WHAT SHOULD EMBODIED AGENTS REMEMBER? BENCHMARKING AND EVALUATING MEMORY MODALITY FOR EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory allows large language model (LLM) agents to use past experience, often stored as textual records, when responding to requests. For embodied agents that control robots, visual and spatial observations introduce a broader choice of memory modalities, affecting both the information available for subsequent tasks and the model inference costs of constructing and using memory. We introduce MeMoREval, a benchmark for studying this problem through a controlled experience-request protocol. Agents first *experience* a sequence of events without knowing the subsequent tasks, and subsequently they use memory to respond to information and action *requests* without receiving new observations. The benchmark comprises 53 tasks across four simulated indoor environments and measures task success alongside the inference costs of memory construction and use. Our reference evaluations reveal that storing frames outperforms storing generated descriptions at lower write-time inference cost for the primary model, while storing frames with scene graphs achieves the highest aggregate task success for every tested model. Extensions to accumulated experience, selective frame retention, and visual input to memory writers demonstrate how the benchmark accommodates further memory-design investigations. Two physical-robot demonstrations establish the feasibility of applying its protocol beyond simulation.

1 INTRODUCTION

A coding agent investigating a bug may uncover facts about a codebase that remain useful long after the immediate fix. Memory systems preserve such findings beyond the model's current context (Chhikara et al., 2025; Xu et al., 2025; Packer et al., 2023; Shinn et al., 2023; Wang et al., 2025b). For LLM agents that control robots, what is worth preserving extends to camera frames, object locations, and whether physical actions succeeded. This experience can be retained as textual descriptions, raw images, structured spatial records, or combinations of these forms, and the choice affects both what survives and what memory costs.

A textual record might note that several mugs were on a counter while omitting the colors needed to later retrieve a particular mug. Generating a description requires selecting what to mention before the future task is known, and discarded details may force a robot to return to a room or reopen a container. A stored camera frame retains visual details for interpretation once the task is known, while a structured spatial record makes object locations and relations directly available.

We introduce MeMoREval, a controlled **Eval**uation protocol and benchmark for studying **Me**mory **Mo**dalities of **R**obotic embodied agents. We distinguish the *memory modality*, which determines the form of retained evidence, from the *memory backend*, which organizes that evidence and makes it available to the agent. Our study addresses three questions: (**RQ1**) How do memory modalities affect success on information and action requests across task categories and models? (**RQ2**) How do these modalities trade off task success against write-time and read-time inference cost? (**RQ3**) How can MeMoREval inform memory-design decisions for deploying embodied agents?

Our evaluation protocol constructs each task from two phases. In the experience phase, the robot follows a scripted sequence of events while the memory module records its observations. In the request phase, the agent starts from a fresh context and uses memory to answer information requests

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.

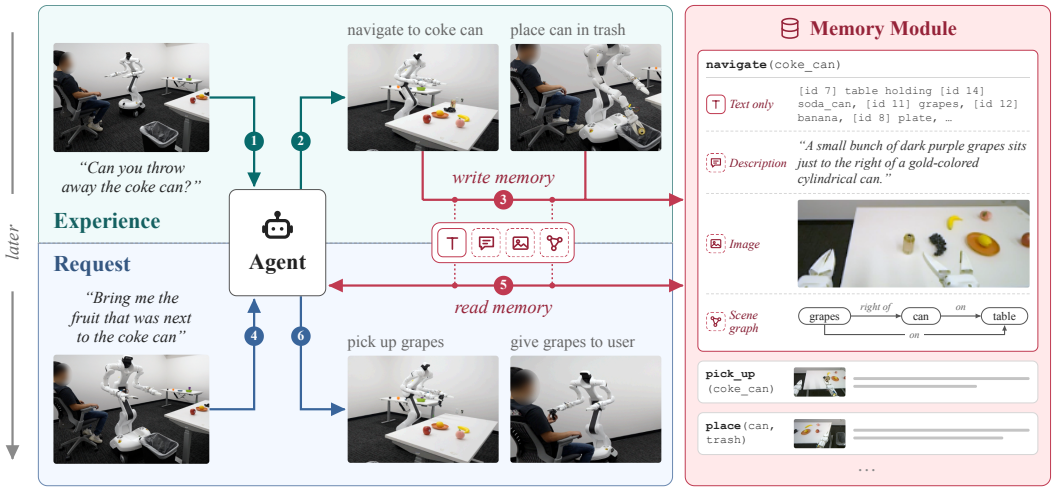


Figure 1: Overview of our embodied agent memory evaluation protocol, on a real robot. During the experience phase, the agent takes in an instruction (1) and executes corresponding actions (2) while writing into its memory (3). During the request phase, the agent receives another instruction (4) and reads its memory (5) for relevant information in order to perform the correct actions (6).

and plan action requests, as a robot would when a user poses a request. Plans are executed without returning new observations, so the agent cannot compensate for missing memory by searching again, which would reward exploration ability rather than memory. Experience phases can also be stacked to evaluate memory over longer histories.

We instantiate this protocol in 53 simulated tasks across four simulated indoor environments and seven categories of remembered information. MeMoREval supports textual records, generated image descriptions, camera frames, scene graphs, and their combinations. Using common recorded observations and a fixed agent implementation, we compare different memory configurations on task success as well as model inference costs, as both write-time and read-time costs are candidates to be optimized in embodied agent design.

Our reference evaluations show that storing camera frames with scene graphs achieves the highest aggregate task success for every tested model. For the primary model, archiving frames achieves higher success and lower write-time inference cost than generating descriptions, while also yielding the lowest read-time inference cost per successful request. The benefits of individual modalities vary across models and task categories, demonstrating the value of evaluating representations together with the models that use them.

We further examine deployment-oriented variants that accumulate experience across tasks, let writers select which frames to retain, or provide images to writers during memory construction. These variants preserve the benchmark's evaluation structure while extending the design choices under study. The resulting differences in task success, retention rates, and inference costs illustrate how MeMoREval can support investigations of memory organization, evidence selection, and writer inputs. Two demonstrations on a physical robot further establish that the protocol and memory interfaces can be instantiated using camera observations and an annotated spatial map.

Our contributions are:

- A controlled memory evaluation protocol for embodied agents that separates experience from subsequent information and action requests, supports accumulated histories, and measures write-time and read-time inference costs.
- MeMoREval, a benchmark of 53 tasks across four simulated environments, with configurable memory modalities and backends and complementary demonstrations on a physical robot.
- Reference evaluations across memory modalities, models, and backends that characterize trade-offs between task success and inference cost and identify opportunities for improving evidence retention and access in deployed embodied agents.

2 Related Work

Memory for LLM agents. Agent memory systems organize retained experience in different ways: MemGPT pages information between the context window and an external archive (Packer et al., 2023), Mem0 extracts and updates facts (Chhikara et al., 2025), A-MEM links evolving notes (Xu et al., 2025), Reflexion and Agent Workflow Memory distill feedback and reusable procedures (Shinn et al., 2023; Wang et al., 2025b), and filesystem memories let agents read and reorganize files through generic file tools (Zhou et al., 2026). Benchmarks such as LoCoMo and LongMemEval chiefly test recall across long conversations (Maharana et al., 2024; Wu et al., 2025), while MemLens and MemEye evaluate whether memory systems preserve visual evidence across extended multimodal interactions (Ren et al., 2026; Guo et al., 2026). We study the corresponding design choice for embodied agents, comparing memory modalities with the agent and observations held fixed and repeating the comparison across memory backends.

