HarnessVLN: Unifying Training-Free Embodied Navigation through an Agent Harness

Yang Chen[1,2] Lirong Che[2,3] Zhenyu Huang[1] Wenbo Fu[1] Chuang Wang[2]
Xu Cao[2] Daqi Liu[2] Yuzhe Yang[2] Jian Su[2,*] Lan-Zhe Guo[1,*]

[1]Nanjing University [2]AGIBOT [3]Tsinghua University

[*]Corresponding authors

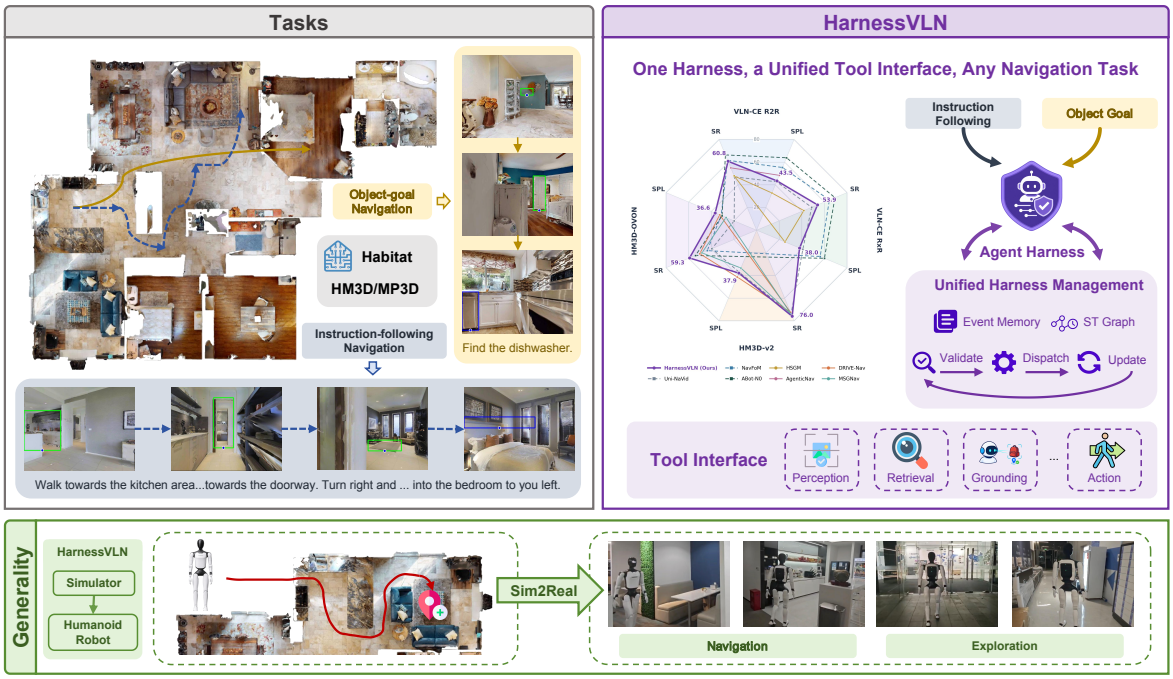

Figure 1. HarnessVLN: One Harness for Navigation across Tasks and Environments. A unified Agent Harness and tool interface support navigation tasks, delivering strong performance across four benchmarks and enabling deployment on a humanoid robot in real-world environments.

Abstract

Embodied navigation requires agents to ground instructions or object goals in spatial observations and translate plans into successful execution. As multimodal large language models (MLLMs) become increasingly capable, they offer stronger support for navigation without task-specific training; however, improved semantic reasoning alone does not ensure that proposed actions remain consistent with spatial evidence, task progress, and execution outcomes. We introduce HarnessVLN, a zero-shot, training-free framework that unifies instruction-following and object-goal navigation through a shared Agent Harness. The Harness coordinates perception, memory, and execution tools through a unified interface, validating planner proposals for evidential support, geometric feasibility, and subgoal consistency before dispatch. It jointly manages hierarchical event memory and a persistent Spatiotemporal Graph to track task progress, preserve spatial evidence, and contextualize failures. Structured execution feedback updates this shared state, guiding subsequent planning, recovery, and termination. Across R2R, RxR, HM3D-v2, and HM3D-OVON, HarnessVLN achieves success rates of 60.8%, 53.9%, 76.0%, and 59.3%, respectively, outperforming prior training-free state-of-the-art methods. Humanoid robot deployment further demonstrates its applicability to both navigation tasks in real-world environments. The project page is available at https://agibot-harnessvln.netlify.app/.

1 Introduction

Navigation is a foundational capability of embodied agents, enabling them to reach specified locations for further interaction with the physical world. Instruction-following and object-goal navigation represent two central forms of this capability, requiring agents to follow a route described in language or locate a specified object, respectively (Zhang et al., 2025a; Chu et al., 2026). Despite their different task definitions, both require agents to ground language in partially observed environments, continuously accumulate spatial knowledge, and ultimately reach the destination.

Training-based methods acquire these capabilities from task-specific demonstrations or large-scale trajectory data, mapping visual observations and language inputs to navigation actions (An et al., 2025; Zhu et al., 2025; Wei et al., 2026a; Yu et al., 2026). Although effective on specific tasks, extending these methods to new task formulations or environments requires additional training data for adaptation. Pretrained multimodal large language models (MLLMs) offer an alternative: leveraging their semantic knowledge and reasoning capabilities to interpret instructions, identify targets, and guide exploration without task-specific navigation training (Yokoyama et al., 2024a; Chen et al., 2026; Li et al., 2026a; Podgorski et al., 2025).

However, in existing MLLM-based navigation systems, the model primarily serves as a planner within a task pipeline: perception modules provide observations, the planner proposes subgoals, and downstream modules perform spatial grounding and action execution. This design relies heavily on the MLLM's semantic reasoning, while evidence retention, proposal validation, and failure handling lack a unified mechanism for coordinating their responsibilities. This exposes a central challenge: ensuring that the actions proposed by the MLLM advance the agent toward task completion as spatial knowledge and navigation progress continually evolve.

This challenge becomes more pronounced as navigation proceeds: execution failures can cause planning to diverge from the actual task state, recognizing a target does not establish arrival, and issuing an action does not establish subgoal completion. Addressing these gaps requires a runtime mechanism that jointly tracks spatial evidence, task progress, and execution outcomes. Agent harnesses provide a foundation for this coordination by organizing reasoning models, tools, memory, and environmental feedback within a unified framework (Lu et al., 2026; Zhang et al., 2026b). Applying this approach to navigation requires explicit spatial and temporal grounding: evidence must preserve where and when it was acquired, proposed targets must be reachable and relevant to the task, and execution outcomes must inform subsequent planning, recovery, and termination.

We introduce **HarnessVLN**, a zero-shot, training-free framework that supports instruction-following and object-goal navigation through a shared *Agent Harness*, as illustrated in Figure 1. The Harness coordinates perception, retrieval, grounding, navigation, recovery, and termination through a unified tool interface. The MLLM planner proposes semantic operations, which the Harness validates against supporting evidence, geometric feasibility, the active subgoal, and relevant failure history before dispatch. Structured execution feedback then updates the agent's state and informs the next decision. This shared interaction protocol accommodates both established navigation tasks through task-specific progress representations and completion criteria.

