INTERACTIVE NAVIGATION BEYOND REACHABILITY: BENCHMARK AND AGENTIC FRAMEWORK

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing embodied navigation tasks commonly assume that traversability and object states remain fixed during an episode. Yet reaching a goal may require opening a door to access a room and then opening a container to reveal a hidden target. We formulate interactive navigation beyond reachability, unifying channel-induced changes in spatial connectivity, container-induced changes in target visibility, and their sequential dependencies. We introduce **MultINav-Bench**, a 3K-episode ObjectGoal benchmark in MolmoSpaces spanning channel, container, and mixed tasks, with interaction-necessity annotations covering required and unnecessary episodes. Its data generation pipeline automates task construction across multiple task types. The evaluation protocol separately measures navigation success, interaction effectiveness, and cost. We further propose **MultINav-Agent**, a three-stage agentic framework built on a persistent, state-conditioned interaction graph. Semantic interactive perception grounds entities and interaction states; a vision-language model selects task-relevant actions based on expected changes in connectivity and target observability; and post-interaction verification updates the graph to guide subsequent navigation, exploration, or interaction. Experiments on MultINav-Bench show that MultINav-Agent consistently outperforms adapted baselines across channel, container, and mixed tasks

1 INTRODUCTION

PointGoal, ObjectGoal, and vision-and-language navigation have driven sustained progress in embodied navigation (Anderson et al., 2018a;b; Chaplot et al., 2020). These tasks test localization, mapping, search, and instruction following, but generally assume that traversability and object states remain fixed within an episode. This abstraction omits a basic property of indoor environments: interaction can change both the space a robot can traverse and the targets it can observe.

A closed door could disconnect the target room or force a long detour, while a closed refrigerator, cabinet, or drawer could conceal a nearby target. These effects can be sequential: opening a door permits entry into a room, where opening a container then reveals the target. The agent should therefore reason about how each interaction changes the environment and enables subsequent navigation, exploration, or interaction. Prior work provides foundations in semantic navigation, scene representation, and physical interaction (Savva et al., 2019; Kolve et al., 2017; Werby et al., 2024). For example, Interactive Gibson studies movable obstacles and the trade-off between path efficiency and physical disturbance (Xia et al., 2020). Beyond free-space recovery, the task-level challenge is to determine whether an interaction restores connectivity or reveals a target, how it enables subsequent actions, and when it is unnecessary. Therefore, we are motivated to formulate the problem of *multiple interactive navigation (MultINav)*, unifying *channel interactions* that change spatial connectivity, *container interactions* that change target visibility, and their sequential dependencies. This calls for a navigation-decision framework that jointly reasons about both types of interaction and their sequential dependencies. That is, the agent decides whether to interact, which object to select, when to act, and how to use the resulting state change to proceed toward the goal. Our motivation is shown in Fig. 1.

In this paper, we first instantiate the definition of MultINav as a benchmark, namely **MultINav-Bench**, a 3K-episode ObjectGoal benchmark in MolmoSpaces (Kim et al., 2026) spanning channel, container, and mixed tasks, with interaction-necessity annotations covering required and unneces-

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

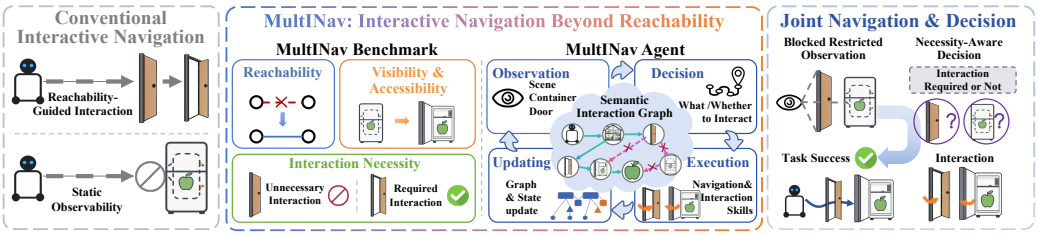

Figure 1: **Overview of MultINav.** Channel interactions change spatial connectivity, while container interactions change target visibility. MultINav-Bench covers both effects and their sequential dependencies, with required and unnecessary episodes. MultINav-Agent uses a persistent, state-conditioned graph to connect perception, action selection, and verified state updates.

sary episodes. Its data generation pipeline automates task construction across multiple task types using shared interaction-state specifications. The evaluation protocol separately measures navigation success, interaction effectiveness, and cost. To address the MultINav problem, we propose **MultINav-Agent**, a three-stage agentic framework built on a persistent, state-conditioned interaction graph. The graph retains observed rooms and entities together with their states and relations across observations. Semantic interactive perception grounds entities and interaction states in this representation. A vision-language model uses the graph and decision history to select task-relevant actions based on expected changes in connectivity and target observability. Post-interaction verification revises affected states and relations, guiding subsequent navigation, exploration, or interaction. Our contributions are summarized as follows:

- We formulate the problem of MultINav, which unifies channel-induced changes in spatial connectivity, container-induced changes in target visibility, and their sequential dependencies, paving the way for more comprehensive studies on interactive navigation.
- We introduce **MultINav-Bench**, a 3K-episode MolmoSpaces ObjectGoal benchmark spanning channel, container, and mixed tasks, with interaction-necessity annotations covering required and unnecessary episodes. As a versatile system, it provides a data generation pipeline that can automate task construction across multiple task types, and an evaluation protocol that separately measures navigation success, interaction effectiveness, and cost.
- We propose **MultINav-Agent**, a three-stage agentic framework built on a persistent, state-conditioned interaction graph that connects perception, action selection, and verified state updates. On MultINav-Bench, MultINav-Agent consistently outperforms adapted baselines across all task types.

2 Related Work

Embodied Navigation and Interactive Benchmarks. PointGoal, ObjectGoal, and vision-and-language navigation evaluate geometric, semantic, and language-specified goals under a fixed episode state Anderson et al. (2018a); Batra et al. (2020); Anderson et al. (2018b). Semantic-navigation methods exploit category maps, scene graphs, and semantic priors for target search (Chaplot et al., 2020; Ravichandran et al., 2022; Werby et al., 2024; Yin et al., 2024; Ginting et al., 2024); Habitat and AI2-THOR provide complementary environment foundations (Savva et al., 2019; Kolve et al., 2017).

This fixed-state abstraction becomes limiting when an action changes the relations used for semantic navigation, such as room connectivity or target exposure. As Table 1 shows, Interactive Gibson, Interactive Visual Navigation, and CaMP retain a navigation endpoint while changing free space or path cost (Xia et al., 2020; Zeng et al., 2021; Wang et al., 2023); InterNav also evaluates door-mediated connectivity in PointGoal (Zhou et al., 2026). ALFRED and BEHAVIOR-1K instead use navigation to support household completion (Shridhar et al., 2020; Li et al., 2024a). Built on MolmoSpaces (Kim et al., 2026), MultINav-Bench retains ObjectGoal while evaluating channel reconfiguration, target-container exposure, their composition, and required/unnecessary controls.

Interactive Navigation and Decision. NAMO introduced selecting movable obstacles to make a route feasible (Stilman & Kuffner, 2005). Interactive Gibson, Interactive Visual Navigation, and CaMP learn obstacle interactions for free-space recovery or path efficiency (Xia et al., 2020; Zeng

Table 1: Comparison of interaction scope and evaluated state effects. Chan., Cont., and Obj. denote channels, containers, and manipulable objects. Reported scale follows the respective publications.

