SOMo: Unified Spatial and Observation-Grounded Scene Memory

Anonymous authors
Paper under double-blind review

Abstract

Questions about a robot's past observations often identify an object through its relationships to other objects and the context of a recorded encounter. An agent answering such questions through separate searches must assemble compatible objects and observation evidence from intermediate results; an early selection can exclude the intended answer and misdirect subsequent tool calls. We introduce SOMo, a scene memory that lets spatial relationships and observation context guide object selection during retrieval. Through a shared interface, the agent specifies object descriptions, required relationships, and spatial and temporal restrictions. SOMo jointly selects compatible objects and returns their mapped locations with the requested observation evidence, supporting proximity, co-observation, and robot-relative bearing queries. A sparse spatial hierarchy connects object groups to recorded robot visits and summarizes their spatial extent and observation intervals, allowing constraints to eliminate incompatible groups before individual verification. Adaptive planning selects the grouping resolution and constraint order using execution-cost estimates and counts of the object pairs remaining after grouped filtering, before expanding those pairs. Joint semantic ranking then selects among the verified combinations. With Qwen3.5-9B for both methods, SOMo reduces positional and temporal error relative to DAAAM by 45.3% and 37.6% on NaVQA, and by 68.09% and 64.32% on CODaAG's category averages. On SG3D, SOMo with Qwen3.5-9B achieves 31.82% sub-task grounding accuracy, compared with 22.16% for DAAAM with GPT-5-mini. On CODaAG, average tool use falls from 5.62 to 1.46 calls and answering time from 31.26 to 6.29 seconds. Separate candidate-generation benchmarks show 7–25$\times$ speedups over blocked NumPy scans on CODa and 16–193$\times$ on KITTI-360 with identical candidate results.

1 Introduction

Mobile robots deployed for long periods accumulate an observation history that supports tasks beyond their current view. A robot navigating a campus, for example, may encounter many bikes and benches and revisit the same locations over time. A user might ask, "Where did you see my bike? I left it near a bench," possibly also specifying when or where to look. The answer can guide the robot's next destination or its execution of a subsequent instruction (Zhang et al., 2024; Werby et al., 2024). Answering such questions requires more than recognizing the named object. Among many plausible bikes, the relevant one is identified not simply by its own appearance, but also by its relationship to the bench beside it. An observation period can further distinguish the intended encounter across repeated visits. As the scene and observation history grow, identifying the intended objects and encounters increasingly depends on combining object descriptions, spatial relationships, and observation context.

To retrieve information accumulated during extended operation, existing memory systems often provide tools that an LLM agent can invoke, while leaving much of the contextual reasoning to the agent. For example, ReMEmbR searches time-stamped video captions by semantic similarity (Anwar et al., 2024), while DAAAM provides explicit object and trajectory information through tools for object search, observation times, locations, and the traveled path (Gorlo et al., 2025). These tools retrieve relevant records, but leave the agent to determine which combinations satisfy the question's constraints. For the bike question, the agent may search for bikes and benches separately, or first select a candidate bench and then search for nearby bikes. It must establish which pair satisfies the

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

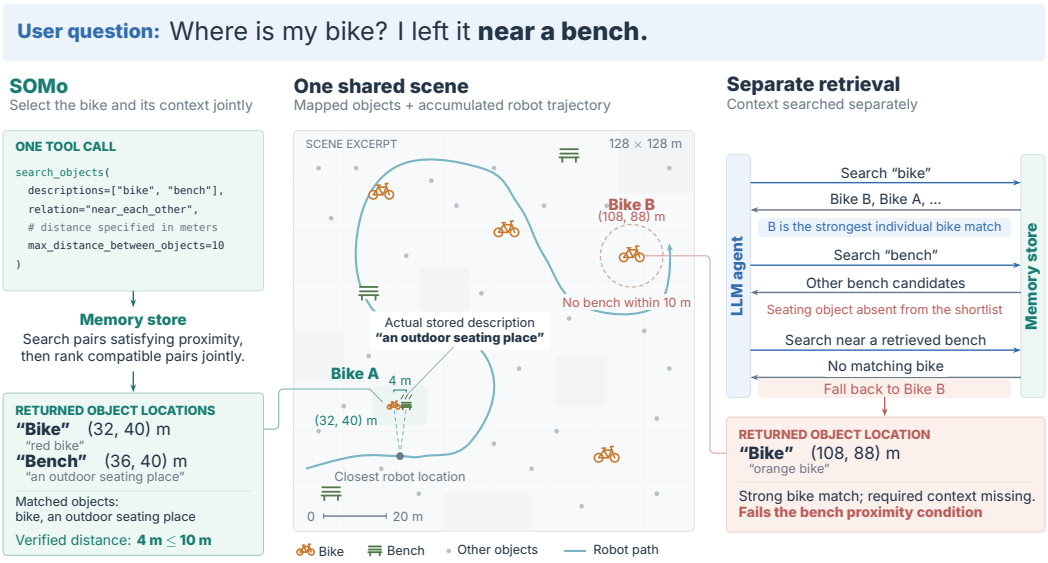


Figure 1: **Context identifies the intended object.** A user asks for a bike near a bench; the agent expresses "near" as a 10 m proximity threshold. In the shared scene, Bike A is 4 m from an object described as "an outdoor seating place." SOMo accepts both descriptions and the proximity condition in one retrieval call, jointly selecting compatible objects and returning Bike A's mapped position with supporting context. A separate-search agent misses the seating object in its bench shortlist and, after an unsuccessful neighborhood search, falls back to a strong individual bike match. The selected Bike B has no bench within 10 m. The scene, candidate ordering, and failure sequence are constructed illustrations, not measured execution results.

requested spatial and observation conditions. Limiting the candidates returned by these intermediate searches can exclude the correct combination, while retrieving broader candidate sets increases both the observation evidence to inspect and the number of candidate combinations whose relationships may need checking. This process can require multiple tool calls, and an incorrect intermediate selection can misdirect subsequent searches and propagate to the final answer. These questions therefore call for search constraints and contextual relationships to guide candidate selection *during* retrieval, allowing the relevant objects and observation evidence to be leveraged jointly.

We introduce SOMo, a scene memory that enables an LLM agent to retrieve objects together with their observation context under a shared set of conditions. Through a single `search_objects` interface, the agent specifies object descriptions, required relationships, and optional constraints on the search area and observation period. The interface supports individual-object searches and joint retrieval of groups of two to four objects. Multi-object requests can specify proximity, co-observation, or both; single-object requests additionally support robot-relative bearing during an observation. For example, the agent can request a bike near a bench, with an additional requirement that they were observed together during a specified period. SOMo identifies pairs satisfying these conditions against stored spatial and observation evidence, then ranks them by how well their members match both descriptions. The bike and bench are selected jointly, allowing their relationship and observation context to inform which objects are returned. Figure 1 illustrates how joint retrieval identifies a compatible bike–bench pair that separate searches can miss. Results include the objects' mapped locations and, when requested, the associated observation times and robot locations. This gives the agent direct access to candidates that satisfy the combined request, reducing the need to commit prematurely to a reference object, inspect long separate candidate lists, or reconstruct the relevant encounter through additional tool calls.

Evaluating searches at the level of individual objects becomes increasingly costly as the scene and observation history grow. Without a way to rule out objects collectively, even filtering by an area or observation period requires checking each object's record to determine whether it qualifies. For requests involving multiple objects, examining possible combinations introduces additional work that grows quadratically for pairs and more rapidly for larger object combinations. To accelerate constraint and relationship checks through grouping, SOMo uses Morton (Z-order) codes to organize

spatial cells at multiple resolutions. Rather than allocating the full grid at each resolution, it stores only cells containing observed objects, together with their coarser parent cells. This yields a sparse hierarchy that provides fine and coarse groupings of the same stored objects. Each level references shared object records, allowing different groupings to retain access to the same underlying evidence without duplicating those records. Alongside object cells, SOMo maintains trajectory cells that track distinct robot visits to the cell region. At each resolution, bidirectional links connect recorded visits to the object cells containing their associated objects. These links allow observation conditions to narrow candidate groups before individual objects are examined, while retaining access to the underlying encounters. These connections support searches for objects observed from a specified region or during a specified period, as well as recovery of their associated observation locations and visits. Each object cell also summarizes its object members' spatial extent and observation times, allowing constraints to eliminate incompatible cells or cell combinations collectively. For example, cell pairs whose spatial bounds are farther apart than the permitted distance can be discarded without checking their member pairs. The underlying object and observation records remain available to verify the surviving candidates.

Given this grouped representation, SOMo first filters cells or cell combinations, then expands the surviving candidates to examine individual objects or object combinations. To reduce search cost, SOMo selects both a spatial resolution and the order in which spatial and observation constraints guide retrieval. Compared with a finer resolution, a coarser resolution reduces the number of cells to process but can leave more individual object combinations to examine after expansion. The order in which search constraints are applied introduces another tradeoff. For requests combining proximity and co-observation, SOMo can first identify potentially nearby cell pairs (proximity-first filtering) and then apply observation conditions, or first identify cell pairs linked through relevant visits (observation-first filtering) and then apply spatial conditions. Dense cells can leave many object pairs to verify, while visits linking many cells, or repeatedly linking the same cell pairs, can increase candidate-generation work. A suitable resolution and constraint order therefore depend on constraint selectivity, cell occupancy, and the distribution of object–visit associations. For example, a restrictive observation window may favor starting from the relevant visits when spatial candidate generation would expose many cell pairs in a densely populated scene. For each query, SOMo uses occupancy statistics and cumulative visit-association counts stored at each resolution to estimate candidate-generation, filtering, and expansion work. These estimates guide the choice of resolution and constraint order. Before committing to individual verification, SOMo applies the constraints to grouped candidates and determines how many object pairs would remain to be examined. This allows it to assess promising search strategies using their remaining workload and memory requirements, then expand and verify the candidates of the selected strategy. Figure 2 illustrates this process, from grouped filtering and candidate counting to individual verification and joint ranking. SOMo jointly ranks the feasible object combinations by how well their members match the requested descriptions and returns the highest-ranked results to the LLM agent. This reduces the agent's need to establish relationships across separate candidate lists and determine which combination best matches the request.

We evaluate SOMo on spatiotemporal question answering and sequential object grounding across indoor and outdoor scenes, including queries that require spatial relationships and observation context. On CODaAG, SOMo reduces category-average positional error from 70.66 to 22.55 m and temporal error from 3.12 to 1.11 min relative to DAAAM, with Qwen3.5-9B for both methods. On SG3D, SOMo with Qwen3.5-9B achieves 16.91% complete-task grounding accuracy, compared with 11.22% for DAAAM with GPT-5-mini, demonstrating benefits for identifying the objects required throughout multi-step instructions. SOMo also reduces the agent effort needed to answer questions, lowering average tool use from 5.62 to 1.46 calls and average answering time from 31.26 to 6.29 seconds on CODaAG. Separate candidate-generation experiments show 7–25$\times$ speedups on CODa and 16–193$\times$ on KITTI-360 over blocked NumPy scans with identical candidate results, demonstrating that the memory can resolve the requested constraints efficiently.

2 Related Work

2.1 Scene Memory and Retrieval

Scene graphs organize objects and relationships for semantic mapping and embodied reasoning (Hughes et al., 2022; Gu et al., 2023; Saxena et al., 2024), while hierarchical maps provide abstractions for navigation and task grounding (Werby et al., 2024; Deng et al., 2024; Maggio et al., 2024). Open-vocabulary maps and fields also support language-conditioned spatial search (Yamazaki et al., 2023; Shafiullah et al., 2022). Long-horizon robot memories extend access to past observations: ReMEmbR retrieves time-stamped captions, and RAVEN retains images with pose and time for visual inspection (Anwar et al., 2024; Hu et al., 2026). DAAAM builds an object-centric scene memory with descriptions, observation timelines, and region and trajectory information, exposed through retrieval tools (Gorlo et al., 2025). SOMo uses this perception and object-memory foundation. For contextual requests, separately retrieving objects and their histories can require the agent to select reference objects, test relationships, and recover the relevant encounter through successive calls. These intermediate selections can exclude a compatible answer before its full context is considered.

Recent systems make memory retrieval more context-sensitive. Embodied-RAG organizes semantic abstractions for hierarchical retrieval; 3D-Mem constructs object-associated snapshots; and STaR selects compact, task-relevant evidence from linked multimodal memory (Xie et al., 2024; Yang et al., 2024; Yuan et al., 2026). Embodied VideoAgent and eMEM retain object or entity associations with observations and provide structured retrieval (Fan et al., 2024; Rasheed et al., 2026). R4 combines semantic, spatial, and temporal retrieval keys, while SpatialMem checks geometric relationships during grounding (Sohn et al., 2025; Zheng et al., 2026). These works establish that observation associations and contextual filtering are useful components of memory retrieval. SOMo focuses on selecting compatible object combinations under a shared request: descriptions define roles, spatial and observation conditions determine feasibility, and joint semantic ranking selects among feasible assignments. Thus, a reference object and its target are selected together, with the requested observation evidence returned alongside their mapped locations. The backend resolves supported relationships directly over stored records, reducing the combinations and historical evidence the agent must assemble itself.

2.2 Spatial Reasoning and Grounding

Relational context also guides language grounding: ReferIt3D studies identifying objects among similar instances using descriptions and surrounding objects (Achlioptas et al., 2020). Spatial foundation models such as VST, SenseNova-SI, and SpatioLM strengthen geometric understanding from visual inputs (Yang et al., 2025; Cai et al., 2025; Wu et al., 2026), while OmniStream processes continuous visual streams (Yan et al., 2026). These advances complement retrieval over accumulated experience. SOMo addresses which stored objects and encounters satisfy a request, including its observation window and robot context. Its contribution lies in making those conditions guide retrieval and returning their supporting evidence, which a reasoning model can use to formulate an answer.

3 Method

SOMo resolves contextual object requests through a shared memory of scene objects and recorded encounters. Spatial groupings and observation associations allow the requested conditions to narrow the search collectively, while adaptive execution controls the individual combinations that remain to be examined. Joint semantic ranking then selects the combinations that best match the descriptions. Figure 2 traces the process from shared memory through plan assessment, pair verification, and joint ranking for a bike and a bench observed together within 5 m of each other.

