luo. Mementogui: Learning agentic multimodal memory control for long-horizon gui agents. *arXiv preprint arXiv:2605.18652*, 2026.

Guibin Zhang, Muxin Fu, Kun Wang, Frank Wan, Miao Yu, and Shuicheng Yan. G-memory: Tracing hierarchical memory for multi-agent systems. *Advances in Neural Information Processing Systems*, 38:12988–13018, 2026.

Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Chain-of-focus: Adaptive visual search and zooming for multimodal reasoning via rl. *arXiv e-prints*, pp. arXiv–2505, 2025a.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images, 2025b.

YiFan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, et al. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? In *International Conference on Learning Representations*, volume 2025, pp. 89655–89701, 2025c.

Beidi Zhao, Yaoqi Chen, Yuru Feng, Menghao Li, Qianxi Zhang, Baotong Lu, Jianan Lu, Zhirui Wang, Xinjiang Wang, Shusen Xu, et al. Mesa: Task-adaptive multi-structure evidence selection for long-horizon agent memory. *arXiv preprint arXiv:2608.10108*, 2026a.

Yujie Zhao, Boqin Yuan, Junbo Huang, Haocheng Yuan, Zhongming Yu, Haozhou Xu, Lanxiang Hu, Abhilash Shankarampeta, Zimeng Huang, Wentao Ni, et al. Ama-bench: Evaluating long-horizon memory for agentic applications. *arXiv preprint arXiv:2602.22769*, 2026b.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, and Chao Shen. Deepeyes: Incentivizing" thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, volume 2026, pp. 126775–126798, 2026.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internvl3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

APPENDIX

A System Implementation **15**

A.1 Agent Roles and Response Formats 15

A.2 Tool Calls and Crop Checks 15

A.3 Inference Settings and Hardware 16

A.4 Event Recording and Memory Updates 16

A.5 Memory Views and Delivery Conditions 16

A.6 Execution Order 17

B Benchmark Protocols and Memory Comparisons **17**

B.1 Benchmarks and Evaluation Metrics 17

B.2 XLRS-Bench Lite Task Breakdown 19

B.3 Memory Reproduction and Adaptation Protocols 19

C Metrics and Statistical Comparisons **20**

D When Correct Candidates Become Available **21**

E Paired Accuracy and Component Analyses **24**

E.1 Reasoning Budgets on TreeBench 24

E.2 Component Ablations 25

F Structural Reliability of Compiled Memory **25**

G Model Allocation and Computational Cost **26**

G.1 Memory-Call-Controlled Comparison 26

G.2 Using a Smaller MemoryAgent 27

G.3 Role-Wise Model Allocation 27

G.4 MemoryAgent Activity and Role-Level Efficiency 28

H Compact Delivery and Recoverable Memory Relations **30**

H.1 Source-State Binding Recovery 30

I Memory Growth and Observation Identity **31**

J End-to-End Cases **32**

A System Implementation

A.1 Agent Roles and Response Formats

Table 6 summarizes the responsibilities and required outputs that define each agent's interface. Each role follows a fixed instruction and response schema, while each call provides the information needed for the role's current task. This information may include the question, a visual target, the relevant image or crop, a requested operation, a proposed answer, and associated record identifiers. These formats allow outputs to be checked and passed directly between roles in the multi-agent system.

Table 6: Agent responsibilities and required outputs within a visual reasoning step.

Role	Responsibility	Required output
Manager	Identify task-relevant visual targets	Ordered target-instance list.
Planner	Choose an inquiry or propose an answer	Action request or candidate for verification.
Executor	Translate the inquiry into executable calls	Tool arguments or a focused *reasoner* call.
Tool agents	Execute requested evidence operations	Visual assets, observations, or derived results.
Reasoner	Infer a claim from the specified evidence	Claim and supporting evidence IDs.
Verifier	Check evidence and candidate consistency	Evidence and candidate decisions, with feedback.
Visual reflector	Assess low-confidence crop proposals	Crop assessment and next inspection region.
Memory agent	Propose updates from observations and feedback	Admit, dispute, link, supersede, or compress.
Answerer	Answer the question from the available information	Answer option or free-form response.

The following subsections connect these interfaces to execution. They describe the available tool calls and crop checks, the default inference settings, the conversion of agent outputs into typed events, replay into compiled memory, and its delivery to each role in the multi-agent system.

A.2 Tool Calls and Crop Checks

Table 7 lists the tool calls available to the Executor. The Planner states what information is needed, and the Executor turns that request into one of these calls. Each call has a fixed set of inputs and returns a record linked to the source image, crop, or earlier observations it uses. The system checks the required identifiers and source assets before the result is added to the execution record.

Table 7: Inputs and recorded outputs of the available tool calls. Parenthesized inputs are optional.

Tool call	Required input	Returned record
zoom_in	Image ID, entity, target ID, (location hint)	Source-linked crop IDs and pixel boxes.
inspector	Entity-patch ID, subject, attribute query	Patch-local attribute observation.
geometry_relation	Reference-patch ID, target-patch ID	Target direction in the image plane.
spatial_perspective	Subject/object patch IDs, (subject prior)	Subject-centered direction of the object.
depth_relation	Two patch IDs	Camera-relative depth relation.
count_instances	Image ID, set query	Count and localized instance boxes.
cross_image_correspondence	Source-patch ID, target-image ID, (query)	Matched target-image patches and scores.
calculator	Expression and observation IDs	Derived value linked to parent records.

The zoom model returns boxes in normalized $[0, 1000]$ coordinates. Each box is clipped to this range, mapped to source-image pixels, and checked for nonzero area. Stored crops keep the source image identifier and pixel box, linking later observations to the original image region.

Thresholds route *geometry_relation*. It uses a marked visual comparison if the intersection covers at least 0.80 of the smaller box, or if one region spans over 0.25 of the image width or height while the box-area ratio exceeds 9 and overlap along the queried axis exceeds 0.25. Other pairs use box-center direction. The marked crop uses 0.12 relative context padding with a minimum pixel margin.

A.3 Inference Settings and Hardware

Unless a model-allocation experiment specifies otherwise, every VLM-based role uses Qwen3-VL-8B. Experiments run on eight NVIDIA RTX A6000 GPUs with 48 GB of memory each. Table 8 summarizes the default inference and image-processing settings used throughout our experiments.

Table 8: Default inference and image-processing configuration.

Setting	Value
Backbone	Qwen3-VL-8B
Precision and attention	bfloat16 with SDPA
Decoding	Greedy decoding with KV caching
Role response limits	Role-specific fixed caps, including Manager 128, Planner 256, Executor and MemoryAgent 768, Reasoner 512, and ordinary answer response 32 tokens
Episode and seed	$T = 8$ control steps, random seed 42
Visual reflector	Confidence threshold 0.9
Image processing	Aspect-ratio-preserving Lanczos resize, maximum dimension 4096 pixels
Worker cache	Up to 64 resized images per worker

We resize images whose longest side exceeds 4096 pixels to provide an overview. Regions selected in this overview are mapped back to the original image and cropped at full resolution for details.

A.4 Event Recording and Memory Updates

The system converts each record directly into one event. Every event retains its identifier, producing role, step, creation status, content, and explicit parent or image references. Table 9 gives the mapping.

Table 9: Deterministic bridge from chronological records to event types.

Raw record kind	Event type	Preserved content
plan	Plan	Selected action or terminal proposal.
candidate	Candidate	Task answer and production step.
crop	Asset	Source image, box, target, acceptance state.
Successful tool or Reasoner observation	Evidence	Subject, predicate, value, confidence, source IDs.
Other action results	Result	Output, error status, parent action.
Candidate verification	Review	Evidence and candidate decisions, with feedback.
Multi-agent system and audit records	Control	Original payload and parent references.

Then, the event types in Table 9 form a chronological log that is replayed to build the state tree, provenance graph, and candidate ledger. Management, reflection, feedback, protocol, and audit records enter this log as Control events, while task observations remain a separate event type. When an audit is triggered, the memory agent proposes one transaction drawn from five update types: *Admit* and *Dispute* mark an existing observation as accepted or disputed; *Link* adds a typed relation between existing items. *Supersede* identifies the currently preferred observation for the same object and property and *Compress* groups repeated attempts under an existing state. The available links are *AddressesTask*, *BindsTaskReferent*, *GroundedIn*, *Supports*, and *ConflictsWith*, representing task relevance, object identity, visual source, support, and disagreement. Before these updates enter the compiled memory, every reference and item type is checked against the current state, and the transaction is applied atomically. The audit event keeps the transaction and its triggering range, and replay applies the validated updates to all three memory structures while retaining the original observations and source links. The task roles and Verifier continue to check visual correctness.

A.5 Memory Views and Delivery Conditions

When requested, the multi-agent system adds a read-only view of the latest compiled state to a role's ordinary task input. Each view selects the records needed by that role while preserving its original actions and output format. Table 10 lists these records and how the role uses them.

All four views come from the same compiled memory, keeping task, source, and review references consistent across roles. The ordinary input provides the question, images, current request, or submitted candidate, which the receiving role combines with its memory view for the current call.

Table 10: Role-specific memory views used by the multi-agent system.

Role	Memory records shown	How the role uses them
Planner	Current state, task and source bindings, unresolved conflicts, reviewed counter-observations, and repeated attempts	Use current progress to choose the next inquiry or submit a candidate for review.
Executor	Latest plan, referenced task and source bindings, accepted image-region identifiers, and related revision links	Resolve the objects, sources, and image regions required by the selected action.
Verifier	Observation identifiers, source and task bindings, effective review states, and support or conflict relations	Compare the submitted candidate with linked observations and recorded conflicts.
Answerer	Accepted source-linked observations, unresolved conflicts, reviewed counter-observations, and typed option relations	Use the compiled task state and ordinary input to return the requested answer.

A.6 Execution Order

Algorithm 1 describes one memory refresh. A record signature skips repeated calls when the task state has not changed. Otherwise, the current record is converted into typed events and replayed to rebuild the compiled memory. When an audit is needed, the MemoryAgent receives one packet, and its validated transaction is recorded and replayed without another audit.

