Persistent Scene-Graph Memory on a Legged Robot: Separating Memory Gains from Execution Failure

Kanan Aliyev[1] **Niveditha Kalavakonda**[2]
[1]Khoury College of Computer Sciences [2]Dept. of Mechanical and Industrial Engineering
Northeastern University, Seattle, WA, USA
{aliyev.k, n.kalavakondachandra}@northeastern.edu

Abstract: A robot that returns to the same building can reuse its map instead of searching from scratch. The lifelong object-navigation benchmarks that persistent-memory navigators report on model the agent as a rigid cylinder, so it is not known how much of the memory benefit survives a body that can catch on furniture and fall. We evaluate persistent scene-graph memory for chained object-goal navigation on a simulated Unitree Go2 quadruped in Isaac Sim, over 2500 paired episodes in 25 houses, five chained subtasks per house. Carrying the scene graph across subtasks raises success rate from 0.38 to 0.51, and the gain grows as the graph accumulates, from none on the first subtask to 0.21 on the fourth. Of 1357 failures, 64% are stalls or falls, and 95% of the stalls end within half a body length of an obstacle. Without the carried scene graph, an object record store raises success at $p = 0.023$; restricted to pairs in which both robots made at least one decision, the same comparison gives $p = 0.47$.

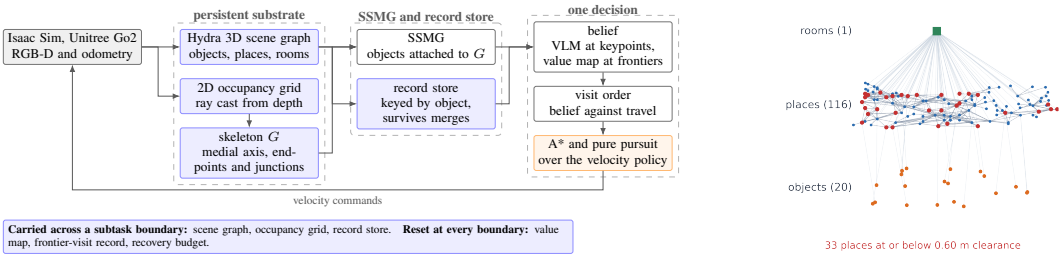
Keywords: Object-goal navigation, 3D scene graphs, Legged robots, Evaluation

1 Introduction

A robot that works in one building over many tasks should not rediscover the layout for every request. Zero-shot object navigators score frontiers with a language-grounded value map [3] or prompt a language model with an online scene graph [7, 8, 9, 12], and discard the map after each goal. Persistent-memory navigators [11, 10, 2] keep it across a chain of goals [4], usually as a 3D scene graph built online from RGB-D frames [1, 6, 13]. The lifelong benchmarks these systems report on run in Habitat [5], where the agent is a rigid cylinder that slides along obstacles instead of catching on them [4, 2].

A legged robot can catch on a doorframe, get stuck between furniture, or fall, and in our setup a fall ends every remaining subtask in the house. Failures like these limit how much any memory can help. They also bias a memory comparison when one of the two robots stops moving more often, because the difference in success rate then partly measures execution rather than memory. To measure the memory effect, each failure must be attributed to a pipeline stage, and the comparison must account for differential execution failure. Earlier failure analyses found that Habitat agents exploit sliding along walls, which lowers the correlation between simulated and real performance [15], and that the error modes of object navigators differ between simulation and real homes [16]. Those studies explore how far simulation predicts hardware for single goals; we explore whether execution failure biases a paired comparison of memory configurations.

We run SSMG-Nav's skeleton-graph planner [2] over a Hydra scene graph [1] on a Unitree Go2 in Isaac Sim [18], and compare a scene graph carried across a chain of five object-goal subtasks against one rebuilt for every subtask, with and without an object record store, over 2500 paired episodes. We report success rate (SR) and success weighted by path length (SPL). Our contributions are:



(a) one decision, end to end

(b) the substrate, scene `val_0`

Figure 1: (a) One decision, end to end; blue persists across subtask boundaries. (b) The Hydra graph of one house after four subtasks: 20 objects, 116 places, 1 room.