HarnessVLN maintains two complementary forms of memory to sustain coordination across decisions. Hierarchical event memory records decision context, subgoal progress, and execution events, while a persistent Spatiotemporal Graph (ST Graph) organizes spatial evidence, observation provenance, and associated failure records. Together, they enable the Harness to retrieve relevant experience, validate proposals against accumulated evidence, and guide recovery after failed attempts. A replaceable Navigation Executor converts validated targets into paths and motion commands, while stop validation checks whether the available semantic and spatial evidence supports task completion.

We evaluate HarnessVLN on VLN-CE R2R and RxR for instruction-following navigation and on HM3D-v2 and HM3D-OVON for object-goal navigation (Anderson et al., 2018; Ku et al., 2020; Yadav et al., 2023; Yokoyama et al., 2024b). Using the same framework across all four benchmarks, HarnessVLN achieves success rates of 60.8%, 53.9%, 76.0%, and 59.3%, respectively, exceeding the best compared training-free results by 5.8, 12.1, 1.6, and 9.1 percentage points. Deployment on a humanoid robot further demonstrates the framework's applicability to navigation in real-world environments.

Our main contributions are:

- We introduce HarnessVLN, a training-free framework that supports instruction-following and object-goal navigation through a shared Agent Harness and a unified tool interface.

- We develop a navigation runtime that combines hierarchical event memory and a persistent ST Graph with proposal validation and execution feedback, supporting evidence-grounded action, failure recovery, and verified termination.
- We demonstrate improved performance over prior training-free methods across four navigation benchmarks and validate the framework's real-world applicability through humanoid robot deployment.

2 Related Work

Training-Based Embodied Navigation. Training-based methods learn navigation policies from task-specific demonstrations or large-scale trajectory data (Chu et al., 2026; Xue et al., 2026; Wei et al., 2026b; Cheng et al., 2025; Zeng et al., 2026). NaVid (Zhang et al., 2024) predicts actions from monocular video and instructions, while Uni-NaVid (Zhang et al., 2025a) and NavFoM (Zhang et al., 2026a) extend unified policy learning across tasks and platforms. Although effective, these approaches rely on navigation-specific data and training, and adapting them to new task settings may require additional supervision. HarnessVLN requires no task-specific navigation training, using a general-purpose MLLM for semantic planning and a replaceable Navigation Executor for path planning and control.

Zero-Shot Embodied Navigation. Zero-shot navigation transfers semantic knowledge from pretrained language and vision models without task-specific navigation training. Object-goal methods combine open-vocabulary perception or commonsense reasoning with mapping and exploration (Gadre et al., 2023; Zhou et al., 2023; Yu et al., 2023; Yokoyama et al., 2024a; Zhang et al., 2025b), while instruction-following methods use language models for instruction decomposition, progress tracking, and target selection (Zhou et al., 2024; Long et al., 2025; Gao et al., 2026; Ding et al., 2026). These systems typically coordinate perception, memory, planning, and control through task-specific pipelines. HarnessVLN organizes these capabilities within a shared Agent Harness that maintains spatial evidence and task progress, validates proposed operations, and incorporates execution feedback to guide recovery and termination across both navigation tasks.

Agent Harnesses in Embodied Systems. Agentic embodied systems coordinate pretrained models, tools, memory, and execution feedback beyond monolithic policies (Shao et al., 2026; Liang et al., 2023; Huang et al., 2023; Fu et al., 2026; Ichter et al., 2023; Li et al., 2026b; Yang et al., 2025; Chen et al., 2025). Aspire (Lu et al., 2026) generates and repairs robot programs, while Harness VLA (Zhang et al., 2026b) coordinates reusable manipulation primitives with analytical tools. These systems primarily target manipulation. HarnessVLN extends the agent-harness paradigm to long-horizon navigation by combining tool-mediated execution, event memory, and a persistent ST Graph within a unified runtime for instruction-following and object-goal navigation.

3 HarnessVLN: A Harness-Mediated Embodied Navigation Agent

As shown in Fig. 2, HarnessVLN uses an *Agent Harness* to coordinate planning, memory, and tool execution. We introduce its decision process, tool interface, and event memory (Section 3.1), followed by the Harness-managed ST Graph for spatial retrieval and updates (Section 3.2), and grounded execution with evidence-based termination (Section 3.3).

3.1 Agent Harness

3.1.1 Harness-Mediated Decision Making

At decision step t, the Harness maintains the state

$$\mathcal{S}_t = \big(x,\, \omega_t,\, p_t,\, z_t,\, \mathcal{M}_t,\, \mathcal{G}_t\big), \tag{1}$$

where x denotes the task specification, namely a route instruction or target object category; ω_t is the current pose-aligned RGB-D observation and local geometric map; p_t is the agent pose; z_t represents task progress; $\mathcal{M}_t$ is a hierarchical event memory that records task-centric events and their execution contexts; and $\mathcal{G}_t$ is a persistent ST Graph that organizes environment-centric relations and evidence.

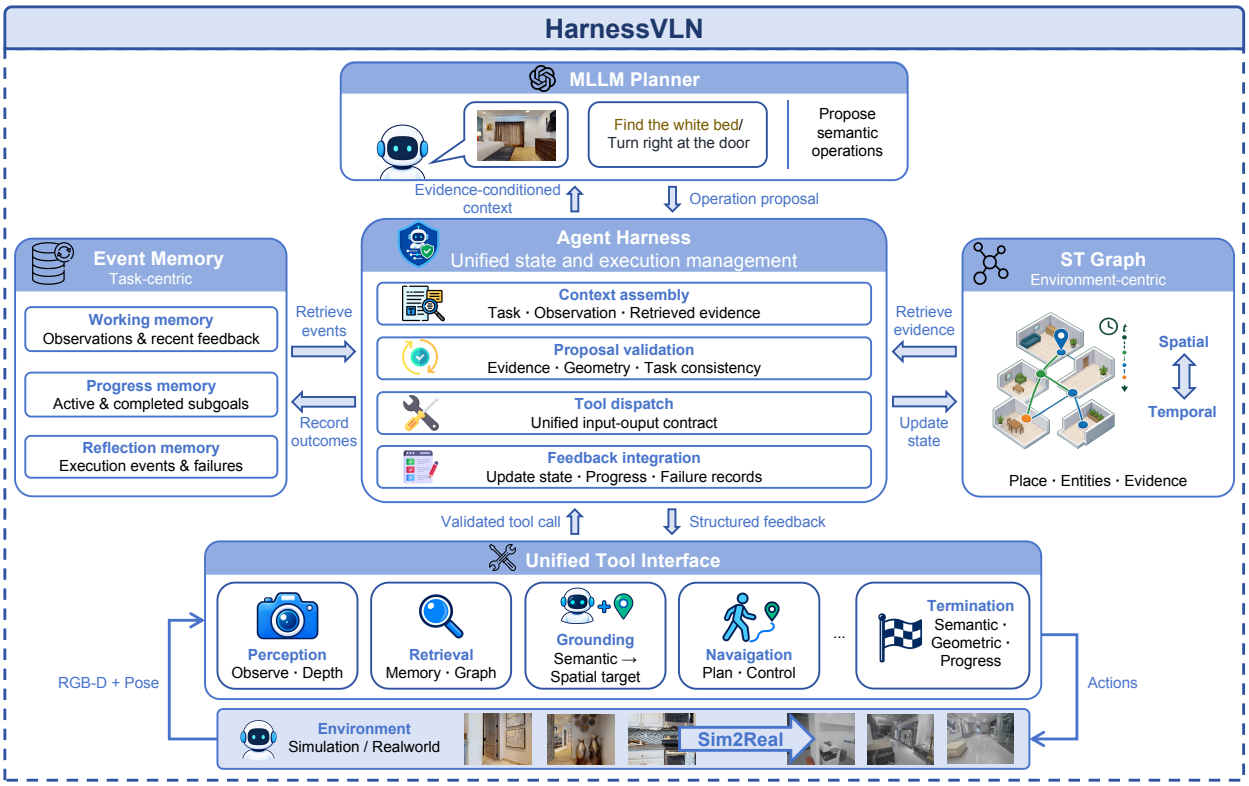

Figure 2. Overview of HarnessVLN. The Agent Harness coordinates MLLM planning, hierarchical event memory, and the ST Graph through a unified tool interface, forming a closed loop of context assembly, proposal validation, tool execution, and feedback integration.