3.1 Constrained Object Retrieval

Through `search_objects`, the agent specifies ordered descriptions $q_1, \ldots, q_k$, with $1 \leq k \leq 4$, and the conditions identifying an answer. Each description defines a semantic role to be filled by a distinct object. Multi-object requests specify proximity, co-observation, or their conjunction; single-object requests additionally support robot-relative bearing at a recorded encounter. Spatial and

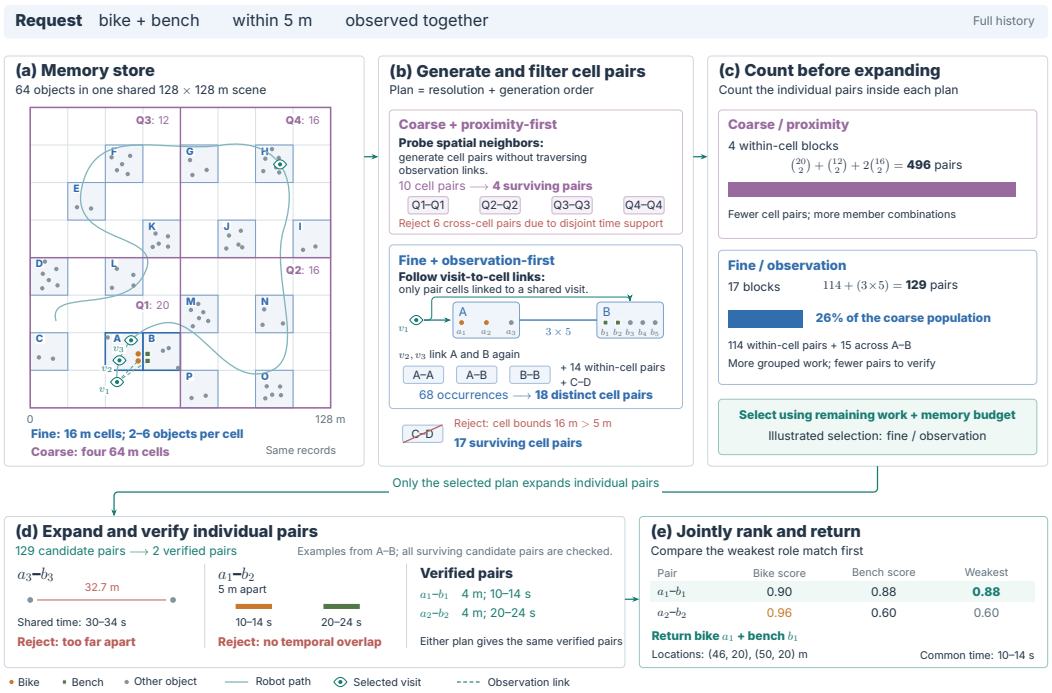


Figure 2: **Assess execution alternatives before expanding object pairs.** From a shared memory (a), SOMo jointly considers cell resolution and proximity-first or observation-first generation (b). Work estimates nominate promising plans; normal pair planning assesses at most two through grouped generation, filtering, and counting (b–c). Only the selected plan expands and verifies individual pairs (d), while valid plans preserve the same verified answers for joint ranking (e). The example illustrates how fewer cell pairs can hide a larger individual workload. The scene and plan selection are illustrative; bars show candidate populations rather than measured runtime.

temporal restrictions specify where to search and which observations are relevant. An object search area restricts mapped object positions, while a robot observation area selects encounters from a region of the robot's trajectory. An observation window restricts the relevant history, and co-observation requires common support within that history.

The retrieval objective is to return the best matching object combinations satisfying the requested conditions, with members assigned to the description roles. For the running example, the bike and bench must satisfy both proximity and co-observation, and their descriptions determine which compatible pair ranks highest. Results preserve description order and include mapped object locations and requested observation information. Appendix E specifies the interface and its relation-specific inputs and outputs.

3.2 Scene and Observation Memory

Objects and recorded encounters. Object o_i stores a mapped position p_i, semantic features, and an observation history represented by an interval union H_i. Trajectory sample s records the robot position r_s, timestamp t_s, and heading when available. A *visit* is a maximal consecutive run of samples within one finest persisted trajectory cell. Returning after leaving creates a distinct visit, preserving repeated encounters with the same place. Let R_v contain the object indices associated with visit v. These associations support retrieval from visits to objects and from objects to their recorded encounters. The intervals H_i supply temporal support for checking individual objects and common observations; Appendix F describes their construction from the recorded history.

Spatial groupings at multiple resolutions. SOMo uses Morton (Z-order) codes to organize occupied spatial cells at multiple resolution levels. Level ℓ has cell width $h_\ell = 2^\ell h_0$, with level zero the finest. The representation supports horizontal XY cells and volumetric XYZ cells. Each object belongs to one cell at each level; writing $g_\ell(i)$ for its cell, the membership set is $\mathcal{O}_\ell(a) = \{i : g_\ell(i) = a\}$. These disjoint sets provide fine and coarse groupings of the same objects. Levels reference shared

object records, so retrieval can select a grouping directly while retaining the positions, features, and observation evidence needed for verification. An individual-object partition, with one object per group, provides an additional choice when direct processing is economical.

Summaries and observation associations. Each object cell stores an enclosing spatial box and the union of its members' observation intervals:

$$B_\ell(a) \supseteq \{p_i^{\mathrm{idx}} : i \in \mathcal{O}_\ell(a)\}, \qquad U_\ell(a) = \bigcup_{i \in \mathcal{O}_\ell(a)} H_i. \tag{1}$$

Here p_i^{idx} denotes the coordinates used by the spatial index. These summaries let constraints act on all members collectively: spatial separation between boxes bounds the distance between their objects, and interval unions expose possible observation overlap. Object–visit associations also induce visit-to-cell links $I_\ell(v) = \{g_\ell(i) : i \in R_v\}$. A link identifies a cell containing an object associated with that visit. Parent cells aggregate their descendants' summaries and links, while trajectory views retain the canonical visits represented by their chronological runs. This connects spatial groupings to encounter history at each resolution, as illustrated in Figure 4.

3.3 Grouped Constraint Evaluation

We first describe retrieval at a given resolution and constraint order. Let A be the object indices eligible under the request's individual restrictions. For an object pair $i, j \in A$, $i < j$, proximity and co-observation respectively require

$$\mathrm{dist}(p_i, p_j) \le d, \qquad H_i \cap H_j \cap W \neq \emptyset, \tag{2}$$

where d is the requested distance threshold, W is the observation window, and dist uses the configured XY or XYZ metric. A combined request requires both conditions. For proximity with a window, eligibility requires each member to have support in W. A robot observation-area restriction supplies selected visits, and temporal pairs additionally require both members to be associated with a common selected visit. These conditions define the pair relation that supplies two-object answers and the edges used to construct larger combinations. Appendix E.2 specifies the observation-area evidence domain.

Generating and filtering cell pairs. The constraint order determines how candidate combinations are first discovered. *Proximity-first* retrieval searches occupied cells in spatial neighborhoods consistent with d, using their boxes to reject incompatible cell pairs. *Observation-first* retrieval forms cell pairs through the links $I_\ell(v)$ for relevant visits. Repeated occurrences of a cell pair across visits are consolidated into one candidate. Both routes include within-cell combinations when multiple eligible objects can participate, then apply the remaining constraints to the grouped candidates.

For example, a cell pair can be rejected when the minimum distance between its boxes exceeds d, or when $U_\ell(a) \cap U_\ell(b) \cap W$ is empty for a co-observation request. These tests eliminate entire blocks of impossible object pairs while preserving combinations that may satisfy the request. Surviving candidates retain their cell memberships until expansion: distinct cells contribute cross-products of eligible members, and a cell paired with itself contributes its distinct member pairs. Individual verification then checks the original positions, observation intervals, and applicable selected-visit evidence. This resolves the remaining member-level conditions and produces the verified pair relation.

Single-object and encounter queries. Single-object area and time searches use cell bounds and interval unions to narrow eligible members before checking their individual positions and support. A robot observation-area request searches trajectory cells and runs, checks recorded samples against the requested region and window, and retrieves objects associated with the selected visits. These operations connect spatial restrictions on the robot's encounters to the object records used by subsequent retrieval.

Robot-relative bearing search identifies objects with qualifying encounters along the trajectory. Relevant runs and their observation associations supply candidate objects and recorded samples; SOMo checks sample-time membership in each object's stored support, range, and heading-relative angle. Computation is organized into local object-by-sample blocks, and each object retains its best qualifying event. Windowed search uses a dedicated trajectory-level execution policy, while a reference-time request evaluates the corresponding resolved sample. Appendices F.4.4 and F.4.5 describe these specialized searches.

3.4 Adaptive Query Planning

The cost of grouped pair retrieval depends on both resolution and constraint order. Coarse cells reduce group-level processing but can expose many member combinations; finer cells provide tighter filtering at greater group-level cost. Observation-first generation depends on the number of cells linked to relevant visits and on how often their pairs recur. SOMo therefore treats the object partition and constraint order as a joint execution choice, adapting it to cell occupancy and the spatial and observational selectivity of the request.

Estimating promising executions. The memory stores occupancy statistics at each resolution, including group counts and membership-size statistics, together with cumulative visit-association counts. A request uses these summaries to obtain counts for relevant visits and estimate spatial candidate generation, repeated visit-based occurrences, collective filtering, and member expansion. The resulting work estimates identify promising plans for assessment, while conservative allocation bounds determine whether their grouped processing fits the working-memory budget.

Counting before expansion. Assessment performs candidate generation and collective filtering while spatial candidates remain represented as cell pairs. For an assessed plan at level ℓ, let F_ℓ contain the unique surviving cell pairs, and let $n_a = |\mathcal{O}_\ell(a) \cap A|$ be the number of eligible members in cell a. The exact number of individual object pairs represented by these candidates is

$$P_\ell = \sum_{\substack{(a,b)\in F_\ell \\ a<b}} n_a n_b + \sum_{(a,a)\in F_\ell} \binom{n_a}{2}. \tag{3}$$

Disjoint memberships assign every individual pair to exactly one cell-pair block, so the sum counts the prospective verification population before its object pairs are enumerated. Figure 2(c) illustrates the consequence: seventeen surviving fine-cell relations represent 129 object pairs, whereas four coarse-cell relations represent 496. Assessment thus exposes the individual workload hidden inside a compact grouped relation. The count replaces the initial expansion estimate and determines the population used to size expansion and subsequent verification arrays.

Selecting the remaining work. SOMo uses the assessed pair count to compare remaining verification work and memory requirements, including regeneration cost when an assessed relation must be rebuilt. Normal pair planning assesses at most two promising alternatives across resolution levels and constraint orders. The selected spatial relation then undergoes member expansion and the remaining individual checks; individual-object candidates reuse evidence already established during their assessment. This allows execution to adapt to each request while preserving the verified relation over the same stored evidence. Appendices F.4.3 and F.5 provide the assessment policy and execution details.

3.5 Joint Semantic Ranking and Returned Evidence

Verified pairs form edges in a graph over eligible objects. A two-object candidate is an edge, while a larger candidate requires an edge between every pair of members. For proximity, this establishes all-pairs compatibility. For co-observation, SOMo also checks common support for the complete combination, ensuring its members share a qualifying observation context. Ordinary requests use the stored interval unions, while observation-area requests use the selected-visit support defined in Appendix E.2.

For each feasible combination C of k objects, semantic role assignment determines which member matches each description. Let $s_{i,r}$ be the similarity between object i's stored semantic features and description q_r, and let $\Pi(C)$ contain the bijections from roles to members of C. SOMo selects the assignment

$$\pi_C^* = \operatorname*{arg\,max}_{\pi\in\Pi(C)}{}_{\text{lex}}\, \operatorname{sort}_\uparrow\big(s_{\pi(1),1}, \ldots, s_{\pi(k),k}\big). \tag{4}$$

Combinations are ranked by these ascending score vectors using *leximin*: improve the weakest description match first, then use the next weakest matches to resolve ties. This evaluates the target and its contextual objects together. For example, scores $(0.90, 0.88)$ outrank $(0.96, 0.60)$ because both roles receive a strong match. Pair and triple search evaluate their role assignments explicitly.

Table 1: Results on NaVQA benchmark (Anwar et al., 2024). We additionally evaluate ReMEmbR, RAVEN, and DAAAM with Qwen3.5-9B, and test the spatial VLMs SenseNova-SI and Spatio-LM on NaVQA. The remaining baseline results are taken from DAAAM (Gorlo et al., 2025). SOMo results are reported as means with standard deviations over three seeded runs or repeated API calls, as appropriate. We report descriptive question accuracy, positional error, and temporal error across short, medium, long, and overall query contexts. Best is bold and second-best is underlined.

	Metric:	Descriptive Question Accuracy ↑				Positional Error [m] ↓				Temporal Error [min] ↓			
	Method	Short	Medium	Long	All	Short	Medium	Long	All	Short	Medium	Long	All
ReMEmbR	NVILA-Lite-8B + GPT-5-mini	0.538	0.630	0.571	0.582	8.467	53.279	53.717	41.485	0.436	1.336	4.122	2.414
ReMEmbR	VILA1.5-13b + GPT-4o	0.62	0.58	0.65	0.61	<u>5.1</u>	27.5	46.25	29.96	0.3	1.8	3.6	2.3
ReMEmbR	NVILA-Lite-8B + Qwen3.5-9B	0.619	0.605	0.650	0.620	8.144	39.297	51.143	36.299	<u>0.253</u>	1.592	4.981	2.847
Multi Frame VLM | GPT-4o		0.55	×	×	×	7.5	×	×	×	0.5	×	×	×
Concept Graphs NVILA-Lite-8B + GPT-5-mini		0.385	0.296	0.143	0.299	85.67	126.35	158.67	130.03	×	×	×	×
RAVEN | Qwen3.5-9B		0.619	0.474	0.700	0.580	9.24	18.36	39.43	25.416	**0.24**	1.36	2.43	1.595
SenseNova-SI		0.667	**0.711**	0.700	0.690	33.731	65.568	72.168	60.023	1.624	2.485	4.102	3.033
Spatio-LM		0.690	0.684	**0.800**	0.710	39.634	58.259	79.934	63.044	1.421	2.689	3.461	2.753
DAAAM | GPT-5-mini		0.654	0.630	0.786	0.672	7.282	46.015	42.116	33.89	0.443	1.030	2.538	1.591
DAAAM | Qwen3.5-9B		0.643	0.658	0.750	0.670	7.581	44.736	45.649	34.360	0.531	1.319	2.349	1.613
SOMo | GPT-5-mini		**0.724** ± 0.105	<u>0.705</u> ± 0.123	<u>0.790</u> ± 0.178	**0.730** ±0.124	6.704 ± 1.673	**16.180** ± 2.491	**25.849** ± 3.192	**18.002** ±2.967	0.365 ± 0.086	<u>0.683</u> ± 0.182	**1.501** ± 0.203	**0.987** ±0.186
SOMo | Qwen3.5-9B		<u>0.714</u> ± 0.095	0.689 ± 0.108	0.780 ± 0.156	<u>0.718</u> ± 0.109	**5.013** ± 1.241	<u>17.750</u> ± 3.262	<u>27.624</u> ± 3.471	<u>18.792</u> ±3.013	0.382 ± 0.097	**0.654** ± 0.134	<u>1.554</u> ± 0.177	<u>1.007</u> ±0.145

Four-object search combines physical graph reduction, common-support checks, and safe semantic pruning with exact role assignment. Appendix F.6 gives the combination-search algorithms.

