SITUATING MEMORY FOR EMBODIED NAVIGATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Memory-based navigation requires an agent to relate stored environmental knowledge to its current observation. When navigation fails, however, task success alone cannot distinguish missing knowledge from errors in using it. We introduce **SCOPE**, a controlled evaluation framework that examines this distinction across **128** navigation tasks in **64** indoor scenes. We vary frozen scene memory and online episode memory while holding the navigation policy and execution pipeline fixed, then replace native scene memory with audited ground-truth scene knowledge. The strongest ground-truth condition succeeds on 93 of 128 tasks. Among its 35 failures, trace-based audits attribute 24 to current localization, goal localization, or egocentric correspondence. One-shot corrections at diagnosed checkpoints recover 15 episodes, compared with five under matched sham interventions, supporting the behavioral relevance of the identified errors. Motivated by these observations, we introduce Experience Skeleton Memory (ESM), which uses simulator-provided poses to organize executed trajectories and express historical observations relative to the current view. With the same ground-truth scene memory, ESM achieves 85.94% success on the full suite, compared with 72.66% for the original structured episode memory. These results identify correspondence between memory and current observations as a navigation bottleneck and demonstrate the benefit of explicitly maintaining this correspondence under accurate pose information.

1 INTRODUCTION

An embodied agent navigating beyond its current field of view must retain information about places it has encountered and use that information to choose where to move. Navigation systems support this process through visual histories, semantic maps, and structured representations of the environment (Wani et al., 2020; Huang et al., 2023; Liu et al., 2024; Werby et al., 2024). Even an accurate record of the environment, however, leaves part of the navigation problem unresolved. Knowing that a corridor connects to a kitchen does not establish which visible doorway leads there, or whether the agent has already passed it. Using memory therefore requires the agent to relate stored spatial knowledge to its changing observation and interaction history.

Prior work has examined both memory-based embodied tasks and mechanisms for aligning memory with egocentric perception. FindingDory evaluates recall and reasoning over historical experience in embodied navigation (Yadav et al., 2025), while Mem2Ego projects cues from global memory into the agent's current view (Zhang et al., 2025). These studies motivate a more specific empirical question: when a memory-equipped agent fails, is the relevant environmental knowledge missing, or does the agent fail to use knowledge that is already available? Aggregate navigation success cannot resolve this distinction. An incomplete scene description, an incorrect belief about the current location, and a mistaken association between a remembered place and a visible entrance can all produce an unsuccessful trajectory. Distinguishing these cases requires interventions on the supplied knowledge and examination of decisions within the trajectory.

We introduce **SCOPE** to study these failure modes under a closed-loop navigation protocol. We evaluate 128 tasks across 64 HM3D scenes, covering direct navigation, target disambiguation, search and collection, and multi-stage navigation. We distinguish scene memory, acquired beforehand and fixed during execution, from episode memory, maintained online from observations and interactions. In the primary comparisons, we vary these memories while keeping the task, starting pose, policy, grounding module, executor, and decision budget fixed. We replace native scene memory with audited ground-truth scene knowledge to examine failures that persist when more complete environmental

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

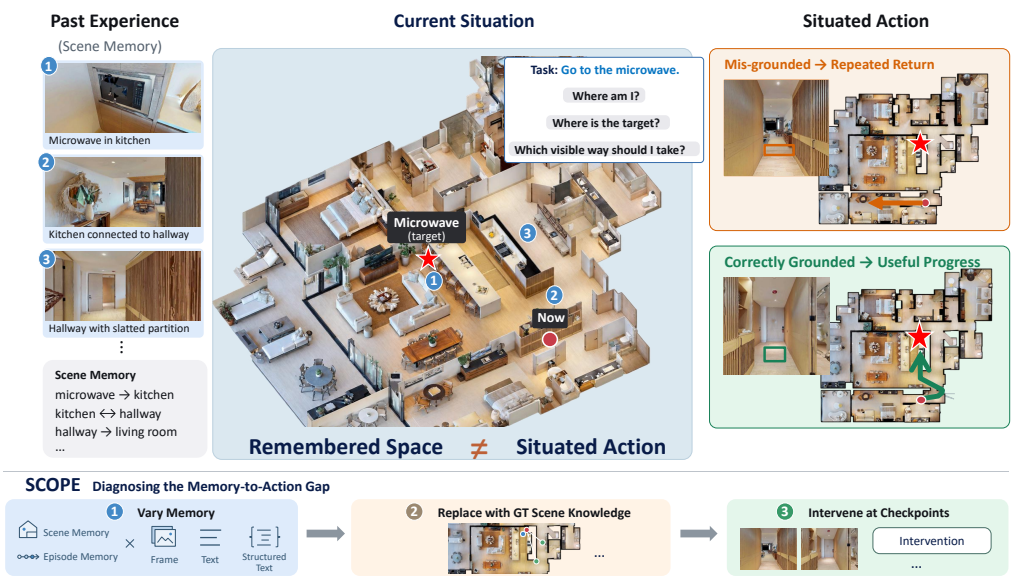

Figure 1: **Overview of the SCOPE evaluation protocol.** We compare scene and episode memory under a shared navigation system, replace native scene memory with ground-truth scene knowledge, and audit the remaining failures. Matched checkpoint interventions test whether correcting an identified local error changes the episode outcome.

information is available. This replacement does not supply the policy with its current ground-truth location, observation-to-memory correspondence, or a route to the goal.

Scene memory provides substantial gains in this setting: a textual summary of pre-task experience raises success from 42.19% to 65.63%. The benefits of episode memory are less consistent across the interfaces. Even with ground-truth scene knowledge, the strongest condition succeeds on only 93 of 128 tasks. We audit the remaining 35 trajectories using decision traces, observations, and independently verified scene evidence. Of these failures, 24 are attributed to current localization, goal localization, or egocentric correspondence; the last is the largest category, with 12 cases. To test whether the identified errors affect subsequent behavior, we resume each episode from a diagnosed checkpoint and apply a single local correction. Targeted corrections recover 15 episodes, compared with five under matched sham interventions. This comparison supports the relevance of the diagnosed errors without treating the policy's reported rationale as a direct account of its internal reasoning.

The diagnosis motivates **Experience Skeleton Memory (ESM)**, which preserves the geometry of executed experience separately from the policy's revisable semantic interpretations. ESM organizes traversed routes into a skeleton, attaches historical observations to their acquisition locations, and renders that experience relative to the camera view. It uses simulator-provided metric poses for these operations; semantic interpretation and task-level action selection remain with the policy. ESM is a geometric memory intervention with additional pose information. With the same ground-truth scene memory, it succeeds on 110 of 128 tasks, recovering 22 of the 35 failures while retaining success on 88 of the 93 tasks. This demonstrates the value of explicit geometric correspondence in this setting.

SCOPE contributes a protocol for separating scene-knowledge limitations from errors in memory use, an audit of residual failures supported by checkpoint interventions, and a geometric memory mechanism motivated by that audit. These contributions help examine not only what information an agent retains, but where its use of that information fails during navigation.

2 Related Work

Environmental Memory for Embodied Navigation. Embodied navigation systems use persistent representations to reason beyond the current observation, including visual histories, textual summaries, topological or semantic maps, scene graphs, and learned spatiotemporal memories (Fang et al., 2019; Chaplot et al., 2020; Zhang et al., 2024; Gu et al., 2024; Anwar et al., 2025; Yang et al., 2025b; Hu

et al., 2026). Recent work further studies how such memories can be retrieved, compressed, and updated over long-horizon or dynamic interaction (Anwar et al., 2025; Liu et al., 2025; Hu et al., 2026). Rather than proposing a new memory architecture, SCOPE treats scene memory and persistent task-time state as controlled variables and studies how they affect closed-loop spatial behavior under a fixed navigation system.

From Past Experience to Navigation Action. FindingDory separates experience collection from downstream tasks and evaluates whether vision-language models can retrieve task-relevant historical locations from long-horizon embodied experience (Yadav et al., 2025). SCOPE extends this question into closed-loop execution: once navigation begins, can a policy continue to relate historical environmental knowledge to its changing current situation? Mem2Ego instead develops an explicit system for converting persistent global memory into egocentric navigation cues using landmarks, exploration frontiers, visitation history, and camera geometry (Zhang et al., 2025). In its primary benchmark conditions, SCOPE fixes the navigation interface and leaves this memory-to-action conversion to the evaluated policy itself. ESM is introduced later as a targeted intervention that explicitly stabilizes the geometric correspondence of previously executed experience.

Evaluating and Diagnosing Memory Use. Recent embodied evaluations increasingly analyze intermediate spatial states and failure sources beyond final task success (Khanna et al., 2024; Majumdar et al., 2024; Yang et al., 2025a; Zhang et al., 2026; Liao et al., 2026; Zhao et al., 2026). Prior work has examined navigation bottlenecks using ground-truth semantic information, compared predicted spatial representations with reference states, studied spatial beliefs during exploration and environmental change, and separated localization from downstream execution. SCOPE focuses specifically on failures along the path from environmental memory to current action: by controlling available memory, supplementing scene knowledge, and intervening at reproducible failure checkpoints, we distinguish limitations in environmental knowledge from failures of situated state maintenance and memory-guided action.

3 SCOPE BENCHMARK

SCOPE defines a controlled closed-loop navigation setting in which the task, starting pose, policy, observation interface, grounding and execution pipeline, evaluation criteria, and high-level decision budget are held fixed, while the memory available to the policy is varied. This enables paired comparisons of how different memory conditions affect behavior on the same navigation problem; the specific memory conditions and diagnostic interventions are introduced in Section 4.

3.1 CONTROLLED MEMORY EVALUATION SETTING

We distinguish two timescales of memory. *Scene memory* contains task-independent environmental experience acquired before navigation and remains frozen during execution. *Episode memory* is accumulated from observations, actions, and interaction history during the current task and may evolve through closed-loop interaction.

At decision step t, the policy receives

$$C_t = \{I_\tau, o_t, q_t, f_t\}, \qquad a_t \sim \pi_\theta(C_t, M_t), \tag{1}$$

where I_τ is the task instruction, o_t the current egocentric observation, and M_t the scene and episode memory permitted by the current condition. The shared task state q_t contains policy-generated bookkeeping for tasks requiring collection, multiple stages, or return, while f_t provides one-step feedback from the previous decision; neither provides evaluator-verified location or task progress, nor the correct next action. In policy-maintained conditions, memory updates are generated from the agent's own observations and interaction evidence, without evaluator-provided ground-truth localization or task-completion labels.

Thus, SCOPE evaluates how environmental knowledge and task-time experience are maintained and used inside the same closed-loop decision process.

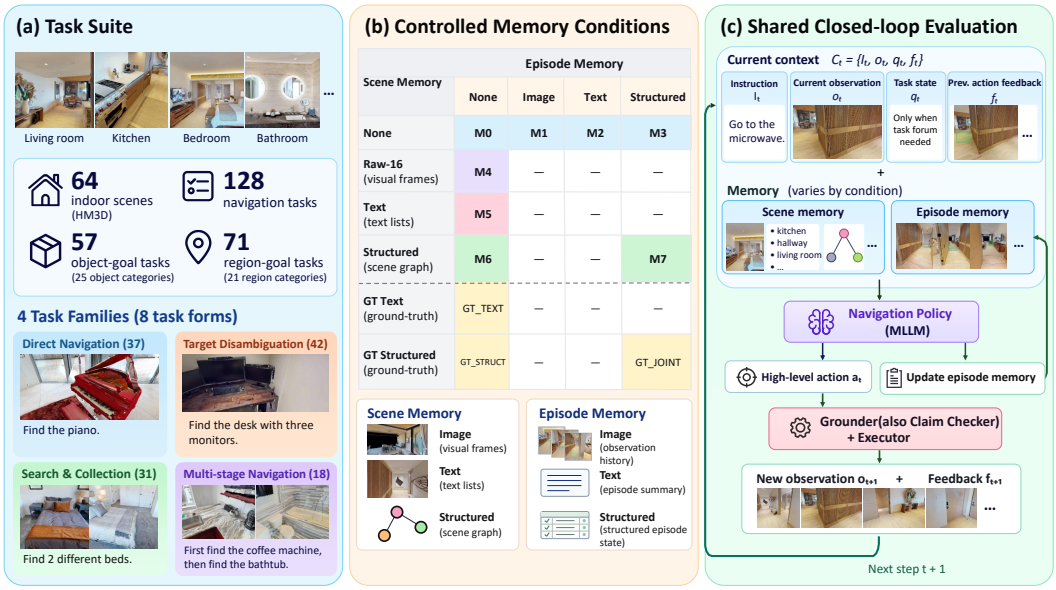

Figure 2: **Overview of the SCOPE benchmark. (a)** We construct 128 navigation tasks across 64 HM3D scenes, covering four task families and eight task forms. **(b)** We control memory at two timescales: task-independent *scene memory* and task-time *episode memory*, and introduce GT scene knowledge as a diagnostic intervention. **(c)** All conditions share the same policy, observation interface, grounding and execution pipeline, and evaluation protocol; only the memory available to the policy is varied.

3.2 Task Suite and Diversity

We construct 128 navigation tasks across 64 HM3D indoor scenes (Ramakrishnan et al., 2021), with two different task forms per scene. The suite includes 57 object-goal and 71 region-goal tasks, spanning 25 object categories and 21 region categories. Scenes contain 5–24 annotated regions, with a median of 13.

We organize the benchmark into four task families: *Direct Navigation*, *Target Disambiguation*, *Search and Collection*, and *Multi-stage Navigation*. Together they cover eight concrete forms: object and region navigation, visual-attribute and spatial-relation disambiguation, candidate-region search, collection, multi-goal navigation, and return/revisit. These tasks require more than recognizing a visible target: agents may need to search among plausible regions, distinguish repeated targets, resolve ambiguous entrances, traverse cross-region or cross-floor transitions, and maintain progress across multiple goals.

Tasks use difficulty-dependent high-level decision budgets of 30, 45, or 60 steps, held fixed across memory conditions. Task-level reference routes and evaluator-side task annotations are used for task construction and evaluation, but are not exposed to the policy as route or action guidance.

3.3 Shared Closed-loop Agent and Evaluation

All native and GT-knowledge conditions use the same closed-loop agent and execution pipeline. The primary experiments use GPT-5.6 Terra as the high-level navigation policy and GPT-5.6 Terra as the visual grounding module. For cross-policy validation, we replace only the high-level navigation policy with GPT-6 Luna, while retaining the same GPT-5.6 Terra visual grounder and execution stack.