Algorithm 1 Memory refresh by deterministic replay after new agent outputs

function REFRESH-MEMORY($R, M, h, Q_\star, t$)
  $h' \leftarrow$ MEMORYSIGNATURE(R)
  if $h' = h$ **then**
    return (M, h)
  end if
  $\tau \leftarrow \Phi(R)$; $M \leftarrow \mathcal{C}_\Gamma(\tau, Q_\star)$
  if $B \leftarrow$ AUDITPACKET($R, M, Q_\star$) is non-empty **then**
    $a \leftarrow V_M(B, \pi_\mu(B))$
    append audit record a to R
    $\tau \leftarrow \tau \| \phi(a, Q_\star)$; $M \leftarrow \mathcal{C}_\Gamma(\tau, Q_\star)$
  end if
  return (M, h')
end function

Algorithm 2 places this refresh routine inside a complete multi-agent episode. Each task step begins by incorporating records produced in the preceding step. A submitted candidate is incorporated before verification, so the Verifier receives the corresponding memory view. Accepted candidates follow the original return path, while rejected candidates add feedback for the next refresh or the original fallback path.

B Benchmark Protocols and Memory Comparisons

B.1 Benchmarks and Evaluation Metrics

We evaluate on four benchmark families that emphasize complementary sources of difficulty in high-resolution visual reasoning. All controlled comparisons use the same sample IDs within each benchmark. Unless specified otherwise, a prediction is correct when its normalized answer matches the benchmark label, and accuracy is the percentage of correctly answered questions. Subgroup accuracies are computed on the questions assigned to that subgroup. The four-benchmark average in Table 1 gives equal weight to the overall accuracies on V*Bench, TreeBench, HRBench-4K, and HRBench-8K. Each benchmark therefore contributes equally to the reported average.

V*Bench. V*Bench (Wu & Xie, 2024) evaluates whether a model can locate and inspect small relevant targets in visually crowded, high-resolution natural images. It contains 191 human-curated questions based on 191 images from SA-1B, with an average image resolution of 2246×1582. The

Algorithm 2 ESTA-Mem within one frozen multi-agent episode

Require: Instance $x = (q, I, \Omega)$, maximum budget T, and frozen task roles
Ensure: Prediction $\widehat{y}$

$R \leftarrow$ INITIALIZERECORD(x)
run the one-shot Manager and append its task targets to R
construct $Q_\star$ from the question stem and copied Manager targets
$M \leftarrow \psi(Q_\star)$ and $h \leftarrow \varnothing$ ▷ no record state refreshed yet
for $t = 1, \ldots, T$ **do**
    $(M, h) \leftarrow$ REFRESH-MEMORY$(R, M, h, Q_\star, t)$
    obtain decision d_t from the frozen task loop, using any scheduled role view
    append the plan record for d_t to R
    if d_t requests an inquiry action **then**
        Executor instantiates d_t as a Reasoner call or validated tool-call batch
        execute the call(s) and append their typed outputs to R
        continue
    end if
    append the submitted candidate y_t to R
    $(M, h) \leftarrow$ REFRESH-MEMORY$(R, M, h, Q_\star, t)$
    $V_t^V \leftarrow \mathcal{P}_V(M)$; obtain and append $v_t \leftarrow \pi_V(q, I, y_t, \Omega, V_t^V)$
    if v_t accepts y_t **then**
        return y_t ▷ unchanged verified early return
    end if
    append the rejection feedback to R
    if the rejected state still contains an unaccepted crop **then**
        $(M, h) \leftarrow$ REFRESH-MEMORY$(R, M, h, Q_\star, t)$
        return the original Answerer fallback from the current memory
    end if
end for
$(M, h) \leftarrow$ REFRESH-MEMORY$(R, M, h, Q_\star, T)$
return the prediction under the native terminal protocol with $V_T^A \leftarrow \mathcal{P}_A(M)$

benchmark contains two subsets. Attribute Recognition has 115 questions about properties such as an object's color or material. Spatial Relationship Reasoning has 76 questions about the relative position of two objects. The released evaluation uses manually reviewed multiple-choice answers, with four choices for converted open-ended questions and two choices for binary questions. We evaluate all 191 questions and report Attribute, Relation, and overall accuracy. The overall score pools the two subsets in their released 115-to-76 proportion for the benchmark-level result.

TreeBench. TreeBench (Wang et al., 2026c) evaluates visual grounded reasoning in dense real-world scenes where the queried targets may be small, visually similar, or involved in second-order spatial relations. Its 405 human-verified question-answer pairs are constructed from high-density SA-1B images and include annotated target regions associated with each question. The benchmark organizes ten competencies into two protocols. The 149 Perception questions cover Attributes, Material, Physical State, Object Retrieval, and OCR-Integrated Question Answering. The 256 Reasoning questions cover Perspective Transform, Ordering, Contact and Occlusion, Spatial Containment, and Comparison. We report Perception, Reasoning, and overall accuracy. The overall score pools all 405 questions across the released 149-to-256 protocol split.

HRBench-4K and HRBench-8K. HRBench (Wang et al., 2025b) is designed to test fine-grained perception at resolutions beyond those of conventional multimodal benchmarks. It contains 200 manually selected 8K images from DIV8K and the Internet. Crops centered on the question-relevant content form the corresponding 4K version, allowing the same capabilities to be evaluated at two resolutions. Each resolution contains 100 Fine-grained Single-instance Perception questions and 100 Fine-grained Cross-instance Perception questions. FSP asks for properties of a specific object, including fine visual attributes such as color and material. FCP requires comparison of multiple objects, with a focus on their relative spatial positions. Questions are multiple choice. The official cyclic-permutation protocol rotates the option order and averages correctness over the resulting presentations to reduce option-position bias. The four rotations yield 800 option-order presentations per resolution from the same 200 questions. We report FSP, FCP, and overall accuracy under the same protocol at both 4K and 8K resolution.

XLRS-Bench Lite. XLRS-Bench (Wang et al., 2025a) evaluates multimodal models on real ultra-high-resolution remote-sensing imagery. The full benchmark contains 1,400 images with an average resolution of 8500×8500, together with 45,942 human-verified annotations spanning VQA, visual grounding, and detailed captioning. Following the official Lite multiple-choice protocol, we evaluate all 13 VQA subtasks: eight perception tasks and five reasoning tasks. The benchmark reports detailed captioning and visual grounding as separate generative and localization tracks. The eight perception tasks are Overall Counting abbreviated as OC, Regional Counting as RC, Overall Land Use Classification as OLUC, Regional Land Use Classification as RLUC, Object Classification as OCC, Object Color as OCL, Object Motion State as OMS, and Object Spatial Relationship as OSR. The five reasoning tasks are Anomaly Detection and Interpretation abbreviated as AD, Environmental Condition Reasoning as ECR, Counting with Complex Reasoning as CCR, Route Planning as RP, and Regional Counting with Change Detection as RCCD. We report accuracy for every task. Table 11 reports all 13 task scores and their equally weighted macro average.

B.2 XLRS-Bench Lite Task Breakdown

Table 11 reports accuracy for every task in the official XLRS-Bench Lite multiple-choice VQA protocol and their macro average. The 13 columns cover all eight perception tasks and all five reasoning tasks in this track. Published results provide external references, while Direct, the Multi-Agent Baseline, and ESTA-Mem are evaluated with the same configuration used in Table 1.

Table 11: XLRS-Bench Lite accuracy on all 13 tasks in its multiple-choice VQA protocol and their macro average. Green boxes indicate improvements over the Multi-Agent Baseline.

Method	**Perception**								**Reasoning**					**Macro**
	OC	**RC**	**OLUC**	**RLUC**	**OCC**	**OCL**	**OMS**	**OSR**	**AD**	**ECR**	**CCR**	**RP**	**RCCD**	
Published task-level results														
GPT-4o (Hurst et al., 2024)	25.0	32.0	15.0	66.0	9.5	11.3	11.7	24.6	73.0	73.0	25.0	35.0	20.0	32.4
LLaVA-OneVision-7B (Li et al., 2024)	25.0	38.0	2.0	69.5	35.9	35.3	65.0	25.2	76.0	83.0	36.0	24.0	43.3	42.9
InternVL3-8B (Zhu et al., 2025)	40.0	39.0	10.0	71.5	44.5	30.8	65.0	25.2	77.0	82.0	50.0	36.0	21.7	45.6
GeoChat (Kuckreja et al., 2024)	16.7	29.0	2.0	23.0	21.1	16.8	35.0	24.2	33.0	43.0	21.0	10.0	24.0	23.0
EarthDial (Soni et al., 2025)	18.3	42.0	1.0	36.0	31.3	31.0	65.0	24.8	62.0	71.0	50.0	43.0	48.3	40.3
ZoomEarth (Liu et al., 2026)	30.0	38.0	8.0	63.0	35.1	33.9	65.0	31.4	66.0	73.0	40.0	22.0	28.3	41.1
GeoLLaVA-8K (Wang et al., 2026a)	26.7	38.0	49.0	69.0	41.6	31.6	65.0	35.0	67.0	78.0	52.0	66.0	50.0	51.5
DeepEyes (Zheng et al., 2026)	31.7	41.0	33.0	75.0	41.6	38.1	68.3	31.4	70.0	78.0	46.0	46.0	50.0	50.0
GeoEyes (Wang et al., 2026b)	38.3	40.0	24.0	73.5	59.5	66.1	68.3	32.2	77.0	80.0	50.0	56.0	40.0	54.2
Our evaluations (Qwen3-VL-8B)														
Direct	23.33	49.00	2.00	67.00	38.12	35.38	66.67	25.60	65.00	80.00	47.00	47.00	50.00	45.85
Multi-Agent Baseline	13.33	44.00	25.00	76.00	44.88	41.25	66.67	30.60	76.00	79.00	49.00	39.00	45.00	48.44
ESTA-Mem (Ours)	**20.00**	**47.00**	25.00	**80.50**	**46.50**	**42.38**	63.33	30.40	75.00	79.00	**50.00**	35.00	45.00	**49.16**

The task breakdown shows whether the aggregate XLRS result is concentrated in one category or shared across different capabilities. The matched rows in Table 11 show improvements on six tasks and ties on three, with gains spanning both perception and reasoning. The positive macro average therefore reflects changes across the task taxonomy and extends the memory effect beyond the benchmarks used for detailed trajectory analysis.