Returned records restore the requested description order and include object identities, mapped positions, and relation evidence. For co-observation, common-support intervals supply a representative event time, which is resolved against the recorded trajectory when robot context is requested. Proximity-only results can provide the robot's closest approach to the group's mapped centroid, while bearing results carry their qualifying recorded event. This gives the agent both the jointly selected objects and the context needed to use them. Appendices E and F.6.4 detail evidence materialization and the ranking policies for single-object and bearing queries.

4 Experiments

We evaluate spatiotemporal question answering on NaVQA and OC-NaVQA, contextual retrieval on CODaAG and SG3DAG, and sequential grounding on SG3D. Memory is built from observation streams using DAAAM's object-centric mapping pipeline. SOMo results with error bars report the mean and standard deviation over three repetitions: seeded runs for Qwen3.5-9B or repeated API calls for GPT-5-mini. Appendix C.1 specifies datasets and scoring; Appendices C.2 and C.3 give comparison settings and timing protocols. Additional results appear in Appendix C.4.

4.1 Question answering and grounding

Contextual question design. CODaAG (CODa-Autogen) tests questions whose answers depend on spatial or observation context. Its 195 questions span 13 types in four categories: single-object context, multi-object co-observation, proximity, and robot-relative bearing, asking for object locations, observer locations, or event times. Appendix D describes generation and lists all formulations in Table 10; SG3DAG extends them indoors.

NaVQA accuracy. With Qwen3.5-9B for both methods, SOMo raises descriptive accuracy from DAAAM's 0.670 to 0.718 and reduces positional and temporal error by 45.3% and 37.6% (Table 1). Positional gains are larger in medium and long contexts (60.3% and 39.5%) than short contexts (33.9%), although SOMo does not lead every individual metric. Appendix G illustrates contextual selection among similarly described objects.

Table 2: CODaAG results across four contextual query categories. Pos. is positional error (m); Temp. is temporal error (min); lower is better. Category Average is the unweighted mean across categories. Best is bold and second-best is underlined.

Method	Single-Object		Multi-Object Co-Observation		Multi-Object in Proximity		Robot-Relative Angle		Category Average	
	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓
ReMEmbR | Qwen3.5-9B	50.92	3.597	75.62	2.965	104.79	4.107	65.28	2.786	74.153	3.364
DAAAM | Qwen3.5-9B	36.61	2.126	89.62	3.623	92.67	3.762	63.73	2.967	70.657	3.119
SOMo | GPT-5-mini	<u>17.71</u>	**0.932**	<u>22.91</u>	**1.159**	**24.35**	<u>1.141</u>	**24.86**	**1.106**	**22.457**	**1.085**
SOMo | Qwen3.5-9B	**17.69**	<u>0.952</u>	**22.82**	<u>1.162</u>	<u>24.42</u>	**1.124**	<u>25.26</u>	<u>1.213</u>	<u>22.547</u>	<u>1.113</u>

CODaAG contextual retrieval. With Qwen3.5-9B for both methods, SOMo improves over DAAAM in all four categories (Table 2). Category-average errors fall from 70.657 to 22.547 m (68.09%) and from 3.119 to 1.113 min (64.32%). Positional reductions are larger for co-observation and proximity (74.5% and 73.6%) than single-object context (51.7%), consistent with the benefit of retrieving related objects jointly.

Additional benchmarks. SOMo also improves OC-NaVQA under full-sequence access and SG3DAG's indoor 3D queries (Tables 4 and 5). On SG3D, SOMo with Qwen3.5-9B achieves 31.82% sub-task accuracy and 16.91% complete-task accuracy, compared with 22.16% and 11.22% for DAAAM with GPT-5-mini (Table 6). Requiring every step to be correct tests consistent grounding across instructions; substantial errors remain.

Table 3: Mean wall-clock answer time (seconds) and number of tool calls per question across benchmarks. All methods use Qwen3.5-9B.

Method	NaVQA		OC-NaVQA		CODaAG	
	Answer time	Tool calls	Answer time	Tool calls	Answer time	Tool calls
ReMEmbR	13.43	3.40	16.61	3.82	28.66	4.97
DAAAM	21.17	4.69	24.12	4.78	31.26	5.62
SOMo	9.21	2.17	9.03	1.92	6.29	1.46

4.2 Agent and backend efficiency

With Qwen3.5-9B, answering time and tool use decrease on all three benchmarks (Table 3); on CODaAG they fall from 31.26 to 6.29 seconds and from 5.62 to 1.46 calls per question. These measurements include reasoning and tool use. Separately, warmed candidate-generation kernels are 7.06–24.54× faster than blocked NumPy scans on CODa and 16.27–193.36× on KITTI-360, with identical candidate results. These ranges exclude model inference, memory construction, and persisted full-history co-observation lookup. Appendix C.3 specifies timing boundaries; Appendix C.4.3 reports individual timings and result sizes. Appendix C.4.4 further examines the benefits and costs of automatically selecting spatial resolution and constraint order per request.

5 Conclusion

We presented SOMo, a scene memory that makes spatial relationships and observation context guide object selection during retrieval. SOMo jointly selects compatible objects and returns their locations with supporting observation evidence. A sparse hierarchy connects object groups to recorded encounters, allowing constraints to eliminate incompatible groups before individual verification. Adaptive planning selects resolution and constraint order using cost estimates and candidate counts, while joint semantic ranking selects among verified combinations. This reduces the evidence and candidate combinations that the agent must assemble through separate searches. Experiments show improved question answering and sequential grounding, fewer tool calls, and lower answering times. Backend measurements demonstrate faster candidate generation with identical results; planning comparisons show both the benefits of adaptation and its selection overhead. These results support joint contextual retrieval for answering questions and grounding instructions in accumulated robot observations.

AI USE STATEMENT

The authors originated the research ideas and intuitions, developed the conceptual framework, and designed the methodology and experiments. AI agents assisted with coding and implementing author-specified methods and experimental procedures, as well as organizing and presenting experimental results. The authors reviewed these outputs, interpreted the findings, and determined the conclusions.

The generated benchmarks were constructed programmatically from ground-truth information in the existing CODa and SG3D datasets; AI was not used to generate their content. No dataset cleaning or reformatting was performed. The use of LLMs as reasoning components within the evaluated systems is described in the methods and appendices.

AI also assisted with drafting, organizing, and editing portions of the manuscript and with presenting qualitative examples and empirical comparisons. The authors assessed these suggestions and checked the resulting claims against the experimental evidence. This work contains no mathematical proofs, and AI was not used for translation. The authors take responsibility for the methods, implementation, analysis, and final manuscript.

REPRODUCIBILITY STATEMENT

We will release the code and the CODaAG and SG3DAG datasets to support reproduction and further study of the proposed approach. The method section and appendices describe the memory representation, retrieval algorithms, and adaptive planning procedure. Appendix C.1 specifies the evaluation datasets, tasks, and scoring procedures, while Appendix C.2 describes the model and memory configurations. Appendix D documents how CODaAG and SG3DAG are constructed from existing ground-truth annotations and lists their question formulations. Runtime measurement boundaries, warmup and repetition procedures, and correctness checks are provided in Appendix C.3, with the automatic-planning workload and controls detailed in Appendix C.3.1.

Our local workstation has an Intel Core i7-13700K CPU, 64 GB of RAM, and an NVIDIA GeForce RTX 4090 GPU with 24 GB of GPU memory.

REFERENCES

Panos Achlioptas, Ahmed Abdelreheem, F. Xia, Mohamed Elhoseiny, and Leonidas J. Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision*, 2020.

Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2838–2845, 2024.

Zhongang Cai, Ruisi Wang, Chenyang Gu, Fanyi Pu, Junxiang Xu, Yubo Wang, Wanqi Yin, Zhitao Yang, Chen Wei, Qingping Sun, Tongxi Zhou, Jiaqi Li, Hui Pang, Oscar Qian, Yu-Rou Wei, Zhiqian Lin, Xuanke Shi, Kewang Deng, Xiaoyang Han, Zukai Chen, Xiangyu Fan, Hanming Deng, Lewei Lu, Liang Pan, Bo Li, Ziwei Liu, Quan Wang, Dahua Lin, and Lei Yang. Scaling spatial intelligence with multimodal foundation models. *ArXiv*, abs/2511.13719, 2025.

Yun Chang, Leonor Fermoselle, Duy Ta, Bernadette Bucher, Luca Carlone, and Jiuguang Wang. Ashita: Automatic scene-grounded hierarchical task analysis. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 29458–29468, 2025.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2829, 2022.

Yi-Nan Deng, Jia-Hui Wang, Jing-Yu Zhao, Xinyu Tian, Guang-Yan Chen, Yi Yang, and Yu-Feng Yue. Opengraph: Open-vocabulary hierarchical 3d graph representation in large-scale outdoor environments. *IEEE Robotics and Automation Letters*, 9:8402–8409, 2024.

Yue Fan, Xiaojian Ma, Rongpeng Su, Jun Guo, Ru-Jie Wu, Xi Chen, and Qing Li. Embodied videoagent: Persistent memory from egocentric videos and embodied sensors enables dynamic scene understanding. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6342–6352, 2024.

Nicolas Gorlo, Lukas Schmid, and Luca Carlone. Describe anything anywhere at any moment. *ArXiv*, abs/2512.00565, 2025.

Qiao Gu, Ali Kuwajerwala, Sacha Morin, Krishna Murthy Jatavallabhula, Bipasha Sen, Aditya Agarwal, Corban Rivera, William Paul, Kirsty Ellis, Ramalingam Chellappa, Chuang Gan, Celso M. de Melo, Joshua B. Tenenbaum, Antonio Torralba, Florian Shkurti, and Liam Paull. Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5021–5028, 2023.

Yi-Xuan Hu, Zhicheng Zheng, Li-Han Zha, Chunwei Xing, Rajdeep Singh, Omar Hossain, Antonio Loquercio, and Dhruv Shah. Raven: Long-horizon reasoning & navigation with a visuo-spatio-temporal memory. *ArXiv*, abs/2606.25206, 2026.

Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *Robotics: Science and Systems XVIII*, 2022.

Long Lian, Yifan Ding, Yunhao Ge, Sifei Liu, Hanzi Mao, Boyi Li, Marco Pavone, Ming-Yu Liu, Trevor Darrell, Adam Yala, and Yin Cui. Describe anything: Detailed localized image and video captioning. *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 21766–21777, 2025.

Dominic Maggio, Yun Chang, Nathan Hughes, Matthew Trang, Daniel Griffith, Carlyn Dougherty, Eric Cristofalo, Lukas Schmid, and Luca Carlone. Clio: Real-time task-driven open-set 3d scene graphs. *IEEE Robotics and Automation Letters*, 9:8921–8928, 2024.

Jianmo Ni, Gustavo Hernández Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Matthew Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *ArXiv*, abs/2108.08877, 2021.

Aleeza Rasheed, Maria Kabtoul Automatika, and Robotics Inria. emem: A hybrid spatio-temporal memory system for embodied agents. *ArXiv*, abs/2606.03374, 2026.

Saumya Saxena, Blake Buchanan, Chris Paxton, Peiqi Liu, Bingqing Chen, Narunas Vaskevicius, Luigi Palmieri, Jonathan Francis, and Oliver Kroemer. Grapheqa: Using 3d semantic scene graphs for real-time embodied question answering. *ArXiv*, abs/2412.14480, 2024.

Nur Muhammad (Mahi) Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. *ArXiv*, abs/2210.05663, 2022.

Tin Stribor Sohn, Maximilian Dillitzer, Jason J. Corso, and Eric Sax. R4: Retrieval-augmented reasoning for vision-language models in 4d spatio-temporal space. *ArXiv*, abs/2512.15940, 2025.

Abdelrhman Werby, Chen Huang, Martin Büchner, Abhinav Valada, and Wolfram Burgard. Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. *ArXiv*, abs/2403.17846, 2024.

Jing Wu, Jianhua Wu, Jiayi Guan, Jiahong Chen, Jinghui Lu, Hangjun Ye, Bingzhao Gao, and Long Chen. Spatiolm: Towards general physical spatial intelligence in vision-language models, 2026. URL https://arxiv.org/abs/2608.01899.

Quan-Ting Xie, So Yeon Min, Peng-Liang Ji, Yue Yang, Tianyi Zhang, Ke-Di Xu, Aarav Bajaj, Ruslan Salakhutdinov, Matthew Johnson-Roberson, and Yonatan Bisk. Embodied-rag: General non-parametric embodied memory for retrieval and generation. *ArXiv*, abs/2409.18313, 2024.

Kashu Yamazaki, Taisei Hanyu, Khoa Vo, Thang Pham, Minh-Triet Tran, Gianfranco Doretto, Anh Nguyen, and Ngan Le. Open-fusion: Real-time open-vocabulary 3d mapping and queryable scene representation. *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9411–9417, 2023.

Yibin Yan, Jilan Xu, Shangzhe Di, Haoning Wu, and Weidi Xie. Omnistream: Mastering perception, reconstruction and action in continuous streams. *ArXiv*, abs/2603.12265, 2026.

Rui Yang, Ziyu Zhu, Yanwei Li, Jingjia Huang, Shen Yan, Siyuan Zhou, Zhe Liu, Xiangtai Li, Shuangye Li, Wenqi Wang, Yi Lin, and Hengshuang Zhao. Visual spatial tuning. *ArXiv*, abs/2511.05491, 2025.

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. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17294–17303, 2024.

Mingfeng Yuan, Hao Zhang, Mahan Mohammadi, Runhao Li, Jinjun Shan, and Steven L. Waslander. Star: Scalable task-conditioned retrieval for long-horizon multi-modal robot memory. *IEEE Robotics and Automation Letters*, 11:5994–6001, 2026.

Arthur Zhang, Chaitanya Eranki, Christina Zhang, Ji-Hwan Park, Raymond Hong, Pranav Kalyani, Lochana Kalyanaraman, Arsh Gamare, Arnav Bagad, Maria Esteva, and Joydeep Biswas. Toward robust robot 3-d perception in urban environments: The ut campus object dataset. *IEEE Transactions on Robotics*, 40:3322–3340, 2023.

Zhuofan Zhang, Ziyu Zhu, Junhao Li, Pengxiang Li, Tianxu Wang, Tengyu Liu, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Siyuan Huang, and Qing Li. Task-oriented sequential grounding and navigation in 3d scenes. *arXiv preprint arXiv:2408.04034*, 2024.

Xu Zhao, Wen-Yan Ding, Yongqi An, Yinglong Du, Tao Yu, Min Li, Ming Tang, and Jinqiao Wang. Fast segment anything. *ArXiv*, abs/2306.12156, 2023.