Benchmark	Endpoint	Chan.	Cont.	Obj.	Evaluated state effect	Reported scale / envs.
Non-interactive benchmarks (fixed environment state)						
Habitat ObjectNav Batra et al. (2020)	ObjectGoal				Static	2.8M eps. / 90 envs.
MolmoSpaces Navigate-to Kim et al. (2026)	ObjectGoal				Static	2K traj. / 679 envs.
R2R Anderson et al. (2018b)	InstructionGoal				Static	21.6K instr. / 90 envs.
VLN-CE Krantz et al. (2020)	InstructionGoal				Static	R2R annot. / 90 envs.
Interactive benchmarks (environment-changing actions)						
Interactive Gibson Xia et al. (2020)	PointGoal	✓		✓	Free space + disturbance	106 envs.
Interactive Visual Nav. Zeng et al. (2021)	PointGoal			✓	Free space	15K eps. / 120 envs.
CaMP Wang et al. (2023)	PointGoal			✓	Free space	600K eps. / 12K envs.
InterNav Zhou et al. (2026)	PointGoal	✓		✓	Free space + connectivity	150 eval / 15 envs.
ALFRED Shridhar et al. (2020)	Household		✓	✓	Object state	8K demos / 120 envs.
BEHAVIOR-1K Li et al. (2024a)	Activity		✓	✓	Goal predicates	1K activities / 50 envs.
MultINav-Bench (ours)	**ObjectGoal**	✓	✓		**Connectivity + target exposure**	**3K eps. / 639 envs.**

et al., 2021; Wang et al., 2023); adaptive pushing, effect-oriented affordances, and joint perception-interaction-navigation extend this setting to unknown dynamics and richer action effects (Dai et al., 2024a; Wang et al., 2024; Li et al., 2020; Uppal et al., 2024). Task-and-motion planning complements policy learning by grounding discrete interaction choices in geometric feasibility (Kaelbling & Lozano-Pérez, 2011; Toussaint, 2015; Garrett et al., 2020). These methods establish that navigation can benefit from interaction, but generally tie the choice to obstacle clearance, path cost, or control feasibility. Language-grounded planners add language-level task decomposition, learned skill feasibility, and execution feedback (Ichter et al., 2023; Huang et al., 2023b; Singh et al., 2023; Huang et al., 2023a), while articulated-object systems provide reusable door-opening executors (Li et al., 2024b; Gupta et al., 2025). The remaining question is whether a candidate produces a goal-relevant navigation-state change. CoINS is the closest comparator, but MultINav-Agent selects candidates by their expected graph effect and verifies that effect before replanning. It therefore distinguishes restoring connectivity from exposing a target; in MultINav-Bench, the latter becomes task-level accessibility and is assessed with required/unnecessary and distractor controls.

Semantic Navigation and Structured Scene Representations. Dynamic and open-vocabulary scene graphs organize places, objects, relations, and traversability while supporting online construction, language queries, and task-relevant resolution (Rosinol et al., 2020; Hughes et al., 2022; Gu et al., 2023; Werby et al., 2024; Koch et al., 2024). They support graph-conditioned search and navigation (Ravichandran et al., 2022; Yin et al., 2024; Ginting et al., 2024); graph-based and language-guided planners further connect these representations to feasible actions and replanning (Jiao et al., 2022; Ray et al., 2024; Dai et al., 2024b). However, navigation graphs typically plan over relations observed during construction: even when the representation is updated online, action-induced changes in connectivity or target visibility are seldom treated as task-level state transitions that must be predicted and verified. Recent interactive graphs move toward such updates by modeling task state or articulated/container structure (Honerkamp et al., 2024; Yan et al., 2025; Rotondi et al., 2025; Ju et al., 2025; Yu et al., 2025; Büchner et al., 2026). MultINav-Agent specializes this interface to verified channel effects on connectivity and container effects on target visibility/accessibility under an ObjectGoal endpoint.

3 MultINav Benchmark

We introduce **MultINav-Bench**, a diagnostic ObjectGoal benchmark for navigation under interaction-induced state changes. Unlike static ObjectGoal navigation, an episode may require opening a passage door, exposing a target concealed inside a container, or sequentially composing both operations before navigation can succeed. MultINav-Bench freezes 3,000 domain-balanced *light* episodes in MolmoSpaces (1,000 channel / 1,000 container / 1,000 mixed across 639 houses) with explicit interaction-necessity labels, distractor objects, and oracle demonstration paths. It evaluates whether an agent reaches the goal while purposefully deciding when and what to interact with. Fig. 2 summarizes the interaction effects, construction pipeline, and dataset composition.

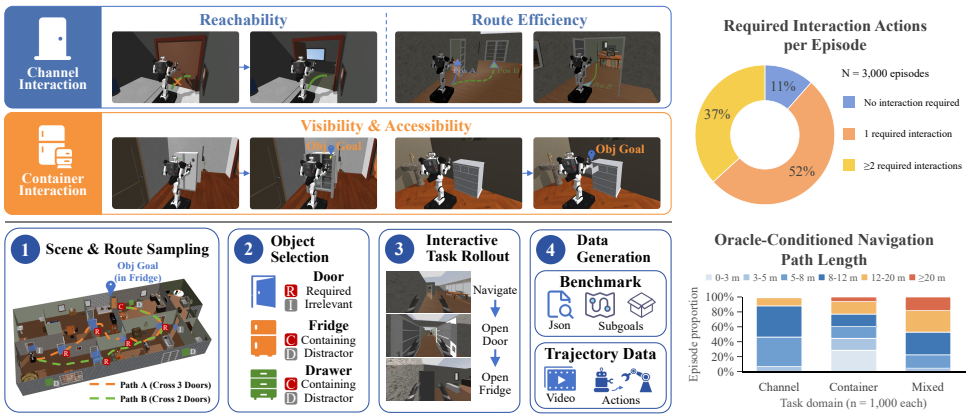

Figure 2: **Construction and composition of MultINav-Bench.** Top: channel interactions alter reachability and, in a route-shortening subset, route efficiency, whereas container interactions change target visibility and accessibility. Bottom: benchmark construction and data collection. Right: required-interaction counts and feasible-route lengths for the 3,000 episodes.

3.1 Task and Interaction Taxonomy

The embodied state is $s_t = (x_t, M_t, z_t)$, comprising the robot state x_t, the geometric/semantic map M_t, and the articulation states of interactive entities z_t. Actions combine navigation $\mathcal{A}_{\text{nav}}$ (updating x_t) and physical interaction $\mathcal{A}_{\text{int}}$ (updating z_t, thereby modifying traversability or target visibility). Each episode specifies the scene $\mathcal{S}_i$, initial states (x_i^0, z_i^0), goal object g_i, annotated necessary interactions $\mathcal{I}_i^\star$, an oracle plan $\pi_i^\star$, and a ground-truth necessity label $y_i \in \{\texttt{required}, \texttt{unnecessary}\}$ (formal specification in Appendix C, "Task Specifications and Collection Richness"). ObjectGoal is the frozen evaluation split.

Success vs. process. Terminal ObjectGoal success $S_i = \mathbb{1}[d_i^{\text{pass}} \wedge v_i^{\text{pass}}]$ requires stopping within planar distance d_i^{pass} with the target visible v_i^{pass} in the head camera. Physical interaction enables state changes but does not establish terminal success: opening a door without locating the target yields $S_i = 0$. ISR, IP, and normalized Total Cost separately measure required-effect progress, target-class physical effects, and navigation and interaction expenditure (Section 3.3). The benchmark exposes two perception modes: an image-based mode using raw visual observations, and a restricted-GT mode that supplies episode-local identities, categories, and spatial observations for currently visible entities. The primary comparison uses the restricted-GT mode to reduce perception and instance-association errors as a confound while keeping exploration, interaction selection, state updating, and terminal verification as agent decisions. In both modes, hidden objects, simulator joint metadata, and oracle interaction plans are withheld from the policy.

Channel, container, mixed. *Channel* interactions (hinged doors) govern spatial reachability, featuring critical closed doors (`required`), distractor doors (`unnecessary`), and composite layouts. *Container* interactions (refrigerators, cabinets, drawers) expose occluded targets, enabling the visibility required for navigation completion. *Mixed* episodes mandate sequential composition (e.g., traversing a closed door to reach a closed refrigerator). Across the 3,000 episodes, 2,654 are labeled `required` and 346 are `unnecessary` negative controls, allowing unnecessary interaction attempts to be measured without treating all exploration as an execution error.

3.2 Data Generation and Collection

Starting from ProcTHOR-10K houses and NavToObj seeds, candidates are synthesized via counterfactual reachability and visibility checks: closing a channel must sever the shortest route to the goal, while concealing a target inside a container must invalidate initial visibility until opened. Candidates pass articulation readback, target-identity verification, and oracle rollout validation before freezing 1,000 balanced episodes per domain.

