Nova-HEI

Nova-Harness Embodied Intelligent

Abstract

Embodied applications require heterogeneous capabilities to be organized into sustained, verifiable workflows across models and robot platforms. We introduce **Nova-Harness Embodied Intelligent (Nova-HEI)**, a Harness framework with a reusable runtime that separates task organization from capability implementations through shared task state, skill interfaces, and execution feedback. To evaluate the framework, we develop **Nova-HEI-based Navigation**, a vision-and-language navigation (VLN) instantiation for initially unseen targets that combines evidence memory with event-driven monitoring, recovery, and completion verification without retraining the navigation backbone. Shared interfaces and backend adapters allow the upper-level workflow to be reused across benchmark evaluation, MuJoCo simulation, and a physical Go2 quadruped. With unchanged LightNav-0 weights and matched environment-action limits, our Scheduling Harness configuration improves success rates on Habitat-GS, HM3Dv2, and HM3D-OVON. On Habitat-GS, success rate increases by 13.262 percentage points (33.3% relative), and success weighted by path length (SPL) improves by 12.5% relative. MuJoCo comparisons show successful target approach with the agent and failed approaches without it in three selected cases. Two Go2 experiments further demonstrate trash-bin and pantry-counter approach using the same upper-level workflow. Together, these navigation results support the value of the Harness framework for improving task completion and reusing task workflows across backends.

1 Introduction

Embodied applications such as industrial inspection, campus maintenance, and indoor services require robots to translate operational requirements into sustained, verifiable physical actions. A complete workflow can span navigation, search, inspection, manipulation, and interaction. For example, an inspection workflow may use a drone for a wide-area survey, a quadruped for close inspection, and a quadruped equipped with a robotic arm for follow-up manipulation. These complementary capabilities require a common way to describe goals, coordinate skills, and verify outcomes.

Building these workflows presents three challenges for developers and deployment teams. Tasks involving initially unseen targets require exploration, evidence accumulation, and explicit completion verification when target locations are not provided. Models and robots differ in their sensing, motion, and control interfaces, which must be aligned with task intent. Robot teams need shared task state so that progress and evidence can pass between stages and platforms. When these responsibilities are embedded in task code tied to particular models, robots, or environment rules, adding capabilities or changing deployments requires repeated integration and validation. These requirements motivate a common Harness runtime that manages tasks and feedback through shared interfaces.

Prior robot-agent systems already combine planning, skills, verification, and memory [6, 7]. We propose **Nova-Harness Embodied Intelligent (Nova-HEI)**, a Harness framework for embodied applications. Its core is a reusable runtime that connects task goals to skill execution through

shared task state, skill interfaces, and execution feedback. The runtime maintains progress and observation-linked evidence, invokes capabilities, and uses execution results to support monitoring, recovery, and completion checks. Model protocols, robot commands, and deployment constraints are handled through adapters and configuration. This separation gives application logic a common execution structure across backends. Figure 1 connects the framework to three challenges: navigation to initially unseen targets, deployment adaptation, and robot coordination.

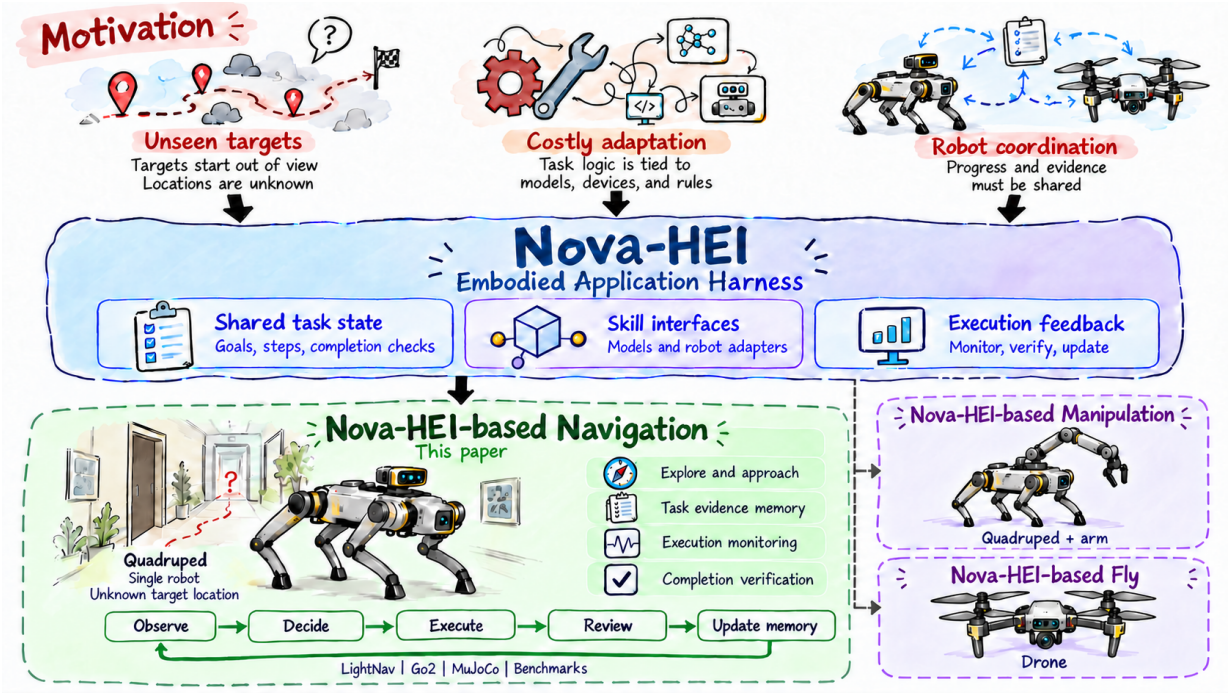


Figure 1: Application requirements motivating Nova-HEI. Navigation to initially unseen targets, deployment adaptation, and robot coordination motivate a shared Harness with task state, skill interfaces, and execution feedback. Nova-HEI-based Navigation is the evaluated instantiation, with benchmark, MuJoCo, and Go2 backends. The two right-hand branches denote Nova-HEI-based Manipulation with a quadruped equipped with a robotic arm and Nova-HEI-based Fly with a drone, both explored in preliminary trials.