Xinyi Zheng, Yunze Liu, Chi-Hao Wu, Fan Zhang, Hao Zheng, Wenqi Zhou, Walterio W. Mayol-Cuevas, and Junxiao Shen. Spatialmem: Metric-aligned long-horizon video memory for language grounding and qa, 2026. URL https://arxiv.org/abs/2601.14895.

A Appendix Overview

The appendices provide a visualization of the spatial and observation memory (Appendix B), experimental settings and results (Appendix C), the CODaAG and SG3DAG benchmarks (Appendix D), the retrieval interface (Appendix E), memory construction and algorithm details (Appendix F), and five qualitative retrieval examples: contextual statue retrieval (Appendix G), a yellow police pole (Appendix H), a bike–pole encounter (Appendix I), a newspaper dispenser relative to a past robot position (Appendix J), and car–truck observation timing (Appendix K).

B Visualization of the Spatial and Observation Memory

Spatial organization across resolutions. Figure 3 illustrates how Morton (Z-order) codes organize space using integer cell identifiers. At level ℓ, coordinates are quantized into cells of width $h_\ell = 2^\ell h_0$; interleaving their coordinate bits produces a cell's integer code. For index dimension $m \in \{2, 3\}$, removing the lowest m bits yields the parent code at level $\ell + 1$, so shared prefixes identify coarser spatial groups. Cell codes can be computed directly from coordinates without instantiating the full grid: only occupied cells and their ancestors require storage. Using the method's notation, $g_\ell(i) = a$ assigns object i to cell a. An inverted lookup maps each occupied cell code to its member set, $\mathcal{O}_\ell(a) = \{i : g_\ell(i) = a\}$, stored as a contiguous span of a shared Morton-ordered array of object indices. Retrieval can therefore look up the members of selected cells without scanning every object. The levels provide alternative groupings over shared records, preserving individual positions and observation evidence for exact verification.

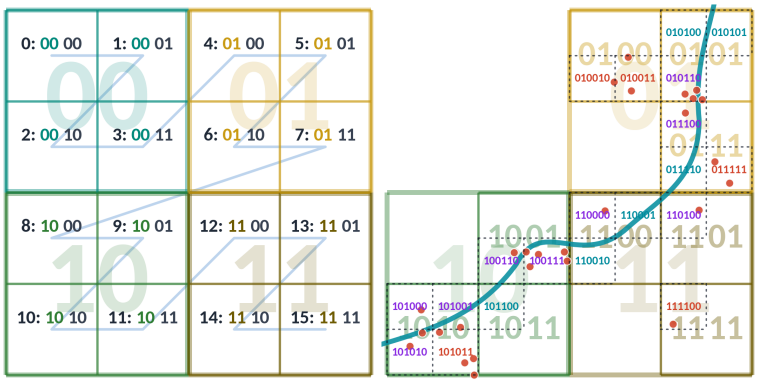


Figure 3: **Spatial cells at multiple resolution levels.** Morton codes organize occupied cells and their coarser groupings. Each level provides a partition over shared object records, allowing retrieval to choose a grouping suited to the request.

Objects, trajectories, and observation associations. Figure 4 shows this organization for sequence 0, used in NaVQA and OC-NaVQA. The top row displays object and trajectory cells at widths of 2, 4, and 8 m, the first three of six resolution levels. The enlarged views below visualize visit-to-object-cell associations: for a visit v, the links $I_\ell(v) = \{g_\ell(i) : i \in R_v\}$ identify cells containing objects observed during that visit. Because visits are associated with trajectory cells, these links explicitly connect regions traversed by the robot to the object cells observed from them. Coarser levels aggregate these spatial connections, while the underlying records preserve distinct visits to the same location and individual observation intervals. This organization lets retrieval use a robot observation area or time window to select relevant visits and follow their links to candidate object groups. Conversely, object-to-visit lookup recovers the encounters supporting a selected object. Spatial relationships and observation context can therefore narrow candidates together before individual objects or pairs are expanded, reducing both backend search work and the evidence the agent must assemble across separate tool calls. Stored observation intervals provide the subsequent temporal-overlap checks required for co-observation.

Together, these views illustrate how spatial grouping and observation associations allow retrieval to narrow candidate objects and encounters before evaluating individual matches.

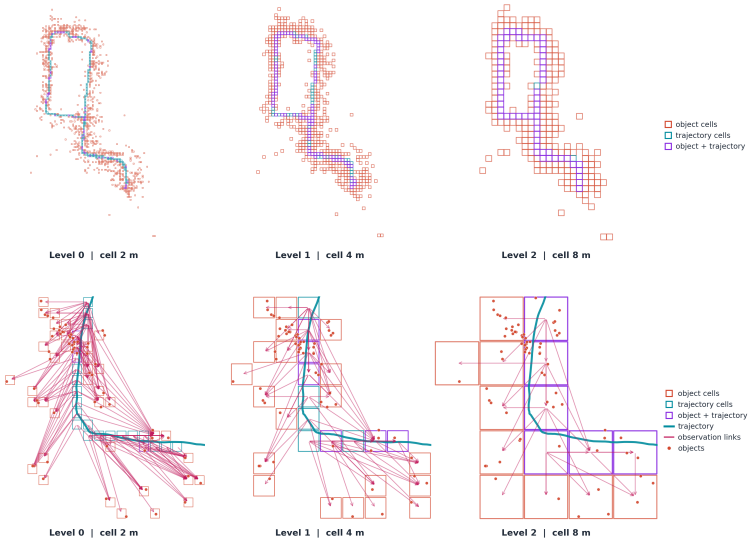


Figure 4: **Spatial extent of the object and observation memory.** Top: occupied object and trajectory cells at three resolutions. Bottom: a local view of spatially aggregated observation associations. These links summarize the memory spatially; the underlying records additionally retain distinct chronological visits and individual observation intervals, as described in Section 3.2. The visualization shows SOMo's Z-order representation of sequence 0 (seq0), used in NaVQA and OC-NaVQA, with the first three of six resolution levels displayed.

C Experimental Settings and Results

C.1 Datasets and evaluation tasks

NaVQA and OC-NaVQA. Both benchmarks use long-horizon CODa exploration sequences, each spanning more than ten minutes. Each benchmark contains 210 questions covering positions, event times or durations, and descriptions. NaVQA position questions use robot-position ground truth, whereas OC-NaVQA, introduced by DAAAM, uses object-position ground truth. We report descriptive-question accuracy, positional error in metres, and temporal error in minutes. NaVQA additionally separates short, medium, and long observation contexts; OC-NaVQA is evaluated over the full sequence without context-window restriction.

Generated contextual benchmarks. CODaAG and SG3DAG evaluate contextual retrieval in CODa's outdoor scenes and SG3D's indoor scenes using shared question formulations. Appendix D describes their construction, query categories, answer types, and formulations.

Sequential grounding on SG3D. The supported HM3D subset of the released SG3D test annotations contains 205 tasks and 880 action steps in 76 subscenes across eight parent scenes. We follow the SG3D/ASHiTA protocol and report sub-task accuracy (s-acc) and task accuracy (t-acc). Task accuracy requires every subtask of a multi-step instruction to be grounded correctly. Instructions can identify objects through contextual relations, for example, "Go to the bathtub near the toilet" or "Sit on the office chair beside the desk."

Scoring and aggregation. Position error is Euclidean distance between predicted and reference 3D coordinates. Time error is absolute deviation from the reference event time or duration, reported in minutes. Descriptive accuracy is the fraction of answers judged correct against the references. NaVQA contains 100 descriptive, 71 positional, and 39 temporal questions; overall values aggregate the applicable question type across context lengths. For generated questions asking for two positions, the per-question error is the mean of the two role-ordered position errors. CODaAG and SG3DAG report position and time separately, and their category-average columns weight the four categories equally. The generated evaluator averages errors over successfully parsed and graded answers,

recording unanswered, parse-failed, and runtime-failed items separately as answer-coverage counts. For SG3D, an incorrect or missing top-ranked target counts as an incorrect step; a task is correct only if all its steps are correct. SOMo entries with error bars report means and standard deviations across three seeded runs for Qwen3.5-9B or three repeated API calls for GPT-5-mini. Entries without error bars retain their source reporting format.

C.2 Memory construction and comparison methods

Scene memory is constructed progressively from observations using DAAAM's object-centric mapping pipeline (Gorlo et al., 2025). SOMo's upstream components use FastSAM (Zhao et al., 2023) for mask proposals, DAM-3B (Lian et al., 2025) for object descriptions, OpenCLIP ViT-L/14 (Cherti et al., 2022) for object-crop image embeddings, and Sentence-T5-xl (Ni et al., 2021) for sentence embeddings.

Question-answering comparisons include ReMEmbR, Concept Graphs, and DAAAM (Anwar et al., 2024; Gu et al., 2023; Gorlo et al., 2025), together with spatial foundation-model baselines (Cai et al., 2025; Wu et al., 2026) as listed in the full tables. Sequential-grounding comparisons include Hydra, HOV-SG, ASHiTA, and DAAAM (Hughes et al., 2022; Werby et al., 2024; Chang et al., 2025; Gorlo et al., 2025). Baseline model configurations are recorded in each table. Our primary SOMo results use Qwen3.5-9B, served through vLLM using FP8 quantization; we also evaluate GPT-5-mini. The NaVQA and CODaAG comparisons with DAAAM hold Qwen3.5-9B fixed. On OC-NaVQA and SG3D, the reported DAAAM baselines use GPT-5-mini; comparisons with SOMo using Qwen3.5-9B therefore differ in reasoning model. The NaVQA table distinguishes newly evaluated baselines from results reported by DAAAM, and the SG3D baseline results are taken from DAAAM and ASHiTA. These are system comparisons: a common reasoning model does not by itself imply identical perception, memory representations, or tools.

Generated-question inputs. The shared DAAAM/SOMo generated-question wrapper supplies the question, observation window, answer format, and an explicit reference time only when that time is given in the question. Expected answers and generation provenance are withheld. The wrapper also supplies family-specific guidance naming `search_objects`, the relation, and relevant output fields, including when the legacy interface is selected. Consequently, these inputs guide SOMo's API use; they do not constitute an unaided comparison of tool selection. The qualitative generated examples document this guidance in their individual traces.

Observation-history access. The CODa question-answering runs query memory constructed from the complete sequence. For NaVQA, temporal restrictions are applied through the agent's tool arguments; memory is not separately truncated to each question's annotated observation window. OC-NaVQA uses the full-sequence protocol.

End-to-end backend configurations. For the CODa-based task evaluations, SOMo uses a two-dimensional XY Morton (Z-order) index with six hierarchy levels and cell widths of 2, 4, 8, 16, 32, and 64 m. The SG3D evaluations use a three-dimensional XYZ index with four levels and cell widths of 2, 4, 8, and 16 m. In both configurations, the spatial bounds encompass all stored object and trajectory positions, with a 5 m margin on each side. The artifacts and hierarchy settings used for the backend microbenchmarks are specified separately in Appendix C.3.

C.3 Runtime measurement protocols

Agent answering time. Table 3 reports per-question wall-clock time and executed tool-call counts with Qwen3.5-9B. The question timer includes agent inference, tool execution, and answer finalization; it is distinct from the backend-only measurements below. A reduction in answering time therefore measures the complete agent workflow and cannot be attributed solely to faster candidate generation.

CODa candidate generation. The retained CODa seq_00 artifact contains 1,501 objects and 1,154 robot trajectory samples, with six hierarchy levels and a 2 m finest cell. We use one CPU thread and report median wall-clock latency per exact candidate-kernel invocation over ten repeats after

three warm-up calls, with diagnostics disabled. Blocked NumPy-vectorized brute force provides the baseline. Comparisons require identical candidate identities and, for bearing queries, identical winners. The tested settings include pair proximity, restricted-window co-observation, and robot-relative bearing.

KITTI-360 candidate generation. Drive 0009 provides 3,556 static train objects and 13,247 camera-pose samples. Real `image_00` 2D instance labels supply observation support for 848 objects. The retained artifact has eight hierarchy levels and a 5 m finest cell. We report median CPU latency over ten repeats after three warm-up calls against blocked NumPy-vectorized scans with identical result sets. Timing excludes parsing, PNG loading, support extraction, artifact construction, backend loading, and embedding generation.

C.3.1 AUTOMATIC-PLANNING PROTOCOL

The planning study freezes the production backend, input artifacts, request grid, and policy definitions before timing. It contains 128 pair requests on the full CODa and KITTI-360 artifacts and 300 requests across 12 controlled scenes. Real-scene requests vary radius (1, 8, or 64 m), history fraction (0.5%, 5%, 50%, or all), and temporal position (15%, 50%, or 85% of the trajectory), with full-history cases included once. Additional controls use proximity alone, co-observation alone, or selected canonical visits. Sixty real-scene combined requests also measure complete two-object retrieval with frozen text features and top-five output.

Eight controlled scenes cross uniform or clustered geometry, local or permuted object–visit membership, and one or eight observation passes, using 1,024 objects. Four further scenes vary size to 256 or 2,048 objects. All use a 128 m square, 32 observation groups, 2 m finest cells, and seven levels. Permutation changes the spatial arrangement of observation memberships while preserving their sizes; repeated passes preserve geometry and memberships while increasing history fragmentation. This controlled grid tests mechanisms rather than their frequency in deployed workloads.

Comparators restrict constraint order while retaining adaptive resolution, fix resolution while retaining adaptive order, fix both, or disable the optional challenger while retaining the primary assessment and safety fallback. Fixed resolutions include singleton, finest, middle, and coarsest partitions. All policies share exact predicates, output materialization, and a 768 MiB temporary-memory budget.

Measurements run serially on one pinned CPU with numerical-library thread pools limited to one and artifacts resident in memory. Each request uses one warmup and five measurements in seeded rotating policy order; we report its median. Pair timing includes normalization, planning, assessment, refinement, and materialized arrays. Complete-retrieval timing additionally includes ranking and serialization. Loading, model inference, correctness checks, and diagnostic recording are excluded. A 10 s per-call timeout applies to every policy. Successful policies must return identical results; automatic pair outputs are also checked against an independent NumPy distance and interval-intersection oracle, including the common-visit condition where required.

C.4 DETAILED EXPERIMENT RESULTS

The following tables supplement the main-text NaVQA and CODaAG results with additional benchmark comparisons and kernel timings.

C.4.1 SPATIOTEMPORAL QUESTION ANSWERING

The NaVQA and CODaAG comparisons and their discussion appear in the main text (Tables 1 and 2).

OC-NaVQA. Table 4 reports retrieval over the full sequence. SOMo with Qwen3.5-9B achieves 31.53 m positional error, 1.362 min temporal error, and 0.734 descriptive accuracy, compared with 41.75 m, 1.792 min, and 0.711 for DAAAM with GPT-5-mini. These results demonstrate gains without annotated context-window filtering, with different reasoning models in this comparison.