B.3 Memory Reproduction and Adaptation Protocols

We compare all memory alternatives within the same multi-agent system. Each method receives agent records and visual assets from its own run on the current question, while the task roles, visual operations, reasoning budget, stopping rule, decoding, and answer selection remain fixed. Memory starts fresh for every question. Reflection and Delta Summary run at the same decision points with the same generation limit, and Flat History matches the length of the delivered role view. Rolling Summary and the adapted memory systems retain their own update and retrieval procedures. Table 12 organizes the comparison by retained state, online update, retrieval rule, and readout. It shows how context controls, adapted structured-memory systems, and ESTA-Mem turn execution records into task-role inputs through their respective memory organizations. Table 22 reports a separate comparison with a controlled memory-call budget, including accuracy and runtime in that setting.

Table 12: Memory mechanisms retained in the within-episode comparisons. Each method uses the same task agents but keeps its own update and retrieval procedure.

Method	Retained state	Online update	Retrieval rule	Delivered readout
Context organization				
Reflection generic audit trigger-matched	Raw event prefix plus one generic reflection record	Frozen audit trigger over current-sample event packets	Read complete prefix without constructing persistent state	Native task context plus the generic reflection readout
Delta Summary textual state trigger-matched	Recursive summary plus the latest recorded state delta	Same trigger and generation cap as the Reflection control	Select the latest rewritten summary and current delta	Native task context plus summary and latest state delta
Flat History ordered context token-matched	Chronological events in an ordered flat history serialization	Append each event, preserve its order, and skip compression	Truncate ordered events to the same role-view budget	Token-matched flat history at the same memory-read point
Rolling Summary recursive textual memory control	One sample-local natural-language recursive summary	Update the summary after each immutable event is appended	Replace the prior summary and read the latest version	Latest recursive natural-language shared summary
Structured memory				
A-MEM linked-note memory system (2026)	Atomic memory notes, semantic neighbors, and dynamic links	Adapt each event into note creation or note evolution	Find MiniLM semantic neighbors, then expand links	Retrieved notes together with their linked neighbors
Online AMA-Agent adaptive aggregation system (2026b)	State/causal graph online memory sufficiency checks	Build online graphs from temporally adjacent events	Retrieve semantic graph neighbors and keyword matches	Seed events with graph and keyword retrieved neighbors
Mem0-Graph entity-relation graph system (2025)	Extracted facts, entity relations, and conflict records	Apply *Add*/*Update*, *Delete*/*None* within each trajectory	Retrieve semantic facts, then expand the entity graph	Retrieved facts together with their incident relations
Task-state memory				
ESTA-Mem typed relational state memory (ours)	Typed event log, source–state graph, tree, and ledger	Replay new events, then audit newly recorded observations	Replay synchronized state and project by receiving role	Read-only view of task-relevant source and state relations

C Metrics and Statistical Comparisons

Each question trajectory is the independent unit, and all comparisons pair predictions for the same question. Prefixes, events, graph edges, model calls, and audit operations are within-trajectory measurements, not independent replicates. Fix/Reg counts corrections and regressions on paired questions. Within each benchmark, their difference divided by the number of questions gives the accuracy gain, and paired differences are assessed with a two-sided exact McNemar test. For component ablations, the cross-benchmark gain is the unweighted mean of the two benchmark-level gains, with stratified paired intervals and tests. We apply Holm correction within the reasoning-budget and component-ablation comparisons. TSPI p values come from within-dataset permutations that preserve workload distributions; reported regression estimates use sample-clustered standard errors.

Outcome-independent mechanism groups. The mechanism view covers all 596 V* and TreeBench examples. Direct readout combines V* direct-attribute questions with TreeBench Perception categories, whereas evidence integration combines V* relative-position questions with TreeBench Reasoning categories. This outcome-independent mapping is fixed from benchmark taxonomy before method outputs are loaded and is used only to localize gains post hoc. Because the interaction interval crosses zero, the subgroup contrast is reported descriptively rather than interpreted causally.

Trajectory dynamics. Let $z_{i,0}^{(m)}$ denote correctness at the first answer-bearing state and $z_{i,T}^{(m)}$ denote terminal correctness under method m. We measure recovery and retention as

$$R_T^{(m)} = \frac{1}{N}\sum_{i=1}^{N}\left(1 - z_{i,0}^{(m)}\right) z_{i,T}^{(m)}, \qquad K_T^{(m)} = \frac{\sum_{i=1}^{N} z_{i,0}^{(m)} z_{i,T}^{(m)}}{\sum_{i=1}^{N} z_{i,0}^{(m)}}. \tag{10}$$

Recovery averages over all N questions, whereas retention conditions on initially correct cases; the two rates are therefore neither additive nor complementary. Since methods may emit their first answer at different steps, they summarize method-specific trajectories rather than a shared initial state.

Trajectory-state pressure. For sample i in dataset d_i, let s_i, n_i^{call}, and ℓ_i denote the baseline trajectory's reasoning steps, VLM calls, and input tokens. Their within-dataset empirical CDFs define

$$\text{TSPI}_i = \frac{1}{3}\left[\widehat{F}_{\text{step}}^{(d_i)}(s_i) + \widehat{F}_{\text{call}}^{(d_i)}(n_i^{\text{call}}) + \widehat{F}_{\text{tok}}^{(d_i)}(\ell_i)\right] \in [0,1]. \tag{11}$$

Each empirical CDF maps a raw count to its within-dataset rank. Their mean is high when the baseline already requires many steps, calls, and tokens. TSPI uses neither ESTA-Mem outputs nor answer correctness, and therefore indexes baseline trajectory pressure rather than a memory failure.

Continuous TSPI association. Let $b_i, m_i \in \{0,1\}$ indicate matched baseline and ESTA-Mem correctness. Their paired change $g_i = m_i - b_i \in \{-1,0,1\}$ follows the dataset-specific model

$$g_i = \alpha_{d_i} + \beta\,\text{TSPI}_i + \varepsilon_i. \tag{12}$$

We report 100β in points per TSPI unit and assess it by within-dataset permutation. Because TSPI spans $[0,1]$, 100β is the fitted change across its full range, not the gain from one extra reasoning step. It describes an association with baseline workload rather than a causal effect of added computation.

Auditor activity and latency. Let C count MemoryAgent calls, W calls committing a non-empty write, Q questions, and Q_W questions containing at least one write. For dataset d, $\bar{t}_{\text{mem}}^{(d)}$ and $\bar{t}_{\text{e2e}}^{(d)}$ denote mean total MemoryAgent and end-to-end time per question. We report

$$r_{\text{call}} = \frac{W}{C}, \qquad r_{\text{sample}} = \frac{Q_W}{Q}, \qquad s_{\text{lat}}^{(d)} = \frac{\bar{t}_{\text{mem}}^{(d)}}{\bar{t}_{\text{e2e}}^{(d)}}. \tag{13}$$

Failed calls count toward C but not W. A valid call returning an empty operation list is a no-op, not a failure. The latency numerator sums all MemoryAgent time within each question before averaging; mean per-call latency cannot replace it.

D When Correct Candidates Become Available

Saved-run reconstruction. We reconstruct each method's answer-state sequence from its saved execution record. Whenever the system records a concrete option for the current question, that state enters the sequence, and every method also contributes its final returned answer. Each timestamp includes all earlier model calls, so the curves show when the complete system first makes a usable answer available under its own execution protocol for every evaluated question. This preserves the native order of answer formation while keeping timing comparisons at the complete system level from the initial search to the returned prediction within one end-to-end reasoning process.

Anytime accuracy and event times. Let $\widehat{y}_i^{(m)}(t)$ be the latest candidate available for sample i under method m by elapsed time t, and set $\widehat{y}_i^{(m)}(t) = \perp$ until its first candidate appears. We evaluate the latest recorded state at time t over the complete set of N questions:

$$A_m(t) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[\widehat{y}_i^{(m)}(t) = y_i\right]. \tag{14}$$

For candidate timestamps $\mathcal{T}_i^{(m)}$, we define when a correct state first appears and when it remains correct through termination as the two answer-availability events used in the analysis below:

$$T_{i,m}^{\text{first}} = \inf\left\{t \in \mathcal{T}_i^{(m)} \mid \widehat{y}_i^{(m)}(t) = y_i\right\}, \tag{15}$$

$$T_{i,m}^{\text{stable}} = \inf\left\{t \in \mathcal{T}_i^{(m)} \mid \widehat{y}_i^{(m)}(u) = y_i,\ \forall u \in \mathcal{T}_i^{(m)},\ u \geq t\right\}. \tag{16}$$

First-correct time marks the first correct answer state. Stable-correct time marks the point after which all later recorded answers remain correct. Both values are reconstructed after execution from the predictions and timestamps stored in each frozen execution record for every compared method. The two events distinguish transient correct states from answers that survive all later updates and remain valid through the final system decision for every evaluated question and method under the same reconstruction protocol used consistently across all candidate-state timing analyses reported here.

Restricted and fixed-quality time. For event type $x \in \{\text{first}, \text{stable}\}$, we set $T^{x}_{i,m} = +\infty$ when the event is not reached. Let $T^{\text{end}}_{i,m}$ be the complete running time for question i and let $Q_{0.95}$ denote the empirical 95th percentile. The pooled completion times define the horizon for a restricted-time measurement shared by all methods and evaluated questions:

$$\tau_d = Q_{0.95}\left(\left\{T^{\text{end}}_{i,m}\right\}_{i,m}\right), \qquad R^{x}_{m,d}(\tau_d) = \frac{1}{N_d}\sum_{i=1}^{N_d} \min\left(T^{x}_{i,m}, \tau_d\right). \tag{17}$$

The shared horizon is 40.56 seconds for V* and 171.37 seconds for TreeBench. For fixed-quality latency, we use 95% of the Baseline's terminal accuracy and find the earliest time from which the anytime curve remains at or above that common quality target throughout the remaining evaluation period, which gives the following fixed-quality latency for each method:

$$q_d = 0.95\, A_{\text{base},d}(\infty), \qquad L_{m,d}(q_d) = \inf\left\{t \mid A_{m,d}(u) \geq q_d,\ \forall u \geq t\right\}. \tag{18}$$

The targets are 83.06% for V* and 48.56% for TreeBench. With terminal reasoning step $s_i^{(m)}$ and final correctness $z_{i,T}^{(m)}$, we also measure how often an incorrect answer is committed before the shared eight-step reasoning budget is exhausted for each evaluated method:

$$E_m = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\left[s_i^{(m)} < 8\right]\left(1 - z_{i,T}^{(m)}\right). \tag{19}$$

Figure 7 traces answer quality over wall-clock time. The left column reports latest-answer accuracy, the middle column the first appearance of a correct answer, and the right column the point at which it remains correct through termination. The rows correspond to V* and TreeBench. Across both benchmarks, ESTA-Mem rises earlier and reaches higher coverage, showing that compiled memory helps the multi-agent system find correct answers sooner and retain them through the final prediction.

