RESIGHT: TASK-DRIVEN SELF-IMPROVING MEMORY FOR EMBODIED AGENTS

Anonymous authors
Paper under double-blind review

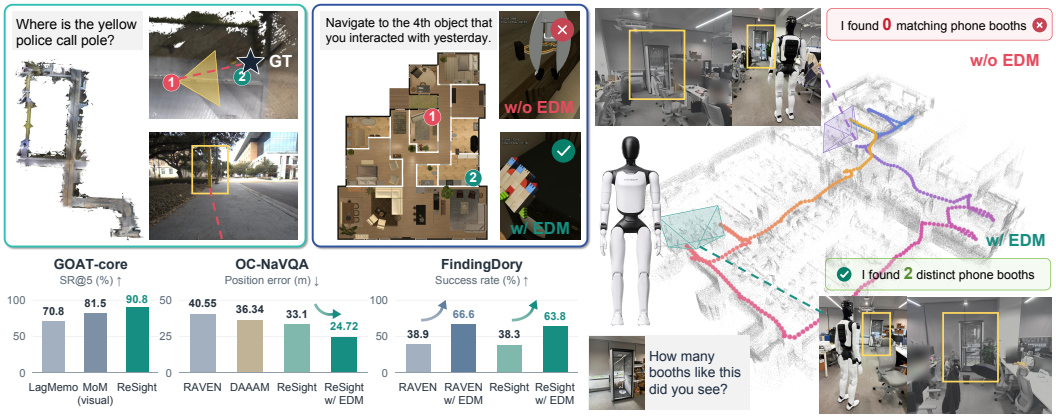


Figure 1: ReSight retains revisitable visual evidence and expands its processing capabilities through EDM. Evaluation across scan-based indoor simulation (GOAT-core), interactive household simulation (FindingDory), real campus recordings (OC-NaVQA), and real-robot office patrols demonstrates effective memory retrieval and task-driven capability expansion.

ABSTRACT

Robots can use historical observations to answer spatiotemporal queries in familiar environments, but the presence of relevant evidence does not imply that existing tools support the operations a task requires. We introduce ReSight, a task-driven self-improving memory system for embodied agents that expands the capabilities for processing and accessing historical observations as task demands arise. ReSight retains keyframes and their spatiotemporal associations, separating observation evidence from extensible derived fields so that stored observations can be reprocessed for new needs. An execution agent and a coding agent share this memory through different interfaces: the former calls registered tools within a bounded online budget, while the latter uses a programmable workspace and a larger exploration budget to inspect evidence, write programs, and acquire models as needed. To realize capability expansion in ReSight, we develop Explore–Distill–Mount (EDM), which identifies capability gaps from execution and exploration records without task reference answers and converts exploration-derived solutions into reusable memory builders, query tools, and skills, extending memory representations and access capabilities as needed. Experiments on three public benchmarks demonstrate effective base memory access and capability expansion. With the execution model and online budget held fixed within each benchmark, EDM reduces mean position error on OC-NaVQA from 33.10 m to 24.72 m and improves high-level success on FindingDory from 38.3% to 63.8%. With the tools and skill frozen, success still improves by 28.3 percentage points on FindingDory episodes excluded from exploration and distillation. Experiments on real robot patrol memories further demonstrate reuse of these capabilities for subsequent queries and new memories, with fewer online tool calls on average.

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.

1 INTRODUCTION

Long-horizon robot tasks such as fetching and delivery often begin with a decision about where to go to satisfy a request. In previously visited environments, past observations can inform this choice. Finding emergency equipment requires recalling where it was seen; finding an object that has moved also requires distinguishing observations from different times. Such place selection depends on spatiotemporal question answering over memory, whose outputs can guide subsequent navigation and manipulation. We study tasks whose relevant evidence is already present in retained observations: how can an agent find, interpret, and combine that evidence to answer the current query?

Embodied memory takes several forms, including scene graphs for structured queries and visual memories associated with time and position (Gorlo et al., 2026a; Hu et al., 2026). Retained keyframes also allow object localization and representation construction to be deferred until query time (Zhou et al., 2026; Maggio et al., 2026). These approaches provide a basis for finding relevant observations, but the computation required by a task may exceed the capabilities of the available tools. Semantic retrieval may return images of a corridor, while distinguishing similar doors requires reading their signs. After retrieving several observations of an object, determining where it stayed longest requires organizing events across time. A task exposes a *capability gap* when the execution agent cannot readily perform these operations with its existing tools and online budget.

We introduce ReSight, which combines revisitable spatiotemporal evidence with extensible memory capabilities. Unlike a fixed memory interface, ReSight can later acquire operations such as text reading, interaction-history reasoning, and object localization by reprocessing observations that are already stored. Its *spatiotemporal evidence memory* retains selected keyframes and their temporal associations, together with available observations such as poses and depth. Scene and object descriptions, along with visual and text features, form initial semantic fields for retrieval. Subsequent needs can prompt new processing of stored images to derive text, event, or geometric fields while preserving their links to source observations. This *late binding* allows interpretation and computation to evolve after observations have been stored, and lets memory builders and query tools be developed together for a new task requirement.

ReSight separates online response from capability development through two agents (Figure 2). The execution agent treats the evidence memory as a database accessed through registered tools, using skills to guide retrieval, inspection, and answering within a bounded online budget. The coding agent treats the same memory as a programmable workspace: it can inspect files and images, write and execute code, and download and install models and dependencies. A separate, larger exploration budget allows it to develop solutions for requirements that are difficult for the current executor. The two agents work with the same evidence, carrying out bounded execution and open-ended exploration, respectively, providing a basis for turning exploration-derived solutions into reusable capabilities for the executor.

ReSight implements task-driven capability expansion through Explore–Distill–Mount (EDM). Prior work uses execution traces to improve programs, skills, and memory-management mechanisms (Lu et al., 2026; Lee et al., 2026; Zhang et al., 2025b). EDM combines execution-side capability gaps with coding-agent exploration traces to extract reusable tools, skills, and memory builders for retained sensor evidence. Builders reprocess stored observations to produce persistent fields and indices; corresponding query tools and skills are mounted on the execution agent so it can use the new capabilities.

Our contributions are as follows:

- We introduce ReSight, which retains revisitable spatiotemporal observations and provides distinct access interfaces for execution and coding agents, allowing historical observations to be reprocessed for new task demands and memory representations and access capabilities to be extended as needed.
- We propose Explore–Distill–Mount (EDM), which identifies capability gaps from execution and exploration records without task reference answers and converts the coding agent's exploration-derived solutions into reusable tools, skills, and memory builders.
- We evaluate ReSight's base memory access, task-driven capability expansion, and frozen-capability reuse on three public benchmarks and real robot patrol memories. With the

execution model and online budget fixed within each benchmark, EDM improves spatial localization on OC-NaVQA and question answering on FindingDory. With the tools and skill frozen, success still improves by 28.3 percentage points on FindingDory episodes excluded from exploration and distillation. Real-robot experiments further demonstrate reuse of these capabilities for subsequent queries and new memories.

2 Related Work

2.1 Spatial Representations and Target Localization

Open-vocabulary maps associate language-aligned features with spatial locations for text queries and target localization (Huang et al., 2023; Peng et al., 2023; Jatavallabhula et al., 2023). Continuous semantic fields, multimodal feature memories, and 3D object instances offer alternative representations (Shafiullah et al., 2023; Zou et al., 2025; Takmaz et al., 2023). Scene graphs organize object relations and spatial hierarchies (Gu et al., 2023; Werby et al., 2024), while dynamic point clouds, spatiotemporal maps, and description histories capture environmental changes (Liu et al., 2025; Schmid et al., 2024; Gorlo et al., 2026a). Task-driven representations adjust semantic granularity according to predefined tasks or subsequent queries (Maggio et al., 2024; 2026).

2.2 Long-Horizon Visual Memory and Retrieval

Observation descriptions, event records, and object tracks can form queryable long-term memories (Anwar et al., 2024; Xie et al., 2024; Fan et al., 2024); multi-view snapshots, visual embeddings, and keyframe-augmented scene graphs retain visual context (Yang et al., 2024; Hu et al., 2026; Werby et al., 2026). Semantic-spatial retrieval and memory fusion further select and integrate relevant information (Mao et al., 2025; Hu et al., 2025b). Recorded tours can supply goal frames for navigation (Chiang et al., 2024), and retrieved RGB-D keyframes support on-demand target localization (Zhou et al., 2026); prediction error can guide which events are stored (Gorlo et al., 2026b). ReSight retains sensor observations as reprocessable evidence and adds structured fields as subsequent needs arise, allowing memory processing and access capabilities to expand.

2.3 Programmatic Problem Solving and Harness Evolution

Programmatic agents compose perception, control, and computation through code (Liang et al., 2023; Surís et al., 2023), amortize generation costs through tool reuse (Cai et al., 2024), and accumulate skills from execution feedback (Wang et al., 2023; Lu et al., 2026). Automatic improvement spans prompts and experiential contexts (Agrawal et al., 2026; Zhang et al., 2025d), as well as agent programs and their organization (Hu et al., 2025a; Robeyns et al., 2025; Zhang et al., 2025c); synthesized code can also constrain action selection or implement policies directly (Lou et al., 2026). Recent harness evolution draws improvement signals from execution traces and demonstrations (Lee et al., 2026; Lin et al., 2026; Zhang et al., 2026a; Che et al., 2026), updating tools, runtime mechanisms, and component organization (Chen et al., 2026; Zhang et al., 2026b), as well as memory construction and retrieval programs (Zhang et al., 2025b). EDM synthesizes tools, skills, and memory builders from execution-side capability gaps and coding-agent exploration traces, extending retained sensor evidence with spatiotemporal fields and access capabilities reusable during online execution.

3 Method

ReSight provides a shared spatiotemporal evidence memory with extensible derived fields, accessed through asymmetric interfaces for execution and coding agents. EDM implements capability expansion within this architecture (Figure 2).

3.1 Problem Setting

Given the robot's accumulated observation memory $\mathcal{M}$ and a task query q, the system returns an answer y and supporting evidence E_q. Task-location queries additionally require a position or can-

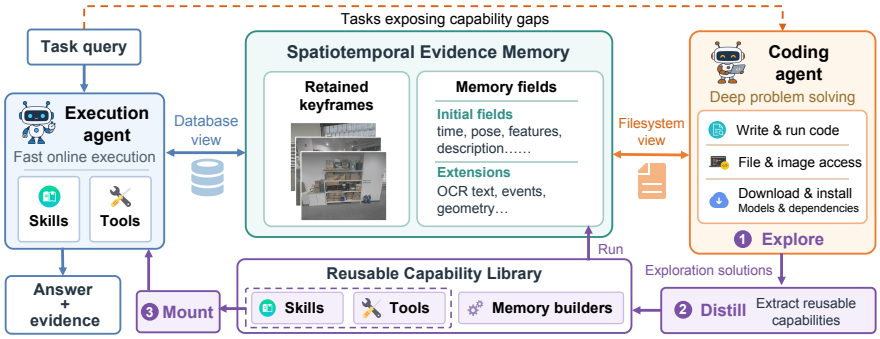


Figure 2: ReSight architecture. The two agents access the same evidence memory through different interfaces. EDM converts exploration-derived solutions into tools, skills, and memory builders for reuse by the execution agent.

didate place suitable for navigation. We consider tasks whose required information is present in retained observations. A task exposes a *capability gap* when the current tools, skills, or memory-access mechanisms cannot solve it reliably within the online budget. For example, a room sign may be visible in memory, yet searching images individually can exhaust the call budget, motivating dedicated text extraction and retrieval.