Memory for embodied agents. Object-centric scene graphs are widely used for robot mapping and planning (Gu et al., 2024; Wang et al., 2025a), so we include them as one storage modality alongside text, generated descriptions, and frames. Other systems combine scene graphs with images (Saxena et al., 2025) or record changing state through text and spatio-temporal relations (Zhang et al., 2026; Gorlo et al., 2026a; Ma et al., 2026). Retrieval and indexing methods improve the efficiency and accuracy of multimodal memory (Anwar et al., 2025; Long et al., 2026; Hu et al., 2026) and are complementary to the choice of what to store. Surprise-gated memory addresses which observations to preserve (Gorlo et al., 2026b); we examine this selection question by comparing writer-selected image retention with archiving every frame.

Embodied benchmarks with memory demands. Household benchmarks test long-horizon embodied interaction (Shridhar et al., 2020; Li et al., 2023; Yang et al., 2025), but memory failures there are confounded with exploration, planning, and manipulation ability. Memory-focused evaluations test question answering (Majumdar et al., 2024; Ginting et al., 2025), memory-dependent action (Yadav et al., 2025; Kwon et al., 2026), or both (Zhang et al., 2026), typically by comparing complete memory systems or agents. When the stored modality does vary, other factors vary with it: in OpenEQA, for example, caption- and scene-graph-based agents use a language model, whereas frame-based agents use a vision-language model (Majumdar et al., 2024). We design our protocol so that the stored modality is the main variable. Within each backend, the main memory conditions share the same recorded observations, agent, and writer input, and descriptions and frames are archived without passing through the writer. Requests start from a fresh context without new observations, so missing memory cannot be recovered by searching again. No-memory and full-information controls bound performance, information and action requests separate recalling a fact from acting on it, and write-time and read-time inference costs are measured separately.

3 Preliminaries: LLM-Based Embodied Agents and Memory

We consider LLM-based embodied agents in which a language or vision-language model selects physical skills for a robot. The interaction loop resembles that of a coding agent: the model receives a task and observations, calls a tool, and uses the returned feedback and updated observations to decide the following action. Each tool corresponds to a physical skill, such as navigating to a room, grasping an object, or opening a drawer, executed by a separate robot controller so the model chooses actions without generating motor commands. This scheme of wrapping robot capabilities as LLM-callable tools is also used by other works (Wang et al., 2026; Flexion Team). Observations include camera images, detected objects and their locations, user messages, and tool feedback. They may reach the model directly, or information from visual and spatial sensing may be converted into textual records. The observation depends directly on the robot's current state and perception. For example, opening a drawer reveals its contents, while moving to another room removes previously visible objects from view.

We treat memory as an external module that the agent uses through tool calls (Packer et al., 2023). The *agent* is the model together with its current context. At each step it infers the next tool call from its context, and information that leaves the context persists only if the *memory module* retains it. The memory module holds all persistent state and records observations, either through a *writer*, a model

that processes observations into memory records, or by archiving artifacts such as camera frames directly. It makes stored information available through context supplied to the agent and retrieval tools for particular records or artifacts, which the agent uses to answer questions or plan actions (Figure 1).

We characterize the memory module by two properties. The *memory modality* is the form of retained evidence, such as textual records, camera frames, or structured spatial records. The *memory backend* is how memory is organized and retrieved, for example as a rolling transcript or interconnected notes.

4 MeMoREval: Evaluation Protocol and Benchmark

We introduce MeMoREval, a benchmark for evaluating memory modalities in robotic embodied agents. We first describe its evaluation protocol, including task composition and measured performance metrics, such as task success and inference costs (§ 4.1), and then discuss the simulation environments and tasks that instantiate it (§ 4.2).

4.1 Evaluation Protocol

To evaluate how stored memory supports later decisions, we divide each task into two phases (Figure 1): experience, during which the agent observes events and constructs memory, and request, during which it uses that memory to respond to a user without receiving new information.

Experience. In deployment, experience comes from the robot's ordinary operation, such as carrying out an earlier request (Figure 1). In the benchmark, the robot instead executes a scripted sequence of actions while the memory module records the resulting observations and action outcomes. These actions and surrounding events provide information needed later: the robot may observe a mug being placed inside a cabinet before it is closed, or hear a user identify which of several mugs is theirs. The agent has not yet received any requests, so the memory module must retain information before knowing which details will matter. The experience phase is fixed per task across trials, so differences in performance reflect the memory configuration rather than what was observed.

Request. The request phase begins with a fresh agent context containing no observations from the experience phase. The agent receives a request that requires it to recover some information from its experience through the memory module without receiving new observations. Requests are of two types. *Information requests*, such as asking which cabinet contains a mug, test whether the agent can recover the relevant information from memory. *Action requests*, such as retrieving the mug, test whether the agent can use that information to act. For action requests, the agent submits a complete plan of skill calls, which the robot executes without returning observations or allowing the plan to be revised. The agent therefore cannot compensate for missing memory by searching the scene, for instance by opening cabinets until the mug appears. It does receive information about the robot's current state to facilitate the first steps of action planning, but this information does not remove the necessity of recovering facts from memory.

Stacked tasks. Separating experience from requests allows tasks to be stacked: the experience phases of several tasks are concatenated or interleaved into a longer history, followed by their requests. This tests whether earlier information remains available for later requests, hence accumulating interactions to form a more deployment-realistic scenario. Each component task retains its original grading criteria, allowing the construction of longer-horizon tasks without redesigning individual ones.

Outcomes and cost. Success for an individual task is defined as whether the agent correctly responds to all requests of the task, and for a stacked composite task, it is measured as the fraction of component tasks succeeded. We measure model inference cost separately for memory construction and use. *Write-time cost* accrues during the experience phase as the memory module processes observations, for example by summarizing text or describing images. *Read-time cost* accrues during the request phase as the agent queries memory and produces an answer or plan. Write-time cost is incurred before knowing downstream requests, while read-time cost arises only when responding to requests. Hence, write-time cost scales with the number of actions taken by the agent, and read-time cost scales with the number of requests issued to the agent. For a deployed embodied agent, both

Table 1: Task categories and example evaluation items.

Category	Tasks	Remembered information	Example evaluation items
Object location	11	where an object was last seen	"Which room is the satchel in?" + fetch
Spatial configuration	11	metric/ordinal layout	"What sat left of the storybook?"; restore a shelf
Appearance binding	10	color/pattern tied to an instance	fetch the *orange* mug among look-alikes
Set aggregation	8	counts and collections across rooms	"How many mugs are in the kitchen?"; gather all toys
User preference	5	user utterance → instance binding	fetch the mug the user said was theirs
Occluded state	4	contents behind doors and lids	"What was inside the wardrobe?"
Skill outcome	4	which action variant succeeded	re-shelve using the placement that worked

costs are highly relevant as robots could take frequent actions in short intervals (e.g., moving through a home through a sequence of navigation skill calls) and could be issued with requests frequently as well. Actual distribution of the two types of cost is determined by the exact application scenario, and therefore we report the two costs separately. To include failed attempts, we also report read-time cost per successful request.