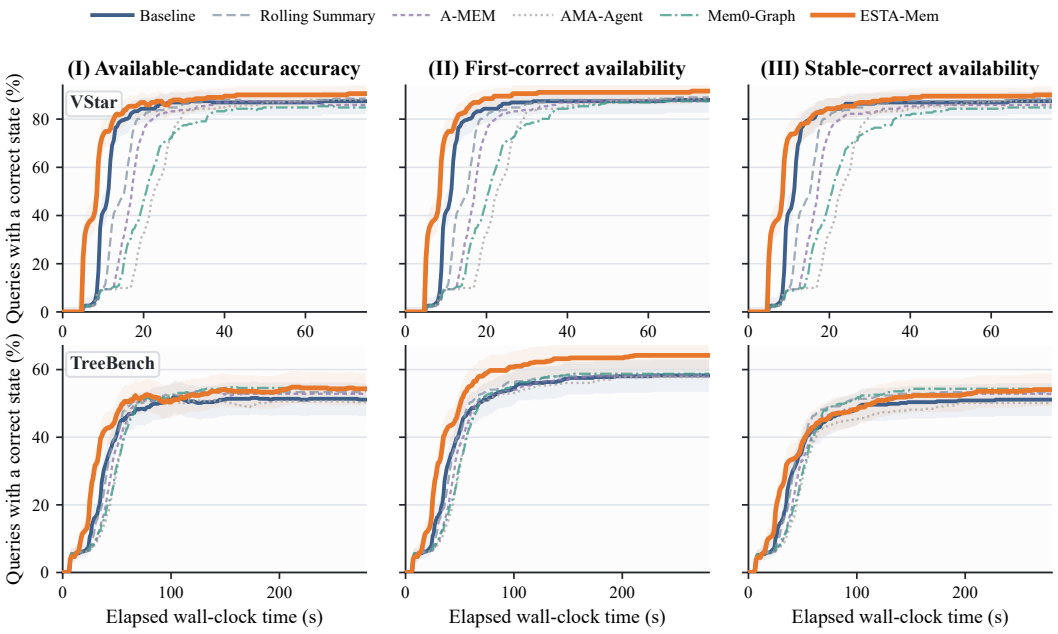

Figure 7: Anytime formation of answer-bearing states. Curves are reconstructed from timestamped states in the frozen traces. Current-candidate accuracy uses the latest answer state recorded for each question by a given time point. First-correct availability is the cumulative fraction that have produced a correct state. Stable-correct availability is the fraction that have reached a correct answer state whose later recorded answers remain correct. Shading shows pointwise 95% Wilson intervals for the Baseline and ESTA-Mem curves.

Table 13 summarizes how broadly and how quickly the two correct-answer events in Figure 7 are reached. First-correct coverage measures how many questions produce a correct answer at least once,

while stable-correct coverage measures how many retain a correct answer through termination. The median reports the typical time among reached events, and restricted time combines event coverage and latency under the shared horizon. ESTA-Mem improves every first-correct measure on both benchmarks and also raises stable-correct coverage, showing that its advantage begins with earlier discovery and continues as those answers remain stable throughout later reasoning and final answer selection across both benchmarks over the complete evaluation horizon of each recorded trajectory.

Table 13: Correct-state availability and latency.

Configuration		First-correct state			Stable-correct state	
Dataset	**Method**	**Coverage (%)**	**Median (s)**	**Restricted (s)**	**Coverage (%)**	**Restricted (s)**
V⋆	Multi-Agent Baseline	87.96	10.79	14.80	87.43	14.87
V⋆	**ESTA-Mem**	**91.62**	**8.34**	**11.36**	**90.05**	**12.42**
TreeBench	Multi-Agent Baseline	58.27	36.84	97.68	51.11	106.20
TreeBench	**ESTA-Mem**	**64.44**	**29.98**	**86.36**	**55.31**	**103.04**

Table 14 connects this earlier discovery to the answer ultimately returned by the system. Ever-correct coverage records whether a correct option appears anywhere in the trajectory, and terminal accuracy records whether the final output is correct. ESTA-Mem achieves the highest value for both measures on V⋆ and TreeBench. The joint increase shows that compiled memory creates more correct answer opportunities and helps the multi-agent system preserve more of them in its final returned predictions.

Table 14: Correct-candidate coverage and terminal accuracy on matched trajectories.

	Ever-correct coverage (%)			Terminal accuracy (%)		
Dataset	**ESTA-Mem**	**Baseline**	**Best other memory**	**ESTA-Mem**	**Baseline**	**Gain**
V⋆	**91.62**	87.96	89.01	**90.05**	87.43	**+2.62**
TreeBench	**64.44**	58.27	58.77	**55.31**	51.11	**+4.20**

Table 15 compares ESTA-Mem and the Multi-Agent Baseline on the same questions, isolating when the correct state becomes available. First-correct time falls by 23.22% on V⋆ and 11.59% on TreeBench, with both paired intervals entirely below zero. Stable-correct time also arrives earlier, including a supported 16.47% reduction on V⋆. These paired results confirm that the earlier rise in Figure 7 comes from question-level timing improvements under the same matched evaluation.

Table 15: Paired candidate-state timing on matched trajectories. Highlighted reductions mark earlier availability, with paired intervals below zero for the same questions under the shared definition.

Event cohort		Paired timing improvement		
Event	**Dataset**	**Paired change (s)**	**95% CI (s)**	**Reduction**
Restricted TTFC	V⋆	-3.44	$[-4.19, -2.74]$	**23.22%**
	TreeBench	-11.32	$[-14.99, -7.86]$	**11.59%**
Restricted stable time	V⋆	-2.45	$[-3.02, -1.87]$	**16.47%**
	TreeBench	-3.16	$[-6.20, -0.12]$	**2.97%**

Table 16 extends the stable-answer analysis from the baseline to alternative memory organizations. On V⋆, ESTA-Mem reaches a stable correct answer earlier than every reference method. On TreeBench, it also improves over A-MEM, AMA-Agent, and Mem0-Graph. The largest reductions appear against adaptive and graph-based memories, indicating that a task-aligned compiled state organizes the current trajectory into a usable answer more quickly than general-purpose memory updates operating over the same question trajectory from initial search through the final prediction under the shared evaluation protocol used for every candidate-state timing comparison reported in this section.

Table 16: Paired stable-correct timing relative to every reference method. Highlighted reductions use the same questions and have paired intervals below zero under the shared event-time definition.

Reference		Stable-time improvement	
Dataset	**Reference method**	**Reduction**	**Paired 95% interval (s)**
V*	Multi-Agent Baseline	**16.47%**	$[-3.02, -1.87]$
	Rolling Summary	**29.23%**	$[-5.74, -4.52]$
	A-MEM	**37.58%**	$[-8.22, -6.75]$
	AMA-Agent	**48.55%**	$[-12.75, -10.64]$
	Mem0-Graph	**46.49%**	$[-11.96, -9.66]$
TreeBench	Multi-Agent Baseline	**2.97%**	$[-6.20, -0.12]$
	Rolling Summary	0.15%	$[-3.61, 3.48]$
	A-MEM	**3.56%**	$[-6.91, -0.80]$
	AMA-Agent	**8.11%**	$[-12.10, -6.03]$
	Mem0-Graph	2.20%	$[-5.55, 0.85]$

Table 17 compares the time needed to reach and sustain the same candidate-accuracy target on each dataset. ESTA-Mem reaches the target before every listed reference and records the lowest incorrect early-commit rate on both benchmarks. Together with the preceding tables, this links earlier correct-state formation to sustained candidate accuracy under matched target definitions.

Table 17: Fixed-quality latency and incorrect early commits.

Configuration			Availability and commitment	
Dataset	**Target (%)**	**Method**	**Latency (s)**	**Incorrect early commit (%)**
V*	83.06	Multi-Agent Baseline	17.33	11.52
		Rolling Summary	21.41	10.99
		A-MEM	25.80	13.09
		AMA-Agent	29.88	10.99
		Mem0-Graph	37.85	13.61
		ESTA-Mem	**15.11**	**8.90**
TreeBench	48.56	Multi-Agent Baseline	76.61	38.27
		Rolling Summary	58.38	41.23
		A-MEM	63.32	40.25
		AMA-Agent	65.08	40.74
		Mem0-Graph	66.93	41.73
		ESTA-Mem	**53.14**	**37.78**

E Paired Accuracy and Component Analyses

E.1 Reasoning Budgets on TreeBench

We test reasoning budgets of 4, 8, 12, and 16 steps on the same TreeBench questions. Table 18 pairs each ESTA-Mem run with its baseline and reports accuracy, fixes and regressions, confidence intervals, and Holm-adjusted McNemar tests. At every budget, fixes outnumber regressions and the paired interval stays above zero, showing that compiled memory remains useful across both short and long reasoning trajectories under the same TreeBench evaluation protocol used throughout.

Table 18: Paired TreeBench accuracy across four reasoning budgets.

Budget	Accuracy			Paired inference			
T	Baseline	ESTA-Mem	Δ	Fix/Reg.	95% CI	Exact p	Holm p
4	51.11	**54.57**	**+3.46**	29/15	[0.25, 6.67]	.0488	.0488
8	51.11	**55.31**	**+4.20**	27/10	[1.48, 7.16]	.0076	.0305
12	51.60	**55.56**	**+3.95**	26/10	[1.23, 6.91]	.0113	.0305
16	51.36	**55.56**	**+4.20**	27/10	[1.23, 7.16]	.0076	.0305

E.2 Component Ablations

Table 19 expands the five component ablations from Table 3 into same-question pairs. Each row restores one component while the other four stay fixed, revealing how that component strengthens the complete design. We report the two benchmarks separately and use their unweighted mean as the cross-benchmark average. Confidence intervals resample paired questions within each benchmark before the two gains are averaged, and exact tests use the same benchmark-stratified comparison.