3.2 Spatiotemporal Evidence Memory

ReSight anchors memory in retained RGB keyframes and their associated observations and acquisition context. Available data vary with the robot's sensors and the benchmark, including temporal associations, depth, poses, calibration, and synchronization information. We preserve these records at ingestion, especially information that is difficult to recover reliably from images, for subsequent processing. Let $\mathcal{O}$ denote the retained records, $\mathcal{F}^{(0)}$ the initial fields, and $\mathcal{X}^{(r)}$ the derived fields and indices accumulated after r extensions. The memory is

$$\mathcal{M}^{(r)} = (\mathcal{O}, \mathcal{F}^{(0)}, \mathcal{X}^{(r)}). \tag{1}$$

Initial fields record or reference observation times and available poses, and include descriptions and features produced during memory construction. A VLM generates scene and object descriptions for selected keyframes without conditioning on subsequent queries. These capture global scene semantics and object categories, attributes, and spatial context, respectively. We encode both types of description into text embeddings and the images into visual embeddings. Fields are linked to images by frame identifiers; events spanning multiple frames refer to the corresponding observation sets. Observation poses record camera locations, while object positions can be computed from depth or multiple views.

Text queries retrieve candidates independently from visual, scene-description, and object-description embeddings, then combine them by reciprocal rank fusion (RRF). Image queries use the visual channel. Appendix A.2 details fusion, filtering, and candidate grouping.

Subsequent tasks can introduce fields for OCR, events, or geometry. *Memory builders* generate these fields and their query indices from retained observations, and new tools expose the operations required by a task. We use *late binding* to describe determining how observations are interpreted and processed after storage: new programs can reprocess retained records to extract information absent from initial fields. *Initial* and *Extensions* distinguish when fields are introduced, independently of their source. Depth, for example, may be supplied by a sensor or dataset, or estimated from retained images.

3.3 Asymmetric Access and Dual-Agent Problem Solving

The execution and coding agents share the same memory but have different interfaces and compute budgets. The execution agent receives a *database view* through registered tools and uses skills

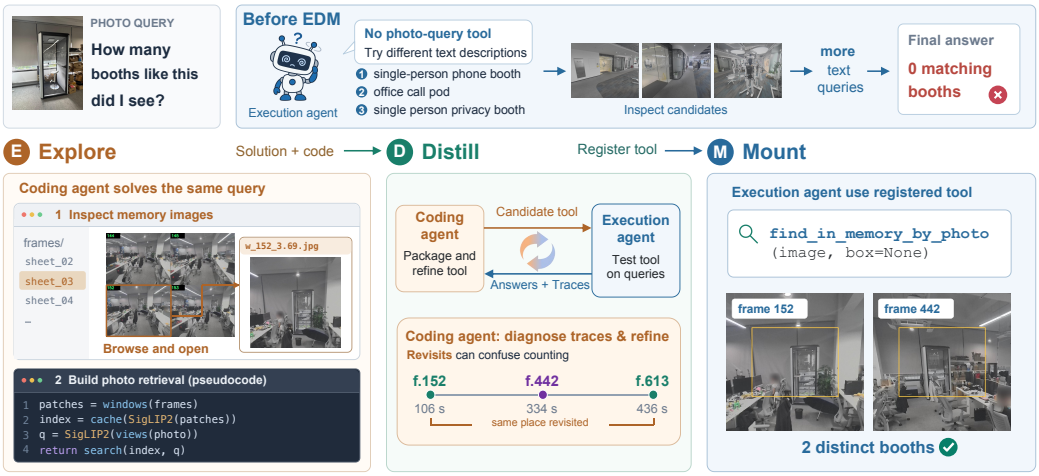


Figure 3: EDM on robot-memory booth counting. After text-based retrieval fails, the coding agent develops photo retrieval and verifies two booths. Executor replay during distillation reveals that revisits confuse counting, prompting pose-and-time merging; the mounted tool returns both booths and supporting frames in one call.

to guide retrieval, verification, and answering. Retrieval initially returns compact information, including frame identifiers, timestamps, locations, and descriptions. The agent requests images when it needs to verify details. This progressive disclosure reduces the context consumed by candidate screening under a bounded online call budget.

The coding agent receives a programmable *workspace view*. It can search and read files and images, write, execute, and debug programs, and download models or install dependencies as needed. Exploration has a compute budget independent of the execution agent's per-query call limit, accommodating model preparation, memory processing, and debugging. The coding agent can reuse existing capabilities or construct new procedures, returning an answer while recording its solution, evidence, and exploration trace. EDM then transfers reusable parts to the execution agent.

3.4 Explore–Distill–Mount

EDM converts execution traces and exploration traces into capability-library updates, implementing demand-driven harness evolution. Let H denote the execution configuration, including callable tools, skills that guide their use, and memory builders that produce persistent fields and indices. With retained records $\mathcal{O}$ and the execution model fixed, let $s_B(q; H)$ denote the expected task score under configuration H and online budget B, with higher scores indicating better performance. EDM's design objective is to improve performance on capability-gap tasks:

$$\max_{\Delta H} \mathbb{E}_{q\sim\mathcal{D}_{\text{gap}}}[s_B(q; H') - s_B(q; H)], \qquad H' = \text{Mount}(H, \Delta H). \tag{2}$$

Here ΔH is the capability extension, H' the mounted configuration, and $\mathcal{D}_{\text{gap}}$ the distribution of tasks requiring capability expansion. EDM extracts reusable operations from exploration traces and integrates them with the host. Figure 3 follows one task, counting phone booths from a reference photo, through the three steps.

Explore: develop a solution. The coding agent receives an evidence memory with all queries posed on it and solves them jointly in one session. It inspects images, invokes models, and writes programs, returning an answer to every query with the programs, notes, and evidence behind it. New fields may first appear at this stage. For example, the coding agent can run OCR to locate a room sign and verify the extracted text against the image. Exploration and distillation use observations and exploration traces without task reference answers.

Distill: extract reusable capabilities. Exploration may interleave scripts, shell commands, direct file inspection, and decisions based on intermediate results. Distillation turns reusable parts of this process into operations callable through the execution agent's registered interface. Failed or inefficient runs in the executor's traces locate the difficulties to address. The coding agent reads the executor's tool inventory and these traces, reuses existing capabilities, and packages missing or partially supported operations as tools with explicit inputs and outputs. Reusable preprocessing is packaged as memory builders.

Mount: integrate with execution. The distilled tools are registered with the execution agent. When a skill is mounted, the executor receives it in its system prompt: a short text on when and how to use the tools. Builder-generated fields and indices are exposed through query tools. For a new builder b with output $\mathcal{X}_b$, the memory update is

$$\mathcal{X}_b = b(\mathcal{M}^{(r)}), \qquad \mathcal{M}^{(r+1)} = (\mathcal{O}, \mathcal{F}^{(0)}, \mathcal{X}^{(r)} \cup \mathcal{X}_b). \tag{3}$$

Model initialization and whole-memory processing can precede mounting; query-dependent computation runs when a tool is called. Builders can be rerun when memory data change. Subsequent tasks use the same execution model and online budget with the updated capabilities.

4 Experiments

We evaluate base memory access on GOAT-core, capability expansion on OC-NaVQA and FindingDory, and reuse on new queries and memories in real-robot patrol experiments.

4.1 Experimental Setup

ReSight denotes the complete system; w/o EDM and w/ EDM compare the same executor before and after capability mounting. We also evaluate RAVEN as a second host. Within each OC-NaVQA and FindingDory comparison, GPT-5.6-sol (Sol) and the online decision-round cap remain fixed. Opus-5 coding-agent results provide an exploration reference under a separate compute budget. Retrieval settings appear in Appendix C.3.

4.2 Base Memory Design on GOAT-core

We evaluate ReSight's initial memory fields and multi-channel retrieval against MoM's SigLIP2-only visual-retrieval configuration.

Following the MoM protocol, we evaluate image, object, and language target queries across four GOAT-core scenes, for a total of 480 queries (Zhou et al., 2026). All reproduced configurations share the localization backend and return up to five candidates per query. We report SR@5 with a 1.5 m success threshold on the navigation plane. ReSight's retrieval configuration returns candidates directly, while its agent configuration can perform further retrieval and inspect images.

Direct retrieval achieves 90.8% SR@5 with Sol annotation and 89.8% with Qwen, above MoM's visual-only 81.5%; the agent achieves a similar 90.6% (Table 1). These results support the effectiveness of the initial memory and retrieval design.

4.3 Capability Expansion with EDM

We assess EDM on OC-NaVQA and FindingDory through before-and-after comparisons, additionally distilling and mounting capabilities for RAVEN as a second host on FindingDory.

OC-NaVQA. This benchmark contains 210 questions from seven sequences, covering binary, text, position, and temporal queries (Gorlo et al., 2026a). Position questions require object coordinates, whereas ReSight's base executor returns an observation's camera pose. EDM adds object-position estimation among other capabilities; evaluation details appear in Appendix C.3.

EDM reduces mean position error from 33.10 m to 24.72 m and increases the fraction within 10 m from 19.7% to 36.6% (Table 2). The object-position tool combines image segmentation, depth backprojection, and multi-view aggregation to return target coordinates with supporting frames (Figure 4(a)).

Table 1: GOAT-core SR@5 (%); no EDM. The first six rows are from MoM Table I (Zhou et al., 2026); HOV-SG averages two supported query types. Retrieval-only variants use no query-time VLM reasoning. Qwen/Sol denote annotation models (Appendix C.2); bold marks group bests.

Method	Image ↑	Object ↑	Language ↑	Average ↑
DynaMem (Liu et al., 2025)	16.1	7.5	5.1	9.6
VLMaps (Huang et al., 2023)	43.3	69.7	61.0	58.8
LagMemo (Zhou et al., 2025)	56.4	88.4	66.8	70.8
OpenFunGraph (Zhang et al., 2025a)	53.4	67.1	69.0	63.1
HOV-SG (Werby et al., 2024)	–	70.8	82.3	76.5
MoM (Zhou et al., 2026)	80.1	79.5	82.3	80.6
ReSight (agent)	**92.5**	**90.7**	**88.6**	**90.6**
Retrieval-only variants				
MoM (visual-only)	88.2	72.7	83.5	81.5
ReSight (retrieval, Qwen)	**90.1**	**91.3**	88.0	89.8
ReSight (retrieval, Sol)	**90.1**	90.1	**92.4**	**90.8**

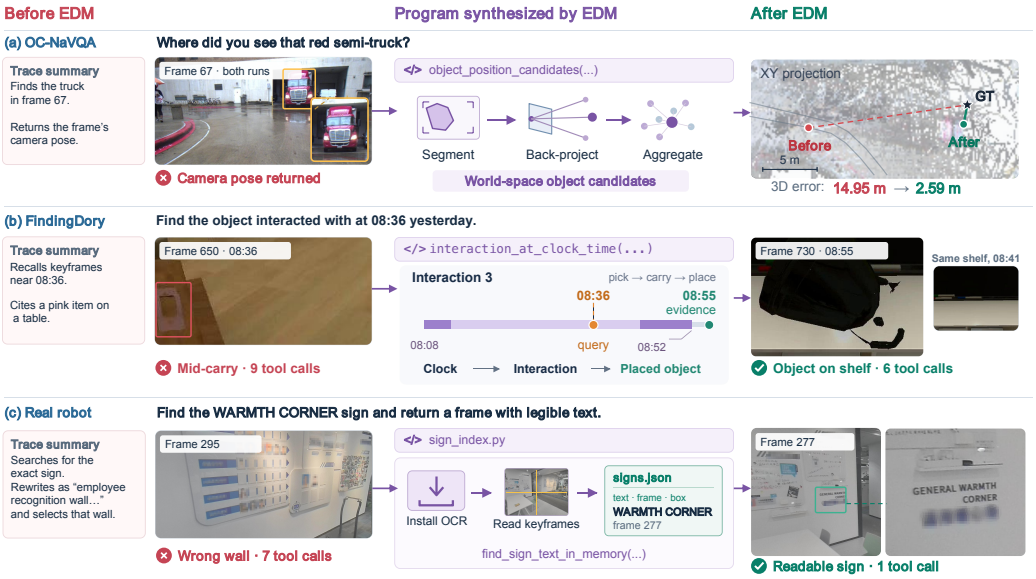


Figure 4: Examples of EDM capability expansion: (a) object localization, (b) interaction-object tracking, and (c) OCR sign retrieval. Columns show execution before mounting, the generated programs, and execution after mounting. Call counts are per illustrated run; full trajectories appear in Appendix E.

