NAVHARNESS: TOWARDS LIFELONG EMBODIED NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Frontier models can now perform well on individual embodied navigation tasks through multi-round multimodal reasoning with simple tools. Across successive tasks, however, an agent must also rely on an evolving map and earlier search records, both of which may be incomplete or conflict with new observations. We present **NavHarness**, a training-free embodied harness towards lifelong navigation that makes memory processing part of the navigation loop. During navigation, its multi-round agentic session draws on maps, task records, and house knowledge, checking them against observations and recording corrections to guide its actions. NavHarness preserves this experience across fresh conversations for new tasks or recovery attempts, while outcome verification and run-end summaries support its later reuse. On GOAT-Bench, NavHarness improves s-SR over context-only independent sessions by 18.6 points with Astra and 22.6 with Opus 5. Using SLAM-estimated poses, NavHarness with GPT-6 Astra achieves state-of-the-art task success of 83.7 s-SR with 36.9 e-SR on GOAT-Bench and 85.9 s-SR on IR2R-CE. To understand these gains, we examine how experience is carried between sessions and find that structured recovery handovers outperform length-matched summaries. In extended deployments across houses, consolidation improves navigation beyond retaining maps and task records, with case studies showing how agents use earlier experience to interpret new goals, investigate unresolved questions, and resume failed searches. We suggest that progress towards lifelong navigation depends on how successive reasoning sessions build on prior experience, alongside improvements in single-task capability.

1 INTRODUCTION

Consider a household robot navigating a home throughout the day, moving from one request to the next, revisiting familiar rooms, and finding its way through places it has yet to explore. Advances in multi-round multimodal reasoning bring this vision closer, enabling frontier models to navigate effectively within a single task using a basic coding harness and simple observation and movement tools, without navigation-specific training (Zhou et al., 2026). For a robot that keeps working, each journey also reveals more of the home and leaves experience that could help with a later request. Extending this capability towards lifelong navigation requires the robot to draw on that experience when deciding where to go and what to inspect next.

In continuous household navigation, each task starts where the previous one ended, so a new request arrives in a home the robot has already begun to explore. Restarting the conversation leaves the robot where it is, so the next reasoning session needs the map and search evidence accumulated so far. Yet a record that an object was not found in a room does not establish that every relevant part was inspected, and a stored description may conflict with the current view. To choose where to search next, the navigator must interpret these records against current observations and revise them when the evidence changes.

Long-horizon navigation has made substantial progress toward this goal by retaining scene knowledge across tasks (Zhao et al., 2024), retrieving past observations for exploration (Yang et al., 2025; Niu et al., 2026; Lu et al., 2026), and learning policies that query accumulated experience (Wang et al., 2026c). Building on this progress, we study how a general-purpose reasoning session can use these different sources together as it navigates. For example, a house note may suggest a room and the

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.

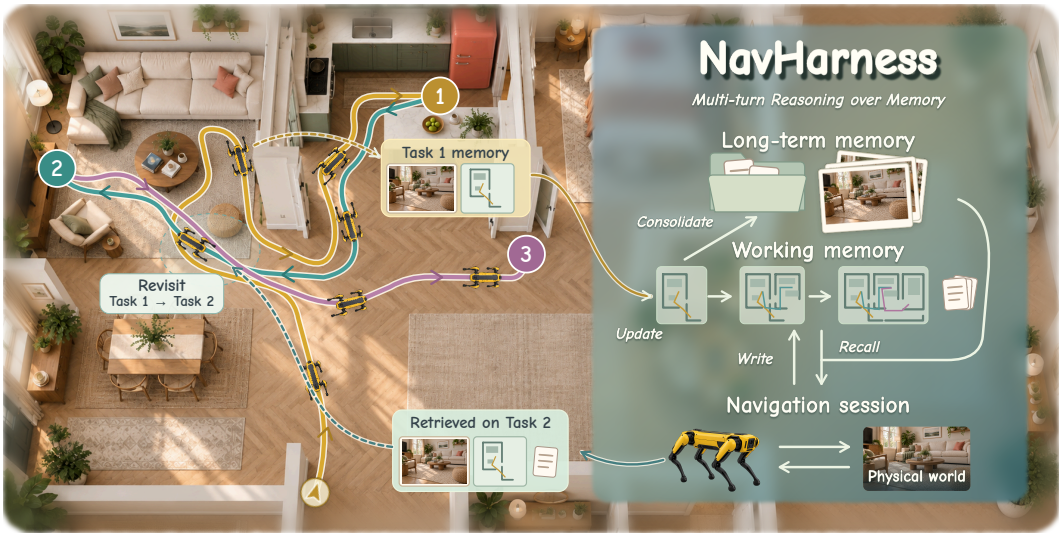


Figure 1: **NavHarness** orchestrates long-horizon embodied navigation through multi-round multi-modal reasoning over working memory and house knowledge, retaining experience for successive tasks.

map a route to it, while a failed-search record points to an overlooked corner. On reaching the room, the robot can compare these leads with what it sees to decide whether to continue searching there, consult another record, or correct the note. This makes memory retrieval and revision part of the ongoing interaction with the environment.

Multi-round agentic sessions offer a practical starting point because memory and coding agents already use tools to inspect external records and pursue further queries based on what they find (Packer et al., 2023; Rajasekaran et al., 2025; Young, 2025). In navigation, this interaction lets a session read a search record, compare it with the current view, and consult the map before choosing its next action, without training a separate memory policy. Keeping these records outside the conversation also allows a fresh session to inherit evidence about inspected places, supported exclusions, and unresolved searches. Since later searches depend on these records, outcome checks help prevent incorrect completion claims from becoming accepted experience.

We introduce **NavHarness**, an embodied harness towards lifelong navigation centered on memory processing without additional model training. Each multi-round agentic session reasons over current observations, spatial and task memory, and longer-term house knowledge, while an outer loop retains these memories as new tasks or recovery attempts start fresh conversations. The loop also checks task outcomes and consolidates the resulting records into house notes for later runs. On GOAT-Bench, NavHarness improves s-SR by 18.6 points with Astra and 22.6 with Opus 5 compared with the same models using a fresh, context-only session for each task, without external memory (Section 4.2). Using SLAM-estimated poses rather than simulator ground-truth poses, NavHarness with GPT-6 Astra reaches 83.7% s-SR and 36.9% e-SR on GOAT-Bench and 85.9% s-SR on IR2R-CE, exceeding previous state-of-the-art results. Extended deployments show how retained experience redirects search, prompts observations to resolve uncertainty, and lets failed attempts inform later decisions.

To summarize, our contributions are as follows:

1. We show that how experience is used matters beyond retaining it. Structured handovers outperform length-matched summaries. NavHarness brings this evidence transfer into a navigation loop where fresh reasoning sessions continue from the robot's current state using evidence from earlier searches.
2. We improve all four backbones and achieve state-of-the-art GOAT-Bench and IR2R-CE performance using SLAM-estimated poses rather than simulator ground-truth poses, with gains in task success and path efficiency over same-backbone independent sessions.
3. Continuous deployment shows that consolidation improves navigation beyond retaining

maps and task records. Case studies reveal how agents reuse failed searches and revise earlier accounts, informing the design of future long-horizon embodied agents.

2 Related Work

Long-horizon and lifelong navigation. IR2R-CE and GOAT-Bench evaluate successive goals in shared environments, where accumulated experience can improve later navigation (Krantz et al., 2023; Khanna et al., 2024). OVER-NAV, 3D-Mem, SSMG-Nav, and GSMem retain spatial knowledge through structured objects, visual snapshots, semantic graphs, and renderable memories (Zhao et al., 2024; Yang et al., 2025; Niu et al., 2026; Lu et al., 2026), helping agents connect past observations to places they can revisit. Beyond scene representations, SeqWalker uses hierarchical planning and trajectory correction (Han et al., 2026), MemoryExplorer trains active retrieval with single-round tool invocation (Wang et al., 2026c), and Uni-Walker and AllDayNav pursue continued policy learning (Wang et al., 2026d; Yin et al., 2026). We share this goal, focusing on how general-purpose multi-round sessions consult and revise spatial memory, task records, and house knowledge without navigation-specific training.

Embodied harnesses and agentic control. MIP shows that multi-round multimodal reasoning can control navigation through observation and movement tools in a basic coding harness (Zhou et al., 2026). Navigation interfaces and spatial tools extend this approach (Li et al., 2026d; Deng et al., 2026; Lei et al., 2026), while embodied harnesses connect reasoning to robot controllers and reusable skills (Zhang et al., 2026; Chen et al., 2026d; Wang et al., 2026b; Elmaaroufi et al., 2026). Persistent experience also supports skill and harness evolution (Wang et al., 2024; Ding et al., 2026; Wang et al., 2026a). HarnessVLN uses event memory and a spatiotemporal graph to validate navigation proposals (Chen et al., 2026b). RoboHarness uses execution memories to select heterogeneous policies and prepare the physical state for policy handoffs (Huang et al., 2026). NavHarness instead studies what must cross a reasoning-session boundary. A fresh conversation starts from the robot's current physical state and needs to know which places were inspected, what those observations exclude, and where search remains incomplete. We test how retrieving, handing over, and consolidating this evidence supports later navigation.

Memory processing in long-horizon agents. External memory lets agents retain experience beyond a context window without encoding each new experience in model parameters. Generative Agents and A-MEM turn observations into retrievable, revisable records (Park et al., 2023; Xu et al., 2025), while Reflexion and ACE preserve lessons from earlier attempts (Shinn et al., 2023; Zhang et al., 2025b). To use a growing memory within a limited context window, agents employ compression, selective eviction, and agent-directed context management (Kang et al., 2026; Semenov & Dorofeev, 2026; Hao et al., 2026; Li et al., 2026b). MemGPT lets a model move information between memory tiers, and CoALA places memory access within the agent's decision process (Packer et al., 2023; Sumers et al., 2024). In coding harnesses, multi-round tool interaction lets agents read and update persistent files to carry progress between sessions (Yang et al., 2024; Wang et al., 2025; Rajasekaran et al., 2025; Young, 2025; Rajasekaran, 2026). NavHarness builds on these practices for physical search, where checking records may require revisits. Fixed-retrieval and ordinary-summary controls test the value of agent-directed access and structured search handovers.

3 NavHarness

Term	Meaning and relation
Run (deployment)	A sequence of K tasks in one house.
Task	Pursuit of goal g_k, possibly through several attempts sharing one task budget.
Attempt	One continuous search for the current goal, supported by one navigation session.
Session (navigation)	The multi-turn multimodal conversation that serves as the reasoning core for an attempt.
Hop	A bounded block of model turns and tool calls within a session.

Task setup. Unlike single-episode navigation evaluation (Anderson et al., 2018b), we study successive navigation goals with reusable experience. Each goal is specified by language, an object

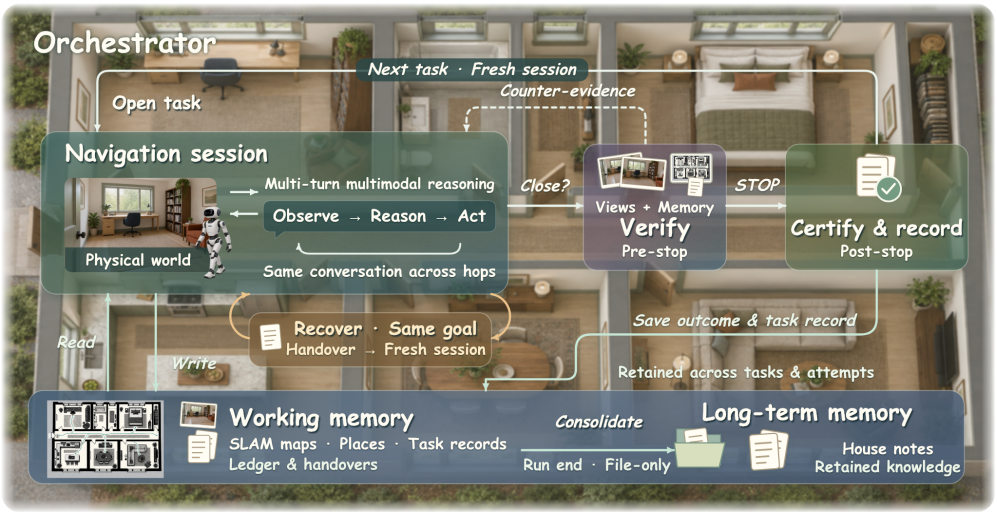


Figure 2: **NavHarness execution flow.** The orchestrator manages navigation sessions and persistent memory. Verify and Certify denote pre-stop verification and post-stop certification, respectively.

category, or an image. Within a continuous sequence, each task starts at the previous task's endpoint. *Recovery* replaces the conversation while retaining physical state, external memory, and remaining budget for another attempt at the same goal. *Task closure* records the outcome before the next goal, including after budget exhaustion. At run end, *consolidation* summarizes task records for future runs.

The outer *orchestrator* manages session boundaries, handles recovery and completion requests, and retains memory. It advances the session in hops without clearing the conversation, tracking the task budget and robot state between them. Figures 1 and 2 show memory reuse and execution, respectively. Appendix A gives the six-component specification (Meng et al., 2026) and full algorithm.

3.1 Starting a task with retained experience

On receiving a new goal, the orchestrator sets model-turn and movement-step budgets and opens a fresh coding-agent conversation with access to persistent working memory, instructing it to read the referenced files before moving. A *handover* preserves search experience for a later session. At task closure, a *task handover* summarizes the goal, searched places, outcome evidence, and unfinished searches. During recovery, a *recovery note* instead passes evidence and remaining options to a fresh attempt at the same goal (Section 3.3). The *ledger* is a compact index with one entry per closed task, listing its goal, recorded status, and named places. It helps later sessions find relevant tasks and open their detailed records through file tools, without carrying over earlier conversations.

The navigation session accesses three memory tiers. The *active context* holds the current attempt's goal, reasoning, observations, and retrieved information. *Working memory* persists outside the conversation across tasks and attempts, pairing task records with spatial memory comprising occupancy maps built using SLAM-estimated poses, named places and photographs, and room and floor connections. Task records, which form the journal, include the ledger, task handovers, recovery notes, and completion-check results (Section 3.4). *Long-term memory* comprises four files, together called *house notes*, containing an index, a house overview, room notes, and navigation skills. They cover room connections, landmarks, useful routes, failed searches, places ruled out by evidence, and unresolved questions. Navigation sessions read these files to guide searches, but only consolidation updates them (Section 3.5). Saved maps and views persist separately from these textual notes for reuse in later runs (Appendix A).

3.2 Navigating by consulting and updating memory

Starting from this context, the navigation session interleaves observation, reasoning, action, and further memory access (Zhou et al., 2026). A house note can suggest a room, a task record can

explain what was already searched, and a map query can show how to reach that room. Tool responses bring retrieved information into the conversation for comparison with new observations, helping the model decide whether to follow an earlier account or query further. Following prior work, we use the standard navigation action space of forward movement (0.25 m), left/right turns (15°), and STOP.

Reaching a remembered place also requires locating the robot and planning through explored space. ORB-SLAM3 (Campos et al., 2021) estimates the robot's position and orientation (pose) from color and depth (RGB-D) observations. A mapper uses these estimates to build an occupancy grid for each floor, distinguishing free, occupied, and unexplored space. As the robot moves, the mapper updates the grids while pose tracking is available, pausing updates if tracking is lost. The navigation session queries this growing map to inspect explored space and preview routes, and can mark named places with saved views for later reference (Appendix A.2).

Closure and recovery tools pass requests to the orchestrator, which handles them when the hop returns. It checks the remaining budget and tracking status. If the environment has already ended the task, the orchestrator records its outcome. Otherwise, it handles a pending closure request before considering recovery. If neither requires a transition, it resumes the same conversation with its context intact. The coding backend manages the transcript within that conversation, while NavHarness opens fresh sessions at task and recovery boundaries. Separating transcript management from these boundaries lets different reasoning cores use the same memory rules (Appendix A.3).

3.3 Recovering with a fresh conversation

When a session repeatedly follows an unproductive plan, recovery lets a fresh conversation reconsider the search using retained evidence. Recovery begins with a session request or when the attempt reaches a configured turn threshold. The orchestrator checks exploration progress, tracking status, and previous attempts before approving a restart and deciding whether location also needs reassessment. Before replacement, the outgoing session writes a *recovery note*, a handover to the next attempt at the same goal. With movement disabled, it records searched places, evidence-supported exclusions, a reliable landmark, untried options, and uncertainty. Distinguishing *searched* from *ruled out* is important because visiting a room does not establish that every relevant object was seen. The fresh session reads the note and resumes the same goal from the current location, retaining the map, task records, house notes, and remaining budget without inheriting the old conversation.

The recovery *scope* specifies what the new attempt should reconsider, from its search plan to its location, route, floor, or understanding of the house, without deleting stored memory. When location needs reassessment, a separate *wake-up session* compares surrounding views with saved place photographs and gives the new navigation session a short account of its likely location. Without saved photographs, the navigation session reasons about its location from the supplied views. This visual recognition helps orient the search, while SLAM estimates the robot's metric pose (Appendix A.5).

3.4 Checking completion and passing on the task record

When the navigation session decides to stop searching, it writes a task handover of its search, outcome, and remaining uncertainty, then requests closure with its own assessment of completion. Because later tasks may rely on this account, a separate *judge session* performs *verification* using the goal, four views at the stopping location, sampled earlier frames, the map, and execution records. Without access to tools, the navigation conversation, or the navigator's self-assessment, the judge returns an evidence-backed verdict of *complete*, *incomplete*, or *unknown*.

For *pre-stop verification*, the orchestrator compares the judge's independent verdict with the navigator's completion claim before issuing STOP. If the first checked request yields a definite verdict that contradicts the claim, the orchestrator returns evidence to the current navigation session, giving it a chance to continue searching. An unknown verdict does not block closure, and a repeated request is honored. *Post-stop certification* records the judge's final assessment without changing the task's score. It reuses the earlier verdict if the action-step count is unchanged and otherwise checks the final evidence (Appendix A.6).

The orchestrator saves the verdict, appends the ledger entry, and retains spatial memory for the next task's conversation, including when the task ends through budget exhaustion. Claims and verdicts remain separate because a completion check does not validate every spatial assertion in a handover.

3.5 CONSOLIDATING EXPERIENCE FOR THE NEXT RUN

To spare later sessions from inspecting every past task account, a separate *consolidation session* reads the journal and existing house notes at run end, then rewrites the index, house overview, room notes, and navigation skills. It adds a skill only when supported by the journal. Using file tools only, it integrates experience and corrections while distinguishing checked outcomes from navigator claims and preserving source records (Appendix A.4). On a later run in the same house, the navigation session reads these notes alongside saved maps and views to connect past experience to its current surroundings, including after a prescribed relocation between sequences. We evaluate this cross-run reuse in Section 4.4, whereas benchmark runs in Sections 4.2 and 4.3 start with empty memory.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETTING

We evaluate NavHarness on the validation-unseen splits of GOAT-Bench (Khanna et al., 2024) and IR2R-CE (Krantz et al., 2023). On GOAT-Bench, we evaluate all 2,669 subtasks in 360 episodes across 36 scenes. Our reasoning cores are Qwen3.8-27B (Qwen Team, 2026), GPT-4o (OpenAI, 2024), Opus 5 (Anthropic, 2026), and GPT-6 Astra (OpenAI, 2026), all provided with the same navigation and memory tools. Comparisons within each model use the same evaluation tasks and task budgets. We report individual-task success (s-SR) and the fraction of sequences in which every task succeeds (e-SR) (Niu et al., 2026), together with SPL for path efficiency (Anderson et al., 2018a) and IR2R-CE's t-nDTW for trajectory fidelity (Krantz et al., 2023; Ilharco et al., 2019). Appendix B provides the backend configurations and detailed evaluation settings. Memory carries from one subtask to the next within a GOAT-Bench episode, and between tasks within an IR2R-CE tour. Each new episode or tour starts with empty maps, task records, and house notes, without inheriting memory from the previous one.

4.2 COMPARISON WITH STATE-OF-THE-ART METHODS

Table 1: Comparison with SOTA methods on GOAT-Bench Val-Unseen. GT pose: simulator ground-truth pose. †: 278-subtask evaluation. Our scores: mean ± s.d. over three seeds.

Method	Base Model	Type	GT pose	s-SR	SPL	e-SR
MemoryExplorer† (Wang et al., 2026c)	Qwen2.5-VL-7B	trained	✓	46.4	28.0	–
SSMG-Nav (Niu et al., 2026)	Qwen-VL-Plus	zero-shot	✓	46.5	34.1	8.6
EvoMemNav (Ge et al., 2026)	Qwen3-VL-8B	zero-shot	✓	59.6	38.9	–
ReExplore† (Zhang et al., 2025a)	GPT-4o	zero-shot	✓	59.8	42.5	–
AstraNav-Memory (Ren et al., 2025)	Qwen2.5-VL-3B	trained	✓	62.7	56.9	–
STEGNav (Chen et al., 2026c)	GPT-5.4-mini	zero-shot	✓	66.3	39.7	–
GSMem (Lu et al., 2026)	GPT-4o	zero-shot	✓	67.2	46.9	–
3D-Mem† (Yang et al., 2025)	GPT-4o	zero-shot	✓	69.1	48.9	–
MetaNav (Li et al., 2026c)	GPT-4o	zero-shot	✓	71.4	51.8	–
HGR† (Chen et al., 2026a)	GPT-4o	zero-shot	✓	72.4	56.2	–
ObsGraph† (Lee et al., 2026)	GPT-4o	zero-shot	✓	72.7	51.5	–
HIMM† (Li et al., 2026a)	GPT-4o	zero-shot	✓	72.8	56.1	–
Ours (same-backbone comparisons)						
Independent sessions	GPT-4o	agentic	×	43.5 ± 0.6	30.2 ± 0.5	6.4 ± 0.6
NavHarness (subset)†				77.6 ± 1.0	56.4 ± 0.8	25.0 ± 2.8
NavHarness				78.3 ± 0.5	57.1 ± 0.4	26.9 ± 0.8
Independent sessions	Qwen3.8-27B	agentic	×	41.4 ± 0.6	27.5 ± 0.5	3.1 ± 0.5
NavHarness				71.7 ± 0.5	48.2 ± 0.4	14.4 ± 0.7
Independent sessions	Opus 5	agentic	×	58.9 ± 0.9	34.9 ± 0.8	5.8 ± 0.6
NavHarness				81.5 ± 0.7	55.0 ± 0.6	28.6 ± 0.8
Independent sessions	GPT-6 Astra	agentic	×	65.1 ± 0.6	40.1 ± 0.5	13.6 ± 0.7
NavHarness				$\mathbf{83.7 \pm 0.4}$	$\mathbf{62.3 \pm 0.3}$	$\mathbf{36.9 \pm 0.9}$

With each reasoning model held fixed, NavHarness improves GOAT-Bench s-SR by 30.3 points for Qwen3.8-27B, 34.8 for GPT-4o, 22.6 for Opus 5, and 18.6 for GPT-6 Astra over independent sessions (Table 1). These MIP-style controls (Zhou et al., 2026) use only the current task's context, so the gains

Table 2: IR2R-CE Val-Unseen. GT pose: simulator pose. Ours: three-seed mean ± s.d.