Table 19: Same-question component analysis. Average is the unweighted mean of V* and TreeBench. Intervals and tests use benchmark-stratified paired questions for every component.

Component	Performance				Stratified paired inference			
Component restored	V* gain	TreeBench gain	Avg. acc.	Avg. gain	Fix/Reg.	95% CI	Exact p	Holm p
online audit	**+0.52**	**+2.22**	71.31 → 72.68	**+1.37**	22/12	$[-0.37, 3.13]$	0.1480	0.2695
state tree	**+2.09**	0.00	71.63 → 72.68	**+1.05**	7/3	$[0.00, 2.33]$	0.0898	0.2695
provenance graph	**+2.62**	**+2.96**	69.89 → 72.68	**+2.79**	30/13	$[0.86, 4.81]$	0.0062	**0.0248**
role-conditioned views	**+2.09**	**+4.44**	69.41 → 72.68	**+3.27**	35/13	$[1.29, 5.32]$	**0.0015**	**0.0075**
candidate ledger	0.00	**+2.22**	71.57 → 72.68	**+1.11**	15/6	$[-0.15, 2.38]$	0.1218	0.2695

Table 19 shows positive average accuracy gains from all five components within the complete system. Online audit, the provenance graph, and role-conditioned views improve accuracy on both benchmarks. The state tree strengthens performance on V*, while the candidate ledger brings additional gains on TreeBench. These results highlight the value of maintaining a clear current state and preserving candidate history, helping agents use accumulated observations to make better decisions as reasoning progresses.

The two provenance analyses further connect stored relations to final task decisions. Table 30 measures how controlled changes to source and review bindings affect relation recovery, while Table 19 measures the online graph's contribution to answer accuracy. Together, these results show that preserving source-aware relations supports reliable memory readout and stronger end-to-end predictions across the complete trajectories of both benchmarks under the same task protocol.

F Structural Reliability of Compiled Memory

Compiled memory carries task, source, and review relations from the online trajectory into every later role-specific read. Table 20 replays every saved prefix and checks six properties of this process. Visual grounding verifies that each observation remains connected to its source image and region, while accepted-review lineage follows each active support relation back to the reviews that established it. Prefix compilation rebuilds the memory state after every recorded step, and typed-edge replay verifies that the same relation types remain available after reconstruction. Occurrence identity applies whitespace, capitalization, and wrapper changes to the observation text and checks that the corresponding memory objects keep the same identity. Sample and asset isolation then traces every record and generated image region to the question that created it. Together, the audit covers 610 visual observations, 5,614 trajectory prefixes, 34,945 typed-edge checks, 1,788 surface-form trace pairs, and 921 generated assets across 596 complete traces drawn from both evaluated benchmarks.

Each trace is one independent audit unit and passes when all applicable checks within that trace succeed. The scale column summarizes the internal records examined for each property. For s passing traces among n eligible traces, the final column reports the one-sided exact 95% lower bound

Table 20: Structural reliability of compiled memory. Pass rates and one-sided exact 95% lower bounds use traces as independent units; scale reports the internal records examined by the deterministic audit.

Invariant and audit unit			Validated result		
Invariant	**Independent unit**	**Scale**	**Pass**	**Rate**	**95% LB**
Visual grounding	508 traces	610 observations	**508/508**	**100%**	99.41%
Accepted-review lineage	339 traces	345 supporting observations	**339/339**	**100%**	99.12%
Prefix compilation	596 traces	5,614 prefixes	**596/596**	**100%**	99.50%
Typed-edge replay	596 traces	34,945 edge checks	**596/596**	**100%**	99.50%
Occurrence identity	596 traces	1,788 trace pairs	**596/596**	**100%**	99.50%
Sample and asset isolation	596 traces	921 generated assets	**596/596**	**100%**	99.50%

$\mathrm{Beta}^{-1}(0.05; s, n-s+1)$ on the trace-level pass rate. The observed 100% pass rates therefore retain exact lower bounds of 99.12–99.50% across all six properties and both evaluated benchmarks.

Table 21: Controlled reference checks. One relation is changed per eligible trace, and the structural checks detect changes in source identity, sample scope, visual binding, and accepted-review lineage.

Reference intervention	Identified cases			Validated capability
Change applied	**V⋆**	**TreeBench**	**Pooled**	**Relation checked**
Remove the referenced source event	159/159	349/349	**508/508**	Source-event integrity
Use a source ID from another sample	159/159	349/349	**508/508**	Within-sample source boundary
Remove the *GroundedIn* relation	159/159	349/349	**508/508**	Visual-binding integrity
Remove the accepted Review lineage	127/127	212/212	**339/339**	Accepted-review lineage

The two tables evaluate complementary sides of structural reliability. Table 20 shows that all six invariants remain intact through prefix replay across grounding, review lineage, typed relations, occurrence identity, and sample scope. Table 21 then changes one recorded reference at a time, and the structural checks detect every applicable source, scope, grounding, and lineage change. Together, these results show that compiled memory keeps its relationships valid and its source and review links intact throughout online updates and all later role-specific reads across both evaluated benchmarks.

G Model Allocation and Computational Cost

G.1 Memory-Call-Controlled Comparison

We compare memory organizations under a shared online-update budget on the full V⋆ evaluation set. Generic Reflection, Delta Summary, Rolling Summary, A-MEM, AMA-Agent, and Mem0-Graph each make exactly two memory-model calls before the final decision. ESTA-Mem keeps its native event-driven schedule and averages 2.01 calls per question. All runs use the same questions, backbone, task agents, step budget, and decoding settings. The V⋆ runtimes in Table 22 and Tables 26–27 come from separate runs of the same frozen configuration.

Table 22: Memory-call-controlled comparison on V⋆. Memory-module and end-to-end latency are reported as median/P90 seconds per question, and memory output tokens count only responses produced by the online memory module during online memory updates.

Comparison		Quality	Memory-controlled profile			
Group	**Method**	**Acc.**	**Memory calls/q**	**Memory output tok/q**	**Memory s/q**	**E2E s/q**
Generic	Generic Reflection	86.91	2.00	119	5.63 / 9.92	29.47 / 44.16
Generic	Delta Summary	86.39	2.00	119	5.80 / 10.40	28.71 / 42.31
Online	Rolling Summary	86.39	2.00	169	8.17 / 15.21	28.86 / 49.39
Modern	A-MEM	86.91	2.00	686	25.44 / 46.43	41.69 / 78.62
Modern	AMA-Agent	86.91	2.00	564	19.90 / 37.01	37.35 / 62.37
Modern	Mem0-Graph	87.43	2.00	885	32.86 / 66.51	49.26 / 97.39
Ours	**ESTA-Mem**	**90.05**	2.01	**49**	**5.79 / 7.39**	**22.60 / 35.31**

Table 22 shows that ESTA-Mem achieves 90.05% accuracy under the approximately matched memory-call budget, exceeding the next-best method, Mem0-Graph, by 2.62 percentage points. Its memory module produces 49 output tokens per question and has a P90 latency of 7.39 seconds. End-to-end median and P90 runtimes are 22.60 and 35.31 seconds, respectively, representing reductions of 21.28% and 16.54% over Delta Summary, the fastest competing method on both measures. ESTA-Mem thus combines the highest accuracy with the lowest end-to-end latency in this comparison.

G.2 Using a Smaller MemoryAgent

We test whether online memory maintenance needs the same capacity as the task-facing roles. On TreeBench, we change only the MemoryAgent from Qwen3-VL-8B to Qwen3-VL-4B. The task agents, prompts, image settings, decoding, final-answer pipeline, and $T = 8$ budget remain fixed. Table 23 compares the two settings, with gains measured against the all-8B Multi-Agent Baseline.

Table 23: TreeBench accuracy and runtime with an 8B or 4B MemoryAgent. The lightweight auditor preserves task accuracy while reducing median and tail latency and capacity-weighted model calls.

Configuration	**Quality**		**Runtime profile**			
Method	**Acc.**	**Gain vs. baseline**	**Median/P90 s/q**	**VLM calls/q**	**Input tok/q**	**B-param calls/q**
ESTA-Mem, MemoryAgent-8B	**55.31**	**+4.20**	82.71 / 186.98	16.61	146,201	132.88
ESTA-Mem, MemoryAgent-4B	**55.31**	**+4.20**	**73.30 / 149.17**	16.40	142,930	**120.17**

Both MemoryAgent sizes reach 55.31% accuracy and improve the matched baseline by 4.20 points. With the 4B auditor, median latency falls from 82.71 to 73.30 seconds, P90 latency falls from 186.98 to 149.17 seconds, and parameter-weighted calls fall from 132.88B to 120.17B per question. B-param calls are obtained by summing model sizes across all invocations. The maintained accuracy and lower runtime show that local memory auditing can use a compact backbone, leaving task-facing capacity for visual reasoning and final decisions throughout the complete task trajectory.

G.3 Role-Wise Model Allocation

Table 24 studies how model capacity can be distributed across the responsibilities of the multi-agent system. We evaluate 51 complete configurations on the full V* test set using the same A6000 setup and frozen control protocol. Four uniform allocations establish the global scaling trend, 27 single-role substitutions isolate each responsibility, and 20 heterogeneous allocations test coordinated choices across several roles while changing only the backbone assigned to each role.

Table 24: Role-wise backbone allocation on V* under one frozen control protocol, covering uniform, single-role, and heterogeneous configurations across all nine agent responsibilities.