- A stage-level failure attribution for persistent-memory object navigation on a legged robot: 63.8% of failures are stalls or falls, and 95% of the stalls end within 0.35 m of an obstacle.
- A paired evaluation in which the carried scene graph raises SR by 0.125, growing from $+0.01$ on the first subtask to $+0.21$ on the fourth.
- A check for execution confounds in memory comparisons: restricted to pairs in which both robots made at least one decision, an apparent record-store gain disappears ($p = 0.023$ before, 0.47 after), while the scene-graph gain remains.

2 Method

Task. In each house, the robot receives $K = 5$ object categories in the order $[A, B, C, A, B]$, so the fourth and fifth subtasks repeat the first two after two intervening goals. A subtask ends when the agent declares arrival, when its budget of $30\,000$ simulation steps runs out, or when the robot falls. A fallen robot is not righted, so a fall also fails every later subtask in that house. A subtask succeeds if the agent declares arrival within 1.0 m of the bounding-box surface of any instance of the category, following GOAT-Bench [4].

System. The robot is an articulated Unitree Go2 in Isaac Sim, walking under a velocity-tracking policy trained with `unitree_rl_lab` [19], with a front RGB-D camera (640×480, $79.9°$ horizontal field of view) and poses from the simulator. Object labels come from the simulator's ground-truth semantics in every configuration, which isolates memory and execution from detection error. Hydra [1] builds the scene graph from the RGB-D stream. In parallel, a 2D occupancy grid is ray-cast from depth, and its free-space medial axis is pruned into a skeleton graph of corridor junctions and endpoints (Fig. 1). The SSMG-Nav planner [2] attaches objects to skeleton keypoints. Qwen3.5-9B [20] scores each keypoint from its attached labels and a bird's-eye crop of the map, and VLFM's value map [3] scores frontiers. The planner visits candidates in an order that trades belief against travel distance. A* plans a path on the occupancy grid, with obstacles inflated by 0.20 m, and a pure-pursuit controller turns it into velocity commands for the locomotion policy. A stalled robot reverses open-loop for 1.5 s, at most twice per subtask, and a further stall ends the subtask (Appendix C). Appendix E lists every departure from the published SSMG-Nav.

Failure stages. Every failed subtask is assigned to exactly one stage, checked in order. GROUND: the agent declared the object absent. Every category exists in every house, so this is always wrong. SELECT: the agent never chose the right target within its budget. MOVE: the robot stopped making progress or fell before the agent finished, whether it was heading for a chosen object or a frontier. We split one cause out of SELECT: subtasks in which Hydra never produced a graph, so the agent had nothing to decide on. All the metrics are defined in Appendix A.

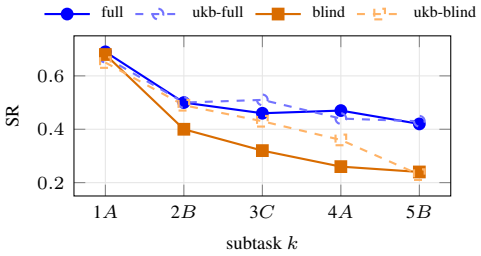


Figure 2: SR by subtask, 125 subtasks per point. Subtasks 4 and 5 repeat the categories of 1 and 2. Names drop the `ssmg-` prefix.

config.	fail	GROUND	SELECT	no graph	MOVE	fallen
full	309	19	67	0	223	66
ukb-full	306	12	87	0	207	53
blind	387	32	99	25	231	94
ukb-blind	355	34	95	21	205	77

Table 1: Failure attribution, 625 subtasks per configuration. The four stage columns partition the failures. No graph: Hydra never produced one (split out of SELECT). Fallen: MOVE failures on or after a fall.

3 Experimental Setup

Houses. We use 25 single-floor ProcTHOR houses [14] of 155 to 399 m^2, each with all five goal categories (sofa, bed, toilet, sink, houseplant), and Latin-square category orders (Appendix F).