Method	Type	GT pose	s-SR	SPL	e-SR	t-nDTW
CMA (Krantz et al., 2023)	trained	×	19	18	–	38
MAP-CMA (Krantz et al., 2023)	trained	✓	35	32	–	47
ETPNav (Han et al., 2026)	trained	✓	28	27	–	41
HNR (Han et al., 2026)	trained	✓	30	28	–	44
OVER-NAV (Zhao et al., 2024)	hybrid	✓	35	33	–	50
SeqWalker (Han et al., 2026)	trained	✓	36	34	–	52
Ours (same-backbone comparisons)						
Independent sessions (Qwen3.8-27B)	agentic	×	29.0 ± 0.9	21.0 ± 1.1	0.0 ± 0.0	36.9 ± 1.2
NavHarness (Qwen3.8-27B)	agentic	×	45.1 ± 1.1	32.0 ± 0.9	0.0 ± 0.0	43.2 ± 1.2
Independent sessions (GPT-4o)	agentic	×	37.0 ± 1.3	29.0 ± 1.2	2.8 ± 2.8	43.1 ± 0.9
NavHarness (GPT-4o)	agentic	×	57.8 ± 1.2	45.2 ± 1.0	13.9 ± 2.8	52.9 ± 1.1
Independent sessions (Opus 5)	agentic	×	59.3 ± 1.1	32.8 ± 1.2	2.8 ± 0.0	45.1 ± 1.3
NavHarness (Opus 5)	agentic	×	78.2 ± 0.9	56.9 ± 1.3	16.7 ± 2.8	58.1 ± 1.0
Independent sessions (GPT-6 Astra)	agentic	×	68.4 ± 1.4	53.6 ± 0.8	9.3 ± 1.6	56.8 ± 1.2
NavHarness (GPT-6 Astra)	agentic	×	$\mathbf{85.9 \pm 0.8}$	$\mathbf{76.1 \pm 0.9}$	$\mathbf{27.8 \pm 2.8}$	$\mathbf{64.2 \pm 1.1}$

reflect the full harness, including mapping, recovery, and completion checks. Sections 4.3 and 4.4 isolate memory's contribution. Astra also improves SPL from 40.1 to 62.3, while its 17.5-point s-SR gain on IR2R-CE is close to its 18.6-point gain on GOAT-Bench.

With the same GPT-4o backbone and the 278-subtask protocol, NavHarness reaches 77.6 s-SR, exceeding HIMM (72.8) and 3D-Mem (69.1) without simulator poses (Li et al., 2026a; Yang et al., 2025). With Astra, NavHarness reaches 83.7% s-SR and 62.3 SPL on GOAT-Bench, exceeding the strongest prior scores in Table 1 by 10.9 and 5.4 points, respectively (Li et al., 2026a; Ren et al., 2025). On IR2R-CE (Table 2), it reaches 85.9% s-SR and 76.1 SPL, compared with SeqWalker's 36% and 34 (Han et al., 2026). The open 27B Qwen executor also reaches 71.7% s-SR on GOAT-Bench, 2.6 points above the published 3D-Mem score (Yang et al., 2025). NavHarness obtains these results without simulator poses, a global navmesh, auxiliary perception models, or navigation-specific learned policies.

For lifelong navigation, these improvements must carry through a whole sequence of goals. With Astra, GOAT-Bench e-SR rises from 13.6% in independent sessions to 36.9% with NavHarness, meaning that over a third of tours finish with every subtask completed. The strongest prior e-SR reported in Table 1 is 8.6% for SSMG-Nav (Niu et al., 2026). On IR2R-CE, Astra's e-SR rises from 9.3% to 27.8%.

4.3 What carries navigation across tasks and attempts?

We use Opus 5 on all 2,669 GOAT-Bench Val-Unseen subtasks with matched task budgets. Table 3 compares session policies (A), cross-task memory (B), recovery handovers (C), and completion checks (D). Recovery shares the original task's budget (Appendices B.6 and B.4).

A. Session policies. A single session continues the conversation across tasks, using backend compaction only when the context window fills. Yet it performs worse than independent sessions (54.9% versus 58.9%), which start afresh for each task with only the current conversation and no external memory. The longer conversation also retains outdated goals and observations, which may compete with information relevant to the current task. NavHarness gives each task a fresh conversation while keeping prior experience available through external memory. Adding pre-stop verification to independent sessions still leaves a 19.1-point gap to NavHarness. The remaining comparisons examine the memory and recovery mechanisms behind this gap.

B. Cross-task memory. All B controls retain within-task mapping, recovery, and verification. Clearing the map and markers preserves earlier task records, whereas hiding those records, including the ledger and handovers, preserves the map. No cross-task memory keeps these task-local mechanisms but removes both forms of carryover, isolating cross-task reuse with a 14.7-point s-SR loss. Hiding records costs 9.7 s-SR points and increases steps by 25%, while clearing spatial memory costs 12.6 points and increases steps by 32%. These interventions leave later tasks to reconstruct routes or repeat earlier searches, adding exploration that retained experience could avoid.

Table 3: Opus 5 comparisons on the full GOAT-Bench Val-Unseen split. Scores are mean ± s.d. over three seeds (%). Δs-SR denotes the paired change from full.

Configuration / intervention	s-SR	SPL	Δs-SR	95% CI
NavHarness (full reference)	81.5 ± 0.7	55.0 ± 0.6	–	–
A. Session policies				
Independent sessions	58.9 ± 0.9	34.9 ± 0.8	-22.6	$[-26.3, -19.0]$
Single session	54.9 ± 1.1	32.8 ± 0.9	-26.6	$[-30.5, -22.8]$
Independent sessions + pre-stop verification	62.4 ± 0.8	36.5 ± 0.7	-19.1	$[-22.4, -16.4]$
B. Cross-task memory				
Clear map after each task (keep records)	68.9 ± 0.8	39.8 ± 0.7	-12.6	$[-15.6, -9.7]$
Hide earlier task records (keep map)	71.8 ± 0.7	41.8 ± 0.6	-9.7	$[-12.5, -7.0]$
No cross-task memory	66.8 ± 0.9	37.4 ± 0.7	-14.7	$[-16.9, -12.4]$
C. Recovery handovers				
No recovery (same task budget)	71.1 ± 0.9	46.1 ± 0.8	-10.4	$[-13.2, -7.7]$
Recovery + empty handover	72.9 ± 0.8	44.7 ± 0.7	-8.6	$[-11.3, -6.0]$
Recovery + another task's handover	68.5 ± 1.0	44.8 ± 0.8	-13.0	$[-16.1, -10.0]$
Recovery + matched-length summary	73.2 ± 0.7	47.0 ± 0.6	-8.3	$[-11.0, -5.7]$
D. Completion checks				
No pre-stop verification	76.1 ± 0.8	47.0 ± 0.7	-5.4	$[-7.8, -3.1]$
No post-stop certification	79.0 ± 0.6	48.2 ± 0.6	-2.5	$[-4.5, -0.6]$
One stop view instead of four	78.6 ± 0.7	47.9 ± 0.6	-2.9	$[-5.0, -0.9]$

C. Recovery handovers. To separate restarting from passing on experience, we keep recovery but remove or replace its handover. An empty handover reaches 72.9% success, compared with 71.1% without recovery, while another task's note performs worse. Resetting the conversation removes its history, but without a useful handover the new attempt also loses the evidence needed to choose a different search. A matched-length ordinary summary still trails the structured handover by 8.3 points with unchanged triggers and budgets. The handover distinguishes places that were visited from those where the target was ruled out by evidence, and retains untried options. This gives the next attempt a basis for choosing where to search.

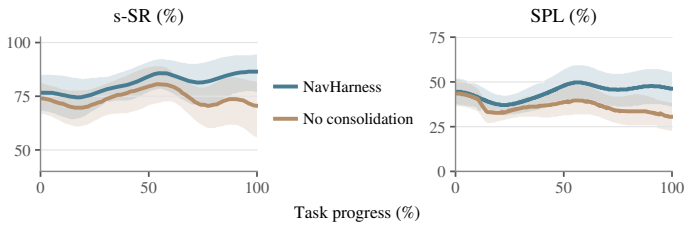
D. Completion checks. Unlike memory used throughout a search, completion checks intervene when the agent proposes to stop. Pre-stop verification can prolong the search, whereas post-stop certification only changes the record that later tasks inherit. This helps explain their smaller s-SR effects (5.4 and 2.5 points). Certification also reaches 90.7% record accuracy against 82.6% for accepting every claim, a benefit not captured by success rate alone (Appendix C.5).

4.4 Continuous deployment

To study memory reuse beyond the benchmark horizon, we concatenate ten GOAT-Bench tours per house across simulated days. At each tour boundary, the robot is placed at the next prescribed start, modeling resumption after a shutdown. It must recover its bearings using SLAM and stored house knowledge before reusing earlier experience. Detailed case studies are provided in Appendix D.2.

To test what consolidation adds beyond retained maps and task records, we compare NavHarness with *No consolidation* across all 36 houses using Opus 5. Both retain maps and task records across tasks and use the same recovery, verification, and task budgets. The control disables only run-end consolidation, so it does not build house notes for later runs. Consolidating experience into long-term memory raises pooled s-SR from 72.8% to 80.5% and SPL from 36.5 to 44.3, with paired gains of 7.7 and 7.8 points (Figure 3). Beyond these quantitative results, we examine the deployment records for qualitative insights into how the agent uses memory to interpret goals, guide its search, and reconsider earlier conclusions.

Interpreting goals through experience. In a repeated dresser task, the phrase "above the staircase handrail" sends the agent without consolidation upstairs, although the target is in the freezer room. NavHarness reads a house note connecting that same wording to the dresser and searches in the



Pooled performance

	s-SR (%)	SPL (%)
No consolidation	72.8	36.5
NavHarness	**80.5**	**44.3**
Gain (pp)	+7.7	+7.8
95% CI	[+6.3, +9.1]	[+6.5, +9.2]

Figure 3: Continuous deployment across 36 houses with Opus 5, varying only consolidation. The table reports pooled scores, paired gains, and 95% CIs.

correct room. The stored association lets the session identify the intended object even when the wording alone suggests a different search location. It reads this note before moving or querying the map, resolving which floor and object the instruction refers to before planning a route.

These associations need not be settled when first recorded. The house notes initially leave open whether two refrigerator descriptions refer to one appliance or two. After completing a later task, the session returns to this question, inspects both faces of the refrigerator, and records its answer for consolidation. When a subsequent task supplies a dark photograph, NavHarness recognizes the appliance, while the agent without consolidation searches for a second refrigerator in the basement. The session has gathered information beyond what its current goal requires, prompted by an unresolved question in memory. That answer then helps a future session interpret a new goal.

Reusing failed attempts and revising memory. Useful experience also comes from tasks that did not succeed. In the mirror search, an earlier attempt runs out of steps but records a wall left unchecked behind an open door. A later session initially follows an incomplete house note to the wrong mirror. After recovery, it compares the goal image with the note and searches the journal, where it finds that earlier suggestion. The failed attempt has preserved a concrete place to resume, even though the consolidated notes do not contain it. Keeping the detailed record lets the session check what was searched and what remained unchecked when the house summary proves insufficient. It can then choose its next search from the earlier observations rather than relying entirely on what consolidation retained.

Sometimes only part of a record needs revision. A stored note describes a corner with two plants but advises avoiding it. When a new goal photograph asks for one of those plants, the agent uses the description to locate the corner, finds a suitable viewpoint, and records a correction that consolidation incorporates into the note. It can keep the spatial knowledge while revising the advice attached to it. Similarly, after a restart changes the SLAM frame, landmark descriptions remain useful while coordinates are re-established. Across these cases, the reasoning session determines which parts of memory apply to the current task and supplies observations that later sessions can reuse.

Memory reliability remains a limitation. An incorrect completion accepted by the judge can propagate into house notes, and correcting one record does not necessarily update its copies or validate other spatial claims. Appendix E discusses these remaining risks.

5 Conclusion

NavHarness improves navigation across four reasoning models by letting multi-round sessions use maps, search records, and house knowledge without additional training. Fresh attempts benefit from structured search evidence that survives the previous conversation, allowing the robot to reconsider its plan without discarding what it has learned about the search. In extended deployments, this experience helps interpret ambiguous goals, investigate unresolved questions, and resume failed searches, while new observations lead to corrections that later sessions can use. These results support building lifelong navigation around successive reasoning sessions that inherit useful experience and continue to check and revise it as they act.

AI USE STATEMENT

Large language models are components of the studied system. GPT-4o, Qwen3.8-27B, Opus 5, and GPT-6 Astra serve as navigation reasoning cores. Model-based completion checks and memory consolidation are described in Section 3 and Appendix B. Separately, we used Claude Code and Codex to draft, revise, and format parts of the manuscript, create and refine scientific figures, and prepare analysis code. Generative AI also supported literature searches. The authors are responsible for verifying AI-assisted material against source publications and experiment records, and take responsibility for the final text, claims, code, figures, citations, and reported results.

REPRODUCIBILITY STATEMENT

We will release the NavHarness implementation, prompts, evaluation configurations, task lists, run records, and analysis scripts. The accompanying documentation will distinguish measured results from cost illustrations and manually coded case analyses. Since model sampling makes individual runs stochastic, the records will document the executions underlying the reported results.

ETHICS STATEMENT

This study evaluates navigation in simulated indoor environments rather than deployment around people. Its results do not establish safe physical operation, collision avoidance in dynamic scenes, or reliability under sensor faults. Model-based completion checks can produce false acceptance and false rejection, so a certified record is not an authoritative safety guarantee. The system retains indoor images and descriptions. A physical deployment would require consent, access controls, and retention policies appropriate to potentially private household data.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents, 2018a. URL https://arxiv.org/abs/1807.06757.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2018b. URL https://arxiv.org/abs/1711.07280.

Anthropic. Claude Opus 5 system card. https://www.anthropic.com/transparency, 2026.

Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial, and multimap SLAM. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. doi: 10.1109/TRO.2021.3075644.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. In *International Conference on 3D Vision (3DV)*, 2017.

Peixin Chen, Guoxi Zhang, Jianwei Ma, and Qing Li. Hypothesis graph refinement: Hypothesis-driven exploration with cascade error correction for embodied navigation, 2026a. URL https://arxiv.org/abs/2604.04108.

Yang Chen, Lirong Che, Zhenyu Huang, Wenbo Fu, Chuang Wang, Xu Cao, Daqi Liu, Yuzhe Yang, Jian Su, and Lan-Zhe Guo. HarnessVLN: Unifying training-free embodied navigation through an agent harness, 2026b. URL https://arxiv.org/abs/2609.15195.

Yang Chen, Zhenyu Huang, Wenbo Fu, Danyang Peng, Shi-Yu Tian, Kun-Yang Yu, and Lan-Zhe Guo. STEGNav: Spatio-temporal event graph reasoning for multimodal lifelong object navigation, 2026c. URL https://arxiv.org/abs/2608.28279.

Yanzhe Chen, Zechen Bai, Zhijun Cao, Wenzheng Zeng, Kevin Qinghong Lin, Yiqi Lin, Guoqiang Liang, Kevin Yuchen Ma, Qiming Huang, and Mike Zheng Shou. Show-Harness: Just a VLM agent can play robots, 2026d. URL https://arxiv.org/abs/2609.10522.

Yucheng Deng, Pingrui Lai, Xinhai Li, Chenjia Bai, Xiaoheng Deng, Chengnuo Sun, Xuelong Li, and Hua Yang. SpaceVLN: A zero-shot vision-and-language navigation agent with online spatial cognitive memory and reasoning, 2026. URL https://arxiv.org/abs/2606.08992.

Xin Ding, Liang Mi, Mingzhe Huang, Zixuan Wang, Chao Zhang, Zixu Hao, Fu Chen, Xiangyu Li, Yikai Zheng, Yaoyu Guo, Weijun Wang, Kun Li, Hao Wu, Yunxin Liu, and Ting Cao. Zetta ζ: An efficient closed-loop embodied harness for self-evolving physical intelligence, 2026. URL https://arxiv.org/abs/2608.16590.

Karim Elmaaroufi, Justin Svegliato, Sarunas Kalade, Graham Schelle, Sanjit A. Seshia, and Matei Zaharia. RHO: Your coding agent is secretly a roboticist, 2026. URL https://arxiv.org/abs/2606.16458.

Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CoWs on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Zuhao Ge, Xiaosong Jia, Chao Wu, Yuchen Zhou, Zuxuan Wu, and Yu-Gang Jiang. EvoMemNav: Efficient self-evolving fine-grained memory for zero-shot embodied navigation, 2026. URL https://arxiv.org/abs/2606.03509.

Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vision language models: Transforming spatial reasoning into question-answering, 2024. URL https://arxiv.org/abs/2411.05755.

Jianyuan Guo, Zhiwei Hao, Chengcheng Wang, Cheng Fan, Tingzhang Luo, Hongguang Li, Ying Gao, Hefei Mei, Jiankun Peng, Rongjian Xu, Minjing Dong, Han Wu, Mengyu Zheng, Kai Han, Shiqi Wang, Chang Xu, and Yunhe Wang. From question answering to task completion: A survey on agent system and harness design, 2026. URL https://arxiv.org/abs/2606.20683.

Zebin Han, Xudong Wang, Baichen Liu, Qi Lyu, Zhenduo Shang, Jiahua Dong, Lianqing Liu, and Zhi Han. SeqWalker: Sequential-horizon vision-and-language navigation with hierarchical planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026. URL https://arxiv.org/abs/2601.04699.

Xubin Hao, Hongjin Meng, Xin Yin, Jiawei Zhu, and Chenpeng Cao. Self-GC: Self-governing context for long-horizon LLM agents, 2026. URL https://arxiv.org/abs/2607.00692.

Jinbang Huang, Yuanzhao Hu, Zhiyuan Li, Ran Qi, Yixin Xiao, Zhanguang Zhang, Mark Coates, Tongtong Cao, and Yingxue Zhang. RoboHarness: Memory-driven orchestration of heterogeneous robot policies for long-horizon planning, 2026. URL https://arxiv.org/abs/2607.18060.

Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. General evaluation for instruction conditioned navigation using dynamic time warping, 2019. URL https://arxiv.org/abs/1907.05446.

MoniJesu James, Marcelino Julio Fernando, Miguel Altamirano Cabrera, and Dzmitry Tsetserukou. AnyGoal: Vision-language guided multi-agent exploration for training-free lifelong navigation, 2026. URL https://arxiv.org/abs/2606.13878.

Zihe Ji, Huangxuan Lin, and Yue Gao. DyNaVLM: Zero-shot vision-language navigation system with dynamic viewpoints and self-refining graph memory, 2025. URL https://arxiv.org/abs/2506.15096.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *International Conference on Machine Learning (ICML)*, 2026. URL https://arxiv.org/abs/2510.00615.

Mukul Khanna, Ram Ramrakhya, Gunjan Chhablani, Sriram Yenamandra, Theophile Gervet, Matthew Chang, Zsolt Kira, Devendra Singh Chaplot, Dhruv Batra, and Roozbeh Mottaghi. GOAT-Bench: A benchmark for multi-modal lifelong navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Khanna_GOAT-Bench_A_Benchmark_for_Multi-Modal_Lifelong_Navigation_CVPR_2024_paper.html.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, 2020.

Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14921–14930, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/html/Krantz_Iterative_Vision-and-Language_Navigation_CVPR_2023_paper.html.

Taekbeom Lee, Youngseok Jang, Jeonghwa Heo, Jeongjun Choi, and H. Jin Kim. ObsGraph: Hierarchical observation representation for embodied reasoning and exploration, 2026. URL https://arxiv.org/abs/2606.24068.

Zixing Lei, Gengze Zhou, Xiong-Hui Chen, Jiazhao Zhang, Yiyang Huang, Hang Yin, Haoqi Yuan, Qi Wu, Weixin Li, and Siheng Chen. Scaffolding foundation models into physical-world agents pushes the frontier of long-horizon navigation, 2026. URL https://arxiv.org/abs/2608.30396.

Ji Li, Bo Wang, Jing Xia, Mingyi Li, and Shiyan Hu. HIMM: Human-inspired long-term memory modeling for embodied exploration and question answering, 2026a. URL https://arxiv.org/abs/2602.15513.

Xiaochuan Li, Ryan Ming, Meng Chu, Shuai Shao, Rong Jin, and Chenyan Xiong. ACM: Agentic context management for long horizon tasks, 2026b. URL https://arxiv.org/abs/2607.23809.

Xueying Li, Feng Lyu, Hao Wu, Mingliu Liu, Jia-Nan Liu, and Guozi Liu. Stop wandering: Efficient vision-language navigation via metacognitive reasoning, 2026c. URL https://arxiv.org/abs/2604.02318.

Yijian Li, Changze Li, Hantian Shi, Jiaying Luo, Jiyuan Cai, Ming Yang, and Tong Qin. AgenticNav: Zero-shot vision-and-language navigation as a tool-calling harness, 2026d. URL https://arxiv.org/abs/2606.10577.

Yiren Lu, Yi Du, Disheng Liu, Yunlai Zhou, Chen Wang, and Yu Yin. GSMem: 3D gaussian splatting as persistent spatial memory for zero-shot embodied exploration and reasoning, 2026. URL https://arxiv.org/abs/2603.19137.

Qianyu Meng, Yanan Wang, Liyi Chen, Yihang Li, Wei Wu, Wenyuan Jiang, Qimeng Wang, Chengqiang Lu, Yan Gao, Yi Wu, and Yao Hu. Agent harness for large language model agents: A survey. Preprints.org, 2026. URL https://doi.org/10.20944/preprints202604.0428.v3. Version 3.

Haochen Niu, Lantao Zhang, Xingwu Ji, Rendong Ying, Peilin Liu, and Fei Wen. SSMG-Nav: Enhancing lifelong object navigation with semantic skeleton memory graph, 2026. URL https://arxiv.org/abs/2603.01813.

OpenAI. GPT-4o system card, 2024. URL https://openai.com/index/gpt-4o-system-card/.

OpenAI. GPT-6 Astra: A new generation of intelligence. https://openai.com/index/gpt-6-astra/, 2026.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems, 2023. URL https://arxiv.org/abs/2310.08560.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023. URL https://arxiv.org/abs/2304.03442.

Qwen Team. Qwen3.8-27B model card. https://huggingface.co/Qwen/Qwen3.8-27B, 2026.

Prithvi Rajasekaran. Harness design for long-running application development. Anthropic Engineering, 2026. URL https://www.anthropic.com/engineering/harness-design-long-running-apps.

Prithvi Rajasekaran, Ethan Dixon, Carly Ryan, and Jeremy Hadfield. Effective context engineering for AI agents. Anthropic Engineering, 2025. URL https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents.

Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X. Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *NeurIPS Datasets and Benchmarks Track*, 2021.

Botao Ren, Junjun Hu, Xinda Xue, Minghua Luo, Jintao Chen, Haochen Bai, Liangliang You, and Mu Xu. AstraNav-Memory: Contexts compression for long memory, 2025. URL https://arxiv.org/abs/2512.21627.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Andrew Semenov and Svyatoslav Dorofeev. Beyond compaction: Structured context eviction for long-horizon agents, 2026. URL https://arxiv.org/abs/2606.11213.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

Theodore R. Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L. Griffiths. Cognitive architectures for language agents. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2309.02427.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=pAMNKGwja6.