HarnessVLN provides a unified information interface for instruction following and ObjectNav, yielding an evidence-conditioned decision cycle. For instruction following, z_t records the active route segment, satisfied motion constraints, and observed landmarks. For ObjectNav, it tracks explored regions, candidate target hypotheses, and their verification status. These task-specific variables affect planning and termination criteria without changing the underlying Harness protocol.

3.1.2 Unified Tool Interface

As summarized in Table 1, HarnessVLN exposes heterogeneous navigation capabilities through a unified tool interface. Perception tools collect directional or panoramic RGB-D observations; retrieval and grounding tools obtain task-relevant evidence from the current observation and the ST Graph and ground semantic subgoals to executable targets; navigation and recovery tools invoke the Navigation Executor or return the agent to a previously visited location; and the termination tool verifies task completion before stopping. This shared input–output contract allows individual tools to be replaced without modifying the MLLM Planner or the Harness protocol.

Before dispatching a tool call, the Harness validates its arguments, the provenance and recency of its supporting evidence, the geometric feasibility of the proposed target, and its consistency with the active subgoal. Each tool returns structured feedback containing the execution status, relevant measurements, and failure evidence. The Harness uses this feedback to update its internal state and the ST Graph, providing an updated context for the next planning decision.

3.1.3 Hierarchical Event Memory

The hierarchical event memory $\mathcal{M}_t$ is a task-centric record of the agent's interaction history. It stores the events required to interpret the current decision, track subgoal completion, and diagnose recent execution outcomes.

Working memory. Working memory stores the current observation, a bounded history of panoramic views, the active target hypothesis, and recent tool feedback. It retains only the short-term context needed for the next decision and thus

Table 1. Core tools exposed by HarnessVLN through a unified interface.

Tool	Role	Function
`observe`	Perception	Capture the current RGB-D observation and agent pose.
`observe_panorama`	Perception	Capture panoramic observations at predefined headings.
`retrieve_memory`	Retrieval	Retrieve task-relevant evidence from memory and graph.
`ground_target`	Grounding	Ground a subgoal to a candidate view and image region.
`query_depth`	Perception	Estimate the depth and uncertainty of a grounded region.
`navigate_to`	Navigation	Invoke the Navigation Executor for a validated target.
`backtrack`	Recovery	Return to a previously visited location and verify arrival.
`request_stop`	Termination	Validate task completion before terminating execution.

does not grow unboundedly with trajectory length.

Progress memory. Progress memory represents task decomposition and completion. Each subgoal has a unique identifier and one of four states: pending, active, completed, or blocked. At most one subgoal is active at a time, and completion must be supported by a pose-aligned observation or a successful tool result. This prevents the planner from advancing the task solely.

Reflection memory. Reflection memory records complete subgoal-specific execution events, including unreachable targets, inconsistent grounding, collisions, lack of progress, failed backtracking, and rejected stopping requests. Each record retains an event identifier, task context, location, timestamp, tool feedback, and supporting observation. Reflection memory remains the authoritative source for these event-level traces. When a failure has reusable spatial implications, the Harness writes only a lightweight annotation and a reference to the original event into the ST Graph. This separation enables failure-aware retrieval without duplicating the full execution history.

3.2 Harness-Managed Spatiotemporal Graph

The ST Graph $\mathcal{G}_t$ maintains an environment-centric abstraction of the agent's accumulated experience. Unlike event memory, which preserves task execution history, the graph consolidates reusable spatial structure and time-indexed evidence for replanning and recovery. It persists across subgoals while tracking when and where each observation was acquired, allowing the Harness to distinguish current evidence from stale or superseded hypotheses.

Persistent representation. We define the graph as

$$\mathcal{G}_t = \big(\mathcal{V}_t^P \cup \mathcal{V}_t^E, \mathcal{E}_t, \mathcal{T}_t\big), \tag{2}$$

where $\mathcal{V}_t^P$ and $\mathcal{V}_t^E$ are place and entity nodes, $\mathcal{E}_t$ contains typed spatial relations, and $\mathcal{T}_t$ stores timestamps and lightweight event annotations. Place nodes summarize visited locations, supporting observations, and traversable waypoints. Spatially compatible observations are merged into an existing place; otherwise, a new node is created. Consecutive places are connected by bidirectional NavigableTo edges, forming a persistent topology for forward navigation and backtracking.

Entity nodes represent ObjectNav targets and instruction-referenced landmarks through semantic aliases, spatial hypotheses, confidence estimates, and supporting views. ObservedFrom edges preserve the viewpoints and times at which an entity was observed, while Contains edges encode coarse place–entity associations. This provenance allows the Harness to retrieve both a target hypothesis and the evidence required to verify it.

Task-conditioned retrieval. Before each MLLM decision, the Harness ranks place nodes according to the active subgoal g_t:

$$s(v_i \mid g_t) = R(v_i, g_t) + \lambda_{\text{rec}} C(v_i) + \lambda_{\text{sal}} A(v_i) - \lambda_{\text{fail}} P(v_i, g_t), \tag{3}$$

where R, C, and A measure semantic relevance, recency, and spatial salience, respectively, and P penalizes applicable prior failures. The top-K places are augmented with nearby nodes and their associated entities, waypoints, spatial relations. This produces a compact retrieval set $\mathcal{R}_t$, allowing the graph to grow without increasing the MLLM context with trajectory length.

Algorithm 1 Harness-Mediated Navigation

Require: Task x and initial observation ω_0
Initialize $\mathcal{S}_0$, event memory $\mathcal{M}_0$, and Spatiotemporal Graph $\mathcal{G}_0$
while $\mathcal{S}_t$ is non-terminal **do**
  if ω_t is inconsistent with the current pose **then**
    $\omega_t \leftarrow$ OBSERVE
  end if
  $\mathcal{M}_t \leftarrow \textsc{RecordEvent}(\mathcal{M}_t, \omega_t)$
  $\mathcal{G}_t \leftarrow \textsc{ConsolidateGraph}(\mathcal{G}_t, \mathcal{M}_t)$
  $g_t \leftarrow \textsc{ActiveSubgoal}(z_t)$
  $\mathcal{R}_t \leftarrow \textsc{RetrieveExperience}(\mathcal{G}_t, g_t, p_t)$
  $\phi_t \leftarrow \textsc{AssembleContext}(\mathcal{S}_t, \mathcal{R}_t)$
  $c_t \leftarrow \pi_\theta(\phi_t)$
  $\widehat{c}_t \leftarrow \textsc{Validate}(c_t, \mathcal{S}_t)$
  $y_t \leftarrow \textsc{DispatchTool}(\widehat{c}_t)$
  $\mathcal{S}_{t+1} \leftarrow \textsc{UpdateHarnessState}(\mathcal{S}_t, y_t)$
end while
return verified terminal state and trajectory

Failure-aware updates. Failures with reusable spatial implications are attached to the corresponding place, entity, or relation as lightweight annotations. Each annotation records its subgoal condition, failure type, timestamp, applicability weight, and a reference to the complete event-memory record. Its relevance decreases when the subgoal changes or new evidence invalidates the failure, and increases when consistent failures recur. The graph therefore provides a spatial index for recovery, while event memory remains the complete record of task execution.