4.2 Benchmark Environments and Tasks

Environments and robot. MeMoREval instantiates the protocol in the Genesis simulator (Genesis Authors, 2024) with four indoor environments adapted from pre-generated SceneSmith scenes (Pfaff et al., 2026): an office, photography studio, home, and clinic, each with seven to nine rooms and 200–300 objects (Appendix B). The simulated robot is a Dexmate Vega 1 bimanual mobile manipulator, which the agent controls through skills for navigation, opening and closing furniture, and picking and placing objects (Appendix B), which comprise typical capabilities for deployed robots. Grasping and placement use collision-checked motion plans, and navigation uses planner-validated locomotion paths.

Task categories and construction. MeMoREval contains 53 tasks across seven categories of remembered information: object location, spatial configuration, appearance binding, set aggregation, user preference, occluded state, and skill outcome (Table 1). These categories describe the information that must be remembered, rather than which memory modality is expected to perform best. Each experience phase consists of a scripted tour of the environment, followed by events such as an object being moved, a container being opened, or a user expressing a preference. We also construct a stacked composite task from 10–11 component tasks for each environment. Appendix B provides the individual tasks and composite construction.

Grading. All requests are graded deterministically. Information requests are checked against reference answers of a specified type, such as a region, object, count, or set. Action requests are checked against task-specific predicates on the final simulator state, such as whether the target object is held or rests on the required surface.

Observations and representations. We construct agent observations to reflect information available during robot operation: camera images provide visual evidence, localization and mapping provide object locations and relations, and interactions provide user instructions and action outcomes. The benchmark exposes this information through visual, spatial, and textual observation streams.

Visual observations are camera frames annotated with bounding boxes and integer object identifiers, which agents also use to refer to objects in tool calls. Spatial observations are scene-graph snapshots whose nodes store object identities and 3D positions, while edges record support and containment relations (Gu et al., 2024; Werby et al., 2024; Zhao et al., 2026). Textual observations report the robot's state, user messages, action outcomes, and currently detected objects with their categories, articulation states, and support and containment relations. In simulation, detections and spatial records are derived from simulator state restricted to observed objects. § 5.3 describes a physical instantiation using camera observations and an annotated spatial map.

Given these observations, a memory module can retain textual records, generate image descriptions, archive camera frames, or preserve spatial snapshots. The choice of memory modality determines how the robot's experience remains available for later requests. Description generation and other transformations of observations are part of memory construction. We subsequently describe the configurations evaluated in our experiments in § 5.1.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Memory conditions. Our main comparison varies the evidence archived for later retrieval while holding the memory writer's textual input fixed. For a given backend, task, and seed, the six main conditions share the textual memory constructed from the common observations in § 4.2. The *text* condition retains only this memory. The *frames* condition additionally archives the annotated camera frames, while *frames+graph* also archives scene graph snapshots. In order to investigate whether textual descriptions of camera frames can replace storing frames themselves, we further design the conditions of *desc@64*, *desc@256*, and *desc@1024*, each archiving model-generated descriptions of recorded frames at the indicated token budget. Each description is generated by a separate model call and, like a frame, stored verbatim for the agent to retrieve without passing through the writer. Stored frames, descriptions, and scene-graph snapshots are indexed by the experience step at which they were recorded. Each backend appends this step to every memory entry it renders, and the agent retrieves artifacts by passing step numbers to the retrieval tool of its condition. Frame and description retrieval return the stored artifacts, while scene-graph retrieval returns textual records of the requested objects and their relations.

Memory backends. We evaluate the memory conditions above across five backends that organize memory differently. A **transcript** retains a rolling history without an LLM-based writer, and a **wiki** writer maintains named pages through append and rewrite operations. We further adapt three published memory systems to this interface, **A-MEM** (Xu et al., 2025), **Mem0** (Chhikara et al., 2025), and **ObsMem** (Zhang et al., 2026), to the embodied setting. Every writer receives the same observations and the same instructions on what to remember, and all backends share the memory-context limit and artifact retrieval described above, so that they differ mainly in how memory is organized. Because this departs from the original systems, we present them as adaptations for the benchmark rather than reproductions (Appendix B).

Reference conditions. In addition to the five memory backends, we implement two control conditions that bypass memory construction. The *no-memory* condition tests performance without memory, while *full-information* supplies hand-authored textual facts sufficient for each task.

Deployment-oriented variants. The single tasks under the six main conditions form a controlled comparison: experience is short, the writer sees only textual observations, and artifact storage is automatic. Actual deployment of embodied agents could benefit from some relaxations, and we evaluate three of them. First, every main condition also runs on the four stacked composite tasks (§ 4.1). Second, because a deployed memory writer would have camera images at hand, *sighted* variants of *frames* and *frames+graph* that also give images to the writer test whether interpreting images during memory construction brings additional benefits over direct archival. Third, because archiving every frame grows storage with the length of deployment, a *selective-storage* variant of *frames* lets the memory writer decide whether to archive each frame using only the textual observations. The writer does not inspect the image, and only retained frames remain retrievable.

Models and runs. The main experiments use Gemini 3.7 Flash at temperature 0 for both the memory writer and the agent. Each of the six main memory conditions and five memory backends is evaluated on all 53 single tasks and the deployment-oriented variants, across three seeds for repeatability. We further test two alternative models, GPT 5.6 Sol and GLM 5.3 Flash, on the single tasks for all six main memory conditions and three backends (transcript, wiki, and A-MEM) with one seed, in order to evaluate the model-dependency of relative performance between memory conditions. Appendix B gives model settings and evaluation budgets.

Cost accounting. We report inference cost in weighted tokens, $T_{\text{in}} \times P_{\text{in}}/P_{\text{out}} + T_{\text{out}}$, where T_{in} and T_{out} are the numbers of input and output tokens and P_{in} and P_{out} their per-token prices. Input tokens include image tokens and output tokens include reasoning tokens. Weighted tokens therefore equal cost divided by the output-token price, so they are proportional to cost within each model. Because tokenizers and prices differ across models, we compare costs only across conditions within a model. Write-time cost covers writer updates and description generation during experience and is reported per experience step. Read-time cost covers all model calls in the request phase and is reported per attempted request as well as per successful request. Costs exclude sensing, spatial mapping, and storage; Appendix B lists prices, and Appendix A gives dollar costs.