Peidong Wang, Zhiming Ma, Ying Chang, Xufang Luo, Yiqun Zhang, Zihan Wang, Xiaocui Yang, Shi Feng, Yuqing Yang, and Dongsheng Li. Self-evolving embodied agents via skill-harness evolution, 2026a. URL https://arxiv.org/abs/2608.11350.

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, 2026b. URL https://arxiv.org/abs/2608.11246.

Sen Wang, Bangwei Liu, Zhenkun Gao, Lizhuang Ma, Xuhong Wang, Yuan Xie, and Xin Tan. Explore with long-term memory: A benchmark and multimodal LLM-based reinforcement learning framework for embodied exploration, 2026c. URL https://arxiv.org/abs/2601.10744. CVPR 2026.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, Hoang H. Tran, Fuqiang Li, Ren Ma, Mingzhang Zheng, Bill Qian, Yanjun Shao, Niklas Muennighoff, Yizhe Zhang, Binyuan Hui, Junyang Lin, Robert Brennan, Hao Peng, Heng Ji, and Graham Neubig. OpenHands: An open platform for AI software developers as generalist agents. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2407.16741.

Xudong Wang, Jiahua Dong, Baichen Liu, Qi Lyu, Lianqing Liu, and Zhi Han. Lifelong embodied navigation learning. In *International Conference on Learning Representations (ICLR)*, 2026d. URL https://openreview.net/forum?id=PaYo96rjij.

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*, 2025. URL https://arxiv.org/abs/2502.12110.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3D-Mem: 3D scene memory for embodied exploration and reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Hang Yin, Yinan Liang, Jiazhao Zhang, Jiahang Liu, Minghan Li, Zhizheng Zhang, and He Wang. AllDayNav: Lifelong navigation via real-world reinforcement learning, 2026. URL https://arxiv.org/abs/2606.10927.

Justin Young. Effective harnesses for long-running agents. Anthropic Engineering, 2025. URL https://www.anthropic.com/engineering/effective-harnesses-for-long-running-agents.

Gengyuan Zhang, Mingcong Ding, Jingpei Wu, Ruotong Liao, and Volker Tresp. ReEXplore: Improving MLLMs for embodied exploration with contextualized retrospective experience replay, 2025a. URL https://arxiv.org/abs/2511.19033.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic context engineering: Evolving contexts for self-improving language models, 2025b. URL https://arxiv.org/abs/2510.04618.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, Zhengru Fang, Yi Nie, Changxu Wei, Yu Wang, Wenbo Ding, and Chao Yu. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents, 2026. URL https://arxiv.org/abs/2607.08448.

Ganlong Zhao, Guanbin Li, Weikai Chen, and Yizhou Yu. OVER-NAV: Elevating iterative vision-and-language navigation with open-vocabulary detection and structured representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2403.17334.

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 *Proceedings of Robotics: Science and Systems (RSS)*, 2026. URL https://superodometry.com/supermap.

Jian Zhou, Xunyi Zhao, Gengze Zhou, Zerui Li, Sihao Lin, Jiajun Liu, and Qi Wu. Embodied agents take control: Minimal-interface zero-shot agents rival industrial-scale policies in vision-and-language navigation, 2026. URL https://arxiv.org/abs/2607.26148.

Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, Siyuan Huang, and Qing Li. Move to understand a 3D scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Filippo Ziliotto, Tommaso Campari, Luciano Serafini, and Lamberto Ballan. TANGO: Training-free embodied AI agents for open-world tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Appendix

The appendix complements the main paper with implementation details and a closer examination of the experimental results. Appendix A presents the full algorithm and six-component harness specification. Appendix B describes the evaluation settings, statistical procedures, and supplementary controls, while Appendix C reports component results and diagnostics. Appendix D examines extended deployments and case studies of memory use, and Appendix E discusses limitations and future directions.

A Harness Specification

Algorithm 1 NavHarness over a sequence of K tasks

Require: reasoning model θ, environment E, goals $g_{1:K}$, recovery controller R, verifier V, task limits (B, b), hop limit H

mount spatial memory M, task workspace L, and house notes D_h
for $k = 1, \ldots, K$ **do**
  write goal file and form instruction $I(g_k, \rho_k)$ with memory-file references
  if the body was relocated since the previous task **then**
    attach surrounding views and a relocation instruction to I
  the new navigation session recognizes the place using retained memory
  end if
  open $C \leftarrow \mathrm{Open}(\sigma, I)$; set $a \leftarrow 1$, reset task counters and closing claim
  while task open and execution budget remains **do**
    advance C for one hop of at most H model turns, including its tool calls
    read execution state z and pending requests from the tool bridge
    if the environment has ended the task **then**
      break
    else if a closure request (c_k, ϵ_k) is pending **then**
      if handover file missing and not yet prompted **then**
        ask C to write the handover; **continue**
      end if
      perform pre-stop verification on the first checked request, obtaining v_k^{pre} from V
      if this is the first checked request and its definite verdict contradicts c_k **then**
        return counter-evidence to C; **continue**
      end if
      issue STOP and close the task
    else
      $u \leftarrow R(\hat{u}, z)$ using any recovery request and the fallback rule
      if $u \neq \mathrm{continue}$ **then**
        let C write $\lambda_{k,a}$ in a final, file-only hop
        apply scope u, retaining map, records, trajectory, and remaining budget
        replace C with a fresh session instructed to read $\lambda_{k,a}$; $a \leftarrow a + 1$
      else
        resume C unless persistent inactivity requires termination
      end if
    end if
  end while
  ensure the task has ended, recording budget or inactivity termination when applicable
  perform post-stop certification to obtain v_k, reusing v_k^{pre} when valid
  read h_k if written and collect place labels p_k; append $r_k = (g_k, c_k, v_k, h_k, p_k)$ to L
  write verdict and ledger files; update place memory and the run checkpoint
  release the acting conversation before opening the next task's session
end for
close the acting session
a separate file-only session consolidates D_h from house notes and task records

NavHarness comprises six components: the loop, the tool registry, the context policy, the memory, the hooks that fire at boundaries, and the verification of a claimed completion (Figure 4). This is the decomposition used by Meng et al. (2026). Guo et al. (2026) also distinguishes observation, action, and governance interfaces. We use the six component names to organize the implementation.

NavHarness specifies the state retained at navigation boundaries and the handover between attempts, as detailed below.



Figure 4: The six components of NavHarness and their responsibilities.

A.1 THE LOOP

A coding-agent session is the reasoning core of NavHarness. It uses multi-round multimodal dialogue to interpret observations, consult memory, and invoke navigation and file tools. The outer orchestrator manages the sequence of tasks and decides when that session should continue, close, or hand its experience to a fresh attempt. Algorithm 1 describes this lifecycle. Tool execution remains inside the coding session, so the orchestrator need not implement another loop for model inference and tool execution.

For a deployment in house h, we write the state at the closure of task k as

$$X_k = (C_k,\, M_k,\, L_k,\, D_h), \qquad M_k = \big(\{G^{(f)}\}_f,\, \mathcal{P}_k,\, \Gamma_k\big), \qquad L_k = (r_1, \ldots, r_k), \tag{1}$$

where C_k is the active session context, $G^{(f)}$ the occupancy grid of floor f, $\mathcal{P}_k$ the named places and their views, Γ_k the place graph, L_k the task records, and D_h the persistent house notes. Within an attempt, ordinary interactions extend the conversation as

$$C_{k,a,t+1} = C_{k,a,t} \,\|\, \big(y_t,\, \mathcal{A}_t,\, o_{t+1}\big), \qquad (y_t, \mathcal{A}_t) = \pi_\theta(C_{k,a,t}, \mathcal{T}), \tag{2}$$

with reply y_t, tool calls $\mathcal{A}_t$, and returned observations o_{t+1}. Here a indexes attempts and t indexes interactions. NavHarness does not rewrite the active conversation, and it opens a fresh session for every task, so no transcript persists across a task boundary. The single-session comparison is the only configuration that keeps one conversation across tasks.

Before task k closes, only L_{k-1} is available as prior task history, since r_k is appended at closure. The opening in Equation 7 therefore refers to L_{k-1}.

The orchestrator advances the session in *hops*. Each hop allows up to H model turns with their tool calls and returns the resulting execution state. Several hops may belong to the same conversation. On return, an environment termination takes priority, followed by a pending closure request and then recovery. Otherwise the orchestrator resumes the session with a continuation instruction. Closure and recovery tools record requests and ask the agent to yield, allowing these transitions to be handled between hops. The task retains its turn and physical-step limits (B, b) across attempts. Budget exhaustion or persistent inactivity also ends a task, even without a closing claim. The implementation defaults to $H = 25$ model turns per hop and an earliest requested recovery at 12 attempt turns. Three consecutive hops without a tool call end an inactive task. Three hops with no model turn instead raise a provider-unavailable error rather than recording navigation failure. Task budgets and enabled fallback settings are given in Appendix B.

A session adapter exposes opening, advancing, and closing a conversation to the orchestrator. Its tool bridge connects the coding loop to the environment and the memory workspace. This separation lets a different reasoning model use the same memory and boundary rules, provided its backend supports the session and tool interfaces. Acting, judging, and consolidation have separate conversations and tool permissions (Appendices A.3 and A.6).

A.2 TOOL REGISTRY

The model acts only through the calls in Table 4. Movement, mapping, and route queries are provided by the navigator. The harness adds file access and requests for closure and recovery. The localisation tool that the spatial calls draw on is described after the table.

Table 4: The tool registry. Only step consumes the task's physical budget, while free calls cost no steps.

Call	Arguments	Returns or effect
observe	none	the forward RGB view and remaining step budget
step	a list of actions	executes them and reports motion, collision, and budget status
get_goal	none	the current goal, as text or as the goal image
get_map	floor, zoom	the top-down occupancy map with the markers on it
mark	a place name	drops or moves a named marker at the current position
preview_path	a map point	the length of a route over known free space; nothing moves
list_files	a directory	lists the available memory files
read_file	a file path	reads a task record or house note
write_file	a file path, text	writes a file assigned to this session
request_recovery	scope, reason, confidence	proposes a recovery; the harness decides
close_task	status, evidence	captures stop views and requests checked closure

Sensing and motion. observe returns the forward camera image. step takes a list of discrete actions, forward 0.25 m, turn left 15°, or turn right 15°, and executes them in order. Every action costs one step of the task's physical budget, and the session also has a model-turn limit. The model is told that turning in place is a cheap way to look around and that it should alternate looking and moving.

A step call accepts up to twelve actions $\alpha = (a_1, \ldots, a_n)$ over the action set $\mathcal{A} = \{\text{forward } 0.25\,\text{m},\ \text{left } 15^\circ,\ \text{right } 15^\circ\}$, consumes one step for each executed action, and is executed as up to $5n$ sub-motions of 0.05 m or 3°, each of which feeds one frame to the localisation tool. Execution ends early when the environment terminates the task. Every other agent-facing call costs no steps.

Action discretization and collisions. The simulator executes each sub-motion separately, and the macro-action ends early if a collision is detected. Sliding is enabled in the simulator's embodiment configuration. This refinement supplies the intermediate observations required by SLAM. Our component and session comparisons share this motion interface. Benchmark goals and scoring are retained.

Goal. get_goal re-reads the current goal. When the goal was given as a photograph, the call returns the photograph, so an image goal can be consulted again after other images have entered the context.

Spatial memory. get_map returns the occupancy map that the navigator builds as the body moves, with its named markers. The map persists across tasks, allowing later searches to reuse explored space and earlier landmarks. mark drops a marker named for the place, not for the object sought, and the harness attaches to it the views taken at that point, bucketed by heading into at most four, and the navigator's pose estimate. Re-marking an existing name moves the marker. preview_path draws a route over the map's known free space to a point and reports its length without moving the body. All three are free.

A mark with name a at time t replaces any place of that name,

$$\mathcal{P} \leftarrow \big(\mathcal{P} \setminus \{z_i : a_i = a\}\big) \cup \big\{\big(a,\ m_t,\ B_t\big)\big\}, \qquad |B_t| \le 4, \tag{3}$$

where m_t is the marker at the current estimated pose and B_t the views taken there, bucketed by heading. get_map returns $\mathrm{Render}\big(G^{(f)}, \mathcal{P}^{(f)}\big)$ for the requested floor and zoom, and preview_path evaluates Equation 6 on $G^{(f)}$.

Localisation. Behind observe, get_map, mark, and preview_path is one estimator, and we treat it as a tool like any other. In the reported runs it is ORB-SLAM3 (Campos et al., 2021) in RGB-D mode, run as a separate process that the navigator feeds and queries. Nothing in the harness depends on that choice. The tool consumes the colour and depth frame of each sub-motion and returns a pose estimate and a status,

$$(\hat{\mathbf{T}}_t,\ s_t) = \mathrm{SLAM}\big(I_t, D_t \,\big|\, \hat{\mathbf{T}}_{<t}\big), \qquad \hat{\mathbf{T}}_t \in SE(3), \quad s_t \in \{\mathrm{ok}, \mathrm{lost}\}, \tag{4}$$

with the world frame anchored at the first frame seen in the house. The planar pose $(\hat{x}_t, \hat{z}_t, \hat{\psi}_t)$ read off $\hat{\mathbf{T}}_t$ is the only position the navigator has. It is what a marker stores, what the map is drawn in, and what a route is planned from. The simulator's pose is never read for any of these.

Each coarse action, forward 0.25 m or a turn of 15°, is executed as five sub-motions of 0.05 m or 3°, and every sub-motion yields one tracked frame at a synthetic 10 Hz. The split exists for the tracker, not for the body: a single frame per 15° turn leaves too few feature matches and loses tracking on a large share of frames, while 3° per frame keeps inter-frame matches available during navigation. For tracking loss within an ongoing run, when s_t is lost no pose is returned and the map is not written. Tracking status is exposed through motion and map responses and the bridge's execution state. The tracker opens a fresh map at its next successful initialisation and the navigator re-anchors it to the last trusted pose, $\hat{\mathbf{T}} \leftarrow \mathbf{A}\,\tilde{\mathbf{T}}$ with $\mathbf{A} = \hat{\mathbf{T}}_{\text{last}}\,\tilde{\mathbf{T}}_{\text{first}}^{-1}$. After a controlled relocation the body turns once in place while feeding the tracker to seek relocalisation against the retained tracker map. Loading saved occupancy maps and markers after a tracker restart does not itself recover their coordinate alignment.

Mapping and routes. The occupancy map is built from depth and the estimated pose rather than from the tracker's own point cloud, so it is the same computation for any estimator that returns a pose. A depth pixel u with $D_t(u) \in [0.3, 4.5]$ m is back-projected and placed in the world, and its height over the floor under the camera is read off,

$$\mathbf{p} = \hat{\mathbf{T}}_t\,\pi^{-1}\big(u, D_t(u)\big), \qquad h(\mathbf{p}) = h_c - \big(p_y - y_{\mathrm{ref}}\big), \tag{5}$$

where the mapper uses a downward-positive y axis. Here y_{ref} is the camera's reference world-y coordinate for the floor, and h_c is its height above that floor. The floor plane is therefore at $y_{\mathrm{ref}} + h_c$, where the formula gives zero height. A point is an obstacle when $0.15 < h < 1.4$ m and floor when $-0.5 < h \le 0.15$ m. Points fall into a grid of $\delta = 0.10$ m cells, $c(\mathbf{p}) = \lfloor (p_x, p_z)/\delta \rfloor$, with three states, unknown, free, and obstacle: an obstacle endpoint marks its cell, the ray from the body to it carves free cells, and an obstacle is never overwritten by free. The grid is kept per floor, and a new layer opens when the estimated height settles more than 1 m from every registered floor. get_map renders the current layer with the markers on it. The grid is written from the online pose, so a later loop-closure correction inside the tracker is not propagated into cells already written, which is one of the limitations in Appendix E.

preview_path answers a route query on the same grid. For a target cell $c^\star$, snapped to the nearest free cell if it lands on unknown or obstacle, it runs A* over the free cells with eight-connected moves and reports

$$L(c^\star) = \min_{\pi:\, c_0 \to c^\star} \sum_i \delta\, \|c_{i+1} - c_i\|_2, \tag{6}$$

or *unreachable* when no such path exists. Nothing moves. A marker is a name paired with the planar pose at the moment of the call, $m_k = (\mathrm{name}_k, (\hat{x}, \hat{z}, \hat{\psi})_{t_k})$, together with the views banked there.

Substituting the estimator. The localisation estimator supplies pose and tracking status. A separate mapping adapter combines these with depth to obtain the world points used by the occupancy map. Another visual or visual-inertial estimator can use the same adapter if it supplies compatible poses. A planar lidar would require a different range-projection and obstacle update, while preserving the navigation tools and orchestrator interface. Thus changing sensors can require tool-side implementation changes without changing the session and memory lifecycle. The reported runs use ORB-SLAM3.

Closing a task. The agent writes its handover before calling close_task(c_k, ϵ_k), where $c_k \in \{\mathrm{complete}, \mathrm{incomplete}\}$ and ϵ_k is a short account of the stopping evidence. The tool captures surrounding views, records the request, and asks the session to yield. It does not issue STOP itself. At the next hop boundary, the orchestrator prompts once if the handover file is absent, then applies

the pre-stop verification of Appendix A.6. An admitted request becomes a physical STOP. Tasks can also end through execution limits, in which case the ledger records the termination without inventing a completion claim.

Recovery. request_recovery lets the model say that part of what it relies on is no longer trustworthy, and how much of it, by naming the smallest scope that fits (Appendix A.5). The call costs no steps and does not end the task. The harness decides what happens: it can refuse, or act at a larger scope than asked.

The proposal is a triple $(\hat{u}, \text{reason}, c)$: a scope $\hat{u} \in \{\text{attempt}, \text{place}, \text{room}, \text{floor}, \text{house}\}$, a textual reason, and a confidence $c \in [0, 1]$. The model never proposes *continue*, which is the harness's answer as decided by Equation 14.

File access. The session reaches textual memory through list_files, read_file, and write_file. Paths are resolved within the run workspace or its mounted house directory. An acting session can read earlier records but can write only its assigned handover and recovery note. The orchestrator owns the ledger and verdict files, while the consolidation session owns the house notes. Coordinate-like expressions are scrubbed from session-written prose so that a later map re-anchoring does not leave stale metric coordinates in text. A read returns at most 20,000 characters and a write stores at most 6,000, with truncation reported by the tool. These are file-access limits, separate from the model's context window. The current read tool has no offset parameter: an overlong file returns only its prefix. Detailed task files can be located through directory listing and read individually, but a single ledger read does not guarantee access to an arbitrarily long history.

A.3 Context management

One session per attempt. The task-boundary policy opens a fresh coding session with the same standing system brief σ. Its first user instruction gives the goal, names the files to read, and identifies the handover file it may write. When available, required reading comprises the previous task's handover, the run ledger, and the existing house notes. Their contents enter the conversation through the session's own file reads, rather than being pasted into the system message.

Writing ρ_k for these file references, the opening is

$$C_{k,1,0} = \text{Open}\big(\sigma, I(g_k, \rho_k)\big), \qquad \rho_k = \text{refs}\big(h_{k-1}, L_{k-1}, D_h\big), \tag{7}$$

where nonexistent files are omitted. An image goal remains accessible through get_goal, and the map through the spatial tools. The session is instructed to read the listed records before moving and may consult additional files as the task develops. It then continues across hops without harness-side editing of the transcript.

Resets at task and attempt boundaries. At a task boundary, the orchestrator records the closing status and makes the actor-written handover available to the next session. At an approved recovery, the current session first writes a recovery note, then a new session resumes the same task from that file. Neither transition carries the old conversation into the new one. Map state, task records, and house notes remain outside the session and can be read again. Recovery preserves the physical trajectory and remaining task budget.

Division of context management. NavHarness controls when a conversation ends, while the coding backend controls how a running conversation is maintained. It introduces no image-eviction window or additional transcript-summarization policy. Keeping these responsibilities separate allows the reasoning core to be replaced without changing the memory interface. Unchanged prefixes also permit prompt-cache reuse where the provider supports it. The navigation-turn budget applies to the acting session. Judging and run-end consolidation use separate model calls.

Session policies and observed context use. The independent-session baseline opens a fresh conversation for each task and relies only on that context, without external working or long-term memory. Single session keeps using the same conversation after each task, accumulating observations and tool responses until the backend's context window fills. The backend then compacts that conversation so it can continue. Task boundaries do not trigger compaction. NavHarness instead

opens fresh conversations at task boundaries while retaining external memory, with mean peak prompts of approximately 20k tokens. Table 3 compares these complete policies, while its memory interventions hold the other harness mechanisms fixed.

Success over successive tasks. Figure 5 compares the fitted success trends of NavHarness and the single-session policy over positions 1 to 10. The curves summarize performance as tasks progress within this range. The paired system comparison and its uncertainty are reported in Table 3. Later positions contain fewer eligible scenes because episode lengths differ. Those aggregate intervals are not confidence bands for the curves. Table 5 separately reports measured position-level success from the NavHarness GOAT-Bench Val-Unseen task-position audit. These are single-run diagnostic observations, not averages over the three-seed component study.

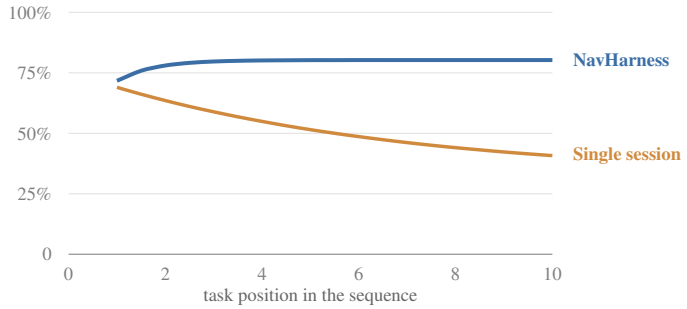


Figure 5: Success over successive tasks. Fitted trends for NavHarness and single session over positions 1 to 10.

Table 5: Observed task-position success in the NavHarness Val-Unseen diagnostic run.

Position	1	2	3	4	5	6	7	8	9	10
s-SR (%)	72.2	72.2	83.3	83.3	77.8	82.1	73.1	85.7	81.2	85.7

Context footprint and input-cost scaling. Figure 6 shows the first 20 tasks of a longer deployment. In the schematic context panel, single session reaches a roughly 1M-token window near task 20, when the backend compacts the conversation and continues. This boundary does not end the deployment. NavHarness instead opens task-level sessions with approximately 20k-token active contexts. Gray denotes single session in both panels.

The cost panel distinguishes a retained session from *stateless context rebuilding*, which reconstructs the prompt on every interaction. The orange curve represents the cache-miss case of the latter, not the single-session baseline or the independent-session baseline. Retaining a stable prefix permits cache reuse, whereas rebuilding a prompt can invalidate that prefix. Actual input cost therefore depends on both uncached and cached input volumes, $C_K = \sum_{k=1}^{K}(p_u U_k + p_c V_k)$, where U_k and V_k are the respective token counts for task k.

The cost comparison uses an assumed input price of \$5 per million tokens. With $2.4 \times 10^6 k$ input tokens at task k, stateless rebuilding without cache hits costs approximately \$2,520 over 20 tasks. Using 0.975M input tokens per task for NavHarness gives approximately \$98 at the same price. For retained sessions, we use a linear reference, $C_K = 528K/20$ dollars, reaching \$528 at task 20. These calculations cover navigation input tokens only, excluding output charges and auxiliary calls, and stop at task 20. Benchmark runtime is reported in Appendix B.3.