FindingDory. We follow RAVEN's long-horizon subset, reporting official high-level success on 596 queries from episodes 91–100 across spatial, temporal, and multi-goal tasks (Hu et al., 2026). ReSight and RAVEN each use a 15-decision-round budget. Capabilities are distilled separately for each host from exploration traces on these episodes. Appendix C.3 details retrieval and evaluation settings.

EDM raises overall success from 38.3% to 63.8% on ReSight's executor and from 38.9% to 66.6% on RAVEN's, enabling both to reuse capabilities developed during exploration (Table 3). After mounting, ReSight answers with 43k tokens per query against RAVEN's 150k: its retrieval returns text, and its added tools return compact text or composed evidence panels, while RAVEN places every retrieved frame in context as an image.

ReSight's executor gains most on multi-goal and temporal tasks, improving from 7.9% to 50.6% and from 41.0% to 82.9%. The distilled tools support interaction enumeration, revisit ordering, duration

Table 2: OC-NaVQA results on 210 questions. The first four rows are from DAAAM (Gorlo et al., 2026a): Mini is GPT-5-mini; 2B/8B are NVILA-Lite models. RAVEN-Sol and ReSight-Sol† return camera poses. Bold compares Sol configurations. Orange denotes the Opus-5 exploration reference.

	Accuracy (%)		Position		Temporal
Method	**Binary ↑**	**Text ↑**	**Mean (m) ↓**	**≤10 m (%) ↑**	**Error (min) ↓**
ReMEmbR-2B (Anwar et al., 2024)	43.2	–	53.47	–	2.287
ReMEmbR-8B	46.3	–	55.89	–	4.106
ConceptGraphs (Gu et al., 2023)	29.9	–	111.29	–	–
DAAAM-Mini (Gorlo et al., 2026a)	71.1	–	41.75	–	1.792
DAAAM-Sol	80.6	57.6	36.34	**52.1**	2.225
RAVEN-Sol (Hu et al., 2026)	80.6	45.5	40.55	22.5	1.128
ReSight-Sol (w/o EDM)†	**86.6**	**63.6**	33.10	19.7	0.587
ReSight-Sol (w/ EDM)	82.1	**63.6**	**24.72**	36.6	**0.475**
CodingAgent-Opus-5	95.5	72.7	26.12	39.4	0.482

Table 3: FindingDory high-level success (%) on episodes 91–100 (596 queries). The first five rows are from RAVEN (Hu et al., 2026); Gemini is Gemini-2.5-Flash. Bold compares Sol configurations; orange denotes the Opus-5 exploration reference. Tokens/q reports mean input-plus-output tokens in thousands. Annotation-model results appear in Appendix C.5.

Method	**Spatial ↑**	**Temporal ↑**	**Multi-goal ↑**	**Overall ↑**	**Tokens/q (k) ↓**
Random	13.4	6.9	0.5	9.2	–
Embedder Only	31.0	8.1	3.4	18.8	–
VLM Only-Gemini	36.0	28.1	2.2	28.2	–
ReMEmbR-Gemini (Anwar et al., 2024)	36.7	17.1	4.5	25.0	–
RAVEN-Gemini (Hu et al., 2026)	45.1	24.8	6.7	32.2	–
RAVEN-Sol	51.9	34.8	5.6	38.9	378
RAVEN-Sol (w/ EDM)	**55.9**	**89.5**	48.3	**66.6**	150
ReSight-Sol (w/o EDM)	45.5	41.0	7.9	38.3	130
ReSight-Sol (w/ EDM)	54.2	82.9	**50.6**	63.8	**43**
CodingAgent-Opus-5	58.2	76.2	43.8	62.4	–

ranking, and clock-time alignment. In Figure 4(b), they link the query time to an interaction and return post-placement evidence, avoiding a frame captured while the object was still being carried.

Reuse on held-out episodes. We freeze the tools and skill distilled on episodes 91–100 and evaluate them on 597 queries from episodes 81–90, which were excluded from exploration and distillation, with no further capability updates. EDM raises overall success from 37.5% to 65.8% (+28.3 percentage points), while reducing tokens per query by 69.4% (Table 4). The gain is comparable to the 25.5-point gain on episodes 91–100, supporting reuse across episodes. Appendix C.3 gives the protocol. Distilling from the same execution traces without exploration programs, notes, or answers reaches 46.4%: its event tool detects pauses of the robot base, which follow placements but miss pickups, and miscounts interactions in 3 of 10 episodes (Appendix C.3). These results support the added value of exploration-derived solutions for building reusable temporal capabilities that generalize to held-out episodes.

4.4 Real-Robot Experiments

We evaluate ReSight on office-building patrol memories collected by an A3 robot, testing reuse of frozen programs on new queries and another floor. Exploration uses only floor-8 memory and three tasks: locating the Warmth Corner sign, finding the point farthest from the patrol start, and counting booths from a reference image. Both execution configurations use Sol with at most ten tool-interaction rounds.

Table 4: Frozen-capability reuse on FindingDory episodes 81–90 (597 queries). EDM w/o Explore distills from the execution traces (Appendix C.3). High-level success (%); Tokens/q reports mean input-plus-output tokens in thousands.

Method	Spatial ↑	Temporal ↑	Multi-goal ↑	Overall ↑	Tokens/q (k) ↓
ReSight-Sol (w/o EDM)	48.3	36.2	4.5	37.5	128.8
ReSight-Sol (EDM w/o Explore)	51.0	48.6	25.8	46.4	**26.9**
ReSight-Sol (w/ EDM)	**57.4**	**83.3**	**52.8**	**65.8**	39.4

Table 5: Real-robot memory queries, grouped into exploration and new tasks. Success is out of five runs; tool calls and input-plus-output tokens are per-run means. The last row aggregates 30 runs; bold marks pairwise bests. Queries and inputs: Appendix C.4.

	Success ↑		Tool calls ↓		Tokens/run (k) ↓	
Task	w/o EDM	w/ EDM	w/o EDM	w/ EDM	w/o EDM	w/ EDM
Warmth Corner (8F)	2/5	**5/5**	8.4	**1.0**	60.0	**10.4**
Farthest point (8F)	1/5	**5/5**	7.2	**1.0**	38.8	**10.0**
Booth count (8F)	0/5	**5/5**	7.6	**1.0**	50.2	**12.9**
Room 805 (8F)	**5/5**	**5/5**	2.0	**1.0**	12.5	**11.2**
Patrol length (1F)	0/5	**5/5**	3.4	**1.8**	**14.5**	15.1
Photo localization (1F)	4/5	**5/5**	2.0	**1.4**	**11.7**	14.3
Total / mean	12/30	**30/30**	5.10	**1.20**	31.29	**12.31**

We freeze the tools and builders, then evaluate the three exploration tasks and three new queries: room 805 on floor 8, and patrol length and photo localization on floor 1. The latter memory was collected on another day and excluded from exploration; frozen builders reconstruct its fields. Each task runs independently five times (Appendix C.4).

Mounting improves success from 12/30 to 30/30 and reduces mean online tool calls from 5.10 to 1.20 (Table 5). The Warmth Corner text index also answers room 805; on the new floor, patrol-length success improves from 0/5 to 5/5 and photo localization from 4/5 to 5/5.

To isolate memory-related performance, we evaluate memory queries separately from downstream execution; Appendix E.4 demonstrates how the retrieved locations support subsequent robot navigation, on-site inspection, and waiting at a user-specified destination.

5 Limitations

Our evaluation uses predefined query sets and task categories, enabling controlled comparisons of memory access and capability expansion. Real-world deployment can introduce new, long-tail queries as users' needs change. While our experiments demonstrate capability reuse across held-out episodes and new robot memories, they only partially capture this evolving task distribution. Evaluating repeated capability expansion over longer deployments and a broader range of emerging queries is an important direction for future work.

6 Conclusion

We introduced ReSight, which retains revisitable visual evidence and expands memory processing through EDM. Its dual-agent architecture separates bounded execution from programmatic exploration, distilling reusable tools, skills, and memory builders. Experiments demonstrate effective memory access, improved spatiotemporal question answering, and capability reuse across new queries and robot memories with fewer online tool calls.

AI USE STATEMENT

We used generative AI tools during early-stage research to brainstorm and refine ideas, identify relevant literature, and assist our assessment of the method's novelty. During experiments, AI assisted code implementation and debugging. We also used AI to review manuscript drafts and suggest improvements; the authors evaluated this feedback and decided whether additional experiments were warranted. The authors wrote initial drafts and used AI to refine wording and improve readability. All AI-assisted outputs were reviewed and verified by the authors. References, experimental settings, quantitative results, and scientific claims were checked against original sources or experimental records. The authors take responsibility for the final content.

REFERENCES

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, Christopher Potts, Koushik Sen, Alex Dimakis, Ion Stoica, Dan Klein, Matei Zaharia, and Omar Khattab. GEPA: Reflective Prompt Evolution Can Outperform Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*, pp. 8479–8565, 2026. URL https://openreview.net/forum?id=RQm2KQTM5r.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. ReMEmbR: Building and Reasoning Over Long-Horizon Spatio-Temporal Memory for Robot Navigation, 2024. URL https://arxiv.org/abs/2409.13682.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large Language Models as Tool Makers. In *International Conference on Learning Representations (ICLR)*, pp. 54067–54089, 2024. URL https://openreview.net/forum?id=qV83K9d5WB.

Lirong Che, Yuzhe Yang, Peiwen Lin, Chuang Wang, Xueqian Wang, and Jian Su. DemoEvolve: Overcoming Sparse Feedback in Agentic Harness Evolution with Demonstrations, 2026. URL https://arxiv.org/abs/2605.24539.

Tingyang Chen, Shuo Lu, Kang Zhao, Weicheng Meng, Hanlin Teng, Tianhao Li, Chao Li, Xule Liu, Jian Liang, Zhizhong Zhang, Yuan Xie, Heng Qu, Kun Shao, and Jian Luan. HarnessX: A Composable, Adaptive, and Evolvable Agent Harness Foundry, 2026. URL https://arxiv.org/abs/2606.14249.

Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, Fei Xia, Jasmine Hsu, Jonathan Hoech, Pete Florence, Sean Kirmani, Sumeet Singh, Vikas Sindhwani, Carolina Parada, Chelsea Finn, Peng Xu, Sergey Levine, and Jie Tan. Mobility VLA: Multimodal Instruction Navigation with Long-Context VLMs and Topological Graphs, 2024. URL https://arxiv.org/abs/2407.07775.