3.3 Grounded Navigation Execution

The MLLM proposes semantic navigation commands, whereas physical execution requires grounded targets and executable controls. HarnessVLN therefore exposes the Navigation Executor as a Harness-managed tool: the Harness grounds and validates each command, while the Executor performs path planning and local control.

Target grounding and execution. For each command c_t, the Harness resolves its target from the current observation and the ST Graph. A command is dispatched only if the target is supported by available evidence, geometrically reachable, and consistent with the active subgoal and applicable failure history. The Navigation Executor converts the validated target into an executable route and returns structured feedback, such as arrival, collision, unreachability, or lack of progress. This feedback updates the Harness state, event memory, and graph. Forward navigation and backtracking therefore follow the same validation–execution–update loop.

Evidence-based termination. Termination is mediated by the Harness. The MLLM may propose stopping but cannot directly issue an environment-level STOP action. A request is accepted only if

$$F_{\text{stop}} = F_{\text{semantic}} \wedge F_{\text{geometric}} \wedge F_{\text{progress}}, \tag{4}$$

where the three terms verify target identity, geometric validity, and task completion. For object-goal navigation, the target must be visually supported and lie within the stopping radius. For instruction following, the current evidence must support the final route segment, referenced landmark, and grounded endpoint. A rejected request is recorded in event memory and triggers further observation, target refinement, approach, or backtracking. Stopping is thus determined by embodied evidence rather than planner confidence alone.

Algorithm 1 summarizes the runtime procedure.

4 Experiments

We evaluate HarnessVLN on instruction-following and object-goal navigation to assess its effectiveness across task families without task-specific navigation training. We further conduct cumulative ablations to examine how hierarchical

event memory, the Spatiotemporal Graph, and stop validation affect success, efficiency, and termination behavior.

4.1 Experimental Setup

Instruction-following navigation. We evaluate on the continuous-environment R2R and RxR benchmarks in MP3D. We report Success Rate (SR), Success weighted by Path Length (SPL), Oracle Success Rate (OSR), normalized Dynamic Time Warping (nDTW), and Navigation Error (NE).

Object-goal navigation. We evaluate on the HM3D-v2 ObjectNav and HM3D-OVON validation splits, containing 1,000 and 3,000 episodes, respectively, in HM3D-Semantics v0.2 scenes. HM3D-v2 uses fixed object categories, whereas HM3D-OVON evaluates open-vocabulary targets.

4.2 Implementation Details

The agent receives synchronized RGB and depth observations at a resolution of 640×480 with a 79° horizontal field of view. GroundingDINO (Liu et al., 2024) and SAM (Kirillov et al., 2023) provide open-vocabulary target regions and segmentation masks, respectively. The Navigation Executor uses an FMM planner for planar motion and invokes NavDP (Cai et al., 2025) when stair traversal is required. For a fair comparison, we use `GPT-5.5` as the base model for instruction-following navigation and `GPT-5.6-luna` for object-goal navigation.

4.3 Main Experiments

Table 2 shows that HarnessVLN achieves the highest training-free SR on R2R and RxR. On R2R, it outperforms AgenticNav with the same `GPT-5.5` by 5.8 percentage points in SR and 7.7 points in OSR, reducing NE from 5.19 m to 4.01 m. Its OSR of 72.7% also exceeds all listed training-based methods. On RxR, HarnessVLN improves over HSGM by 12.1 points in SR and 12.9 points in SPL, reduces NE by 1.01 m, and maintains comparable nDTW (54.8% versus 54.9%), indicating improved completion and path efficiency with similar trajectory fidelity.

For ObjectNav, HarnessVLN achieves the highest training-free SR on HM3D-v2 at 76.0%, surpassing MSGNav by 1.6 points and approaching the best listed training-based result of 77.0% (Table 3). On HM3D-OVON, it achieves the highest SR and SPL among all listed methods at 59.3% and 36.6%, outperforming DRIVE-Nav by 9.1 and 4.0 points, respectively. Its SR also exceeds the strongest listed training-based result, achieved by ABot-N0, by 5.3 points. These results demonstrate strong task completion across both navigation families through a shared Harness protocol without task-specific navigation training, although efficiency gains vary across benchmarks.

4.4 Ablation Studies

We cumulatively enable hierarchical event memory (Mem.), the ST Graph (Graph), and stop validation (Stop) on fixed 100-episode subsets of R2R and HM3D-OVON, holding the base model and evaluation settings fixed within each task. As shown in Table 4, event memory improves SR by 8.0 and 7.0 percentage points, respectively, while graph-based retrieval adds 6.0 and 1.0 points and improves SPL on both subsets. Stop validation further increases SR by 4.0 and 2.0 points. It reduces the R2R OSR–SR gap from 15.0 to 13.0 points, but slightly increases the HM3D-OVON gap from 18.0 to 19.0 points and lowers SPL from 34.2% to 33.0%. Thus, its completion gains do not uniformly improve efficiency or termination behavior. Overall, the full Harness improves SR by 18.0 and 10.0 points over the base agent and reduces the OSR–SR gap on both tasks, supporting the combined design. These cumulative comparisons measure each component's incremental contribution given those already enabled.

5 Real-World Deployment

We deploy HarnessVLN on a humanoid robot. All tasks use the same Harness protocol, without training task-specific policies in the deployment environment.

5.1 Robot Platform and System Configuration

The AgiBot A3U is a full-size humanoid robot measuring 1.74 m in height. It is equipped with a 3D LiDAR and multiple RGB-D and fisheye cameras, with an onboard computing architecture based on NVIDIA Thor. We use the

Table 2. Comparison with existing supervised and training-free methods on the VLN-CE R2R and RxR val-unseen splits. NE is reported in meters, while all other metrics are reported as percentages.