Beyond the frozen *light* evaluation split designed for benchmark diagnostics, our pipeline supports a *full collection mode* recording multimodal expert demonstrations. For each verified rollout, full collection captures synchronized egocentric/allocentric visual streams, continuous base trajectories, 6D

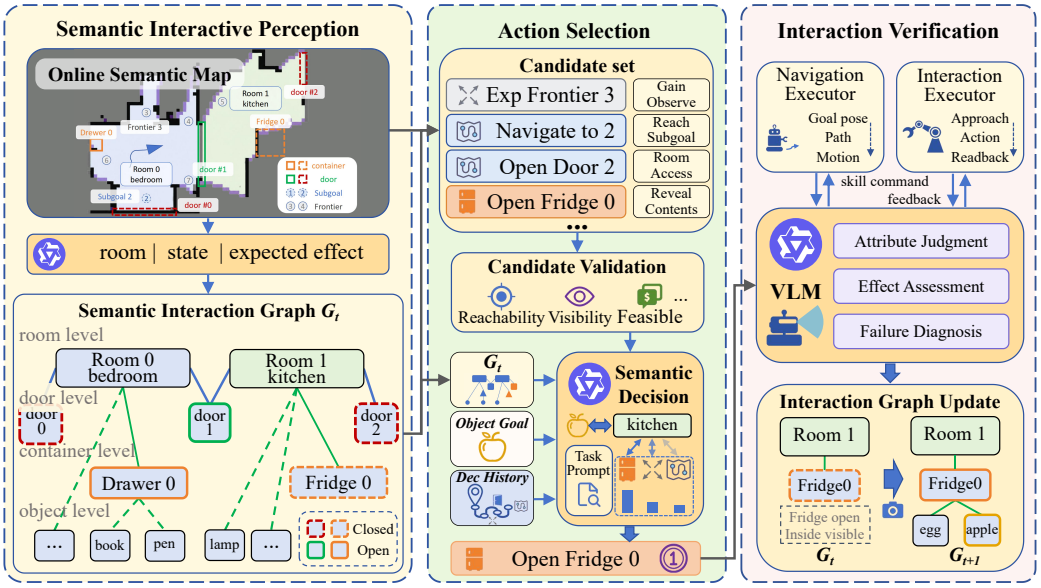

Figure 3: **MultINav-Agent framework**. Perception builds a persistent, state-conditioned interaction graph; candidate plans are formed; interaction proposals are filtered for reachability, observability, and freshness; a VLM scores admissible plans by task relevance; and post-interaction verification updates the graph to guide subsequent action selection.

end-effector targets, and articulation states formatted into unified HDF5 records. These trajectories provide an expert pretraining corpus for vision-language-action (VLA) models, while an accompanying interactive web inspector enables fine-grained visual auditing of base-manipulator motions. Table 1 compares MultINav-Bench with prior benchmarks.

3.3 Evaluation Metrics

We decouple terminal goal achievement from intermediate interaction efficiency using five complementary metrics:

- **SR (%)**: Terminal ObjectGoal success under standard distance and visibility thresholds.
- **SPL**: Success-weighted planar path efficiency relative to a feasible oracle route, reported on $[0, 1]$ using the same terminal navigation-success flag as SR.
- **ISR (%)**: Mean fraction of verified necessary interaction effects on required episodes; partial completion of a valid plan receives partial credit.
- **IP (%)**: Category-level Interaction Precision, the episode-mean fraction of attempts producing a verified physical effect in a target interaction class. The acted-on instance need not match the annotated instance; reopening after closure can receive credit again.
- **Total Cost**: All-episode mean of normalized operation cost, $\min((L_i + 0.3A_i + E_i)/30, 1)$ on success and 1 on failure, where E_i counts attempts without a qualifying physical effect. The normalization constant $B = 30$ is fixed before evaluation; it is not a step or time limit and does not stop rollouts (see Appendix E, "Metric Formulas").

4 MultINav-Agent Framework

MultINav-Agent maintains a persistent, state-conditioned interaction graph connecting semantic perception, action selection, and post-interaction verification (Fig. 3). Verified changes update the graph used to select subsequent navigation, exploration, or interaction.

4.1 Semantic Interactive Perception

The graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ represents rooms, interactive entities, objects, and the robot through connectivity, containment, and visibility relations. Perception associates entities across observations, while visual attribute inference estimates interaction class and coarse state (open, closed, or uncertain). Interactive nodes retain candidate operations, observation confidence, and recency alongside these state estimates.

The graph is *state-conditioned* because relation validity depends on the observed interaction state: a closed door blocks its mediated room connection, whereas a verified opening permits traversal. Opening a container permits inspection of its interior; target visibility is established only when the target is observed. Newly observed entities extend the graph, while previously observed entities persist beyond the current view. Their stored states and relations provide context for later decisions, with confidence and recency distinguishing historical estimates from fresh evidence (formal definitions in Appendix D, "Interaction Graph and State Updates").

4.2 Effect-Guided Action Selection

At each decision point, the graph and navigation map generate candidates for navigation, frontier exploration, and interaction. An interaction candidate specifies an entity, operation, and approach viewpoint. Eligibility checks use route feasibility, available visual evidence, and state reliability. These checks constrain the current executable candidates; remembered entities remain in the graph even when no interaction with them is currently eligible.

Each interaction candidate carries a coarse expected effect derived from its interaction class and graph relations: opening a channel may enable traversal, while opening a container may reveal its contents. This describes a possible change, without assuming an unseen target is present. A vision-language model receives the goal, relevant subgraph, candidate states and effects, route costs, and decision history. It ranks eligible actions by their expected contribution to reaching or observing the target, accounting for route prerequisites, plausible target locations, and previous attempts.

Navigation and exploration remain available when interaction offers little task progress. The selected candidate resolves to an executable goal or interaction specification; a structured reason code summarizes the model's task-level judgment.

4.3 Interaction Verification and Graph Updates

For a selected interaction, the agent reaches the approach viewpoint and grounds the operable region and operation from the current view. Navigation and interaction skills execute the resulting specification. Post-interaction verification reconciles fresh visual observations with execution feedback to assess the achieved state and expected graph effect (executor details in Appendix F, "Additional Implementation Notes"; visual interaction prompting is summarized in Appendix G).

Confident evidence updates the affected entity state; conflicting or occluded evidence is marked uncertain. Channel updates revise room connectivity, while container updates incorporate newly observed objects and their containment and visibility relations through perception. Failed or uncertain attempts remain in decision history to discourage immediate repetition. The updated graph regenerates candidates for subsequent navigation, exploration, or interaction.

This loop supports sequential interaction dependencies. Opening a door enables entry into a room; observations there can introduce a container candidate; verified container opening exposes objects whose observations support the next navigation decision. Each step updates the graph and available actions, composing connectivity and visibility changes during exploration.

5 Experiments

5.1 Experimental Setup

We evaluate on MolmoSpaces NavToObj and our MultINav-Bench for non-interactive and interactive navigation, respectively. MultINav-Bench contains 3,000 ObjectGoal episodes spanning 639 houses in total, with balanced Channel, Container, and Mixed task splits.

Evaluation scope. The benchmark provides both raw image-based perception and a restricted ground-truth (RGT) perception mode. The primary comparison uses the same RGT interface for both interactive methods: it supplies episode-local object IDs, categories, and spatial observations for currently visible objects. This reduces perception and instance-association errors as a confound when evaluating interaction-aware representations and decisions; agents must still explore, select interactions, and update their beliefs from observed outcomes. Hidden objects, simulator joint metadata, and oracle plans are withheld. Implementation details are provided in Appendix A, "Experimental Platform and Reproducibility," and Appendix F, "Additional Implementation Notes."

Interactive baseline and metrics. We compare MultINav-Agent (Ours) with a MolmoSpaces adaptation of MoMa-LLM (Honerkamp et al., 2024). Both MultINav-Agent and the MoMa-LLM adaptation use Qwen3.6-35B-A3B-FP8 for high-level decisions and share the same interaction executor. We select this model to support real-time decision-making during navigation. We report SR, SPL, ISR, IP, and Cost as defined in Section 3.3, with formulas in Appendix E, "Metric Formulas."

5.2 Main Results

Non-interactive NavToObj. Without interaction, our method achieves 53.0% SR on MolmoSpaces NavToObj (Table 2). The RING (Eftekhar et al., 2024) and DualVLN (Wei et al., 2025) leaderboard results are 48.0% and 21.0%; SPOC (Ehsani et al., 2024) and VLFM (Yokoyama et al., 2024) score 20.15% and 11.90% in local evaluations. All baselines execute valid commands; lower transfer scores stem from continuous control shifts lacking sliding contact feedback.

Interactive benchmark. Table 3 shows that MultINav-Agent achieves better aggregate values on all five metrics than the MolmoSpaces adaptation of MoMa-LLM. Ours completes 1,249/3,000 episodes, reaching 41.63% SR versus 10.53% (+31.10 percentage points), with SPL increasing from 0.0848 to 0.2903. ISR increases from 23.69% to 49.38%, IP from 27.59% to 46.79%, and normalized Cost decreases from 0.9309 to 0.7743.

Table 2: **MolmoSpaces NavToObj success** ([†]leaderboard).