At each high-level decision, the policy receives four egocentric RGB views and either selects a visible local target for the next movement or issues a task-completion claim. A shared visual grounding module maps the policy-specified target to a local image-space goal from the current observation, and a fixed local executor carries out the movement using RGB-D geometry and navmesh-based local

planning. Task-level spatial reasoning, including current localization, memory correspondence, and high-level action selection, remains with the navigation policy.

For process-level analysis, the closed-loop interface records a structured decision trace at each step, including the policy's reported current-region belief, goal belief, selected high-level action and rationale, together with the grounding result and execution feedback. These logged signals support the failure analysis in Section 5.3.

We use strict Success Rate (SR) as the primary metric: a task is counted as successful only when the policy issues a completion claim satisfying the target, ordering, and instance constraints. Following the standard oracle-success notion in VLN (Anderson et al., 2018b), we additionally report Oracle Success Rate (OSR), which records whether the executed trajectory ever satisfies the task's spatial requirements even if the policy does not issue a valid completion claim. We also report Success weighted by Path Length (SPL) for navigation efficiency (Anderson et al., 2018a). All metrics are averaged over the 128 tasks.

4 Controlled Memory Conditions and Interventions

The primary SCOPE benchmark varies the memory available to the same closed-loop navigation policy while keeping the task, observation interface, grounding and execution pipeline, and evaluation fixed. We first compare native scene- and episode-memory conditions, then improve the supplied scene knowledge with matched GT variants. We next diagnose the residual failures through checkpoint interventions and use the resulting diagnosis to motivate ESM, a targeted geometric memory intervention.

4.1 Native Memory Conditions

M0 provides no persistent memory. M1–M3 preserve task-time experience as a compact image history, textual events, or a policy-maintained structured state. M4–M6 are constructed from the same finite pre-task visual experience: M4 retains 16 representative observations, M5 compresses the experience into a task-independent textual description, and M6 organizes it into places, objects, and spatial relations. M7 combines the Structured scene memory of M6 with a Structured episode state maintained online.

Condition	Scene Memory	Episode Memory
M0	None	None
M1	None	Image history
M2	None	Text events
M3	None	Structured state
M4	Raw-16	None
M5	Text	None
M6	Structured	None
M7	Structured	Structured state

Because native scene memory is constructed from finite visual experience, it may omit relevant observations or contain inaccurate semantic or spatial relations. Episode memory introduces a separate source of error as the policy updates task-time state during navigation. Scene memory remains frozen during execution, whereas episode memory evolves with interaction. These conditions probe representative changes in scene and episode memory rather than forming a complete factorial; M7 provides one matched comparison of their joint use.

4.2 GT Scene-Knowledge Intervention

Errors in native scene memory make it difficult to distinguish insufficient environmental knowledge from failures to use available knowledge during action. Unlike native scene memory, which is constructed by summarizing finite pre-task visual experience, the GT variants are built directly from ground-truth scene data. This provides substantially more complete and reliable, task-independent knowledge of places, characteristic visual cues, contained objects, floor assignments, and directly passable connectivity.

We expose this knowledge through matched textual and structured interfaces, yielding $\mathrm{M5} \rightarrow \mathrm{GT_TEXT}$, $\mathrm{M6} \rightarrow \mathrm{GT_STRUCTURED}$, and $\mathrm{M7} \rightarrow \mathrm{GT_JOINT}$. GT_JOINT replaces only M7's scene-memory component while retaining the same policy-maintained Structured episode memory. The intervention therefore improves the environmental knowledge supplied to the policy while pre-

Table 1: **Performance under native memory conditions.** ΔSR reports the paired change in SR with 95% scene-bootstrap confidence intervals. M1–M6 are compared with M0; M7 is compared with M6.

Condition	Scene Mem.	Episode Mem.	SR	OSR	SPL	ΔSR [95% CI]
M0	None	None	42.19	46.88	0.2640	–
M1	None	Image	50.78	53.91	0.3231	+8.59 [0.78, 16.41]
M2	None	Text	42.97	47.66	0.2943	+0.78 [-7.81, 9.38]
M3	None	Structured	49.22	53.13	0.3260	+7.03 [-0.78, 14.84]
M4	Raw-16	None	51.56	57.03	0.4040	+9.38 [-0.78, 19.53]
M5	Text	None	**65.63**	**69.53**	**0.5181**	+23.44 [13.28, 33.59]
M6	Structured	None	62.50	63.28	0.4514	+20.31 [9.38, 31.25]
M7	Structured	Structured	63.28	65.63	0.4633	+0.78 [-6.25, 8.59]

serving the downstream navigation interface. Current localization, memory-to-view correspondence, and action selection remain part of the policy's situated reasoning. This allows us to ask how much failure can be recovered by improving scene knowledge, and what failures remain beyond it.

4.3 Checkpoint-based Failure Diagnosis

We diagnose residual failures by selecting a diagnostically supported decision checkpoint at which an observable error can be independently verified from the trajectory and scene evidence. For offline analysis, we jointly inspect the executed path and current state with the corresponding egocentric observations, memory, interaction evidence, and policy trace. Privileged top-down information is used only to localize and verify the diagnostic checkpoint.

For each localized failure, we compare three matched branches from the same checkpoint: an unmodified replay (**B0**), a **Targeted** branch, and a matched **Sham**. Targeted uses verified local information to correct only the identified erroneous decision, after which the original policy continues the episode. The intervention is applied once; subsequent failures are analyzed but not corrected again.

4.4 Experience Skeleton Memory

The preceding diagnosis motivates **Experience Skeleton Memory (ESM)**, a targeted intervention for keeping executed experience spatially usable as the agent's situation changes. ESM organizes task-time experience into an executed route skeleton, revisable region hypotheses with attached observations, and an egocentric rendering of relevant past experience.

ESM uses simulator-provided metric poses as an execution-side odometric signal to preserve the geometry of motion actually executed by the agent. The poses are used to construct the executed route skeleton and transform previously traversed segments into the current camera frame; region identities remain model-generated and revisable.

At each decision, ESM presents relevant historical observations together with previously traversed structure in the current egocentric frame. It therefore provides an explicit geometric scaffold for re-grounding past experience, while semantic localization, goal reasoning, and action selection remain with the policy. We treat ESM as a *diagnosis-driven geometric memory intervention*, rather than a pure memory-format comparison.

5 Experiments and Results

5.1 How Do Different Memories Affect Closed-loop Navigation?

We first evaluate the eight native memory conditions in Section 4.1. Table 1 reports the complete navigation results, while Figure 3 summarizes the main memory-to-action evidence across the native and GT interventions.

Scene memory yields the clearest aggregate gains. Relative to M0 (42.19% SR), Text and Structured scene memory reach 65.63% and 62.50%, corresponding to improvements of +23.44 pp (95% CI:

Table 2: **Cross-policy robustness on the same 128 tasks.** We report Success Rate (SR) under no persistent memory (M0), joint Structured scene and episode memory (M7), and GT_JOINT. Paired differences report 95% scene-bootstrap confidence intervals.

	Success Rate (SR)			Paired ΔSR [95% CI]	
Policy	**M0**	**M7**	**GT_JOINT**	**M7 – M0**	**GT_JOINT – M7**
Primary Policy	42.19	63.28	**72.66**	+21.09 [11.72, 31.25]	+9.38 [0.00, 18.75]
Policy 2	25.00	47.66	49.22	**+22.66 [14.06, 31.25]**	+1.56 [-7.03, 10.16]

[+13.28, +33.59]) and +20.31 pp ([+9.38, +31.25]), respectively. Raw-16 yields a smaller change (+9.38 pp; [-0.78, +19.53]). The corresponding OSR and SPL results show the same broad pattern: task-independent environmental experience can substantially improve closed-loop navigation when retained in a compact textual or structured form.

Persistent episode memory has bidirectional task-level effects. Without scene memory, Image, Text, and Structured episode memory change SR by +8.59, +0.78, and +7.03 pp, respectively; adding Structured episode memory to Structured scene memory changes SR by only +0.78 pp. These small net differences can nevertheless hide substantial behavioral change: M6→M7 contains 14 failure-to-success and 13 success-to-failure transitions, while M0→M2 contains 16 and 15. Persistent episode memory therefore changes which individual tasks succeed, but does not provide a consistently positive aggregate benefit.

The joint-memory effect transfers across navigation policies. Replacing only the high-level navigation policy with GPT-6 Luna raises SR from 25.00% under M0 to 47.66% under M7 ($+22.66$ pp; 95% CI: $[+14.06, +31.25]$), while GT_JOINT reaches 49.22% ($+1.56$ pp over M7; $[-7.03, +10.16]$). For the primary policy, the corresponding changes are $+21.09$ pp ($[+11.72, +31.25]$) and $+9.38$ pp ($[+0.00, +18.75]$), respectively. Thus, the large M0→M7 difference is reproduced with a second navigation policy, while the additional GT gain is smaller for Policy 2.

5.2 Does Better Scene Knowledge Resolve Navigation Failure?

We next replace native scene memory with the substantially more complete and reliable GT scene knowledge defined in Section 4.2, while holding episode memory fixed. The three matched interventions are summarized in Figure 3b.

Better scene knowledge improves navigation but does not close the gap. GT_TEXT changes SR from 65.63% to 70.31% (+4.69 pp; 95% CI: [-3.12, +12.50]), GT_STRUCTURED from 62.50% to 69.53% (+7.03 pp; [-1.56, +15.62]), and GT_JOINT from 63.28% to 72.66% (+9.38 pp; [+0.00, +18.75]). Thus, substantially better scene knowledge recovers part of the observed failure, with the largest point improvement under GT_JOINT.

Substantial residual failure remains. As shown in Figure 3c, GT_JOINT succeeds on 93/128 tasks but still fails on **35/128** (27.34%). These residual failures span all four task families, showing that insufficient scene knowledge alone cannot account for the remaining failures. We therefore audit these 35 episodes to determine where the memory-to-action process breaks down despite substantially better environmental knowledge.

5.3 Auditing the Situated Memory-to-Action Chain

A failed trajectory alone does not reveal where the memory-to-action process breaks. Similar behavioral symptoms—such as route deviation, repeated backtracking, and local oscillation—can arise from fundamentally different errors. For the policy studied here, a high-level navigation decision can be operationalized as a situated sequence: identify the current location, locate the goal in remembered space, relate that remembered structure to the current egocentric observation, and select the most plausible visible transition or local target. The selected target must then be visually grounded, executed, and eventually verified against task completion.

Accordingly, we audit the 35 GT_JOINT residual episodes along six sequential stages: **Current Localization**, **Goal Localization**, **Egocentric Correspondence**, **Visual Grounding**, **Execution**, and

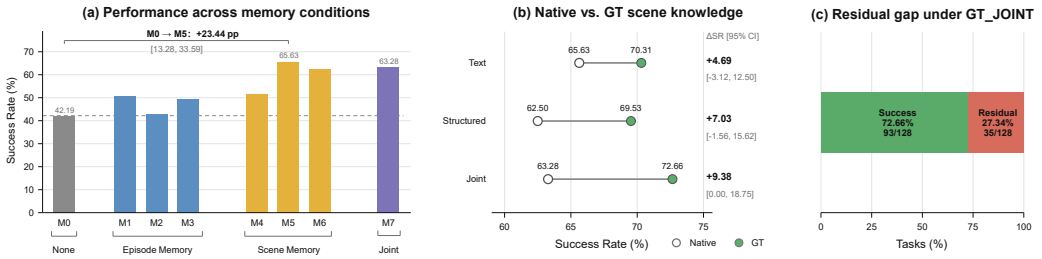

Figure 3: **Quantitative evidence for the memory-to-action gap. (a)** Success rate across the eight native memory conditions. Scene memory produces the clearest aggregate gains, with M0→M5 improving SR by 23.44 percentage points. **(b)** Matched replacement of native scene memory with substantially more complete and reliable GT scene knowledge yields positive SR changes across Text, Structured, and Joint conditions; error intervals denote 95% scene-bootstrap confidence intervals. **(c)** Despite reaching 72.66% SR, GT_JOINT still leaves 35/128 tasks unsuccessful, motivating the residual-failure analysis.

Claim. Persistent episode-memory errors that affect multiple stages are recorded separately as **STM Record Error**.

The recorded decision trace makes these failures auditable. Using the per-step signals defined in Section 3.3, we inspect the policy's reported current-region belief, goal belief, selected action, and self-reported rationale together with grounding and execution outcomes, the current observation, and the available memories. We assign each failed episode one primary behavioral attribution. The self-reported rationale is treated only as an observable diagnostic signal, not as direct access to the model's latent reasoning. Two additional annotators independently reviewed all 35 episodes; each agreed with the primary attribution on 34/35 cases (97.1%; pairwise Cohen's $\kappa = 0.962$).

Most residual failures occur before local grounding and execution. As shown in Figure 4a, **24/35 (68.6%)** residual failures arise in Current Localization, Goal Localization, or Egocentric Correspondence. **Egocentric Correspondence (12/35)** is the largest single category. Thus, even with substantially more complete scene knowledge, the dominant residual difficulty is often not whether useful environmental knowledge is available, but whether that knowledge can be situated correctly with respect to the agent's current observation and state.

These attributions turn otherwise similar trajectory-level failures into distinct intervention targets. We next test whether correcting the diagnosed state changes downstream behavior.

5.4 Behavioral Validation of the Diagnosed States

The attributions in Section 5.3 identify a concrete, diagnostically supported local error, but observational diagnosis alone does not establish that the error is behaviorally consequential. We therefore apply the matched one-shot interventions defined in Section 4.3 to all 35 residual episodes. At the diagnosed checkpoint, Targeted corrects only the identified erroneous decision using verified local information, after which the original policy continues the episode.

Targeted repair changes downstream outcomes. As shown in Figure 4b, B0 and Sham each succeed on 5/35 episodes, whereas Targeted succeeds on 15/35. Relative to Sham, Targeted improves SR by **+28.57 pp** (95% scene-bootstrap CI: **[+13.89,+45.45]**), with 10 failure-to-success transitions and no success-to-failure transitions. B0 and Sham have identical aggregate SR and only one transition in each direction, supporting the interpretation that the diagnosed decision—rather than checkpoint perturbation alone—is behaviorally consequential.

A single repair is not sufficient for full recovery. Twenty of the 35 episodes still fail after targeted correction. A locally corrected decision therefore does not guarantee that the policy will maintain and reuse the corrected situated understanding throughout the remaining closed loop.