Configuration	**Role-wise backbone allocation**									**Outcome**		
Configuration	Mgr.	Plan.	Exec.	Tools	Reas.	Refl.	Memory Agent	Verifier	Answerer	Correct count	Accuracy (%)	Change vs. uniform 8B
Uniform model scaling (4 configurations)												
Uniform 2B	2B	2B	2B	2B	2B	2B	2B	2B	2B	134/191	70.16	−19.90
Uniform 4B	4B	4B	4B	4B	4B	4B	4B	4B	4B	158/191	82.72	−7.33
Uniform 8B (reference)	8B	8B	8B	8B	8B	8B	8B	8B	8B	172/191	90.05	0.00
Uniform 32B	32B	32B	32B	32B	32B	32B	32B	32B	32B	164/191	85.86	−4.19
Single-role substitutions, with all unlisted roles kept at 8B (27 configurations)												
Manager 2B	2B	8B	8B	8B	8B	8B	8B	8B	8B	169/191	88.48	−1.57
Manager 4B	4B	8B	8B	8B	8B	8B	8B	8B	8B	173/191	**90.58**	+0.52
Manager 32B	32B	8B	8B	8B	8B	8B	8B	8B	8B	172/191	90.05	0.00
Planner 2B	8B	2B	8B	8B	8B	8B	8B	8B	8B	158/191	82.72	−7.33
Planner 4B	8B	4B	8B	8B	8B	8B	8B	8B	8B	169/191	88.48	−1.57
Planner 32B	8B	32B	8B	8B	8B	8B	8B	8B	8B	167/191	87.43	−2.62
Executor 2B	8B	8B	2B	8B	8B	8B	8B	8B	8B	148/191	77.49	−12.57
Executor 4B	8B	8B	4B	8B	8B	8B	8B	8B	8B	169/191	88.48	−1.57
Executor 32B	8B	8B	32B	8B	8B	8B	8B	8B	8B	168/191	87.96	−2.09
Tools 2B	8B	8B	8B	2B	8B	8B	8B	8B	8B	159/191	83.25	−6.81

Continued on next page

Table 24 (continued)

Configuration	Role-wise backbone allocation									Outcome		
Configuration	Mgr.	Plan.	Exec.	Tools	Reas.	Refl.	Memory Agent	Verifier	Answerer	Correct count	Accuracy (%)	Change vs. uniform 8B
Tools 4B	8B	8B	8B	4B	8B	8B	8B	8B	8B	165/191	86.39	−3.66
Tools 32B	8B	8B	8B	32B	8B	8B	8B	8B	8B	163/191	85.34	−4.71
Reasoner 2B	8B	8B	8B	8B	2B	8B	8B	8B	8B	168/191	87.96	−2.09
Reasoner 4B	8B	8B	8B	8B	4B	8B	8B	8B	8B	168/191	87.96	−2.09
Reasoner 32B	8B	8B	8B	8B	32B	8B	8B	8B	8B	170/191	89.01	−1.05
Reflection 2B	8B	8B	8B	8B	8B	2B	8B	8B	8B	171/191	89.53	−0.52
Reflection 4B	8B	8B	8B	8B	8B	4B	8B	8B	8B	169/191	88.48	−1.57
Reflection 32B	8B	8B	8B	8B	8B	32B	8B	8B	8B	170/191	89.01	−1.05
MemoryAgent 2B	8B	8B	8B	8B	8B	8B	2B	8B	8B	171/191	89.53	−0.52
MemoryAgent 4B	8B	8B	8B	8B	8B	8B	4B	8B	8B	172/191	90.05	0.00
MemoryAgent 32B	8B	8B	8B	8B	8B	8B	32B	8B	8B	170/191	89.01	−1.05
Verifier 2B	8B	8B	8B	8B	8B	8B	8B	2B	8B	170/191	89.01	−1.05
Verifier 4B	8B	8B	8B	8B	8B	8B	8B	4B	8B	169/191	88.48	−1.57
Verifier 32B	8B	8B	8B	8B	8B	8B	8B	32B	8B	174/191	**91.10**	+1.05
Answerer 2B	8B	8B	8B	8B	8B	8B	8B	8B	2B	167/191	87.43	−2.62
Answerer 4B	8B	8B	8B	8B	8B	8B	8B	8B	4B	170/191	89.01	−1.05
Answerer 32B	8B	8B	8B	8B	8B	8B	8B	8B	32B	173/191	**90.58**	+0.52
Multi-role heterogeneous allocations (20 configurations)												
Heterogeneous 1	4B	8B	8B	8B	8B	8B	4B	8B	32B	174/191	**91.10**	+1.05
Heterogeneous 2	4B	8B	8B	8B	8B	8B	8B	8B	32B	174/191	**91.10**	+1.05
Heterogeneous 3	4B	8B	8B	8B	8B	8B	4B	32B	32B	174/191	**91.10**	+1.05
Heterogeneous 4	8B	8B	8B	8B	8B	8B	8B	32B	32B	174/191	**91.10**	+1.05
Heterogeneous 5	4B	8B	8B	8B	8B	8B	4B	32B	8B	174/191	**91.10**	+1.05
Heterogeneous 6	4B	8B	8B	8B	8B	8B	8B	32B	8B	174/191	**91.10**	+1.05
Heterogeneous 7	8B	8B	8B	8B	8B	8B	4B	32B	8B	174/191	**91.10**	+1.05
Heterogeneous 8	8B	8B	8B	8B	8B	8B	4B	8B	32B	173/191	**90.58**	+0.52
Heterogeneous 9	4B	8B	8B	8B	8B	8B	4B	8B	8B	173/191	**90.58**	+0.52
Heterogeneous 10	4B	8B	8B	8B	8B	8B	2B	8B	8B	172/191	90.05	0.00
Heterogeneous 11	8B	8B	8B	8B	8B	4B	4B	4B	8B	171/191	89.53	−0.52
Heterogeneous 12	8B	8B	8B	8B	8B	8B	4B	4B	8B	169/191	88.48	−1.57
Heterogeneous 13	2B	8B	8B	8B	8B	8B	2B	8B	8B	169/191	88.48	−1.57
Heterogeneous 14	2B	8B	8B	8B	8B	8B	4B	8B	8B	169/191	88.48	−1.57
Heterogeneous 15	8B	8B	8B	8B	4B	8B	8B	4B	8B	167/191	87.43	−2.62
Heterogeneous 16	4B	8B	4B	4B	8B	4B	4B	8B	8B	161/191	84.29	−5.76
Heterogeneous 17	4B	4B	4B	4B	4B	8B	8B	8B	4B	158/191	82.72	−7.33
Heterogeneous 18	4B	4B	4B	4B	4B	4B	2B	4B	4B	158/191	82.72	−7.33
Heterogeneous 19	4B	4B	4B	4B	4B	4B	8B	4B	4B	157/191	82.20	−7.85
Heterogeneous 20	2B	8B	2B	2B	8B	2B	2B	8B	8B	148/191	77.49	−12.57

The allocation results reveal a clear role-specific capacity pattern. The Planner, Executor, and tool agents benefit from retaining the visual and action capacity of the 8B backbone. A 4B MemoryAgent preserves the 90.05% accuracy of the uniform 8B system, while a 32B Verifier reaches 91.10%. Several mixed allocations also reach 91.10% by combining compact Manager or MemoryAgent roles with added capacity in the Verifier or Answerer. Model capacity is therefore most effective when assigned according to role responsibility. In the table, Mgr., Plan., Exec., Reas., and Refl. abbreviate Manager, Planner, Executor, Reasoner, and the reflection roles used throughout the study.

G.4 MemoryAgent Activity and Role-Level Efficiency

We characterize the MemoryAgent through three complementary measurements. Table 25 measures invocation scale and transaction validity, Table 26 locates memory maintenance within the complete episode, and Table 27 attributes runtime across agent roles. Together, they connect reliable execution with an actionable capacity-allocation strategy for the full multi-agent system at evaluation scale.

Table 25: MemoryAgent invocation and transaction validation.

Dataset	Invocation and validation		
Dataset	**Calls**	**Calls/q**	**Valid/failed**
V*	384	2.01	382/2
TreeBench	1123	2.77	1049/74
HRBench-4K	1720	2.15	1700/20
HRBench-8K	1763	2.20	1718/45
XLRS-Bench	7496	2.43	7440/56
All	12486	2.37	12289/197

Across 12,486 invocations, 12,289 transactions pass validation, yielding a 98.42% rate. Calls remain within 2.01–2.77 per question across all five benchmarks. This consistency supplies deterministic replay with reliable typed updates throughout complete multi-agent reasoning trajectories.

Table 26 measures MemoryAgent latency at both the call and question levels. Memory maintenance occupies a consistent 18.43–23.19% of episode time across the five benchmarks. Its bounded and measurable share makes MemoryAgent capacity a direct lever for improving system-level latency.

Table 26: Episode-level and MemoryAgent runtime across five benchmarks.

Metric	V*	TreeBench	HRBench-4K	HRBench-8K	XLRS-Bench
Per question					
End-to-end mean	24.77	106.04	104.80	73.28	158.62
End-to-end P50	22.10	82.71	102.35	60.34	125.22
End-to-end P95	43.30	265.40	192.67	149.81	403.36
Mean model time	24.20	105.35	102.34	66.35	146.49
Mean MemoryAgent calls	2.01	2.77	2.15	2.20	2.43
Mean MemoryAgent time	5.43	21.67	24.30	14.51	29.24
MemoryAgent time share	21.93%	20.43%	23.19%	19.81%	18.43%
Per MemoryAgent call					
Mean	2.70	7.81	11.30	6.59	12.01
P50	2.07	7.50	12.40	5.54	12.92
P95	4.97	14.65	17.78	12.74	18.85
Cumulative time (hours)					
End-to-end	1.31	11.93	23.29	16.29	135.71
MemoryAgent	0.29	2.44	5.40	3.23	25.02

Table 27 attributes runtime to each agent role. MemoryAgent has the largest accumulated role time on V* and the two HRBench settings, while Planner time is largest on TreeBench and XLRS-Bench. This profile directly guides model allocation. Table 23 shows that replacing only the MemoryAgent with a 4B model preserves the 55.31% TreeBench accuracy of the all-8B system and lowers both median and tail latency, while the Planner, Executor, Verifier, and Answerer retain their full task-facing capacity.

Table 27: Mean per-question latency by agent role across five benchmarks (seconds).

Role	V*	TreeBench	HRBench-4K	HRBench-8K	XLRS-Bench
Manager	0.27	0.61	0.31	0.25	0.55
Planner	3.88	23.63	21.70	12.16	33.21
Executor	2.50	5.26	2.43	2.07	3.98
Tool	4.23	14.11	14.98	12.48	19.31
Answerer	2.47	13.94	15.92	9.57	19.14
Verifier	4.28	18.81	17.30	10.82	27.14
MemoryAgent	**5.43**	**21.67**	**24.30**	**14.51**	**29.24**
Reflector	0.90	2.40	1.97	2.38	4.74
Reasoner	0.24	4.92	3.43	2.11	9.19

H COMPACT DELIVERY AND RECOVERABLE MEMORY RELATIONS

Compiled memory should remain concise for each receiving role while preserving the relations used by later decisions. We evaluate these properties in sequence. Table 28 measures compact delivery and state folding, while Table 29 connects the resulting terminal signals to answer quality.