Method	Tasks	SR (%) ↑	Domain
MultiNav-Agent (ours)	2,000	**53.0**	ProcTHOR
RING[†]	2,000	48.0	ProcTHOR
DualVLN[†]	2,000	21.0	Habitat
SPOC	2,000	20.15	ProcTHOR
VLFM	2,000	11.90	Habitat

Comparison across interaction types. Our MultiNav-Agent achieves 55.80%, 45.00%, and 24.10% SR on Channel, Container, and Mixed tasks, exceeding MoMa-LLM by 36.20, 34.70, and 22.40 percentage points, respectively. Channel ISR differs by only 5.81 points (60.09% vs. 54.28%), despite the much larger SR gap, underscoring the distinction between achieving an interaction effect and reaching the final goal. Container tasks show the largest ISR gap (46.70% vs. 2.00%), alongside higher target-class IP (38.91% vs. 2.37%). Mixed tasks remain the hardest for both methods: MoMa-LLM reaches 25.37% ISR but only 1.70% SR, while our method reaches 45.07% ISR and 24.10% SR. Partial interaction progress therefore remains insufficient for completing combined channel-container tasks.

Interaction-to-goal gap. ISR averages partial completion over required episodes, so it need not track terminal SR. For Ours, 393/654 required Channel episodes complete all necessary effects, and 297/393 also reach the goal (75.6%). Container and Mixed complete-effect counts are 466/1,000 and 247/1,000, of which 435/466 (93.3%) and 223/247 (90.3%) reach the goal. Completing the required interaction chain is thus a prominent difficulty for Ours on Container and Mixed tasks; Channel retains a larger navigation gap after the effects are achieved. Mixed ISR of 45.07% includes partial credit, whereas only 24.70% of episodes complete all necessary effects.

Efficiency and failure patterns. Ours has higher SPL and lower normalized Cost in every split. Nevertheless, both methods incur near-ceiling Cost on Mixed tasks (0.9493 for Ours and 0.9951 for MoMa-LLM). Since Cost is one for navigation failures, the 759 Mixed failures for Ours contribute 0.759 to its mean Cost. Similarly, SPL is gated by success. These aggregate improvements therefore reflect both completion and trajectory efficiency, rather than establishing shorter paths or cheaper execution conditional on success. Ours records exploration stalls on 516 Mixed episodes, compared with 304 Container and 158 Channel episodes, motivating further analysis of progress across interacting subgoals. These terminal labels do not establish a causal failure mechanism.

Table 3: **Interactive navigation results on MultINav-Bench**, grouped by method. Each method is evaluated on 3,000 episodes, with 1,000 per split. Each method's first row reports Overall, followed by task splits. Bold indicates the best Overall value for each metric.

	Navigation & Task		Interaction		Overall
Method / Split	**SR (%) ↑**	**SPL ↑**	**ISR (%) ↑**	**IP (%) ↑**	**Cost ↓**
MoMa-LLM	10.53	0.0848	23.69	27.59	0.9309
Channel	19.60	0.1758	54.28	40.61	0.8651
Container	10.30	0.0685	2.00	2.37	0.9324
Mixed	1.70	0.0101	25.37	39.81	0.9951
MultINav-Agent (ours)	**41.63**	**0.2903**	**49.38**	**46.79**	**0.7743**
Channel	55.80	0.4567	60.09	54.29	0.6517
Container	45.00	0.3008	46.70	38.91	0.7220
Mixed	24.10	0.1135	45.07	47.16	0.9493

Table 4: **Ablation results** on the same fixed 200-episode **Mixed subset**. Bold indicates the best value in each column.

	Navigation & Task		Interaction		Overall
Method	**SR (%) ↑**	**SPL ↑**	**ISR (%) ↑**	**IP (%) ↑**	**Cost ↓**
Full	**23.0**	0.107	41.8	**40.4**	0.961
Flat Object Memory	21.0	0.100	**43.8**	26.7	0.963
Greedy Selection	22.0	**0.134**	40.9	36.9	**0.933**
Perception-only Update	5.5	0.033	26.6	36.6	0.988

5.3 Ablation Study

All four variants use the same fixed 200-episode subset of the 1,000 Mixed tasks, spanning 165 houses. We sample proportionally without replacement by container class and oracle-restored reference path length, using benchmark metadata only for sampling. The episode list is frozen before evaluation without using policy outcomes; construction details appear in Appendix B.5, "Ablation Subset Construction."

Table 4 examines interaction memory, action selection, and state updating. Full achieves the highest SR (23.0%) and IP (40.4%), but the variants reveal different trade-offs rather than uniform gains across all metrics.

Interaction memory. Flat Object Memory reduces IP from 40.4% to 26.7% and SR from 23.0% to 21.0%, despite increasing ISR from 41.8% to 43.8%. The graph-based memory therefore shows its clearest advantage in interaction precision, not required-effect completion. This pattern is consistent with relational context helping select effective target-class interactions, while the higher ISR of the flat variant shows that partial progress alone need not translate into goal completion.

Action selection. Greedy Selection attains higher SPL (0.134 vs. 0.107) and lower Cost (0.933 vs. 0.961), with slightly lower SR (22.0%) and IP (36.9%). Full completes only two additional episodes out of 200, so these results do not establish a decisive success advantage for semantic selection. Instead, they expose a completion-efficiency trade-off: goal-conditioned selection improves the observed SR and IP, whereas the greedy variant performs better on aggregate efficiency. Since SPL and Cost also depend on success, these scores do not isolate efficiency conditional on completion.

State updating. Perception-only Update produces the largest SR decrease, from 23.0% to 5.5% (17.5 percentage points), alongside lower ISR (26.6%), lower SPL (0.033), and higher Cost (0.988). IP remains 36.6%, indicating that some interactions still achieve target-class effects without completing the task. Among the tested variants, this is the strongest evidence for retaining execution-informed graph updates when coordinating channel and container interactions. The smaller SR differences for memory and selection should be interpreted cautiously on this fixed subset, rather than as established statistically significant gains.

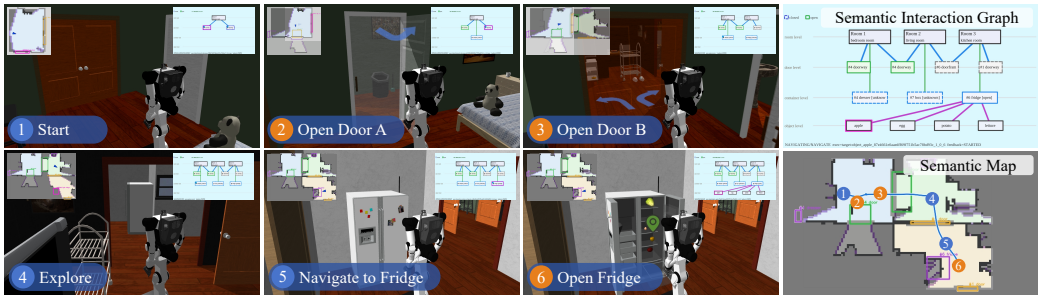

Figure 4: **Sequential interactions and graph updates.** The robot opens two doors to reach a refrigerator, then opens it to reveal the target. Insets show evolving semantic maps and interaction graphs; right: final graph and trajectory (blue: navigation/exploration; orange: interactions).

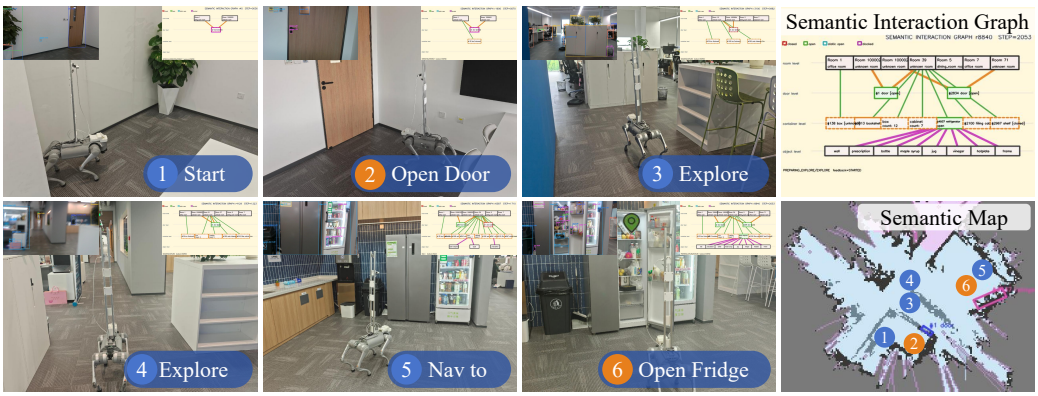

Figure 5: **Real-world navigation to find an apple with Go2.** The robot explores and navigates to a refrigerator, with human-assisted door and refrigerator opening. Insets show observations and interaction graphs; map markers indicate navigation (blue) and interaction (orange).

5.4 Case Studies

Fig. 4 illustrates an episode combining channel and container interactions: the robot opens two doors to expand reachability, navigates to the refrigerator, and opens it to expose the target. Per-step insets show how verified state updates guide continuous replanning.