Configurations. In `full`, one Hydra instance runs through all five subtasks of a house. In `blind`, each subtask starts a new Hydra instance, and objects seen in earlier subtasks are hidden from target selection. Both carry the robot's pose and the occupancy grid, so they plan on the same free space. The record store, a unified knowledge base (UKB), is off in `ssmg` and on in `ssmg-ukb`. It keeps per-object records through Hydra's node merges and gives the planner each candidate's observation history and a log of where earlier subtasks searched (Appendix D). We treat it as a second factor, not a contribution.

Statistics. Each configuration runs all 125 subtasks five times, giving 625 subtasks per configuration and 2500 in total. Under identical code, the median house's SR differs by 0.60 between its best and worst repeat, so a single repeat is too noisy. Two configurations are compared on pairs of episodes with the same repeat, house and subtask, run on the same machine, with an exact two-sided sign test over the pairs whose outcomes differ. Because the five subtasks in a house share a map and a robot, the 95% intervals come from a 10 000-sample bootstrap that resamples whole houses.

4 Results

4.1 Effect of the carried scene graph

`full` and `blind` both start the first subtask with an empty graph, so a gain from carried memory can only appear from the second subtask on. Over 625 pairs, carrying the graph raises SR from 0.381 to 0.506 (interval $[+0.059, +0.189]$, 142 pairs succeed only with the graph carried and 64 only blind, $p < 0.001$) and SPL from 0.194 to 0.303 (interval $[+0.065, +0.154]$; all metrics in Table 2). By subtask the gap is $+0.01$, $+0.10$, $+0.14$, $+0.21$ and $+0.18$ (Fig. 2).

On a repeat, the carried graph helps mainly when the first request already found the object: the fourth subtask succeeds in 53 of 86 runs (0.62) against 28 of 85 (0.33) blind after a first-request success, and in 6 of 39 (0.15) against 5 of 40 (0.12) after a failure (Table 4).

4.2 Failure attribution

Of 1357 failures across all 2500 subtasks, 866 (63.8%) are MOVE (Table 1). SELECT accounts for 348 (25.6%), GROUND for 97 (7.1%), and a missing graph for 46 (3.4%). Within MOVE, the robot stalled while exploring in 401 subtasks and while approaching a chosen object in 174, and 290 subtasks ended on a fall or after one (21 of these falls are inferred, Appendix A). The agent had chosen an object in only 191 of the 866; in 217 the robot was down before any decision in that subtask.

We compared each stall site with the simulator's colliders and with the robot's live grid (Appendix C). Of 575 stalls, 545 (95%) end within 0.35 m of an obstacle, half the Go2's length, against 10.4% of path points. Median clearance is 0.21 m, the edge of the 0.20 m inflation. The live grid held that obstacle in 495 stalls (86%) and lacked it in 50 (9%). Unlike the stalls, 53 of the 89 falls happen more than 0.35 m from any obstacle, which points to locomotion instability, though we logged no contacts. Only 17% of the 907 subtasks with a retreat succeed, against 62% of the rest.

On the repeated subtasks, the carried graph cuts exploration stalls from 50 to 25 and raises approach stalls from 8 to 39, for 64 stalls in total against 58 blind. With the record store the counts move the same way (47 to 33, 13 to 34).

The blind configurations lose more subtasks before their first decision, and these losses count toward the scene-graph gain: 133 against 52 for `ssmg`, of which 71 against 52 are falls and 62 against 0 are stalls. In 25 of these 62 stalls, the Hydra instance restarted for the subtask never produced a graph (Section 5(c)); in the other 37, the robot had a live graph and stayed within 1 m of its start. As a sensitivity analysis, we restrict to pairs in which both robots made at least one decision; this removes the gap but conditions on a post-treatment variable. Under it the scene-graph gain is $+0.090$ SR (interval $[+0.027, +0.154]$, 454 pairs) and still grows from $+0.01$ on the first subtask to $+0.18$ on the fourth. Without conditioning on decisions, counting all 25 no-graph subtasks as blind successes, the worst case for the comparison, leaves $+0.085$, and dropping only those pairs gives $+0.112$ (interval $[+0.047, +0.178]$, 600 pairs). With the record store on, the gain is smaller on every estimate: $+0.045$, $+0.063$ (interval $[+0.002, +0.123]$), and $+0.034$ under the decision restriction, whose interval includes zero (Appendix C).