Method	VLN-CE R2R Val-Unseen				VLN-CE RxR Val-Unseen			
	NE↓	OSR↑	SR↑	SPL↑	NE↓	SR↑	SPL↑	nDTW↑
Supervised Learning (Training-Based Methods)								
NaVid [RSS24]	5.47	49.1	37.4	35.9	8.41	34.5	23.8	–
Uni-NaVid [RSS25]	5.58	53.3	47.0	42.7	6.24	48.7	40.9	–
LookStep [EMNLP26]	5.34	55.9	49.7	45.3	6.89	46.9	39.9	–
NaVILA [RSS25]	5.22	62.5	54.0	49.0	6.77	49.3	44.0	58.8
ETPNav [TPAMI24]	4.71	65.0	57.0	49.0	5.64	54.7	44.8	–
InternVLA-N1 [ICLR26]	4.83	63.3	58.2	54.0	5.91	53.5	46.1	65.3
MapDream [ICML26]	4.59	64.4	59.8	54.4	4.96	59.4	49.2	–
NavFoM [ICLR26]	3.78	70.8	61.7	55.3	3.83	64.4	56.2	–
ABot-N0 [CVPR26]	3.78	70.8	66.4	63.9	3.83	69.3	60.0	–
Zero-Shot (Training-Free Methods)								
Open-Nav [ICRA25]	6.70	23.0	19.0	16.1	–	–	–	–
CA-Nav [TPAMI25]	7.58	48.0	25.3	10.8	10.40	19.0	6.0	13.5
InstructNav [CoRL24]	6.89	47.0	31.0	24.0	–	–	–	–
GC-VLN [CoRL25]	7.30	41.8	33.6	16.3	8.80	33.8	13.8	–
HSGM [CVPR26]	5.42	58.7	47.9	32.8	7.43	41.8	25.1	54.9
AgenticNav-GPT-5.5 [Arxiv26]	5.19	65.0	55.0	48.41	–	–	–	–
HarnessVLN-GPT-5.5 (Ours)	**4.01**	**72.7**	**60.8**	**43.5**	**6.42**	**53.9**	**38.0**	**54.8**

Table 3. Comparison with existing methods on HM3D-v2 and HM3D-OVON. Training indicates whether task-specific navigation training is required. All metrics are reported as percentages.

Method	Training	HM3D-v2		HM3D-OVON	
		SR↑	SPL↑	SR↑	SPL↑
Uni-NaVid [RSS25]	✓	73.7	37.1	39.5	19.8
FiLM-Nav [arXiv25]	✓	77.0	41.3	40.8	24.4
MTU3D [ICCV25]	✓	–	–	40.8	12.1
NavFoM [ICLR26]	✓	–	–	43.6	31.3
ABot-N0 [CVPR26]	✓	–	–	54.0	30.5
InstructNav [CoRL24]	✗	58.0	20.9	–	–
TANGO [ICRA25]	✗	–	–	35.5	19.5
VLFM [ICRA24]	✗	63.6	32.5	38.5	22.2
STEGNav [arXiv26]	✗	69.4	28.2	–	–
DRIVE-Nav [arXiv26]	✗	72.4	41.3	50.2	32.6
MSGNav [CVPR26]	✗	74.4	33.4	48.3	27.0
HarnessVLN (Ours)	✗	**76.0**	**37.9**	**59.3**	**36.6**

head-mounted stereo cameras as the primary visual sensors. Fast FoundationStereo estimates dense metric depth from synchronized stereo images. Qwen-3.8-27B serves as the planning model within the Harness, while a local path planner converts validated spatial goals into executable waypoints and motion commands.

Table 4. Cumulative ablation of HarnessVLN on fixed 100-episode subsets.

Harness Components			Instruction Following (R2R)				ObjectNav (HM3D-OVON)			
Mem.	Graph	Stop	SR ↑	SPL ↑	OSR ↑	Gap ↓	SR ↑	SPL ↑	OSR ↑	Gap ↓
✗	✗	✗	46.0	23.7	72.0	26.0	45.0	32.0	72.0	27.0
✓	✗	✗	54.0	33.0	73.0	19.0	52.0	32.4	72.0	20.0
✓	✓	✗	60.0	35.4	75.0	15.0	53.0	34.2	71.0	18.0
✓	✓	✓	**64.0**	**35.8**	**77.0**	**13.0**	**55.0**	**33.0**	**74.0**	**19.0**

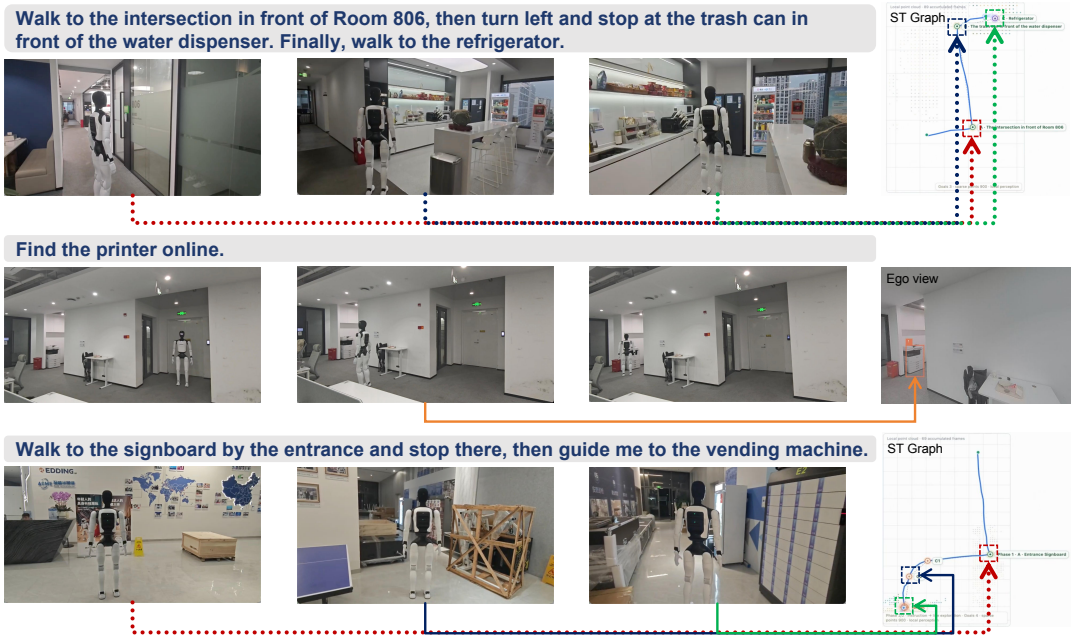

Figure 3. Real-world navigation examples: sequential instruction following, open-vocabulary object search, and a combined task involving route following and object search.

5.2 Real-World Navigation Experiments

We evaluate HarnessVLN in three real-world settings (Figure 3): instruction following through the intersection outside Room 806, followed by a left turn and a stop at the trash can near the water dispenser, before proceeding to the refrigerator; open-vocabulary search for a printer; and a combined task that first requires stopping at the entrance signboard and then locating a vending machine. Across these settings, the shared Harness tracks subgoal progress, grounds landmarks and targets in visual observations, and validates stopping requests using semantic and spatial evidence. The ST Graph retains landmark and target evidence across navigation stages, supporting continuity between route following and object search. HarnessVLN manages task progress, spatial evidence, and stop validation through a shared runtime protocol.

6 Conclusion

We presented HarnessVLN, a training-free framework that unifies instruction-following and object-goal navigation through a shared Agent Harness. By coordinating MLLM planning, hierarchical memory, spatial evidence, and validated tool execution, HarnessVLN bridges semantic reasoning and grounded action. Benchmark results demonstrate improved task completion across both tasks, while humanoid deployment shows real-world applicability. The current Harness relies on predefined orchestration and validation; future work will explore self-evolving mechanisms refined through interaction in open environments.

References

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. ETPNav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5130–5145, 2025.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

Wenzhe Cai, Jiaqi Peng, Yuqiang Yang, Yujian Zhang, Meng Wei, Hanqing Wang, Yilun Chen, Tai Wang, and Jiangmiao Pang. NavDP: Learning sim-to-real navigation diffusion policy with privileged information guidance. *arXiv preprint arXiv:2505.08712*, 2025.