Yue Fan, Xiaojian Ma, Rujie Wu, Yuntao Du, Jiaqi Li, Zhi Gao, and Qing Li. VideoAgent: A Memory-augmented Multimodal Agent for Video Understanding, 2024. URL https://arxiv.org/abs/2403.11481.

Nicolas Gorlo, Lukas Schmid, and Luca Carlone. Describe Anything Anywhere At Any Moment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 35002–35013, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Gorlo_Describe_Anything_Anywhere_At_Any_Moment_CVPR_2026_paper.html.

Nicolas Gorlo, Derek K. Wise, Alberto Speranzon, and Luca Carlone. Worth Remembering: Surprise-Gated Robot Episodic Memory, 2026b. URL https://arxiv.org/abs/2606.03787.

Qiao Gu, Alihusein Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Rama Chellappa, Chuang Gan, Celso Miguel de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning, 2023. URL https://arxiv.org/abs/2309.16650.

Shengran Hu, Cong Lu, and Jeff Clune. Automated Design of Agentic Systems. In *International Conference on Learning Representations (ICLR)*, pp. 21344–21377, 2025a. URL https://openreview.net/forum?id=t9U3LW7JVX.

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, 2025b. URL https://arxiv.org/abs/2505.22657.

Yixun Hu, Zhicheng Zheng, Lihan Zha, Chunwei Xing, Rajdeep Singh, Omar Hossain, Antonio Loquercio, and Dhruv Shah. RAVEN: Long-Horizon Reasoning & Navigation with a Visuo-Spatio-Temporal Memory, 2026. URL https://arxiv.org/abs/2606.25206.

Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual Language Maps for Robot Navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10608–10615, 2023. URL https://doi.org/10.1109/ICRA48891.2023.10160969.

Krishna Murthy Jatavallabhula, Alihusein Kuwajerwala, Qiao Gu, Mohd Omama, Tao Chen, Alaa Maalouf, Shuang Li, Ganesh Iyer, Soroush Saryazdi, Nikhil Keetha, Ayush Tewari, Joshua B. Tenenbaum, Celso Miguel de Melo, Madhava Krishna, Liam Paull, Florian Shkurti, and Antonio Torralba. ConceptFusion: Open-set Multimodal 3D Mapping, 2023. URL https://arxiv.org/abs/2302.07241.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-End Optimization of Model Harnesses, 2026. URL https://arxiv.org/abs/2603.28052.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language Model Programs for Embodied Control. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. URL https://doi.org/10.1109/ICRA48891.2023.10160591.

Jiahang Lin, Shichun Liu, Chengjun Pan, Lizhi Lin, Shihan Dou, Zhiheng Xi, Xuanjing Huang, Hang Yan, Zhenhua Han, Tao Gui, and Yu-Gang Jiang. Agentic Harness Engineering: Observability-Driven Automatic Evolution of Coding-Agent Harnesses, 2026. URL https://arxiv.org/abs/2604.25850.

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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13346–13355, 2025. URL https://doi.org/10.1109/ICRA55743.2025.11127619.

Xinghua Lou, Miguel Lázaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, and Kevin P. Murphy. AutoHarness: improving LLM agents by automatically synthesizing a code harness, 2026. URL https://arxiv.org/abs/2603.03329.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, Mosharaf Chowdhury, Yuke Zhu, Linxi "Jim" Fan, and Guanzhi Wang. ASPIRE: Agentic /Skills Discovery for Robotics, 2026. URL https://arxiv.org/abs/2607.00272.

Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Dan Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time Task-Driven Open-Set 3D Scene Graphs, 2024. URL https://arxiv.org/abs/2404.13696.

Dominic Maggio, Nicolas Gorlo, Kris Hauser, and Luca Carlone. FOUND-IT: Foundation-model-first Task-driven 3D Scene Graphs with Granularity on Demand, 2026. URL https://arxiv.org/abs/2605.25371.

Yufan Mao, Hanjing Ye, Wenlong Dong, Chengjie Zhang, and Hong Zhang. Meta-Memory: Retrieving and Integrating Semantic-Spatial Memories for Robot Spatial Reasoning, 2025. URL https://arxiv.org/abs/2509.20754.

Songyou Peng, Kyle Genova, Chiyu "Max" Jiang, Andrea Tagliasacchi, Marc Pollefeys, and Thomas Funkhouser. OpenScene: 3D Scene Understanding with Open Vocabularies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–824, 2023. URL https://doi.org/10.1109/CVPR52729.2023.00085.

Maxime Robeyns, Martin Szummer, and Laurence Aitchison. A Self-Improving Coding Agent, 2025. URL https://arxiv.org/abs/2504.15228.

Lukas Schmid, Marcus Abate, Yun Chang, and Luca Carlone. Khronos: A Unified Approach for Spatio-Temporal Metric-Semantic SLAM in Dynamic Environments, 2024. URL https://arxiv.org/abs/2402.13817.

Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. CLIP-Fields: Weakly Supervised Semantic Fields for Robotic Memory. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. URL https://www.roboticsproceedings.org/rss19/p074.html.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual Inference via Python Execution for Reasoning, 2023. URL https://arxiv.org/abs/2303.08128.

Ayça Takmaz, Elisabetta Fedele, Robert W. Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. OpenMask3D: Open-Vocabulary 3D Instance Segmentation, 2023. URL https://arxiv.org/abs/2306.13631.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An Open-Ended Embodied Agent with Large Language Models, 2023. URL https://arxiv.org/abs/2305.16291.

Abdelrhman Werby, Chenguang Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical Open-Vocabulary 3D Scene Graphs for Language-Grounded Robot Navigation, 2024. URL https://arxiv.org/abs/2403.17846.

Abdelrhman Werby, Dennis Rotondi, Fabio Scaparro, and Kai O. Arras. KeySG: Hierarchical Keyframe-Based 3D Scene Graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2026. URL https://keysg-lab.github.io/.

Quanting Xie, So Yeon Min, Pengliang Ji, Yue Yang, Tianyi Zhang, Kedi Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-RAG: General Non-parametric Embodied Memory for Retrieval and Generation, 2024. URL https://arxiv.org/abs/2409.18313.

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, 2024. URL https://arxiv.org/abs/2411.17735.

Chenyangguang Zhang, Alexandros Delitzas, Fangjinhua Wang, Ruida Zhang, Xiangyang Ji, Marc Pollefeys, and Francis Engelmann. Open-Vocabulary Functional 3D Scene Graphs for Real-World Indoor Spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19401–19413, 2025a. URL https://doi.org/10.1109/CVPR52734.2025.01807.

Guibin Zhang, Haotian Ren, Chong Zhan, Zhenhong Zhou, Junhao Wang, He Zhu, Wangchunshu Zhou, and Shuicheng Yan. MemEvolve: Meta-Evolution of Agent Memory Systems, 2025b. URL https://arxiv.org/abs/2512.18746.

Hangfan Zhang, Shao Zhang, Kangcong Li, Chen Zhang, Yang Chen, Yiqun Zhang, Lei Bai, and Shuyue Hu. Self-Harness: Harnesses That Improve Themselves, 2026a. URL https://arxiv.org/abs/2606.09498.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Godel Machine: Open-Ended Evolution of Self-Improving Agents, 2025c. URL https://arxiv.org/abs/2505.22954.

Luan Zhang, Ruochen Zhou, Dandan Song, Zhengyu Chen, Yuhang Tian, Jun Yang, Huipeng Ma, Chenhao Li, Guangyuan Feng, Xudong Li, Yizhou Jin, and Yan Xu. HarnessCompass: Guiding Automatic Harness Evolution toward Generalizable and Effective Agent Harnesses, 2026b. URL https://arxiv.org/abs/2608.01918.

Qizheng Zhang, Changran Hu, Shubhangi Upasani, Boyuan Ma, Fenglu Hong, Vamsidhar Kamanuru, Jay Rainton, Chen Wu, Mengmeng Ji, Hanchen Li, Urmish Thakker, James Zou, and Kunle Olukotun. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models, 2025d. URL https://arxiv.org/abs/2510.04618.

Haotian Zhou, Xiaole Wang, He Li, Zhuo Qi, Jinrun Yin, Haiyu Kong, Jianghuan Xu, and Huijing Zhao. LagMemo: Language 3D Gaussian Splatting Memory for Multi-modal Open-vocabulary Multi-goal Visual Navigation, 2025. URL https://arxiv.org/abs/2510.24118.

Rui Zhou, Xander Yap, Jianwen Cao, Allison Lau, Boyang Sun, and Marc Pollefeys. Memory Over Maps: 3D Object Localization Without Reconstruction, 2026. URL https://arxiv.org/abs/2603.20530.

Xueyan Zou, Yuchen Song, Ri-Zhao Qiu, Xuanbin Peng, Jianglong Ye, Sifei Liu, and Xiaolong Wang. M3: 3D-Spatial MultiModal Memory, 2025. URL https://arxiv.org/abs/2503.16413.

A Base Memory and Tools

A.1 Base Tool Interface

Table 6 gives the arguments and function of the base tools used on OC-NaVQA and FindingDory. Brackets mark optional arguments. Retrieval limits and online rounds are given in Appendix C.3.

Table 6: Base execution interface for OC-NaVQA and FindingDory. Retrieval tools return text summaries; position recall requires camera poses.

Tool	Arguments	Function
`semantic_search`	`query`, [`top_k`]	RRF semantic search
`recall_around_time`	`clock_time`, [`limit`]	Time-neighbor recall
`recall_near_position`	`position` $[x, y, z]$, [`limit`]	Pose-neighbor recall
`inspect_frames`	`frame_ids`, [`focus`]	Show original images
`submit_answer`	`type`, `reasoning`, `source_frame_ids`, type-specific answer field	Submit answer

Position retrieval returns the camera location of an observation. Estimating an object's location within that image requires geometry; recovering an interaction sequence requires reasoning across frames. EDM adds tools for these operations.

A.2 Retrieval Fusion and Progressive Disclosure

Content retrieval combines a SigLIP2 visual embedding with e5 embeddings of the scene description and per-frame object text. Each channel ranks candidates independently. Reciprocal rank fusion then scores frame i for query q by

$$\text{score}(i, q) = \sum_{c \in \mathcal{C}} \frac{w_c}{\kappa + \text{rank}_c(i, q)}. \tag{4}$$

A time interval can filter candidates. Where poses exist, nearby viewpoints can be grouped for retrieval while each frame remains individually accessible.

The first return is compact text: frame id, description, object fields, and available temporal and spatial metadata, cropped and formatted from stored records. The agent opens the original images only when it needs to verify text, object state, or visual detail. This separates candidate screening from image inspection; measured online usage appears in Appendix D.

A.3 Keyframe Fields and an Annotation Example

Each retained keyframe links an image through `frame_id` to its capture time and, when available, camera pose and depth. These metadata come from acquisition and processing. A vision-language annotator reads the image without seeing the later questions and generates semantic text.