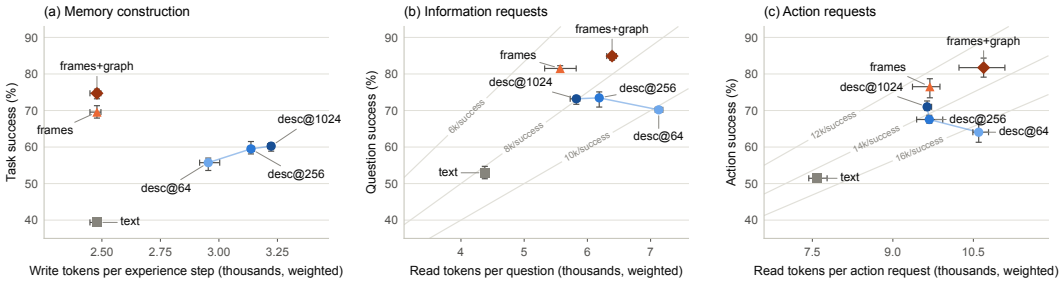


Figure 2: (a) Task success against per-experience-step write cost. (b) Question success against per-information-request read cost. (c) Action success against per-action-request read cost. Aggregated across all tasks, memory backends, and three seeds, for Gemini 3.7 Flash. The transcript backend is excluded from (a) as its writer is not LLM-based. Error bars show the range across seeds.

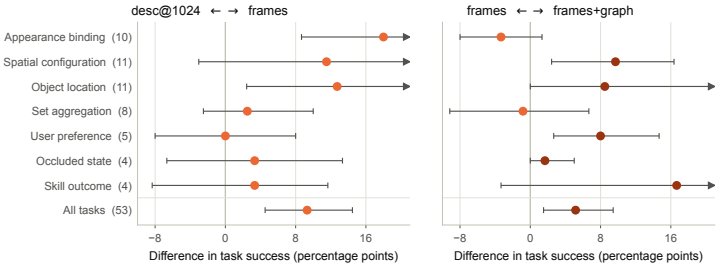


Figure 3: By-category breakdown of differences in task success (%) between *desc@1024*, frames, and frames with scene graphs. Error bars show 95% intervals from a paired bootstrap that resamples tasks with replacement.

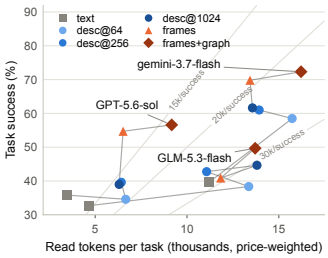


Figure 4: Task success (%) vs. tokens for three base models, six memory modalities, and three backends on one seed.

5.2 Results

Figure 2 compares task success against memory cost for Gemini 3.7 Flash, averaged across the five memory backends and three seeds. Error bars show the minimum and maximum across seeds. Appendix A reports the values behind each figure. We first examine the effect of memory modality on information and action requests, then its effect on inference costs, and lastly the behavior of different backends under deployment-oriented variants.

Controls. Without memory, the agent solves 3 of the 53 tasks; with full-information memory, it solves all 53. Every task is therefore solvable when the relevant facts are explicitly available as text. The following comparisons examine how effectively the tested memory representations preserve evidence that the agent can retrieve and use.

5.2.1 RQ1: How does memory modality affect task success?

Storing descriptions of camera frames achieves task success from 56% to 60% as the description budget grows from 64 to 1024 tokens, compared against 39% without any descriptions and 70% if storing the frames directly (Figure 2a). Since descriptions and frames are archived independently of the writer and retrieved through the same mechanism, the description gap therefore does not arise from a memory writer further summarizing or omitting the generated descriptions and instead reflects what information the different modalities store and how effectively the reader uses them.

Figure 3 shows the paired category-level differences between frames and the longest descriptions, and between frames with and without scene graphs. Error bars show 95% intervals from a paired bootstrap that resamples tasks with replacement and evaluates both conditions on the same resampled tasks. Frames improve mean success in six of the seven categories, with advantages in appearance binding and object location as shown through the bootstrap interval. Adding scene graphs raises overall success to 75%, but its effects also vary by category, with advantages on spatial config-

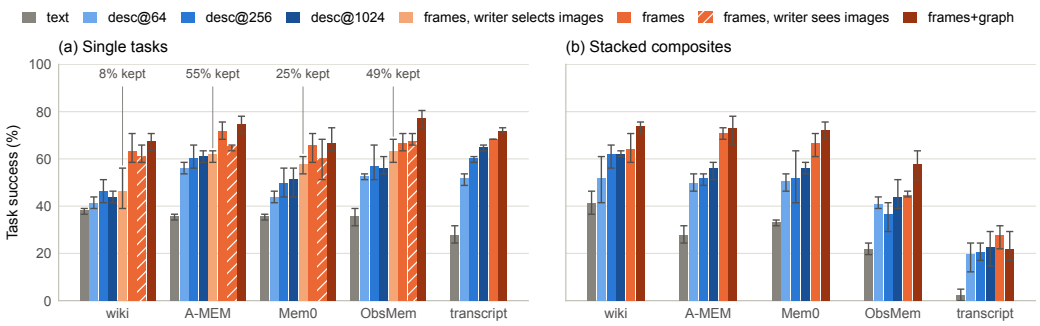


Figure 5: Task success (%) by memory backend and condition with Gemini 3.7 Flash. Both panels evaluate the same component tasks, individually in (a) and after stacking their experience histories in (b). Error bars show the range across seeds. "Kept" refers to the percentage of retained frames.

uration and user preference as reflected by the bootstrap. The paired-bootstrap intervals support improvements in aggregate task success from *desc@1024* to *frames* and from *frames* to *frames+graph*, although category-level effects vary. Storing frames together with scene graphs therefore achieves the highest aggregate task success among the main memory conditions under our implementation.

Figure 4 extends the comparison to three base models. We primarily compare changes across memory conditions within each model. In terms of task success, storing camera frames and spatial records achieves the highest aggregate task success among the six main conditions for every tested model. Storing frames alone achieves higher task success than *desc@1024* for Gemini 3.7 Flash and GPT 5.6 Sol, but not for GLM 5.3 Flash. Text, frames, and spatial records can encode the same information in different forms: for example, relative object positions may be visible in an image and explicitly recorded in a scene graph. The differing memory-condition rankings suggest that models vary in how effectively they use these representations. Combining textual memory, frames, and scene graphs offers alternative ways to recover shared facts and complementary information, potentially explaining why the combined condition achieves the best performance for each model. Differences in read-time token usage further show that models vary in the token expenditure associated with retrieving and using this evidence, yielding distinct trade-offs between task success and token efficiency. We next analyze this trade-off in depth for Gemini 3.7 Flash.

5.2.2 RQ2: How do memory modalities trade off success against cost?

For Gemini 3.7 Flash, archiving frames, with or without scene graphs, incurs lower write-time inference cost than generating descriptions, which requires additional model calls (Figure 2a). Longer descriptions cost more to generate than shorter descriptions but reduce read-time cost, with fewer tool calls and fewer reasoning tokens during requests (Appendix A). The longest descriptions have approximately the same read-time cost per request as stored frames.

Accounting for failed requests, frames achieve higher aggregate task success than descriptions while requiring approximately 15% fewer weighted tokens per successful information request and 7% fewer per successful action request than the longest descriptions. Adding scene graphs further improves success but increases read-time cost per successful request. The diagonal guides in Figure 2b—c show these success-adjusted costs.