As a concrete instantiation, we develop **Nova-HEI-based Navigation** for vision-and-language navigation (VLN) to initially unseen targets. Here, "initially unseen" means that the target is outside the robot's field of view at task start and its location is not provided. Search and approach expose partial observability, decisions across task stages, physical execution errors, and completion verification, making them a useful setting for assessing the runtime. The implementation integrates LightNav [9] without modifying its weights or internal inference process. Evidence memory and event-driven supervision support exploration, target pursuit, recovery, and completion review, while backend adapters reuse the workflow across benchmark evaluation, MuJoCo, and Go2.

We assess this instantiation through matched benchmark comparisons and complementary qualitative experiments. With unchanged LightNav-0 weights and identical environment-action limits, Scheduling Harness improves success rate (SR) on Habitat-GS, HM3Dv2, and HM3D-OVON. On Habitat-GS, SR increases by 13.262 percentage points (33.3% relative), and success weighted by path length (SPL) improves by 12.5% relative. Three selected MuJoCo comparisons show successful target approach with the agent and failed outcomes without it. Two Go2 tasks further demonstrate trash-bin and pantry-counter approach using the same upper-level workflow.

Our contributions are threefold:

1. **A Harness framework for embodied applications.** We organize task state, skill in-

terfaces, and execution feedback into a reusable runtime, separating application logic from model and robot implementations.

2. **A navigation instantiation with runtime supervision.** We implement Nova-HEI-based Navigation with observation-linked evidence memory, event-driven monitoring and recovery, and completion verification around an unchanged navigation backbone. Shared interfaces support reuse across benchmark, simulation, and physical-robot backends.
3. **Evaluation of runtime supervision in navigation.** We demonstrate success-rate gains through matched comparisons on three benchmarks and complement them with agent-versus-VLN simulation comparisons and two physical target-approach tasks.

2 Related Work

2.1 VLN

Vision-and-language navigation maps language instructions and visual observations to navigation actions. Recent work on real robots has mainly improved reliability and deployment efficiency through hierarchical execution, long-term memory, and unified navigation models.

NaVILA [1] connects a vision-language model to a movement policy through language-based intermediate actions, enabling long-distance navigation and deployment across different legged robots. The authors report failures to correct deviations, showing that task recovery remains an issue in hierarchical execution. DualVLN [2] combines low-frequency semantic planning with a high-frequency diffusion policy. It generates continuous trajectories from pixel goals and latent features to improve avoidance of moving obstacles. However, its local policy still has limits when the selected goal has a large error or refers to the wrong target. These methods focus on coordination between navigation decisions and movement execution, and transfer requires matching their policies or internal representations. We keep existing navigation and control components and study how explicit task state, execution feedback, and target checks can handle deviations during long tasks.

Modeling a long history requires a balance between retaining information and controlling computation. StreamVLN (ICRA 2026) [3] uses a sliding-window KV cache, slowly updated memory, and spatial token pruning to support streaming navigation with low latency. AstraNav-Memory (CVPR 2026) [4] uses visual compression to store more history and support navigation memory across tasks. Its ablation analysis [4] also shows that excessive compression and overly long context can reduce performance. These methods improve memory and reasoning inside the model, so integration depends on their encoding, caching, or training mechanisms. Nova-HEI-based Navigation instead links observations, target sightings, and actual action results at the task level. It selects relevant evidence to support decisions across stages, and its current memory is limited to a single task.

NavFoM (ICLR 2026) [5] combines different navigation capabilities through joint training across tasks and robot platforms. It uses time and viewpoint labels, along with sampling that accounts for the available budget, to handle different observation setups. Its results demonstrate the potential for generalization at the model level, while practical deployment still requires connections to each platform's planning and control components. We focus on runtime issues beyond a unified model, including execution monitoring, target confirmation, and interface adaptation. Models such as NavFoM can provide the underlying navigation capability, but model generalization and task workflow reuse need separate evaluation.

2.2 VLN Agents

VLN agents organize long tasks through skill calls, memory, and feedback, allowing local navigation to serve an ongoing task goal. Previous work already covers active exploration, self-correction, and completion verification. The differences in our work therefore need to be established through how evidence is organized, how execution is constrained, and what integration requires.

AgentVLN [6] is the work most directly related to ours. It uses a vision-language model to select perception and planning skills. Three-dimensional path projection, active exploration, and geometric queries made as needed support navigation and target localization. Its approach combines depth, pose, SLAM, and dedicated instruction tuning, and reports deployment on local devices. These requirements should be evaluated separately from computational efficiency; see its method and deployment details [6]. Nova-HEI-based Navigation currently calls an existing VLN model through external interfaces. At the upper level, it explicitly maintains coverage of observed directions, candidate failures, target pursuit, and checks based on current observations. It does not require access to internal VLN features or changes to the model's inference process. This difference offers a way to reuse existing systems. Its reliability and cost benefits still need to be compared with AgentVLN under comparable conditions.

ABot-AgentOS (2026) [7] combines planning, isolated skill execution, verification at multiple stages, and persistent graph memory in a general robot runtime layer. It supports interaction across tasks and execution across robot platforms. Results on an initial benchmark subset support the value of runtime improvements, but are not enough to establish the full costs and benefits under different real deployment conditions. We focus on target search and approach. Observation references connect spatial evidence, action results, and later reviews within a task. The current system does not include graph memory across tasks or automatic self-improvement. With this focused scope, Nova-HEI-based Navigation addresses reliability during long tasks through state transitions, failure constraints, and completion criteria that can be inspected.