The following excerpt is from frame 0 of OC-NaVQA sequence 0, annotated by Qwen3.8-27B. It shows the scene description and the first two objects; other fields and objects are omitted:


```
{
  "global_description": "An outdoor industrial alley or service
  walkway flanked by brick buildings, containing large green
  utility structures, a gray electrical or mechanical housing
  unit, and a paved concrete floor.",
  "objects": [
    {
      "label": "green utility enclosure",
      "description": "A large, long, dark green rectangular
  structure, likely a utility tunnel cover or storage
  container, sitting on the left side of the alley."
    },
    {
      "label": "gray utility housing",
      "description": "A light gray, box-like metal cabinet or
  small structure with vents and a door, located at the end of
  the concrete path."
    }
  ]
}
```


The scene description captures layout; object labels and descriptions provide appearance and local context. Both text types are encoded separately and linked to the image through `frame_id`. Later memory builders can reprocess retained images for OCR, interactions, or geometry without collecting the observations again. The annotator replacement experiment is in Appendix C.5.

A.4 Two Ways to Access the Same Memory

The execution agent uses registered tools and mounted skills within an online round limit. The coding agent reads observations and processing programs in a workspace, inspects images, writes code, and prepares models or dependencies during exploration. The capability library holds tools, skills, and memory builders. EDM adds reusable artifacts from exploration to that library for subsequent execution.

B Explore–Distill–Mount Implementation

B.1 Diagnosing Capability Gaps from Execution

After exploration, distillation inventories the host's tools, memory fields, and execution traces in a capability ledger. Each operation used during exploration is classified as directly supported, composable from existing operations, partially supported, or missing. A script flags queries whose traces show a difficulty, such as a forced final answer, repeated identical calls, or an answer given without inspecting frames, and queries on which the executor's answer disagrees with an explorer's answer. A gap map groups the flagged queries by cause and relates each gap to the operations and exploration programs that address it.

Diagnosis uses execution traces and answer-disagreement flags to identify candidate gaps, without task reference answers or evaluation scores.

B.2 Exploration, Distillation, and Mounting

In *Explore*, one coding-agent session per memory answers all queries posed on that memory. The agent inspects retained observations, writes programs, and delivers an answer to every query together

Algorithm 1 One EDM round

Require: execution configuration H, memories $\{\mathcal{M}_e\}$ with queries $\{\mathcal{Q}_e\}$, online budget B
Ensure: mounted configuration H' under the same budget B

$\mathcal{T} \leftarrow \textsc{Run}(H, \bigcup_e \mathcal{Q}_e, B)$ ▷ traces, without correctness labels
for each memory e, in parallel **do**
  $\mathcal{E}_e \leftarrow \textsc{Explore}(\mathcal{M}_e, \mathcal{Q}_e)$ ▷ answers, programs, notes
end for
$F \leftarrow \textsc{Flag}(\mathcal{T}, \mathcal{E})$ ▷ behavior flags; answers that differ from the explorers'
$L, G \leftarrow \textsc{Diagnose}(H, \mathcal{T}, F, \mathcal{E})$ ▷ capability ledger, gap map
$\mathcal{A} \leftarrow \textsc{Build}(H, \mathcal{T}, L, G, \mathcal{E})$ ▷ tools and builders for the gaps
for $r = 1, \ldots, R$ **do** ▷ optional replay; $R = 0$ skips refinement
  $\mathcal{T}_r \leftarrow \textsc{Run}(\textsc{Mount}(H, \mathcal{A}), \mathcal{Q}_r, B)$; $\mathcal{A} \leftarrow \textsc{Refine}(\mathcal{A}, \mathcal{T}_r)$
end for
$K \leftarrow \textsc{Skill}(\mathcal{A}, \mathcal{E}, \mathcal{T})$
return $\textsc{Mount}(H, \mathcal{A}, K)$

with its programs, notes, and supporting frames. A program may answer a query directly or first construct a field for the whole memory, such as an OCR index or a pose trajectory.

In *Distill*, the agent compares these operations with the capability ledger. Missing or partially supported operations become tools or memory builders, each linked to a diagnosed gap and its evidence, with the exploration programs as starting implementations. An optional replay round mounts the tools, lets the executor answer queries with them,, and revises the tools from the resulting traces. A skill then describes when and how the executor should use the tools.

In *Mount*, distilled tools are registered, the skill, if any, enters the executor's context, and builders expose their fields through query tools. Whole-memory processing and model initialization can run before queries; query-dependent computation runs when a tool is called. The execution model and online budget remain the same before and after mounting. For example, an OCR builder stores text with its source `frame_id`; a text-query tool then returns matching frames, which the executor may inspect. The builder can be rerun over a new patrol memory.

B.3 EDM Procedure

Algorithm 1 lists one EDM round. Run, Flag, and Mount are scripts; each other step is a coding-agent session whose prompt is given in Appendix B.4. B is the online budget used on both sides.

B.4 EDM Prompts

Explore, Diagnose, Build, Refine, and Skill in Algorithm 1 are coding-agent sessions. Below we give the instructions that define each session, with its inputs, deliverables, and constraints; operational details such as file layout, output schemas, and session management are omitted. Text in braces stands for inputs that vary across benchmarks, such as the memory, the executor's tools, and its online budget (Appendix C).

Explore (one session per memory).

> You are a coding agent with full access to this workspace. It holds {memory}: {memory description}, and {query set}: the queries posed on this memory, without reference answers. Work on all the queries in this session.
>
> 1. Answer every query yourself, by any method: inspect images, write and run programs, compute statistics, or use models. Support each answer with frames you have checked.
>
> 2. Keep each operation you used as a program that can be run again.
>
> Deliverables: an answer to every query, with the frames that support it; programs for these operations, each opening with what it computes and how to call it; notes on the facts you measured, the query types each operation serves, and the pitfalls you encountered; and, optionally, operations already packaged as tools or builders that follow {interface}.
>
> Do not read, search for, or infer reference answers or evaluation results.

Diagnose.

> The execution agent has {executor tools} and {online budget}. {execution traces} record its behavior on the task pool. {flags} mark the queries to examine: queries whose behavior shows a difficulty, and queries where its answer differs from an explorer's answer. A flag is a reason for doubt, not a judgment of correctness, and no reference answers are involved. {exploration records} contain the explorers' notes and programs. Do not write code in this step.
>
> 1. Capability ledger. List each operation the explorers used, merging repeats across recordings. Classify it against the execution agent as direct (a base tool returns the same evidence), composable (a combination of base tools returns it), partial (a related base tool lacks an essential affordance, which you name), or missing. Cite the tool description or the execution traces that justify each class.
>
> 2. Gap map. Group the flagged queries by the cause behind their flags. For each gap, list the affected queries and their {query categories}, the flags that expose it, the operations that would resolve it, and the exploration programs that implement them. Assign a remedy: none if every operation it needs is direct or composable, a builder if an operation needs one pass over the whole memory, and a tool otherwise.
>
> Deliverables: the capability ledger and the gap map.

Build.

> Build only what the gap map calls for. You have the capability ledger and the gap map from the diagnosis, which you take as given; the execution agent's {executor tools} and {execution traces}; {exploration records}, whose programs are the implementations to start from; and recordings to test your tools on.
>
> 1. For each gap whose remedy is a tool or a builder, build it from the exploration programs the gap lists, reusing any other exploration program or packaged tool that helps. Implement a tool for computation at query time and a builder for computation over the whole memory; tools read the builders' outputs.
>
> 2. For a partial operation, extend the base tool's evidence with the missing affordance instead of building a parallel tool. Never rebuild a direct or composable operation.
>
> 3. Tools follow {interface}. They return evidence, such as frame identifiers, times, tables, or rendered images, and never answers. Replies stay within {reply and time limits}. Thresholds are measured on the memory at hand, with no constants specific to one recording.
>
> 4. Each tool's description states when the execution agent should call it and how it complements the base tools.
>
> Run every tool on the test recordings before submitting. Deliverables: the tools and builders, and a mapping from each tool to the gaps it closes.

Refine (optional).

> The distilled tools were mounted, and the execution agent answered {replay queries} with them; {replay traces} record these runs, without correctness labels. Examine how the tools' replies shaped its behavior: calls spent completing a reply, fields it misread, candidates it could not tell apart, and answers that differ across repeated trials. Revise the tools at the level of method, without special cases for particular queries or recordings, and without adding tools for operations the ledger classes as direct or composable.
>
> Deliverables: the revised tools, and a note on what changed and which queries showed each problem.

Skill.

> The execution agent now has the distilled tools {distilled tools} in addition to {executor tools}, under {online budget}. Write a skill of at most {N} lines that will be appended to its system prompt. State which tool to call first for which query type, in what order, what counts as sufficient evidence, which heuristics mislead, and how to spend the budget. Base each line on the explorers' notes or the execution traces, and end it with its source. Do not state or hint at any query's answer, and do not mention frame numbers, times, recording or query identifiers, or object and room names; write only what holds across recordings.
>
> Deliverable: the skill, one instruction per line, each line ending with its source.

C Experimental Settings and Comparison Protocol

C.1 Comparison Principles and Reproduced Configurations

Within a host, the before-and-after comparison fixes the execution model, base interface, decision-round cap, input, and scoring rule. Across hosts we retain each baseline's memory representation and agent loop and report their measured rounds, tokens, and image access. The coding-agent exploration reference solves an episode jointly with a separate workspace and compute budget.

For GOAT-core, the visual-only MoM reproduction uses SigLIP2 retrieval with the common segmentation, depth-backprojection, and multi-view localization backend. ReSight's retrieval-only rows change the retrieval representation and interface but do not invoke a query-time language model to choose candidates. The DAAAM reproduction uses its own scene graph and agent loop on OC-NaVQA, receives no original images online, and has no additional decision-round cap. RAVEN retains its benchmark-specific loop and image-bearing retrieval; on FindingDory, its temporal return size is reduced from 200 to 30 to limit single-request image input.

C.2 GOAT-CORE

GOAT-core has 480 image, object, and language queries across four scenes and 1,606 retained keyframes. A predicted location succeeds when it lies within 1.5 m of the target on the horizontal XZ plane. Retrieval-only and agent configurations share the localization backend. SR@1, SR@3, and SR@5 use prefixes of the same ranked candidate list; the main table reports SR@5.

The image encoder is SigLIP2-SO400M with patch size 14 at 384×384 resolution; e5 encodes text fields. Each enabled retrieval channel returns 40 candidates, combined with equal-weight RRF and $\kappa = 60$. When poses exist, viewpoint grouping uses 0.75 m and 25° thresholds. Retrieval-only returns at most five candidate views. The GPT-5.6-sol agent runs at most three rounds; semantic search returns eight entries by default and at most twelve, `inspect_frames` opens at most four images, and `submit_result` sends the candidates. The writing-time annotator comparison uses GPT-5.6-sol and Qwen2.5-VL-7B.

C.3 OC-NAVQA AND FINDINGDORY

Both benchmarks use GPT-5.6-sol in the same ReSight execution loop. Semantic retrieval combines visual, scene-description, and object-text channels, each with 40 candidates; RRF uses equal weights and $\kappa = 60$. Base retrieval returns text, while `inspect_frames` opens at most four low-detail images per call. FindingDory has no position recall because its data lack camera poses.

Host / benchmark	Semantic / temporal return	Position return	Rounds
ReSight / OC	12 text / 8 default, 12 max	8 default, 12 max text	6
ReSight / FD	12 text / 8 default, 12 max	None	15
RAVEN / OC	5 each, with images	5 with images	3
RAVEN / FD	30 each, with images	None	15

Table 7: Base retrieval limits per call and decision-round caps. EDM keeps each host's limits unchanged. Added tools can return event summaries, frame ids, or rendered panels.