A.4 Memory

Memory is organized by lifetime and access, rather than by disjoint storage locations. The coding session maintains the current conversation. Working memory contains the sensor-built map, named

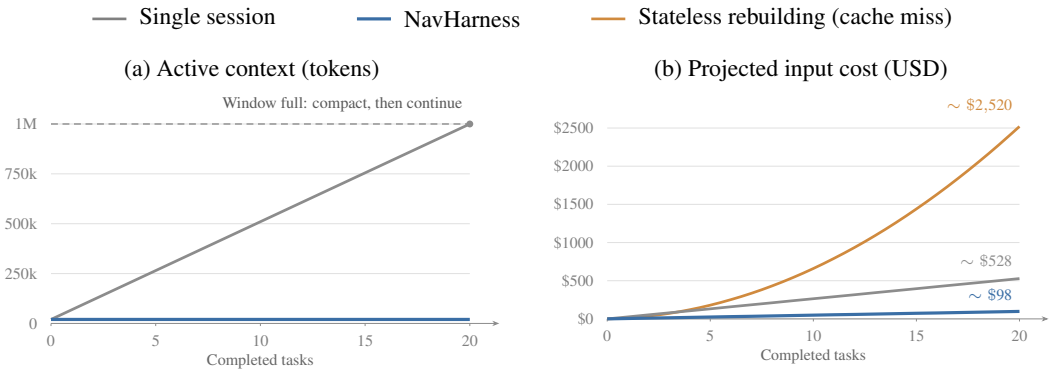


Figure 6: Context and input-cost scaling over the first 20 tasks of a longer deployment. Orange shows stateless context rebuilding without cache hits.

places, and task records shared across successive tasks. When retained for later runs, maps and place records also form part of long-term house memory, alongside textual notes and experience archives. The same map can thus serve both roles without being copied into a textual summary.

The ledger and task handovers. We use *handover* for experience passed across a session boundary. A task handover records a task at closure, whereas a recovery note passes search evidence to a new attempt at the same goal (Appendix A.5). These are separate files, as listed in Table 6. Before requesting closure, the acting session writes h_k, a brief account of the goal, visited rooms and landmarks, outcome evidence, and unsuccessful searches. The prompt requests fewer than 120 words. The orchestrator reads this file at the task boundary and combines it with the claim, checking verdict, and named places,

$$r_k = \big(g_k,\ c_k,\ v_k,\ h_k,\ p_k\big), \qquad \text{status}(r_k) = \begin{cases} \text{CERTIFIED} & v_k = \text{complete}, \\ \text{NOT DONE} & v_k = \text{incomplete}, \\ \text{UNRESOLVED} & v_k = \text{unknown}, \\ \text{CLAIMED} & \text{no certification ran.} \end{cases} \tag{8}$$

The full run record also retains the claim and verdict evidence, searched regions, step count, and termination reason. Environment scores are stored separately for evaluation and are not rendered in agent-facing memory. A certified completion concerns the goal predicate, so the actor's route descriptions and spatial interpretations remain its own account. CLAIMED denotes the implementation's unchecked status, not a claim of success. When a task ends without a model closing claim, the record uses stop and retains the termination reason. This remains distinguishable from an explicit complete claim even when certification is disabled.

The ledger file is a compact index with one line per closed task, containing its status, a shortened goal, and up to three marker names. Detailed handovers and verdicts remain separate files. A later session reads the ledger to orient its search and can open the relevant task directory for details. This keeps the account of an individual task addressable without copying every account into a new conversation. The complete task record r_k includes the handover and verdict, while the ledger file provides its short index entry. The ledger index itself grows with the number of tasks.

The house directory is mounted from persistent storage, whereas the run directory holds the current sequence. Repeated recoveries receive distinct note filenames. The permission checks of Appendix A.2 prevent an acting session from overwriting the ledger, judge's verdict, or consolidated house notes.

Places. A place is a marker, at most four views taken at it and bucketed by heading, and the navigator's estimate of where it was taken. The views are observations, the estimate is an estimate, and the name is a claim; the three are kept apart. During spatial recovery, current views can be compared with these banked observations to identify a familiar place. Formally a named place is

$$z_i = \Big(a_i, m_i, \{(I_j, \hat{\xi}_j)\}_{j \in B_i}\Big), \qquad |B_i| \le 4. \tag{9}$$

Table 6: Textual memory in the workspace. Each file has a designated writer.

Path	Writer	Content
run/ledger.md	Orchestrator	Task-status index
run/tasks/k/goal.md	Orchestrator	Goal and modality
run/tasks/k/handover.md	Acting session	Task experience
run/tasks/k/lambda.md	Acting session	Recovery handover
run/tasks/k/verdict.json	Orchestrator	Judge's assessment
house/	Consolidation session	House notes: index, overview, room notes, skills

with a_i the label, m_i the marker, and B_i its banked views with their SLAM pose estimates $\hat{\xi}_j$.

The places form a graph,

$$\begin{aligned} \Gamma &= \big(V_{\text{place}},\ E_{\text{conn}}\big), \\ E_{\text{conn}} &= \big\{(z_i, z_j) :\ z_i, z_j \text{ marked consecutively},\ f(z_i) \neq f(z_j)\big\}, \end{aligned} \tag{10}$$

where each place record carries a room label and floor association $f(z)$. The room label defaults to the marker's name. These are attributes of places, rather than a separate, fully connected room-and-floor graph. Room connections described in house notes remain textual knowledge. The graph is updated from task-boundary marker records and the reported map layer. An edge between consecutively registered places on different layers is a candidate connector for planning, rather than a directly observed staircase. Graph addresses are labels and relations, with metric geometry retained in the spatial map.

The house archive. Long-term memory is the persistent knowledge maintained for a house, not a single file. Its textual part, the house notes, comprises four files containing an index, a house overview, room notes, and navigation skills, denoted together by D_h in the equations. All four are written only by the consolidation session. Saved maps, place records, and experience archives retain their own representations.

Persistent records need not wait for consolidation. At task closure, the orchestrator updates the enabled place and experience stores and saves spatial state when checkpointing is enabled. Consolidation instead summarizes what happened over a sequence of tasks. It reads the existing house notes, ledger, and handovers to retain useful routes, failed searches, corrections, and unresolved questions without requiring later sessions to reread every account. In the current implementation this separate session runs at run closure, with the house-note update

$$D_h^{+} = \mathcal{C}_\theta(D_h, L_K), \tag{11}$$

where L_K includes the task accounts accessible through their files. The prompt asks the model to retain the distinction between certified outcomes and the robot's own interpretations, and to add a navigation skill only when supported by the journal. The workspace enforces file-write permissions, while these content requirements are instructions to the model.

Consolidation has file tools but no navigation tools, and its write permission covers only the house notes. It can summarize descriptions of rooms and landmarks, but does not rebuild or modify map geometry. The house notes appear in a subsequent run's required reading, while detailed records remain separately accessible. The summarization interval is a scheduling choice: the same operation could be applied after a task, a group of tasks, or a longer deployment. Run closure is the invocation point used here, not a requirement that all long-term memory be written at once. Storing or summarizing a claim does not certify it.

A.5 Hooks

Hooks are the orchestrator's responses to boundary events. A session can request closure or recovery through a tool, but the orchestrator decides and carries out the transition after the hop returns. Four events connect the reasoning session to persistent memory.

Task boundary. After closure, the orchestrator obtains the final checking verdict, reads the handover already written by the actor, and appends the task record. It writes the verdict file and ledger index, updates the enabled named-place and experience stores, and saves spatial state when checkpointing is enabled. The next task's instruction refers to the newly available records. With the task-boundary session policy, the transition is

$$(C,\, M,\, L,\, D_h) \;\mapsto\; \big(\varnothing,\; M^{+},\; L \,\|\, r_k,\; D_h\big), \tag{12}$$

where M^{+} includes the updated place graph. The house notes D_h are unchanged at this boundary even when other persistent records are updated. A task ended by its execution limit follows the same recording path, even if it has no handover or completion claim.

Attempt boundary. Recovery begins with a model request or an enabled duration fallback. The controller checks whether a new attempt is warranted, then selects how much of the current interpretation to reconsider. Every scope keeps the map, markers, ledger, and archive. Broader scopes add a reassessment of location or route to the conversation reset (Table 7), so restarting does not erase the observations that the robot has already collected.

The controller aims to choose the smallest scope consistent with the observed failure,

$$u_t^{*} = \min_{u \in \mathcal{U}} \{u : I(u) \text{ covers the beliefs implicated by the observed failure}\}, \tag{13}$$

where $I(u)$ denotes the beliefs to reconsider. The rules implement this objective using measurable execution signals, rather than an oracle for the cause of failure. At each hop they read

$$z = (t_a,\, \Delta A,\, \Delta E,\, \text{lost},\, \text{mismatch},\, \text{first},\, u_{\text{last}},\, n_r).$$

Here t_a counts turns in the attempt, ΔA is newly explored area, and $\Delta E = \lfloor \Delta n_{\text{frames}}/4 \rfloor + \Delta n_{\text{markers}}$ is an observation-count proxy since the previous checkpoint. The remaining signals indicate tracking loss, a reported floor mismatch, a first visit, the previous recovery scope, and the number of recoveries already used. Progress is defined as $\text{prog}(z) = [\Delta A \geq 1.5\,\text{m}^2] \vee [\Delta E \geq 1]$. New frame counts need not imply new scene content, so this test is used as a practical guard against premature recovery.

Let f indicate that an enabled fallback threshold T_f has been reached, and let N be the task's recovery cap. Given a proposal $\hat{u}$, the controller applies the following rules in order. A *continue* decision returns immediately.

$$\begin{aligned}
&u = \text{continue} && \text{if no proposal and } \neg f, \text{ or } n_r \geq N,\\
&u = \text{continue} && \text{if proposed, } \neg f, \text{ and } t_a < t_{\min},\\
&u = \text{continue} && \text{if proposed, } \neg f,\ \text{prog}(z) \wedge \neg\text{lost} \wedge \neg\text{mismatch},\\
&u = \hat{u} \text{ (proposed) or attempt (fallback)}, &&\\
&u \leftarrow \max(u, \text{place}) && \text{if lost},\\
&u \leftarrow \max(u, \text{floor}) && \text{if mismatch},\\
&u \leftarrow \text{room} && \text{if } u = \text{house and first},\\
&u \leftarrow u_{\text{last}}^{+} && \text{if escalation is enabled, } n_r > 0,\ u \leq u_{\text{last}},\\
&u \leftarrow \min(u, u_{\max}). &&
\end{aligned} \tag{14}$$

The configuration supplies T_f, $t_{\min}$, N, and the maximum scope $u_{\max}$. The notation u^{+} denotes one rung above u. Proposal confidence is recorded but is not a threshold in these rules. When $N = 1$, repeated-recovery escalation is inactive. Floor and house scopes use the same retained map and place-graph brief. Their distinction is the breadth of reassessment requested in the recovery prompt, not a different physical reset.

Before the session is replaced, it receives one final hop with file tools only. It writes a JSON handover at its assigned recovery-note path,

$$\lambda_{k,a} = \text{WriteNote}_{\theta}(C_{k,a}) = \big(S,\; R,\; a^{\star},\; U,\; \widetilde{U},\; \hat{u}\big), \tag{15}$$

covering searched places, evidence-supported exclusions, the last reliable anchor, untried options, uncertainties, and a suggested scope. The prompt explicitly separates *searched* from *ruled out*

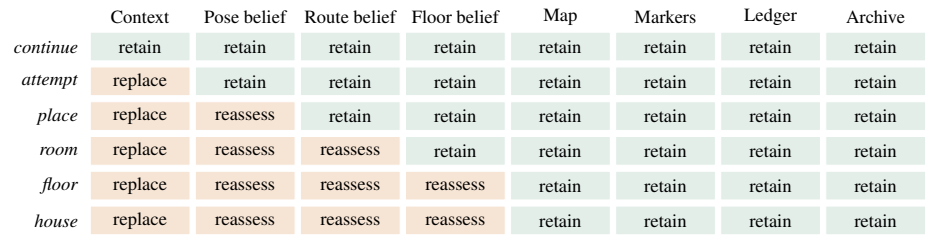


Figure 7: **Recovery scopes and retained state.** Broader recovery reconsiders additional beliefs while preserving accumulated observations and task records.

Table 7: Recovery scopes. All retain the map, markers, ledger, and archive.

Scope	State reconsidered
continue	No reset
attempt	Active conversation
place	Conversation and current location
room	Conversation, location, and route
floor	Conversation, location, route, and floor association
house	The same beliefs as floor, with house-level reassessment

and asks the agent to name the observation supporting each exclusion. The session that made the attempt therefore selects what its successor should know. These are the actor's evidence-backed interpretations, not independently verified exclusions. The recovery prompt discourages repeating them, so a mistaken exclusion can also be carried into the next attempt.

The orchestrator applies the selected scope and opens a new conversation,

$$C_{k,a+1,0} = \mathrm{Open}\big(\sigma, I_{\mathrm{recover}}(g_k, \mathrm{ref}(\lambda_{k,a}))\big). \tag{16}$$

The recovery instruction names the scope, any location brief, and the note to read first. The map and other files remain accessible through their usual tools, while the robot retains the same goal, position, and remaining task budget. Reconsidering a pose or route belief does not delete the stored map or reset the physical trajectory.

For scopes above *attempt*, the harness collects four surrounding views. When stored place photographs are available, a separate wake-up session compares them with current views and returns a location brief. Floor- and house-level recovery also provide the named-place graph. These observations help the new attempt establish its location, while metric tracking remains the localisation tool's responsibility.

Relocation. A relocation between tasks differs from an in-task recovery. The environment has already moved the body to the next starting point. The orchestrator retains the spatial state and files, collects four surrounding views, and attaches them to the next task's opening,

$$C_{k,1,0} = \mathrm{Open}\big(\sigma, I(g_k, \rho_k) \,\|\, I_{\mathrm{relocated}} \,\|\, F_k^{\circlearrowleft}\big). \tag{17}$$

This reasoning session establishes which room it is in before planning from the retained map. No separate room-identification session is required at this boundary. Visual recognition and the tracker's metric relocalisation remain distinct, since recognizing a room does not by itself determine the robot's pose.

Consolidation. At run closure, the acting session ends and a file-only consolidation session updates the house notes by Equation 11. Its reading list contains the ledger, task handovers, and previous house notes. Its write permission covers only the four house-note files, which become available to subsequent runs.

A.6 VERIFICATION

Verification is the completion-checking mechanism. It serves two roles at closure (Figure 8). Pre-stop verification can return evidence to the running actor. Post-stop certification determines the status stored in memory. Both use the same evidence interface and, by default, the same model as the actor, but neither receives the actor's conversation or closing self-assessment.

Evidence and verdict. Calling close_task captures four instrument-selected views at the proposed stopping point before either check runs. The judge receives the goal, these views, a sample of earlier frames, the map, and a mechanical record containing task position, steps used, step budget, frame count, and marker names. The frame selector prioritizes stop views within a twelve-frame allowance and fills the remaining slots uniformly from the other frames, presenting the result in temporal order. Writing $\mathcal{S}_{12}$ for this selector, the checks have the form

$$v_k^{\mathrm{pre}} = V_\theta\big(g_k, \mathcal{S}_{12}(F_k^{\mathrm{pre}}), M_k^{\mathrm{pre}}, m_k^{\mathrm{pre}}\big), \qquad v_k = V_\theta\big(g_k, \mathcal{S}_{12}(F_k), M_k, m_k\big). \tag{18}$$

The superscript identifies evidence available at the first checked request, while the second expression uses the final task evidence. Both include the stopping views already captured by the tool.

The verdict is *complete*, *incomplete*, or *unknown*, with an evidence statement, searched-place names, and confidence. The judge has no action tools. If its response cannot be parsed, the same checking conversation is prompted once for the JSON verdict alone. A failed check or an unresolved response is recorded as unknown. The run recorder retains this verdict separately from incomplete when cross-tabulating verdicts with task success.

Pre-stop verification. If the handover file is missing, the orchestrator first prompts the actor once to write it. This file check is separate from verification. On the first request that reaches the judge, pre-stop verification withholds STOP exactly when

$$v_k^{\mathrm{pre}} \in \{\text{complete}, \text{incomplete}\} \quad \text{and} \quad v_k^{\mathrm{pre}} \neq c_k.$$

It returns the evidence to the existing acting session, allowing another observation or a revised search. An unknown verdict does not block closure, and a subsequent request is honored without another veto. The checked-request counter belongs to the task, survives recovery, and resets only at the next task. The one-time missing-handover prompt does not consume this checking opportunity. Only the orchestrator issues the resulting STOP. This bounded check gives the actor a chance to respond without allowing the judge to withhold closure indefinitely.

Post-stop certification. Certification assigns the ledger status in Equation 8 without changing the completed task's score. If model-based pre-stop verification and certification are both enabled and the action-step count has not changed since pre-stop verification, the stored verdict is reused. Otherwise certification opens a fresh checking session on the final evidence. Unknown remains UNRESOLVED in memory even though it permits the current task to close.

The two checks have separate switches because only pre-stop verification can change the current rollout. Sharing model parameters does not make actor and judge errors independent. Their separation instead controls which evidence the judge sees and which records the actor may write. Environment scores enter evaluation logs, not the model-based judge or the agent-facing ledger. The single-view intervention in Section 4.3 changes the visual evidence supplied to the judge.

B BENCHMARKS AND EVALUATION PROTOCOL

Table 8 summarizes our validation-unseen evaluations. The GOAT-Bench main comparisons across all four executors and the Opus 5 session and component study use the full Val-Unseen split, comprising 2,669 subtasks in 360 episodes across 36 scenes.

B.1 GOAT-BENCH

GOAT-Bench evaluates lifelong multimodal navigation in Habitat (Savva et al., 2019) on HM3D scans (Ramakrishnan et al., 2021). An episode places the agent in one house and gives it a sequence

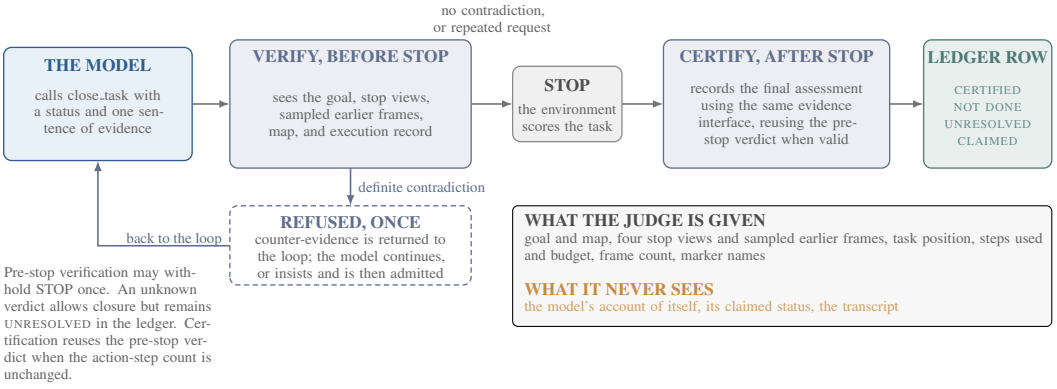


Figure 8: Verification at closure. Pre-stop verification can return counter-evidence. Certification records the outcome afterward, reusing the pre-stop verdict when the action-step count is unchanged.

Table 8: Evaluation scope on validation-unseen data.

Benchmark	Scenes	Episodes / tours	Tasks	Tasks per sequence
GOAT-Bench	36	360 episodes	2,669	5–10
IR2R-CE	11	36 tours	1,824	3–100

of goals to reach in turn, and the agent is not reset between them. Each goal is specified in one of three ways: as an object category, as a natural-language description that picks out one instance, or as an image of that instance. A goal counts as reached when the agent stops within the success radius of a navigable view point of the target instance, and efficiency is scored by SPL against the shortest path. We evaluate all 360 validation-unseen episodes across 36 scenes, retaining all 2,669 subtasks. The full-split comparisons use identical evaluation tasks across executors and their independent-session baselines in Table 1, and across all arms of the Opus 5 study in Table 3. Each GOAT-Bench episode and each IR2R-CE tour is a separate benchmark run, starting with empty maps, task records, and house notes. Memory persists between tasks of that run but is not inherited by another run, even in the same scene. Run-end notes therefore provide no input to these evaluations. In our protocol the agent's position, map, and markers come from the navigator's own visual SLAM rather than from the simulator, and simulator truth is used only to score.

B.2 IR2R-CE

We evaluate all 36 tours in the standard IR2R-CE validation-unseen split, retaining all 1,824 tasks across its 11 scenes. IR2R-CE builds on VLN-CE (Krantz et al., 2020) in Matterport3D (Chang et al., 2017) and evaluates instruction following over tours. A tour joins consecutive Room-to-Room episodes from one scene in the continuous environment, and the agent keeps whatever it has learned from one episode into the next, even when transferred to the next episode's start. Each episode is scored as in R2R-CE by success and SPL, and the tour as a whole by t-nDTW, the normalised dynamic time warping between the tour's driven path and its reference. We use the validation-unseen split, as for GOAT-Bench. We compute t-nDTW with the benchmark's standard trajectory handling and scoring procedure, using the same convention as the comparison methods.

B.3 Comparison sources and settings

The Base Model column in Table 1 identifies the main language or vision-language model used for the reported navigation result. It does not list every perception or mapping module. For example, ObsGraph additionally uses GPT-4o-mini to construct its room hierarchy (Lee et al., 2026), while AstraNav-Memory augments Qwen2.5-VL-3B with a DINOv3 visual encoder (Ren et al., 2025). The published 3D-Mem GOAT result uses GPT-4o to select targets from memory and frontier snapshots, with movement handled by Habitat's pathfinder (Yang et al., 2025). GSMem also uses GPT-4o for GOAT reasoning, but combines it with Gaussian scene reconstruction, retrieval-rendering, and

exploration planning (Lu et al., 2026). Matching the reasoning-model name would leave these perception and control pipelines different. Our same-backbone controls instead retain the executor and navigation interface while changing the surrounding harness. Their gains measure the full system change, including memory, recovery, and completion checks.

Prior rows in the main tables retain the source evaluation protocols. The dagger in Table 1 identifies the 278-subtask evaluation used by the marked methods, including our GPT-4o subset result. Our unmarked rows and the same-backbone gains use the full split. We report 3D-Mem's subset result of 69.1 s-SR and 48.9 SPL, rather than its full-set result of 62.9 and 44.7 (Yang et al., 2025). MemoryExplorer reports the same 36-scene, one-episode-per-scene protocol (Wang et al., 2026c). GSMem describes the validation-unseen benchmark with ten episodes per scene (Lu et al., 2026). The GT-pose columns in Tables 1 and 2 identify methods that use simulator-provided localization. The CMA, TourCMA, PoolCMA, and PoolEndCMA policies instead use RGB-D, instructions, and action history without a pose input (Krantz et al., 2023). Our navigator uses visual SLAM pose, while simulator truth is reserved for evaluation. Table 9 retains additional published baselines omitted from the main table for space.

Table 9: Additional published GOAT-Bench Val-Unseen results. All listed methods use simulator-provided pose.