4.3 Execution failure in memory comparisons

The record store has no measurable effect with the graph carried: SR 0.510 against 0.506, interval $[-0.058, +0.072]$, $p = 0.89$. In the blind configurations it appears to help: $+0.051$ SR, $p = 0.023$ by the sign test. Those configurations also differ by 33 in subtasks that ended before the first decision (133 against 100), close to their 32-success gap, and a store that helps by tracking merges should help more with the graph carried, where Hydra merges more. Restricting to pairs in which both robots made a decision leaves $+0.021$ ($p = 0.47$), and the clustered interval on the unrestricted gain, $[-0.003, +0.106]$, includes zero.

5 Conclusion and limitations

On a simulated legged robot, a carried scene graph raises SR from 0.381 to 0.506, and the gain grows as the graph accumulates. Most failures (63.8%) are stalls and falls. Stalls cluster at obstacles already on the robot's map, which suggests testing a body-sized clearance model before a better map, and memory moves them from exploration to the approach without reducing them. Because execution failure can differ between configurations, memory evaluations on legged robots should report execution-failure rates for each configuration, alongside the effect restricted to pairs in which both robots made a decision.

Limitations. (a) All results come from one simulator, one locomotion policy and one house generator, so the execution share may differ on hardware. (b) Object labels and poses come from the simulator, which isolates memory from perception error but makes absolute success an upper bound and leaves the record store little to correct. (c) Restarting Hydra in the blind configurations produces the 46 no-graph subtasks. Keeping one Hydra instance and hiding earlier objects is not a clean control: Hydra creates no second node for an object it already holds, so hidden objects stay hidden, and in an earlier pilot 9 of 10 such subtasks never saw a candidate. A matched control must re-detect objects and reopen frontiers at each boundary; we have not run one. (d) Recovery is open-loop, a fallen robot is not righted, and we logged no contact forces, so obstacle proximity is evidence of a clearance failure, not proof. (e) We ran no ideal-body baseline, so we cannot say how much SR the legged body itself costs.

References

[1] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Robotics: Science and Systems (RSS)*, 2022.

[2] H. Niu, L. Zhang, X. Ji, R. Ying, P. Liu, and F. Wen, "SSMG-Nav: Enhancing lifelong object navigation with semantic skeleton memory graph," arXiv:2603.01813, 2026.

[3] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "VLFM: Vision-language frontier maps for zero-shot semantic navigation," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024.

[4] M. Khanna, R. Ramrakhya, G. Chhablani, S. Yenamandra, T. Gervet, M. Chang, Z. Kira, D. S. Chaplot, D. Batra, and R. Mottaghi, "GOAT-Bench: A benchmark for multi-modal lifelong navigation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024, pp. 16373–16383.

[5] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra, "Habitat: A platform for embodied AI research," in *IEEE/CVF Int. Conf. on Computer Vision (ICCV)*, 2019, pp. 9339–9347.

[6] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024, pp. 5021–5028.

[7] A. Rajvanshi, K. Sikka, X. Lin, B. Lee, H.-P. Chiu, and A. Velasquez, "SayNav: Grounding large language models for dynamic planning to navigation in new environments," in *Proc. Int. Conf. on Automated Planning and Scheduling (ICAPS)*, vol. 34, 2024, pp. 464–474.

[8] H. Yin, X. Xu, Z. Wu, J. Zhou, and J. Lu, "SG-Nav: Online 3D scene graph prompting for LLM-based zero-shot object navigation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[9] H. Yin, X. Xu, L. Zhao, Z. Wang, J. Zhou, and J. Lu, "UniGoal: Towards universal zero-shot goal-oriented navigation," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[10] J. Loo, Z. Wu, and D. Hsu, "Open scene graphs for open-world object-goal navigation," *The International Journal of Robotics Research*, 2025.

[11] X. Xu, Y. Wu, T. Xue, Y. Xu, X. Dong, X. Yang, and C. Xu, "VTM-Nav: Harnessing cross-episode experience for object-goal navigation with hierarchical visual-topological memory," arXiv:2607.14514, 2026.