SG3DAG. Table 5 extends the category comparison to indoor scenes. SOMo has lower positional and temporal error than ReMEmbR and DAAAM in every displayed query category. The category-average columns report the unweighted arithmetic means across the four categories.

Table 4: Results on OC-NaVQA. ReMEmbR, Concept Graphs, and DAAAM use GPT-5-mini for reasoning; Spatio-LM uses its native model. SOMo's reasoning model is indicated in each row.

Method	Question Accuracy ↑	Positional Error [m] ↓	Temporal Error [min] ↓
ReMEmbR - NVILA-Lite-2B	0.432	53.466	2.287
ReMEmbR - NVILA-Lite-8B	0.463	55.894	4.106
Concept-Graphs	0.299	111.29	×
Spatio-LM	0.539	71.52	2.624
DAAAM | GPT-5-mini	0.711	41.75	1.792
SOMo | GPT-5-mini	**0.752** ± 0.123	**30.89** ± 3.425	**1.318** ± 0.186
SOMo | Qwen3.5-9B	<u>0.734</u> ± 0.137	<u>31.53</u> ± 3.468	<u>1.362</u> ± 0.189

Table 5: Results on SG3DAG benchmark. We report positional and temporal error across single-object, multi-object co-observation, multi-object in proximity, and robot-relative angle query settings. Pos. denotes positional error in meters, and Temp. denotes temporal error in minutes; lower is better for both. Category Average is the unweighted arithmetic mean across the four categories, reflecting the overall performance across questions that focus on different query scenarios.

Method	Single-Object		Multi-Object Co-Observation		Multi-Object in Proximity		Robot-Relative Angle		Category Average	
	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓	Pos. ↓	Temp. ↓
ReMEmbR | Qwen3.5-9B	12.59	3.142	15.29	3.952	18.69	3.191	17.24	3.810	15.953	3.524
DAAAM | Qwen3.5-9B	9.24	1.623	10.62	2.795	11.90	2.994	12.59	2.744	11.088	2.539
SOMo | Qwen3.5-9B	**5.63**	**0.748**	**7.83**	**0.855**	**6.31**	**1.124**	**7.98**	**0.913**	**6.938**	**0.910**

C.4.2 Sequential task grounding

Table 6 gives the full SG3D comparison. SOMo with Qwen3.5-9B achieves 31.82% sub-task accuracy and 16.91% complete-task accuracy, compared with 22.16% and 11.22% for DAAAM with GPT-5-mini. The table also reports SOMo with GPT-5-mini, which achieves 32.23% and 17.56%, respectively. These tasks test whether contextual retrieval identifies the objects needed at successive instruction steps.

C.4.3 Answering time and backend microbenchmarks

End-to-end answering time. Table 3 in the main text gives the complete three-benchmark comparison. With Qwen3.5-9B, average answering times decrease from DAAAM's 21.17 to 9.21 seconds on NaVQA, 24.12 to 9.03 seconds on OC-NaVQA, and 31.26 to 6.29 seconds on CODaAG. These measure the combined agent and memory workflow rather than attributing all savings to candidate generation.

CODa candidate kernels. Table 7 reports exact-result comparisons under the protocol in Appendix C.3. Proximity speedups range from 7.06 to 12.09×, co-observation from 7.88 to 24.54× across restricted windows, and bearing from 12.06 to 12.14×. The table reports result-set sizes alongside latency so that query selectivity remains visible.

KITTI-360 candidate kernels. Table 8 reports speedups of 16.27–45.23× for proximity, 122.87–193.36× for restricted-window co-observation, and 71.76–73.64× for bearing. The larger speedups than on CODa show the benefit on this larger artifact, although the scenes also differ in observation support and candidate populations.

C.4.4 Automatic-planning controls

We aggregate matched per-request ratios of comparator latency to automatic latency using their geometric mean; a ratio above one favors automatic planning. Ratios use requests completed by both policies, with failures retained separately in the study's coverage records. All 428 automatic pair

Table 6: Sequential task grounding results on SG3D. We report sub-task accuracy (s-acc) and task accuracy (t-acc), where task accuracy requires all sub-tasks in an instruction to be grounded correctly. Baseline results are taken from DAAAM and ASHiTA.

Method	s-acc [%]	t-acc [%]
Hydra + GPT-5-mini	8.18	2.44
Hydra (GT Seg) + GPT-5-mini	14.2	6.34
HOV-SG	8.98	1.95
ASHiTA	21.7	8.78
DAAAM + GPT-5-mini	22.16	11.22
SOMo | GPT-5-mini	**32.23** $\pm$ 0.35	**17.56** $\pm$ 0.49
SOMo | Qwen3.5-9B	<u>31.82</u> $\pm$ 0.57	<u>16.91</u> $\pm$ 0.28

Table 7: CODa seq_00 candidate-generation microbenchmarks.

Query family	Setting	z-order [ms/call]	NumPy-vec baseline [ms/call]	Speed-up vs. NumPy-vec	Result-set size
Co-observation	20 s window	**0.191**	4.545	**23.85**×	1,498 pairs
Co-observation	60 s window	**0.584**	4.607	**7.88**×	6,037 pairs
Co-observation	120 s window	**0.935**	22.935	**24.54**×	9,174 pairs
Pair proximity	2 m	**1.600**	16.152	**10.09**×	2,857 pairs
Pair proximity	3 m	**1.323**	16.002	**12.09**×	5,376 pairs
Pair proximity	5 m	**2.290**	16.169	**7.06**×	12,082 pairs
Robot-relative bearing	$\pm 15^\circ$	**4.209**	50.769	**12.06**×	69 objects
Robot-relative bearing	$\pm 45^\circ$	**4.208**	51.088	**12.14**×	215 objects

outputs pass the independent oracle, and successful complete-retrieval policies produce identical serialized outputs. Table 9 shows two illustrative crossover cases from the scene with 1,024 uniformly distributed objects, permuted object–visit membership, and eight passes. These cases were selected after inspecting the full results; the aggregate comparisons below use the complete applicable panels.

With a 1 m radius and full observation history, distance-first execution quickly limits the candidate pairs. With a 64 m radius and only 0.5% of the history, observation-first execution instead restricts the candidates before distance evaluation. Automatic planning selects the favorable order in both cases, avoiding the much higher cost of the other order. It remains slightly slower than the appropriate fixed-order policy because selecting a plan also takes time. Both fixed-order comparators retain adaptive resolution, so this comparison concerns the choice of constraint order.

Across 216 controlled combined requests, automatic planning is 1.31× faster than always distance-first and 1.14× faster than always observation-first execution, with adaptive resolution retained in both comparators. Across 102 real combined requests, the corresponding ratios are 1.43× and 0.80×. The latter is an unfavorable case for automatic selection: observation-first already suits much of this panel, and selection overhead exceeds the savings on other requests.

Proximity-only controls isolate resolution selection, yielding 1.31× and 1.32× speedups over the finest level on real and controlled scenes, respectively. The optional challenger provides little aggregate benefit. These controls support adapting resolution and order to workload structure, without establishing universal gains from every planning component. Larger speedups on KITTI-360 than CODa in the separate kernel study likewise do not isolate scene size: object distribution and observation support differ between the two artifacts.

D CODAAG AND SG3DAG BENCHMARKS

CODaAG (CODa-Autogen) is constructed from CODa's ground-truth object annotations, mapped locations, robot trajectories, and observation timestamps (Zhang et al., 2023). SG3DAG applies the same question formulations to SG3D's indoor scenes (Zhang et al., 2024), using scene-specific objects, locations, and observation context. The 13 question types in Table 10 span four categories: single-object context, multi-object co-observation, multi-object proximity, and robot-relative bearing.

Table 8: KITTI-360 drive 0009 candidate-generation microbenchmarks on static train objects.

Query family	Setting	z-order [ms/call]	NumPy-vec baseline [ms/call]	Speed-up vs. NumPy-vec	Result-set size
Co-observation	20 s window	**0.701**	104.462	**149.02**×	198 pairs
Co-observation	60 s window	**1.175**	144.414	**122.87**×	479 pairs
Co-observation	120 s window	**1.818**	351.562	**193.36**×	921 pairs
Pair proximity	2 m	**1.390**	62.865	**45.23**×	1,997 pairs
Pair proximity	3 m	**1.416**	62.751	**44.31**×	2,609 pairs
Pair proximity	5 m	**2.581**	64.026	**24.80**×	3,991 pairs
Pair proximity	10 m	**3.889**	63.274	**16.27**×	10,391 pairs
Robot-relative bearing	left, $\pm 15^\circ$	**20.438**	1,466.625	**71.76**×	222 objects
Robot-relative bearing	left, $\pm 45^\circ$	**19.960**	1,469.855	**73.64**×	271 objects

Table 9: Constraint-order crossover: query time (ms) in a controlled scene. History is the queried fraction of the observation history. Both fixed-order policies retain adaptive resolution. Cases were selected after inspecting the results.

Radius [m]	History [%]	Automatic	Distance first	Observation first
1	100	4.90	4.45	49.53
64	0.5	4.88	88.14	4.50

D.1 Source objects and observation records

Generation starts from a catalog of annotated objects and a synchronized sequence of camera observations and poses. Each object supplies a category name, a world-frame position, and the frames in which it is observed. Each frame supplies a timestamp, observer position, and heading. These records provide both the ingredients of the question and its numerical answer.

For CODaAG, object locations come from CODa's 3D annotations, and robot positions and headings come from its trajectory. Camera projection checks retain observations with sufficient image coverage, including a projected footprint of at least 1,250 pixels. For SG3DAG, we use the annotated objects in SG3D subscenes and their rendered HM3D camera traversals. Semantic masks provide observation support: a supporting frame contains at least 4,096 pixels of the object, and a retained observation episode spans at least 2 s with gaps no larger than 0.21 s. Object positions are computed by back-projecting the object's depth pixels into the world frame and aggregating them with coordinate-wise medians within and across frames. The synchronized camera trajectory supplies the observer positions and headings; SG3DAG questions therefore refer to the camera where CODaAG questions refer to the robot.

D.2 Constructing questions and ground-truth answers

We first construct candidate situations from these records, then insert the object categories, spatial parameters, and applicable time cues into fixed question templates. Every question carries an observation window. CODaAG samples windows of 25–60 s around supporting observations; proximity windows contain observations of both participating objects. SG3DAG uses 40 s windows, adjusted at traversal boundaries, and scans proximity windows at a 10 s stride. Spatial and temporal answers are computed directly from the selected situation.

Single-object context (four types). For a nearest-object question, we select an observation time and object category, find the nearest observed object of that category to the observer at that time, and use its 3D position as the answer. For a location-constrained question, we choose an observed object and a nearby query location and radius containing it; the answer is again the object's position. The other two types start from an object's observation support and ask where the observer was or when the observation occurred. Their reference event is the supporting frame nearest the mean support timestamp, with the corresponding trajectory position or timestamp as the answer.

Multi-object co-observation (three types). We enumerate pairs of object categories observed in the same camera frames and collect their common observation support into episodes. CODaAG

requires at least two supporting frames, grouping frames separated by at most 0.3 s; SG3DAG uses the 2 s episode requirement above. The reference event is a recorded frame nearest the mean timestamp of the selected episode's supporting frames. One template asks for both object positions, another for the observer's position at that event, and the third for its time. The pair-position answer preserves the order in which the two categories appear in the question.

Multi-object proximity (four types). We select two observed objects whose mapped separation is within a chosen radius. Their observations must fall within the question's window, but need not occur in the same frame. CODaAG uses horizontal XY distance and radii of 5–25 m; SG3DAG uses full 3D Euclidean distance and radii from $\{1, 2, 3, 5\}$ m. These conventions also apply to the location-constrained single-object questions. Two templates ask for both object positions or the position of a target object near an anchor. The remaining two ask where or when the observer came closest to the pair. We compute the pair's midpoint and select the trajectory sample within the window with minimum distance to it, using XY distance for CODaAG and 3D distance for SG3DAG. That sample provides the reference position and time.

Robot-relative bearing (two types). At an object's observation frames, we compute the signed world-XY angle from the observer's heading to the object. We retain absolute bearings between $15°$ and $165°$, round the requested angle to a $5°$ increment, and require agreement within $3°$. The sign determines whether the question says left or right. The two templates ask for the observer's position or the time at the selected bearing event, using the associated trajectory sample as ground truth.

D.3 Sampling and question formulations

We sample valid situations with a fixed random seed, favoring coverage of different object categories and category pairs. Related templates reuse selected situations: for example, a co-observation episode supplies the pair-location, observer-location, and event-time questions. Thus, question counts include distinct requests about shared underlying events. For each benchmark, we generate three questions per type per parent scene, yielding 39 questions per scene. Five CODa sequences and eight HM3D parent traversals contribute 195 CODaAG and 312 SG3DAG questions, respectively; SG3DAG situations are constructed within their constituent SG3D subscenes.

Table 10 lists all 13 formulations with CODaAG examples. SG3DAG substitutes camera references and 3D distance conditions where applicable. Both benchmarks report positional and temporal error separately for each category. Appendices E.2 and F.4.6 describe the stored evidence and geometric predicates used at retrieval time.

Table 10: Shared question formulations for CODaAG and SG3DAG. Each row corresponds to one generated question type; examples are drawn from CODaAG.

Category	Answer requested	Template	Example selected question
Single-Object	Object location	At {time} seconds, what is the 3D position of the {object} nearest to the robot?	At 133.1 seconds, what is the 3D position of the traffic sign nearest to the robot?
Single-Object	Object location	What is the 3D position of the {object} whose mapped XY position is within {radius} meters of location {location}?	What is the 3D position of the bike whose mapped XY position is within 24.5 meters of location (-40.04, 315.77, 0.43)?
Single-Object	Robot location	Where was the robot while observing the {object}?	Where was the robot while observing the newspaper dispenser?
Single-Object	Observation time	Around what time was the robot observing the {object}?	Around what time was the robot observing the table?
Multi-Object Co-Observation	Object-pair locations	What are the 3D positions of the {object A} and {object B} observed together in the same frame?	What are the 3D positions of the bench and floor sign observed together in the same frame?