Overall, existing work provides effective navigation models and agent mechanisms. Nova-HEI-based Navigation further studies a reusable task loop built on existing systems. It uses evidence memory and execution review to improve extended target search while keeping the navigation model and movement execution components unchanged. Clear interfaces separate the task workflow from differences between models and robot platforms.

3 Agent Framework

Nova-HEI-based Navigation provides a task-level control loop for searching for and approaching targets that are initially out of view and whose locations are unknown. It leaves the parameters and inference process of the existing vision-and-language navigation model unchanged. Section 3.1 outlines the task loop, and Section 3.2 explains its core mechanisms in the order in which a task runs, from start to finish.

3.1 Overall Workflow

Given a natural-language goal and a task budget, Nova-HEI-based Navigation advances the task through a cycle of observation, decision, execution, review, and memory update. The upper-level agent organizes evidence, selects skills, and maintains task state. The underlying VLN model generates local paths, while the robot execution layer tracks those paths and returns feedback. During navigation, upper-level decisions are mainly triggered by feedback events. Current observations and historical memory jointly support subsequent adjustments.

A navigation stop or a detection provides a clue that still needs review. The system uses current

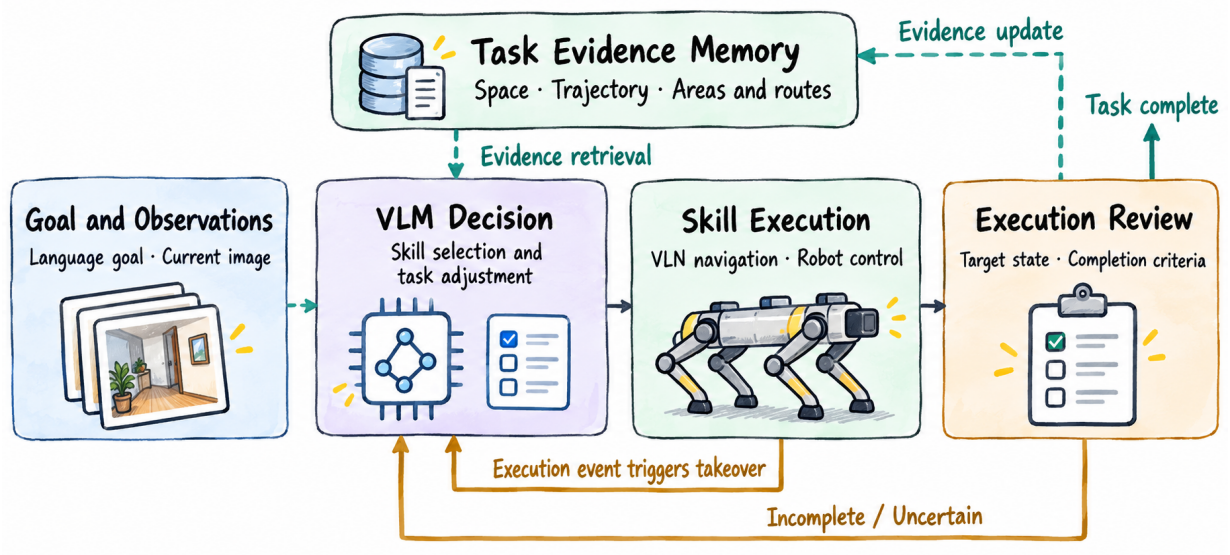


Figure 2: The overall Nova-HEI-based Navigation task loop. The VLM combines the goal, current observation, and evidence memory to select skills, then adjusts the task based on execution events and review results. Relevant evidence is stored in memory within the task, and the task ends when its completion criteria are met. Solid lines show execution and feedback; dashed lines show evidence exchange.

evidence to assess target state and task completion, records action history and review results in memory, and decides whether to continue exploration, maintain target pursuit, or end the task. Figure 2 summarizes this loop, and Figure 3 shows its operation in more detail.

3.2 Core Mechanisms of the Task Loop

3.2.1 Task Initialization and Component Assembly

The task takes a natural-language goal g and a budget B as input. The system loads skill descriptions and environment constraints, creates the initial task state and task memory, and uses application configuration to assemble the model client, navigation session, and robot execution components. Environment rules are expressed through configuration, prompts, and deployment validators. Dedicated adapters handle differences in model protocols and robot platforms.

Shared interfaces define observations, predictions, actions, feedback, and component lifecycles so that task logic can be reused. Existing compatible combinations can be selected through configuration. Adding a model or robot requires implementing the relevant interfaces and validating compatibility. LightNav is the current built-in model, and the native Go2 service still uses the default LightNav setup. Other models require the corresponding protocol and server support. The adaptation layer at the bottom of Figure 3 represents the interface support available throughout the task.

3.2.2 Observation Capture and Evidence Context

Each observation is written as $o_i = (I_i, p_i, \tau_i, \mathrm{id}_i)$, containing an image, the robot pose at image capture, a timestamp, and a stable observation ID. Images captured once per second during navigation are collected as passive evidence; image capture alone does not trigger the Planner. When an upper-level decision or verification is needed, the system selects the current observation and gathers historical evidence relevant to the task.