[12] X. Xue, J. Hu, M. Luo, S. Xie, J. Chen, Z. Xie, K. Quan, W. Guo, M. Xu, and Z. Chu, "OmniNav: A unified framework for prospective exploration and visual-language navigation," 2026.

[13] Y. S. Fang, Z. Shi, J. Qiu, Z. Chen, J. Shi, H. Xu, J. Huo, and Y. Gao, "INHerit-SG: Incremental hierarchical semantic scene graphs with RAG-style retrieval," arXiv:2602.12971, 2026.

[14] M. Deitke, E. VanderBilt, A. Herrasti, L. Weihs, J. Salvador, K. Ehsani, W. Han, E. Kolve, A. Farhadi, A. Kembhavi, and R. Mottaghi, "ProcTHOR: Large-scale embodied AI using procedural generation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[15] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2Real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, 2020, arXiv:1912.06321.

[16] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot, "Navigating to objects in the real world," *Science Robotics*, vol. 8, no. 79, 2023.

[17] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, M. Deitke, K. Ehsani, D. Gordon, Y. Zhu, A. Kembhavi, A. Gupta, and A. Farhadi, "AI2-THOR: An interactive 3D environment for visual AI," arXiv:1712.05474, 2017.

[18] NVIDIA Corporation, "NVIDIA Isaac Sim," 2025. [Online]. Available: https://developer.nvidia.com/isaac/sim

[19] Unitree Robotics, "unitree_rl_lab: reinforcement learning environments for Unitree robots," 2025. [Online]. Available: https://github.com/unitreerobotics/unitree_rl_lab

[20] Qwen Team, "Qwen3.5: Towards native multimodal agents," 2026. [Online]. Available: https://qwen.ai/blog?id=qwen3.5

[21] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[22] J. Li, D. Li, S. Savarese, and S. Hoi, "BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models," in *Proc. Int. Conf. on Machine Learning (ICML)*, 2023, pp. 19730–19742.

Appendix

A Metric and failure-stage definitions

- **SR (success rate).** The fraction of subtasks in which the robot declares arrival *and* its final position is within 1.0 m of the target surface. A subtask that ran out of budget next to a sofa the agent never identified is not a success.
- **SR@0.5.** The same rule with a 0.5 m radius, applied to the same runs. SSMG-Nav and GOAT-Bench use the stricter radius. A sweep over more radii is left for future work.
- **OS (oracle success).** The fraction of subtasks in which the robot passed within 1.0 m of a target at any point, whether or not it stopped. The path is recorded every 5 cm.
- **SPL (success weighted by path length).** $\frac{1}{N}\sum_i S_i\, l_i / \max(p_i, l_i)$, with S_i the success indicator, l_i the shortest path from start to target and p_i the distance walked.
- **NE (navigation error).** Geodesic distance from the final position to the nearest target. Subtasks where the agent declared the target absent or recorded no valid final distance are excluded, so NE averages over 554, 572, 460 and 491 subtasks in table order, and is optimistic for all four.
- **dec.** Mean number of high-level decisions per subtask. **0-dec**: subtasks that ended before the first decision.
- **falls.** House runs that ended in a fall. One fall fails the current subtask and every later one in that run, so the fallen count in Table 1 is larger.

The stages are checked in order, and every failure lands in exactly one. A RESOLVE stage (the request could not be turned into a category) is 0 in every configuration and omitted. **Near-miss** is not a stage: it flags subtasks that ended inside the success radius without the robot declaring arrival (97 in total, 93 of them MOVE failures).

Inferred falls. In the first repeat of the two `full` configurations, 21 falls predate the agent's fall flag. They are inferred from the run of zero-decision, budget-exhausted subtasks a fall leaves behind. On repeats 2–5, where the flag exists, this rule finds 84 of 88 flagged falls, and 84 of the 85 subtasks it marks did fall.