Method	Type	s-SR	SPL
SenseAct-NN Monolithic (Khanna et al., 2024)	trained	12.3	6.8
Modular CLIP on Wheels (Gadre et al., 2023; Khanna et al., 2024)	zero-shot	16.1	10.4
VLMnav (Goetting et al., 2024; Ji et al., 2025)	zero-shot	20.1	9.6
Modular GOAT (Khanna et al., 2024)	zero-shot	24.9	17.2
DyNaVLM (Ji et al., 2025)	zero-shot	25.5	10.2
SenseAct-NN Skill Chain (Khanna et al., 2024)	trained	29.5	11.3
TANGO (Ziliotto et al., 2025)	zero-shot	32.1	16.5
AnyGoal, single agent (James et al., 2026)	zero-shot	41.9	14.4
MTU3D (Zhu et al., 2025)	trained	47.2	27.7

GOAT e-SR follows the all-subtasks criterion of Niu et al. (2026). The original Monolithic baseline reports 12.3 s-SR and 6.8 SPL without e-SR. The 0.8 e-SR reported by Niu et al. (2026) belongs to their reimplementation, with 12.6 s-SR and 8.8 SPL, and is not combined with the original row. For IR2R-CE, CMA variants and MAP-CMA come from Krantz et al. (2023). OVER-NAV uses its continuous-environment result (Zhao et al., 2024). ETPNav and HNR use the IR2R-CE evaluations in Table 2 of Han et al. (2026), not their original single-episode R2R-CE scores. SeqWalker also uses that table, not its distinct SH IR2R-CE benchmark. Table 10 retains three CMA variants omitted from the main table for space.

Table 10: Additional published IR2R-CE Val-Unseen results (Krantz et al., 2023).

Method	Type	GT pose	s-SR	SPL	e-SR	t-nDTW
TourCMA	trained	×	18	17	–	36
PoolCMA	trained	×	16	15	–	36
PoolEndCMA	trained	×	18	16	–	38

We evaluate NavHarness with Qwen3.8-27B, GPT-4o, Opus 5, and GPT-6 Astra. All four executors use the same mapping, memory, and recovery mechanisms. Following MIP (Zhou et al., 2026), the reasoning core is a multi-round coding-agent session. GPT-6 Astra uses Codex with medium reasoning effort, Opus 5 uses Claude Code, and both Qwen3.8-27B and GPT-4o use mini-SWE-agent. NavHarness manages the task and recovery boundaries around these sessions rather than replacing their inner reasoning loops. Controlled component comparisons use Opus 5 throughout.

Models assigned to auxiliary roles. The standard launcher routes every role to one model. Pre-stop verification and post-stop certification use the same judge model. The acting session writes its

recovery handover before replacement. Consolidation uses a separate session with the same model when enabled. Except in the judge-swap comparison in Appendix B.7, one model fills all auxiliary roles as well as navigation, keeping model assignment fixed within each same-backbone comparison. None of these roles receives evaluator success labels. The saved Opus and Astra GOAT runs confirm same-model judging. The independent-session baseline has no judge or cross-task consolidation.

Default stopping procedure. The benchmark configuration checks the first stopping request and certifies the completed task (precheck=model, certify=model). "No post-stop certification" in Table 3 removes only the post-stop verdict and leaves the stopping procedure unchanged. The independent-session baseline omits both checks. The additional independent-session-plus-verification control enables pre-stop verification. The full-system pre-stop verification ablation uses precheck=none while retaining certify=model.

The two baseline sessions. Independent sessions use a fresh coding session for each task, with only observation and action tools. The goal, observations, reasoning, and tool responses remain in the current conversation. The session has no access to external working or long-term memory, including maps, markers, ledger entries, or handover files. Single session continues that conversation from one task to the next, with backend compaction when its context window fills (Appendix A.3). The session baselines use the same task budget as the full harness and compare complete system policies. The reference configuration uses full-text ledger access (ledger=text).

Method labels. The Type labels in the main tables are descriptive rather than mutually exclusive method classes. *Agentic* identifies tool-mediated model reasoning, *trained* and *zero-shot* describe task-specific training, and *hybrid* denotes a combination of learned navigation and higher-level reasoning. A system can satisfy more than one of these descriptions. The labels do not establish backbone equivalence.

Task budgets. Table 11 gives the common navigation budgets for the GOAT controlled comparisons. Recovery shares the task's remaining allowances. The motion interface is specified in Appendix A.2, and session management in Appendix A.3.

Table 11: Budget accounting for the controlled GOAT comparisons.

Quantity	Limit or counting rule
Movement budget	500 executed forward or turning actions per task
Model-turn budget	200 navigation-model turns per task
Action batching	Each executed action is charged individually
Recovery allowance	At most one recovery, sharing the remaining task budgets
Fallback trigger	80 turns, without resetting either budget
Observation and memory	No movement-action charge

Runtime and context footprint. In GOAT-Bench Val-Unseen profiling runs, Opus 5 and GPT-6 Astra have amortized wall times of 129.5 s and 67.8 s per task, respectively. We sum the elapsed times of the evaluated scene runs and divide by the number of tasks, including setup, navigation, and boundary processing. The mean per-task peak prompt sizes are 20.7k and 23.2k tokens, respectively. These describe context occupancy. End-to-end wall time includes recorded auxiliary processing. The input-cost illustration in Appendix A.3 uses stated volume and price assumptions, rather than a measured API bill.

B.4 Statistics

The GOAT-Bench full-harness and independent-session comparisons for all four executors and the Opus 5 session and component study use three seeds on the same 2,669 subtasks in 360 validation-unseen episodes across 36 scenes. Tables 1 and 3 report mean scores and standard deviations across runs. The IR2R-CE NavHarness and independent-session results likewise report means and standard deviations over three seeds for all four executors (Table 2). The Opus GOAT-Bench comparisons

pair shared tasks by scene and episode identifier, counting each task once per run. Paired 95% confidence intervals pool the repetitions and use 5,000 scene-clustered bootstrap resamples, keeping paired outcomes together. These are pointwise intervals for component analysis, without multiple-comparison adjustment. For continuous deployment (Figure 3), scores and paired changes pool the tasks across all 36 houses. The 95% CIs use houses as the resampling unit, keeping the Full and No consolidation results paired.

B.5 Chain success

Chain success is the fraction of sequences in which every task succeeded: an episode on GOAT-Bench, a tour on IR2R-CE. It is the quantity that a deployment cares about and that task success hides, since one failure anywhere in the sequence fails the chain.

B.6 Interventions

Table 3 separates system comparisons (A) from component interventions (B–D). All use Opus 5 and the same ordered 2,669-subtask full-split evaluation. The reference retains maps and task records, uses structured recovery handovers, and enables both pre-stop verification and post-stop certification with four stop views. Each B–D row starts from this reference independently. No row inherits the changes of the preceding row. Differences and CIs are always relative to full, including the independent-session-plus-verification row.

A. Complete session policies. Independent sessions start a context-only coding session for each task, without external memory even within a task. Single session continues the same conversation across tasks as defined in Appendix A.3. Independent sessions + pre-stop verification adds the stopping check to this policy without adding memory. Comparing this control with full tests the remaining harness features jointly while holding pre-stop verification enabled. None of these rows is a memory-only ablation. Comparing 62.4 with 58.9 describes a different contrast from the paired change in Table 3, which compares 62.4 with the full-system 81.5.

B. Cross-task memory. Clear map after each task (keep records) removes the map and spatial markers while retaining task records and the ability to map during each task. Unlike the context-only independent sessions, this control still uses working memory. Hide earlier task records (keep map) keeps the map but withholds textual history from earlier tasks, including the ledger, previous handovers, and historical task files. These records are neither supplied in the opening context nor accessible through file tools, although they may still be saved for evaluation. The current task can write its own records and pass recovery notes between attempts. These interventions ask whether spatial knowledge and records of earlier attempts can substitute for each other. Recovery, structured handovers, and both verification functions remain enabled. The additional No cross-task memory control combines these two interventions. Every task starts without an inherited map, spatial markers, ledger entries, or handovers, but can build and consult its own map and records, including recovery notes passed between attempts of that task. Task budgets and both completion checks are unchanged. Its standard benchmark results appear in Table 3. The extended deployment in Section 4.4 uses a different intervention: No consolidation retains maps and task records with the same persistence rules as full, but disables the run-end session that writes house notes. Both use the same executor, task sequences, budgets, recovery, and completion checks. This comparison measures the added value of consolidated long-term memory beyond retained spatial and task-level records.

C. Recovery availability and handover content. No recovery removes both requested and fallback restarts, while retaining the complete 200-turn and 500-step task budget. The three handover interventions keep recovery triggers, scope control, and the shared budget unchanged. Empty handover supplies no failed-attempt note. Another task's handover supplies a mismatched note. Matched-length summary replaces the structured note with an ordinary summary. The first contrast tests whether recovery helps. The remaining contrasts test what information must cross the restart. They do not switch recovery off, and the summary comparison tests information selection and organization together.

D. Verification use and evidence. No pre-stop verification retains post-stop certification but never withholds STOP. No post-stop certification retains pre-stop verification but leaves the task account without a post-stop certified status. One stop view instead of four keeps both verification functions and changes the stop-position evidence from a look-around to the forward frame. It therefore tests the evidence supplied to verification, not the existence of either check. These are separate rollouts, so changing a mechanism can change subsequent observations, trajectories, and memory contents.

B.7 Memory access and judge choice

Fixed-access historical memory. To test whether choosing when and what to retrieve matters beyond retaining records, we replace agent-directed retrieval with a fixed-access policy. Mapping tools, record writing, recovery handovers, and completion checks remain unchanged. Both conditions use Opus 5, receive the same initial memory entries, and can query the online map freely. The control retrieves historical text at task start, after recovery, and every ten navigation-model turns. A fixed query concatenates the goal description and latest observation description, using the same description interface in both conditions. BM25 retrieves up to five original excerpts within a 2,000-token budget, including sources and recorded status. Records are divided into 400-token chunks with 50-token overlap. Overlapping excerpts are merged, and ties favor newer records. The navigator can still consult current-task records and query the map, but cannot bypass the fixed policy by reading historical text through other file tools.

We use the same full-split GOAT-Bench evaluation, three-run protocol, and task budgets as Table 3. Agent-directed access raises s-SR by 7.0 percentage points over fixed access (95% CI $[+5.9, +8.2]$), with SPL increasing from 49.0 to 55.0 (Table 12). Thus, supplying original records on a fixed schedule does not recover the full benefit of letting the navigator choose its queries as observations arrive. The comparison tests query timing, selection, and follow-up together. Both conditions start with empty memory and retain records in the same way, although their later contents may differ as trajectories diverge. Neither inherits house notes from earlier runs.

Table 12: Agent-directed and fixed-access historical memory on GOAT-Bench with Opus 5. Scores are mean $\pm$ standard deviation over three runs. The paired difference is NavHarness minus fixed access. The reference wall time comes from the profiling runs in Appendix B.3.

Metric	NavHarness	Fixed access
s-SR (%)	81.5 ± 0.7	74.5 ± 0.8
SPL	55.0 ± 0.6	49.0 ± 0.7
Mean time / task (s)	129.5	137.2
Paired Δs-SR (pp)	$+7.0$	
95% CI (pp)	$[+5.9, +8.2]$	

Table 13: Same-model and cross-model judging. Both completion checks use the listed judge, with all other roles unchanged. Scores are mean $\pm$ standard deviation over three runs.

Navigator	Judge	s-SR (%)	SPL
Qwen3.8-27B	Qwen3.8-27B	71.7 ± 0.5	48.2 ± 0.4
Qwen3.8-27B	Opus 5	71.3 ± 0.4	48.5 ± 0.5
Opus 5	Opus 5	81.5 ± 0.7	55.0 ± 0.6
Opus 5	Qwen3.8-27B	81.7 ± 0.6	55.2 ± 0.5

Cross-model judging. We swap the judges between Qwen3.8-27B and Opus 5 to test whether navigation performance depends on using the navigator's own model for completion checks. Each navigator is evaluated with its default judge and with the other model judging both pre-stop verification and post-stop certification. The navigator, recovery-note writer, and consolidation model remain unchanged, as do the evidence, instructions, stopping rules, and task budgets supplied to each configuration.

The swaps change s-SR by -0.4 points for Qwen and $+0.2$ for Opus, with similarly small changes in SPL (Table 13). Using a different judge therefore leaves navigation performance close to the default in both configurations. This does not mean completion checks are unimportant: group D of Table 3 tests their contribution separately. Nor do similar navigation scores establish equal judgment accuracy or exclude errors shared by the two models.

C Complete Results

This appendix gives the paired statistics behind the main tables and the diagnostics behind each component. The GOAT-Bench score comparisons use all 2,669 subtasks in 360 validation-unseen episodes, matching Tables 1 and 3. Three-seed score estimates and confidence intervals are identified explicitly. The event percentages below are descriptive log summaries and should not be read as additional three-seed effect estimates.

C.1 Component effect profile

Figure 9 summarizes the controlled Opus interventions in Groups B–D of Table 3, including the matched-length ordinary-summary control. The pre-stop verification intervention is included separately from certification. It displays the aggregate effect of each intervention with paired uncertainty. The diagnostics below connect these effects to repeated exploration, failed attempts, and inaccurate completion records. Table 3 reports the scores and paired confidence intervals together. The intervals pool three seeds per configuration using the scene-clustered procedure in Appendix B.4.

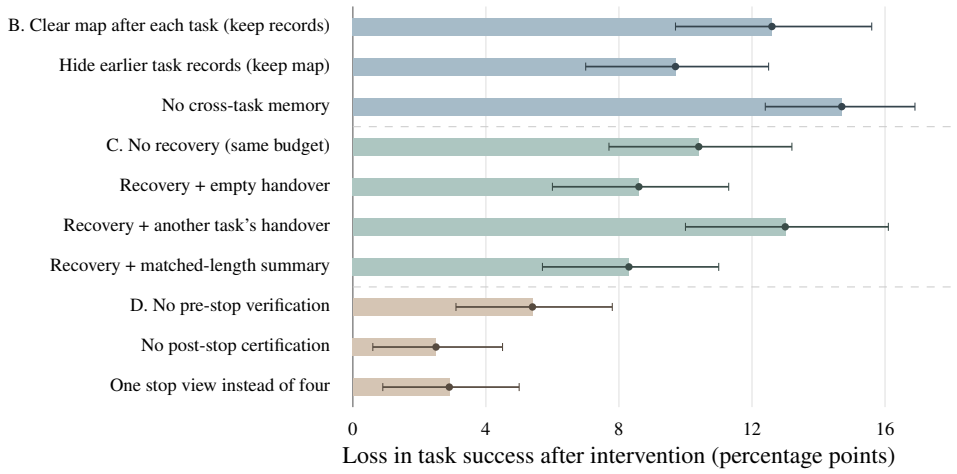


Figure 9: GOAT-Bench full-split component effects with Opus 5, grouped as in Table 3. Bars show the loss relative to the full harness, with 95% confidence intervals of the paired change over three seeds. Positive values favor the full harness. Every interval excludes zero.

C.2 Session and system comparisons

Independent sessions use only the current task's conversation, with no external map, task records, or long-term memory. They differ from the map-reset ablation, which retains mapping within each task. Tables 1–3 report the system and session-policy comparisons, including the IR2R-CE independent-session results for all four reasoning models.

With Qwen3.8-27B held fixed on the full 2,669-subtask evaluation, NavHarness improves s-SR by 30.3 points, SPL by 20.7 points, and e-SR by 11.3 points. The corresponding gains with GPT-4o are 34.8, 26.9, and 20.5 points. With GPT-6 Astra held fixed on the same evaluation, NavHarness improves s-SR by 18.6 points, SPL by 22.2 points, and e-SR by 23.3 points over independent sessions. These are aggregate differences. Paired uncertainty requires the corresponding task-level records, and the paired intervals below concern Opus only.

In the full-split Opus study, independent sessions lose 22.6 success points relative to full (95% CI $[-26.3, -19.0]$), and a single session loses 26.6 points ($[-30.5, -22.8]$). Their SPL scores are lower by 20.1 and 22.2 points, respectively. These comparisons evaluate complete system policies. SPL combines task success with path efficiency, so these aggregate differences do not isolate path savings on tasks completed by both systems. Table 3 gives all component scores and paired success intervals, including the empty-note and ordinary-summary controls. The persistent-map and ledger removals also reduce SPL by 15.2 and 13.2 points.

Table 14: The full harness by goal modality on GOAT-Bench. Percentages.

Goal	s-SR	SPL
Object category	86.5	62.4
Language description	75.2	48.8
Image	82.2	53.0

C.3 Within-task recovery and handovers

The matched-length ordinary-summary control reaches 73.2% s-SR and 47.0% SPL, compared with 81.5% and 55.0% for the full harness. Its paired change in s-SR is -8.3 percentage points, with a 95% confidence interval of $[-11.0, -5.7]$ points. This comparison tests the structured note against a general summary, complementing the empty-note and task-mismatched-note interventions in Table 3.

Figure 10 breaks the recovery events on GOAT-Bench down three ways. The full harness recovers on about a quarter of its tasks, with 39% of recoveries requested by the model and 61% fired by the 80-turn fallback. Among the recovery events recorded in each arm, 56.1% lead to success with the attempt's own note, against 31.7% with an empty note and 35.0% with the note of a different task. The model asks early, at a median of 32 turns against 84 for the fallback, and its requests succeed more often, 61.5% against 52.5%, although the two triggers select different failures. The recovered-task sets can also differ across note interventions. These conditional rates describe the events observed in each run, while Table 3 tests the effect of the note intervention on overall task success.

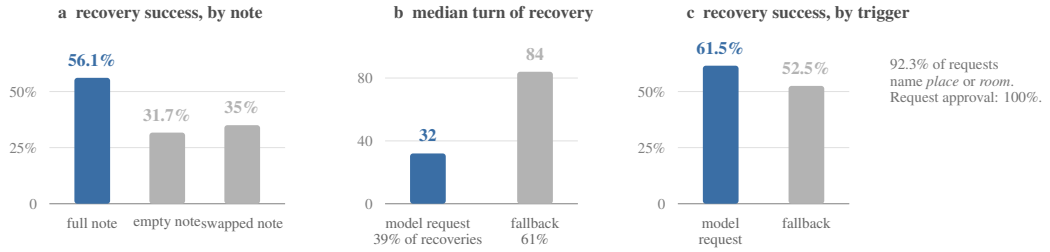


Figure 10: Recovery on GOAT-Bench. Left: the share of recoveries that lead to success, by what is handed to the fresh attempt. Middle and right: the model requests recovery far earlier than the fallback and its requests succeed more often.

C.4 Cross-task memory

Disabling all cross-task memory while retaining within-task mapping, recovery, and verification yields 66.8% s-SR and 37.4% SPL. The paired s-SR change is -14.7 points (95% CI $[-16.9, -12.4]$), isolating the contribution of memory carried between tasks within the otherwise intact harness. Removing either task records or spatial persistence increases physical exploration. Mean steps per task go from 127 with the full harness to 159 without the ledger and 168 without the persistent map, and without the map the model drops 3.4 markers per task instead of 1.3 (about $2.6\times$) and requests recovery 1.8 times as often, rebuilding each task what the previous one had already recorded.

C.5 VERIFICATION USES AND EVIDENCE

Pre-stop verification interventions. Disabling only pre-stop verification reduces full-system success from 81.5% to 76.1%, a change of -5.4 points with paired 95% CI $[-7.8, -3.1]$. SPL changes from 55.0% to 47.0%. The interval excludes zero over the three seeds. With pre-stop verification enabled in both configurations, the full system exceeds independent sessions by 19.1 success points (95% CI 16.4–22.4). This comparison holds the stopping check fixed, but evaluates the remaining harness features jointly, including persistent state and recovery.

Local stopping effects from logs. The first refused stopping request provides a second view of pre-stop verification. For task i, let y_i^{stop} be evaluator success if STOP were executed at that recorded state, and y_i^{final} the task's eventual success after the refusal. A rescue has values $(0, 1)$ and a harm has values $(1, 0)$. The local net change is

$$\widehat{\delta}_{\text{stop}} = \frac{1}{N} \sum_{i \in \mathcal{R}} \left(y_i^{\text{final}} - y_i^{\text{stop}} \right), \tag{19}$$

where $\mathcal{R}$ contains tasks with a refused first request and N is the number of tasks in the GOAT-Bench Val-Unseen diagnostic logs. This paired comparison conditions on the history produced by the policy with pre-stop verification enabled. It estimates the current-task consequence of continuing at those states, not the effect of rerunning an entire sequence without pre-stop verification. Recovering y_i^{stop} requires the benchmark's success criterion at that state, not the agent's claim or the judge's verdict.

Table 15: Pre-stop diagnostics. Values are percentages except the net change, which is in percentage points.

Diagnostic	Full	Independent + pre-stop verification
Tasks with a refused first stop	11	20
Rejection precision	73	75
Rescue among correctly refused stops	45	35
Rescues / all tasks	3.6	5.0
Harms / all tasks	0.7	1.1
Net local success change	+2.9	+4.0

The reported summaries suggest that independent sessions reach questionable stopping states more often, while the rejection precision is similar in the two configurations. The local estimates measure continuation after a refused STOP, whereas rollout differences include changes to the entire trajectory and later state.

Post-stop certification. Among claimed completions, the four-view certification judge correctly rejects false claims on 12.2% and incorrectly rejects true claims on 4.1% (Table 16). Both percentages use all claimed completions as their denominator. The next task receives the verdict alongside the claim and account (Figure 11). Success of the task that follows a false record is 66.7% with certification and 61.4% without. After a true record it is 82.7% and 81.6%. These successor rates condition on the histories produced by each configuration, rather than matched predecessor states, and do not isolate the effect of certification. Table 16 compares the certification judge's verdicts with environment truth under the four stop views and under a single forward frame. The two runs have different terminal observations, so the comparison describes the runs rather than isolating the view count.

On the same claimed-completion population, four-view accuracy is 90.7%, compared with 82.6% for accepting every claim. The corresponding values for the single-view run are 82.9% and 80.6%. Cohen's κ is approximately 0.67 and 0.40, respectively, computed from the rounded percentages above. The four-view checker rejects 70.1% of false claims and accepts 95.0% of true claims. Of its rejections, 25.2% concern true completions. Both checkers improve on accept-all accuracy, with a larger margin in the four-view run. These results support using certification to distinguish supported outcomes from unsupported completion claims, while retaining the possibility of false acceptance and false rejection. The post-stop certification ablation leaves the pre-stop verification unchanged (Appendix B), so it does not isolate that check's effect on navigation success.

Table 16: Post-stop certification outcomes and accuracy (% of claimed completions). Parentheses give accept-all accuracy.

Evidence	Certified, true	Certified, false	Rejected, true	Rejected, false	Acc. (accept-all)	κ
Four stop views	78.5	5.2	4.1	12.2	90.7 (82.6)	0.67
One forward frame	74.2	10.7	6.4	8.7	82.9 (80.6)	0.40

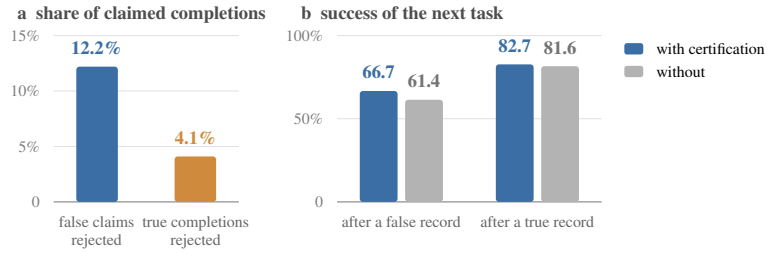


Figure 11: Verification on GOAT-Bench. Left: four-view rejections as a share of claimed completions. Right: the success of the task that follows a false or a true ledger record, with and without certification.