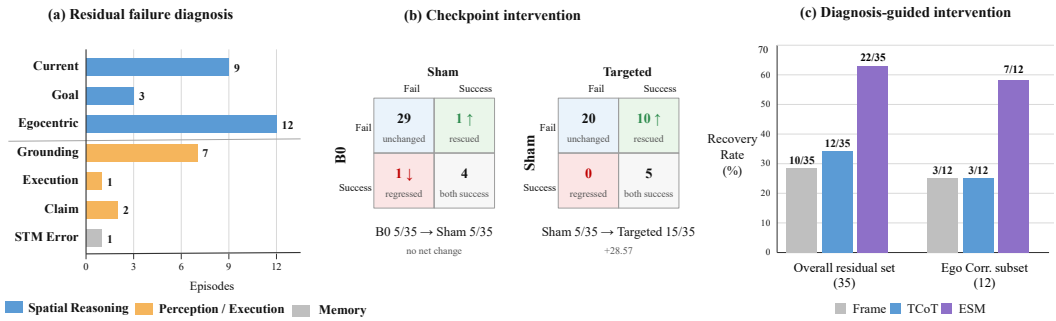

Figure 4: **Diagnosing and intervening on residual failures.** **(a)** Primary attribution of the 35 GT_JOINT residual failures; 24/35 (68.6%) occur before local grounding and execution, with Egocentric Correspondence the largest category (12/35). **(b)** Matched checkpoint intervention: B0 and Sham each succeed on 5/35 episodes, whereas Targeted correction succeeds on 15/35 (+28.57 pp over Sham; 95% scene-bootstrap CI: $[+13.89, +45.45]$). **(c)** Diagnosis-guided episode-memory intervention: Frame, TCoT, and ESM recover 10/35, 12/35, and 22/35 episodes, respectively; ESM improves over TCoT by $+28.57$ pp (95% scene-bootstrap CI: $[+7.69, +51.35]$). Within Egocentric Correspondence, ESM recovers 7/12 versus 3/12 for both baselines.

5.5 A Diagnosis-Driven Intervention for Situated Reuse

The residual analysis suggests that previously executed experience must remain spatially relatable to the agent's changing observation. We therefore evaluate ESM (Section 4.4) as a diagnosis-driven geometric intervention that preserves executed structure and re-expresses historical experience relative to the current view. Because ESM uses execution-side metric geometry, we treat it as a targeted intervention rather than a pure memory-format comparison.

ESM substantially improves recovery on the residual set. As shown in Figure 4c, ESM recovers **22/35** episodes, compared with 10/35 for Frame and 12/35 for TCoT, corresponding to +34.29 pp over Frame (95% CI: [+14.71,+54.84]) and +28.57 pp over TCoT ([+7.69,+51.35]). For Egocentric Correspondence, ESM recovers **7/12** cases versus 3/12 for both baselines, consistent with the bottleneck that motivated the intervention.

Table 3: **Full-suite ESM comparison.**

	Success	SR	ΔSR
GT_JOINT	93/128	72.66	–
ESM	**110/128**	**85.94**	**+13.28**

95% paired scene-bootstrap CI: [+6.25,+21.09] pp.

The gain extends to the full task suite. ESM reaches **110/128 (85.94%)** versus 93/128 (72.66%) under GT_JOINT, a paired improvement of **+13.28 pp** (95% CI: [+6.25,+21.09]), with 22 failure-to-success and 5 success-to-failure transitions.

6 Conclusion

We introduced SCOPE to study the gap between environmental memory and situated action in closed-loop navigation. Our results show that memory correctness does not imply memory actionability: even with substantially more complete GT scene knowledge, 35/128 tasks still fail, and 24/35 of these residual failures are localized before local visual grounding and execution, with Egocentric Correspondence as the largest category. Matched checkpoint interventions further show that these diagnoses are behaviorally consequential: targeted repair recovers 15/35 episodes, compared with 5/35 under matched Sham. Finally, the diagnosis-driven ESM intervention recovers 22/35 residual episodes and reaches 110/128 successes on the full task suite, showing that the diagnosed bottlenecks can inform a practically consequential change in how executed experience is maintained and reused.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript organization, drafting and language editing, as well as software implementation and debugging. All AI-assisted text and code were reviewed and verified by the authors. The authors independently verified the experimental results, factual claims, and citations, and take full responsibility for the final content of the paper.

REFERENCES

Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018a.

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 *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 3674–3683. IEEE, 2018b.

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

Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological slam for visual navigation. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 12872–12881. IEEE, 2020.

Kuan Fang, Alexander Toshev, Li Fei-Fei, and Silvio Savarese. Scene memory transformer for embodied agents in long-horizon tasks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 538–547. IEEE, 2019.

Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, et al. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028. IEEE, 2024.

Wenbo Hu, Yining Hong, Yanjun Wang, Leison Gao, Zibu Wei, Xingcheng Yao, Nanyun Peng, Yonatan Bitton, Idan Szpektor, and Kai-Wei Chang. 3dllm-mem: Long-term spatial-temporal memory for embodied 3d large language model. *Advances in Neural Information Processing Systems*, 38:67856–67884, 2026.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10608–10615. IEEE, 2023.

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 *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16373–16383. IEEE, 2024.

Chih-Ting Liao, Xi Xiao, Chunlei Meng, Zhangquan Chen, Yitong Qiao, Weilin Zhou, Tianyang Wang, Xu Zheng, and Xin Cao. Spamem: Benchmarking dynamic spatial reasoning via perception–memory integration in embodied environments. In *European Conference on Computer Vision*, pp. 425–442. Springer, 2026.

Peiqi Liu, Zhanqiu Guo, Mohit Warke, Soumith Chintala, Chris Paxton, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Dynamem: Online dynamic spatio-semantic memory for open world mobile manipulation. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13346–13355. IEEE, 2025.

Rui Liu, Wenguan Wang, and Yi Yang. Volumetric environment representation for vision-language navigation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16317–16328. IEEE, 2024.

Arjun Majumdar, Anurag Ajay, Xiaohan Zhang, Pranav Putta, Sriram Yenamandra, Mikael Henaff, Sneha Silwal, Paul Mcvay, Oleksandr Maksymets, Sergio Arnaud, et al. Openeqa: Embodied question answering in the era of foundation models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16488–16498. IEEE, 2024.

Santhosh K Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alex Clegg, John Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.

Saim Wani, Shivansh Patel, Unnat Jain, Angel Chang, and Manolis Savva. Multion: Benchmarking semantic map memory using multi-object navigation. *Advances in Neural Information Processing Systems*, 33:9700–9712, 2020.

Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*, 2024.

Karmesh Yadav, Yusuf Ali, Gunshi Gupta, Yarin Gal, and Zsolt Kira. Findingdory: A benchmark to evaluate memory in embodied agents. *arXiv preprint arXiv:2506.15635*, 2025.

Jihan Yang, Shusheng Yang, Anjali W Gupta, Rilyn Han, Li Fei-Fei, and Saining Xie. Thinking in space: How multimodal large language models see, remember, and recall spaces. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10632–10643. IEEE, 2025a.

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17294–17303. IEEE, 2025b.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024.

Lingfeng Zhang, Yuecheng Liu, Zhanguang Zhang, Matin Aghaei, Yaochen Hu, Hongjian Gu, Mohammad Ali Alomrani, David Gamaliel Arcos Bravo, Raika Karimi, Atia Hamidizadeh, et al. Mem2ego: Empowering vision-language models with global-to-ego memory for long-horizon embodied navigation. *arXiv preprint arXiv:2502.14254*, 2025.

Pingyue Zhang, Zihan Huang, Yue Wang, Jieyu Zhang, Letian Xue, Zihan Wang, Qineng Wang, Keshigeyan Chandrasegaran, Ruohan Zhang, Yejin Choi, et al. Theory of space: Can foundation models construct spatial beliefs through active exploration? In *The Fourteenth International Conference on Learning Representations*, 2026.

Yanpeng Zhao, Wentao Ding, Hongtao Li, Baoxiong Jia, and Zilong Zheng. Espire: A diagnostic benchmark for embodied spatial reasoning of vision-language models. *arXiv preprint arXiv:2603.13033*, 2026.

A Benchmark Construction and Task Suite

This appendix provides additional details on the construction, taxonomy, and validation of the SCOPE task suite. The benchmark contains 128 navigation tasks across 64 HM3D indoor scenes, with two distinct task forms selected for each scene. Of the 128 tasks, 57 primarily target objects and 71 target regions, covering 25 object categories and 21 region-category labels. Scenes contain 5–24 annotated regions, with a median of 13. The task suite is organized into four broad families and eight concrete task forms, summarized in Table 4.

Table 4: Task taxonomy of SCOPE. The four families are used for analysis, while the eight task forms specify the construction of individual tasks. "Completion" summarizes the evaluator-side completion structure.

Family	Task form	#	Completion structure
Direct Navigation	Object navigation	23	Reach and complete a single specified object target.
	Region navigation	14	Reach and complete a single specified region target.
Target Disambiguation	Visual-attribute disambiguation	24	Select the intended target using distinguishing visual or content attributes.
	Spatial-relation disambiguation	18	Select the intended target using a specified relation to an anchor object or region.
Search & Collection	Candidate-region search	10	Find a fixed object target given one or more candidate semantic region types in which it may occur.
	Collection	21	Complete a set of distinct valid instances, either up to a required quota or over the complete target set.
Multi-stage Navigation	Multi-goal	10	Complete multiple targets in the specified order.
	Return / revisit	8	Complete an outbound target and then revisit a designated earlier location or starting region.

A.1 Task Taxonomy and Coverage

The four task families describe different closed-loop demands rather than different evaluator implementations. *Direct Navigation* contains single-target object and region navigation. *Target Disambiguation* requires identifying the intended target from visual attributes or spatial relations. *Search & Collection* includes both object search over candidate region types and collection of multiple distinct instances. *Multi-stage Navigation* requires ordered completion of several goals or a return to a previously designated location.

The task-form labels are benchmark annotations and should not be conflated with the underlying loader types. For example, attribute and relation tasks may share the same low-level object- or region-target representation as direct navigation. Their distinction lies in how the intended target is specified and bound during task construction.

Collection tasks further contain two completion modes. Six tasks use a *quota* objective, in which any two distinct valid instances are sufficient; five of these tasks contain three valid candidates and one contains four. The remaining 15 tasks are *exhaustive*: all valid instances in the task-defined set must be completed. Twelve exhaustive tasks state the set size in the instruction, whereas three leave the total unannounced. Repeated completion of the same physical instance does not provide additional credit.

A.2 Task Construction and Validation

Each task freezes its natural-language instruction, starting pose, valid target set, completion semantics, and execution budget before evaluation. Private identifiers such as object instance IDs and region IDs are retained by the evaluator to define the target precisely, but are not inserted into the task instruction. Thus, the natural-language task can refer to observable attributes, semantic region types, or spatial relations while evaluation remains tied to a fixed physical target.

For attribute-based tasks, construction first identifies the target instance and then specifies visual or semantic properties that distinguish it from alternatives. For spatial-relation tasks, a target–anchor binding is frozen together with the relation used by the instruction. Relations such as `connected_to` denote the validated relation used to select the target; the online evaluator does not re-interpret the language relation at test time. Candidate-region search similarly fixes the target object while exposing only semantic candidate regions, e.g., that a monitor may occur in a study or bedroom. Visiting each candidate region is not itself an evaluation subgoal.

Table 5: Representative instructions from the frozen SCOPE benchmark. Each row is an actual evaluated task.

Task form	Instruction	Difficulty
Object navigation	"Go to the stove"	Easy
Region navigation	"Go to the laundry room."	Medium
Visual-attribute disambiguation	"Go to the long, multi-seat sofa in the corridor."	Medium
Spatial-relation disambiguation	"Go to the bedroom with an en-suite bathroom."	Medium
Candidate-region search	"Find the computer monitor. It may be in a study or bedroom."	Hard
Collection	"Find all 2 bedrooms"	Medium
Multi-goal	"Go to the bathroom on the half-level immediately below the kitchen level, then the bathroom on the half-level immediately above the kitchen level, then the kitchen."	Hard
Return / revisit	"Go to the dining area, then return to the study with glass double doors."	Medium

Collection tasks freeze the complete set of physically valid instances. Quota tasks succeed after completing the required number of distinct valid instances, whereas exhaustive tasks require the full set. Multi-goal tasks store an ordered target sequence and are evaluated stage by stage. Return/revisit tasks additionally bind a later stage to a designated earlier location; the return criterion is spatial and does not require reproducing an exact previous camera pose.

Task feasibility is checked independently of the evaluated policy. The 96 newly constructed tasks each have at least one successful reference execution that verifies navigable starts, required target locations, and the necessary sequence of navigation legs in Habitat-Sim. The remaining 32 tasks were retained from an earlier frozen suite together with their existing validation records rather than being retrospectively regenerated through the newer construction pipeline. Reference executions serve only as executable witnesses: they are not assumed to be globally shortest routes and are never provided to the navigation policy as route or action guidance.

A.3 Difficulty and Execution Budgets

The benchmark contains 13 easy, 51 medium, and 64 hard tasks. These labels are stored as task-construction metadata and are not inferred from the observed success rates of the evaluated policies. They determine the high-level decision budget as follows:

$$B_\tau = \begin{cases} 30, & \text{easy}, \\ 45, & \text{medium}, \\ 60, & \text{hard}. \end{cases}$$

All tasks additionally use a low-level environment-action cap of 900. For a given task, the starting state and both budgets are identical across all memory conditions. Checkpoint continuations, when used for the diagnostic experiments in Section F, inherit the budget already consumed by the original prefix rather than receiving a new task budget.

A.4 Representative Tasks

Table 5 gives one exact instruction from the frozen benchmark for each task form. These examples are taken directly from the evaluated task set rather than written as illustrative paraphrases.

B Native Memory Construction

This appendix details how the native scene- and episode-memory conditions in Section 4.1 are constructed. All native scene memories originate from task-independent pre-task experience, whereas episode memory is produced online from observations and interactions during the current navigation task. The eight native conditions are summarized in Table 6.

B.1 Shared Pre-task Experience

For each scene s, we freeze a pre-task experience sequence

$$E_s = (o_1, a_1, o_2, \ldots, o_T), \tag{2}$$

before constructing the downstream navigation tasks. The two tasks associated with the same scene share the same underlying experience and the same frozen scene-memory artifacts.

The 64 scene experiences were collected using two sources: 59 scenes use an annotation-guided automatic acquisition procedure, while five early scenes use human demonstrations collected during protocol development before the downstream tasks were defined. Thus, neither acquisition process uses the subsequent task instruction or target to search for task-specific evidence.