5.2.3 RQ3: What do deployment-oriented variants show about memory design?

We next examine the deployment-oriented variants introduced in § 5.1: accumulated experience, selective image retention, and visual input to the memory writer. These variants extend the main modality comparison to practical memory-design choices while retaining the benchmark's experience–request structure and grading criteria.

Accumulated experience. Figure 5 shows that memory backends respond differently to longer histories. Transcript degrades substantially under stacking, because its rolling history exceeds the shared memory-context budget, truncating earlier observations and their retrieval cues from the

reader's context. The other adapted backends exhibit different responses to accumulated experience, suggesting that memory organization and retrieval influence how useful evidence remains accessible as histories grow. These results motivate further investigation of the underlying memory design.

Selective image retention. Archiving every observed frame causes storage to grow with experience. Selective retention therefore requires deciding which observations may support requests that are not yet known; recovering omitted evidence may later require additional physical interaction. Under the same selection instruction, the tested writers retain 8–55% of frames, with lower mean task success than automatic archival for each backend (Figure 5). The differing retention rates and performance changes expose the balance between reducing archival volume and preserving useful evidence, motivating retention policies suited to each backend's memory organization.

Visual input to the writer. Giving the writer access to camera images does not consistently improve task success (Figure 5). Its cost effects also vary, with some backends incurring lower write-time inference cost with approximately unchanged read-time cost (Appendix A). Thus, visual input during memory construction can affect construction cost without improving downstream success.

The changes in relative performance show why memory configurations should be evaluated under the history lengths and retention policies expected in their intended deployment. Together, these variants demonstrate how MeMoREval supports studying memory variants beyond the main modality comparison, exposing design questions relevant to deployed embodied agents.

5.3 Real-World Instantiation

We deploy our memory evaluation protocol on a physical Dexmate Vega 1 bimanual mobile manipulator, matching the simulated embodiment, using existing controllers for navigation and manipulation. We construct a 3D map using SLAM and manually annotate objects and semantic relations, including support relations, to obtain scene-graph records. Visual observations come from the robot's head-mounted camera, with object segmentation performed using SAM 3 (Carion et al., 2025).

We design two demonstrations focused on memory of object arrangements. In the first, the robot discards a soda can and should subsequently retrieve the fruit that had been beside it (Figure 1). In the second, the robot moves a lemon to another table, and should then recall that its original plate was orange, followed by retrieving the apple that had been to the lemon's right.

In the two demonstrations, the robot correctly used its memory and completed both deliveries without human assistance. We used *wiki* as the memory backend, using the *frames+graph* memory modality. These demonstrations establish that the protocol and memory interfaces can be instantiated on a physical robot using camera observations and a manually annotated spatial map, providing a concrete basis for future comparisons of memory configurations in physical environments.

6 Conclusion

We present MeMoREval, a benchmark for evaluating memory in embodied agents through a controlled experience–request protocol, which tests whether retained observations support subsequent recall and action, while measuring inference costs separately for memory construction and use. Our reference evaluations show that memory modality affects both task success and inference cost, with benefits varying across models and task categories. For the primary model, retaining camera frames achieves higher success than generating descriptions at lower write-time inference cost, while adding scene graphs further improves success at higher read-time cost.

Deployment-oriented variants demonstrate how the benchmark supports investigating additional memory-design choices. Accumulated experience affects backends differently, and selecting frames from textual observations reduces archival volume with losses in task success. Giving images to the writer does not consistently improve success, although it can reduce write-time cost for some backends. These findings motivate studying how memory organization, retention decisions, and writer inputs interact. Two physical-robot demonstrations instantiate the protocol using camera observations and an annotated spatial map, showing that the benchmark's protocol design can support evaluations beyond simulation. By providing common experiences, interfaces, and grading, MeMoREval offers an extensible experimental basis for developing memory systems for embodied agents.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on research methodology and experiments, implement methods, and support qualitative and thematic data analysis. We have not used generative AI tools to propose or refine hypotheses or interpret results. Generating synthetic datasets, developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, writing proofs, translation, and cleaning or reformatting datasets are not applicable to this work. Additionally, we used generative AI tools to create scientific figures, assist with literature review, and proofread and polish the manuscript. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Section 4 describes the evaluation protocol and grading procedures, and Appendix B provides experimental settings and memory-backend implementation details. We plan to publicly release the MeMoREval benchmark, evaluation code, and reference memory implementations upon acceptance.

REFERENCES

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. ReMEmbR: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2838–2845, 2025. doi: 10.1109/ICRA55743.2025.11127706.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025. URL https://arxiv.org/abs/2511.16719.

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. URL https://arxiv.org/abs/2504.19413.

Flexion Team. Flexion Reflect v1.0: The path towards long-horizon autonomous humanoid work. Technical blog. URL https://flexion.ai/news/flexion-reflect-v1.0. Accessed September 20, 2026.

Genesis Authors. Genesis: A generative and universal physics engine for robotics and beyond, December 2024. URL https://github.com/Genesis-Embodied-AI/Genesis.

Muhammad Fadhil Ginting, Dong-Ki Kim, Xiangyun Meng, Andrzej Marek Reinke, Bandi Jai Krishna, Navid Kayhani, Oriana Peltzer, David Fan, Amirreza Shaban, Sung-Kyun Kim, Mykel Kochenderfer, Ali-akbar Agha-mohammadi, and Shayegan Omidshafiei. Enter the mind palace: Reasoning and planning for long-term active embodied question answering. In *Proceedings of The 9th Conference on Robot Learning (CoRL)*, volume 305 of *PMLR*, pp. 5072–5106, 2025. URL https://proceedings.mlr.press/v305/ginting25a.html.

Nicolas Gorlo, Lukas Schmid, and Luca Carlone. Describe anything anywhere at any moment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026a. URL https://arxiv.org/abs/2512.00565.

Nicolas Gorlo, Derek K. Wise, Alberto Speranzon, and Luca Carlone. Worth remembering: Surprise-gated robot episodic memory. *arXiv preprint arXiv:2606.03787*, 2026b. URL https://arxiv.org/abs/2606.03787.

Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam

Paull. ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028, 2024. doi: 10.1109/ICRA57147.2024.10610243.

Minghao Guo, Qingyue Jiao, Zeru Shi, Yihao Quan, Boxuan Zhang, Danrui Li, Liwei Che, Wujiang Xu, Shilong Liu, Zirui Liu, Mubbasir Kapadia, Vladimir Pavlovic, Jiang Liu, Mengdi Wang, Yiyu Shi, Dimitris N. Metaxas, and Ruixiang Tang. MemEye: A Visual-Centric Evaluation Framework for Multimodal Agent Memory. *arXiv preprint arXiv:2605.15128*, 2026. URL https://arxiv.org/abs/2605.15128.