B Full results and paired contrasts

Table 2: Pooled results with all metrics, 625 subtasks per configuration.

configuration	SR	SR@0.5	OS	SPL	NE (m)	dec.	0-dec	falls
`ssmg-full`	0.506	0.483	0.594	0.303	4.41	12.1	52	20
`ssmg-ukb-full`	0.510	0.490	0.587	0.295	4.89	13.2	41	18
`ssmg-blind`	0.381	0.373	0.448	0.194	5.06	14.4	133	26
`ssmg-ukb-blind`	0.432	0.414	0.494	0.219	4.71	14.4	100	25

Subtask-1 success also selects easier houses and runs without an early fall. Excluding runs that fell before the fourth subtask gives 0.68 against 0.38 after a success. The fifth subtask given the second shows the same pattern as the fourth given the first.

The blind configurations do not walk farther than the full ones (12.9 km against 12.4 km in total for `ssmg`), but more of their house runs end in a fall (26 against 20). With these counts, we cannot say whether falls depend on how the robot explores.

C Diagnosis of MOVE failures and the restart confound

Recovery. A retreat is a blind reverse at 0.5 m/s for 1.5 s, with no pose feedback. While exploring, it fires when eight consecutive decisions are all made within 0.25 m of the first of them, and the

Table 3: Paired contrasts. Δ is the first configuration minus the second. Brackets: house-clustered 95% bootstrap interval. W/L: pairs that succeed only under the first configuration / only under the second. "moved" keeps pairs in which both subtasks made at least one decision (the sensitivity analysis of Section 4.2).

contrast	pairs	n	ΔSR	ΔSPL	W/L, p
scene graph: full − blind					
`ssmg`	all	625	+0.125 [+.059,+.189]	+0.109 [+.065,+.154]	142/64, <.001
	moved	454	+0.090 [+.027,+.154]	+0.089 [+.045,+.134]	85/44, <.001
`ssmg-ukb`	all	625	+0.078 [+.019,+.138]	+0.075 [+.026,+.126]	127/78, .001
	moved	496	+0.034 [−.031,+.101]	+0.053 [−.003,+.110]	83/66, .190
record store: ukb − ssmg					
`full`	all	625	+0.005 [−.058,+.072]	−0.008 [−.054,+.037]	103/100, .888
	moved	542	+0.000 [−.060,+.060]	−0.013 [−.063,+.032]	85/85, 1.00
`blind`	all	625	+0.051 [−.003,+.106]	+0.026 [−.004,+.054]	109/77, .023
	moved	427	+0.021 [−.039,+.079]	+0.003 [−.026,+.032]	67/58, .474

Table 4: SR on a repeated category given the outcome of its first request. Successes over runs.

configuration	k=4 | k=1 ok	k=4 | k=1 fail	k=5 | k=2 ok	k=5 | k=2 fail
`ssmg-full`	53/86 (0.62)	6/39 (0.15)	44/62 (0.71)	8/63 (0.13)
`ssmg-ukb-full`	50/84 (0.60)	5/41 (0.12)	44/62 (0.71)	10/63 (0.16)
`ssmg-blind`	28/85 (0.33)	5/40 (0.12)	24/50 (0.48)	6/75 (0.08)
`ssmg-ukb-blind`	37/81 (0.46)	8/44 (0.18)	23/61 (0.38)	6/64 (0.09)

frontier the robot wedged on is retired. While approaching a chosen object, it fires when the route has been replanned more than five times without closing 0.25 m on the goal, or when no route remains. Each subtask allows two retreats, and the next stall ends it.

Table 5: Subtasks by number of retreats, all four configurations.

retreats	subtasks	succeeded	ended on a stall
0	1593	989 (62%)	0
1	206	89 (43%)	0
2	701	65 (9%)	575

Stall and fall sites. The site is the last point of the recorded path. Obstacles come from a grid of the simulator's colliders between 0.03 and 0.80 m, the same height band as the live grid, stored inflated by 0.20 m and eroded back for this analysis, which can close gaps narrower than the inflation. We call a site *near* if an obstacle lies within 0.35 m of the robot's center, half its 0.70 m standing length; the planner treats the robot as a 0.20 m disc, which covers its 0.155 m half-width but not its half-length. A near site is *mapped* if the live grid at the end of that subtask holds an occupied cell within 0.10 m of the nearest obstacle cell. Across path points sampled from repeats 2–5, 10.4% are near and the median clearance is 0.80 m. Of the 89 falls, 70 came with no earlier retreat in the subtask, so the retreat itself does not explain most falls. We inspected 16 sampled sites by eye against both grids. We logged no contact forces, so these classes describe where the robot stopped, not what touched it.