5.5 Real-World Demonstration

Setup. We conduct the real-world demonstration on a Go2 quadruped in an indoor environment (Fig. 5). Door and refrigerator opening are performed with human assistance. Egocentric observations, intermediate interaction graphs, and the semantic map record the evolving scene representation throughout the rollout.

Task and rollout. The task is to find an apple. Starting near a closed door, the robot encounters a channel interaction: the door is opened with human assistance, enabling access to the adjacent space. The robot then explores the newly accessible area, extending its semantic map and interaction graph as rooms and objects are observed. It subsequently navigates to the refrigerator, which is opened with human assistance so that the robot can inspect its contents for the target apple. This sequence links two distinct state changes within a single object-search task: door opening changes spatial connectivity, while refrigerator opening exposes previously occluded contents. The intermediate graphs in Fig. 5 show the corresponding evolution of room, door, container, and object relations.

6 Conclusion

We formulated MultINav, unifying channel and container interactions within ObjectGoal navigation, and introduced MultINav-Bench (3,000 episodes) together with MultINav-Agent, whose persistent interaction graph with verified state updates achieves 41.63% SR versus 10.53% for the adapted baseline and transfers to a real-world Go2 demonstration. Extending to broader interaction types, learned perception, and end-to-end policy training are promising future directions.

AI USE STATEMENT

In this work, we used generative AI tools to support research ideation and execution, retrieve and identify relevant literature, draft sections of the paper, and edit and polish the manuscript. We did not use generative AI tools to generate synthetic datasets or prove mathematical claims. The authors reviewed and revised the AI-assisted material and retained responsibility for research decisions and the reported results. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

REFERENCES

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

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2018b.

Dhruv Batra et al. ObjectNav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020.

Martin Büchner, Adrian Röfer, Tim Engelbracht, Tim Welschehold, Zuria Bauer, Hermann Blum, Marc Pollefeys, and Abhinav Valada. Articulated 3d scene graphs for open-world mobile manipulation. *arXiv preprint arXiv:2602.16356*, 2026.

Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Adv. Neural Inform. Process. Syst.*, 2020.

Cunxi Dai et al. Interactive navigation with adaptive non-prehensile mobile manipulation. *arXiv preprint arXiv:2410.13418*, 2024a.

Zhirui Dai, Arash Asgharivaskasi, Thai Duong, Shusen Lin, Maria-Elizabeth Tzes, George J. Pappas, and Nikolay Atanasov. Optimal scene graph planning with large language model guidance. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14062–14069, 2024b. doi: 10.1109/ICRA57147.2024.10610599.

Ainaz Eftekhar, Rose Hendrix, Luca Weihs, Jiafei Duan, Ege Caglar, Jordi Salvador, Alvaro Herrasti, Winson Han, Eli VanderBilt, Aniruddha Kembhavi, Ali Farhadi, Ranjay Krishna, Kiana Ehsani, and Kuo-Hao Zeng. The one RING: A robotic indoor navigation generalist. *arXiv preprint arXiv:2412.14401*, 2024.

Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, Ranjay Krishna, Dustin Schwenk, Eli VanderBilt, and Aniruddha Kembhavi. SPOC: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2024.

Caelan Reed Garrett, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. PDDLStream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, volume 30, pp. 440–448, 2020. doi: 10.1609/icaps.v30i1.6739.

Muhammad Fadhil Ginting, Sung-Kyun Kim, David D. Fan, Matteo Palieri, Mykel J. Kochenderfer, and Ali-akbar Agha-Mohammadi. SEEK: Semantic reasoning for object goal navigation in real world inspection tasks. In *Proceedings of Robotics: Science and Systems*, 2024.

Qiao Gu et al. ConceptGraphs: Open-vocabulary 3d scene graphs for perception and planning. *arXiv preprint arXiv:2309.16650*, 2023.

Arjun Gupta, Michelle Zhang, Rishik Sathua, and Saurabh Gupta. Demonstrating MOSART: Opening articulated structures in the real world. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.033. URL https://www.roboticsproceedings.org/rss21/p033.html.

Daniel Honerkamp, Martin Büchner, Fabien Despinoy, Tim Welschehold, and Abhinav Valada. Language-grounded dynamic scene graphs for interactive object search with mobile manipulation. *IEEE Robotics and Automation Letters*, 9(10):8298–8305, 2024. doi: 10.1109/LRA.2024.3441495.

Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. VoxPoser: Composable 3d value maps for robotic manipulation with language models. In *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562. PMLR, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 2023b.

Nathan Hughes, Yun Chang, and Luca Carlone. Hydra: A real-time spatial perception system for 3d scene graph construction and optimization. *arXiv preprint arXiv:2201.13360*, 2022.

Brian Ichter et al. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 2023.

Ziyuan Jiao, Yida Niu, Zeyu Zhang, Song-Chun Zhu, Yixin Zhu, and Hangxin Liu. Sequential manipulation planning on scene graph. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022. doi: 10.1109/IROS47612.2022.9981735.

Yuanchen Ju, Yongyuan Liang, Yen-Jen Wang, Nandiraju Gireesh, Yuanliang Ju, Seungjae Lee, Qiao Gu, Elvis Hsieh, Furong Huang, and Koushil Sreenath. MomaGraph: State-aware unified scene graphs with vision-language model for embodied task planning. *arXiv preprint arXiv:2512.16909*, 2025.

Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1470–1477, 2011. doi: 10.1109/ICRA.2011.5980391.

Yejin Kim et al. Molmospaces: A large-scale open ecosystem for robot navigation and manipulation. *arXiv preprint arXiv:2602.11337*, 2026.

Sebastian Koch, Narunas Vaskevicius, Mirco Colosi, Pedro Hermosilla, and Timo Ropinski. Open3DSG: Open-vocabulary 3d scene graphs from point clouds with queryable objects and open-set relationships. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2017.

Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision*, pp. 104–120, 2020. URL https://arxiv.org/abs/2004.02857.

Chengshu Li, Fei Xia, Roberto Martín-Martín, and Silvio Savarese. HRL4IN: Hierarchical reinforcement learning for interactive navigation with mobile manipulators. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 603–616. PMLR, 2020. URL https://proceedings.mlr.press/v100/li20a.html.

Chengshu Li et al. BEHAVIOR-1K: A human-centered, embodied AI benchmark with 1,000 everyday activities and realistic simulation. *arXiv preprint arXiv:2403.09227*, 2024a.

Yu Li, Xiaojie Zhang, Ruihai Wu, Zilong Zhang, Yiran Geng, Hao Dong, and Zhaofeng He. Unidoormanip: Learning universal door manipulation policy over large-scale and diverse door manipulation environments. *arXiv preprint arXiv:2403.02604*, 2024b.

Zachary Ravichandran, Lisa Peng, Nathan Hughes, J. Daniel Griffith, and Luca Carlone. Hierarchical representations and explicit memory: Learning effective navigation policies on 3d scene graphs using graph neural networks. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9272–9279, 2022. doi: 10.1109/ICRA46639.2022.9812179.

Aaron Ray, Christopher Bradley, Luca Carlone, and Nicholas Roy. Task and motion planning in hierarchical 3d scene graphs. In *International Symposium on Robotics Research (ISRR)*, 2024.

Antoni Rosinol, Arjun Gupta, Marcus Abate, Jingnan Shi, and Luca Carlone. 3d dynamic scene graphs: Actionable spatial perception with places, objects, and humans. In *Proceedings of Robotics: Science and Systems*, 2020. doi: 10.15607/RSS.2020.XVI.079.

Dennis Rotondi, Fabio Scaparro, Hermann Blum, and Kai O. Arras. FunGraph: Functionality aware 3d scene graphs for language-prompted scene interaction. *arXiv preprint arXiv:2503.07909*, 2025.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *IEEE Int. Conf. Comput. Vis.*, 2019.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2020.

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. ProgPrompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Michael Stilman and James J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(4):479–504, 2005.

Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1930–1936, 2015.

Shagun Uppal, Ananye Agarwal, Haoyu Xiong, Kenneth Shaw, and Deepak Pathak. SPIN: Simultaneous perception, interaction and navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18133–18142, 2024.

Xiaohan Wang, Yuehu Liu, Xinhang Song, Beibei Wang, and Shuqiang Jiang. CaMP: Causal multi-policy planning for interactive navigation in multi-room scenes. In *Adv. Neural Inform. Process. Syst.*, 2023.

Xiaohan Wang, Yuehu Liu, Xinhang Song, Yuyi Liu, Sixian Zhang, and Shuqiang Jiang. An interactive navigation method with effect-oriented affordance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16446–16456, 2024.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*, 2025.