Yang Chen, Hong-Jie You, Jie-Jing Shao, Xiao-Wen Yang, Ming Yang, Yu-Feng Li, and Lan-Zhe Guo. Re^2 Agent: Reflection and re-execution agent for embodied decision making. In *NeurIPS 2025 Challenge on Foundation Models for Embodied Agents*, 2025.

Yang Chen, Zhenyu Huang, Wenbo Fu, Danyang Peng, Shi-Yu Tian, Kun-Yang Yu, and Lan-Zhe Guo. STEGNav: Spatio-temporal event graph reasoning for multimodal lifelong object navigation. *arXiv preprint arXiv:2608.28279*, 2026.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Biyik, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation. In *Proceedings of Robotics: Science and Systems*, 2025.

Zedong Chu, Shichao Xie, Xiaolong Wu, Yanfen Shen, Minghua Luo, Zhengbo Wang, Fei Liu, Xiaoxu Leng, Junjun Hu, Mingyang Yin, et al. ABot-N0: Technical report on the VLA foundation model for versatile embodied navigation. *arXiv preprint arXiv:2602.11598*, 2026.

Hongyu Ding, Sizhuo Zhang, Ziming Xu, Jinwen Guo, Hongxiu Liu, Xingzhi Cheng, Zixuan Chen, Haifei Qi, Duo Wang, Hao Xu, et al. Uni-LaViRA: Language-vision-robot actions translation for unified embodied navigation. *arXiv preprint arXiv:2605.27582*, 2026.

Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. CaP-X: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CoWs on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23171–23181, 2023.

Maoguo Gao, Zejun Zhu, Zhiming Sun, Zhengwei Ma, Longze Yuan, Zhongjing Ma, Zhigang Gao, Jinhui Zhang, and Suli Zou. DRIVE-Nav: Directional reasoning, inspection, and verification for efficient open-vocabulary navigation. *arXiv preprint arXiv:2603.28691*, 2026.

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*, pages 540–562, 2023.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, pages 287–318, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

Alexander Ku, Peter Anderson, Roma Patel, Eugene Ie, and Jason Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 4392–4412, 2020.

Kailing Li, Tianwen Qian, Lijin Yang, Yuqian Fu, Jingyu Gong, Xiaoling Wang, and Liang He. Bridging the 2D-3D gap: A hierarchical semantic-geometric map for vision language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15243–15252, 2026a.

Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, et al. RoboClaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026b.

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as Policies: Language model programs for embodied control. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 9493–9500, 2023.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *Proceedings of the European Conference on Computer Vision*, pages 38–55, 2024.

Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. InstructNav: Zero-shot system for generic instruction navigation in unexplored environment. In *Proceedings of the 8th Conference on Robot Learning*, pages 2049–2060, 2025.

Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. ASPIRE: Agentic /skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

Stefan Podgorski, Sourav Garg, Mehdi Hosseinzadeh, Lachlan Mares, Feras Dayoub, and Ian Reid. Tango: Traversability-aware navigation with local metric control for topological goals. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2399–2406, 2025.

Jie-Jing Shao, Haiyan Yin, Yueming Lyu, Xingrui Yu, Lan-Zhe Guo, Ivor W. Tsang, James T. Kwok, and Yu-Feng Li. Lifting traces to logic: Programmatic skill induction with neuro-symbolic learning for long-horizon agentic tasks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

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. In *Proceedings of the 14th International Conference on Learning Representations*, 2026a.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, et al. StreamVLN: Streaming vision-and-language navigation via SlowFast context modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2026b.

Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Kuichen Quan, Wei Guo, Zedong Chu, Mu Xu, et al. OmniNav: A unified framework for prospective exploration and visual-language navigation. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Theo Gervet, John Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, et al. Habitat-Matterport 3D semantics dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4927–4936, 2023.

Zhejian Yang, Yongchao Chen, Xueyang Zhou, Jiangyue Yan, Dingjie Song, Yinuo Liu, Yuting Li, Yu Zhang, Pan Zhou, Hechang Chen, et al. Agentic Robot: A brain-inspired framework for vision-language-action models in embodied agents. *arXiv preprint arXiv:2505.23450*, 2025.

Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. VLFM: Vision-language frontier maps for zero-shot semantic navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 42–48, 2024a.

Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5543–5550, 2024b.

Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3MVN: Leveraging large language models for visual target navigation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3554–3560, 2023.

Kun-Yang Yu, Yingzhe Li, Hongyu Xu, Shi-Yu Tian, Zhi Zhou, Yang Chen, Ming Yang, Sheng Wang, Qing Yu, Lan-Zhe Guo, and Yu-Feng Li. LookStep: Efficient vision-language navigation with linguistic foresight and event driven memory. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing*, 2026.

Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, and Xing Wei. JanusVLN: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. In *Proceedings of the 14th International Conference on Learning Representations*, 2026.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. NaVid: Video-based VLM plans the next step for vision-and-language navigation. In *Proceedings of Robotics: Science and Systems*, 2024.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-NaVid: A video-based vision-language-action model for unifying embodied navigation tasks. In *Proceedings of Robotics: Science and Systems*, 2025a.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Embodied navigation foundation model. In *Proceedings of the 14th International Conference on Learning Representations*, 2026a.

Mingjie Zhang, Yuheng Du, Chengkai Wu, Jinni Zhou, Zhenchao Qi, Jun Ma, and Boyu Zhou. ApexNAV: An adaptive exploration strategy for zero-shot object navigation with target-centric semantic fusion. *IEEE Robotics and Automation Letters*, 10(11): 11530–11537, 2025b.

Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness VLA: Steering frozen VLAs into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026b.

Gengze Zhou, Yicong Hong, and Qi Wu. NavGPT: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, pages 7641–7649, 2024.

Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. ESC: Exploration with soft commonsense constraints for zero-shot object navigation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 42829–42842, 2023.

Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, et al. Move to understand a 3D scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8120–8132, 2025.

Appendix

1 Additional Experimental Settings

Benchmarks and model. We consider the same four benchmarks as the main text: VLN-CE R2R and RxR in Matterport3D, and HM3D-v2 and HM3D-OVON in HM3D-Semantics v0.2. RxR uses guide instructions in US English. A single pretrained MLLM serves all language and vision reasoning functions within an experiment, including task decomposition, navigation planning, target grounding, stop verification, and recovery. These functions use different prompts and share the same model. Following the main text, the base model is `GPT-5.5` for the main instruction-following experiments and `GPT-5.6-luna` for the main ObjectNav experiments. GroundingDINO, SAM, FMM, and NavDP provide the perception and navigation tools described in the main text; no task-specific fine-tuning is performed.

Simulation and context. Table 5 lists additional configuration details. A four-directional observation contains forward, left, behind, and right views at relative headings of 0°, 90°, 180°, and 270°. Acquiring a full panorama consumes twelve 30° rotations; these actions count toward the environment-step budget. Forward-view history is sampled every two steps and thinned to fit the image budget, including current observations. The simulator seed is zero; stochastic model responses can still introduce variation between runs.

Table 5. Additional configuration settings. Shared values apply to both task families. An environment step is one primitive simulator action.