Yixun Hu, Zhicheng Zheng, Lihan Zha, Chunwei Xing, Rajdeep Singh, Omar Hossain, Antonio Loquercio, and Dhruv Shah. RAVEN: Long-horizon reasoning & navigation with a visuo-spatio-temporal memory, 2026. URL https://arxiv.org/abs/2606.25206.

Taeyoon Kwon, Dongwook Choi, Hyojun Kim, Sunghwan Kim, Seungjun Moon, Beong-woo Kwak, Kuan-Hao Huang, and Jinyoung Yeo. Embodied agents meet personalization: Investigating challenges and solutions through the lens of memory utilization. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2505.16348.

Chengshu Li, Ruohan Zhang, Josiah Wong, Cem Gokmen, Sanjana Srivastava, Roberto Martín-Martín, Chen Wang, Gabrael Levine, Michael Lingelbach, Jiankai Sun, Mona Anvari, Minjune Hwang, Manasi Sharma, Arman Aydin, Dhruva Bansal, Samuel Hunter, Kyu-Young Kim, Alan Lou, Caleb R. Matthews, Ivan Villa-Renteria, Jerry Huayang Tang, Claire Tang, Fei Xia, Silvio Savarese, Hyowon Gweon, Karen Liu, Jiajun Wu, and Li Fei-Fei. BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation. In *Proceedings of The 6th Conference on Robot Learning (CoRL)*, volume 205 of *PMLR*, pp. 80–93, 2023. URL https://proceedings.mlr.press/v205/li23a.html.

Lin Long, Yichen He, Wentao Ye, Yiyuan Pan, Yuan Lin, Hang Li, Junbo Zhao, and Wei Li. Seeing, Listening, Remembering, and Reasoning: A Multimodal Agent with Long-Term Memory. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2508.09736.

Ke Ma, Yamin Mao, Weiming Li, Shuai Tan, Yijie Zhong, Hao Chen, Haofen Wang, and Meng Wang. R4DSG: Relative 4D scene graph memory for object-centric question answering in long egocentric video. *arXiv preprint arXiv:2608.11017*, 2026. URL https://arxiv.org/abs/2608.11017. To appear at ACM Multimedia 2026.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating Very Long-Term Conversational Memory of LLM Agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024. doi: 10.18653/v1/2024.acl-long.747. URL https://arxiv.org/abs/2402.17753.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, Karmesh Yadav, Qiyang Li, Ben Newman, Mohit Sharma, Vincent Berges, Shiqi Zhang, Pulkit Agrawal, Yonatan Bisk, Dhruv Batra, Mrinal Kalakrishnan, Franziska Meier, Chris Paxton, Alexander Sax, and Aravind Rajeswaran. OpenEQA: Embodied question answering in the era of foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16488–16498, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Majumdar_OpenEQA_Embodied_Question_Answering_in_the_Era_of_Foundation_Models_CVPR_2024_paper.html.

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. URL https://arxiv.org/abs/2310.08560.

Nicholas Pfaff, Thomas Cohn, Sergey Zakharov, Rick Cory, and Russ Tedrake. SceneSmith: Agentic generation of simulation-ready indoor scenes. In *Forty-third International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2602.09153.

Xiyu Ren, Zhaowei Wang, Yiming Du, Zhongwei Xie, Chi Liu, Xinlin Yang, Haoyue Feng, Wenjun Pan, Tianshi Zheng, Baixuan Xu, Zhengnan Li, Yangqiu Song, Ginny Wong, and Simon See. MemLens: Benchmarking Multimodal Long-Term Memory in Large Vision-Language Models. *arXiv preprint arXiv:2605.14906*, 2026. URL https://arxiv.org/abs/2605.14906.

Saumya Saxena, Blake Buchanan, Chris Paxton, Peiqi Liu, Bingqing Chen, Narunas Vaskevicius, Luigi Palmieri, Jonathan Francis, and Oliver Kroemer. GraphEQA: Using 3D semantic scene graphs for real-time embodied question answering. In *Proceedings of The 9th Conference on Robot Learning (CoRL)*, volume 305 of *PMLR*, pp. 2714–2742, 2025. URL https://proceedings.mlr.press/v305/saxena25a.html.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. URL https://arxiv.org/abs/2303.11366.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10740–10749, 2020. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Shridhar_ALFRED_A_Benchmark_for_Interpreting_Grounded_Instructions_for_Everyday_Tasks_CVPR_2020_paper.html.

Qi Wang, Tianyi Wang, Chengyang Li, Shikun Ban, Yurun Chen, Yizhong Ge, Jason Qin, Chengtai Li, and Wentao Zhu. Towards the harness of embodied agents, 2026. URL https://arxiv.org/abs/2608.11246.

Zixuan Wang, Bo Yu, Junzhe Zhao, Wenhao Sun, Sai Hou, Shuai Liang, Xing Hu, Yinhe Han, and Yiming Gan. KARMA: Augmenting embodied AI agents with long-and-short term memory systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025a. doi: 10.1109/ICRA55743.2025.11128047.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent Workflow Memory. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025b. URL https://arxiv.org/abs/2409.07429.

Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3D scene graphs for language-grounded robot navigation. In *Robotics: Science and Systems (RSS)*, 2024. doi: 10.15607/RSS.2024.XX.077.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. LongMemEval: Benchmarking Chat Assistants on Long-Term Interactive Memory. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.10813.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-Mem: Agentic memory for LLM agents. In *Advances in Neural Information Processing Systems*, volume 38, pp. 17577–17604, 2025. doi: 10.52202/085713-0593. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/19909c36f51abc4856b4560aff3d36d6-Abstract-Conference.html.

Karmesh Yadav, Yusuf Ali, Gunshi Gupta, Yarin Gal, and Zsolt Kira. FindingDory: A benchmark to evaluate memory in embodied agents, 2025. URL https://arxiv.org/abs/2506.15635.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, Heng Ji, Huan Zhang, and Tong Zhang. EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. In *International Conference on Machine Learning (ICML)*, 2025. URL https://arxiv.org/abs/2502.09560.

Yehang Zhang, Jianchong Su, Haojian Huang, Yifan Chang, Tianhao Zhou, Xinli Xu, Yingjie Xu, Yinchuan Li, Zexi Li, and Ying-Cong Chen. Worldlines: Benchmarking and modeling long-horizon stateful embodied agents. *arXiv preprint arXiv:2606.18847*, 2026. URL https://arxiv.org/abs/2606.18847.

Shibo Zhao, Guofei Chen, Honghao Zhu, Zhiheng Li, Changwei Yao, Nader Zantout, Seungchan Kim, Wenshan Wang, Ji Zhang, and Sebastian Scherer. SuperMap: A spatio-temporal SLAM system for visual-language navigation. In *Robotics: Science and Systems (RSS)*, 2026. URL https://www.roboticsproceedings.org/rss22/p052.pdf.

Sizhe Zhou, Sheldon Yu, Hui Wei, Junda Wu, Siru Ouyang, Yizhu Jiao, Shijia Pan, Julian McAuley, Yu Zhang, Tong Yu, and Jiawei Han. Filesystem-based memory for LLM agents: Organization, evolution, and sustainability. *arXiv preprint arXiv:2607.26637*, 2026. URL https://arxiv.org/abs/2607.26637.