Abdelrhman Werby, Chenguang Huang, Martin Büchner, and Abhinav Valada. HOV-SG: Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation. In *Robotics: Science and Systems*, 2024.

Fei Xia, William B. Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2): 713–720, 2020.

Zhijie Yan, Shufei Li, Zuoxu Wang, Lixiu Wu, Han Wang, Jun Zhu, Lijiang Chen, and Jihong Liu. Dynamic open-vocabulary 3d scene graphs for long-term language-guided mobile manipulation. *IEEE Robotics and Automation Letters*, 10(5):4252–4259, 2025. doi: 10.1109/LRA.2025.3547643.

Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. SG-Nav: Online 3d scene graph prompting for LLM-based zero-shot object navigation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. VLFM: Vision-language frontier maps for zero-shot semantic navigation. In *IEEE Int. Conf. Robot. Autom.*, 2024.

Alan Yu, Yun Chang, Christopher Xie, and Luca Carlone. Pandora: Articulated 3d scene graphs from egocentric vision. In *British Machine Vision Conference (BMVC)*, 2025. arXiv:2603.28732.

Kuo-Hao Zeng, Luca Weihs, Ali Farhadi, and Roozbeh Mottaghi. Pushing it out of the way: Interactive visual navigation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

Kangjie Zhou et al. CoINS: Counterfactual interactive navigation via skill-aware VLM. *arXiv preprint arXiv:2601.03956*, 2026.

TECHNICAL APPENDIX

A EXPERIMENTAL PLATFORM AND REPRODUCIBILITY

Runtime environment. The simulation experiments use NVIDIA A100 GPUs with 80 GB of memory. The real-world demonstration uses one NVIDIA GeForce RTX 4090 with 48 GB of memory. The simulation software stack uses Python 3.11.15, MuJoCo 3.4.0, and PyTorch 2.7.1.

Evaluation configuration. The simulation uses the MolmoSpaces ObjectGoal protocol, with terminal success requiring both goal distance and head-camera visibility. Both compared methods receive RGB-D observations and the same restricted records for currently visible objects, and share the interaction executor. The reported operation-cost metric uses $\lambda = 0.3$, $\mu = 1$, and $B = 30$ (Section E); B is a normalization constant rather than a rollout limit.

B MULTINAV-BENCH DISTRIBUTION DETAILS

This appendix expands the light ObjectGoal split summarized in the main paper (3,000 episodes; 1,000 channel / 1,000 container / 1,000 mixed; 639 houses). All counts are taken from the frozen balanced manifest and quality reports. Primary claims in the main paper do not depend on reading this appendix.

B.1 DOMAIN, RECIPE, AND NECESSITY

Table 5: Domain recipes and necessity labels in MultINav-Bench (light).

Split / recipe	N	Notes
Channel total	1000	253 houses
single-path door closed	366	required
distractor doors closed	346	unnecessary
critical+distractor	288	required
Container total	1000	569 houses
Mixed total	1000	369 houses
Required (all)	2654	
Unnecessary (all)	346	channel only

Container and mixed episodes are labeled required under the construction-time initial closed state and its validated reference plan. This label describes the intended interaction dependency; terminal success is assessed separately from completion of the annotated effects. Unnecessary episodes are exclusively the channel distractor-door recipe. Ambient articulated objects outside the listed necessary set are not separate episode families; spurious interactions are penalized by Interaction Precision and Total Cost in the main metrics.

B.2 PATH LENGTH AND INTERACTION COUNT

Table 6: GT path length (m) and mean required-interaction count from the light manifest.

Domain	N	Path med	Path mean	Int. mean
Channel	1000	8.12	8.71	0.65
Container	1000	5.93	7.99	1.10
Mixed	1000	11.50	13.58	2.01
All	3000	8.69	10.09	1.25

Mixed episodes are longer on average because they chain a channel interaction and a container interaction. Channel interaction counts below 1 on average reflect the unnecessary distractor subset (zero required interactions).

B.3 TARGET-CATEGORY SKEW

ObjectGoal targets are not uniform over categories. The top categories in the balanced 3K split include egg (439; 14.6%), tomato (199; 6.6%), apple (193; 6.4%), lettuce (168; 5.6%), and potato (150; 5.0%). Long-tail categories (e.g., plate, soapbar, spraybottle) appear only a few times. We report this skew for transparency; main tables use split macro averages so that no single category dominates the headline comparison by construction, though residual imbalance remains.

Container interaction hardware types in the 3K light set are dominated by hinged container doors and sliding drawers (plus channel hinged doors in channel/mixed episodes). Exact type histograms are released with the manifest audit JSON.

B.4 PER-HOUSE MULTIPLICITY

Episodes span 639 unique houses. Per-domain house counts are 253 (channel), 569 (container), and 369 (mixed). Balancing applies soft per-house caps during collection; the frozen set still admits multiple episodes per house (empirical maximum 16 episodes in one house; high tail near 11 at the top 5% of houses by count). Train/eval separation is enforced at the house level in the collector configuration so that the same house does not appear in both partitions of a formal split.

B.5 ABLATION SUBSET CONSTRUCTION

We constructed a fixed 200-episode subset from the 1,000 mixed episodes in the benchmark, covering 165 houses. Every episode requires both channel and container interactions. Sampling preserves variation in container class and reference path length without using policy outcomes. Full and all three ablated variants use the same fixed subset. The subset is sampled without replacement from benchmark indices 2000–2999 with pseudorandom seed `20260926`. For auditability, the source benchmark and frozen selection-file SHA256 values are:

> Benchmark: `4021ca2bebd9c875ccc4df70c746d9ed7f2376d13247fa1b7e98f2e9690b219f`
> Selection file: `30264942a1caf203711b55442352e2a5b0e33ea701f05a3efffb03cca136c1d3`

B.6 SCORING ELIGIBILITY AND QUALITY GATES

Before formal aggregation, the evaluator applies a runtime goal-consistency gate: frozen oracle terminal waypoints must remain consistent with the live pose of the selected instance. Traces that fail are retained for diagnosis but marked scoring-ineligible and excluded from primary denominators. Schema audit on the balanced 3K reports 3,000/3,000 valid episodes, 0 duplicate case IDs, and 0 missing comparable GT path lengths in the offline health report; runtime eligibility is a separate online check.

B.7 DOMAIN-SPECIFIC COLLECTION GATES

All builders emit the shared episode schema, but use domain-specific acceptance tests. For *channel* episodes, the collector starts from a fixed Navigate-to scene-start-goal tuple, identifies doors associated with candidate room routes, and validates closed- and open-state path feasibility. Required episodes must recover a feasible goal route after the annotated door effect; unnecessary episodes retain a feasible route while closed distractor doors remain present. A small route-shortening pilot additionally retains cases in which both states are feasible but opening reduces route length; it is diagnostic rather than a balanced MultINav-Bench stratum.

For *container* episodes, the collector binds a specific rigid target to a moving fridge, cabinet, or drawer compartment, closes the relevant articulation, and verifies the target-compartment association. It then validates an interaction pose, the opening effect, and a terminal pose that satisfies the same distance-and-visibility predicate as Navigate-to. Other articulated objects remain in the scene as ambient distractors. For *mixed* episodes, the full ordered door-container rollout must pass; the post-door state alone is not terminal, and omitting either annotated effect must break the validated

chain. Across domains, failed state readback, inconsistent target identity, oracle rollout failure, duplicates, and split leakage cause rejection.

C Task Specifications and Collection Richness

The generator varies task specification and record richness independently. Only the ObjectGoal-light cell is frozen as MultINav-Bench; the other cells reuse the same interaction entities, object roles, expected effects, and oracle transitions.

Table 7: Three task specifications supported by the shared collector.

Task form	Stored goal specification
ObjectGoal	Target category/instance, distance and head-camera visibility success criteria; primary MultINav-Bench split.
PointGoal	Grounded planar goal (optional yaw) with interaction-aware route feasibility and path-delta annotations.
InstructionGoal	Text grounded to oracle steps and entity IDs, with hidden/partial/explicit interaction disclosure.

Light evaluation records. Light records formally specify each benchmark episode as

$$e_i = (\mathcal{S}_i, x_i^0, g_i, z_i^0, \mathcal{I}_i^\star, \pi_i^\star, y_i), \tag{1}$$

storing scene identifier $\mathcal{S}_i$, initial robot pose x_i^0, goal specification g_i, initial articulation states z_i^0, annotated task-relevant interaction set $\mathcal{I}_i^\star$, oracle navigation-interaction plan $\pi_i^\star$, ground-truth necessity label $y_i \in \{\texttt{required}, \texttt{unnecessary}\}$, and construction quality evidence. They are sufficient to reconstruct and score an episode without storing a policy rollout.