Setting	Instruction following	ObjectNav
Maximum environment steps	1,000	500
Task success distance	3.0 m	1.0 m
Forward step / turn angle	0.25 m / 30°	
Sensor height / agent radius	0.88 m / 0.10 m	
Depth interval / map resolution	$[0.1, 5.0]$ m / 5 cm	
History interval / image budget	2 steps / 50 images per request	
Place / entity merge radius	0.75 m / 1.00 m	
Retrieval seeds / place limit	3 / 5	
Recency / salience weights	0.3 / 0.3	
Failure penalty / penalty cap	0.5 per applicable failure / 1.0	

Memory retrieval and generation. For the ST Graph score in the main text, semantic relevance is implemented using local IDF-weighted cosine similarity. The top three seed places are expanded by one graph hop, and at most five places are returned. Failure penalties are conditioned on the active subgoal and capped to avoid permanently excluding a previously unsuccessful branch. The requested sampling temperature is 0.7 for initialization, planning, verification, and recovery, and 0 for image-region grounding. Their output-token limits are 4,096, 8,192, 8,192, 2,048, and 10,240, respectively.

Distance measurements. Benchmark success distance and the Harness's geometric stopping measurements are distinct. The latter use either fresh depth at a grounded region or distance to a projected navigation target. The corresponding thresholds are 2.5 m and 1.0 m. Fresh depth is the median of a 5×5 patch at the bounding-box anchor, with a maximum patch standard deviation of 0.5 m. Unavailable or unreliable depth triggers the projected-target fallback. These measurements validate physical proximity; they do not replace the benchmark's geodesic distance to the goal or goal viewpoints.

Model Sensitivity and Framework Comparison. To assess model sensitivity and framework performance, we will evaluate HarnessVLN with `GPT-5.6-luna`, `Qwen3.8-flash`, and `GPT-6-astra`, and re-evaluate MSGNav with

Table 6. Model sensitivity and framework comparison on a fixed 100-episode HM3D-OVON subset.

Method	Model	SR ↑	SPL ↑	OSR ↑	Gap ↓
MSGNav	`GPT-5.6-luna`	37.0	18.9	46.0	9.0
HarnessVLN	`GPT-5.6-luna`	55.0	33.0	74.0	19.0
	`GPT-6-astra`	63.0	39.4	76.0	13.0
	`Qwen3.8-flash`	51.0	32.8	75.0	24.0

`GPT-5.6-luna`, on the same fixed 100-episode HM3D-OVON subset (Table 6). Holding the Harness protocol, tools fixed across models assesses model sensitivity, while comparing the two frameworks with a shared model evaluates their performance under matched planner settings.

2 Unified Model Prompts

The same MLLM performs initialization, planning, grounding, verification, and recovery through function-specific prompts. Each call receives the relevant task, labeled observations, TODO progress, retrieved ST Graph evidence, and tool feedback. Below we show two key prompt excerpts, condensed for presentation; the JSON blocks illustrate selected response fields.

Navigation planning: update progress before acting

Inputs: [instruction], [history], [current directional views], [TODO list], [retrieved memory], [feedback], [available actions].

Review all TODO items using their stable `todo_id`, rather than list position. Keep at most one item active. Mark an item completed only with a concrete observed result and its evidence; a blocked item requires a reason. Revisit earlier items when later subgoals have already been completed.

Update progress first, then choose one offered navigation or backtracking action from the updated context. Set `stop` when the instruction endpoint has been reached; pending TODOs do not prevent a stop request.

```
{ "todo_updates": [
    { "todo_id": "todo-1", "status": "completed",
      "result": "<observed evidence of completion>",
      "evidence_ids": ["ob-000001"] }
  ],
  "action": "navigate to left",
  "stop": false }
```

ObjectNav stop verification: identify and localize

Inputs: [target category], [current directional views].

Check that the target is clearly visible and belongs to the requested category; reject look-alikes. Do not estimate distance from RGB: the depth tool measures it. If confident, select the clearest view and return a tight box around the object itself on a normalized $[0, 1000]$ scale. Otherwise return `CONTINUE`, a null view, and an empty box.

```
{ "analysis": "<visible evidence for the category>",
  "decision": "STOP",
  "target_view": "forward",
  "bbox_2d": [350, 200, 600, 800] }
```

Execution contract. A model's stop decision is a request to the Harness, which applies the geometric checks in Appendix 1 before issuing the simulator STOP action. Route verification checks the instruction endpoint, whereas ObjectNav verification checks the target category. Boxes use $[x_1, y_1, x_2, y_2]$; normalized coordinates map to pixels as $x_{\text{px}} = Wx/1000$ and $y_{\text{px}} = Hy/1000$ for a $W \times H$ image.

3 Simulation Visualizations

We illustrate six recorded rollouts across the four benchmarks, using `GPT-5.5` for R2R/RxR and `GPT-5.6-luna` for ObjectNav, with one shared model across all reasoning functions in each rollout. These selected cases show execution behavior and failure modes; they are not aggregate performance estimates.

3.1 Reading the visualization

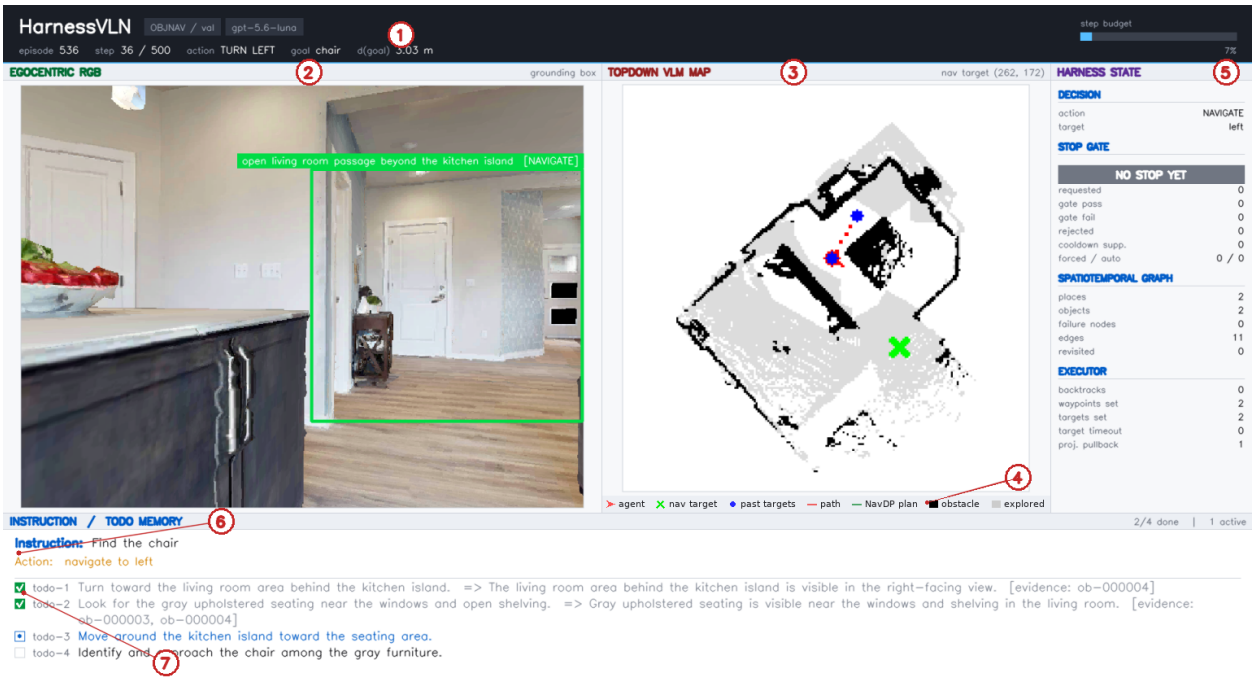

Figure 4. Runtime dashboard. HM3D-v2 episode 536 at step 36. The numbered panels connect the observation, navigation state, and progress memory; the key below describes their contents.