Table 28 shows how compilation concentrates the event log before delivery. Token statistics use 486 trajectories with saved memory views for answer generation, comprising 160 V* and 326 TreeBench examples. Across both benchmarks, role-conditioned views retain 7.12% of the full compiled text, while deterministic folding consolidates 43.71% of repeated state occurrences. Projection selects the information needed by the receiving role, and state folding merges recurring configurations inside the compiled memory. Together, they keep role inputs compact while retaining event addresses and typed relations for deterministic replay.

Table 28: Compact role delivery and deterministic state folding.

Dataset	Delivered text			Folded state		
Dataset	**Full tokens**	**View tokens**	**Reduction**	**States before**	**States after**	**Reduction**
V*	6,226.0	372.6	**94.02%**	772	485	**37.18%**
TreeBench	8,237.0	621.0	**92.46%**	1,914	1,027	**46.34%**
Overall	**7,575.0**	**539.2**	**92.88%**	**2,686**	**1,512**	**43.71%**

Table 29 connects the compact terminal state to answer quality. An explicit support link, an accepted observation, and a verified ledger candidate each mark higher conditional accuracy on both benchmarks. Their pooled gaps reach 14.88, 11.00, and 15.75 points, respectively, with significant Fisher tests. The same direction within each dataset shows that the compiled signals remain informative across distinct visual reasoning settings with different search, comparison, and information-integration demands.

Table 29: Terminal memory signals and conditional accuracy.

Terminal signal			Conditional accuracy			Test
Signal	**Dataset**	n_+/n_-	**Present**	**Absent**	**Gap**	**Fisher** p
Explicit support link	V*	127/64	98.43	73.44	**+24.99**	–
	TreeBench	212/193	57.55	52.85	**+4.70**	–
	Pooled	339/257	72.86	57.98	**+14.88**	1.63×10^{-4}
Accepted observation	V*	130/61	98.46	72.13	**+26.33**	–
	TreeBench	238/167	55.46	55.09	**+0.37**	–
	Pooled	368/228	70.65	59.65	**+11.00**	7.33×10^{-3}
Verified ledger candidate	V*	154/37	92.86	78.38	**+14.48**	–
	TreeBench	265/140	58.49	49.29	**+9.20**	–
	Pooled	419/177	71.12	55.37	**+15.75**	2.94×10^{-4}

H.1 SOURCE-STATE BINDING RECOVERY

We select 162 completed trajectories whose final Planner memory view contains an observation. A fixed Qwen3-VL-8B reader identifies the observation's visual source and current review state. Joint recovery requires both to match the saved record. The intact typed condition recovers 161 of the 162 pairs, yielding 99.38% joint recovery.

We examine three perturbations of the saved memory: permuting visual sources, permuting review states, and removing the added memory packet. The permutations preserve the marginal source and state distributions, while packet removal retains the original history. Input token counts are matched across conditions. Table 30 compares each perturbation with intact memory to assess how source and review bindings support accurate readout.

Table 30: Joint recovery under controlled source-state perturbations.

Intervention	Paired binding response		
Intervention	**Joint change**	**95% CI**	**McNemar p**
Source permuted	$\downarrow$ 62.35 pp	$[-69.75, -54.94]$	7.89×10^{-31}
State permuted	$\downarrow$ 53.09 pp	$[-60.49, -45.68]$	2.58×10^{-26}
Audit packet discarded	$\downarrow$ 64.81 pp	$[-72.22, -57.41]$	4.93×10^{-32}

Recovery decreases when source or review-state bindings are permuted, and when the added memory packet is removed (Table 30). Preserving these bindings helps the reader identify an observation's visual source and current review state.

I Memory Growth and Observation Identity

This section examines how the compiled representation responds to task demand and how event addresses preserve observation identity when different records share the same text. Together, the two analyses connect memory growth with the work performed during reasoning and show how repeated content remains traceable throughout the multi-agent reasoning process and downstream role use.

Table 31: Structure growth across benchmark-native task families. Changes compare relation with attribute questions on V* and reasoning with perception questions on TreeBench.

Quantity	V*		TreeBench		Interpretation
Quantity	**Change**	p	**Change**	p	**Interpretation**
Immutable events	**+30.54%**	$< 10^{-4}$	**+45.56%**	$< 10^{-4}$	More execution and review events
Graph nodes	**+28.57%**	$< 10^{-4}$	**+34.58%**	$< 10^{-4}$	More assets, observations, reviews, and candidates
Typed edges	**+46.56%**	$< 10^{-4}$	**+48.39%**	$< 10^{-4}$	Greater dependency and provenance demand
Visual assets	**+53.30%**	$< 10^{-4}$	**+63.28%**	$< 10^{-4}$	More visual acquisition on complex tasks
Observation nodes	**+32.18%**	$< 10^{-4}$	**+53.32%**	$< 10^{-4}$	Observation volume follows task demand
Folded states	**+11.72%**	.0113	**+22.38%**	$< 10^{-4}$	States summarize repeated progress across steps

Table 31 compares benchmark-native task families using the saved trajectories: relation versus attribute questions on V*, and reasoning versus perception questions on TreeBench. Relative changes are computed from the group means, with p values from 10,000 two-sided permutation tests. Both comparisons produce more events, graph nodes, typed relations, visual assets, observations, and folded states for the task family requiring richer relational processing. Visual assets and typed relations grow most, while folded states grow more gradually as replay consolidates repeated progress into a compact sequence of distinct reasoning configurations for later role use.

Table 32: Observation identity under repeated content. Event addresses keep observation records distinct, including extra occurrences beyond the first record in each shared-content group.

Dataset	Occurrence audit				Identity result	
Dataset	**Traces**	**Observation occurrences**	**Ambiguous content groups**	**Extra shared occurrences**	**Identity unique**	**Extra shared rate**
V*	191	163	1	1	**163/163**	0.61%
TreeBench	405	447	6	7	**447/447**	1.57%
Pooled	596	610	7	8	**610/610**	1.31%

We next identify observations whose normalized text is shared by more than one occurrence. Table 32 identifies seven shared-content groups containing 15 observations, including eight occurrences beyond the first in each group. Event addresses keep all 610 observations distinct. The repeated strings refer to different targets, argument subjects, or terminal states, so their surrounding task and source relations remain distinct even when the surface text matches. Occurrence-level addressing therefore lets the compiler reuse observation content while preserving where each record came from, which task referent it describes, and where it belongs in the reasoning process. These identities remain available for later role-specific retrieval and review.

J END-TO-END CASES

We present ten same-ID repairs as the original task input, full-history replay, and source-resolved ESTA-Mem replay.

CASE 1: REVIEWING THE VISUAL SOURCE BEFORE ACCEPTING AN ATTRIBUTE

Example



Question What is the color of the straw?

Options A. black B. red C. pink D. white

Reference answer A. black

Full-history replay

Evidence acquisition 1

- **Source** A crop localizes the storefront decoration and the thin straw.
- **Observation** Green leaves dominate the selected region.
- **Candidate** C. pink.
- **Verification** DISPUTE. The observation mainly describes nearby decoration and does not establish the straw color.

Candidate review

- **Source** The same crop and inspection record remain in the running history.
- **Observation** The Verifier reports that the straw itself is obscured, so the green region cannot support Candidate C.
- **Candidate** C is resubmitted several times and the raw run eventually returns D. white.

Terminal context The raw chronology retains the green-leaf description, disputed candidate, repeated submissions, and answer mentions in one sequence without an explicit target–source relation.

Model output D. white ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the straw and T_2 binds its requested color.
- Every observation retains its source crop, task target, occurrence ID, and audit state.

Audited update

- E_1 retains the green-leaf description together with the crop that produced it. Candidate C is preserved with state *Disputed*.
- The next candidate is written as a separate answer state. The Verifier entails A from the available visual context and accepts Candidate A.

Compiled terminal view

```
State S1 [crop inspected]
Source P1 -> E1 -> T1,T2
Ledger C [disputed] -> A [verified]
```

Model output A. black ✓

CASE 2: BINDING BOTH REFERENTS IN A LEFT–RIGHT COMPARISON

Example



Question Is the red chair on the left or right side of the road?

Options A. left B. right

Reference answer B. right

Full-history replay

Evidence acquisition

- **Source** Separate crops localize the red chair and the road.
- **Observation** Both visual targets are available, while their ordered left–right relation is not recorded explicitly.
- **Candidate** A. left.
- **Verification** The full-history run accepts Candidate A.

Relation readout

- **Source** The chair and road crops remain two separate entries in the chronological context.
- **Observation** No source-bound relation is added before the baseline returns its answer.
- **Candidate** A. left remains the terminal candidate.

Terminal context The two visual regions and their left–right statements appear chronologically, but the target chair and reference road are not retained as one ordered relation.

Model output **A. left** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the red chair and T_2 binds the road used as the reference axis.
- The requested relation is stored as chair-relative-to-road, preserving the order of the two referents.

Audited update

- A reasoning observation states that the red chair is on the right sidewalk relative to the road.
- E_1 is linked to the question, T_1, and T_2, then admitted. Candidate B is recorded under this resolved relation.

Compiled terminal view

```
State S1 -> S2
Source Pchair + Proad -> E1 -> T1,T2
Ledger A [unresolved] -> B [supported by E1]
```

Model output **B. right** ✓

CASE 3: RESOLVING A VIEWPOINT-RELATIVE DIRECTION

Example



Question From the red car's perspective, where is the man walking a dog?

Options A. front left B. front right C. left rear D. right rear

Reference answer B. front right

Full-history replay

Evidence acquisition 1

- **Source** Separate crops localize the red car and the man walking the dog.
- **Observation** The perspective tool places the man at the car's front right.
- **Candidate** B. front right.
- **Verification** DISPUTE. The review instead interprets the man's image-side location as left rear.

Repeated search

- **Source** The same two object crops are reused by later perspective calls.
- **Observation** Repeated calls again return the front-right relation, while the earlier review remains in the history.
- **Candidate** The baseline ultimately follows C. left rear.

Terminal context The raw history contains both directional statements and the later review, while the identities of viewpoint anchor and target must be reconstructed from nearby text.

Model output C. left rear ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the red car as viewpoint anchor and T_2 binds the man as the queried target.
- The two occurrence IDs remain fixed across localization, perspective calculation, and review.