A Additional Results

Tables 2, 4, 5, and 6 list the values behind Figures 2–5. Table 3 gives dollar costs and read-time usage for the main conditions, and Table 7 gives the cost of writer image access by backend.

Scope and interpretation

Our reference evaluations characterize memory configurations under MeMoREval's task distribution, observation interfaces, prompts, model choices, and evaluation budgets. Results for adapted backends apply to the implementations described here. The task suite probes selected memory demands, and its success rates should not be interpreted as estimates of success in deployment. The benchmark provides a common basis for investigating design trade-offs; the applicability of specific findings to other implementations, workloads, or deployment settings requires separate evaluation.

Table 2: Values behind Figure 2: success and inference cost by memory condition with Gemini 3.7 Flash, pooling 53 tasks, five backends (wiki, A-MEM, Mem0, ObsMem, and transcript), and three seeds (795 runs per condition). Info. and Action give success on information and action requests. Write tokens per step are the price-weighted write-time tokens of the scene tours and task experience segments divided by their number of steps, computed over the backends with a learned writer (wiki, A-MEM, Mem0, and ObsMem). Text, frames, and frames+graph give the writer identical input, so their write cost is pooled over the three conditions. Read tokens per success are the price-weighted read-time tokens for a request type divided by the number of such requests passed, the quantity marked by the diagonals in Figure 2b–c. Weighted tokens are $T_{\text{in}} \times P_{\text{in}}/P_{\text{out}} + T_{\text{out}}$, at the model's prices in Table 10. The reference conditions run once per task at seed 0. Per-seed task success varies by at most 3.4 percentage points within a condition.

	Success (%) ↑			Write tokens per step ↓	Read tokens per success ↓	
Condition	Task	Info.	Action	(k, weighted)	Info. (k)	Action (k)
text	39	53	51	2.48	8.3	14.7
desc@64	56	70	64	2.95	10.2	16.5
desc@256	59	73	68	3.14	8.4	14.3
desc@1024	60	73	71	3.22	8.0	13.6
frames	70	82	77	2.48	6.8	12.7
frames+graph	75	85	82	2.48	7.5	13.1
no memory	6	17	0	–	–	–
full information	100	100	100	–	–	–

Table 3: Dollar cost and read-time usage by memory condition with Gemini 3.7 Flash, pooling five backends (wiki, A-MEM, Mem0, ObsMem, and transcript) and three seeds, at the prices in Table 10. Write cost is per thousand experience steps, computed as in Table 2. Read cost is per successful request of each type. Calls are model calls per request, each issuing one tool call (a retrieval, an answer, or a plan); reasoning tokens are per request.

	Write $ per 1k steps ↓	Read $ per success ↓		Per request	
Condition		Info.	Action	Calls	Reasoning tokens (k)
text	9.30	0.031	0.055	1.0	4.2
desc@64	11.08	0.038	0.062	2.7	4.0
desc@256	11.76	0.032	0.054	2.4	3.4
desc@1024	12.09	0.030	0.051	2.3	3.3
frames	9.30	0.026	0.047	2.1	3.1
frames+graph	9.30	0.028	0.049	2.5	3.1

Table 4: Values behind Figure 3: task success (%) by category with Gemini 3.7 Flash, pooling five backends (wiki, A-MEM, Mem0, ObsMem, and transcript) and three seeds, and the plotted differences with 95% intervals from a paired bootstrap that resamples tasks (1,000 resamples). Each task contributes fifteen runs per condition.

		Success (%)			Difference (pp) [95% interval]	
Category	Tasks	desc@1024	frames	frames+graph	frames − desc@1024	frames+graph − frames
Appearance binding	10	55	73	69	+18.0 [8.7, 27.3]	−3.3 [−8.0, 1.3]
Spatial configuration	11	56	67	77	+11.5 [−3.0, 27.9]	+9.7 [2.4, 16.4]
Object location	11	77	90	98	+12.7 [2.4, 24.2]	+8.5 [0.0, 22.4]
Set aggregation	8	58	61	60	+2.5 [−2.5, 10.0]	−0.8 [−9.2, 6.7]
User preference	5	63	63	71	+0.0 [−8.0, 8.0]	+8.0 [2.7, 14.7]
Occluded state	4	48	52	53	+3.3 [−6.7, 13.3]	+1.7 [0.0, 5.0]
Skill outcome	4	53	57	73	+3.3 [−8.3, 11.7]	+16.7 [−3.3, 36.7]
All tasks	53	60	70	75	+9.3 [4.5, 14.5]	+5.2 [1.5, 9.4]

Table 5: Values behind Figure 4: task success and read tokens by base model at seed 0, pooling three backends (159 runs per cell). Each model serves as writer, describer, and agent on the same recorded tours. Weighted tokens are $T_{\text{in}} \times P_{\text{in}}/P_{\text{out}} + T_{\text{out}}$ per task, at each model's own prices (Table 10).

Model	text	desc@64	desc@256	desc@1024	frames	frames+graph
Task success (%)						
Gemini 3.7 Flash	40	58	61	62	70	72
GPT 5.6 Sol	36	35	40	39	55	57
GLM 5.3 Flash	33	38	43	45	41	50
Read tokens per task (thousands, price-weighted)						
Gemini 3.7 Flash	11.2	15.7	13.9	13.6	13.4	16.2
GPT 5.6 Sol	3.4	6.7	6.4	6.3	6.5	9.2
GLM 5.3 Flash	4.7	13.4	11.1	13.8	11.8	13.7

Table 6: Values behind Figure 5: task success (%) by memory backend and condition with Gemini 3.7 Flash, pooling three seeds. Single-task rows cover the 41 tasks that appear in the stacked composites. In a stacked composite, a component task passes when all of its requests pass. In the writer-selects condition, the writer chooses which frames to archive; kept is the percentage of offered frames it retains, with one frame offered per experience step across the scene tours and task experience segments. When the writer sees images, it also receives the camera frames. The transcript has no writer. All backends are our adaptations (Table 11).

	Main conditions						frames variants		
Backend	text	desc@64	desc@256	desc@1024	frames	frames+graph	writer selects	kept (%)	writer sees images
Single tasks									
wiki	38	41	46	44	63	67	46	8	61
A-MEM	36	56	60	61	72	75	62	55	65
Mem0	36	44	50	51	66	67	58	25	60
ObsMem	36	53	57	56	67	77	63	49	67
transcript	28	52	60	65	68	72	–	–	–
Stacked composites (component tasks passed)									
wiki	41	52	62	62	64	74	–	–	–
A-MEM	28	50	52	56	71	73	–	–	–
Mem0	33	50	52	56	67	72	–	–	–
ObsMem	22	41	37	44	45	58	–	–	–
transcript	2	20	20	23	28	22	–	–	–

Table 7: Task success and cost of writer image access by backend with Gemini 3.7 Flash, on all 53 tasks and three seeds. The default writer receives only textual observations; when the writer sees images, it also receives the camera frames. Default write cost is pooled over text, frames, and frames+graph, and tokens are price-weighted, as in Table 2. The transcript has no writer and is omitted.