D Continuous Deployment and Mechanism Analysis

D.1 Component effects and failure diagnostics

Table 17 links aggregate intervention losses to the failure patterns observed in the corresponding runs. The diagnostics help explain what each component supports, but are not separate effect estimates for tasks stratified by difficulty or failure type.

Table 17: Component effects and associated diagnostics. Changes in s-SR are aggregate interventions, not effects estimated within each stress condition. The note intervention substitutes another task's note.

Component	**Δs-SR**	**Absorbs**	**What shows when it is removed**
Map and markers	-12.6	revisits and displacement	steps $+32\%$, markers written about $2.6\times$
Ledger	-9.7	repeated goals	steps $+25\%$ despite retaining the map
Recovery note	-13.0	a stalled search	recovery success 35% instead of 56%
Recovery	-10.4	an unproductive attempt	requests come long before the fallback
Four-view evidence	-2.9	an ambiguous closure	claimed-completion κ: 0.40 vs. 0.67 in separate runs
Certification	-2.5	an unsupported claim	12.2% false and 4.1% true completions rejected (share of claims)

D.2 With and without consolidation across multiple houses

The deployment schedule extends GOAT-Bench to ten tours in the same house, organized across simulated days. Tasks within a tour retain GOAT-Bench's continuous motion, but tour boundaries are discontinuous: the robot is placed at the next tour's prescribed starting location. This models a household robot resuming work after a shutdown. It must recover its bearings from current observations, SLAM, and remembered landmarks before reusing stored routes and target locations. The spatial-record cases below examine how sessions handle inherited coordinates when the SLAM frame changes and how descriptive knowledge can still guide navigation.

The extended evaluation covers all 36 houses (Figure 3). This appendix provides detailed profiles and case studies for three of these houses, labeled A, B, and C, comparing two configurations of the same executor on matched task sequences. Both retain maps, markers, the place graph, and task records,

including the ledger and handovers, with the same persistence and access rules. The *No consolidation* control disables only run-end consolidation. The *full* configuration also distills the journal into house notes for later runs, which navigation sessions read through file tools. All other settings are identical: Claude Opus 5, one session per task, 200 turns and 500 steps per task, at most one recovery, the four-view certification judge and the pre-stop verification. On the first day of each house no notes exist yet; the executor can only read the journal being appended that day. The executor may read its notes but not write them: Read, Glob and Grep are fenced to the notes directory, and every Write or Edit is refused. In the case-study logs, *ledger* or *journal* also labels the full task records, including the actor's account, rather than just the compact index in run/ledger.md. We label these detailed excerpts as task records below and retain the original wording in quotations. Boxed passages are the model's own writing, quoted verbatim, with our highlights: a judgement about its own memory, an admission of error or cost, advice to its future self, or a diagnosis.

What was asked for, and what was not. Both configurations can build a working map, add markers, and produce task records during navigation, and both can retrieve these records later. Only the full configuration consolidates this experience into an index, house overview, room notes, and navigation skills. The comparison tests what this long-term organization adds beyond persistent maps and task records.

Both configurations receive the same deployment task prompt, tool descriptions, and handover instruction. The logs use CLI-native names such as Read, Glob, and Grep. Table 18 describes the instructions associated with the case-study records. Both configurations have access to the ledger and detailed task records; only full receives the house notes produced by consolidation.

Table 18 separates explicitly requested operations from more specific behaviors observed during execution. A behavior not explicitly scripted can still follow from the general memory instructions.

Table 18: Deployment instructions and observed behaviors. A behavior not explicitly scripted may still be supported by general memory instructions.

Behaviour	**Instruction or tool support**	**Relation**
Report the goal, outcome, and rooms passed	Handover instruction	Requested
Report which markers were dropped	Handover instruction	Requested
Write down unsuccessful searches	Handover instruction	Requested
Move a marker	Marker tool supports re-placement	Tool-supported
Question an earlier task account	Warning that accounts may be wrong	Requested
Correct a particular marker	General permission to maintain markers	Not scripted
Diagnose a particular instrument fault	General reasoning and recovery tools	Not scripted
Reject a stored route and navigate by sight	Access to memory and current views	Not scripted
Check a note against current views	Instruction to treat notes as recollection	Requested
Identify a particular stale note	General warning about stale notes	Not scripted
Attempt to edit inherited notes	Writes outside permitted paths are refused	Unrequested attempt

Both arms use the same task order in each case-study house. Figure 15 shows the separate house profiles, and Figure 12 summarizes the complete runs. Figure 3 reports the pooled 36-house evaluation.

Different environments, different gains. In House A both success and route efficiency improve, and the SPL gain splits evenly between tasks that flip to success and tasks both runs complete. In House B success rates are already high and differ only slightly, while route efficiency improves substantially: the both-completed tasks are shorter, and the share of tasks closed within ten steps rises from 8% to 18%. In House C the gain is almost entirely in success, and the both-completed tasks appear no more efficient; that is an artefact of chained starts, since on the subset whose starts coincide the efficiency rises in each environment (Figure 13). By goal modality the gain is broadest on described goals: success on them rises from 77% to 95% in House A and from 60% to 72% in

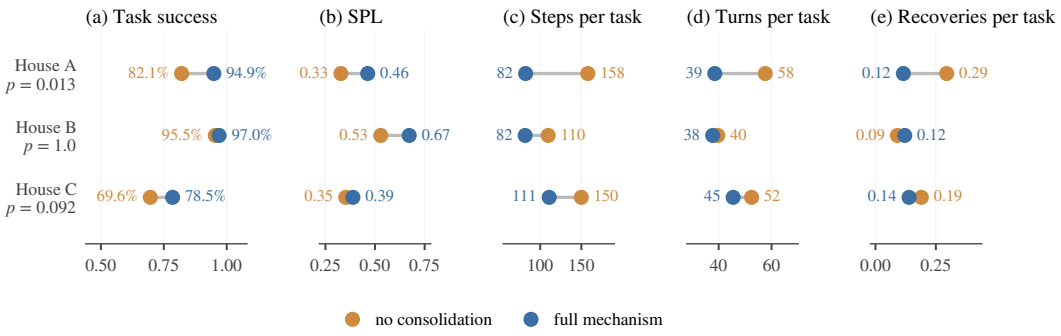


Figure 12: Aggregate comparison for deployments A–C. Success and SPL use environment truth. Steps, turns, and recoveries are means per task; p is an exact McNemar test on task success.

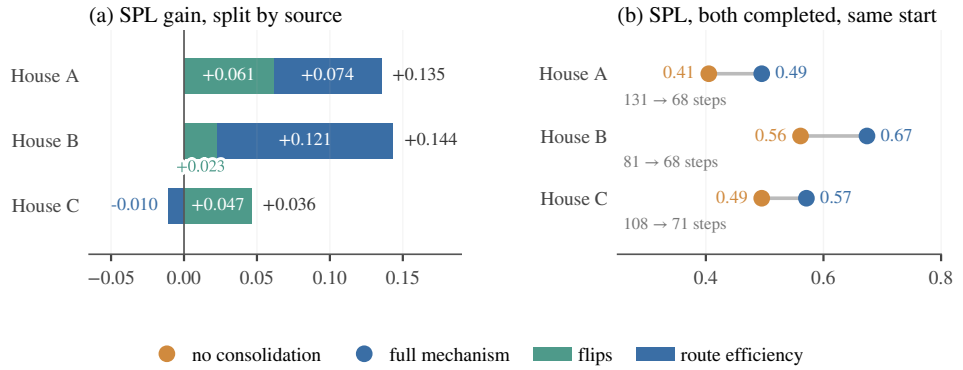


Figure 13: Where the SPL gain comes from. **(a)** The gain split into tasks that only one run completes (*flips*) and tasks both runs complete (*route efficiency*); in House C the efficiency term is negative. **(b)** Because tasks chain, the two runs do not always start a task from the same place, and SPL is normalised by each run's own shortest path; on the both-completed tasks whose starts differ by less than 0.5 m, efficiency rises in each environment.

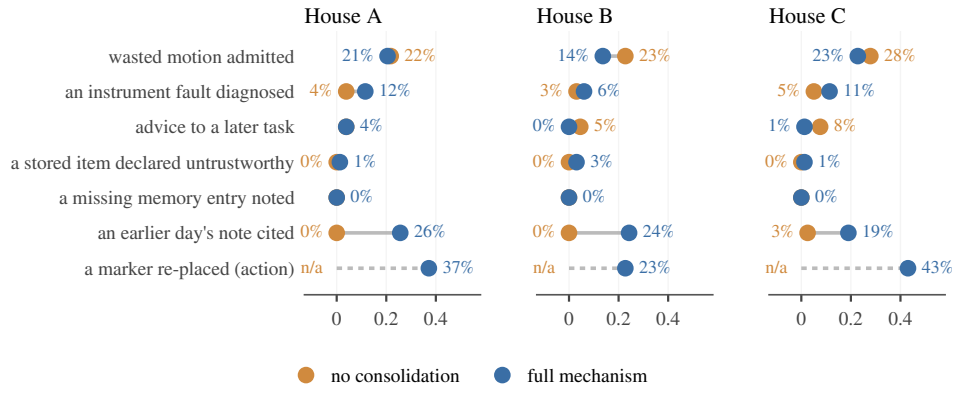


Figure 14: Keyword-match rates in task accounts. Textual mentions need not correspond to a verified memory read or revision. The last row is an action rather than a phrase: a marker re-placed under a label that already existed. Both configurations retain markers; untabulated rates are marked n/a.

House C, and their steps per task fall from 217 to 93 and from 204 to 157; the object goals of House B are all completed by both runs and differ only in steps, 66 against 41 per task. The closing judge accepts a false completion on 10%, 1.5% and 19% of tasks without consolidation and on 3%, 0% and 18% with it (Houses A, B, C), and refuses a true one on 8–11% without and 6–15% with. The

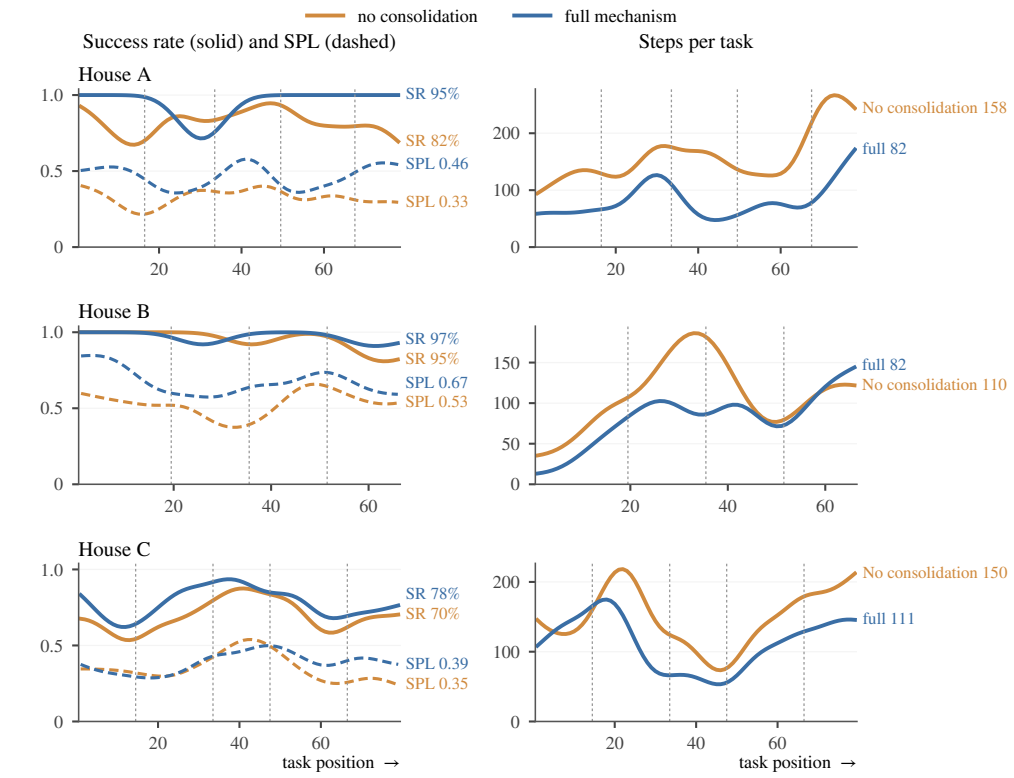


Figure 15: Performance trends in deployments A–C. Left, success rate (solid) and SPL (dashed); right, steps per task. Ochre denotes no consolidation and blue the full mechanism. The label at each line end is the run's mean over all tasks; dashed verticals are the day boundaries of the full run. The raw outcome of every task is in Figure 16.

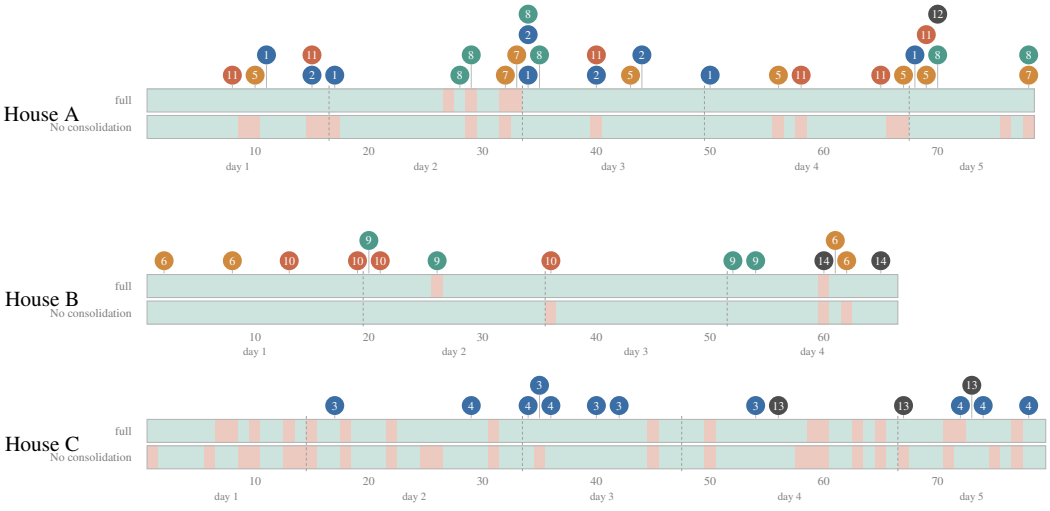


Figure 16: Where the cases sit in deployments A–C. One cell per task in order, teal for success and red-brown for failure by environment truth; the upper strip of each house is the full run, the lower the run without consolidation, and dashed lines are the day boundaries of the full run. Numbered dots mark the tasks each case quotes: revising spatial records (Cases 1–4), recording failure and checking completion (Cases 5–7), instruments and procedure (Cases 8–9), auditing and consolidating memory (Cases 10–11), declining stored experience (Cases 12–14). The cases are spread over the days rather than clustered at their starts, and most of them fall on tasks the two runs decided differently.

fourteen cases below are selected from forty-eight analyzed cases to illustrate spatial revision, failure recording, instrument diagnosis, consolidation, and selective reuse. Each includes the corresponding no-consolidation execution where available. This behavior-based selection includes both benefits and failures, rather than estimating their frequency across all navigation tasks. The cases trace how memory enters individual decisions. Component effects are tested by the controlled interventions, while differences between these full-system and no-consolidation trajectories are not decomposed into independent gains from each record or tool.

Task by task (Figure 15), the full run's success in House A drops below the no-consolidation run's only on the second day, where two failures on the same lamp table (Case 7) coincide, and its steps stay below throughout. In House B the success curves are close, while the larger difference is in the steps of the first two days, when the closet tasks are answered from the marker under the body (Case 6); in House C the SPL curves cross several times, the gain being in tasks that flip to success. Figure 16 places the cases on the same strips.

D.2.1 Revising spatial records (Cases 1 to 4)

Case 1. Correction that tracks the map frame, not the object. This case distinguishes a tracker restart between runs from tracking loss within a run. Saved markers survive either event, but their coordinates are usable only when aligned with the current tracker frame. The following measurements track re-placements in the full runs. In House A the executor re-placed a marker under an existing label 29 times during the deployment. Each re-placement can be measured twice: in the map, as the distance between the old and the new marker, and against the simulator, as the distance between the two standing positions. Sixteen are refinements, with both distances small. Thirteen are frame events, nine at the start of a day, when SLAM restarts from the origin and yesterday's markers keep yesterday's coordinates, and four after a tracking loss within a day; in seven of these the "correction" was itself placed in a frame that was already broken. Houses B and C show the same shape (Figure 17): of the 15 re-placements in House B, 11 are frame events and 2 are moves the simulator confirms; of the 34 in House C, 21 are frame events.

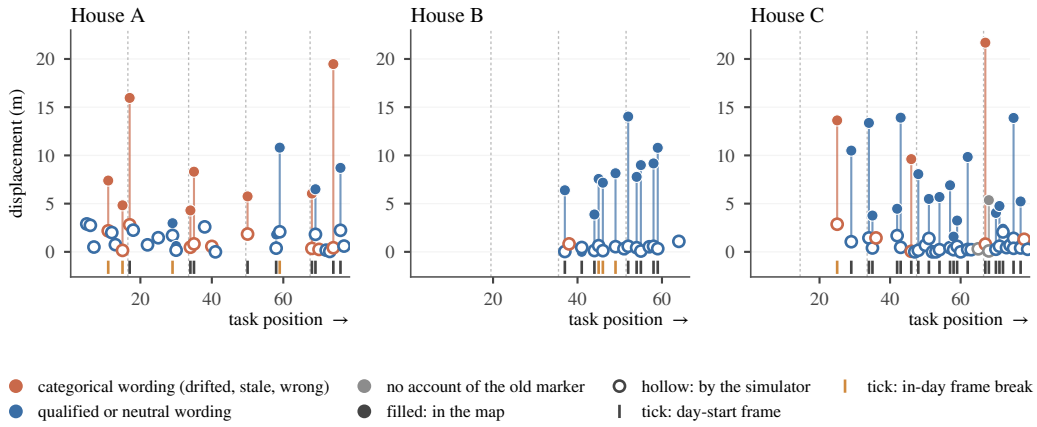


Figure 17: Every re-placement of an existing label in the three full runs, placed at its task position. For each, the filled dot is the displacement in the map and the hollow dot the displacement of the standing position by the simulator, joined by a line; the colour is the wording of the account about the old marker. Ticks below the axis mark the cause: black, a day-start frame reset; ochre, a frame break within a day. Categorical wording follows the map, not the object: the large map displacements with small true ones are frame events, and the accounts call them drift.

House A · task 17 · day 2 · full · task record first task of the day; in the new frame the inherited marker was 16 m from the rug

Find any carpet.

I re-dropped the “freezer room” marker on the rug — the old one was badly drifted, and previewing routes to it failed. […] once red fills the bottom of the frame you are already on it.

House A · task 17 · no-consolidation · task record 24 steps; closed on the hallway carpet, accepted by the verifier, a failure by the environment (4.6 m)

Find any carpet.

Goal was to stand on the hallway carpet. I reached it — I'm on the beige carpet now, in the corridor lined with wooden doors […] I dropped one marker, “carpet hallway”, where I stopped. Nothing else explored yet.

What it shows. The accounts track the size of the observed offset more reliably than its cause. Categorical words such as *drifted* and *stale* appear in nine of twelve large offsets, while fifteen of sixteen small offsets receive qualified or no comment. The position records associate the large offsets with coordinate-frame changes, but the notes generalize them into *Markers here drift.* Retaining a marker therefore also requires checking whether its frame remains usable. On the thirteen frame-event tasks the full run still completes more tasks with fewer steps while using its other retained knowledge.

Case 2. A misplaced marker, caught nineteen tasks later. In House A the label *laundry bath* was first placed at task 4 and placed again at task 15, in a frame that a tracking loss had already displaced: the new marker sat 4.8 m from the old one in the map while the body stood 0.15 m from where it had stood before (the account: “the old ‘laundry bath’ marker had drifted into the entry hall — I re-dropped it in front of the boiler”). Nothing used the label until task 34, the first task of day 3, when the day-start reset put the inherited marker 4.1 m off in the new frame. The executor followed its stored route toward it, ended on a spurious “floor 2”, asked for a floor-scope recovery that was refused (“the robot created and placed me on ‘floor 2’, while the whole house map with all my markers is floor 1”), walked in by landmarks and re-placed the label, calling it drifted. From then on the marker was accurate in that day's frame, the preview reported it reachable, and the next two boiler tasks walked straight to it; the account of task 44 reads “the laundry bath one was accurate” (Figure 18).

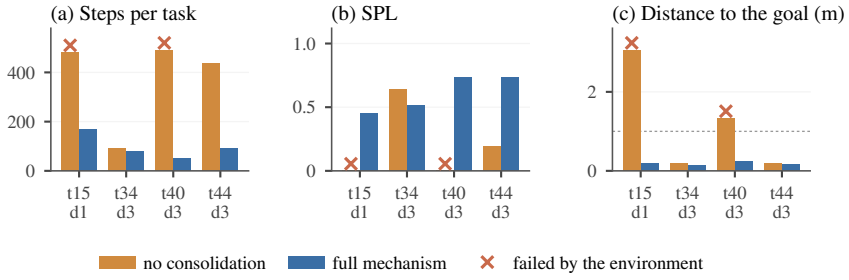


Figure 18: House A, the tasks that placed or used the *laundry bath* marker, in both runs: steps, SPL and distance to the goal by the simulator (the dashed line is the 1 m success radius; a cross marks a failure). The starts coincide at tasks 34 and 40 and lie 1.4 m and 0.1 m apart at tasks 15 and 44; the no-consolidation run recovered once at each of tasks 15, 40 and 44, the full run never.

House A · task 40 · day 3 · full · task record 51 steps, SPL 0.73; the preview reported the marker reachable at 2.5 m

Find the specific boiler shown in the goal image.

From the start I followed the saved route — tiled entry hall, out to the carpeted hallway past the kitchen opening and staircase, then the door on the right into the laundry/bath room. Re-dropped the "laundry bath" marker inside the room, since the old one had drifted.

House A · task 40 · no-consolidation · task record 491 steps, one recovery, failure 1.33 m from the goal

The same goal.

Searched for the boiler from the photo and never found it. I started in the carpeted bedroom next to the laundry door, went through the laundry with the washer and dryer, followed the corridor [...] checked the couch room, the bathroom hall, the red-rug rec room, and the alcove with the chest freezers and fridge.

What it shows. The misplaced marker is not diagnosed when first written. It remains unused for nineteen tasks and is re-placed after a subsequent frame reset (Figure 20a). On tasks 40 and 44 the session then combines the reachable marker with its written landmark route. It takes 51 versus 491 steps on task 40, and 93 versus 436 on task 44, without recovery. On task 34, however, following the stored route west adds 11 steps relative to no-consolidation. The records show why checking current location matters before reusing a previously successful route.