For the automatically collected scenes, scene annotations and navigable geometry are used to select viewpoints and obtain broad spatial coverage, but these annotations are not directly exposed as native memory. The resulting experience stores temporally ordered RGB observations and executed actions. The complete experience is finite: it reflects what was actually recorded during Phase 1 rather than an exhaustive observation of every object, viewpoint, or relation in the scene. Across the 64 scenes, the recorded RGB pools contain 79–809 frames.

Importantly, the native scene-memory conditions share this underlying experience but do not necessarily consume identical subsets of frames. Raw-16 and textual memory use different fixed sampling procedures, described below.

B.2 Scene-memory Construction

Raw visual memory (M4). M4 retains 16 RGB observations from the recorded Phase 1 sequence. Frames are selected approximately uniformly along the cumulative traveled path length: the sampler constructs a distance axis from consecutive recorded positions, selects observations closest to equally spaced locations along this axis, and then restores their original temporal order. Thus, Raw-16 is not obtained by uniform temporal subsampling. The policy receives the selected RGB images but not the positions used by the sampler, ground-truth region identities, or object-level annotations.

Text scene memory (M5). M5 converts a larger portion of the same Phase 1 experience into a task-independent textual description. Up to 160 RGB observations are selected in their original temporal order, with endpoints preserved. The selected frames are processed in overlapping chunks of eight frames with an overlap of two, and a fixed builder consolidates the resulting descriptions into a scene-level travelogue.

The builder is not given the downstream navigation instruction. Its output is therefore a description of the recorded exploration rather than a task-conditioned retrieval result. The resulting text may describe places, objects, appearance cues, traversed transitions, and inferred relative floor structure, but it remains a compression of finite visual evidence.

Structured scene memory (M6). M6 is constructed *from the frozen M5 textual memory*, rather than by independently re-processing the Phase 1 RGB observations. A second fixed builder organizes the complete travelogue into a structured spatial representation containing places, objects, connections, route information, relative floor structure, and explicit uncertainties.

Places are assigned memory-local identifiers and may contain a semantic label, visual cues, relative level, visit status, and supporting evidence. Objects are associated with remembered places, while connections represent relations such as traversed transitions or visually suggested openings. Floor identifiers are relative to the recorded experience rather than simulator floor IDs.

This construction gives the native scene-memory lineage

$$E_s \rightarrow \begin{cases} \text{16 sampled RGB observations} & \text{(M4)}, \\ \text{textual scene description} & \text{(M5)}, \\ \text{text description} \rightarrow \text{structured scene memory} & \text{(M6)}. \end{cases}$$

Consequently, M5 and M6 share the same underlying visual experience, but M6 also inherits information losses or errors introduced during the RGB-to-text conversion before applying its additional structuring step.

B.3 Episode-memory Construction

Episode memory is reset at the beginning of each navigation task and is updated only from information generated during that episode. We evaluate three persistent interfaces in addition to the shared one-step interaction feedback.

Image history (M1). M1 retains selected decision-time images from earlier steps together with the actual visual-grounding box associated with the selected local target. At most 16 historical images are presented to the policy at a decision. When the history exceeds this capacity, the six most recent images are retained and up to ten earlier images are sampled in temporal order. This interface stores decision-level visual evidence rather than every low-level simulator frame or the complete sequence of intermediate walking observations.

Text event memory (M2). M2 maintains a chronological sequence of compact textual events written by the navigation policy itself. After reviewing the preceding interaction, the policy emits an event summary that becomes available at subsequent decisions. The interface retains at most 64 events, with a per-event length limit of 1,200 characters. Older events are discarded in first-in–first-out order when the capacity is exceeded. The system stores these summaries but does not rewrite them using simulator ground truth.

Structured episode state (M3). M3 maintains a compact policy-generated spatial state using the `cot_full_place_ledger_v9` schema. Its principal components are a current-place belief, a set of remembered places, connections between places, and compact exploration state. A place may additionally record a model-generated correspondence to a scene-memory place, visual signatures, visit and exploration status, task-related state, and bounded coverage descriptions.

The state supports at most 64 places and 128 connections; each place retains up to six `checked` and four `unresolved` coverage items. The policy updates the state through explicit deltas. Place and connection entries are upserted by their model-generated identifiers, while omitted entries remain unchanged. Coverage updates similarly add or remove explicitly named items rather than rewriting the complete state.

Crucially, these identifiers and relations represent the policy's own belief. Episode place IDs are distinct from scene-memory place IDs, and any correspondence between them is proposed by the policy. The system does not use ground-truth pose, region identity, or object identity to merge places, correct connections, or repair an incorrect current-place belief. Errors can therefore persist across subsequent decisions.

B.4 Joint Scene and Episode Memory

M7 combines the two timescales without modifying either memory source. For every task, M6 and M7 load the same frozen structured scene-memory artifact, while M3 and M7 use the same structured episode-memory mechanism. The online episode state is not written back into the frozen scene memory.

The combination nevertheless does not imply that the realized state in M7 is identical to the state that would have been produced in M3. In M7, the policy can explicitly associate an episode-level place with a place in scene memory, and the additional scene context may change subsequent observations, actions, and state updates. M7 should therefore be interpreted as a matched composition of the two interfaces rather than as an arithmetic combination of independently generated memories.

B.5 Sources of Information Loss and Error

Native memory is intentionally derived from finite visual experience rather than privileged scene state. Several stages can therefore introduce information loss or incorrect beliefs. First, relevant content may never have been observed during Phase 1. Second, fixed sampling can omit observations from Raw-16 or from the subset used to construct textual memory. Third, the RGB-to-text builder can omit or misdescribe scene content. Because M6 is built from the frozen M5 text, such errors can propagate into structured memory, with an additional opportunity for incorrect place identity,

Table 6: Native memory conditions in SCOPE. Scene memory is frozen before task execution; episode memory is reset for each task and updated online. All conditions additionally receive the shared current observation, task state when applicable, and one-step interaction feedback defined in Section 3.1.

Cond.	Scene memory	Episode memory	Construction / update
M0	None	None	Shared current context only; no additional persistent memory.
M1	None	Image history	Up to 16 sampled decision images with their executed grounding boxes.
M2	None	Text events	Policy-generated event summaries retained chronologically, up to 64 events.
M3	None	Structured state	Policy-maintained place, connection, exploration, and task state using the v9 structured ledger.
M4	Raw-16	None	Sixteen Phase 1 RGB observations sampled approximately uniformly along traveled path length.
M5	Text	None	Task-independent textual travelogue produced from up to 160 Phase 1 RGB observations.
M6	Structured	None	Structured spatial representation constructed from the frozen M5 travelogue.
M7	Structured	Structured state	The same frozen scene memory as M6 together with the same online structured episode-state interface as M3.

floor, object, or connectivity extraction. Finally, M2, M3, and M7 depend on policy-generated online updates, which may record an incorrect event, place identity, connection, or exploration state.

These error sources motivate the controlled GT scene-knowledge intervention described in Section C, where scene knowledge is replaced while the downstream navigation interface remains matched.

C GT SCENE-KNOWLEDGE INTERVENTION

Native scene memory is constructed from finite pre-task visual experience and may therefore omit relevant observations or contain inaccurate semantic and spatial relations. To distinguish such scene-knowledge limitations from failures to use available knowledge during navigation, we construct substantially more complete, reliable, and task-independent GT scene knowledge of the static environment and replace the corresponding native scene-memory component while leaving the downstream navigation setting matched.

C.1 CANONICAL GT SCENE KNOWLEDGE

Unlike native scene memory, which is derived from the recorded Phase 1 experience E_s, GT scene knowledge is compiled directly from audited scene data. The construction uses region annotations, object-instance records, visual identity and attribute evidence, floor assignments, connectivity evidence, and associated confirmation or exclusion records. Consequently, the GT source is not restricted to content that happened to be observed along the Phase 1 exploration trajectory and may include scene facts omitted from native memory.

For each scene s, these sources are normalized into a canonical static fact set G_s. Within its declared scope, G_s contains floors and places, together with characteristic visual cues, contained objects, object appearance attributes, floor assignments, and directly passable connections between places. The canonical representation uses scene-local opaque identifiers for places and objects; metric coordinates and simulator-private source identifiers are not part of the policy-visible memory.

We use "complete" with respect to this declared static-scene scope rather than to every fact available in the simulator. The intervention targets the environmental information most relevant to the navi-

Table 7: Canonical scene knowledge and its policy-visible renderings. Both GT interfaces are generated from the same audited canonical fact set.

Canonical fact	**GT_TEXT**	**GT_STRUCTURED**
Place identity / type	Natural-language place description	Explicit place entry with an opaque scene-local ID
Visual cues	Appearance cues described in text	`visual_cues` associated with the place
Contained objects	Object categories and appearance described within the place	Explicit object entries attached to the place
Floor assignment	Floor or relative-level information expressed in text	Explicit floor assignment for each place
Connectivity	Directly passable neighboring places described in text	Explicit bidirectional passable connections

gation setting—place identity and appearance, object containment, floor structure, and traversable connectivity—without constructing an exhaustive metric world model.

C.2 Text and Structured Renderings

Both GT memory interfaces are rendered from the *same* canonical fact set:

$$G_s \longrightarrow \left\{T_s^{\mathrm{GT}}, S_s^{\mathrm{GT}}\right\}, \tag{3}$$

where T_s^{GT} is the textual rendering used by GT_TEXT, and S_s^{GT} is the structured rendering used by GT_STRUCTURED and GT_JOINT.

The textual rendering converts the canonical facts into a continuous scene-level description. It expresses places, characteristic visual cues, object containment, relative floor information, and connectivity in natural language, without exposing the canonical place/object identifiers. The structured rendering preserves the same underlying information as explicit place entries containing floor assignment, visual cues, objects, and bidirectionally passable connections. Thus, differences between GT_TEXT and GT_STRUCTURED arise from their presentation interface rather than from different underlying GT scene facts.

Before evaluation, the final GT artifacts are subjected to automated consistency checks over the canonical representation and its renderings. These checks verify the declared fact scope, schema validity, rendering consistency, and paired provenance of the Text and Structured variants. The final frozen pool uses only validated artifacts; scenes requiring correction are repaired and revalidated before entering evaluation.

C.3 Matched Native-to-GT Replacements

We use the GT memories as controlled replacements for the corresponding native scene-memory conditions:

$$\mathrm{M5} \rightarrow \mathrm{GT_TEXT}, \qquad \mathrm{M6} \rightarrow \mathrm{GT_STRUCTURED}, \qquad \mathrm{M7} \rightarrow \mathrm{GT_JOINT}.$$

Table 8 summarizes the intervention. M5 is replaced by the GT textual rendering, and M6 by the GT structured rendering. For GT_JOINT, only the frozen scene-memory component of M7 is replaced: the persistent Structured episode memory remains the same policy-maintained online state used in M7. It is not replaced by oracle task-time state.

Within each matched pair, the navigation task, starting pose, policy, observation interface, grounding and execution stack, success criterion, and decision budget remain fixed. The intervention therefore changes the quality and source of frozen environmental knowledge while preserving the surrounding closed-loop evaluation protocol.

C.4 Information Boundary

The GT intervention improves *static environmental knowledge*; it does not directly situate the policy in that knowledge during task execution. Specifically, the policy is not given its current absolute

Table 8: Matched Native→GT scene-knowledge replacements. The intervention changes the frozen scene-memory source while preserving the corresponding episode-memory interface.

Native	GT condition	Scene-memory replacement	Persistent episode memory
M5	GT_TEXT	Native Text → GT Text	None in both conditions
M6	GT_STRUCTURED	Native Structured → GT Structured	None in both conditions
M7	GT_JOINT	Native Structured → GT Structured	Same policy-maintained Structured episode-memory interface

pose or heading, its current GT region, a GT correspondence between the current observation and a memory place, a task-specific global route, or the next high-level action. Floor annotations describe where remembered places lie in the static scene, rather than revealing the floor on which the agent is currently localized.

The policy must therefore still infer its current situation from its egocentric observations and interaction history, relate that situation to the provided scene knowledge, select the relevant visible transition or local target, and continue updating its task-time state as navigation unfolds. The GT intervention is consequently not a deployable navigation method; it is a controlled scene-knowledge intervention that substantially reduces missing or inaccurate environmental knowledge as an explanation for failure. Residual failures under these conditions can then be analyzed as failures in situating, maintaining, grounding, or acting on knowledge that is already available to the policy.

D Shared Closed-loop Agent and Evaluation Details

This appendix expands the shared agent interface introduced in Section 3.3. Across memory conditions, SCOPE keeps the observation interface, high-level action space, visual grounding and claim handling, local execution stack, task-state protocol, and evaluator fixed. The experimental variable is therefore the memory available to the navigation policy, except in the focused cross-policy replication where the high-level policy itself is replaced while the surrounding agent stack remains shared.

D.1 Evaluated Policies and Shared Agent Stack

The primary navigation policy is GPT-5.6 Terra. We additionally instantiate the same closed-loop interface with GPT-6 Luna as a second navigation policy for the focused M0, M7, and GT_JOINT replication. In both cases, the visual Grounder remains GPT-5.6 Terra, and the task definitions, starting poses, memory-condition definitions, observation and action interfaces, local executor, and evaluator are kept matched. Cross-policy results are reported only after the corresponding experimental pool is frozen.

For the primary policy, the recorded inference configuration uses temperature 0, medium reasoning effort, and an output budget of 8192 tokens. The visual Grounder uses temperature 0, medium reasoning effort, and an output budget of 4096 tokens. Full request-level configurations, model-specific API differences, and experiment provenance are reported in Section H.

The shared closed loop follows

$$(C_t, M_t) \xrightarrow{\pi_\theta} a_t^{\text{high}} \xrightarrow{\text{ground / verify}} \tilde{a}_t \xrightarrow{\text{execute}} (o_{t+1}, f_{t+1}) \xrightarrow{\text{state / memory update}} (C_{t+1}, M_{t+1}), \quad (4)$$

where the navigation policy determines the task-level spatial decision, while the downstream modules standardize how that decision is visually grounded and physically executed.

D.2 Policy Input and High-level Decision Interface

At each high-level decision, the common policy context is

$$C_t = \{I_\tau, o_t, q_t, f_t\},$$

Table 9: Policy configurations used with the shared SCOPE agent stack. Policy-2 changes the high-level navigation policy while retaining the surrounding grounding, execution, and evaluation interfaces.