Table 10 (continued)

Category	Answer requested	Template	Example selected question
Multi-Object Co-Observation	Robot location	Where was the robot when the {object A} and {object B} were observed together in the same frame?	Where was the robot when the bike and railing were observed together in the same frame?
Multi-Object Co-Observation	Observation time	Around what time were the {object A} and {object B} observed together in the same frame?	Around what time were the bench and floor sign observed together in the same frame?
Multi-Object in Proximity	Object-pair locations	What are the 3D positions of the {object A} and {object B} whose mapped XY positions are within {radius} meters of each other?	What are the 3D positions of the fire hydrant and bench whose mapped XY positions are within 18 meters of each other?
Multi-Object in Proximity	Object location	What is the 3D position of the {object B} that is within {radius} meters of the {object A}?	What is the 3D position of the freestanding plant that is within 18 meters of the railing?
Multi-Object in Proximity	Robot location	Where was the robot when it was closest to the nearby {object A} and {object B} pair within {radius} meters?	Where was the robot when it was closest to the nearby fire hydrant and bench pair within 18 meters?
Multi-Object in Proximity	Closest-approach time	Around what time was the robot closest to the nearby {object A} and {object B} pair within {radius} meters?	Around what time was the robot closest to the nearby railing and freestanding plant pair within 18 meters?
Robot-Relative Angle	Robot location	Where was the robot when the {object} was about {angle} degrees to its {side} in robot-relative world XY bearing?	Where was the robot when the freestanding plant was about 40 degrees to its right in robot-relative world XY bearing?
Robot-Relative Angle	Bearing-event time	Around what time was the {object} about {angle} degrees to the robot's {side} in robot-relative world XY bearing?	Around what time was the bench about 30 degrees to the robot's left in robot-relative world XY bearing?

E Method Detail - Request Semantics and Interface

E.1 Descriptions, relations, and restrictions

A request to `search_objects` specifies an ordered list of descriptions $q_1, \ldots, q_k$, a relation, and applicable spatial and observation conditions. The public interface supports $1 \leq k \leq 4$ roles and returns up to the requested number K of results. Each result assigns a distinct object to every role, including when two roles use the same description. Table 11 summarizes the relations. Single-object requests also support robot-relative bearing and ranking modes that combine semantic relevance with proximity to a specified location or robot pose.

The corresponding relation values are `individual`, `near_each_other`, `observed_together`, and `observed_together_and_near`. For example, the following request specifies two semantic roles, their proximity, and a common observation period:

```
search_objects(
    descriptions=["bike", "green bench"],
    relation="observed_together_and_near",
    max_distance_between_objects=5.0,
    start_time=60.0,
    end_time=100.0,
    return_robot_observation_info=True
)
```

Table 11: Object-search relations. Individual restrictions define the eligible objects; relational conditions determine their compatible combinations.

Relation	Roles	Required evidence
Individual	1	An eligible object; bearing mode additionally requires a qualifying recorded encounter.
Proximity	2–4	Every unordered member pair satisfies the distance threshold.
Co-observation	2–4	The members share observation support in the applicable evidence domain.
Proximity and co-observation	2–4	Both all-pairs proximity and common observation support.

Spatial restrictions identify either an object's mapped search area or the robot's observation area. The former uses `search_center` and `search_radius`; the latter uses `robot_observation_center` and `robot_observation_radius`, or resolves its center from `robot_observation_center_time`. Temporal restrictions use `start_time` and `end_time`. Bearing requests specify a side, angle, angular tolerance, and maximum range, with an optional robot reference time. Normalization establishes a consistent request from these inputs before candidate generation. Direct programmatic search functions normalize into the same execution paths. The default agent interface also exposes trajectory and scene-region tools for complementary context.

E.2 Eligible objects and pair conditions

Let A denote the eligible object indices after individual restrictions. An external identity restriction and a mapped search area constrain this set directly. For single-object or proximity-only retrieval with an observation window W, eligibility requires $H_i \cap W \neq \emptyset$ for each member. If the request omits a window, W covers the available history. Write $\delta(p_i, p_j)$ for the configured XY or XYZ distance. For eligible indices $i < j$, define

$$D(i,j) \iff \delta(p_i, p_j) \leq d, \tag{5}$$

$$T(i,j) \iff H_i \cap H_j \cap W \neq \emptyset. \tag{6}$$

Proximity requests require D, co-observation requests require T, and combined requests require both. These predicates operate on the same candidate relation. Temporal comparisons include shared interval endpoints.

A robot observation-area restriction first selects recorded trajectory samples in the requested region and window. Their canonical visits form $\mathcal{V}$, and objects associated with those visits become eligible. Temporal pairs additionally require

$$J_{\mathcal{V}}(i,j) \iff \exists v \in \mathcal{V} : \; i, j \in R_v. \tag{7}$$

Thus the selected-visit witness and interval-overlap condition establish complementary parts of the request: the relevant encounter association and common stored temporal support.

For group support and returned evidence under an observation-area restriction, define the selected-visit support

$$H_i^{\mathcal{V}} = \bigcup_{\substack{v \in \mathcal{V} \\ i \in R_v}} \big(\mathrm{span}(v) \cap W\big). \tag{8}$$

The backend obtains common area-restricted evidence from selected visits supported by all members and their clipped time extents. Ordinary requests use intersections of the stored intervals H_i within W. This fixes the evidence granularity for area-restricted retrieval: a qualifying in-region sample selects a visit, whose associated objects and recorded extent supply the observation context. Pair verification applies T and the required $J_{\mathcal{V}}$, while whole-combination support and output construction use the applicable support domain.

E.3 Returned information

Results retain role-ordered object identities, mapped positions, semantic scores, and relation evidence. Requested robot information adds an observation event or trajectory context appropriate to the relation. An object's mapped location and the robot's observation location remain separate quantities.

Appendix F.6.5 describes event and pose selection. The trajectory tool supplies recorded samples over a requested interval, and the region tool exposes scene-region information and associated context.

F Method Detail - Memory Construction and Storage

F.1 Object records and observation support

Each object o_i has a public identity, an internal index i, mapped coordinates p_i, semantic features, and a union H_i of stored observation intervals. Trajectory samples retain timestamp t_s, position r_s, and heading θ_s when available. The internal indices provide a common domain for cell membership, observation associations, pair relations, and semantic scores. Object identities remain stable in the returned records.

A canonical visit is a maximal consecutive run of trajectory samples in one finest persisted trajectory cell. Its identity records an encounter, so a later return to the same cell creates a separate visit. In the native artifact construction, object observation timestamps are matched to nearby trajectory samples within a tolerance, then mapped to canonical visits. Consecutive supported visit IDs are merged into stored intervals spanning their recorded run endpoints. These intervals define the temporal evidence used by retrieval. The construction preserves encounter support as interval unions, including separate components for disjoint periods.

Let R_v be the object indices associated with visit v. Compressed association structures support both visit-to-object and object-to-visit lookup. Recorded intervals and canonical visits provide the authoritative evidence for runtime verification; spatial and trajectory search views are derived from these records. Runtime level selection therefore preserves the stored evidence. Artifact construction fixes the finest cells and visit segmentation from which that evidence is formed.

F.2 Occupied cells and shared membership

The index dimension $m \in \{2, 3\}$ selects XY or XYZ quantization. At level ℓ, cell width is $h_\ell = 2^\ell h_0$. Interleaving the bits of the quantized coordinates gives a Morton code; removing the lowest m coordinate bits identifies the parent at the next coarser level. Only occupied cells are retained. Figure 3 illustrates how the resulting levels provide alternative spatial groupings.

For cell assignment $g_\ell(i)$, the sets $\mathcal{O}_\ell(a) = \{i : g_\ell(i) = a\}$ partition the objects. Memberships occupy contiguous spans of a shared Morton-ordered array of object indices, with level-specific boundaries identifying each span. The singleton partition s has $\mathcal{O}_{\mathrm{s}}(i) = \{i\}$. It supplies direct object-level execution within the same choice of partitions.

Each spatial group stores a conservative box $B_\ell(a)$ and support union $U_\ell(a)$. XY bounds enclose the nominal cell and actual member extrema; XYZ bounds use actual member extrema. Outward numerical padding preserves conservative geometric comparisons. Exact distance checks use the original mapped coordinates. The hierarchy dimension and distance metric are independently configured: projected XY queries can search an XYZ hierarchy, while compatible XY candidate bounds can support subsequent XYZ verification. Bearing and robot observation-area predicates operate in world XY.

F.3 Associations across levels

Object–visit associations induce

$$I_\ell(v) = \{g_\ell(i) : i \in R_v\}. \tag{9}$$

At coarser levels, child cells are mapped to parents and repeated links are removed within each visit. Spatial extrema and support unions are likewise aggregated from children. Visit identities remain available throughout this construction, preserving the connection from a candidate cell to its member objects and their encounters.

Trajectory views organize chronological runs using the same spatial encoding. Adjacent runs can merge when their parent cell codes agree, and the merged view retains the canonical visits it covers. Object partitions, spatial occupancy views, and trajectory search views can use their own configured level sets. All reference the shared object or trajectory records needed for individual verification.

F.4 Constraint-Evaluation Algorithms

This subsection describes execution for a fixed partition and constraint order. Appendix F.5 then explains how these alternatives are assessed and selected.

F.4.1 Generating grouped pair candidates

For proximity-first retrieval, the requested radius and cell pitch determine a conservative neighborhood of cell offsets. The backend probes occupied cells in that neighborhood, retaining pairs whose spatial boxes can satisfy the distance condition. With an XYZ hierarchy and an XY distance request, lookup accounts for occupied heights through spatial columns. Within-cell candidates represent distinct eligible members of one cell. The singleton alternative operates directly on eligible object pairs.

For observation-first retrieval, each relevant visit supplies the linked groups $I_\ell(v)$. The backend generates unordered group pairs within each visit and consolidates repeated occurrences. Spatial partitions include diagonal pairs because two objects can occupy one cell. Let $d_v = |I_\ell(v)|$. The number of generated occurrences is

$$O_\ell = \begin{cases} \displaystyle\sum_v \binom{d_v}{2}, & \ell = \mathrm{s}, \\ \displaystyle\sum_v \frac{d_v(d_v+1)}{2}, & \ell \text{ is spatial.} \end{cases} \tag{10}$$

This count captures generation work before deduplication. Repeated visits can produce a large O_ℓ even when the set of distinct cell pairs is small.

F.4.2 Collective filtering and expansion

Candidate group pairs receive the remaining capacity, eligibility, spatial, temporal, and applicable visit checks. For a spatial pair (a, b), a box-distance lower bound greater than d rejects all member pairs. For co-observation, $U_\ell(a) \cap U_\ell(b) \cap W = \emptyset$ establishes the corresponding temporal rejection. Eligible counts also remove groups with no participating members and diagonal pairs with fewer than two.

Let F_ℓ contain the unique surviving group pairs. Its individual expansion has the exact size P_ℓ in Equation (3). Distinct cells contribute $n_a n_b$ unordered pairs, while a diagonal contributes $\binom{n_a}{2}$. Every object belongs to one cell at the selected level, so different group-pair blocks have disjoint individual expansions. This makes both the count and the resulting pair identities well defined before verification. The grouped relation can be assessed and retained without enumerating those individual pairs.

F.4.3 Individual verification

The selected spatial candidate expands into its eligible cross-cell products and within-cell distinct pairs. Exact checks evaluate the requested predicates using object coordinates and stored interval unions. When a selected-visit witness remains to be established, object-level associations verify a common allowed visit. A local witness check can avoid constructing the entire object-level witness relation; an admitted relation-based procedure supplies the corresponding fallback.

For combined requests, observation-first candidates check temporal support before distance, and proximity-first candidates check distance before temporal support. Evidence already established at singleton level can be reused. The backend orders the surviving pairs and aligned evidence consistently after filtering. Across valid execution choices over the same records, verification produces the same physical relation.

F.4.4 Single-object area and time retrieval

Unary retrieval applies the same grouping principle to individual membership. Candidate groups can be generated from eligible-object membership, spatial bounds, or temporal support summaries. Remaining group conditions are applied together, eligible members are counted, and the selected candidates are expanded. Individual checks then establish the mapped-area condition and support in

the requested window. This lets area and time restrictions reduce a shared search before semantic ranking.

Unary planning considers its applicable seed families and partitions. Its normal comparison permits up to two assessments per nontrivial seed family, with a separate bounded fallback using all groups. This policy is tailored to member expansion, while the pair policy in Appendix F.5 controls the larger combinatorial workload of member pairs.

F.4.5 Observation areas and trajectory context

A robot observation-area query uses trajectory cell bounds and run times to identify relevant portions of the path. Exact recorded samples establish membership in the requested circle and window. Their canonical visit IDs are deduplicated into $\mathcal{V}$, and visit associations supply the eligible objects. Selected samples, visits, and their interval extents remain available for later evidence construction.

Trajectory context can also be resolved by time or position. Timestamp lookup chooses a nearest recorded sample with deterministic ties. Spatial closest-approach search visits trajectory cells in order of their distance lower bounds and verifies enclosed samples, stopping once the remaining bounds exceed the best distance. Its dedicated level policy chooses an available cell width near the configured target scale. These operations supply the robot context associated with retrieved objects and events.

F.4.6 Robot-relative bearing

At a recorded sample with finite heading, define the object's horizontal bearing relative to the robot by

$$\beta_{is} = \mathrm{wrap}_{\pi}(\mathrm{atan2}(p_i^y - r_s^y, p_i^x - r_s^x) - \theta_s)\,, \tag{11}$$

$$e_{is} = |\mathrm{wrap}_{\pi}(\beta_{is} - \beta^*)|\,. \tag{12}$$

Here β^* is the signed target angle, positive for left and negative for right, and angles in these equations are in radians. A qualifying event has sample-time support $t_s \in H_i$, nonzero horizontal displacement, range at most $r_{\max}$, and $e_{is} \le \varepsilon$ for the requested tolerance.

Windowed retrieval selects relevant trajectory runs, obtains their associated eligible objects, and evaluates finite-heading samples within W. Computation uses local object-by-sample blocks. Run associations supply candidates, and individual sample-time support checks establish the observation evidence for each tested event. Each object retains its best qualifying sample under the ordering (e_{is}, t_s, s), prioritizing angular agreement and then earlier time and sample identity.

The trajectory-level policy estimates work from the number of relevant runs, associated objects, and local object-by-sample products. If run u supplies n_u candidate objects and m_u finite-heading samples, the model includes $\sum_u n_u m_u$ as the evaluation workload and $\max_u n_u m_u$ for the largest local allocation. Coarser runs reduce run overhead while potentially increasing these products. The selected level must satisfy the modeled allocation checks, and runtime checks also cover the retained per-object event state. Appendix F.5.5 records the selection model.

An explicit robot reference time resolves a nearest finite-heading sample and evaluates that sample directly. The output records both the requested and resolved times. This preserves the meaning of the query when recorded samples are temporally sparse. Objects with qualifying events proceed to the bearing ranking policy in Appendix F.6.4.

F.5 Adaptive Planning and Resource Control

F.5.1 Shared statistics and request-specific counts

A pair execution plan combines an object partition with proximity-first or observation-first candidate generation. The catalog stores occupied-group counts and membership-size statistics for each partition, including mean occupancy μ_ℓ, second moment $m_{2,\ell}$, and maximum occupancy. Spatial pitch and scene bounds support neighborhood estimates, while interval counts describe temporal fragmentation.