On FindingDory both hosts cap each request at 30 images and manage image history according to their own loop. ReSight's rendered panels use a separate image configuration. One ReSight round is one model request and may emit several tool calls. OC-NaVQA allows six decision rounds and an additional finalization request if no answer was submitted; RAVEN allows three tool-calling decisions followed by finalization and answer generation. Appendix D counts all online model requests.

OC-NaVQA contains 210 questions in seven sequences and 3,973 keyframes. Position ground truth is an object coordinate; ReSight without EDM and the RAVEN adaptation return the selected observation's camera position, whereas the mounted position tool estimates an object coordinate. We report mean position error and the share within 10 m. Temporal MAE pools 30 timestamp and nine duration questions, in minutes. Text answers follow the common three-vote judge protocol.

Gains concentrate on localization, a major gap identified by the ledger; localization improves with both annotators (Table 9, Appendix C.5). With Sol annotations, mounting EDM corrects two of the 67 binary answers but introduces five errors (exact McNemar $p = 0.45$); binary accuracy remains unchanged with Qwen annotations. Text accuracy is unchanged in both settings, while temporal error decreases with Sol and increases with Qwen.

FindingDory evaluates 596 questions in episodes 91–100: 297 spatial, 210 temporal, and 89 multi-goal. A frame is retained when its SigLIP2 feature has cosine similarity below 0.90 to the preceding retained frame, yielding 4,660 keyframes with scene and object annotations. Without poses, each frame forms its own viewpoint group. The official high-level success rule scores each query; Overall is the micro-average across 596 questions. Submitted frame ids are converted to memory time through the episode mapping, with ordered or unordered evidence as requested and an empty list for an absent target.

The two FindingDory hosts each perform Distill and Mount against their own capability ledger, reusing exploration material from episodes 91–100. For the main results on these episodes, diagnosis and evaluation use the same public task pool and memory; the EDM procedure is described in Appendix B.3.

Frozen-capability evaluation on episodes 81–90. We evaluate 597 queries from episodes 81–90, excluded from exploration and distillation, using the frozen tools and skill developed on episodes 91–100. The same memory-construction pipeline yields 4,016 keyframes. Both configurations use GPT-5.6-sol through the same API backend, with 15 decision rounds and at most 30 images per request. We use the official high-level success rule and micro-average over all 597 queries. Tokens per query include model input and output; results appear in Table 4.

Distillation without exploration. This configuration repeats Distill and Mount for episodes 91–100 with the exploration stage removed. The coding agent receives the same execution traces, host tool inventory, interface specification, and tool implementation constraints as the full run. Both FindingDory configurations use three distillation steps: capability ledger and gap map, tool synthesis with code self-tests, and skill writing. It receives no exploration programs, notes, answers, or session records, so its flags contain no answer disagreements. In this configuration, directly viewing memory images or reading memory annotation files is prohibited. Its code may process the self-test recordings, and the agent may inspect the resulting text diagnostics and tool outputs to debug the tools; transcripts are audited for compliance. The distilled tools and skill are mounted and evaluated with the protocol above.

The run produces four tools: an event index, a revisit roster, a multi-frame zoom, and a place timeline. The event index segments intervals in which the robot base stops moving. These stops follow placements, so the index records one interval per interaction and no pickup. It reports 9, 9, and 11 interactions in three of the ten episodes, whereas every episode contains ten and the full run's event tool finds ten in each. Durations are measured as the length of a stop instead of the span from pickup to placement. Table 8 gives success by query type; the largest gaps to full EDM are ordered revisitation, duration tracking, and interaction order. The configuration answers 110 queries that the unmounted executor misses and misses 57 that it answers (exact McNemar $p = 5 \times 10^{-5}$); against full EDM the counts are 28 and 144 ($p = 5 \times 10^{-20}$). Its tools return short tables and the executor submits after 4.1 rounds on average, which lowers tokens per query to 26.9k.

Table 8: Success (%) on FindingDory episodes 81–90 by the benchmark's low-level query category; n is the number of queries.

Query type	n	**w/o EDM**	**EDM w/o Explore**	**w/ EDM**
Interaction Order	160	31.9	46.9	82.5
Conditional Interaction	68	42.6	44.1	51.5
Interaction	60	68.3	68.3	73.3
Unordered Revisitation	59	6.8	18.6	30.5
Room Visitation	50	74.0	72.0	86.0
Spatial Relationship	50	4.0	8.0	12.0
Object Attributes	50	52.0	64.0	68.0
Ordered Revisitation	30	0.0	40.0	96.7
Duration Tracking	30	33.3	36.7	76.7
Object Recall	20	45.0	45.0	45.0
Time-Based	20	75.0	80.0	100.0

C.4 Real Patrol Memory

We use an AGIBOT A3 robot to collect RGB observations and LiDAR-based mapping poses in an office building. The evaluation uses 648 keyframes from floor 8 and 501 from floor 1. GPT-5.6-luna annotates the memories, and visual and text embeddings are computed locally. Robot motion is disabled during evaluation; answers and cited historical evidence are scored. The real-robot interface offers `semantic_search` for English content queries (three returns by default, at most ten, with coordinates and frame paths) and `inspect_frames` for at most three original images. EDM adds text lookup, trajectory computation, photo matching, and a skill.

Execution is limited to ten tool-interaction rounds; exploration has a separate 400-round limit. Each of six tasks runs independently five times with the same prompt and image input in both configurations. Warmth Corner, farthest point, and booth counting are exploration tasks. Room 805 is a new query on the same memory; patrol length and photo localization use the new floor-1 memory. Frozen programs are reused and, where needed, rebuild indices on floor 1.

Sign queries need a cited frame showing the text, with a one-frame tolerance. Farthest point requires frame 349 ± 3 and distance 31.07 ± 1.5 m. Booth counting needs evidence for both instances. Patrol length succeeds within 10% of 111 m. Photo localization accepts a coordinate or cited camera pose within 3 m of the reference.

These are the six actual English prompts. `[BOOTH-REF]` stands for the booth reference photo. `[PHOTO-LOC]` is a re-saved image of floor-1 memory frame 285. Both configurations receive identical images.

Warmth Corner (8F). "Where is the Warmth Corner? Find in the memory the wall sign that reads WARMTH CORNER and give a frame_id in which that sign is clearly legible."

Farthest point (8F). "Taking the patrol start (the position of the first frame) as the reference, where on the whole trajectory is the point farthest from the start in straight-line distance? Give the frame_id at that point and the straight-line distance in meters."

Booth count (8F). "Here is a photo of a phone booth: [BOOTH-REF]. How many booths like this one did the robot see on this patrol? Give one frame_id for each."

Room 805 (8F). "Where does it say 805? Find in the memory the door plate or sign that reads 805 and give a frame_id in which the number is clearly legible."

Patrol length (1F). "How far did the robot walk in total during this patrol? Give the total walked distance in meters along the robot's own path (not a straight-line distance)."

Photo localization (1F). "Here is a photo: [PHOTO-LOC]. Where was it taken? Give the frame_id closest to that spot and its map coordinates (x, y in meters)."

C.5 Store Annotator Replacement

For OC-NaVQA and FindingDory we replace the GPT-5.6-sol writing-time annotator with locally served Qwen3.8-27B-FP8 and rebuild text embeddings. Keyframes, prompt, parser, retrieval channels, execution model, and online limits remain fixed. Programs distilled on the Sol-annotated store are reused directly on the Qwen-annotated store. Local Qwen annotation incurs no annotation API charge.

On either annotated store, EDM improves all three FindingDory categories.

Table 9: OC-NaVQA annotator replacement on 210 questions. All rows use the GPT-5.6-sol executor. Tokens/q reports mean online input-plus-output tokens in thousands. Green denotes configurations with EDM; bold marks column bests.

	Accuracy (%)		Position		Temporal	
Annotator / configuration	**Binary ↑**	**Text ↑**	**Mean (m) ↓**	**≤10 m (%) ↑**	**Error (min) ↓**	**Tokens/q (k) ↓**
GPT-5.6-sol (w/o EDM)	**86.6**	63.6	33.10	19.7	0.587	**18.8**
GPT-5.6-sol (w/ EDM)	82.1	63.6	**24.72**	**36.6**	0.475	20.4
Qwen3.8-27B (w/o EDM)	85.1	**69.7**	31.54	16.9	**0.427**	19.1
Qwen3.8-27B (w/ EDM)	85.1	**69.7**	28.29	35.2	0.463	21.3

Table 10: FindingDory annotator replacement on episodes 91–100 (596 queries). High-level success (%); all rows use the GPT-5.6-sol executor. Tokens/q reports mean online input-plus-output tokens in thousands. Green denotes configurations with EDM; bold marks column bests.

Annotator / configuration	**Spatial ↑**	**Temporal ↑**	**Multi-goal ↑**	**Overall ↑**	**Tokens/q (k) ↓**
GPT-5.6-sol (w/o EDM)	45.5	41.0	7.9	38.3	130
GPT-5.6-sol (w/ EDM)	**54.2**	82.9	**50.6**	63.8	43
Qwen3.8-27B (w/o EDM)	48.8	37.6	7.9	38.8	144
Qwen3.8-27B (w/ EDM)	**54.2**	**87.1**	**50.6**	**65.3**	**42**

D Compute Cost and Online Usage

D.1 Offline Construction and Capability Expansion

Keyframe annotation, vector encoding, and EDM capability construction are performed offline and can serve multiple queries. Appendix C.5 compares full task results for Sol and locally annotated Qwen stores. Figures 5 and 6 report online execution costs.

D.2 Online Tokens per Query

Figure 5 sums input and output tokens across online model requests, including context carried into later requests. OC-NaVQA usage is measured on the 60-query subset from sequences 0 and 21, with all four systems calling GPT-5.6-sol through Azure. FindingDory uses all 596 questions and one deployment across its four configurations.

On FindingDory, EDM reduces online tokens on both hosts; the ReSight executor uses slightly more tokens on the OC subset. GOAT-core retrieval-only makes no online model request during candidate selection, while the agent uses 6.7k tokens per query on average.

D.3 Online Model Requests per Query

Figure 6 counts every online executor model request: decisions to retrieve or inspect, answer submission, and any additional finalization. Multiple tool calls emitted by one model response count once. The task sets match Figure 5.

Both FindingDory hosts make fewer requests after mounting. ReSight makes slightly more on the OC subset. The decision limits for each host appear in Appendix C.3.

E Complete Execution Trajectories

We present paired execution trajectories before and after EDM for two OC-NaVQA queries, two FindingDory queries, and six real-robot task types. Each case shows the model requests, tool calls,

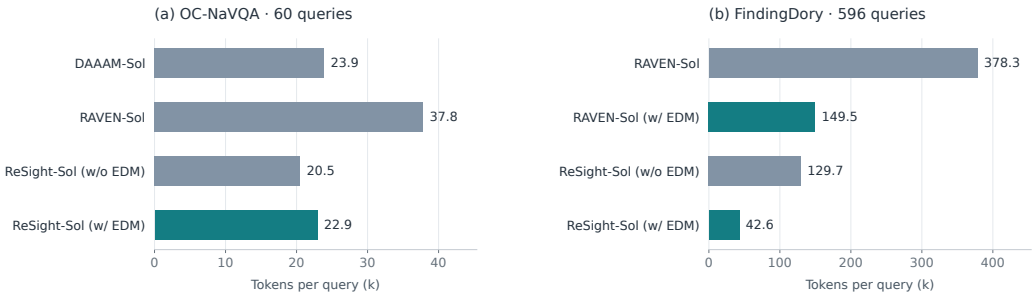


Figure 5: Mean online tokens per query (thousands). Gray denotes the base host and green its EDM-mounted counterpart. The two benchmarks use separate horizontal scales.