Task memory $\mathcal{M}$ contains three types of records. **Spatial Memory** stores images, poses, se-

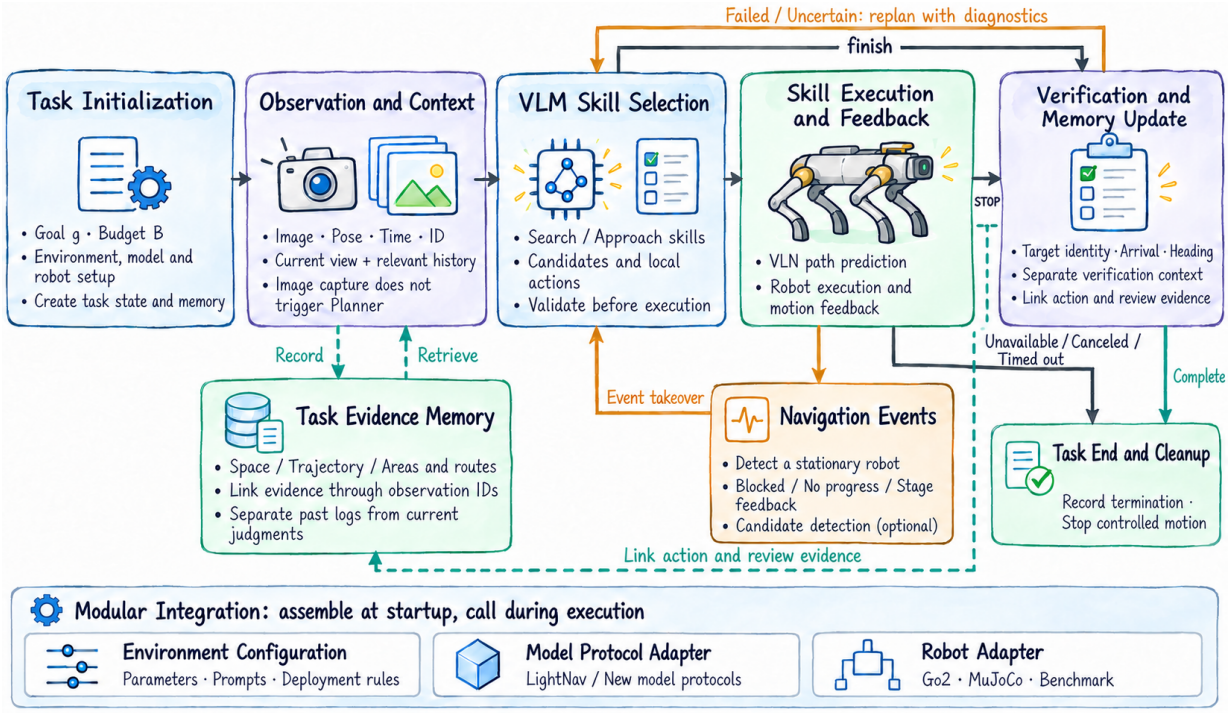


Figure 3: Task core and modular integration for Nova-HEI-based Navigation. The main workflow covers task initialization, observation and evidence organization, VLM decisions, skill execution, and completion verification, corresponding to Sections 3.2.1–3.2.5. Execution events trigger upper-level control. If verification fails, its diagnostic feedback is passed back to the decision stage. Action and review evidence is linked and stored in memory. The system performs task cleanup when completion is confirmed or another termination condition is met. The adaptation layer at the bottom is assembled during initialization and provides shared interfaces during execution.

mantic reports, and target sightings to support retrieval of relevant observations and identity checks. An observation location cannot be treated directly as an object's coordinates. **Trajectory Memory** links action intent, actual motion, and reviews before and after an action. It helps identify repeated attempts and returns to previously visited locations; missing intermediate routes are not inferred from their start and end points. **Topology Memory (Areas and Routes)** records area identity, previously used entrances, and routes that still need inspection. Connections between areas require support from both actual motion and visual review.

Before the kth upper-level decision, the system retrieves context using the goal, task state s_k, current observation, and event diagnostics e_k:

$$\mathcal{C}_k = \text{Retrieve}(\mathcal{M}; g, s_k, o_{i_k}, e_k).$$

The system applies separate limits to retrieved observations, routes, and action histories, and includes each original image only once in a context. When needed, it compares the current image captured while stopped with historical images of areas and entrances to support place matching. Historical sightings, execution feedback, and current visual judgments are recorded separately. An empty list of area candidates does not establish that the search is complete.

3.2.3 VLM Skill Selection and Event Scheduling

The upper-level VLM acts as the Planner. Using skill descriptions and tool specifications, it explicitly selects a skill z_k and an action a_k based on the current observation, task state, relevant evidence, and event diagnostics:

$$(z_k, a_k) = \text{Planner}(g, s_k, o_{i_k}, \mathcal{C}_k, e_k).$$

The `search_navigation` skill supports exploration when no specific target clues are available. The `approach_target` skill supports approaching and checking a clear or possible target, as well as searching nearby to find it again. Both skills share candidates, exploration objectives, and target pursuit state. Common observation, turning, and recovery actions do not have to specify a navigation skill.

During navigation, the Planner is called mainly in response to feedback events, rather than at fixed intervals. `DIRECT`, `STAGING`, and `VERIFY` denote the underlying execution, upper-level control, and completion verification states in this decision process. Algorithm 1 gives the scheduling rules. The scheduler handles one event per invocation; passive image capture alone does not invoke the Planner. Completion verification and task termination follow Section 3.2.5.

Algorithm 1 Event-driven scheduling for VLM decisions.