Visit associations carry cumulative incidence and pair-occurrence counts. After locating the visit slice relevant to W, unrestricted counts follow from prefix differences. An additional selected-visit

restriction requires reducing the corresponding per-visit contributions. The resulting scalar facts are shared across alternatives, supplying selected-visit counts, association counts, generated occurrences, and a bound on distinct group pairs before candidate relations are constructed.

The planner distinguishes five populations: generated occurrences, distinct seed pairs, group pairs surviving collective filtering, prospective individual pairs, and verified survivors. These quantities determine different costs. Repeated visits affect generation and deduplication; cell occupancy affects expansion; individual predicates determine the final verified relation.

F.5.2 Estimating execution work

For spatial candidate generation, the requested distance and cell pitch determine an approximate neighborhood size. Multiplying its probe count by occupied-cell density gives an estimate of candidate group pairs, capped by the pair universe. For visit-based generation with O_ℓ occurrences over U possible group pairs, the distinct-count approximation is

$$\widehat{D}_\ell = U\big(1 - \exp(-O_\ell/U)\big), \tag{13}$$

with zero counts handled directly. This accounts for repeated generation while leaving the actual distinct relation to assessment. Estimates for combined constraints restrict one another, and the eligible-object fraction adjusts relevant populations.

Estimated member expansion uses occupancy moments. A cross-cell pair contributes approximately μ_ℓ^2 members, while a diagonal contributes $(m_{2,\ell} - \mu_\ell)/2$. These quantities estimate the object-pair population exposed by grouped filtering. Remaining-work estimates also include endpoint materialization, distance checks, interval comparisons, applicable visit-witness checks, and output handling. They provide structural work comparisons across plans.

Grouped assessment replaces estimated expansion with the exact P_ℓ. Individual survivor counts and execution time remain estimated until verification. This distinction lets the planner improve its decision using actual candidates without paying every alternative's full member-level cost.

F.5.3 Bounded assessment and selection

Among alternatives whose grouped assessment fits the modeled memory budget, the primary plan minimizes a structural estimate combining assessment and remaining work. A second candidate can be nominated for a favorable assessment-versus-refinement tradeoff or uncertainty in the initial estimate. After the primary is assessed, the planner evaluates whether that comparison is worth performing. Normal pair selection assesses at most two plans in total across partitions and generation orders; a separate admission-failure fallback can try at most two additional choices.

The comparison uses magnitudes of predicted work to identify materially different profiles. A nominated challenger is assessed when the primary is rejected, when its predicted refinement advantage warrants the additional grouped work, or when the primary's assessment-derived remaining-work score substantially exceeds its prediction and the alternative predicts an improvement. The current underestimation trigger uses a factor greater than four. The decision is bounded so that selecting an execution does not require running every available alternative.

Final selection concerns work still to be performed. Retained candidates are compared using remaining verification and output work. If a candidate relation must be regenerated, its comparison includes that regeneration cost. Thus grouped-assessment cost informs initial nomination and any required replay, while already completed work remains separate from the final remaining-work decision. Algorithm 1 summarizes the process.

F.5.4 Memory admission and candidate retention

Conservative bounds control the allocations required for candidate generation and filtering. After grouped assessment, P_ℓ determines the size of the prospective member-pair arrays and supports bounds for subsequent checks. Work estimates prioritize alternatives, while these allocation bounds govern whether their execution can proceed.

An assessed relation may be retained for direct refinement. If its continued presence would exceed the modeled budget during another assessment or refinement, the planner can retain the scalar assessment

Algorithm 1 Adaptive grouped pair retrieval

Require: eligible objects A; requested predicates; object and observation memory
Prepare shared statistics and request-specific visit counts
Estimate work and check assessment bounds for available plans
Nominate a primary plan and a possible comparison plan
Generate and collectively filter the primary's grouped candidates
Count their exact prospective expansion and check refinement bounds
if a comparison is justified **then**
    Assess the nominated alternative in the same way
end if
Apply the bounded admission-failure fallback when needed
Select an admissible candidate using remaining work and regeneration cost
Retain or regenerate its grouped relation as required
Expand the selected spatial relation into individual pairs
Verify remaining distance, temporal, and selected-visit predicates
return the verified pair relation and aligned evidence

and regenerate the relation if selected. This keeps the choice of execution coupled to the memory it requires. Resource exhaustion is reported explicitly when the considered exact executions cannot be admitted. The accounting covers modeled working arrays and their execution schedules; it does not represent total process memory. Broad requests can still require a quadratic pair population.

F.5.5 Trajectory-level planning

Windowed bearing uses its own work model over trajectory levels. For each level, the backend obtains relevant-run count V_ℓ, associated-row work L_ℓ, and local object-by-sample work M_ℓ. The selection score has the form

$$\widehat{C}_\ell = b + w_v V_\ell + w_l L_\ell + w_m M_\ell. \tag{14}$$

These terms are computed from stored memberships and the requested window before individual range and angle evaluation. The largest local product supplies the corresponding allocation estimate. Admissible levels are considered in score order, with runtime checks determining whether the selected execution can complete. Reference-time lookup and nearest-position context use their respective sample- or geometry-based policies.

F.6 Combination Search, Ranking, and Evidence

F.6.1 Physical combinations and common support

Let E be the verified pair relation. A k-object combination requires a clique of size k, establishing every requested pair relationship. For temporal queries, the full combination also requires joint support. For example,

$$H_1 = [0,1] \cup [4,5], \qquad H_2 = [0,1] \cup [2,3], \qquad H_3 = [2,3] \cup [4,5] \tag{15}$$

have pairwise overlaps, yet $H_1 \cap H_2 \cap H_3 = \emptyset$. Checking the whole combination therefore resolves a temporal condition that its edges alone leave open. Ordinary requests use $H_i \cap W$; observation-area requests use the selected-visit support defined in Appendix E.2.

For three objects, the backend orients pair edges by object index and intersects forward neighbor lists. Each common forward neighbor of an edge produces one canonical triangle. Whole-group support is checked before accepting the candidate, and the associated pair evidence supplies its distances and observation information.

F.6.2 Joint semantic assignment

For a feasible combination C and an assignment π from descriptions to distinct members, define

$$\lambda(\pi) = \mathrm{sort}_\uparrow\big(s_{\pi(1),1}, \ldots, s_{\pi(k),k}\big). \tag{16}$$

Assignments maximize this vector lexicographically, and each combination is represented by its best assignment. This prioritizes the weakest role match, followed by progressively stronger matches.

Algorithm 2 Joint search for larger object combinations

Require: verified graph E; descriptions $q_1, \ldots, q_k$; support domain $\mathcal{H}$; result count K
Reduce E to its $(k-1)$-core and score the remaining objects
Remove semantically invalid vertices, reduce again, and orient adjacency
Initialize the retained answer set $\mathcal{B}$
function EXTEND(C, D, J)
    if $|C| = k$ **then**
        Solve the distinct-object role assignment by subset dynamic programming
        If a valid assignment exists, update $\mathcal{B}$ using leximin and deterministic ties
        return
    end if
    if $|C| + |D| < k$ or ($|\mathcal{B}| = K$ and the semantic bound is below the cutoff) **then**
        return
    end if
    for each candidate $v \in D$ in the traversal order **do**
        $J' \leftarrow J \cap \mathcal{H}_v$ for temporal queries; unrestricted otherwise
        if J' is nonempty **then**
            EXTEND($C \cup \{v\}, D \cap N^+(v), J'$)
        end if
    end for
end function
EXTEND($\emptyset, V(E)$, unrestricted support)
return the best K combinations in requested role order

With two roles, comparing the minimum score and then the mean is equivalent to this ordering. With more roles, the complete vector matters: $(0.6, 0.8, 0.8)$ outranks $(0.6, 0.6, 1.0)$ after the weakest scores tie.

Pair and triple candidates evaluate their two and six role assignments, respectively. Semantic features are prepared for relevant unique objects, and repeated descriptions share computation. Exact pair selection can restrict sorting to a pool above the primary-score cutoff while preserving all boundary ties. Physical combinations are therefore ranked jointly, with relation evidence and stable identities resolving later ties.

Leximin is consistent with coordinate-wise dominance: improving one or more role scores without decreasing any produces a better sorted vector at its first difference. Top-K retrieval orders feasible combinations by this objective, including the best distinct combinations after the leading result.

F.6.3 FOUR-OBJECT SEARCH WITH SEMANTIC PRUNING

Four-object requests use additional graph and semantic pruning to control combination search. The backend first removes vertices that cannot belong to a sufficiently large clique by repeatedly deleting those with fewer than $k-1$ remaining neighbors. It scores the remaining objects, removes vertices unable to fill any role, and repeats this graph reduction. A deterministic degeneracy ordering orients adjacency so that clique extensions are explored once.

During search, a partial combination maintains its possible extensions and, for temporal queries, the intersection of its members' support. Branches terminate when too few extensions remain or common support becomes empty. Once K answers are available, role-wise best scores among the selected and remaining objects give an optimistic semantic vector. A branch whose bound is strictly worse than the retained cutoff can be discarded. The bound permits reuse of an object across hypothetical roles, which makes it optimistic; completed combinations use a distinct-object assignment.

The completed assignment is solved by subset dynamic programming. A state for member subset M stores the best assignment of the first $|M|$ roles to those members. Extending the state assigns the next role to an unused member and compares the resulting sorted score vectors. States with the same subset have the same possible continuations, so the best state is sufficient for subsequent extension. The full subset gives the exact role assignment for the combination. Algorithm 2 summarizes this search.

The traversal applies bounds to its graph, semantic scores, support state, assignments, and retained answers. Work and completed-combination limits are checked during execution; exhaustion is reported explicitly. The public interface uses this search for four-object requests, while its construction is expressed in terms of the combination size k.

F.6.4 SINGLE-OBJECT AND BEARING RANKING

Ordinary single-object retrieval ranks eligible objects by semantic similarity, with stable identity ties. Nearest-ranking modes preserve strong semantic matches while also selecting nearby candidates within a semantic plausibility band. The resulting list combines semantic anchors, nearby plausible objects, and candidates whose semantic quality improves with distance. This lets proximity influence selection among semantically relevant alternatives.

Bearing first selects each object's best qualifying recorded event. It then ranks objects using semantic rank and angular-error rank, prioritizing their sum and resolving ties through the component ranks, angular error, event time, and identity. Strong semantic and angular candidates enter the candidate pool before identity deduplication and truncation. Returned event information remains attached to the selected object's qualifying sample.

F.6.5 EVIDENCE MATERIALIZATION

Returned combinations preserve description order and report each member's identity, mapped position, and semantic match. Proximity evidence records the requested metric and pair distances. Common-observation evidence summarizes the common support through its component count, total duration, enclosing time range, and a representative component. SOMo chooses the longest common-support component, resolving equal durations toward the earliest, and uses its midpoint as the representative event time.

When robot context is requested for ordinary co-observation, the selected event is resolved to a recorded trajectory sample. The response distinguishes the event time, resolved sample time, and their difference. Under an observation-area restriction, context is selected from the qualifying visits and in-region samples, using a shared visit for temporal combinations. Proximity-only context can instead identify the robot's closest approach to the combination's mapped centroid. Bearing returns its verified event sample, with the requested reference time included when applicable. Unavailable robot context is represented explicitly.

These outputs let the agent use the selected objects together with the evidence relevant to the request. The interpretation follows the relation: common-support events describe co-observation, closest-approach context describes proximity along the recorded path, and bearing events identify a qualifying robot-relative view.

G Qualitative Demonstration: Contextual Statue Retrieval

Question: "Where is the statue in front of the Bill and Melinda Gates Computer Science building?"

SOMo and legacy DAAAM use Qwen3.5-9B, the same scene memory source, and identical question and robot context for the CODa sequence-0 OC-NaVQA `LONG` instance. Both must return an object position using the window 151.56–760.52 s. This historical window-restricted example is separate from the main evaluation's full-sequence OC-NaVQA protocol.

SOMo: pillar sculpture

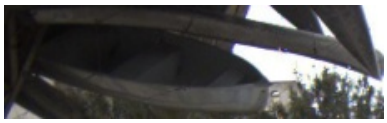




Corresponding object crop

DAAAM: curved sculpture





Corresponding object crop

Figure 5: **Matched visual presentation of the final selections.** Each row shows an original camera view (left) and the corresponding object crop (right). Boxes locate the crops; both rows use the same display widths and preserve image proportions. The curved sculpture is only partly visible in its source frame. Crops retain surrounding image content for visualization.



Building candidate returned with the pillar sculpture. This separate camera view supplies visual context for SOMo's paired building record. The returned pair is 7.97 m apart in the mapped horizontal plane, satisfying the agent's 10 m proximity condition. The view comes from a different observation than the sculpture image; the query does not require simultaneous visibility.

Figure 6: Additional context for the joint result, distinct from the two methods' final object selections above.

G.1 Retrieval Traces and Interpretation

Call traces. Tool names are exact; arguments and returns are abbreviated. "Building name" refers to the question; search centers replace coordinate vectors with descriptions.

Call	Tool and key arguments	Selected evidence / role of the call
SOMo: two calls		
1	`search_objects`: statue + building name; `near_each_other`, 10 m; supplied window	Top pair: pillar sculpture + building candidate; horizontal separation 7.97 m.
2	`search_objects`: statue; `individual`; same window	Curved sculpture ranks first; pillar sculpture third. The agent retains the joint result.
Final: pillar sculpture		**Position error: 10.64 m**
Legacy DAAAM: fourteen calls		
1	`get_matching_subjects`: building name	Top return: a light-gray building candidate.
2	`get_matching_subjects`: statue in front of [building name]	Top return describes a building, not a statue.
3	`get_matching_subjects`: statue	Curved sculpture ranks first; pillar sculpture is already present, in third place.
4–8	`get_objects_in_radius`: five candidate locations, radius 15 m each	Checks neighborhoods of four building candidates and a wall with science-hall lettering.
9	`get_matching_subjects`: Gates Computer Science building statue	Reformulates the query; top return is another building candidate.
10	`get_objects_in_radius`: call 9's top building, radius 15 m	Inspects that candidate's neighborhood.
11	`get_region_information`: no arguments	Current-region description mentions a statue.
12	`get_objects_in_radius`: current robot location, radius 15 m	Searches locally; leading returns are trash cans.
13	`get_matching_subjects`: statue, sorted by distance	Switches from relevance ranking to distance from the current robot location.
14	`get_objects_in_radius`: curved sculpture's location, radius 15 m	Inspects its surroundings before context-limit finalization.
Final: curved sculpture		**Position error: 141.52 m**

Why the selections differ. The pillar sculpture is available to both agents: it appears third in DAAAM's statue search and in SOMo's later statue-only search. SOMo's first call instead ranks the sculpture together with a building candidate under the proximity constraint. Its final answer explicitly relies on that joint context, despite the curved sculpture's stronger individual semantic rank. DAAAM continues searching after seeing both sculptures and, at the tool-phase context limit, selects the curved sculpture while acknowledging that the building association remains unconfirmed.