Audited update

- The front-right tool result is kept with the two source crops and the ordered anchor–target binding.
- Candidate B is evaluated against that same relation and is accepted by the Verifier without another conflicting search branch.

Compiled terminal view

```
State S1 [car + man localized]
Source Pcar,Pman -> E1 -> T1,T2
Ledger B [front-right relation] -> B [verified]
```

Model output B. front right ✓

CASE 4: PRESERVING SUBJECT IDENTITY ACROSS A VIEWPOINT TRANSFORM

Example



Question From the child in blue's viewpoint, where is the person in gray beneath the blue shelter?

Options A. front left B. front right C. left rear D. right rear

Reference answer D. right rear

Full-history replay

Evidence acquisition 1

- **Source** One crop localizes the child in blue and another localizes the gray-clothed person beneath the shelter.
- **Observation** The perspective tool places the gray-clothed person behind and to the right of the child.
- **Candidate** D. right rear.
- **Verification** The first review is unavailable, so the baseline continues searching instead of returning the candidate.

Repeated search

- **Source** Later perspective and geometry calls reuse the same two subject crops.
- **Observation** The accumulated sequence mixes the original right-rear result with image-plane left/right descriptions.
- **Candidate** The baseline eventually returns C. left rear.

Terminal context Clothing descriptions, perspective output, and geometric feedback remain in one sequence, leaving the final call to recover which person defines the viewpoint.

Model output **C. left rear** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the child in blue as viewpoint anchor and T_2 binds the person in gray as the target.
- Their occurrence identities remain separate and connected to the original image regions.

Audited update

- The right-rear relation remains attached to the child-as-anchor and person-as-target ordering.
- Candidate D is evaluated under that same task-bound state and accepted by the Verifier before unrelated geometry descriptions can replace it.

Compiled terminal view

```
State S1 [two subjects localized]
Source Pchild,Pperson -> E1 -> T1,T2
Ledger D [right-rear relation] -> D [verified]
```

Model output **D. right rear** ✓

CASE 5: REPLACING AN UNSUPPORTED ORDERING JUDGMENT

Example



Question Among the trees behind the pink sign, which is the shortest from left to right?

Options A. first tree B. second tree C. third tree D. fourth tree

Reference answer D. fourth tree

Full-history replay

Evidence acquisition 1

- **Source** A crop localizes the pink sign and the tree group.
- **Observation** The trees are positioned above the sign.
- **Candidate** C. third tree.
- **Verification** DISPUTE. The observation identifies the group but does not establish relative height.

Evidence acquisition 2

- **Source** A later whole-image reasoning call revisits the tree group.
- **Observation** It identifies the second tree from the left as shortest.
- **Candidate** B. second tree.

Terminal context The raw chronology presents observations, candidate mentions, and audit feedback in one undifferentiated sequence.

Model output **B. second tree** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the pink sign and T_2 binds the trees behind it.
- Every observation retains its source crop, task target, occurrence ID, and audit state.

Audited update

- E_1 is linked to T_2. Candidate C is preserved with state *Disputed*.
- A later reasoning observation E_2 identifies the rightmost visible tree as shortest. It is linked to the question and T_2, then admitted. Candidate D is evaluated under this updated state.

Compiled terminal view

```
State S1 -> S2
Source Ptree -> E2 -> T2
Ledger C [disputed] -> D [verified after E2]
```

Model output **D. fourth tree** ✓

CASE 6: COMBINING ORDINAL REFERENCE WITH ATTRIBUTE READING

Example



Question What color is the basket of the second bicycle from right to left?

Options A. white B. black C. blue D. pink

Reference answer B. black

Full-history replay

Evidence acquisition

- **Source** A crop localizes the basket of the second bicycle under the requested right-to-left ordering.
- **Observation** The reasoner describes the localized basket as white.
- **Candidate** B. black is submitted from the running context.
- **Verification** DISPUTE. The baseline review entails A. white and returns that option.

Terminal context

- **Source** The target crop, ordinal phrase, reasoner text, and answer proposal are presented in one chronological sequence.
- **Observation** The selected bicycle occurrence is not carried as a separate task-bound record at the final read.
- **Candidate** A. white becomes the terminal answer.

Model output **A. white** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the right-to-left bicycle ordering and T_2 binds the basket of the selected second bicycle.
- Each bicycle and basket retains a distinct occurrence ID and source region.

Audited update

- E_1 is linked to the question and the selected-bicycle task target, so the crop and ordinal referent remain one record.
- The Verifier independently checks Candidate B against that localized target and accepts B. black as the task answer.

Compiled terminal view

```
State S1 -> S2
Source Pbicycle -> E1 -> T1,T2
Ledger B [task-bound candidate] -> B [verified]
```

Model output **B. black** ✓

CASE 7: KEEPING A COUNT ATTACHED TO ITS ORDERING SCOPE

Example



Question From right to left, which is the first flag on the left side of the streetlight?

Options A. first B. second C. third D. fourth

Reference answer C. third

Full-history replay

Evidence acquisition

- **Source** A crop localizes the streetlight and nearby flags.
- **Observation** Four flags are visible in the selected area.
- **Candidate** C. third.
- **Verification** The observation is accepted, while the baseline Verifier entails D. fourth.

Later reasoning

- **Source** The same image is reread after the initial candidate is rejected.
- **Observation** A natural-language description identifies the flag closest to the streetlight on its left.
- **Candidate** D. fourth is retained as the final answer.

Terminal context Count, spatial scope, and traversal order appear as separate records, so the count is mapped directly to an answer option.

Model output **D. fourth** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the streetlight, T_2 binds the flags on its left, and T_3 binds the right-to-left ordering rule.
- The count remains attached to the region and ordering scope that produced it.

Audited update

- The four localized flags, the streetlight crop, and the count remain attached to the same scoped state.
- Candidate C is read under the right-to-left task binding and is accepted by the Verifier in that state.

Compiled terminal view

```
State S1 [four scoped flags]
Source Pflags -> E1 -> T1,T2,T3
Ledger C [ordered count] -> C [verified]
```

Model output **C. third** ✓

CASE 8: REVISING A LOCATION AFTER REVIEW

Example



Question Where are the four potted plants in the foreground?

Options A. patch by the wall B. foreground grass C. garden bed behind the grass D. patch right of the stairs

Reference answer C. garden bed behind the grass

Full-history replay

Evidence acquisition

- **Source** A crop localizes the four potted plants in the foreground.
- **Observation** The crop is available to the final decision, but the named location alternatives remain to be resolved.
- **Candidate** A. patch by the wall.
- **Verification** The full-history run accepts Candidate A.

Terminal context

- **Source** The target crop and four textual location options remain in the raw chronology.
- **Observation** No explicit record connects the plant group to the grass boundary and garden-bed alternative.
- **Candidate** A. patch by the wall remains terminal.

Model output **A. patch by the wall** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the four potted plants and T_2 binds their requested location among the named regions.
- The wall, grass boundary, and garden bed retain separate source-linked occurrences.

Audited update

- Candidate A is stored with the localized plant state and then marked rejected when the Verifier entails C.
- Candidate C is written as a new answer state and accepted on the next review, preserving the revision instead of returning to A.

Compiled terminal view

```
State S1 [plant crop localized]
Source Pplant -> T1,T2
Ledger A [rejected] -> C [verified]
```

Model output **C. garden bed behind the grass** ✓

CASE 9: DELIVERING AN ACCEPTED ATTRIBUTE TO THE ANSWERING ROLE

Example



Question What color are the clothes worn by the man sitting on the bricks at the left?

Options A. black/white B. dark blue/white C. light/dark blue D. light blue/white stripes

Reference answer B. dark blue/white

Full-history replay

Evidence acquisition 1

- **Source** One crop localizes the seated man and another focuses on his clothing.
- **Observation** The clothes appear dark blue and white.
- **Candidate** B. dark blue/white.
- **Verification** The observation is visually consistent with the requested person.

Candidate review

- **Source** The same clothing observation is reviewed against the submitted option.
- **Observation** The baseline review reinterprets the stripes as black and white.
- **Candidate** A. black/white becomes terminal.

Terminal context Both attribute statements remain adjacent in the chronological record without one stable subject–source–review chain.

Model output **A. black/white** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the seated man and T_2 binds the clothing color requested for that person.
- The person crop and clothing crop remain distinct but linked occurrences.

Audited update

- E_1 links the dark-blue-and-white attribute to T_1 and T_2 and is retained as an accepted observation.
- The observation is linked to the question, T_1, and T_2, then admitted. Candidate B is accepted under the same subject-specific record.

Compiled terminal view

```
State S1 -> S2
Source Pman -> E1 -> T1,T2
Ledger B [supported by E1] -> B [verified]
```

Model output **B. dark blue/white** ✓

CASE 10: COMPARING DISTANCES ACROSS THREE LINKED REGIONS

Example



Question Is the green tree nearer to the leafless tree or to the T-junction road?

Options A. nearer the leafless tree B. equidistant C. farther from the leafless tree than from the road D. the road is nearer to the leafless tree

Reference answer C. farther from the leafless tree than from the road

Full-history replay

Evidence acquisition 1

- **Source** Separate crops localize the green tree, leafless tree, and T-junction road.
- **Observation** The green tree initially appears nearer to the leafless tree.
- **Candidate** A. nearer the leafless tree.
- **Verification** DISPUTE. The comparison does not preserve all three reference regions in one geometric state.

Candidate review

- **Source** The same three localized regions are presented to the Verifier.
- **Observation** The full-history run accepts the first pairwise interpretation.
- **Candidate** A. nearer the leafless tree remains terminal.

Terminal context The initial and revised pairwise comparisons remain distributed across the raw history, and the final call returns to the first relation.

Model output **A. nearer the leafless tree** ×

ESTA-Mem source-resolved replay

Task binding

- T_1 binds the green tree, T_2 the leafless tree, and T_3 the T-junction road.
- The requested comparison keeps T_1 as subject and T_2, T_3 as the two reference alternatives.

Audited update

- Candidate A is kept with the three-region comparison and marked rejected when the Verifier entails C.
- Candidate C is then recorded as a separate answer state and accepted under the same three task referents.

Compiled terminal view

```
State S1 [three regions localized]
Source Ptree,Proad -> E1 -> T1,T2,T3
Ledger A [rejected] -> C [verified]
```

Model output **C. farther from the leafless tree** ✓