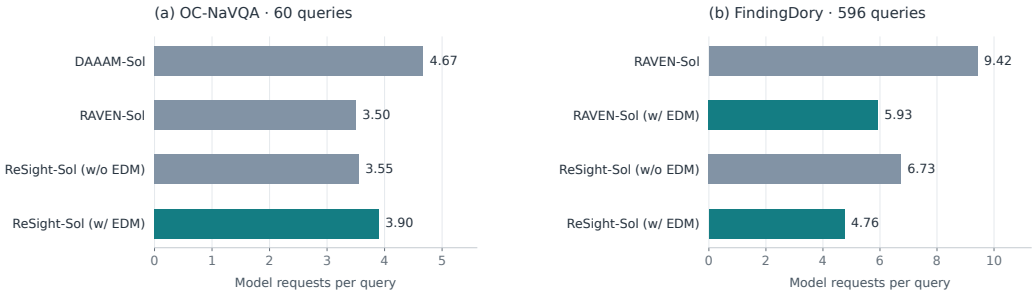


Figure 6: Mean online model requests per query. ReSight includes decision and finalization requests; RAVEN includes agent decisions and answer generation; DAAAM uses step-level request logs.

selected returns, visual evidence, and final answers. We then demonstrate how memory-query results support subsequent physical execution.

E.1 OC-NAVQA

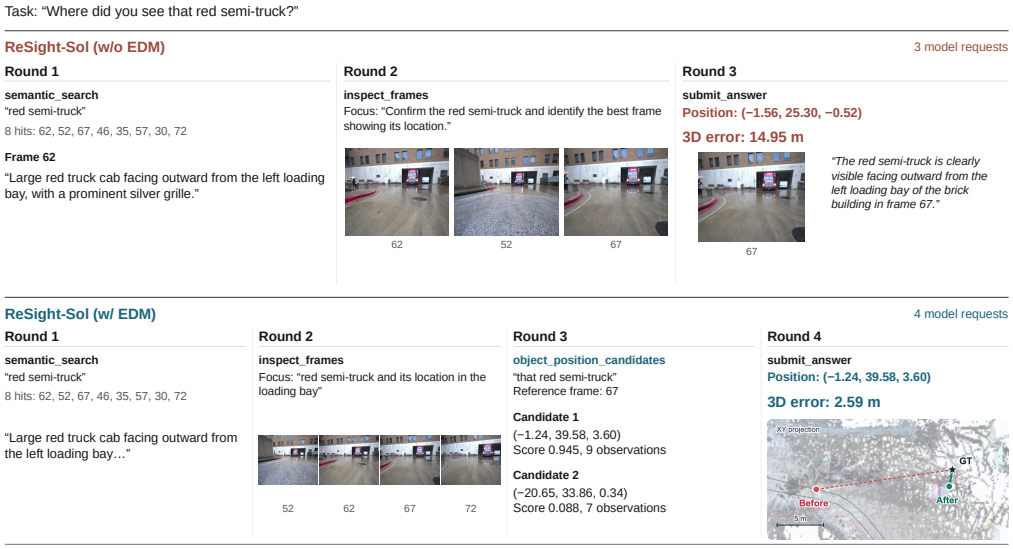


Figure 7: OC-NaVQA: red semi-truck. Both executions cite frame 67; EDM uses several views to estimate the truck's position rather than returning the camera pose.

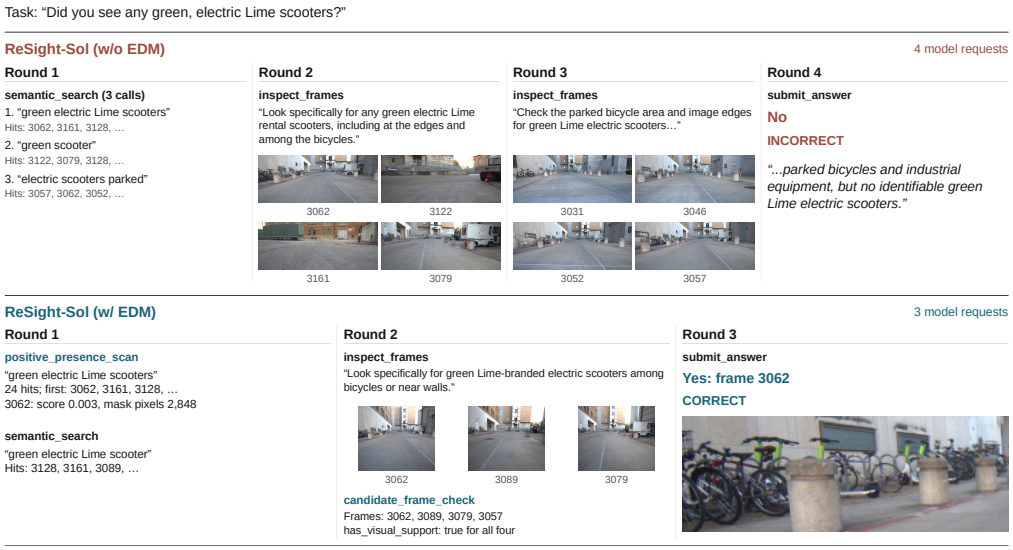


Figure 8: OC-NaVQA: green Lime scooters. The base execution misses the target. EDM's presence scan narrows the candidate frames for image verification.

E.2 FINDINGDORY

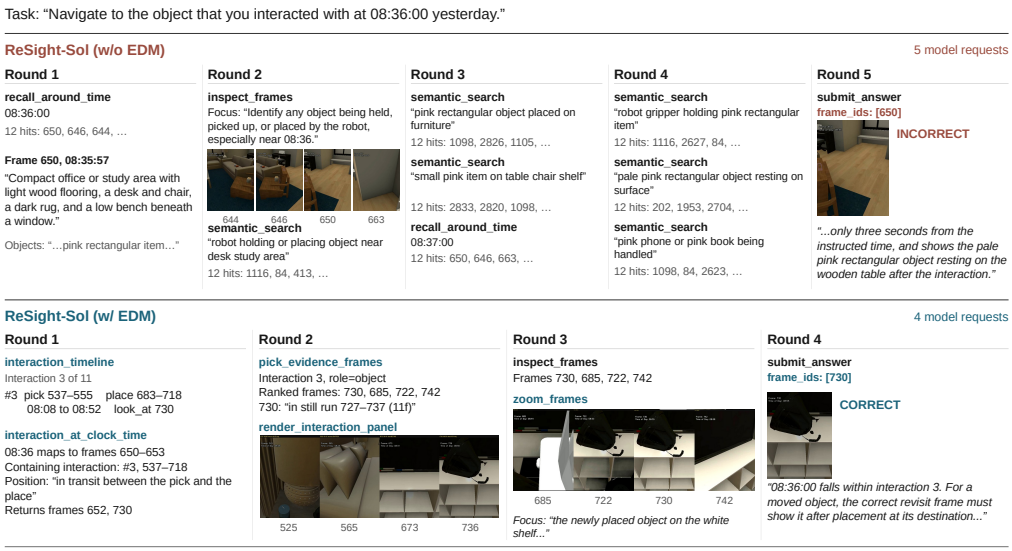


Figure 9: FindingDory: interaction time. The base execution cites a frame before placement; EDM identifies the interaction and cites evidence after placement.

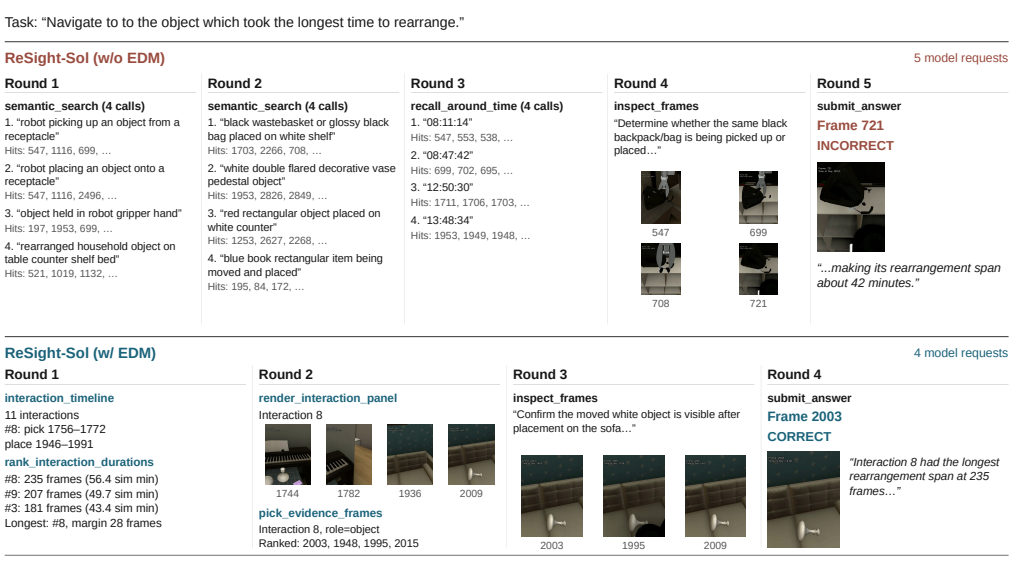


Figure 10: FindingDory: the object with the longest rearrangement. The base execution estimates duration from retrieved frames; EDM ranks indexed interactions by duration and checks the selected object's evidence.

E.3 Real Robot Patrol Memory

Task: "Where is the Warmth Corner? Find in the memory the wall sign that reads WARMTH CORNER and give a frame_id in which that sign is clearly legible."

ReSight-Sol (w/o EDM) — 8 model requests

Round 1
semantic_search
"wall sign reading WARMTH CORNER clearly legible"
Hits: 575, 610, …

Round 2
inspect_frames
"Which image clearly shows a wall sign whose exact text is WARMTH CORNER?"

296 610 238

Round 3
semantic_search
"WARMTH CORNER text sign community wall"
Hits: 238, 296, 608, …

Round 4
inspect_frames
"Find the exact wall text WARMTH CORNER."

296 219 201

Round 5
semantic_search
"employee recognition wall labeled WARMTH CORNER beside portraits and notes"
Hits: 295, 262, …

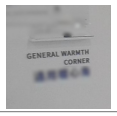
Round 6
inspect_frames
"Verify whether the wall sign WARMTH CORNER is clearly legible…"

295

Round 7
semantic_search
"exact English words WARMTH CORNER on a wall sign"
Hits: 610, 180, …

Round 8
Answer
Frame 295
INCORRECT
"...on the employee-recognition wall in the reception/office lobby..."

ReSight-Sol (w/ EDM) — 2 model requests

Round 1
find_sign_text_in_memory
"WARMTH CORNER", top_k: 3
Text: "GENERAL WARMTH CORNER"
Frames: 277, 241, 217, 241
Sign position: (21.10, −6.14)


Round 2
Answer
Frame 277
CORRECT
"It is clearly legible in frame_id 277, reading 'GENERAL WARMTH CORNER.'"


Figure 11: Robot patrol, held-in: Warmth Corner sign. The base execution selects an unrelated recognition wall; text lookup returns a frame with the legible sign.

Task: "Taking the patrol start (the position of the first frame) as the reference, where on the whole trajectory is the point farthest from the start in straight-line distance? Give the frame_id at that point and the straight-line distance in meters."