Require: Event e; control state $q \in \{\texttt{DIRECT}, \texttt{STAGING}, \texttt{VERIFY}\}$
Require: Goal g; task state s; task memory $\mathcal{M}$
Ensure: Updated control state and an optional Planner decision
if e reports a LightNav `STOP` **then**
  $q \leftarrow$ `VERIFY`
  $o \leftarrow$ CurrentStoppedObservation
  VerifyCompletion(g, o) ▷ Identity, arrival, and heading
  return $(q, \varnothing)$
else if e is a verification result with any false or uncertain criterion **then**
  $e \leftarrow$ VerificationDiagnostics(e)
else if e reports two consecutive frames with the same category and matching boxes **then**
  InterruptNavigation ▷ Review the candidate in STAGING
else if e reports $\Delta p < 0.05\,\text{m}$ and $\Delta\theta_{\text{acc}} < 10^\circ$ over $\approx 8\,\text{s}$ **then**
  InterruptNavigation ▷ Inspect the cause in STAGING
else if e reports an execution problem or the end of a stage **then**
  $e \leftarrow$ ExecutionDiagnostics(e)
    ▷ Blocked motion, no progress, local observation budget end, etc.
else if e reports successful `look`/`orient` with a remaining Planner instruction u **then**
  $q \leftarrow$ `DIRECT`
  ResumeLightNav(u)
  return $(q, \varnothing)$
else
  return $(q, \varnothing)$ ▷ No Planner call for other events
end if
$q \leftarrow$ `STAGING`
$o \leftarrow$ CurrentStoppedObservation
$\mathcal{C} \leftarrow \text{Retrieve}(\mathcal{M}; g, s, o, e)$
$(z, a) \leftarrow \text{Planner}(g, s, o, \mathcal{C}, e)$
return $(q, (z, a))$

Before an action is submitted, the system checks its output structure, skill requirements, current evidence, and environment rules. Validation runs on a copy of the state, so rejected actions do not change the actual execution state. Navigation choices must refer to a currently visible candidate. Exploration candidates must also state what observation question is expected to be answered by moving closer.

3.2.4 Skill Execution and Event Feedback

The tool layer executes validated actions. The `navigate` tool calls the VLN model to generate local paths. The `look`, `orient`, and `recover` tools use the robot's observation, turning, and

recovery capabilities. The navigation coordination layer manages sessions, receives feedback, and handles execution interruptions. The robot execution layer handles path tracking and motion constraints. Each predicted path is paired with the original image capture timestamp and pose. It is interpreted in the robot's body frame at capture time, then transformed by the executor. Predictions without a matching pose are not executed.

Pose monitoring actively detects when the robot remains stationary. Blocked movement, lack of progress, and the end of a local observation segment's budget are handled based on feedback actually returned by the execution interface. The current system does not continuously detect semantic progress. The Planner uses these events and the current observation to assess the cause, then decides whether to gather more observations, adjust the strategy, or call a recovery action supported by the backend. After `look` or `orient` succeeds and the Planner supplies the remaining instruction, control returns to LightNav.

YOLO-World [8] detection during search navigation can be enabled through configuration. `TargetWatch` runs asynchronous detection on images captured along the route. When two consecutive frames have the same category and matching detection boxes, the system checks image freshness, session state, and whether the result belongs to the current navigation action, then interrupts navigation. The Planner reviews the candidate using the triggering image, detection boxes, and the current observation captured while stopped. It then decides whether to continue searching or start target approach. The system briefly suppresses rejected candidates and prevents late results from affecting the current state. Detection is disabled during target approach, turning, and recovery. This module is off by default and requires streaming observations and a navigation interruption interface. If it is unavailable, the system continues basic navigation and handles other events.

3.2.5 Completion Verification, Memory Update, and Task Termination

After execution results return, the system links action intent, actual motion, and subsequent review, then updates candidate, target, and area states. Candidate failures are counted by observation location. The first and most recent reliable target sightings support continued pursuit. When a target is temporarily lost, the robot can make a limited number of attempts to find it through directed observation or currently visible viewpoints. It can defer pursuit and resume exploration when needed. Updated memory is retrieved for the next decision, linking evidence across task stages.

A LightNav `STOP` or an upper-level `finish` proposal triggers completion verification, which checks target identity, arrival, and heading. By default, verification uses a separate request through the same VLM client, providing only the target requirements and the current image captured while stopped. This separates verification from the historical decision context. If any completion criterion is false or uncertain, the system returns to `STAGING` with diagnostic feedback. Once the task requirements are confirmed, it records completion and ends the task. It does not declare success if verification encounters an error. Internal verification supports task scheduling; experimental outcomes are still determined by independent scoring.

The task can also end when the agent gives up, the budget is exhausted, or the task is canceled. Navigation service errors, cancellation, and timeouts are treated as unavailable execution, and the task lifecycle performs the corresponding cleanup. In contrast, the end of a local observation segment's budget is passed to the Planner as stage feedback. When an action ends or exits because of an error, the backend stops motion controlled through that connection.

4 Experiments

We evaluate Nova-HEI-based Navigation through two complementary studies. Benchmark experiments (Section 4.1) quantify the effect of task-level supervision on completion and navigation efficiency while keeping the navigation backbone fixed. Simulation and real-robot experiments (Section 4.2) examine target-approach outcomes in MuJoCo and on a physical Go2 robot, providing paired visual evidence of task completion and illustrating reuse of the upper-level workflow across backends.

4.1 Benchmark Experiments

We compare LightNav-0 [9] with its Scheduling Harness configuration to assess the benefit of adding Nova-HEI's task-level runtime to an existing navigation policy. The comparison retains the same navigation weights, evaluation episodes, and environment-action budgets across three benchmarks.

4.1.1 Experimental Setup

We evaluate on Habitat-GS, HM3Dv2, and HM3D-OVON. **Habitat-GS** uses the fixed INSIGHT-Bench v1 subset [10] constructed from Habitat-GS scenes [15]: 279 language-guided tasks across 10 indoor and outdoor scenes. **HM3Dv2** uses the ObjectNav validation set with 1,000 episodes, 36 scenes, and six target categories, based on HM3D scenes [16] and semantic annotations [11]. **HM3D-OVON** [12] uses all 3,000 episodes of the `val_unseen` split. Results from the separate 120-episode subset are excluded.