Map coverage. For `ssmg-full` at the end of each chain, 87.5% of obstacle boundary cells within 0.10 m of observed space have an occupied live cell within 0.10 m. The grid fuses depth from 0.4 to 5.0 m. We have no separate count for low furniture, because the height band is not split.

Near-misses. Of the 93 MOVE near-misses, 78 are approach stalls, 8 exploration stalls and 7 falls. Arrival is declared only when the robot reaches the goal point, and the 78 approach stalls ended a median 0.67 m short of it, inside the success radius. The agent does not check the success radius, so the stall rule ends the subtask first. Declaring arrival at these sites would add at most 93 successes (+0.037 SR pooled), split 28, 22, 25 and 18 in table order, so it would not change the comparisons.

Table 6: Where the robot stopped. Clearance is the median distance from the site to the nearest obstacle.

ending	sites	near, mapped	near, unmapped	open floor	clearance (m)
stall while exploring	401	346	36	19	0.21
stall on approach	174	149	14	11	0.22
first fall of a run	89	34	2	53	0.45

Restart sensitivity. Table 7 repeats the scene-graph contrast under each treatment of the blind subtasks that failed at startup. By subtask, `ssmg-blind` has 0, 2, 5, 8, 10 no-graph subtasks and `ssmg-ukb-blind` has 0, 3, 5, 4, 9, all after a restart. Under the worst case, the `ssmg` gain by subtask is $+0.01$, $+0.08$, $+0.10$, $+0.14$, $+0.10$. Across our five repeats, the single-repeat `ssmg` contrast ranges from 0.00 to $+0.26$, so a new control would need several repeats.

Table 7: Scene-graph gain in SR, full − blind. Worst case: every no-graph subtask counted as a blind success. Brackets: house-clustered 95% interval.

	all pairs	worst case	drop no-graph	drop 0-dec stalls	both decided
`ssmg`	$+0.125$	$+0.085$	$+0.112\,[+.047,+.178]$	$+0.087\,[+.021,+.153]$	$+0.090\,[+.027,+.154]$
`ssmg-ukb`	$+0.078$	$+0.045$	$+0.063\,[+.002,+.123]$	$+0.042\,[-.023,+.106]$	$+0.034\,[-.031,+.101]$

D The object record store

Hydra merges adjacent object nodes as new views arrive, reports merges only as aggregate counters, and drops the metadata of the absorbed node. Its ROS interface also broadcasts full graph snapshots (about 1 Hz), which replace the agent's in-memory graph, so metadata written into graph nodes is overwritten within seconds. The record store (UKB) therefore keeps object records beside the graph. The record of object j is the tuple

$$r_j = \langle\, \iota_j,\ h_j,\ d_j,\ q_j^{\text{first}},\ q_j^{\text{last}},\ c_j,\ V_j,\ M_j,\ \mathbf{x}_j \,\rangle.$$

Geometry, class, and room are read from the live node h_j.

Table 8: Record fields and how they merge when record r_b is folded into r_a, whose node survives. Ties: ι goes to the smaller id, d to the record seen first.

field	meaning	merged value
ι	record id, fixed at creation	id of the record seen first
h	Hydra node currently holding the object	h_a
d	appearance description from the VLM	d of the record with more observations
$q^{\text{first}}, q^{\text{last}}$	subtasks of the first and latest observation	$\min, \max$
c	decisions at which the node was present	$c_a + c_b$
V	view references, at most 8	$V_a \cup V_b$, first 8 kept
M	ids of records merged into this one	$M_a \cup M_b \cup \{\text{the losing } \iota\}$
$\mathbf{x}$	2D position of the node when last seen	$\mathbf{x}_a$

At every decision, the store creates a record for each new object node and updates q^{last}, c and $\mathbf{x}$ for each node it already tracks. When a tracked node disappears, Hydra does not report which node absorbed it, so its record is merged into that of the nearest live node within 1.0 m of its last position $\mathbf{x}$ (ties to the smaller node id), by the rules in Table 8. A record with no live node within 1.0 m stays in the store and is checked again at every later decision, so its history survives if a node later appears near its last position.