Full trajectory records. Full records contain the light schema plus timestamped synchronized visual observations (head and external views), executed base actions, 6D end-effector target poses, and rendered video from a chosen navigation/interaction policy. They may support learning, behavior analysis, or failure diagnosis, but are not required to score MultINav-Bench and are not assumed to train the evaluated policy.

PointGoal. Candidates are sampled on the robot-inflated occupancy map with start-goal distance bounds and require an open-state GT path. Comparing closed- versus open-state paths yields required (no closed path), beneficial (closed path longer by a margin), or unnecessary labels. Beneficial strata can be denser than in the MultINav-Bench ObjectGoal split because path-delta filtering is built into the sampler.

InstructionGoal. Layered generation over a grounded oracle plan supports rule, LLM, VLM, and graph-grounded text, with stored entity IDs, plan-step indices, disclosure level (hidden / partial / explicit), and optional VLM evidence frames. Instructions are auditable annotations rather than free text detached from the interaction graph.

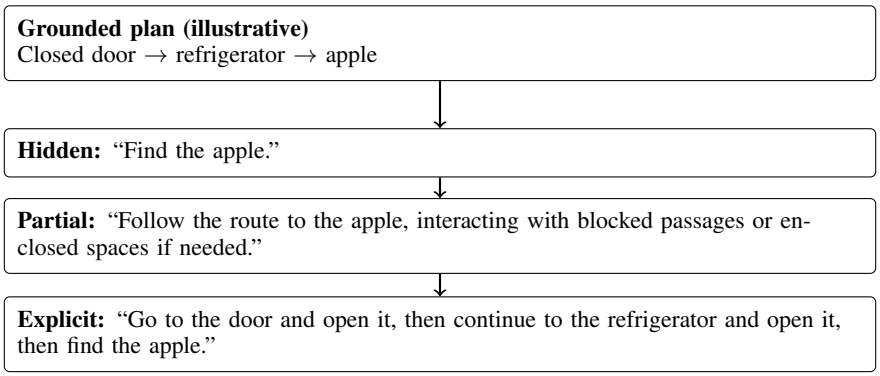

Figure 6: Illustrative InstructionGoal disclosure levels generated from one grounded interaction plan. These are task-specification examples, not results from a frozen InstructionGoal evaluation split.

D Interaction Graph and State Updates

Let $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t(z_t))$ be the interaction graph. Its nodes encompass rooms $\mathcal{R}$, interactive entities $\mathcal{U}$, and candidate objects $\mathcal{O}$; each entity $u \in \mathcal{U}$ maintains a state $z_t(u) \in \{\texttt{open}, \texttt{closed}, \texttt{uncertain}\}$. The mediated room-edge weight and container visibility relation are

$$w_t(R_i, R_j \mid u_{\text{door}}) = \begin{cases} d(R_i, R_j) & \text{if } z_t(u_{\text{door}}) = \texttt{open}, \\ \infty & \text{if } z_t(u_{\text{door}}) \in \{\texttt{closed}, \texttt{uncertain}\}, \end{cases} \tag{2}$$
$$\text{Vis}_t(o \mid u_{\text{cont}}) = \mathbb{1}[z_t(u_{\text{cont}}) = \texttt{open} \wedge o \in \text{FOV}_t],$$

where w_t governs door-mediated traversal and Vis_t governs exposure of an object inside a container. An uncertain door is conservatively blocked until its state is verified; an uncertain container does not establish visibility. The graph is persistent across observations: an entity that leaves the current field of view is retained with its last state, confidence, and observation time. A later observation is associated with an existing entity when the instance evidence is compatible; otherwise it creates a new candidate entity. Fresh evidence replaces stale attributes, while contradictory evidence reduces confidence and can move the state to `uncertain`. Absence from a new view is therefore not treated as disappearance, closure, or negative evidence. Candidate edges are regenerated from the current map and graph state, while the underlying entity history remains available to action selection.

For an interaction targeting $u^\star$ with requested state z_{target} and post-interaction observation o_{t+1}, the verifier updates the entity by

$$z_{t+1}(u^\star) = \begin{cases} z_{\text{target}} & \text{if post-action evidence verifies the requested state,} \\ \texttt{uncertain} & \text{if evidence is conflicting, partial, or occluded,} \\ z_t(u^\star) & \text{if execution fails or times out without observed change,} \end{cases} \tag{3}$$

The verifier requires confidence at least τ_{verify} that the observed state matches z_{target}; confidence in a different state cannot justify writing the requested state. A verified effect takes precedence over a failed macro acknowledgement, while partial effects retain uncertainty until the resulting state can be established. The graph update is

$$\mathcal{G}_{t+1} = \Phi(\mathcal{G}_t, z_{t+1}(u^\star), o_{t+1}) = \big(\mathcal{V}_t \cup \mathcal{V}_{\text{new}}(o_{t+1}),\ \mathcal{E}_t(z_{t+1}) \cup \mathcal{E}_{\text{new}}(o_{t+1})\big), \tag{4}$$

reweighting door-mediated room edges or adding newly exposed objects and their containment and visibility relations after a container is opened.

E Metric Formulas

Let N be the number of scoring-eligible episodes in the aggregation scope, $S_i = \mathbb{1}[d_i^{\text{pass}} \wedge v_i^{\text{pass}}]$ the verified ObjectGoal success indicator, L_i the executed planar base path length, A_i all interaction

attempts, and E_i attempts without any qualifying physical effect, counted at most once per attempt. A verified partial macro effect is not an error, even if the macro reports failure; an effect in a non-target class is likewise not automatically an error. For the restricted ROS policy, success additionally requires a completion claim verified against published target evidence and the benchmark distance criterion.

SR. $\mathrm{SR} = \frac{1}{N}\sum_i S_i$. Interaction completion is neither necessary nor sufficient in this definition. An episode can satisfy the terminal goal through an alternative observed route or a visibility change without completing every annotated reference effect. Such cases retain their construction-time necessity label and are scored by the terminal goal predicate rather than being retroactively relabeled.

SPL.

$$\mathrm{SPL}_i = S_i \frac{L_i^{\mathrm{ref}}}{\max(L_i^{\mathrm{ref}}, L_i)}, \qquad \mathrm{SPL} = \frac{1}{N}\sum_i \mathrm{SPL}_i,$$

where L_i^{ref} is a feasible reference that permits necessary interactions (oracle plan), not the initial static shortest path. Failed episodes contribute zero; successful zero-reference, zero-motion episodes contribute one. Missing or invalid reference lengths make the aggregate SPL unavailable rather than silently changing its denominator. Both L_i^{ref} and L_i measure planar base motion only; interaction duration, attempt count, and interaction cost are excluded from SPL and handled by IP and Total Cost.

ISR. Let R_{ip} be the required interaction IDs in valid plan p for episode i, and G_i the IDs with verified required effects. On required episodes,

$$\mathrm{ISR}_i = \max_{p:\, R_{ip}\neq\emptyset} \frac{|G_i \cap R_{ip}|}{|R_{ip}|},$$

and

$$\mathrm{ISR} = \frac{1}{|\mathcal{D}_{\mathrm{required}}|} \sum_{i\in\mathcal{D}_{\mathrm{required}}} \mathrm{ISR}_i,$$

with unnecessary episodes excluded. A required episode with no valid nonempty plan makes aggregate ISR unavailable. A required ID earns credit at most once per episode, including when its verified local effect occurs inside a failed macro. Opening requires the terminal 0.8 threshold or a verified transient opening. ISR does not independently test passage or target visibility after each effect; complete-plan success is a separate binary diagnostic. Thus a mixed episode may receive partial ISR even if the full chain and final navigation goal fail.

Target-Class IP. For episode i, let $\mathcal{C}_i$ denote the set of annotated interaction classes, each specified by an interaction domain (channel or container) and an object category. If A_i is the number of interaction attempts and V_i^{class} is the number of attempts producing at least one verified physical effect on an object whose class is in $\mathcal{C}_i$, let $U_i = 1$ only if the benchmark declares interaction unnecessary, and $U_i = 0$ otherwise. We define

$$\mathrm{IP}_i = \begin{cases} V_i^{\mathrm{class}}/A_i, & A_i > 0, \\ U_i, & A_i = 0, \end{cases} \qquad \mathrm{IP} = \frac{1}{N}\sum_{i=1}^{N} \mathrm{IP}_i. \tag{5}$$

The interacted object need not be the particular instance in the reference plan; multiple successful effects in one action count as one attempt. An open effect must cross from below 0.8 to at least 0.8 during the attempt, including verified transient drawer opening. Reopening after closure can earn credit again; an effect-free repeated request cannot. A partially successful macro can earn one IP credit even if its overall acknowledgement reports failure. Interactions with other classes remain in the denominator, reflecting exploration under incomplete information, but do not count as errors solely because they are outside $\mathcal{C}_i$.