Component	Primary policy	Policy-2	Shared setting
Navigation policy	GPT-5.6 Terra	GPT-6 Luna	Same high-level input/output interface
Visual Grounder	GPT-5.6 Terra	GPT-5.6 Terra	Same visual grounding and claim-handling modules
Task pool	128 tasks / 64 scenes	Same 128 tasks	Matched instruction, start, and task budget
Memory evaluation	M0–M7 and GT conditions	M0, M7, GT_JOINT	Same condition definitions
Executor / evaluator	Shared	Shared	Same local motion and scoring implementation

together with the memory M_t permitted by the current condition. The current observation o_t consists of four separately presented 480×480 egocentric RGB views with approximately 90° fields of view, ordered as FORWARD, LEFT, BEHIND, and RIGHT. The policy is not shown a single stitched top-down or panoramic representation.

In addition to the task instruction and current observation, the common input contains the current high-level decision index and, when applicable, the shared task state q_t. The one-step feedback f_t describes the immediately preceding interaction and is detailed in Section D.5. Scene and persistent episode memory are condition-specific and enter through M_t.

The formal high-level action space contains two primary action types: `goto` and `claim`. A `goto` action specifies

$$a_t^{\text{goto}} = (d_t, \ell_t, h_t, r_t, s_t), \tag{5}$$

where d_t selects one of the four current views, ℓ_t is a discriminative description of a currently visible local target, h_t is a coarse within-image spatial hint, r_t is a distance bin (`short`, `medium`, or `far`), and s_t optionally indicates upward or downward stair traversal. The policy therefore chooses *what visible local transition or target to pursue*, but does not predict a bounding box or simulator coordinate.

A `claim` action instead declares that a task milestone has been reached. Object claims describe the object and its visible room context; region claims describe the region and distinguishing visible features. The policy also specifies whether it believes the claim completes the final task milestone. These fields are policy-generated statements rather than evaluator-confirmed task state.

D.3 VISUAL GROUNDING AND CLAIM HANDLING

For navigation actions, the visual Grounder receives only the RGB image selected by the policy together with the target description, coarse spatial hint, and distance bin. It predicts a normalized image-space bounding box for the selected local target. The navigation policy itself never emits or estimates this box.

Task claims use a separate visual verification path. Object claims are checked against the selected current RGB view using the claimed object description and local context. Region claims are verified from the current four-view observation because determining whether the agent is inside or immediately outside a region may require evidence outside a single selected image. When a region or object appears semantically appropriate but still requires a small physical approach, the claim handler may convert the proposal into a standardized local approach action instead of accepting the claim immediately.

Under the final claim protocol, a locally accepted claim may be written to the shared policy-visible task state when that task uses such state. Claims that are converted into approach actions or rejected for insufficient current visual evidence are not appended as completed records. Importantly, local

Table 10: Division of responsibility in the shared closed-loop agent. Task-level localization, memory correspondence, and target selection remain with the navigation policy; downstream modules convert the selected local decision into bounded physical execution.

Component	Primary responsibility	Information boundary
Navigation policy	Current localization, goal reasoning, memory-to-view correspondence, view selection, semantic local-target selection, and task claims	Receives current RGB and allowed memory, but not current GT pose/region or a task-level oracle route
Visual Grounder	Convert the policy-selected visible target into a local image-space goal	Operates on current visual evidence and policy-specified target information; does not solve the task-level route
Claim handling	Check whether current visual evidence supports the proposed local object or region claim, or whether additional local approach is needed	Does not expose evaluator-side instance identity or verified task completion
Local executor	Convert the grounded local target into bounded low-level motion	May internally use depth, pose, and navmesh geometry; these variables are not directly exposed to the navigation policy
Evaluator	Compute task success, trajectory success, instance matching, ordering, and path-based metrics	Operates offline from the policy and does not return GT task judgments during navigation

visual acceptance is distinct from evaluator-side task correctness: the Grounder does not have access to hidden target instance identity, task ordering truth, or final evaluator credit.

D.4 Local Navigation Executor

The shared executor converts a grounded image-space target into physical motion. It uses RGB-D geometry and camera calibration to back-project the visual target into a local 3D point. The formal execution stack then uses simulator pose and the Habitat navmesh to snap the local goal to navigable space and to follow a bounded geodesic path toward that *policy-selected local target*.

The formal configuration uses an oracle navmesh-based local planner internally. This privileged execution geometry is intentionally separated from the navigation policy: it is used to execute the local target already selected by the policy, not to determine the task-level destination, current semantic region, observation-to-memory correspondence, or global route. The policy is never given the exact metric pose, depth map, navmesh path, or current GT region produced or used by these execution components.

Low-level forward actions use a $0.25\,\mathrm{m}$ step and turns use 30° increments. A local follow segment has a nominal cap of 15 planner steps and a reach radius of $0.75\,\mathrm{m}$, while every complete task has an overall 900-environment-action budget. A high-level decision may therefore produce multiple low-level actions. Partial or failed local execution returns control to the high-level policy, which must interpret the resulting visual evidence at the next decision.

D.5 Immediate Feedback and Shared Task State

The immediate feedback f_t is designed to expose the observable consequence of the previous high-level decision without supplying a simulator verdict. For a navigation action at decision $t-1$, the next decision receives the previous action summary, the selected-view image annotated with the actual Grounder box, and up to two temporally ordered intermediate forward-motion frames. The current four-view observation serves as the endpoint of this interaction and is not duplicated as an additional review frame.

The policy then produces an `execution_review` describing whether the previous local objective appears completed, partial, failed, uncertain, or not applicable, together with a self-attributed failure source when relevant. This review is generated by the navigation policy from visual evidence; it is

not a simulator-provided moved/blocked/success signal. Immediate evidence is available only at the next decision unless the active episode-memory condition retains information from it.

The shared task state q_t is enabled for the 21 collection, 10 multi-goal, and 8 return/revisit tasks. It stores a chronological ledger of policy claims using the common `self_report` protocol. Each record contains the decision at which the claim was issued, its task type, the policy's completion description, and visible claim details. This ledger records *what the policy has declared*, not evaluator-verified progress. It does not reveal hidden target-set size, correct instance identity, duplicate status, true remaining milestones, or the correct next target.

Consequently, even with shared task bookkeeping, the policy must still use its own observations and memory to determine whether two encounters correspond to the same physical instance, whether a previously visited place has been reached again, and what spatial decision should follow.

D.6 Structured Decision Trace for Diagnosis

To support the residual-failure analysis in Section F, SCOPE records a structured decision trace at each high-level step. Policy outputs include an explicit current-location belief, a goal-location belief, an execution review of the preceding interaction, a self-reported action rationale, and the selected high-level action. The trace also records the scene and episode memory supplied at that decision when applicable.

System-side records include the selected view, visual-grounding result, claim-dispatch result, local execution outcome, and the task/episode state before and after the decision. Current four-view images, grounded-box review frames, and intermediate execution frames are retained for audit.

Simulator pose, GT region identity, and top-down trajectory information may additionally be recovered for *offline* diagnosis. These audit-only signals are not part of the policy input. Likewise, the policy's self-reported rationale and localization fields are treated as observable diagnostic signals, not as privileged access to an internal latent reasoning process or as ground-truth explanations of failure.

This separation allows the analysis to compare three distinct objects: what the policy believed and selected, what the visual/execution modules actually did, and what offline scene evidence shows to have been correct.

D.7 Termination, Budgets, and Evaluation

The high-level decision budget depends on the frozen task difficulty:

$$B_\tau \in \{30, 45, 60\},$$

for easy, medium, and hard tasks respectively. Every task additionally has a 900-step low-level environment-action budget. Both budgets are fixed for a given task across memory conditions. Checkpoint continuations preserve the already consumed prefix and inherit only the remaining budget.

An accepted claim marked by the policy as final terminates the rollout. Navigation is not automatically stopped simply because the evaluator's hidden task criterion has already been satisfied. Episodes may also terminate after exhausting the high-level or low-level action budget.

Strict Success Rate. SR is the primary task metric. A task succeeds only when the executed claims collectively satisfy the hidden target, ordering, cardinality, and physical-instance constraints defined by the benchmark. For collection tasks, the evaluator first binds claims to physical instances and gives credit only to distinct valid instances; quota tasks require the prescribed number of distinct instances, whereas exhaustive tasks require the full task-defined set. Sequential and return tasks additionally enforce their required ordering.

Oracle Success Rate. OSR is computed on the *same executed trajectory*. It asks whether the trajectory ever satisfies the task's spatial requirements even if the policy fails to issue the corresponding valid completion claim. OSR therefore does not run a separate oracle navigation policy or replace failed actions with an oracle route.

Success weighted by Path Length. For task i,

$$\mathrm{SPL}_i = \mathrm{SR}_i \frac{L_i^{\mathrm{ref}}}{\max(L_i^{\mathrm{agent}}, L_i^{\mathrm{ref}})}, \tag{6}$$

where L_i^{agent} is the length of the actually executed low-level trajectory. The reference length L_i^{ref} follows the task structure: a single-target task uses a start-to-target geodesic; ordered and return tasks use the required sequence of geodesic legs; and collection tasks use the shortest supported route over the required number of valid targets under the task's quota or exhaustive semantics. Thus, the reference path is task-structure dependent rather than a single start-to-goal definition shared by all tasks.

We additionally retain stage-level SR and OSR for multi-stage analysis. For Stage@k, the denominator contains only tasks with at least k required stages; shorter tasks are not padded with failures. Statistical aggregation and uncertainty estimation for all reported metrics are detailed in Section E.

E Statistical Protocol and Complete Results

This appendix reports the statistical protocol and complete quantitative results underlying the comparisons reported in the main paper. The primary evaluation contains 128 tasks from 64 scenes, with two tasks per scene. We evaluate eight native memory conditions and three GT scene-knowledge conditions, yielding 128 complete selected episodes per condition.

Overall SR, OSR, and SPL are task-level macro averages. Because every scene contributes exactly two tasks to the primary benchmark, this weighting also gives every scene equal total weight in the full 128-task aggregate.

E.1 Scene-clustered Paired Uncertainty

For a paired comparison between conditions A and B, we report

$$\widehat{\Delta}_{A\to B} = \mathrm{SR}(B) - \mathrm{SR}(A), \tag{7}$$

in percentage points. Uncertainty is estimated by scene-clustered paired bootstrap. For the primary 128-task benchmark, each bootstrap replicate samples 64 scenes with replacement and retains both tasks from every sampled scene together with their paired outcomes under all compared conditions. If a scene is sampled multiple times, both of its tasks are repeated with the same multiplicity. We then recompute the task-weighted paired SR difference.

We use 100,000 bootstrap replicates with NumPy PCG64 and analysis seed `20260926`. Reported intervals are the 2.5th and 97.5th percentiles of the bootstrap distribution. These are pointwise 95% confidence intervals and are not symmetrized around the observed effect.

The 35 residual GT_JOINT tasks used for the checkpoint and ESM analyses span 28 scenes. For these experiments, we analogously sample 28 complete scenes with replacement while preserving all tasks from each sampled scene and retain the original task-weighted estimand.

For supplementary paired binary tests, we additionally compute two-sided exact McNemar tests on discordant task pairs and an exact scene-level condition-swap test in which the complete A/B labels of a scene are exchanged jointly. McNemar assumes task-level independence and is therefore reported only as a supplement to the scene-clustered analysis. Holm correction is applied separately within the predefined comparison families. Confidence intervals remain pointwise rather than simultaneous, and we do not use significance stars in the reported tables.

E.2 Complete Primary-condition Results

Table 11 reports the complete aggregate results for the eight native and three GT scene-knowledge conditions. These are the same frozen outcomes used in the main-paper analyses.

Table 11: Complete aggregate results for the primary policy. SR and OSR are percentages; SPL is averaged over all 128 tasks. "None" in the episode-memory column means no additional persistent episode history beyond the shared context defined in Section 3.1.

Cond.	Scene memory	Episode memory	Success	SR	OSR	SPL
M0	None	None	54/128	42.19	46.88	0.2640
M1	None	Image	65/128	50.78	53.91	0.3231
M2	None	Text	55/128	42.97	47.66	0.2943
M3	None	Structured	63/128	49.22	53.13	0.3260
M4	Raw-16	None	66/128	51.56	57.03	0.4040
M5	Text	None	84/128	65.63	69.53	0.5181
M6	Structured	None	80/128	62.50	63.28	0.4514
M7	Structured	Structured	81/128	63.28	65.63	0.4633
GT_TEXT	GT Text	None	90/128	70.31	73.44	0.5384
GT_STRUCTURED	GT Structured	None	89/128	69.53	74.22	0.5553
GT_JOINT	GT Structured	Structured	93/128	72.66	74.22	0.5469

Table 12: Success rate (%) by task family. Parentheses denote the number of tasks in each family.

Condition	Direct (37)	Disambig. (42)	Search & Coll. (31)	Multi-stage (18)
M0	62.16	38.10	32.26	27.78
M1	62.16	45.24	45.16	50.00
M2	56.76	42.86	32.26	33.33
M3	62.16	42.86	41.94	50.00
M4	59.46	57.14	45.16	33.33
M5	67.57	78.57	54.84	50.00
M6	67.57	69.05	61.29	38.89
M7	70.27	69.05	51.61	55.56
GT_TEXT	72.97	71.43	70.97	61.11
GT_STRUCTURED	78.38	64.29	67.74	66.67
GT_JOINT	78.38	76.19	74.19	50.00

E.3 Results by Task Family

Table 12 provides the corresponding SR breakdown over the four task families defined in Section 3.2. These subsets contain 18–42 tasks and are therefore reported as descriptive analyses rather than as independent rankings of memory representations or intrinsic task difficulty.

E.4 Paired Memory and Knowledge Effects

Aggregate changes can conceal substantial task-level turnover. We therefore report both the paired SR difference and the numbers of failure-to-success (F→S) and success-to-failure (S→F) transitions. Table 13 includes the episode-memory comparisons emphasized in the main text, the three scene-memory comparisons against M0, and the three matched Native→GT interventions.

The F→S and S→F counts are included to distinguish small net changes from genuinely small behavioral effects. For example, M0→M2 and M6→M7 have near-zero net improvements but contain substantial changes in both directions.

E.5 Checkpoint Intervention Statistics

The checkpoint analysis uses the fixed set of 35 GT_JOINT residual failures spanning 28 scenes. Each case has three matched full-episode continuations from the same checkpoint: B0, Sham, and

Table 13: Paired SR effects. Confidence intervals are scene-clustered percentile bootstrap intervals. F→S and S→F count task-level outcome changes under the paired conditions.