The store changes two planner inputs. (1) For each object candidate in the VLM's retrieval prompt, it adds the record's description and observation history. (2) In exploration, a frontier within 0.5 m of one that an earlier subtask was sent to is demoted by 0.5 belief units in the visit order. In a pilot

sweep, 44% of exploration goals in later subtasks lay within 0.5 m of an earlier subtask's goal, so the agent spent travel searching ground it had already covered. These frontiers are demoted rather than removed because the log records where the robot was sent, not whether it arrived.

Merges are frequent: in `ssmg-ukb-full`, 2317 of 5687 records (41%) were reattached, and 320 of 625 subtasks contained at least one reattachment. We did not measure whether reattachments chose the correct survivor. The store has no effect on navigation (Table 3), including on the repeated subtasks (0.436 against 0.444, 250 pairs). For the current set of experiments, the number of reattachments in a house does not predict the store's gain (Spearman $\rho = +0.33$ over 25 houses with the graph carried, -0.15 blind), though 25 houses is a small sample for this test. With the graph carried, the store changes the stage at which subtasks fail but not how many fail: `ssmg-ukb-full` has 16 fewer MOVE failures and 20 more SELECT failures than `ssmg-full`, and 306 failures against 309 (Table 1).

E Departures from the published SSMG-Nav

Table 9: Departures from the published SSMG-Nav [2].

Theirs	**Ours**	**Why**
VER point-nav policy	A* plus pure pursuit	Their policy assumes a Habitat cylinder and discrete actions
Attach at highest-confidence viewpoint pose	Attach at object centroid	Our object nodes carry no view buffer or confidence
Belief scale and τ both unstated	Per-source scale, $\tau = 1$	The planner weights beliefs exponentially, so their scale sets the trade-off against distance
Grounding-DINO and MobileSAM detections	Ground-truth semantic labels	Shared by all configurations; isolates memory from perception

Belief normalization. Each candidate has a belief b, a score for how likely the target is near it. For a keypoint, b is the VLM's reply, clamped to $[0, 1]$. For a frontier, b is the value map's CLIP cosine similarity between the images observed there and the goal text. The visit order weights each candidate by $\exp(b/\tau)$ against its travel distance, with $\tau = 1$. Across frontiers, the cosines lie between about 0.05 and 0.10, so two frontiers' beliefs differ by at most 0.05 and their weights by a factor of at most $\exp(0.05) \approx 1.05$. Distance then decides the order, and the planner visits the nearest frontier regardless of the target. We divide each source's beliefs by its spread before weighting: by 0.05 for the value map, the measured range of its cosines, and by 1.0 for the VLM, whose replies already span $[0, 1]$, so its beliefs are unchanged. Neither constant was tuned on navigation success.

Smaller departures, shared by all four configurations: continuous simulation steps replace the 500-step limit; CLIP ViT-L/14 [21] replaces BLIP-2 ITC [22] in the value map; geodesic distances are computed over skeleton edges; beliefs refresh when the skeleton changes rather than every step; and at most 150 frontier candidates are scored per decision.

F Houses and protocol

The 25 houses are from the ProcTHOR-10k validation split [14], converted to USD for Isaac Sim. Each has one floor, 155 to 399 m^2 of floor area, and all five goal categories from AI2-THOR [17]. Sinks and houseplants often occur more than once (1 to 4 sinks and 2 to 6 houseplants in the first ten houses), so the agent has to reach one valid instance among several. Doors are locked open and the exterior door is shut, so the simulator and the occupancy grid agree on every doorway. The robot spawns at the point of maximum free-space clearance. Category orders follow a balanced Latin square, so every category appears in every sequence position five times. Repeats 1–3 ran on one RTX 5090 machine and 4–5 on a second, with identical code, about 160 h in total.