Evaluation protocol. Both configurations use the same environments and scoring rules. Episodes allow at most 300 environment actions on Habitat-GS and 500 on either HM3D benchmark; internal control handoffs do not consume environment actions. Habitat-GS success requires the final pose to satisfy the annotated planar-distance radius and a horizontal target-visibility constraint of 120°: 2.0 m for 51 indoor episodes and 3.0 m for 228 outdoor episodes. In the evaluated HM3D configurations, success requires `STOP` within a geodesic distance of 0.1 m (HM3Dv2) or 0.25 m (HM3D-OVON) from a valid goal viewpoint.

The baseline **LightNav-0** [9] executes navigation directly. **Scheduling Harness** retains the same LightNav-0 weights without benchmark-specific fine-tuning and uses Qwen3.8-Max (`qwen3.8-max-0902`) [18] for event-driven stage management, stop verification, and limited recovery. The agent receives instructions, visual observations, and navigation history, with no ground-truth target positions, target distances, or evaluator feedback. We also include available **InternVLA-N1** [17] results on Habitat-GS and HM3Dv2 as an additional comparator. The shared-weight LightNav-0 comparison is the primary evaluation of the runtime contribution.

We assess these methods using success rate (SR), success weighted by path length (SPL) [13], final navigation error (NE), and oracle success (OS). SR measures completed episodes; SPL weights success by shortest-path length relative to executed path length. NE measures final target distance, and OS indicates whether a trajectory reaches the evaluator's goal region during execution. SR and OS are percentages; NE is in meters. Habitat-GS nDTW [14] provides an auxiliary measure of reference-path similarity.

Implementation. Experiments run on Ubuntu 22.04.5 with eight RTX 5090 GPUs (32 GB each), two Xeon Platinum 8575C CPUs (96 physical cores total), 1.5 TiB RAM, and driver 580.126.09. HM3D evaluations use Habitat-Lab 0.3.20231024 and Habitat-Sim 0.3.1 [19]; Habitat-

GS uses Isaac Sim 5.1.0 and Isaac Lab package 0.54.2 [20]. LightNav-0 uses Python 3.11, PyTorch 2.10.0 [21] (CUDA 12.8), and vLLM 0.19.1 [22]. Qwen3.8-Max runs through a cloud API. We evaluate navigation outcomes; API latency and cost are not measured.

4.1.2 Quantitative Results

Table 1 consolidates completion, path efficiency, and arrival diagnostics. Scheduling Harness achieves the highest observed SR and OS on all three benchmarks among the available results, with the largest completion gain and the best SPL on Habitat-GS. These gains are obtained with unchanged navigation weights and fixed environment-action limits.

Table 1: Quantitative benchmark results. All scores are from our evaluation. Scheduling Harness adds task-level supervision to the same LightNav-0 weights. Bold marks the best available result within each benchmark. Arrows indicate whether higher or lower is better; dashes denote unavailable results.

Method	SR (%) ↑	SPL ↑	NE (m) ↓	OS (%) ↑
Habitat-GS [10, 15] (279 episodes)				
LightNav-0 [9]	39.785	0.376	**5.095**	55.0
InternVLA-N1 [17]	6.810	0.060	9.380	38.7
Scheduling Harness (ours)	**53.047**	**0.423**	5.577	**71.6**
HM3Dv2 [11, 16] (1,000 episodes)				
LightNav-0 [9]	77.10	**0.413**	1.789	83.0
InternVLA-N1 [17]	17.99	0.170	3.02	72.3
Scheduling Harness (ours)	**77.50**	0.409	**1.629**	**85.4**
HM3D-OVON [12] (3,000 episodes, `val_unseen`)				
LightNav-0 [9]	46.400	**0.237**	3.754	64.467
InternVLA-N1 [17]	—	—	—	—
Scheduling Harness (ours)	**47.667**	0.236	**3.612**	**67.967**

On Habitat-GS, Scheduling raises SR from 39.785% to 53.047%, a gain of 13.262 percentage points (33.3% relative), while improving SPL from 0.376 to 0.423 (12.5% relative). OS increases by 16.6 percentage points, from 55.0% to 71.6%, showing that more trajectories reach the goal region during execution. The joint improvement in SR, SPL, and OS indicates that task-level supervision can convert more of the backbone's navigation capability into completed tasks under the distance and visibility constraints. The shared weights and 300-action limit establish this as a benefit of the augmented execution system, without additional navigation-model fine-tuning.

The runtime also improves completion on both HM3D benchmarks. HM3Dv2 SR increases from 77.10% to 77.50%, accompanied by an 8.9% reduction in NE (1.789 m to 1.629 m) and a 2.4-point increase in OS. On the full HM3D-OVON split, SR improves by 1.267 percentage points, from 46.400% to 47.667%; NE decreases by 3.8% (3.754 m to 3.612 m), and OS increases by 3.500 points. Thus, completion, final proximity, and reaching the goal region improve together in both settings. SPL remains close to LightNav-0, decreasing by 0.004 on HM3Dv2 and 0.001 on HM3D-OVON.

Together, these results support Nova-HEI's use of a task-level runtime to improve an existing navigation policy without modifying its parameters. SR and OS improve across all three benchmarks, and Habitat-GS additionally shows a clear SPL gain. This motivates runtime supervision as a practical complement to model development. Trajectory refinement remains useful: Habitat-GS NE rises from 5.095 m to 5.577 m and nDTW falls from 0.530 to 0.410.

4.2 Simulation and Real-Robot Experiments

Complementing the benchmark results, we examine target-approach outcomes in MuJoCo and on a physical Go2 robot. The simulation comparison visualizes the effect of enabling the agent, while the Go2 experiments validate execution of the task workflow with physical sensing and motion.

Across both backends, agent-enabled runs reuse the same upper-level workflow. LightNav provides local navigation, and the agent manages task state and completion review following Section 3.2.5. Paired external and robot views document the robot's position relative to the target and the target appearance at the final state.