Comparison	ΔSR (pp)	95% scene CI	F→S	S→F
Persistent episode memory				
M0 → M1	+8.59	$[+0.78, +16.41]$	23	12
M0 → M2	+0.78	$[-7.81, +9.38]$	16	15
M0 → M3	+7.03	$[-0.78, +14.84]$	19	10
M6 → M7	+0.78	$[-6.25, +8.59]$	14	13
Scene memory				
M0 → M4	+9.38	$[-0.78, +19.53]$	23	11
M0 → M5	+23.44	$[+13.28, +33.59]$	37	7
M0 → M6	+20.31	$[+9.38, +31.25]$	37	11
Native → GT scene knowledge				
M5 → GT_TEXT	+4.69	$[-3.12, +12.50]$	17	11
M6 → GT_STRUCTURED	+7.03	$[-1.56, +15.62]$	21	12
M7 → GT_JOINT	+9.38	$[+0.00, +18.75]$	24	12

Table 14: Full-episode checkpoint-intervention results on the 35 GT_JOINT residual tasks. Confidence intervals use scene-level bootstrap over the 28 represented scenes.

Condition	Success	SR (%)	95% scene CI
B0	5/35	14.29	$[3.12, 26.47]$
Sham	5/35	14.29	$[3.12, 26.47]$
Targeted	15/35	42.86	$[25.71, 60.61]$

Targeted. All three branches preserve the frozen trajectory prefix and remaining task budget. The detailed intervention and annotation protocol is given in Section F.

Relative to Sham, Targeted changes full-episode SR by $+28.57$ pp, with ten F→S transitions and no S→F transitions. B0 and Sham have identical aggregate SR but differ on two individual tasks, one in each direction.

E.6 ESM Results

We evaluate Experience Skeleton Memory (ESM) at two complementary scales. We first compare Frame, TCoT, and ESM on the same 35 GT_JOINT residual tasks that motivated the diagnosis-driven intervention. We then evaluate ESM over the complete 128-task set to characterize its behavior beyond this difficult residual subset.

Residual-set evaluation. The residual set contains 35 tasks from 28 scenes. Frame, TCoT, and ESM are all rerun from the beginning of the episode rather than resumed from a diagnostic checkpoint. Frame succeeds on 10/35 tasks (28.57%), TCoT on 12/35 (34.29%), and ESM on 22/35 (62.86%).

We use paired scene-level bootstrap with 100,000 resamples (seed `20260926`), preserving all tasks from each sampled scene. Relative to Frame, ESM improves SR by $+34.29$ percentage points (95% CI: $[+14.71, +54.84]$), with 14 failure-to-success transitions and 2 success-to-failure transitions. Relative to TCoT, ESM improves SR by $+28.57$ points (95% CI: $[+7.69, +51.35]$), with 14 failure-to-success and 4 success-to-failure transitions.

The paired results also reveal substantial task-level turnover. For Frame→ESM, 14 tasks change from failure to success and 2 from success to failure, while 8 remain successful and 11 remain failed. For TCoT→ESM, 14 failures are recovered and 4 previous successes regress. Thus, the aggregate ESM improvements are driven by a substantial number of recovered tasks rather than by a small number of isolated changes.

Table 15: Episode-start evaluation on the 35 GT_JOINT residual tasks. Frame and TCoT are alternative episode-memory interfaces, while ESM is the diagnosis-driven geometric memory intervention described in Section G. The ESM interval uses scene-level bootstrap over the 28 represented scenes.

Episode memory	Success	SR (%)	95% scene CI
Frame	10/35	28.57	–
TCoT	12/35	34.29	–
ESM	**22/35**	**62.86**	$[\mathbf{45.95}, \mathbf{79.41}]$

Table 16: Paired residual-set comparisons against ESM. Confidence intervals use paired scene-level bootstrap over the 28 represented scenes. F→S and S→F denote task-level rescue and regression, respectively.

Comparison	ΔSR (pp)	95% paired scene CI	F→S	S→F
Frame → ESM	+34.29	$[+14.71, +54.84]$	14	2
TCoT → ESM	+28.57	$[+7.69, +51.35]$	14	4

Because each residual task has an original diagnostic attribution, we additionally examine recovery by failure category. Table 17 reports the task-level outcomes of Frame and TCoT together with ESM and its scene-bootstrap uncertainty. These subgroup analyses are descriptive: several categories contain only one to three tasks, and the corresponding point differences should not be interpreted as stable rankings of intrinsic category difficulty or as separate confirmatory tests.

The category-level pattern is heterogeneous. Current Localization improves from 3/9 under Frame to 5/9 under TCoT, but remains 5/9 under ESM. In contrast, ESM recovers all three Goal Localization cases and 5/7 Visual Grounding / BBox cases. Because these categories are small, these changes are best treated as descriptive evidence rather than category-specific performance estimates.

The most diagnosis-aligned pattern appears for Egocentric Correspondence. Frame and TCoT each succeed on 3/12 such episodes, whereas ESM succeeds on 7/12. These cases involve difficulty relating remembered or previously executed spatial experience to a concrete transition in the current egocentric observation. ESM directly targets this interface by preserving the geometry of executed experience and re-expressing relevant historical observations with respect to the current viewpoint. The increase from 3/12 to 7/12 is therefore consistent with the type of correspondence failure that motivated the intervention.

The improvement is nevertheless not uniform across the residual taxonomy. ESM does not improve over TCoT on Current Localization, and the single STM Record Error case succeeds under TCoT but not under ESM. The residual analysis therefore supports ESM as a targeted mechanism for making executed experience more spatially usable rather than as a mechanism that resolves every source of navigation failure.

Evaluation over the complete task set. The residual-set evaluation deliberately conditions on failure under GT_JOINT. We therefore also evaluate ESM over the complete 128-task set. ESM succeeds on 22 of the 35 original residual tasks and on 88 of the remaining 93 tasks. This yields 110/128 successes overall, corresponding to an SR of 85.94%. Scene-level bootstrap over all 64 scenes gives a 95% interval of $[79.69, 92.19]$.

The two analyses have different roles. The 35-task comparison is diagnosis-conditioned and provides paired evidence against alternative episode-memory interfaces on the failures that motivated ESM. The complete-set evaluation instead reports ESM behavior over the full task distribution without conditioning on prior GT_JOINT failure. We therefore keep the residual-set recovery rate and complete-set SR separate rather than using the former as an estimate of full-set performance.

Table 17: Frame, TCoT, and ESM outcomes on the 35 residual tasks, grouped by the original diagnostic attribution. Δ_F denotes ESM−Frame and Δ_T denotes ESM−TCoT in percentage points.

Original attribution	Tasks / scenes	Frame	TCoT	ESM	ESM 95% CI	Δ_F / Δ_T (pp)
Current Localization	9 / 9	3/9 (33.33%)	5/9 (55.56%)	**5/9 (55.56%)**	[22.22, 88.89]	+22.22 / +0.00
Goal Localization	3 / 3	1/3 (33.33%)	1/3 (33.33%)	**3/3 (100.00%)**	[100.00, 100.00]†	+66.67 / +66.67
Egocentric Correspondence	12 / 11	3/12 (25.00%)	3/12 (25.00%)	**7/12 (58.33%)**	[28.57, 90.91]	+33.33 / +33.33
Visual Grounding / BBox	7 / 7	2/7 (28.57%)	1/7 (14.29%)	**5/7 (71.43%)**	[42.86, 100.00]	+42.86 / +57.14
Execution	1 / 1	0/1 (0.00%)	1/1 (100.00%)	**1/1 (100.00%)**	–	+100.00 / +0.00
Claim	2 / 2	1/2 (50.00%)	0/2 (0.00%)	**1/2 (50.00%)**	[0.00, 100.00]	+0.00 / +50.00
STM Record Error	1 / 1	0/1 (0.00%)	1/1 (100.00%)	**0/1 (0.00%)**	–	+0.00 / -100.00
Total	**35 / 28**	**10/35 (28.57%)**	**12/35 (34.29%)**	**22/35 (62.86%)**	**[45.95, 79.41]**	**+34.29 / +28.57**

†All three Goal Localization tasks come from distinct scenes and succeed under ESM in every scene-bootstrap resample, yielding a degenerate percentile interval at 100%. Confidence intervals are omitted for categories represented by only one scene.

Table 18: ESM success on the original residual subset and on the complete 128-task evaluation set. The 93-task complement is included to make the composition of the full-set result explicit; only its point estimate is reported.

Task set	N	ESM success	SR (%)	95% scene CI
Original GT_JOINT residual set	35	22	62.86	[45.95, 79.41]
Remaining tasks	93	88	94.62	–
Complete task set	**128**	**110**	**85.94**	**[79.69, 92.19]**

F Residual Failure Diagnosis and Checkpoint Interventions

This appendix details how we diagnose the 35 residual failures under GT_JOINT and how the resulting diagnoses are tested through checkpoint interventions. The goal is not to infer an unobservable internal cause from a failed trajectory, but to identify a concrete, externally verifiable error in the observable memory-to-action process and test whether correcting that error changes subsequent behavior.

F.1 Residual Set and Diagnostic Evidence

The diagnostic set contains all 35 failed GT_JOINT episodes from the primary 128-task evaluation, spanning 28 scenes. No successful GT_JOINT episode is added to this set, and no residual failure is removed based on whether a later intervention succeeds.

For each episode, we reconstruct the decision-level evidence available around the diagnosed failure. The policy-visible record contains the task instruction, current four-view RGB observation, frozen GT scene knowledge, policy-maintained episode memory, shared task state, and immediate interaction feedback. The policy additionally emits explicit diagnostic fields describing its current-location belief, goal-location belief, selected high-level action, and a self-reported rationale for that action.

We jointly inspect these policy-visible signals with system-side records of visual grounding, claim processing, and local execution. For offline audit, we additionally use simulator pose, ground-truth region identity, object identity, task target information, and top-down trajectory views. These privileged signals are used only to determine whether an observable policy belief, correspondence, grounding decision, or execution result is inconsistent with the environment; they are never supplied to the navigation policy during the original evaluation.

The policy's reported localization and rationale are therefore treated as *observable diagnostic signals*. They are neither ground-truth explanations of the episode nor direct access to the model's latent reasoning. Likewise, trajectory symptoms such as backtracking or oscillation are not sufficient on their own to determine the underlying error.

Table 19: Operational taxonomy used for the 35 GT_JOINT residual failures. Categories identify observable failure locations in the closed-loop memory-to-action process rather than assumed internal modules of the model.

Attribution	**Operational criterion**	**Key distinction**
Current Localization	The policy's current place or region belief is inconsistent with independently verified current-location evidence.	A mistaken current belief is separated from an incorrectly identified task goal or an incorrect mapping from a known place to a visible transition.
Goal Localization	The task target or its spatial constraints are mapped to the wrong remembered place.	The agent may correctly identify its current location while reasoning about the wrong destination.
Egocentric Correspondence	Remembered places, connections, or prior experience are incorrectly bound to a concrete entrance, direction, or local region in the current observation.	The relevant scene-level relation may already be available; the failure lies in determining which currently visible transition instantiates it.
Visual Grounding / BBox	A currently visible semantic target is misidentified, or the Grounder binds the requested local target to the wrong image region or physical instance.	A failed final claim is not automatically assigned here; independent visual or bounding-box evidence must identify the grounding discrepancy.
Execution	A local visual target has been selected but the grounding-to-motion boundary fails to produce an appropriate physical transition.	The adopted case lies near the grounding/execution boundary and is therefore not interpreted as evidence for a general low-level planner failure.
Claim	The completion, instance, arrival, or duplicate-instance claim interface is incorrectly bound despite the relevant local evidence.	Claims caused by an earlier visual misidentification remain attributed to the upstream visual failure rather than to the claim interface itself.
STM Record Error	A persistent episode-memory record is incorrect or inadequately maintained, and that stored state is subsequently used by the policy.	A mistaken belief written into memory during the same erroneous decision is not sufficient; there must be evidence that a prior persistent record affects a later decision.

F.2 Operational Failure Taxonomy

We organize the observable memory-to-action process into the operational categories in Table 19. Each residual episode receives one primary attribution corresponding to a concrete discrepancy supported by the decision trace and independent audit evidence.

The adopted distribution is:

$$\begin{aligned} &9 && \text{Current Localization},\\ &3 && \text{Goal Localization},\\ &12 && \text{Egocentric Correspondence},\\ &7 && \text{Visual Grounding / BBox},\\ &1 && \text{Execution},\\ &2 && \text{Claim},\\ &1 && \text{STM Record Error}. \end{aligned} \tag{8}$$

The first three categories together account for $24/35 = 68.6\%$ of the residual set. Thus, most diagnosed failures occur before local visual grounding and physical execution. Egocentric Correspondence is the largest single category, with 12/35 cases. In these episodes, the problem is often not that a relevant place or connection is absent from scene knowledge, but that the policy associates that remembered structure with the wrong currently visible transition.

F.3 Annotation and Attribution Procedure

For each residual episode, attribution proceeds from the task and available memory to the concrete decision at which an independently verifiable local error is identified. We first inspect the policy's

current-location and goal-location beliefs, then compare its selected action and self-reported rationale with the current four-view observation and the supplied scene and episode memory. We subsequently inspect the Grounder output, claim-processing trace, and local execution result. Privileged offline evidence is used only afterward to verify the specific discrepancy.

The adopted checkpoint is a *diagnostically supported local checkpoint*. We do not require it to be the earliest error in the trajectory, and we do not interpret it as a uniquely identified latent root cause. Earlier errors may exist, and multiple failures may interact within the same episode. The diagnosis instead establishes that a particular observable variable is incorrect at a recoverable decision state and can be corrected without replacing the full navigation policy.

Two additional annotators independently reviewed all 35 residual episodes. Each agreed with the adopted primary attribution on 34/35 cases (97.1%), corresponding to pairwise Cohen's $\kappa = 0.962$. This agreement supports the reproducibility of the operational categories while leaving the final analysis tied to the concrete evidence stored for each episode.

F.4 Checkpoint Intervention Protocol

An observational attribution does not by itself establish that the diagnosed error matters for downstream behavior. We therefore restore each episode immediately before its selected diagnostic checkpoint and construct three matched continuations:

$$\text{B0}, \qquad \text{Sham}, \qquad \text{Targeted}. \tag{9}$$

All branches share the same executed prefix, task, checkpoint, policy-visible state before intervention, current RGB observation, policy and Grounder configuration, executor, and remaining high- and low-level budgets. Across the 35 cases, all 105 formal continuations pass the stored checkpoint-restoration checks for current RGB, prompt state, working memory, task bookkeeping, recent grounding information, and remaining budget.