Scope of the comparison. Both predictions are measured against the same annotated statue position, $[-67.66, 207.50, 1.17]$ m. The recorded advantage is fewer tool calls and lower position error, with 10.64 m residual error for SOMo. The agent interprets "in front of" as proximity; neither the image layout nor this trace establishes a verified directional front-of relation. The example illustrates how joint context supports the final selection, rather than isolating the contribution of an individual backend component.

H QUALITATIVE DEMONSTRATION: RETRIEVING A YELLOW POLICE POLE

Question: "Where did you see the yellow police pole?"

Both methods use Qwen3.5-9B and the same CODa sequence-16 scene source. The identical input specifies history 487.69–561.08 s, the current robot position, and an *object-position* answer. This historical OC-NaVQA example uses an observation window, unlike the full-sequence protocol in the main evaluation. The comparison below uses the same question and ground truth for both methods.

SOMo: yellow call pole





Corresponding object crop

DAAAM: dark pole fallback





Corresponding object crop

Figure 7: **Matched views of the selected objects.** Original camera frames and corresponding crops, with boxes locating the displayed objects. Both rows use the same frame widths and crop display bounds, preserving proportions. Crops retain their backgrounds. The frames were recorded at 558.28 s (SOMo) and 518.88 s (DAAAM), both within the requested window.

The yellow call pole has a blue top and light fixture; the other record is a dark metal pole. DAAAM acknowledges that it did not identify a yellow police pole and uses the dark pole as its best estimate. The error comparison concerns these mapped object positions, not the robot locations from which the images were acquired.

H.1 Retrieval Traces and Interpretation

Tool names are exact; key arguments and selected returns are summarized. DAAAM's semantic searches use relevance ordering. Times are seconds from the start of the sequence.

SOMo: one call		
Call	Tool and key arguments	Selected evidence / role of the call
1	`search_objects`: yellow police pole; `individual`; window 487.69–561.08 s; `semantic_then_nearest_to_search_center`, centered at the current robot position	Top return: yellow pole with a blue light; support 555.98–561.08 s. The agent returns its object position.
Final: yellow call pole		**Object-position error: 0.60 m**
Legacy DAAAM: seven calls		
1	`get_matching_subjects`: yellow police pole	Top return: a yellow pole observed at 96.09–102.90 s, outside the requested window.
2	`get_matching_subjects`: yellow pole police	Reordered wording; the same yellow pole remains first.
3	`get_matching_subjects`: yellow pole	Broader description; the same pole remains first.
4	`get_matching_subjects`: police	Top return: a yellow HELP banner from a later observation, outside the window.
5	`get_matching_subjects`: pole	Returns the eventual dark-pole fallback, observed at 518.18–519.98 s, among the candidates.
6	`get_agent_trajectory_information`: 487.69–561.08 s	Retrieves the robot's route during the requested period.
7	`get_objects_in_radius`: center $[210, 78, 7.5]$ m, radius 15 m	Inspects a neighborhood along that route; leading results are stone slabs.
Final: dark metal pole fallback		**Object-position error: 46.00 m**

How context enters the search. SOMo applies the observation window during object retrieval and returns a semantically appropriate candidate in one call. DAAAM first receives semantic matches from other parts of the history, then examines time information and the trajectory. Its final fallback satisfies the requested period but not the yellow-pole description. Both final answers are valid numerical predictions; the difference is in their grounding.

What the result supports. The object-position errors are 0.60 and 46.00 m against the same annotation. This trace illustrates contextual candidate selection, with time filtering, ranking, and agent decisions all contributing to the observed execution. The nearest-to-current-position ranking is the agent's choice; the user question does not explicitly ask for the nearest pole. These are position errors, not navigation-success measurements.

I Qualitative Demonstration: Recovering a Shared Encounter

Question: "Where was the robot when the bike and pole were observed together in the same frame?"

This CODaAG sequence-16 example asks for a *robot position* within 1519.88–1557.35 s, without supplying an event time. Both methods use Qwen3.5-9B, the same scene source, and identical question inputs and ground truth. This is a guided example: both input wrappers explicitly prescribe SOMo's joint retrieval API and the robot-position output field, including in the legacy input.

SOMo: bicyclist record





Corresponding object crop

DAAAM: fallback pole





Corresponding object crop

Figure 8: **Records used to construct the answers.** Matched camera views and background-preserving crops, with boxes locating each crop. SOMo's bike-role record centers on the rider. DAAAM uses the displayed pole's object coordinates as its answer, although the question asks for a robot pose. These camera frames are separate observations.





Pole returned with the bicyclist record. This sign-pole record completes SOMo's top pair. Its source view is at 1521.05 s; the rider's is at 1525.85 s. The stored observation intervals provide the shared support used to recover an encounter pose.

Figure 9: The second role in SOMo's result. Separate source images identify the records; they do not by themselves verify a same-frame encounter.

I.1 Retrieval Traces and Interpretation

The requested output is the robot's location at an encounter. Images on the preceding page identify the retrieved records; they are not the final answer coordinates. Tool names are exact; arguments and returns are abbreviated.

SOMo: one call		
Call	Tool and key arguments	Selected evidence / role of the call
1	`search_objects`: bike + pole; `observed_together`; window 1519.88–1557.35 s; `return_robot_observation_info=true`	Top pair: bicyclist record + sign pole. Common support selects event 1525.90 s and returns a robot pose.
Final: returned encounter pose		**Robot-position error: 0.07 m**
Legacy DAAAM: six calls		
1	`get_matching_subjects`: bike	Top bicycle was observed at 1684.95–1697.25 s, outside the window.
2	`get_matching_subjects`: pole	Top pole's support is 790.87–812.37 s, also outside the window.
3	`get_matching_subjects`: bicycle	Synonym search returns the same leading bicycle.
4	`get_matching_subjects`: bike	Repeats the bike search; the leading result is unchanged.
5	`get_matching_subjects`: metal pole	The same leading pole remains outside the requested period.
6	`get_matching_subjects`: vertical pole	Includes the eventual fallback pole, observed at 1562.25–1567.65 s, after the window.
Final: fallback pole's object coordinates		**Robot-position error: 27.89 m**

From shared support to a robot pose. SOMo's returned common-support interval is 1520.95–1530.85 s. Its representative event, 1525.90 s, resolves to a trajectory sample at 1525.75 s and pose $[-52.095, 92.826, -0.600]$ m. The annotated robot pose is $[-52.164, 92.848, -0.596]$ m. DAAAM reports that it could not establish a qualifying encounter, then substitutes a pole's mapped position for the requested robot position.

Accurate encounter location, imperfect object grounding. The bike-role record describes a bicyclist and centers on the rider. Its mapped geometry is inaccurate, and correspondence to the annotated bike and pole instances has not been independently verified. The stored overlap lasts 9.9 s, whereas the annotation identifies approximately 0.10 s of same-frame support around 1525.45 s. The low robot-position error therefore does not validate the object geometry or continuous same-frame visibility throughout the stored interval.

Guidance in the input. Both wrappers name `search_objects`, `observed_together`, and `robot_observation.robot_position`; they interpret same-frame evidence through overlapping stored support. This favors SOMo's interface, so the case demonstrates guided encounter retrieval rather than an unaided tool-selection comparison. All six legacy searches use relevance ordering.

J QUALITATIVE DEMONSTRATION: A PAST ROBOT REFERENCE

Question: "At 710.5 seconds, what is the 3D position of the newspaper dispenser nearest to the robot?"

This CODaAG sequence-16 example requests an *object position*, with history 704.56–748.47 s and reference time 710.48 s in the supplied context. Both methods use Qwen3.5-9B, the same scene source, and identical question inputs and ground truth. Both input wrappers name SOMo's historical-reference API, including in the legacy input; this is a guided retrieval demonstration.

SOMo: newspaper dispenser





Corresponding object crop

DAAAM: gate fallback





Corresponding object crop

Figure 10: **Matched views of the selected objects.** Original camera frames and corresponding background-preserving crops, with equal frame widths and crop display bounds. Boxes locate the crops. The dispenser frame is at 708.38 s; the gate frame is at 1687.14 s, outside the requested window.

The dispenser is recognizable in its camera view and crop. DAAAM's gate is an acknowledged fallback, not a claimed visual match to the requested object. Its source view comes from a later part of the sequence, outside the question's observation window.

J.1 Retrieval Traces and Interpretation

Tool names are exact; arguments and selected returns are abbreviated. Every legacy call below uses `sort_by=distance`; the returned distances refer to the current robot position.

Call	Tool and key arguments	Selected evidence / role of the call
SOMo: one call		
1	`search_objects`: newspaper dispenser; `individual`; window 704.56–748.47 s; `robot_trajectory_time=710.48`; `semantic_then_nearest_to_robot`	Top return: white/orange dispenser, 1.172 m from the resolved historical robot position. The agent returns the object's mapped coordinates.
Final: newspaper dispenser		**Object-position error: 0.19 m**
Legacy DAAAM: seven calls		
1	`get_matching_subjects`: newspaper dispenser	Leading result: a light-switch plate.
2	`get_matching_subjects`: newspaper	Leading result: a diagonally striped signboard.
3	`get_matching_subjects`: dispenser	The light-switch plate leads again.
4	`get_matching_subjects`: kiosk	Leading result: the gate later used as the fallback.
5	`get_matching_subjects`: newsstand	The same gate remains first.
6	`get_matching_subjects`: vending machine	Leading result: a weathered metal signboard.
7	`get_matching_subjects`: cabinet	The light-switch plate leads again; no clear dispenser is identified.
Final: gate fallback		**Object-position error: 366.38 m**

Resolving the historical reference. SOMo resolves the requested 710.48 s to the recorded trajectory sample at 710.38 s, position $[357.336, 59.966, 8.206]$ m. It returns the dispenser at $[358.303, 60.627, 7.770]$ m, with stored observation support 704.56–712.58 s. Its separately returned representative observation time is 708.57 s; this is distinct from the reference time used for nearest-object ranking. DAAAM does not recover a historical robot pose in this trace. Its final reasoning acknowledges the missing dispenser and uses a gate's coordinates as a best estimate.

Interpreting the comparison. The reference to a past robot position is expressed directly in SOMo's retrieval request. Semantic matching and DAAAM's fallback also contribute to the difference in errors. The annotation contains only one same-class candidate at the reference event, so the example does not establish disambiguation among several simultaneously eligible dispensers.

Guidance in the input. Both wrappers prescribe `search_objects` with `robot_trajectory_time` and the `semantic_then_nearest_to_robot` ranking mode. These instructions favor SOMo's interface and limit the comparison as evidence of independent agent planning. The trace illustrates historical reference handling under that guidance, with object-position errors of 0.19 and 366.38 m against the same annotation.

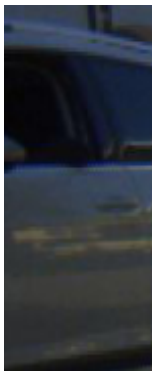
K Qualitative Demonstration: Timing a Shared Observation

Question: "Around what time were the car and delivery truck observed together in the same frame?"

This CODaAG sequence-6 example asks for an event time within 874.33–918.43 s. Both methods use Qwen3.5-9B, the same scene source, and identical question inputs and ground truth. Both select the same silver-car and yellow-truck records. SOMo returns their shared observation support in one call; legacy DAAAM reconstructs an overlap from separate results in three calls. The timing errors are modest: 1.60 s and 4.40 s, respectively. As in the preceding generated examples, both input wrappers explicitly prescribe SOMo's API and event-time field.

Car record: both methods





Corresponding object crop

Truck record: both methods





Corresponding object crop

Figure 11: **The same selected records for both methods.** Full camera views and corresponding crops identify the partial car-side record and the yellow Penske truck. Boxes locate the crops. These source views are at 912.56 s and 915.56 s, respectively; they identify the records rather than depict a single simultaneous observation.

K.1 Retrieval Traces and Interpretation

Tool names are exact; arguments and returned evidence are abbreviated. All times are seconds from the start of the sequence.

SOMo: one call

Call	Tool and key arguments	Selected evidence / role of the call
1	`search_objects`: car + delivery truck; `observed_together`; window 874.33–918.43 s; `return_robot_observation_info=true`	Top pair: silver car + yellow Penske truck. Shared support 906.76–913.16 s; returned event time 909.96 s.
Final: 909.96 s		**Time error: 1.60 s**

Legacy DAAAM: three calls

1	`get_matching_subjects`: car; `sort_by=relevance`	Includes the selected silver car, observed at 902.06–912.66 s.
2	`get_matching_subjects`: delivery truck; `sort_by=relevance`	Top result is the same yellow truck, observed at 907.16–922.16 s.
3	`get_objects_in_radius`: center at the car's returned position; radius 15 m	Returns the car and nearby records. The final answer uses the overlap of the two previously returned observation ranges.
Final: 907.1638 s		**Time error: 4.40 s**

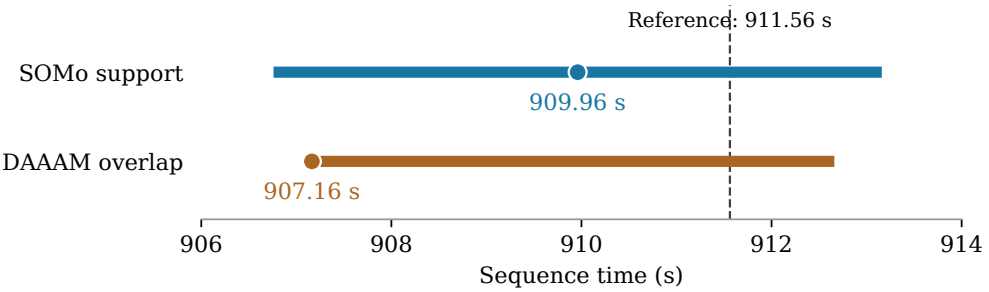


Figure 12: **Two supported intervals, different representative times.** Dots mark the answers; the dashed line marks the evaluated reference time, 911.56 s. SOMo uses the midpoint of its returned common support. DAAAM uses the start of the overlap it computes from first/last-observed times. Both intervals contain the reference time.

A modest gain with a shorter reasoning path. The benefit here is direct access to a joint observation and its event time. DAAAM successfully identifies the same pair and derives a plausible interval, 907.16–912.66 s. Its larger point error reflects its choice of the overlap's start, while SOMo's representative time lies closer to the annotation. This case illustrates reduced tool use and explicit temporal evidence; it does not establish that DAAAM failed to find the encounter.

Scope of the temporal evidence. The stored support intervals differ between interfaces and are not frame-by-frame visibility guarantees. The annotation records four supporting frames spanning 911.46–911.76 s, with an evaluated reference at 911.56 s. Both query wrappers name `search_objects`, `observed_together`, and `robot_observation.event_time_s`. The comparison therefore demonstrates guided temporal retrieval rather than unaided tool selection.