4.2.1 Qualitative Results in Simulation

Figure 4 compares agent-enabled and VLN-only runs on three tasks. With the agent, the robot reaches the basketball, bed, and refrigerator in the selected cases. Without the agent, it ends in another area during basketball search (a), remains near a doorway during bed approach (b), and faces a wall away from the refrigerator (c). The paired views show successful target approach with the agent and failure to reach the requested target without it. These contrasts provide visual evidence consistent.

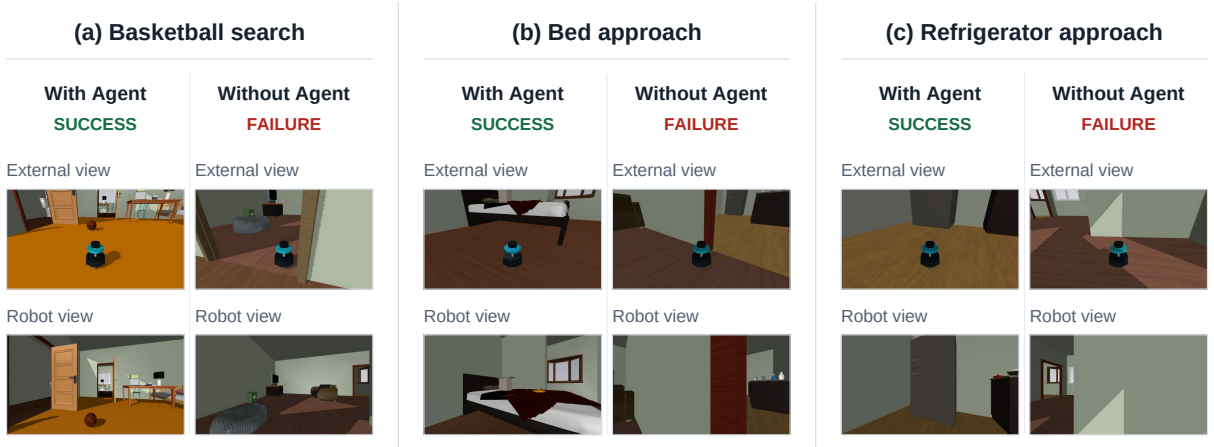


Figure 4: Agent versus VLN-only execution in MuJoCo. Task groups show (a) basketball search, (b) bed approach, and (c) refrigerator approach. Each group places the successful agent-enabled run on the left and the failed VLN-only run on the right. External views are above the corresponding robot views. All images show final observations.

4.2.2 Qualitative Results on the Go2 Robot

We further validate the agent workflow on two physical target-approach tasks: a silver trash bin and a pantry counter. The Go2 adapter supplies observations and executes motion commands while retaining the upper-level task logic.

The navigation stack also supports an AI Box equipped with an NVIDIA Jetson AGX Orin as an edge deployment option. Figure 5 and Table 2 show the device and its specifications.



Figure 5: AI Box for edge deployment. The device supports Nova-HEI-based Navigation.

Table 2: AI Box hardware specifications.

Component	Specification
Processor	NVIDIA Jetson AGX Orin
Peak AI performance	Up to 275 TOPS
Memory	64 GB, 256-bit LPDDR5 DRAM

For the silver trash-bin task, Figure 6 shows Go2 close to the requested bin. The external view establishes proximity, and the robot view resolves the silver body and dark lid. Together, they provide visual evidence for target identity and arrival in the physical environment.

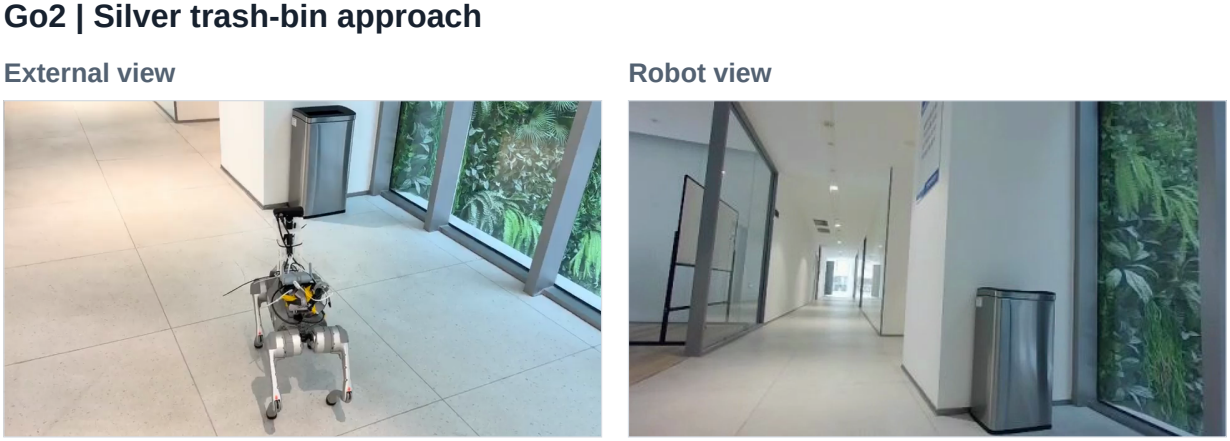


Figure 6: Silver trash-bin approach on Go2. Final external (left) and robot (right) views show the robot near the requested bin and its identifying appearance.

Figure 7 shows the final state of the pantry-counter task. The external view places Go2 beside the lounge seating, while the robot view reveals the adjacent counter, cabinetry, and drinking-water equipment. This pair supports target confirmation and demonstrates execution of the approach task in a furnished indoor environment.