B0. B0 restores the checkpoint and continues the original policy without adding corrective information. It is an unmodified continuation from the same restored historical state and is not required to reproduce the exact original future trajectory.

Targeted. Targeted applies one evidence-supported local correction at the selected checkpoint. The delivery channel depends on the diagnosed variable.

For 30/35 cases, the correction is an *evidence note*. A single `<checkpoint_evidence>` block containing a verified local fact is appended immediately before policy generation at that decision. Examples include correcting the current place identity, distinguishing two visible doorways, or disambiguating a currently visible object. The note is injected only at the selected decision.

For 4/35 cases, the diagnosed error lies in persistent structured episode state. These interventions use the same guarded state editor used by the checkpoint framework to modify the relevant allowed field before the policy prompt is rebuilt. The four state-patch cases are CKPS04, CKPS10, CKPS31, and CKPS38.

The remaining case, CKPS27, lies at the grounding–execution boundary. After the policy has selected the local target and the Grounder has produced its box, Targeted replaces the unusable Grounder box with an evidence-supported local box in the same view before execution. The policy's high-level target, direction, distance bin, and task reasoning are not replaced.

These intervention types are therefore collectively described as *one-shot evidence-supported local corrections*, rather than as a single uniform STM-editing operation.

Matched Sham. Sham is applied at the same checkpoint, through the same intervention channel and timing, but without supplying the diagnosed corrective information. Evidence-note cases receive a neutral description of current visual content; state-patch cases use the same guarded editing pathway without introducing the target correction; and CKPS27 uses the same bounding-box edit pathway while retaining the branch's original Grounder box.

The matching is thus by checkpoint, intervention channel, timing, and surrounding execution protocol. We do not claim exact token- or character-length matching between Targeted and Sham prompts.

Table 20: Full-episode outcomes of the matched checkpoint continuations, grouped by the adopted primary attribution. Counts denote successful continuations within each diagnostic category.

Attribution	N	B0	Sham	Targeted
Current Localization	9	0/9	0/9	4/9
Goal Localization	3	1/3	1/3	2/3
Egocentric Correspondence	12	1/12	1/12	3/12
Visual Grounding / BBox	7	2/7	2/7	3/7
Execution	1	0/1	0/1	1/1
Claim	2	1/2	1/2	1/2
STM Record Error	1	0/1	0/1	1/1
Total	**35**	**5/35**	**5/35**	**15/35**

One-shot semantics. "One-shot" refers to the number of *external interventions*: exactly one correction is applied at the selected checkpoint, after which the original policy controls the remainder of the episode without a second child repair. It does not imply that the correction can influence only one subsequent decision. A structured state edit may persist through later episode memory, and a one-time evidence note may itself be incorporated by the policy into a subsequent state update. No external process repeatedly maintains the correction.

The primary checkpoint outcome is the final success of the complete remaining episode. The 15/35 Targeted successes therefore denote full-task recovery, not merely a correct next action or short-window local recovery.

F.5 Attribution-specific Intervention Outcomes

Table 20 reports full-episode outcomes by diagnosed category. Aggregate uncertainty and paired statistical tests for these three branches are reported in Section E.5.

Targeted correction changes the final outcome of a substantial subset of the residual episodes, whereas B0 and Sham have the same aggregate success count. The paired results in Section E.5 show ten failure-to-success transitions from Sham to Targeted and no success-to-failure transitions. These interventions therefore provide behavioral evidence that the diagnosed local errors are consequential at the selected checkpoints rather than merely correlates of failed trajectories.

At the same time, correcting a supported local error once recovers only 15/35 episodes. The remaining trajectories may encounter a new error later, lose a previously corrected localization, or establish another inconsistent correspondence as the agent moves and receives new observations. This is particularly visible for Egocentric Correspondence: although it is the largest residual category, only 3/12 episodes reach full task success after the one-shot Targeted correction.

The checkpoint analysis therefore separates two questions. First, the Targeted–Sham comparison asks whether a diagnosed local error is behaviorally relevant. Second, incomplete full-episode recovery asks whether the policy can continue to maintain and re-establish the required situated information as its observation changes. The latter motivates the diagnosis-driven memory intervention described in Section G.

G Experience Skeleton Memory

The residual diagnosis in Section F suggests that retaining environmental information is not sufficient when the policy cannot reliably situate past experience with respect to its changing observation. Experience Skeleton Memory (ESM) is designed around this distinction. Rather than storing task-time experience only as images, textual events, or a semantic place ledger, ESM preserves the geometry of the agent's *actually executed experience*, attaches revisable semantic interpretations to that experience, and re-expresses relevant historical observations relative to the agent's current egocentric view.

Table 21: Main geometric parameters of the implemented Experience Skeleton Memory. Distances are measured in the simulator metric frame.

Parameter	Value	Role
Skeleton snap radius	1.0 m	Assign a translated trajectory sample to a nearby compatible node.
Heading-axis tolerance	40°	Require compatible local motion axes when reusing a skeleton node.
Graph stretch	1.25	Reuse an existing skeleton path when its graph distance is within $1.25\times$ the direct node-center distance; otherwise add an edge.
Observation-anchor merge radius	1.0 m	Associate nearby observations only when they share both the same semantic hypothesis and the same skeleton node.
Local route support	3.0 m	Limit the actually traversed route segment used for local entrance support.
Entrance readout distance	0.75 m	Determine the local direction of an experienced route entrance.
Depth occlusion tolerance	0.05 m	Determine whether a projected historical standing point is consistent with the current depth observation.
Recent observation sequence	16 IDs	Maintain a compact recent-order index without deleting older stored observations or geometry.

In the ESM experiments, the frozen scene-memory component remains the same GT structured scene knowledge used by GT_JOINT. ESM replaces the persistent task-time memory interface. Its implementation consists of three coupled layers:

1. an *executed route skeleton* constructed from the trajectory that the agent has physically traversed;
2. *revisable semantic hypotheses* and observation anchors associated with this executed experience; and
3. an *egocentric rendering* that recomputes how historical experience relates to the current camera view at every decision.

This separation keeps geometric experience persistent while allowing its semantic interpretation to change when subsequent observations contradict an earlier belief.

G.1 Executed Experience Geometry

ESM maintains an `OnlineTrajectorySkeleton` from the actual low-level trajectory produced by the shared executor. For each completed execution interval, the system records the raw metric trajectory samples and incrementally updates a compact graph containing spatial nodes, edges, and bindings from raw samples to graph nodes.

The skeleton nodes are *not* high-level decision nodes. They are online clusters of positions along the executed trajectory. When a new translated trajectory sample arrives, ESM either assigns it to a compatible existing node or creates a new one. Compatibility depends on both spatial proximity and the axis of local motion. Reverse traversal along the same motion axis can reuse an existing node.

When consecutive samples are assigned to different nodes, ESM either adds a new graph edge or reuses an existing path according to a graph-stretch criterion. The graph therefore acts as a compact summary of experienced geometry rather than as an exact replacement for the raw trajectory. The raw executor samples remain stored separately, and local route evidence used by the memory can be recovered from these actually traversed samples.

Each raw execution interval preserves the actual sequence of positions reached during that high-level action, including partial movement. Rotation-only or zero-translation samples do not create new skeleton nodes or edges. The resulting graph edges connect node centers and should therefore be

understood as a compact geometric summary; the stored raw trajectory remains the direct record of the path that was executed.

ESM does not introduce untraversed geometry into this skeleton. Its internal graph search operates only over the skeleton constructed from experienced motion. In particular, the memory does not query the simulator navmesh to complete unseen connections or construct a task-level shortest route.

G.2 Revisable Semantic Experience

Geometric experience alone does not determine what semantic place the agent was traversing. ESM therefore separates the persistent geometric skeleton from policy-generated semantic hypotheses.

A region hypothesis H_i stores a current semantic label together with its label history. The policy maintains these hypotheses through two explicit operations:

- `BEGIN_REGION(`B`)` declares that the most recently executed transition entered a new region hypothesized as B;
- `RELABEL_REGION(`H_i, B`)` revises the semantic interpretation of an already experienced hypothesis H_i to B.

A change in the policy's free-form localization text does not by itself create a region transition. A new boundary is recorded only when the policy explicitly issues `BEGIN_REGION`. The system checks that the referenced update is well formed and that the preceding execution interval contains physical translation, but it does not query the GT current region to determine whether a semantic boundary was actually crossed.

This design makes the geometry and semantics evolve differently. Once executed, the geometric trajectory is preserved. A later `RELABEL_REGION` changes the interpretation of the corresponding experience without rewriting where the robot actually moved. Stored transition labels and observation labels are then derived from the updated hypothesis references.

For example, if an experienced segment was initially represented as

$$H_7 = P_{11},$$

the policy may later issue

$$\texttt{RELABEL_REGION}(H_7, P_{16}),$$

after acquiring evidence that the earlier semantic interpretation was incorrect. The executed route associated with H_7 remains unchanged; only its semantic interpretation is revised.

Observation anchors. At each decision, the policy may also provide a semantic description of its current observation. The system associates this description with a historical *standing-position anchor*. An anchor records where the agent stood when the observation was made, rather than estimating the metric location of every object or region mentioned in the description.

Nearby observations are merged only when they belong to the same semantic hypothesis, map to the same skeleton node, and lie within 1 m of the original anchor. The policy may subsequently revise or delete an observation statement. These semantic edits do not modify the underlying standing pose or executed trajectory.

Transitions. Semantic boundaries retain references to the source and destination hypotheses. Consequently, if a hypothesis is relabeled, the semantic labels shown for its associated transitions are recomputed from the revised hypotheses. ESM therefore preserves the fact that an experienced transition occurred while allowing the policy to revise what that transition connected.

G.3 Egocentric Re-alignment

The third layer converts stored experience into information expressed relative to the agent's *current* situation. Internally, ESM uses the exact simulator pose associated with the executed trajectory together with the rendered camera calibration and current depth observation. These signals are used

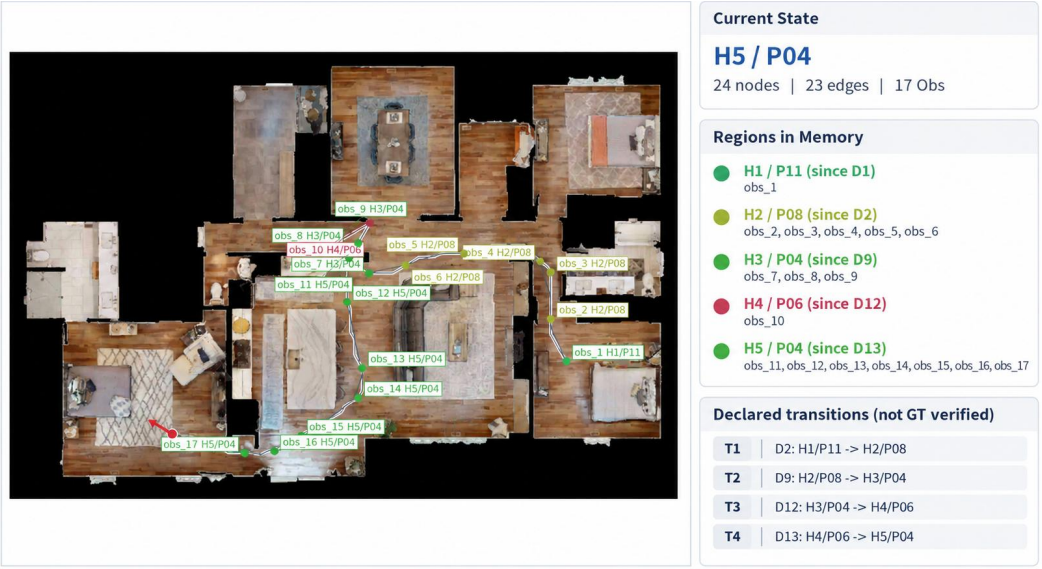

Figure 5: **Example stored ESM state after a successful trajectory.** The visualization shows the executed route skeleton together with policy-maintained region hypotheses, observation anchors, and declared semantic transitions at decision 21. The top-down view is used only to visualize and audit the stored ESM state; it is never provided to the navigation policy. Region labels and transitions are policy-generated hypotheses rather than GT semantic annotations.

as execution-side geometry; raw coordinates and camera matrices are not presented directly to the navigation policy.

For a historical standing point p_h and current agent pose (p_t, q_t), its agent-centric coordinates are computed as

$$p_{\text{local}} = R(q_t)^\top (p_h - p_t), \tag{10}$$

and converted to right, forward, and vertical components by

$$(r, f, u) = (p^x_{\text{local}}, -p^z_{\text{local}}, p^y_{\text{local}}) \,. \tag{11}$$

ESM discretizes the horizontal bearing into eight egocentric sectors:

$$\{\text{forward}, \text{forward-right}, \text{right}, \text{behind-right}, \text{behind}, \text{behind-left}, \text{left}, \text{forward-left}\}.$$

Historical anchors within 1 m are additionally described as *near*; larger distances are described as *farther*. Vertical displacement greater than 1 m is represented as *above* or *below*.

To determine whether a historical standing position is directly visible in a current image, the point is projected using the actual optical camera pose:

$$p_{\text{cam}} = R^\top_{\text{cam}} (p_h - p_{\text{cam}}). \tag{12}$$

For camera depth $d = -p^z_{\text{cam}}$ and focal length

$$f_{\text{pix}} = \frac{W}{2\tan(\text{HFOV}/2)},$$

the image coordinates are

$$u = \frac{W-1}{2} + f_{\text{pix}} \frac{p^x_{\text{cam}}}{d}, \qquad v = \frac{H-1}{2} - f_{\text{pix}} \frac{p^y_{\text{cam}}}{d}. \tag{13}$$

A historical anchor is rendered as directly visible only if it projects in front of and inside the current image and its projected depth is consistent with the current depth map within a 5 cm tolerance.