	Task success (%)		Write tokens per step (k)		Read tokens per task (k)	
Backend	default	writer sees images	default	writer sees images	default	writer sees images
frames						
wiki	67	64	1.48	1.81	14.1	14.6
A-MEM	69	64	1.73	1.90	11.3	11.6
Mem0	69	60	3.22	2.68	13.9	14.9
ObsMem	68	68	3.48	2.08	14.8	15.1
frames+graph						
wiki	70	73	1.48	1.89	15.6	17.5
A-MEM	74	71	1.73	1.88	13.3	13.4
Mem0	72	72	3.22	2.67	15.1	16.0
ObsMem	80	77	3.48	2.09	16.6	16.7

B EXPERIMENTAL DETAILS

Table 8 summarizes the four scenes and their stacked composites, and Table 9 lists the 53 tasks. Table 10 gives the model settings, evaluation budgets, and prices shared by all conditions, and Table 11 describes the memory backends and how they differ from the published systems.

Table 8: Scenes and stacked composites. Every task's experience phase begins with the scene's scripted tour. A stacked composite combines the tour with the experience segments of its member tasks (Table 9), followed by all of their requests, under the same memory-context limit as single tasks. Experience steps count the tour and the member segments.

			Stacked composite		
Scene	Tasks	Tour steps	Member tasks	Requests (info / action)	Experience steps
office	12	10	10	11 / 10	37
studio	14	8	10	11 / 10	69
home	14	11	10	11 / 10	71
clinic	13	12	11	11 / 10	100

Table 9: The 53 tasks by scene and category: loc (object location), spat (spatial configuration), app (appearance binding), set (set aggregation), pref (user preference), occl (occluded state), and skill (skill outcome). Unless marked, a task has one information request and one action request. Tasks marked info only have no action request, skill-outcome tasks have only an action request, and three preference tasks and qa_inventory have two information requests. Tasks marked † are not members of the scene's stacked composite.

Scene	Tasks (category)
office (12)	loc_kettle (loc), reloc_tumbler (loc), vis_book_room (loc), spatial_row_neighbor (spat), vis_tumbler_color (app), vis_mug_perch (app), vis_bowl_stack (app), vis_bowl_station (app), qa_inventory† (set, info only), collect_succulents (set), pref_water_carafe (pref), proc_credenza_return† (skill)
studio (14)	loc_gaffer_tape (loc), reloc_satchel (loc), vis_mug_room (loc), spatial_shelf_vacate (spat), vis_row_neighbor (spat), vis_stack_order† (spat), vis_cart_clearout (spat), vis_candle_color (app), qa_office_twins† (app, info only), qa_inventory_candles† (set, info only), collect_shoes (set), pref_client_plant (pref), vis_drawer_reveal (occl), proc_shelf_return† (skill)
home (14)	loc_teakettle (loc), reloc_remote (loc), vis_teddy_rooms (loc), spatial_row_neighbor (spat), spatial_shelf_vacate (spat), vis_stack_order† (spat), vis_shelf_clearout (spat), vis_mug_color (app), vis_storybook_cover† (app, info only), qa_inventory_mugs† (set, info only), collect_toys (set), pref_snack (pref), vis_container_reveal (occl), proc_desk_return† (skill)
clinic (13)	loc_diagnostic_kit (loc), reloc_clipboard (loc), spatial_row_neighbor (spat), spatial_shelf_vacate (spat), vis_sanitizer_color (app), vis_medbox_shelf (app), qa_inventory_tissues (set, info only), collect_gloveboxes (set), pref_allergy_med (pref), pref_exam_setup (pref), vis_cabinet_reveal (occl), vis_locker_reveal† (occl, info only), proc_pharmacy_return† (skill)

Table 10: Settings shared by all memory conditions and backends.

Observations	
Text	User messages, skill feedback, and detected objects arranged by support and containment, with categories and articulation states but no color or texture. An object is detected when its segmentation mask exceeds a minimum pixel area.
Frames	640×480, with a box and integer identifier for each detected object.
Scene graph	Object identifiers, world-frame 3D positions, and support and containment relations, taken from the simulator. No appearance attributes and no shelf-level, row-order, or viewpoint fields.
Identifiers	Assigned when an object is first observed in a run; the assignment order varies with the seed.
Memory	
Context limit	20,000 characters of rendered memory per agent turn in every backend. Retrieved frames, descriptions, and scene-graph records do not count toward this limit.
Step references	Memory entries are tagged "[step N]"; frames, descriptions, and scene-graph snapshots are retrieved by step.
Descriptions	Generated by the same model as the writer and agent, from the annotated frame and the textual observation of the same step. For a budget of k tokens, the describer is asked for about $0.75k$ words; the output is trimmed to a sentence boundary within 10% of the budget and regenerated once if cut off.
Requests	
Context	Every request starts from a fresh agent context containing only the request and the name of the room the robot is in.
Order	Within a task, information requests are issued before the action request.
Action plans	One complete plan, executed open-loop without observations or replanning; execution continues past failed steps.
Stacked composites	Experience segments run in a fixed order. All information requests follow, then all action requests, each group in a fixed order that is independent of the segment order. Before each action request, the simulator is restored to its state at the end of the experience phase, so an earlier plan's effect on the scene cannot make a later, correct plan unexecutable under collision-checked motion.
Agent	
Model settings	Temperature 0.
Turn budget	Six turns per request; each retrieval call or answer uses one turn.
Retrieval	Frames and descriptions: up to four steps per call. Scene graph: up to six objects from one snapshot.
Skills	`navigate`, `open`, `close`, `pick`, `place`; request actions `answer`, `submit_plan`, `done`.
Cost accounting	
Prices	USD per million prompt/completion tokens: Gemini 3.7 Flash 0.75/3.75, GPT 5.6 Sol 4/20, GLM 5.3 Flash 0.15/0.50.
Tokens	Image tokens count as prompt tokens and reasoning tokens as completion tokens. Prompt-cache discounts are ignored.

Table 11: Memory backends. All five are our implementations under a shared interface: writers receive the same salience instruction, without the list of task categories, attach the same step references to entries, and render into the same context limit. Results describe these adaptations rather than the original systems.

Backend	Organization	Changes from the published system
transcript	Rolling log of observations, keeping the most recent history that fits the context limit; no learned writer.	–
wiki	Named pages that the writer maintains through append and rewrite operations.	–
A-MEM	Interconnected notes that the writer links and evolves as new notes arrive (Xu et al., 2025).	Lexical (Jaccard) retrieval instead of dense embeddings; note evolution batched per ingest step rather than per neighboring note.
Mem0	Memory entries extracted by the writer, plus a recent-window context (Chhikara et al., 2025).	Lexical retrieval; rolling summary and graph variant omitted.
ObsMem	Observational memory: beliefs about the scene that the writer revises as observations arrive (Zhang et al., 2026).	Lexical, flat retrieval instead of the query planner; LLM belief revision instead of deterministic counters; commitment track omitted.