Case 3. Moves the simulator never saw. In House C a calendar hangs on the refrigerator surround and is asked for five times. The same label was placed four times, at positions within 0.5 m of one another by the simulator, while its map coordinates changed with each day's frame; the accounts call the earlier marker *moved*, *misled* and *drifted*, and only one of the three matches the truth; at task 40, 58 steps later, "the existing 'back hall by fridge' one sufficed". Figure 19 shows every task in both runs; the starts coincide within 0.9 m in all three pairs discussed here. The full run re-placed *back hall by fridge* at tasks 17, 35, 54 and 70, at map positions (4.54, −3.53), (1.34, −5.51), (0.01, 0.02) and (−2.12, −3.41), and placed *galley kitchen* at task 42; the no-consolidation run named the place afresh each time (*kitchen*, *kitchen entry*, *kitchen fridge corner*, *fridge passage*, *fridge end wall*, and four labels across the house on its failed task 35). The no-consolidation run found the calendar at task 40 in 283 steps after a first close from the doorway was refused, and at task 54 it overshot in long turn-and-walk batches. At task 42 the repaired marker's own label pulled the full run toward the wrong goal ("trusting the 'back hall by fridge' marker as a shortcut misled me").

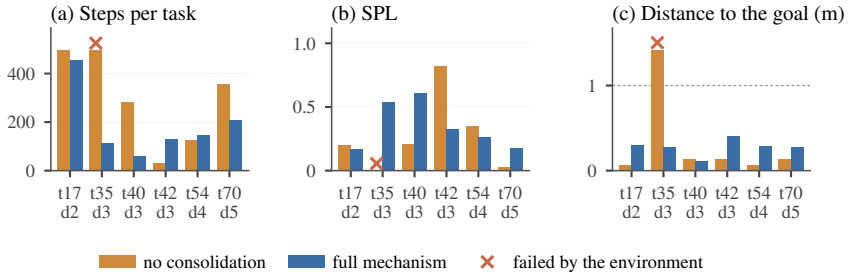


Figure 19: House C, the calendar and the refrigerator beside it, every task in both runs: steps, SPL and distance to the goal by the simulator (dashed line, the 1 m success radius; a cross marks a failure).

House C · task 35 · day 3 · full · task record 113 steps; the marker moved 0.02 m in truth

Find any calendar.

I found it and stood in front of it, the picture panel above the dated grid on the panelled fridge surround where the back hall opens off the kitchen [...] Path previews called every kitchen-side

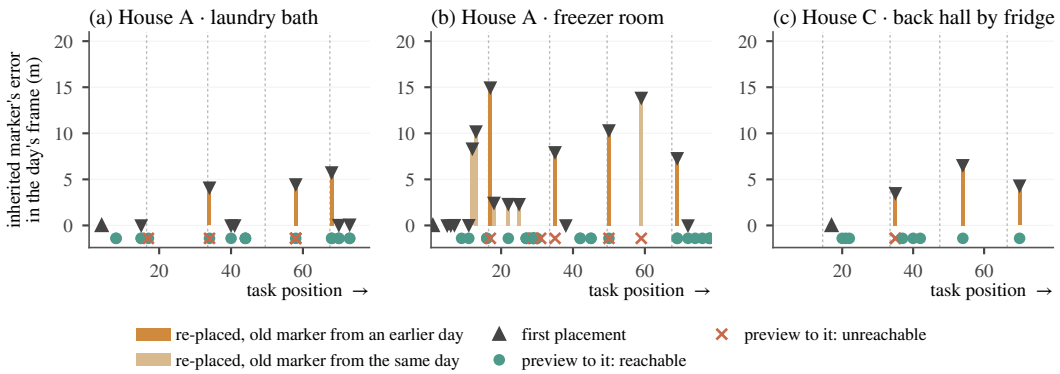


Figure 20: The life of one label in the frame of each day. Bars are the inherited marker's error in the frame of the day at the moment it was re-placed (ochre when the old marker came from an earlier day, pale when from the same day); triangles mark placements; the dots and crosses under the axis are previews aimed at that label, reachable (teal) or refused (red-brown). A marker is accurate within its day and wrong at the start of the next; previews fail until it is re-placed. (a) and (b) House A; (c) House C.

marker unreachable — ignore that. […] Moved the "back hall by fridge" marker to where the calendar is visible.

House C · task 35 · no-consolidation · task record 496 steps, one recovery, failure 1.42 m from the goal
Find any calendar.

Looked for any calendar and did not find one. Starting at the piano corner, I worked through the open living and dining area, into the kitchen, then down the hallway […] Several hallway doors stayed shut; I ran out of steps before opening or reaching them, and white paper on the hall wall turned out not to be a calendar.

What it shows. Re-placement brings a marker into the current coordinate frame, after which it can guide the next visit (58 steps at task 40). At task 35 the session instead reads the calendar skill and four stored photographs while the marker remains unreachable. Task 42 shows the opposite behavior: following the label adds about one hundred steps. These records distinguish useful room recognition from unreliable metric guidance.

Case 4. Narrowing a label over repeated visits. In House C the label *piano corner* was placed on day 1 and re-placed six times. Its map coordinates jumped by 10 to 13 m at each day start; by the simulator the placements lie within 2.6 m of the goal, and the last three narrow from 1.64 to 1.30 to 0.00 m (Figure 21). The one placed at task 72 recorded the executor's own habit of standing back to frame the piano, and that task failed 1.27 m from the goal while the no-consolidation run succeeded in 90 steps; the account blamed a drift that the day's sheet did not contain. Two tasks later the executor re-placed the label at the piano itself and said why; at tasks 74 and 78 both runs succeed, the full run in 136 and 71 steps against 171 and 87.

House C · task 72 · day 5 · full message on arrival, then the task record; the run ends 1.27 m from the goal
Find the piano.

I'm at the piano corner, parked about three paces back and facing it square […] The 'piano corner' marker has been re-dropped here — the old stored marker had indeed drifted, as the notes warned. Ledger: […] Re-dropped the "piano corner" marker.

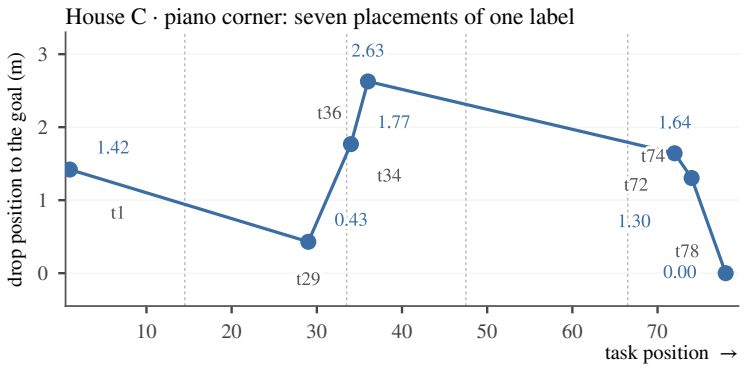


Figure 21: House C, the *piano corner* label: distance from each placement to the goal by the simulator. The map coordinates of the same placements differ by up to 13 m between days; the true positions lie within 2.6 m, and the last three narrow to the goal.

House C · task 78 · day 5 · full message, the task record, then the piano skill as consolidated that evening; the re-placement ends 0.08 m from the goal

Find the specific piano shown in the goal image.

I'm standing right at the piano from the goal photo. Let me correct the drifted marker to the true spot. Ledger: [...] I re-dropped the "piano corner" marker at the true spot, since the old one had drifted and pointed me at empty floor.
skills/find-a-piano.md, day 5: The stored 'piano corner' marker had drifted onto empty floor.

House C · task 72 · no-consolidation · task record 90 steps, success 0.08 m

Find the piano.

I was sent to the piano and reached it: I ended up standing right beside the upright piano and its bench by the window, in the log-walled living room. [...] I dropped markers at the kitchen sink island, the log dining room, and the piano corner.

What it shows. Re-placing a named marker updates its stored location. The final placements approach the goal from 1.64 to 1.30 to 0.00 m over tasks 72, 74, and 78. The position records suggest that an earlier marker represented a viewing position rather than the object; the quoted account itself describes drift and empty floor. The full run fails task 72, which no-consolidation completes, but takes 71 rather than 87 steps on task 78. Across the seven piano tasks it averages 74 steps against 122. Marker updates and note-based recognition accompany this improvement, without guaranteeing a shorter search on every visit.

D.2.2 Recording failure and checking completion (Cases 5 to 7)

Case 5. One sentence, read through a note. In House A the description "wooden dresser with a mirror on it, to the right of the hunting trophy, above the staircase handrail, next to the shelf" is asked three times. The no-consolidation run writes an honest exclusion list each time and only on the third attempt, after 493 steps and a recovery, records that "the dresser was on the basement level all along, near the stairs I started beside", although its own successful accounts of two earlier dresser tasks already contained the answer. At task 56 both runs start inside the freezer room, about one to two metres from the goal, and at task 69 both start beside the boiler (Figure 22). The full run placed no marker on these tasks except *freezer room* at task 69, and its successful end poses at tasks 56, 67 and 69 lie within 0.22 m of one another; the no-consolidation run left three labels on each failed search (*front entry foyer*, *office bedroom*, *basement sitting room* at task 10; *red rug freezer room*, *main floor dining*, *upstairs kitchen* at task 56). The full run reaches it through the journal, a room note and a skill: the day-1 room note carried "a tall wooden dresser with a mirror", the day-3 skill added "Correction from t10: no mirror was visible from this side" after the message of task 43 ("the mirror detail may be stale"), and the day-4 skill made the sentence itself the key.

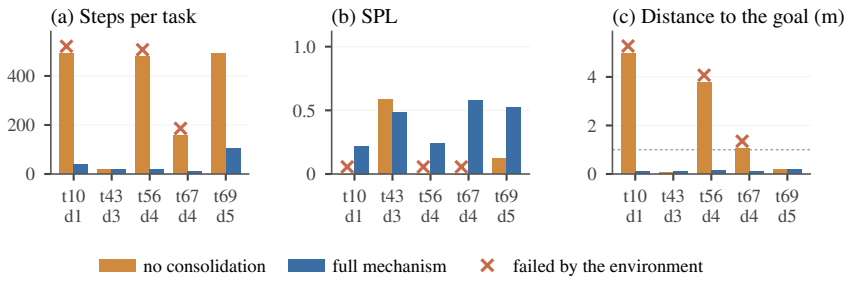


Figure 22: House A, the mirrored dresser, every task that asked for it, in both runs: steps, SPL and distance to the goal by the simulator (dashed line, the 1 m success radius; a cross marks a failure).

House A · skills/find-dresser.md · day-4 version, then the opening messages of tasks 56 and 69 · full
Find this specific dresser: wooden dresser with a mirror on it . . .

Goal texts for it can mislead: [. . .] certified to this dresser (09-22 t7), even though no mirror is visible from this side. Identify by the antler plaque above, the yellow tool shelf beside, the corner freezer past it — never by a mirror.
Task 56: [. . .] (the hunting trophy = antlers is a freezer-room landmark).
Task 69: My notes say this exact goal text certified to the freezer-room dresser.

House A · task 56 · no-consolidation mid-task messages in order, then the verifier; 478 steps, one recovery, ends upstairs 3.78 m from the goal
The same goal.

Found the up-staircase with handrail. Going up. I see what looks like antlers (hunting trophy) on the upper-left wall. I've searched the whole main floor.
Verifier: the only dresser-like furniture seen (frame 7, near the mounted trophy) was passed mid-route and left behind.
Ledger: [. . .] The only antlers hung high on the entry wall, with nothing beneath.

What it shows. The no-consolidation run records useful information but does not reuse it in these later dresser searches. The phrase *above the staircase handrail* sends it upstairs three times. The full run instead reads a note connecting that wording to the freezer-room dresser. An observation at task 43 passes through the task account and consolidation into a skill that later sessions cite. At task 56 both runs start in the same room, and the full session reads the note before moving without first querying the map. The retrieved description resolves the floor and target before route planning begins.

Case 6. Arriving already there, and accepted elsewhere. In House B, the small gap in success rates contrasts with a much larger gain in path efficiency. Retained memory reduces the movement needed to recognize and reach familiar targets (Figure 23): 18% of tasks close within ten steps with the full mechanism, 8% without, and on the first day, before any notes exist, the whole difference is one line of the map, the marker's coordinates equalling the current pose (task 2: "I woke standing on my 'walk-in closet' marker from yesterday, facing the rails", 0 steps, SPL 1.00; the marker was in fact dropped at task 1 the same day). Task 62 is the extreme: the house has two closets with hanging clothes, the no-consolidation run stands in the wrong one, closes after four steps and is accepted from one frame; the full run wakes on the within-day marker of the right one and closes with no step, although the skill note it had read pointed to the wrong closet ("go to the 'walk-in closet' marker . . . expect it may be stale"). The day-4 consolidation later wrote "Do not head for the 09-19 'walk-in closet' marker first."

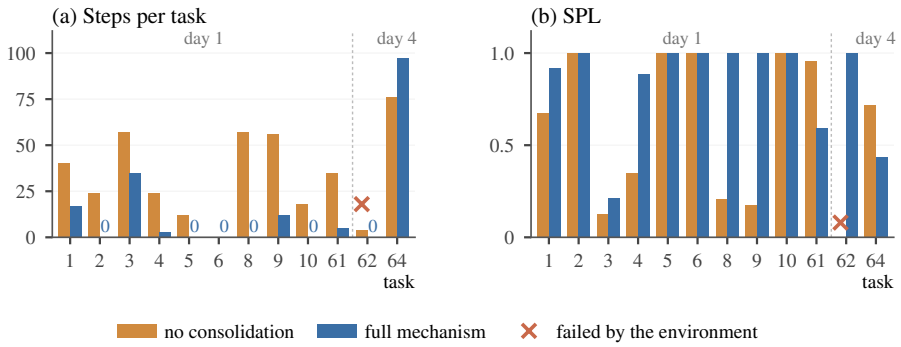


Figure 23: House B, the twelve hanging-clothes tasks given in words, in both runs: the first day's block (tasks 1 to 10; task 7, whose goal is an image, is left out), then tasks 61, 62 and 64 on day 4. **(a)** Steps per task, zero-step closures labelled; the cross marks the no-consolidation run's task 62, closed after four steps in the wrong closet, accepted by the verifier and 9.8 m from the goal. **(b)** SPL. At task 62 the runs start in different closets, about 8.8 m apart, because task 61 diverged.

House B · task 62 · day 4 · full messages in order, then the task record; 0 steps, success 0.19 m

Find the hanging clothes in the closet.

I'm standing exactly on my 'upstairs closet rail' marker from a previous task, and the camera confirms it: coats, shirts and jeans hanging on rails […] This is the same spot that was certified for a hanging-clothes goal before — no movement needed.
Ledger: Worth noting my older skill note pointed at the grey bedroom's walk-in closet instead, which was unnecessary here.

House B · task 62 · no-consolidation messages, the completion claim, the verifier, then the task record; 4 steps, accepted, 9.79 m from the goal

The same goal.

The map is nearly blank and no markers survive. Starting fresh.
Verifier: The final frame is a close-up of densely packed hanging garments on a rack inside the closet, matching the target
Ledger: […] The map came up nearly blank with no surviving markers, so I had no prior route to reuse; nothing else was attempted.

What it shows. In the first day's block, the two runs start within a metre of one another. The full session recognizes a familiar marker and checks its view, avoiding additional translational search. About a third of House B's SPL gain occurs in this block. Task 62 has different starting locations and illustrates a separate issue: the verifier accepts the wrong closet in the no-consolidation run. The full session recognizes its current within-day marker, sets aside a conflicting skill note, and records the conflict for consolidation. Similar overall success rates can therefore conceal substantial differences in search efficiency.

Case 7. A false completion certified into memory. This case shows a mistaken object identification surviving the closing checks and entering memory. In House A, the checker accepted false completions on 10% of tasks without consolidation and 3% with it. The no-consolidation trace below contains a first-request refusal followed by a repeated claim, which the stopping rule permits (Appendix A.6). With memory, one false completion enters the journal as CERTIFIED, is written into the living-room note ("CERTIFIED, 09-20 t16: a wooden chest of drawers with a table lamp on it") and the dresser skill ("the answer to 'dresser next to the armchair' "), and is repeated at zero cost by the next task, which reads no file; on day 5 it pulls task 78 26 steps off course ("that matches my living-room 'dresser next to the armchair' note. Let me approach").

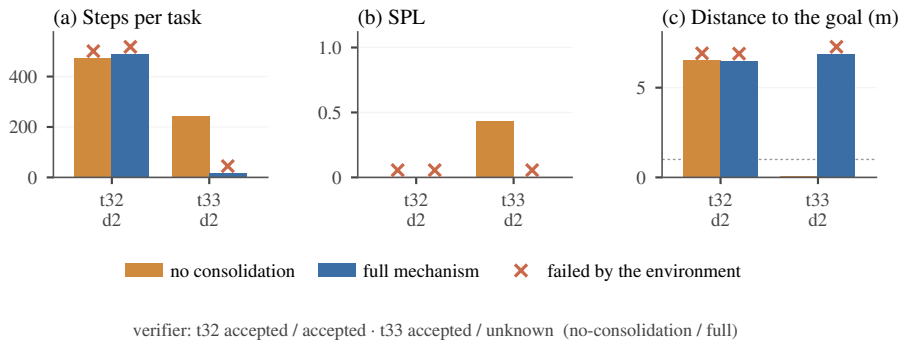


Figure 24: House A, the lamp table beside the armchair. Both runs close task 32 on the same object and task 33 starts where task 32 ended; the verifier's decision on each closure, for the run without consolidation and the full run, is given beneath the panels.

House A · tasks 32 and 33 · day 2 · full completion claim, ledger index entry, the next task's first message, then the verifier

Find this specific dresser: dresser next to the armchair. / Find any dresser.

I am standing beside a wooden chest of drawers with a table lamp on it, directly next to the black leather recliner armchair.
Journal: task 16 · CERTIFIED · [...] markers: living room
Task 33: [...] this is the living room corner from task 16.
Verifier: reads more like a nightstand/end table than a dresser.

House A · task 32 · no-consolidation the pre-stop refusal, a second claim accepted, then the task record; failure 6.51 m

Find this specific dresser: dresser next to the armchair.

Gate: no dresser and no armchair adjacent to it [...]
Ledger: I closed it as complete at the black leather recliner in the living room, with the lamp chest beside it under the framed picture — but the checker disagreed once and I overrode it, so treat that identification as shaky.

What it shows. Here, an incorrect certification lets an object identification be retained as fact in the house notes. The next task's "unknown" verdict records uncertainty about that task but does not automatically retract the earlier account.

D.2.3 Instruments and procedure (Cases 8 and 9)

Case 8. Diagnosing the same fault once, or ten times. In House A the tracker snaps the pose back to the origin after a loss. Both runs recognise that the map is right and the position wrong. The no-consolidation run diagnoses it afresh in ten tasks and prescribes contradictory remedies; the full run carries forward the advice to follow landmarks when the preview is unreliable. The more precise explanations occur in a refused recovery request, Edit, and Write, while the retained skill still says *Markers here drift*. At task 35 a refused edit named "a frame offset from the old floor-1 map", and at task 78 the run rode out a frozen pose on sight: "my pose readout is frozen but the views genuinely change — I'll trust the views".

House A · tasks 29, 34 and 70 · full three correct diagnoses, three refusals; the surviving note is quoted last

Task 29: [...] the map merges both half-levels.
Task 34, recovery request, refused: the robot created and placed me on 'floor 2' [...]

Task 70, refused Write: the marker has NOT drifted. [...] Rule: use the marker to know you have ARRIVED, never to choose a heading.
skills/verify-markers-before-trusting.md, day 5: Even a marker that was accurate all yesterday can go stale.

House A · task 28 · day 2 · full message, the task record, then the rule it became: the first rung of the ladder
Find any boiler.

my pose jumped west while I walked into a previously-unmapped area.
Ledger: SLAM lost itself on the way in and the marker's position looked wrong; recovery changed nothing, so trust the route, not the marker.
skills/ignore-route-preview.md, day 2: If SLAM appears lost, keep walking the landmark chain rather than trying recovery — recovery changed nothing in t12.

House A · task 71 · no-consolidation · task record 429 steps, 204 turns, SPL 0.01; the remedy contradicts the one of task 49
Find the specific boiler shown in the goal image.

I never confirmed I reached it — movement kept stalling, with forward steps colliding and my position repeatedly snapping back to where I started, so the same laundry-nook view came round again and again. [...] Turning in place to find a clear opening mostly did not help.

What it shows. Both runs diagnose recurring tracking faults. The full run reuses an earlier diagnosis through the preview skill derived from task 28. It records seven recovery requests over the deployment against thirteen without consolidation. At task 78 it proceeds using changing camera views despite a frozen pose, whereas the no-consolidation run requests recovery on a similar fault at task 71 and spends 429 steps at SPL 0.01. These are examples from different tasks, not a matched-task effect estimate. They show how a stored procedure changes the response to a recurring instrument problem.

Case 9. A spurious floor, diagnosed but lost from the summary. Only a run with memory meets this fault: markers persist across days, but each day SLAM opens a new frame and the floor counter increments, so every old marker falls in a dead frame and the preview reports it unreachable. In House B the executor wrote the fault up as a bug report the moment it occurred on day 2, and by day 4 had the mechanism (task 52: "the coordinate frame shifted between sessions ... it points at a location in a dead frame"), the cure and "do not request recovery for this". The detailed diagnosis appears in attempted edits that were refused, whereas the saved task account below records only an unreachable marker and the use of landmarks. The later skill note retains a remedy without the explanation of the frame change. Consolidation can read the ledger, task handovers, and existing notes (Appendix A.4); the failure here is loss of that explanation from the retained summary, not a restriction to reading the ledger alone.

House B · task 20 · day 2 · full recovery request at the moment of the fault, the refused journal draft, then the task record that reached the journal
Find any microwave.

After only turning in place, get_map switched from 'floor 1/1' (127 m2, all my markers) to a brand-new 'floor 2/2' with 17.85 m2 and no markers, and preview_path to the kitchen marker went from reachable (12.5 m) to unreachable.
Refused journal draft: The map now shows TWO floor layers [...]
Ledger: My old kitchen marker was unreachable, so I navigated by landmark instead.

House B · task 54 · day 4 · full refused Edit, then the first line of the skill note as consolidated on day 5

WHOLE-FRAME SHIFTS HAPPEN, not just single stale markers. [...] So the map origin, not the furniture, had moved. [...] Do NOT call request_recovery for this — nothing about the robot's position is wrong. Day 5 note: Saved marker positions go stale. [...] Ignore the preview, walk in by landmark, re-drop under the same name where you finish.

What it shows. The executor diagnoses the frame change three times, with increasingly specific explanations, but the retained note keeps the remedy rather than the cause. The recorded layer changes do not correspond to actual storeys. A warning intended for a frame reset consequently becomes a general distrust of route previews, even when the current frame is usable. The traces include extra steps and one recovery at task 20, a failure at task 26, and four to eight preview probes per task on day 4. Summaries need to preserve the conditions under which a remedy applies, not just the remedy itself.