1 **Run status:** model, episode, step budget, action, and evaluator goal distance.
2 **RGB observation:** the current view and any grounding box recorded at that step.
3 **Navigation map:** occupancy, trajectory, and projected navigation targets.
4 **Map key:** agent, current/past targets, path, plan, obstacles, and explored area.
5 **Harness state:** decision, stop checks, ST Graph counts, and executor counters.
6 **Task and action:** the instruction or target category and current navigation choice.
7 **TODO memory:** subgoals, status updates, and supporting observations.

Interpretation. The map is a geometric navigation map; the ST Graph is summarized by counts in panel 5. The header's goal distance is an evaluator diagnostic, not an agent input. In the case figures, each row contains two recorded RGB views and the final runtime map. Frame indices and stop-check event indices are reported separately; a verification view can differ from the displayed forward RGB view. Captions report stop-check distances; terminal benchmark distances are explicitly labeled as goal distances.

3.2 Successful navigation across tasks

RGB: step 51 RGB: step 150 Map: step 150

Figure 5. R2R 650: reaching a route endpoint. The instruction asks the agent to cross the room and wait in the archway in front of the podium. The projected-target check passes at step 149 (0.56 m); the episode succeeds at step 150 with a terminal goal distance of 2.39 m.

RGB: step 0 RGB: step 926 Map: step 926

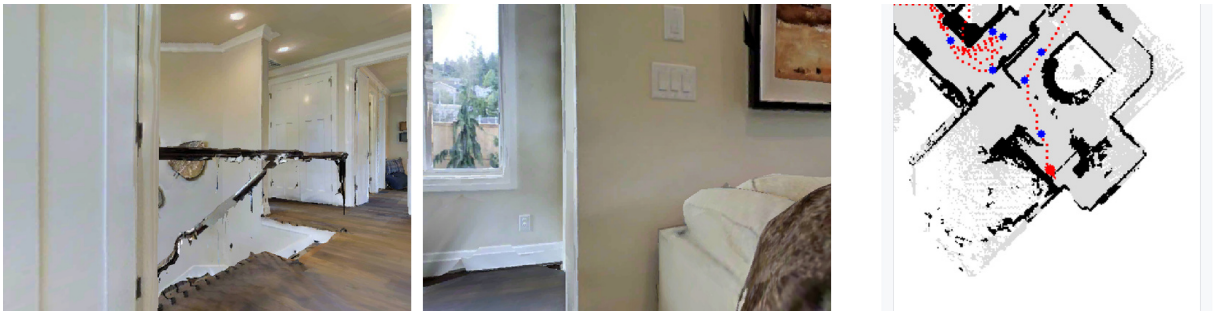

Figure 6. RxR 154: a long instruction-following rollout. The route proceeds from stairs through dining and kitchen areas to a corner near a glass door. The projected-target check passes at step 925 (0.49 m), and the episode succeeds at step 926 with a terminal goal distance of 0.43 m, within the 1,000-step budget.

RGB: step 36 RGB: step 95 Map: step 95

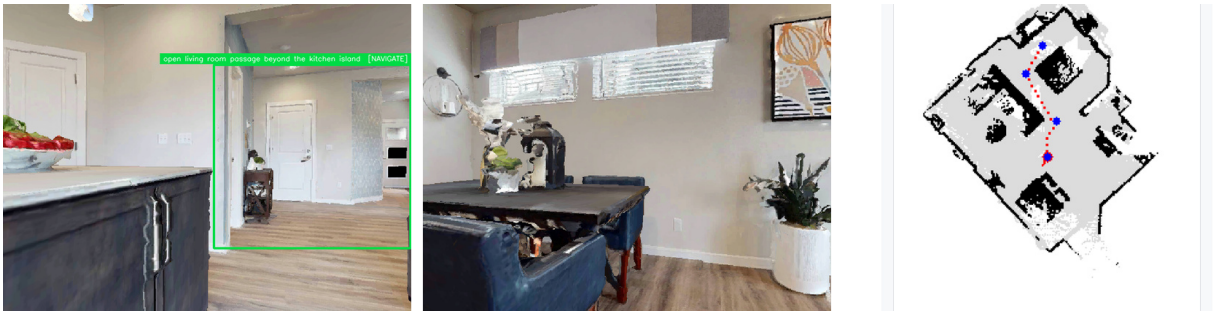

Figure 7. HM3D-v2 536: finding a chair. The early view grounds a passage into the seating area; the later view shows the final position. A fresh-depth check passes at step 94 (0.64 m), followed by success at step 95 with a terminal goal distance of 0.05 m.

3.3 Stopping decisions and failure cases

Figure 8. HM3D-OVON 1297: a rejected stop followed by success. While searching for a picture, fresh depth rejects stopping at step 216 (4.65 m), despite a projected-target distance of 0.40 m. A later depth check passes at step 429 (1.53 m); the episode succeeds at step 430 with a terminal goal distance of 0.13 m.

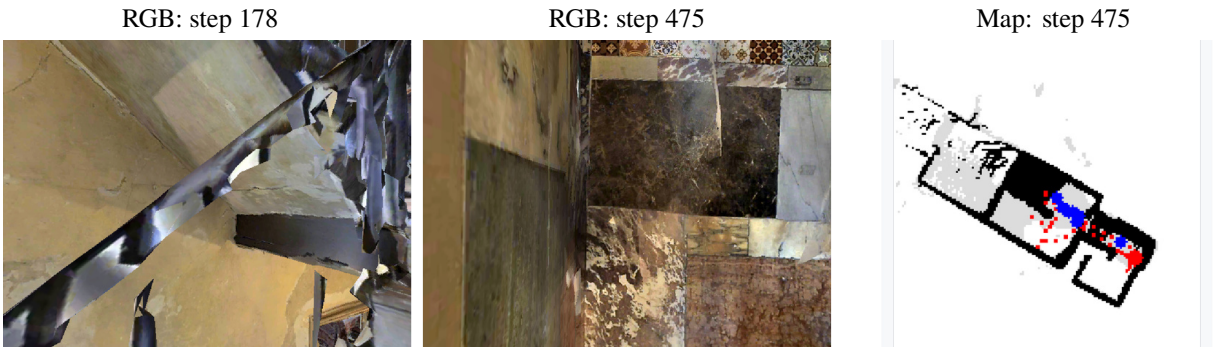

Figure 9. R2R 839: failing to reach the instructed room. The task asks the agent to descend the stairs and enter the room on the left. The projected-target check eventually passes at step 474 (0.998 m), but termination is 6.48 m from the goal. Proximity to the selected target does not establish that the instruction endpoint is correct.

Figure 10. HM3D-OVON 2469: passing the gate outside the success region. For the target category *picture*, fresh depth passes at step 46 (1.20 m). The episode stops at step 47 with a benchmark goal distance of 2.15 m and fails. Sensor proximity and benchmark success remain distinct.