Total Cost. The operation cost and normalized episode score are

$$C_i^{\mathrm{op}} = L_i + \lambda A_i + \mu E_i, \qquad \mathrm{Cost}_i = \begin{cases} \min(C_i^{\mathrm{op}}/B, 1), & S_i = 1, \\ 1, & S_i = 0, \end{cases} \tag{6}$$

and $\text{TotalCost} = N^{-1} \sum_i \text{Cost}_i$. The default protocol freezes $\lambda = 0.3$, $\mu = 1$, and $B = 30$ before evaluation. B is a shared weighted operation-cost ceiling, not a rollout stopping rule. Failures and successes over budget both score one. Successful effects outside target classes pay the ordinary attempt cost without the additional error charge. The unnormalized operation cost remains available as a diagnostic. Results with different metric definitions, budgets, or weights are not pooled.

Aggregation. Each method's primary row averages scoring-eligible completed episodes; indented rows use the channel, container, and mixed subsets. Overall is not an unweighted average of split means. IP first computes a ratio within each episode and then averages those ratios, rather than pooling attempts. ISR includes required episodes only.

Split-level full-model diagnostics. Ours reaches 60.09% ISR on required Channel episodes and 46.70% on Container episodes. Container IP is 38.91%, whereas Mixed IP is 47.16% despite 24.10% navigation SR. These quantities measure different outcomes: ISR credits completed required effects, IP credits target-class effects among attempted interactions, and SR requires the final distance and visibility conditions. Mixed Cost is 0.9493: 759/1,000 Mixed episodes fail navigation, and the metric assigns a cost of one to failures.

The main results table reports scores for 3,000 Ours episodes, with 1,000 per split. ISR uses 2,654 required episodes (654 Channel and 1,000 each Container and Mixed). The MoMa-LLM adaptation also reports scores for 3,000 scoring-eligible episodes, 1,000 per split, with the same ISR denominators and $B = 30$ cost ceiling. Its Channel, Container, and Mixed SRs are 19.60%, 10.30%, and 1.70%; Mixed ISR is 25.37% and Mixed IP is 39.81%, illustrating the distinction between partial physical effects and terminal navigation success.

Ablation evaluation. The ablation study compares Full with Flat Object Memory, Greedy Selection, and Perception-only Update on the same fixed 200-episode Mixed subset (Section B.5). Full is the reference within this evaluation, with 23.0% SR, 0.107 SPL, 41.8% ISR, 40.4% IP, and 0.961 Cost; it is reported separately from the main benchmark aggregate. Flat Object Memory has the highest ISR (43.8%), while Greedy Selection has the highest SPL (0.134) and lowest Cost (0.933). Full has the highest SR and IP. Perception-only Update yields 5.5% SR, highlighting the largest observed completion gap among these variants.

Target visibility gain after opening a container is an optional diagnostic, not a primary construction field.

F Additional Implementation Notes

Graph observations and attributes. Both MultINav-Agent and the MoMa-LLM adaptation use the same front-end perception and the same restricted-GT observation interface in the primary benchmark comparison. The interface provides episode-local IDs, semantic categories, spatial observations, 2D masks or boxes, and 3D boxes for visible entities; it does not expose hidden entities, joint metadata, or oracle plans. The benchmark also supports an image-based raw-perception mode, but that mode is not mixed with the restricted-GT comparison. Instance labels, segmentations, and bounding boxes initialize and associate observed entities. Interactive nodes store the interaction class, coarse state, candidate operation, operable-part description, confidence, and observation recency. Visual attribute inference refines these records from the current view. The coarse candidate effects distinguish channel traversal from container-content revelation; they do not establish the presence of an unobserved target.

Shared simulation interaction executor. Both interactive methods use the same executor in the primary evaluation. The evaluator applies force-driven transitions to the selected articulated joints, with a kinematic completion fallback when a transition does not reach its requested target. Execution feedback is returned through the public interaction interface; simulator joint metadata are withheld from the policies. Agent graph updates maintain scene estimates, while the terminal evaluator independently checks goal distance and head-camera visibility.

Scope of aggregate comparisons. Aggregate benchmark comparisons characterize end-to-end behavior; by themselves, they do not isolate graph updates as the cause of a performance difference.

MoMa-LLM observation and decision interface. The reported MolmoSpaces adaptation receives head-camera RGB-D and restricted simulator-derived observations of currently visible objects. Each restricted record contains an episode-local opaque instance ID, a semantic category, a 2D bounding box and run-length mask, and a 3D axis-aligned bounding box (center, size, and frame ID). Simulator object names, joint identities and states, and oracle interaction plans are withheld from the policy. The adapter uses these observations and depth to maintain an occupancy map and a dynamic room-object graph. At high-level decision points, vLLM-served Qwen3.6-35B-A3B-FP8 receives a summary of known rooms, objects, frontiers, closed doors, and recent actions, then selects among `navigate(room_name, object_name)`, `go_to_and_open(room_name, object_name)`, `explore(room_name)`, and `done()`. The adapter converts the selected high-level action to navigation steps or a public opening request; the evaluator executes the physical interaction and verifies the terminal navigation goal. The model is queried at high-level decision points rather than at every simulation step. Its prompt summarizes the current room, nearby and previously discovered object categories, unexplored frontiers, closed doors, and recent action outcomes. Local navigation expands each selected high-level action into waypoints; an opening request is sent through the benchmark's public interaction interface, and its returned result updates the adapter's object-state belief.

Full rollouts use the same episode identifiers and success criteria as their light counterparts and may be recorded under different navigation/interaction policies (default force joint control), allowing trajectory data to be joined with the frozen evaluator manifest without changing benchmark membership.

Instruction disclosure. InstructionGoal supports hidden (goal only), partial (interaction may be needed), and explicit (ordered objects/actions) language as an experimental factor over the same grounded oracle plan.

G Vision-Language Model Decision Protocol and Prompts

This section details the vision-language model (VLM) architecture, input representation, prompt templates, structured output schema, and decision context employed by MultINav-Agent.

G.1 Model Configuration and Serving

MultINav-Agent uses `Qwen3.6-35B-A3B-FP8` for high-level decisions, served through an OpenAI-compatible chat-completions API.

Visual inputs are formatted as a composite RGB image containing:

- **Global egocentric view**: The full head-camera observation with the candidate interaction target bounded by an explicit visual outline, providing geometric and spatial context relative to the robot and room.
- **Actionable region inset**: A padded, normalized crop around the target entity ($x, y \in [0, 1]$), allowing fine-grained localization of actionable door handles, push panels, or drawer pull surfaces.

G.2 Subgoal Categorical Reason Codes and Subgraph Context

When evaluating candidate actions from the state-conditioned graph, the VLM receives serialized topological context containing:

1. **Room and Anchor Entities**: Discovered room partitions and up to 4 major anchor objects per room (e.g., `refrigerator`, `countertop`, `bed`, `table`).
2. **Candidate Portals and Containers**: Articulated doors and containers, their current states (z_t), and their mediated relational effects.
3. **Decision Reason Codes**: The VLM classifies the semantic justification for each candidate into standardized reason codes:

- `TARGET_VISIBLE`: The goal object is observed in line-of-sight; navigate to terminal observation pose.
- `UNLOCK_ROUTE`: Opening a door is necessary to restore traversal connectivity to an unvisited topological room.
- `REVEAL_TARGET_CONTAINER`: Opening a container (e.g., refrigerator or cabinet) to expose a hidden target object matching semantic category priors.
- `EXPLORE_TARGET_ROOM`: Navigating to a newly connected or unexplored room when direct target evidence is absent.

This structured protocol makes candidate choices inspectable; the selected action still requires execution and outcome verification.

G.3 High-Level Candidate Selection Prompt

At each high-level decision point, the model receives the target request, observed graph, current executable candidates, and recent action outcomes. The selection instruction asks it to distinguish observations from semantic priors, respect route prerequisites, avoid repeated failed actions, and return only IDs from the current candidate list. The compact prompt form is:

```
Find the requested target.  Rank up to three CURRENT
executable candidate IDs.  Prefer an observed target
goal; otherwise account for post-interaction traversal,
route prerequisites, plausible target containers, useful
frontiers, and recent failures.  Do not infer hidden
target locations or invent candidates.  Return only
{"ranked_ids":[...],"reason":"...",
"confidence":"low|medium|high"}.
```

The response is checked against the current executable candidate IDs before the chosen action is sent to navigation or interaction execution.