D.2.4 Auditing and consolidating memory (Cases 10 and 11)

Case 10. Closing an open question. The question was not the executor's; the day-1 consolidation of House B set two descriptions of the kitchen refrigerator side by side and asked whether they were one appliance seen from two sides or two ("still open: whether the kitchen holds one fridge or two"; house.md: "possibly one appliance seen from two sides, possibly two"). On day 2 the executor read the question, finished its task, read the overview three times more and went to look at both faces of the appliance. The answer travelled through the ledger, the day summary and a skill without loss. Two days later a very dark goal photograph arrived; the no-consolidation run reasoned from darkness to a basement and searched for a second refrigerator, ending beside the target. The full run placed *kitchen dining doorway* at task 21 and *fridge nook* at task 36; the no-consolidation run placed *kitchen* or *kitchen fridge* on each success and, on its failed task 36, four labels ending at *basement stairhead*. At task 21 the full run spends 36 of its 56 steps answering the open question; at task 36 both runs start from the same place (Figure 25), and the full run's ledger line reads "the old notes were right that you should come at it from the dining-doorway side rather than head-on from the range" (36 steps, SPL 0.70).

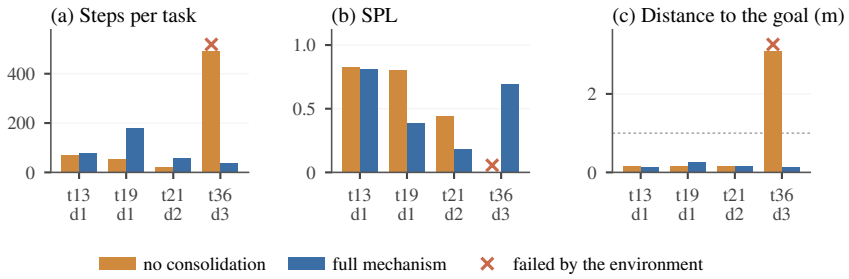


Figure 25: House B, the kitchen refrigerator, every task that asked for it, in both runs: steps, SPL and distance to the goal by the simulator (dashed line, the 1 m success radius; a cross marks a failure). At task 36 the goal is a dark photograph.

House B · task 21 · day 2 · full messages after the task was complete, the refused write, then the task record; house.md was read three more times

Find this specific refrigerator: ...

I'll resolve the long-standing open question in my notes, then close out. [...]
Refused write: the "tall dark fridge by the doorway" (task 13) and the "stainless side-by-side with ice and water dispenser" (task 19) are one appliance, not two. [...]
Ledger: That dark panel and the stainless front are one fridge, not two.

House B · task 36 · no-consolidation messages, then the task record; 490 steps, one recovery, failure 3.08 m

The same goal.

The goal image is very dark [...] a very dark refrigerator in an unlit space — almost certainly a basement, not this bright kitchen.
Ledger: I did not find it. [...] the kitchen off it whose fridge is bright white and clearly not the one pictured [...] The dark fridge is most likely down those basement stairs.

What it shows. Consolidation turns conflicting descriptions into an open question. The executor gathers additional views to resolve it, and the answer survives subsequent summaries. At task 36, it recognizes a dark goal photograph as another view of the known refrigerator, while the no-consolidation run searches for a second appliance. The runs take 36 and 490 steps, respectively. The trace connects the stored answer to goal interpretation, but does not partition this 454-step difference between the note, the map, and the rest of the retained state.

Case 11. Overturned rules, and a route nobody read. Each of the seven skill notes of House A has a version history, and each overturning can be asked where its evidence came from. The three driven by the executor's own observations (the carpet stopping rule, the mirror's visibility, the bare-lid criterion) were right; the one driven by a verifier verdict was wrong, since after one "unknown" on a goal the simulator had certified three times the day-5 consolidation reversed the rule and now sends that wording to the other floor; the two driven by the harness's own artefacts (a "dead end", a "drift") kept a right prescription with a wrong reason. The no-consolidation records contain useful experience that later sessions do not reuse in these searches: the route to the boiler, written in full at task 8, was never read, and the three later boiler tasks each spent their whole budget on a fresh hypothesis (task 15: "the storage space past the workbench corner is the one utility nook I never entered"; task 58: "the ceiling ductwork runs toward it, so the boiler is likely behind it"). Over the nine boiler tasks the no-consolidation run succeeds on 89% at 274 steps per task, the full run on all nine at 107.

House A · skills/find-dresser.md · day-5 version, and the verifier verdict at task 69 that drove it · full the simulator had confirmed the freezer-room dresser three times

Certified once for exactly that wording (09-22 t7); no mirror is visible on it, and a repeat attempt on 09-23 t2 went unresolved. If the text names both a mirror and the trophy, prefer the window-bedroom chest and note the conflict.
Verifier, task 69, unknown: [...] but no mirror or recognizable full dresser is visible.

House A · task 65 · day 4 · full the bare-lid rule as read, the messages, the task record, then the day-4 consolidation

Find the specific freezer shown in the goal image.

Skill, day 3: [...] the certified answer was the right-hand one, not the corner one.
Close-up lid [...] my notes say the right-hand freezer. [...] that matches the corner freezer under the yellow shelf.
Ledger: My "bare lid means right-hand freezer" rule nearly misled me; the corner and rust spots decided it.
Day-4 consolidation: [...] treat that rule as doubtful and let rust spots and the corner decide.

House A · task 8 · no-consolidation · task record a complete route not reused in the later boiler searches

Find this specific boiler: gas boiler ... below the heater and slightly to the right ...

Found it and stood beside it — the upright cylinder with its gas valve, under the furnace ducting, next to the washer and dryer. Route went from the freezer alcove where I started, down the hall past a doorway, then right into the laundry utility room at the end.

What it shows. Source references make the revisions auditable. In these examples, direct observations correct misleading rules, while an uncertain verifier output can cause consolidation to replace a useful rule with an incorrect one. The no-consolidation run also writes a complete boiler route, but subsequent boiler searches do not reuse it. Across the boiler tasks, the full run combines a remembered landmark route with current markers and averages 107 steps rather than 274. This is a system-level contrast, not a measured step saving attributable to one unread sentence.

D.3 When stored experience is declined (Cases 12 to 14)

The following cases illustrate three forms of selective reuse, each with a no-consolidation counterpart: a route declined on a prior from the notes (Case 12), a rule declined because the goal contradicts it (Case 13), and a bad entry caught by its consequences and repaired from the journal (Case 14). The control retains maps and task records but has no consolidated house notes. Its quoted doubts concern pose and preview within the task.

Case 12. Declining a correct marker and following a remembered route. At task 70 of House A the marker is accurate to 0.06 m and the preview says reachable; the executor declines the straight line on the prior in its notes that this marker "has drifted before", walks the landmark chain, and corrects its reason on arrival (the refused write: "my old note #8 claimed the 'laundry bath' marker had drifted. It had *not*"); the ledger line keeps the recantation ("the marker is true, but only the hallway route gets you there"). Both runs start beside the freezer-room dresser and both reach the boiler within 0.2 m; the full run takes 94 steps (SPL 0.80) and no recovery, the no-consolidation run 317 steps, one recovery and SPL 0.20.

House A · task 70 · day 5 · full mid-task messages in order, after the notes were read and the preview reported the marker reachable at 4.7 m

Find any boiler.

Reachable at 4.7 m — but my notes warn this marker has drifted before. [...] The marker points straight into the dresser wall — it has drifted, exactly as my notes warned. Falling back on the landmark route. [...] The yellow slatted pine door is on my left — my key landmark. [...]
On arrival: The marker was accurate all along (0.36 m away) — my straight-line bearings were cutting through walls.

House A · task 70 · no-consolidation mid-task message after a tracking loss, then the task record; 317 steps, one recovery, SPL 0.20

Find any boiler.

My reported position is clearly wrong — the map says I'm at the rec room origin, but I'm in the west plaid-couch room.
Ledger: [...] Early on I lost my bearings against a wall near the plaid-couch room and wasted turns re-orienting before the hallway route worked.

What it shows. The session initially distrusts a correct marker because an older note warns about drift and the direct bearing crosses a wall. It follows the remembered landmark route and revises its diagnosis on arrival. Both runs succeed, in 94 and 317 steps, respectively. The trace shows that a useful route can coexist with an incorrect explanation for choosing it. It does not establish that rejecting the marker has zero cost relative to using it with a valid route.

Case 13. Declining a rule the goal photograph contradicts. Consolidation can turn "I did not need to" into "do not". In House C the account of task 56 said two floor plants in a corner were too tight to frame and that it had chosen another; the day-4 skill made this an *Avoid*. The goal photograph of task 67 is one of those plants. Both runs start at the episode's reset point; the full run succeeds in 123 steps (SPL 0.75, 0.17 m from the plant); the no-consolidation run fails.

House C · task 56 · day 4 · full · task record the account that the consolidation turned into a prohibition
Find any decorative plant.

I started in the treadle desk nook in the northeast corner; its two floor-level plants — a dried arrangement in a wooden bucket and a low leafy pot by the picture window — sat too tight in the corner to frame, so I left them. I crossed [...] to the dining window nook.

House C · skills/find-a-decorative-plant.md · day-4 version, then tasks 67 and 73, then the day-5 version · full
Find the specific decorative plant shown in the goal image.

Avoid: the two floor-level plants [...] sit too tight in the corner to frame — 09-22 t9 left them [...]
Task 67: The old note calling that corner too tight to frame was wrong; standing a step off the desk frames it fine.
Task 73: [...] I could not save that correction; the note-writing tools were disabled, so the stale advice is still on file.
Day 5: The old warning [...] is dead — stop about a step short of the desk and it frames (09-23 t7, borne out by t9/t13).

House C · task 67 · no-consolidation · task record 470 steps, one recovery; closed 7.1 m from the goal, verdict unknown
The same goal.

The matching plant sat on a wooden chest beside a yellow lamp in the bright window parlor past the dining room, and I closed there. [...] I wasted many steps circling the kitchen and hearth area before trying the far south rooms.

What it shows. The session uses the stored room description and photograph to recognize the corner, but rejects the instruction to avoid it because the new goal explicitly asks for that plant. It records this conflict in the task account, and consolidation revises the rule. The important behavior is selective reuse: the room information remains useful even when the associated instruction is inappropriate for the current goal.

Case 14. A note that misleads, and a journal that recovers. On day 4 the notes of House B mention one mirror, a black-framed one in the entry hall. The goal image was returned at the start, again before a recovery and once more right after it, yet the executor followed the note downstairs and then wrote that it had not seen the photograph in this session; only after a fourth call did it look at the goal ("re-reading the actual goal photograph instead of trusting my notes"), see that it was not that mirror, and find in the same day's journal the failed attempt at the same goal five tasks earlier, which left an exact lead. Memory was both the cause and the cure. The starts differ by 1.2 m, both in the same guest room.

House B · task 60 · day 4 · full · task record, as it entered the journal 500 steps, one recovery, failure 1.21 m
Find the specific mirror shown in the goal image.

I hunted for a wooden-framed mirror from a photo and did not find it. I woke upstairs on the bedroom floor, not the main floor my old notes describe [...] Past the master's door a hardwood corridor runs by a descending staircase to a teal bed guest room, which I marked; I ran out of steps before seeing the wall behind its open door. Try that wall first.

House B · task 65 · day 4 · full recovery request at step 241, then the messages that followed
The same goal, five tasks later the same day.

I still haven't actually seen the goal photograph in this session — I've been reasoning from notes. [...] the real goal image: a pine-framed square mirror on a slate blue-grey wall [...] not the entry hall (black frame, cream walls).
Reading the journal: Task 9 was this same mirror goal, and it left a precise lead: '[...] I ran out of steps before seeing the wall behind its open door.' [...] I need to go back up.

House B · task 65 · no-consolidation · task record 156 steps, SPL 0.38; it never went downstairs
The same goal.

I found it in the bedroom with the red bookshelf, hanging on the blue-grey wall next to a small framed print, opposite the open door to the hall — I stopped facing it and closed complete. [...] Wandering the first room cost steps for nothing.

What it shows. The session treats an incomplete inventory as exhaustive, following the only mirror in its notes despite having received the goal image. After recovery it re-examines that image and retrieves task 60's unfinished-search lead from the journal. This lead directs the search that ends in a 468-step success. The full run is nevertheless less efficient than no-consolidation (SPL 0.08 versus 0.38). Detailed failure records help repair a search misled by a summary, without showing that another failure was inevitable in their absence.

How often, and was it right. Every statement in the full run's accounts, messages, claims, recovery requests and refused writes that declares an inherited item wrong, unreliable or inapplicable was counted and checked against the simulator (Figure 26). This analysis concerns judgements recorded by the full runs, rather than the availability of memory in the control. In all three houses a judgement is right with the wrong cause when the offset or unusability it names is real but blamed on drift or a wrong floor. In House A, of seventeen doubts of an inherited marker, nine were right, four right in effect with the wrong cause and four wrong; of eight doubts of a note's rule, route or target, three were right, three wrong and two cannot be checked. In House B all 21 checkable judgements hold in effect, two with the wrong cause; in House C, of 17 checkable judgements (19 in all), eight hold, seven hold with the wrong cause and two are contradicted (tasks 36 and 72).

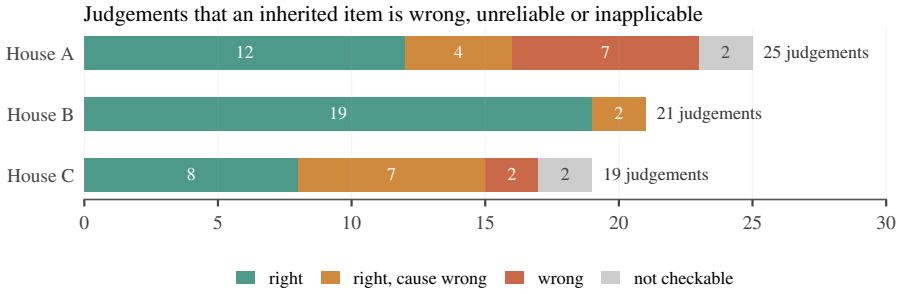


Figure 26: Judgements in the full run that an inherited item is wrong, unreliable or inapplicable, checked against the simulator. Counts describe the full runs' recorded judgements.

Declining an inherited coordinate is almost always right in effect and almost never right about the cause: every account of a day-start frame reset calls it drift or a wrong floor. Three wrong judgements doubted an accurate marker (House A, tasks 3, 40 and 70) and one cost a task (House A, task 27). Only written judgements are counted; a memory silently not used leaves no account, and the simulator checks position only.

Memory use across the analyzed cases. For each of the forty-eight cases, we manually coded the recorded memory effect as positive, negative or mixed, or showing no difference. We also labeled the records involved, allowing multiple labels for notes, within-day markers, and the ledger, with a

separate label for harness or judge involvement. Figure 27 summarizes these case-level interpretations. They describe the analyzed corpus, not independent ablation effects or rates over all tasks.

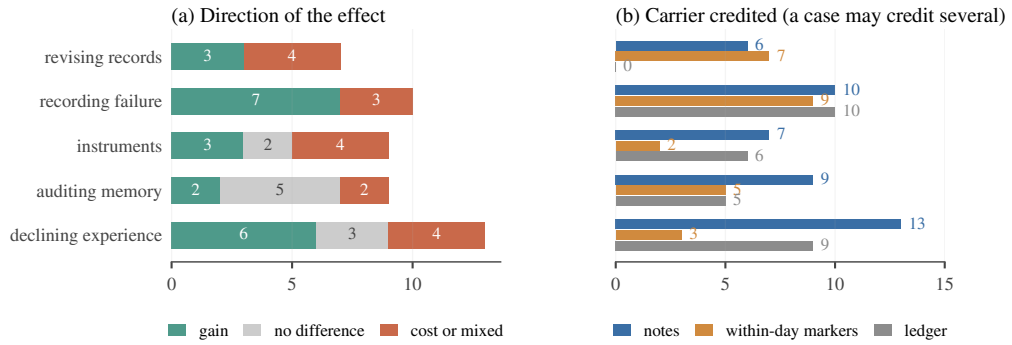


Figure 27: Manual coding of the forty-eight analyzed cases, including the fourteen presented here. (a) The direction of the recorded memory effect. (b) Memory sources involved in the case assessment, with multiple labels allowed. Counts describe cases, not task-level component contributions.

What the comparison shows. Four kinds of note content guide the recorded decisions: a rule that maps a goal sentence to an object (Cases 5 and 10), a room's inventory and photographs (Cases 3 and 13), a landmark chain in a skill (Cases 11 and 12), and a spot recorded as certified, which outweighed a wrong skill (Case 6). These textual cues work alongside the map and the robot's current location. Within-day markers carry one kind: waking on the goal, or a marker reachable in the day's frame (Cases 2, 4 and 6). The ledger is the carrier that turns a day's observation into the next day's note (Cases 5, 10 and 13), and also the channel through which a wrong observation becomes a note (Cases 7 and 11). The records also show five risks of reusing memory: inherited coordinates after a day-start frame reset (Cases 1, 2 and 3), a spurious floor (Cases 8 and 9), an accepted false completion written as a certified fact (Case 7), a gap in the notes or a failed account's lead followed into a dead end (Case 14), and a prohibition generalized from a "no need" (Case 13). These behaviors help interpret the different profiles in Figure 12, without assigning each profile to a single memory component.

D.4 Archive construction and access

During deployment, long-term memory carries task experience across simulated days through house overviews, room notes, and skill files. The overview provides an entry point to this collection rather than replacing its detailed records. The executor reads these records through file tools. Task accounts preserve new experience as it is recorded, while a separate consolidation session summarizes that experience for later use. The 36-house comparison in Section 4.4 holds map and task-record retention fixed and switches only this consolidation session off. Pooled s-SR rises from 72.8% to 80.5% with consolidation, a paired gain of 7.7 points (95% CI $[+6.3, +9.1]$), while SPL rises from 36.5 to 44.3, a gain of 7.8 points (95% CI $[+6.5, +9.2]$). Thus house notes improve later navigation even when maps and detailed task records remain available. The cases in Appendix D.2 trace this process from observations to later decisions, including a corrected dresser description, a refrigerator identity resolved across views, and a revised instruction about a plant's location. The archive preserves successful routes, failed searches, and open questions alongside their sources so that later sessions can revisit the evidence behind a note.

E Limitations and Future Directions

The deployment findings suggest several directions for extending NavHarness, from faster local control to memory access and changing environments.

Scope of the deployment evidence. The continuous-deployment comparison covers finite task sequences in multiple static simulated houses. It isolates run-end consolidation while holding map and task-record retention, recovery, and completion checks fixed. The benchmark interventions

separately test map and ledger reuse. Paired goal sequences need not produce identical starting poses after earlier trajectories diverge. The route analysis therefore additionally examines tasks completed by both runs from nearby starts. The case studies explain recorded decisions and memory revisions, rather than estimating how frequently every mechanism would occur in a wider population of homes. Extending the study to changing environments and longer deployments remains important.

E.1 Ownership of the control loop

Navigation is paced by repeated model decisions. In deployments A–C, the full system takes 38–45 model turns per task on average. A turn can issue tool calls containing several primitive actions, so model turns, tool calls, and movement steps are different quantities. A stronger and faster embodied policy could instead own the local rollout while a larger model maintains goals, memory, and task boundaries. This requires a local controller that can both navigate reliably and return observations that support subsequent reasoning and verification. Separating the executor from the harness allows it to be replaced without changing how persistent state and its sources are recorded.

E.2 Execution speed and physical time

Repeated model inference and tool interaction introduce substantial latency even when actions are batched. Task boundaries, recovery, and memory consolidation are event-triggered, so their scheduling can also incorporate elapsed time without changing the archive format.

E.3 Scope of real-world evaluation

The evaluations reported here use simulation. Sensor-based SLAM removes dependence on simulator pose, but physical deployment also requires robust communication, low-latency obstacle avoidance, and independently measured trajectories. Collecting and checking a full day's real-world observations and outcomes remains a substantial evaluation burden.

E.4 Spatial representation

The current spatial state uses an ORB-SLAM3 map, named markers, and small place-photo sets. It cannot represent object motion, temporal change, or the state of manipulable objects, and its metric frame is never re-anchored to ground truth, so drift and coordinate misalignment can persist. Moreover, tracker loop-closure corrections are not propagated into previously written occupancy cells. Recognizing a familiar place and recovering its metric pose are distinct requirements. Place recognition supports reuse of room-level knowledge, whereas coordinate-based navigation also requires a reliable alignment with the stored map. Richer backends could provide dense 3D geometry or change-aware 4D memory, as in SuperMap (Zhao et al., 2026), which tracks object appearance, disappearance, and relocation. Integrating these representations would require adapting the map-query and update tools. The harness specifies which state to retain, when to update it, and how to record its source without requiring a particular map representation.

E.5 Archive access over longer deployments

The archive operates throughout the multi-tour deployments, with refreshed house overviews entering later sessions. Consolidation organizes task experience into house, room, and skill notes with source references. These overviews form part of the navigator's subsequent input. The consolidation comparison shows gains beyond retained maps and task records. Longer deployments make the access policy increasingly important, including when to retrieve a detailed account and when to revise an older belief. The current design exposes the house overview and supports on-demand recall of individual notes. As experience accumulates, retrieval must remain selective enough to keep relevant information accessible without overloading the active session. Handling stale or conflicting accounts is equally important, as the deployment cases show.

E.6 Static environments and deployment horizon

Our evaluations use static scanned environments from GOAT-Bench and IR2R-CE. The extended deployments test memory reuse across finite task sequences and prescribed relocations, but do not evaluate adaptation to moved objects, changing layouts, or people. Although agents encounter incomplete or incorrect records, revising these records does not establish that the system can detect and track changes in the environment itself. Evaluating memory revision under environmental change and over longer deployments remains future work.

E.7 Capacity of smaller models

With Qwen3.8-27B held fixed, the harness raises GOAT task success from 41.4% to 71.7%. This supports its value beyond the largest frontier models, while still requiring an executor capable of multi-round multimodal navigation. In our tests, Qwen3.5-4B and Qwen3.5-9B struggled with multi-round multimodal navigation and had low task-completion rates. Using these models as the common backbone would test a reasoning-core limitation as well as the harness. We therefore compare harness and independent-session configurations within each of the four capable executors. Adapting the interface to smaller models remains future work.

Process records for future training. The runs align goals, frames, execution records, claims, checking verdicts, and memory updates. After independent validation of the fallible judge labels, these records could support long-horizon post-training or process supervision. The current study does not update model parameters.

E.8 Placement of supervision

Both benchmark evaluations and continuous deployments check the first stopping request and assess the completed rollout. Other designs could inspect every fixed number of steps or monitor an executor continuously. Frequent checks may interrupt useful work and multiply judge calls, while continuous supervision requires a full-duplex interface and a low-latency verifier, neither of which the present transports provide. The choice also interacts with loop ownership: when a local policy owns the rollout, the boundary at which control returns is the natural place for a check, and the three schedules become three trigger rules for the same pre-stop verification mechanism. The verification schedule can be changed independently of the executor and the rules for retaining memory.

E.9 Simulation and model-based judgement

A benchmark scene identifier selects the house archive, while a real robot would need cross-house place recognition. Finally, the learned judge can hallucinate and shares model-family biases with the acting policy. Although its claimed-completion accuracy exceeds accept-all, the four-view checker still accepts false claims on 5.2% and rejects true claims on 4.1% of claimed completions (Appendix C.5). Goal verification also leaves spatial and procedural claims in memory unvalidated. Reported navigation scores come from environment truth. We map failed or unparsable judgements to UNRESOLVED, and avoid treating the continuous-deployment results as benchmark leaderboard scores. Real-robot operation, place recognition, and independently trained judges remain necessary tests.