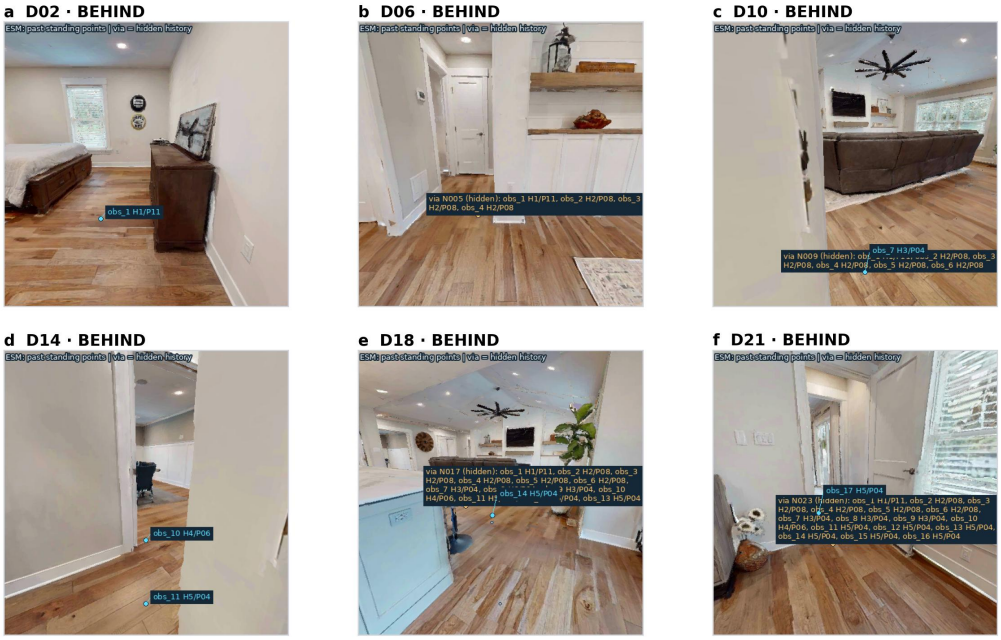

Figure 6: **Policy-facing ESM overlays along one successful trajectory.** Representative decisions show how previously acquired observations are re-expressed relative to the current egocentric view as the agent moves. Cyan markers denote directly associated historical standing observations, while *via* annotations associate occluded historical experience with visible anchors on previously executed routes. The panels preserve the actual runtime overlays presented to the policy.

When a historical observation is occluded, ESM does not simply attach it to an arbitrary visible doorway. Instead, it searches the already experienced skeleton path between the current area and that historical anchor and may associate the hidden experience with the last visible skeleton anchor before the occluded portion. If no such visible association can be established, the historical observation remains explicitly unassociated.

The final policy-facing visual representation therefore contains historical standing anchors and these hidden-history associations overlaid on the current four egocentric RGB observations. It does not expose a top-down map or draw a continuous global route polyline over the images. Complementary textual state provides the current relative directions of stored observations and the directions of entrances associated with previously executed route segments.

G.4 Policy Interface and Update Cycle

At every high-level decision, the navigation policy receives the same frozen GT structured scene memory used by the corresponding GT condition, the common one-step interaction feedback described in Section D.5, and the current ESM state.

The ESM-specific input contains:

- current region hypotheses and their latest semantic labels;
- stored observation semantics attached to historical standing points;
- current egocentric directions of those historical observations;
- semantic transitions induced by previously declared region changes;
- experienced-route entrance directions derived from executed geometry;
- visibility associations between current views and historical observation anchors; and
- the four current RGB views augmented with ESM annotations.

Table 22: Division of responsibility in ESM. Metric signals are used internally to maintain and re-express executed experience; they are not exposed as raw localization or route answers to the policy.

Component	Policy responsibility	System responsibility
Current semantic region	Infer current place and issue region updates	Store hypotheses; no GT semantic correction
Semantic revision	Decide when and how to relabel a hypothesis	Validate references and maintain label history
Executed geometry	Select high-level local actions	Record actual executor trajectory and update skeleton
Metric pose	Not estimated or output as raw coordinates	Read simulator pose for executed geometry and transforms
Observation semantics	Describe and revise experienced observations	Associate them with standing-position anchors
Region transition	Declare semantic crossing with `BEGIN_REGION`	Preserve boundary references and derive current labels
Egocentric correspondence	Interpret rendered historical evidence for decision making	Transform, project, and depth-test historical anchors
Goal reasoning	Resolve instruction, destination, and next action	No ESM task-level route computation

The policy remains responsible for semantic interpretation. Its output contains the shared current- and target-localization fields and action, plus ESM state updates such as region and observation operations. ESM also requires an exploration assessment over the four directions LEFT, FORWARD, RIGHT, and BEHIND. Each direction is classified as `visited`, `partially_visited`, `no_visit_record`, or `uncertain`. When the policy chooses a previously visited route despite an unresolved or apparently unvisited alternative, it is required to state a revisit reason.

These assessments remain policy judgments. The parser validates their structure but does not replace them with a geometric frontier oracle or ground-truth exploration label.

After the policy acts, the shared Grounder and executor carry out the local decision. The system then ingests the actual executed trajectory interval into the skeleton, associates new observations with their experienced standing positions, and uses the resulting state to construct the next decision's egocentric memory.

G.5 System and Policy Responsibilities

Table 22 summarizes the division of responsibility. ESM deliberately uses privileged metric geometry on the *system side* to preserve the geometry of executed experience, while leaving scene interpretation and task-level reasoning to the policy.

More specifically, ESM internally uses simulator current and historical poses, camera calibration, and current depth to maintain executed geometry and compute relative correspondence. The policy receives only derived relative directions, semantic memory, and image annotations rather than absolute XYZ coordinates, quaternions, camera matrices, or depth arrays.

The current GT semantic region is not used to update ESM hypotheses. Likewise, the ESM skeleton is not initialized from the canonical GT scene graph and does not use navmesh connectivity to add routes that the agent has not traversed. The shared GT structured scene memory remains available as the frozen scene-memory condition, but relating those scene facts to current experience remains a policy decision.

G.6 Implementation Scope and Failure Boundaries

ESM stabilizes the geometry of *experienced* motion, but this does not make its internal graph a complete navigability map. Skeleton nodes are online position clusters, and graph edges connect cluster centers; the graph does not perform a separate wall-intersection or floor-topology validation.

The raw executed trajectory is retained precisely because the compact skeleton is an index and structural summary rather than a replacement for execution evidence.

Similarly, ESM's semantic state can remain wrong. Region boundaries are declared by the policy, semantic labels may be misassigned, and an explicit relabel operation is itself a policy decision. The system preserves the geometric experience against such semantic changes but does not decide the correct region identity on the policy's behalf.

ESM also does not provide an automatic exhaustive frontier map. Its prompt explicitly asks the policy to reason about visited, partially visited, and unvisited directions and to justify revisits, but these assessments are not verified by a complete free-space exploration model. Consequently, ESM can make already acquired experience easier to situate and reuse while still leaving unresolved whether an unseen corner, occluded entrance, or untraversed branch contains the required continuation.

Finally, the visual correspondence layer projects *historical standing positions*, not automatically localized historical objects. An observation anchor indicates where the agent acquired an experience; the semantic content attached to that anchor remains model-generated. ESM can therefore preserve the spatial relation between past experience and the current viewpoint without guaranteeing that every stored semantic interpretation or current visual grounding is correct.

Quantitative evaluation is reported in Section E.6. We evaluate ESM both on the 35 diagnosis-conditioned residual tasks, where Frame and TCoT provide matched episode-start baselines, and on the complete 128-task evaluation set. This separates the targeted question of whether ESM helps on the failures that motivated its design from its behavior over the full task distribution.

H Prompts, Configuration, and Reproducibility

This appendix summarizes the implementation details needed to reproduce the reported evaluation without repeating the memory definitions, agent design, or statistical protocol described in the preceding appendices. We report the actual prompt interfaces, model and simulator configurations, inference resource usage, and artifact provenance used by the selected experimental runs.

H.1 Experimental Scope and Provenance

The primary evaluation contains M0–M7 and the three GT scene-knowledge conditions, with 128 selected task outcomes per condition. The diagnostic experiments additionally include the 35 residual tasks, their B0/Sham/Targeted checkpoint continuations, Frame and TCoT reruns, and ESM evaluation over the complete 128-task set.

We additionally evaluate a second navigation policy using the same shared agent interface defined in Section D. This focused cross-policy replication evaluates M0, M7, and GT_JOINT on the same 128 tasks, while retaining the same observation, action, Grounder, executor, and evaluation interfaces. The corresponding Policy-2 results are reported in Table 2.

The authoritative provenance order is the selected episode or intervention index, followed by its saved configuration, prompt log, and completion receipt, followed by the corresponding frozen manifest or code snapshot. The current repository HEAD alone is not treated as a complete identifier of historical runs because selected results include preserved trajectories, checkpoint continuations, and runs produced under multiple frozen execution snapshots. For the completed full-set ESM evaluation, the 35 residual runs and the additional 93 runs are linked through their selected-source indices.

H.2 Prompt and Memory Interfaces

Every navigation condition shares the same high-level task and action contract. The policy receives the natural-language goal, immediate review of the previous decision when available, permitted scene and episode memory, shared task state, and four current egocentric RGB observations. The final request also includes the shared action and claim protocol. Actual saved request messages, rather than only the static prompt source, are treated as the authoritative representation of model input.

Condition-specific prompts differ only in the memory interface they expose. M0, M4, M5, and M6 use the current-review interface with, respectively, no scene memory, Raw-16, Na-

Table 23: Main inference configurations. Saved model names denote the aliases used by the evaluation logs rather than immutable model-weight identifiers.

Component	Model	Temp.	Reasoning	Max tokens
Primary navigation / Frame / TCoT / ESM	GPT-5.6 Terra	0	medium	8192
Navigation / claim Grounder	GPT-5.6 Terra	0	medium	4096
Native Text builder	GPT-5.6 Terra	0	disabled	4096 per chunk
Native Structured builder	GPT-5.6 Terra	0	low	8192
Policy-2 navigation	GPT-6 Luna	0	medium	8192

tive Text, and Native Structured scene memory. M1 and Frame use the sampled visual-history interface (`sampled_visual_history_v5`); M2 and TCoT use the textual event-log interface (`text_event_log_v3`); and M3, M7, and GT_JOINT use the structured place ledger (`cot_full_place_ledger_v9`). GT_TEXT and GT_STRUCTURED use the current-review interface with their corresponding frozen GT scene memories. ESM uses `ego4v_bp_experience_skeleton` with `experience_skeleton_v3` and GT Structured scene memory.

Although some historical configuration metadata labels the structured episode-memory schema as v8, the policy-visible state and saved decision logs used by M3, M7, and GT_JOINT follow the v9 interface; we therefore report v9 throughout the manuscript.

Native Text memory is constructed by GPT-5.6 Terra from ordered visual experience, while Native Structured memory is compiled from the frozen Native Text artifact rather than independently reconstructed from RGB. GT Text and GT Structured memories are deterministic renderings of the audited canonical scene facts. At task time, Frame stores a bounded sampled visual history, TCoT stores policy-written textual events, Structured memory applies policy-generated state deltas, and ESM uses the update operations defined in Section G.

H.3 Grounding and Intervention Interfaces

The navigation Grounder receives one policy-selected 480×480 RGB view together with the local target description, spatial hint, and distance bin, and returns an image-space bounding box. Object and region claims use separate visual-verification prompts; locally accepted claims enter the shared task state, while approach or rejected claims do not.

The checkpoint study uses three actual intervention channels: 30 evidence-note cases, four guarded structured-state patches, and one grounding–execution boundary patch. Targeted and Sham branches use the same checkpoint, intervention channel, and timing; the Sham omits the diagnosed corrective information. Full intervention semantics are described in Section F.4.

ESM uses the shared navigation action interface. Its additional policy output contains region and observation updates such as `BEGIN_REGION`, `RELABEL_REGION`, `SET_OBSERVATION`, and `DELETE_OBSERVATION`; executed geometry and egocentric rendering remain system-maintained.

H.4 Model and Simulator Configuration

The simulator uses four current 480×480 RGB views with approximately 90° horizontal fields of view. Primitive forward motion is 0.25 m and turning is 30°. The shared local executor uses the oracle navmesh-based planner described in Section D.4, with a 15-step local tracking cap and a 0.75 m reach radius. Task-level decision budgets are 30, 45, or 60, with an additional 900 low-level environment-action budget. Formal evaluation uses environment seed 42.

H.5 Inference and Experimental Cost

We report inference cost for the selected runs used by the current manuscript. The values are reconstructed from logged token usage and fixed local per-million-token prices; they are *estimated*

Table 24: Inference usage for the experiments currently reported in the manuscript. ESM combines the residual 35 tasks and the additional 93 full-set runs. Policy-2 includes the focused M0, M7, and GT_JOINT replication.

Experiment	Outcomes	Policy calls	Grounder calls	Input / output tokens	Cost (USD)
Native M0–M7	1,024	30,474	29,016	427.03M / 31.41M	$1,021.01
GT conditions	384	9,640	9,338	129.52M / 10.24M	$316.54
CKP B0 / Sham / Targeted	105 cont.	2,378	2,337	39.41M / 3.07M	$98.79
Frame-35	35	1,318	1,217	21.41M / 1.27M	$48.74
TCoT-35	35	1,218	1,119	17.86M / 1.26M	$42.58
ESM-128	128	3,144	3,127	53.21M / 4.72M	$141.42
Policy-2	384	15,181	13,933	196.21M / 12.81M	$83.02
Total	**1,990 + 105 cont.**	**63,353**	**60,087**	**884.65M / 64.79M**	**$1,752.09**

Table 25: Authoritative artifacts used to reproduce and audit the study.

Layer	Authoritative artifact	Verification record
Tasks	Frozen task manifest, scene IDs, instructions, and inputs	Input and selection hashes
Memory	Frozen native memory artifacts and GT canonical/rendered memories	Builder configuration and source/canonical hashes
Runs	Selected episode, saved configuration, prompt log, and execution receipt	Per-episode hashes and completion records
CKP / ESM	Selected intervention revision or ESM result index	Restore/intervention receipts and source hashes
Scores	Selected episode plus adopted scoring overlay	Frozen evaluator protocol and score record
Statistics	Selected-source index and analysis scripts	Scene pairing, bootstrap seed, and generated tables

inference costs, not provider invoices or total project expenditure. Development runs, discarded attempts, memory-construction calls, and unlogged HTTP-level retries are not included.

The pricing reconstruction uses the recorded local rates for Terra rather than a live provider price. Accordingly, the table is intended to characterize the resource scale of the reported evaluation, not the amount billed for the project.

H.6 Reproduction and Artifact Provenance

Reproduction should begin from the frozen task and memory artifacts rather than from current command-line defaults. The main execution entry points cover native evaluation, GT evaluation, residual Frame/TCoT reruns, ESM, checkpoint replay, rescoring, and paired scene-bootstrap analysis.