ReSight-Sol (w/o EDM) — 6 model requests

Round 1
semantic_search
"patrol trajectory first frame start position farthest point straight-line distance"
Hits: Scene descriptions

Round 2
semantic_search
"earliest first frame beginning start of patrol trajectory"
Hits: 578, 283, 159

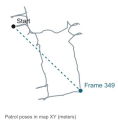
Round 3
semantic_search
"open plan office desks computer monitors rolling chairs far end of patrol"
Hits: 388, 514, 599

Round 4
semantic_search
"conference table chairs floor-to-ceiling windows beginning of patrol"
Hits: 41, 276, 241

Round 5
semantic_search
"most distant office location at far southeast end of the patrol route near x 22 y minus 22"
Hits: 420, 174, 351

Round 6
Answer
Frame 351
33.33 m
INCORRECT

ReSight-Sol (w/ EDM) — 2 model requests

Round 1
patrol_path_geometry
mode=extremes, reference=start, top_k=5
Start (0.01, 0.01) → frame 349 (22.43, −21.49)
31.068 m; runners-up: 350, 355, 351, 352


Round 2
Answer
Frame 349 · 31.068 m
CORRECT

Figure 12: Robot patrol, held-in: farthest point. EDM computes the farthest frame from all 648 stored poses. The plotted trajectory is reconstructed from stored poses.

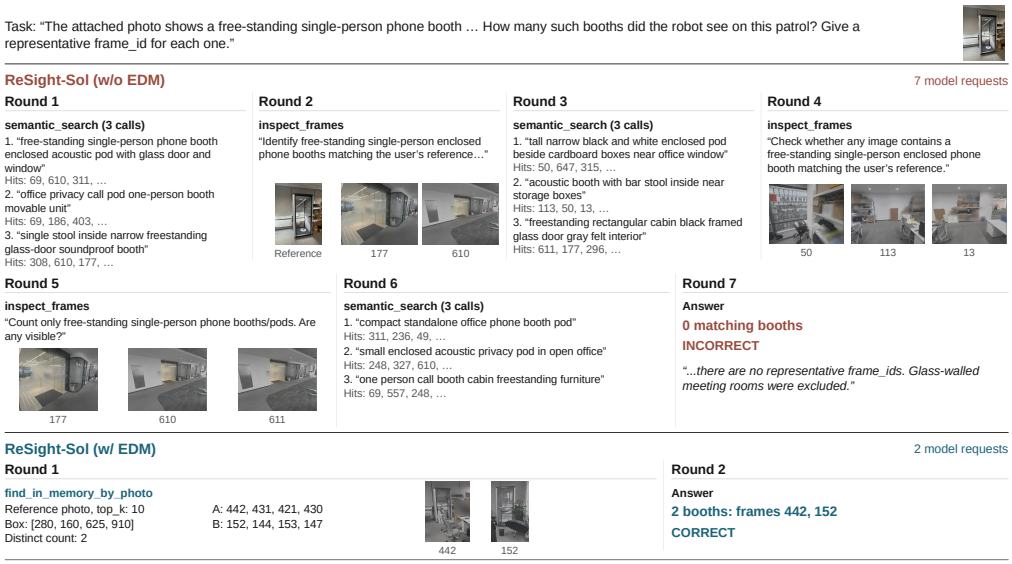


Figure 13: Robot patrol, held-in: booth count. With the same reference photo, EDM returns evidence for both instances; the base execution answers zero using an image of a glass meeting booth.

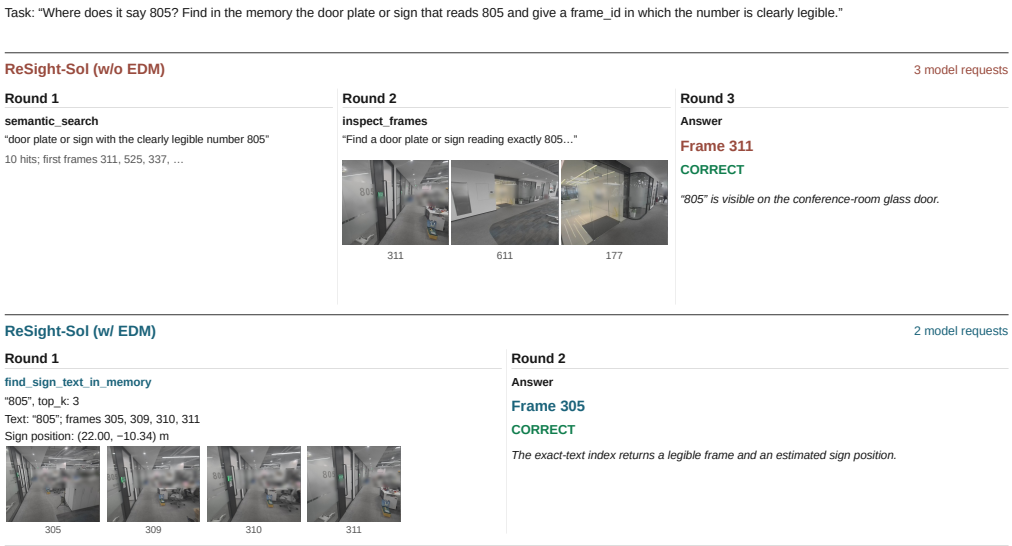


Figure 14: Robot patrol, held-out: room 805 sign. Both executions answer correctly; the mounted text index reduces the count from three model requests to two.

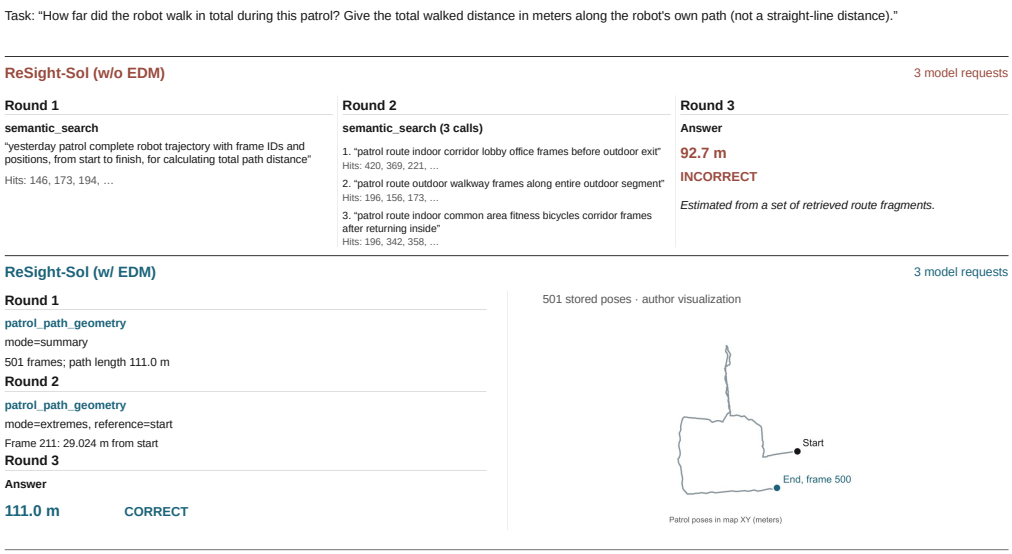


Figure 15: Robot patrol, held-out: total distance walked. The base execution estimates 92.7 m from route fragments; EDM sums consecutive positions across 501 frames to obtain 111.0 m. The plotted trajectory is reconstructed from stored poses.

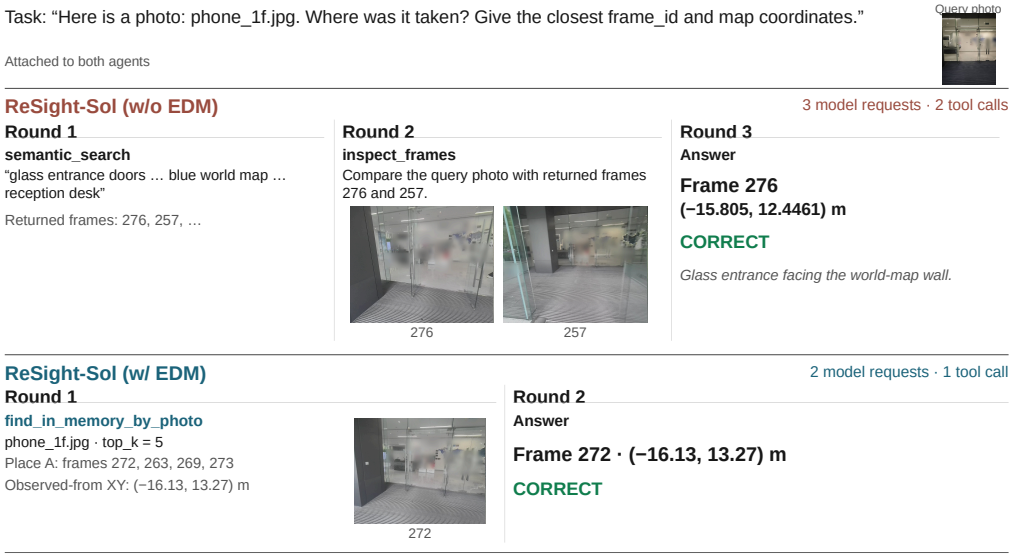


Figure 16: Robot patrol, held-out: independent phone-photo localization. Both executions answer correctly. In this paired run, EDM uses one tool call and two model requests, versus two tool calls and three model requests without EDM.

E.4 From Memory Queries to Physical Execution

The following demonstrations continue the booth-counting and photo-localization queries with physical actions requested by the user. Retrieved locations guide navigation, while fresh observations support on-site verification.

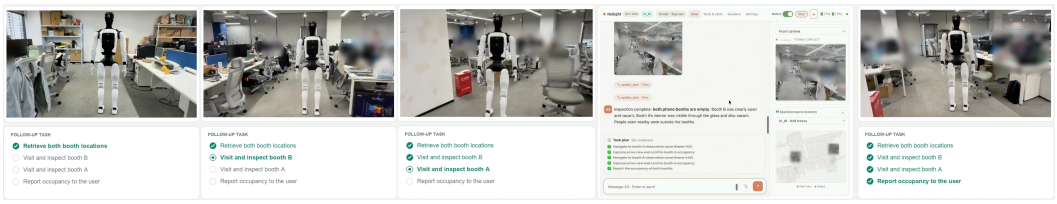


Figure 17: Visiting and inspecting two phone booths on floor 8. After the memory query identifies two booths, the user asks: "Help me plan a route to visit these two phone booths right now and confirm in person whether anyone is there." The robot visits booth B and then booth A, checks their current occupancy, and reports that both are empty.

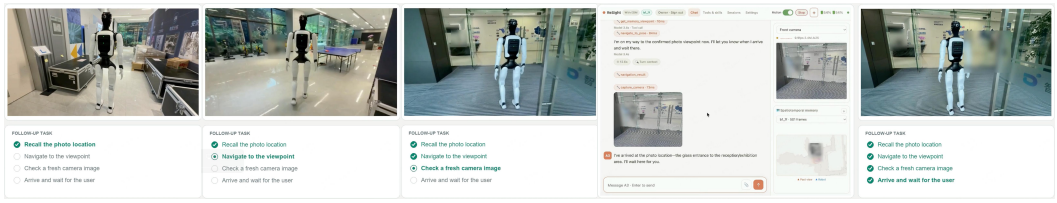


Figure 18: Navigating to a photo-matched location on floor 1. After the memory query locates the reference photograph, the user asks: "Go there and wait for me." The robot navigates to the retrieved viewpoint, checks a fresh camera image, and waits at the destination.