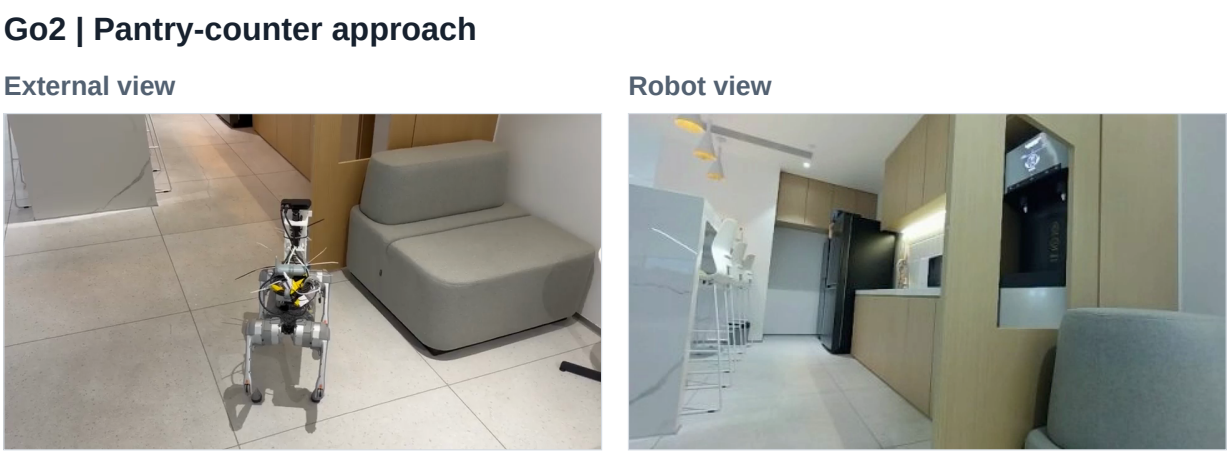


Figure 7: Pantry-counter approach on Go2. The final external view (left) locates the robot beside the lounge, and the robot view (right) shows the adjacent pantry target.

Table 1 quantifies the completion benefit of task-level supervision. Figure 4 illustrates corresponding successful and failed approach outcomes, and the Go2 cases show execution through a physical robot adapter. The shared upper-level workflow connects benchmark evaluation, simulated comparisons, and real-robot deployment. These selected cases establish qualitative outcomes and deployment feasibility; the benchmark scores provide the aggregate performance comparison.

5 Conclusion

We presented Nova-HEI, a Harness runtime for embodied task execution, and evaluated its navigation instantiation, Nova-HEI-based Navigation. With unchanged LightNav-0 weights and fixed environment-action limits, Scheduling Harness improves success rates on Habitat-GS, HM3Dv2, and HM3D-OVON. On Habitat-GS, SR increases by 13.262 percentage points (33.3% relative), accompanied by a 12.5% relative improvement in SPL. These results demonstrate the completion benefit of augmenting an existing navigation model with task-level runtime supervision.

MuJoCo comparisons provide visual evidence of improved target approach, while the Go2 experiments demonstrate execution of the trash-bin and pantry-counter tasks with the same upper-level workflow. The quantitative gains and physical executions support a practical value of the Harness design: improving task completion without retraining the navigation backbone and reusing task logic across simulation and real-robot backends. Runtime supervision therefore offers a complementary direction to navigation-model development for embodied deployment.

6 Future Work

Reliability and efficiency. The benchmark results do not isolate the contributions of verification, monitoring, and recovery, and gains in trajectory quality vary across datasets. We will conduct controlled ablations and optimize intervention frequency to improve completion and path efficiency jointly, while quantifying additional inference latency and API cost.

Broader validation. Current quantitative evidence centers on LightNav-0, while the Go2 results provide qualitative validation. We will test additional navigation backbones and conduct repeated physical trials, measuring false completion, recovery success, and integration effort alongside task success. These evaluations will establish how broadly the observed benefits hold across model choices and real-world conditions.

Manipulation and aerial tasks. Building on preliminary experiments with quadruped robots equipped with robotic arms and with drones, we will extend systematic evaluation to navigation–manipulation workflows and aerial tasks. The priority is to assess skill transitions, embodiment-specific execution constraints, and outcome verification through end-to-end task success and failure recovery.

Reference

[1] An-Chieh Cheng, Yandong Ji, Zhaojing Yang, et al. *NaVILA: Legged Robot Vision-Language-Action Model for Navigation*. Robotics: Science and Systems (RSS), 2025. Paper.

[2] Meng Wei, Chenyang Wan, Jiaqi Peng, et al. *Ground Slow, Move Fast: A Dual-System Foundation Model for Generalizable Vision-and-Language Navigation*. International Conference on Learning Representations (ICLR), 2026. Paper.

[3] Meng Wei, Chenyang Wan, Xiqian Yu, et al. *StreamVLN: Streaming Vision-and-Language Navigation via SlowFast Context Modeling*. IEEE International Conference on Robotics and Automation (ICRA), 2026. Paper.

[4] Botao Ren, Junjun Hu, Xinda Xue, et al. *AstraNav-Memory: Contexts Compression for Long Memory*. arXiv:2512.21627, 2025. Preprint. CVPR 2026 publication.

[5] Jiazhao Zhang, Anqi Li, Yunpeng Qi, et al. *Embodied Navigation Foundation Model*. International Conference on Learning Representations (ICLR), 2026. Paper.

[6] Zihao Xin, Wentong Li, Yixuan Jiang, et al. *AgentVLN: Towards Agentic Vision-and-Language Navigation*. European Conference on Computer Vision (ECCV), 2026. Paper. Official code.

[7] Jiayi Tian, Shiao Liu, Yuting Xu, et al. *ABot-AgentOS: A General Robotic Agent OS with Lifelong Multi-modal Memory*. arXiv:2607.10350, 2026. Paper.

[8] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. *YOLO-World: Real-Time Open-Vocabulary Object Detection*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 16901–16911, 2024. Paper.

[9] Shaoan Wang, Aocheng Luo, Fei Huang, et al. *LightNav-0: Eliciting VLM Spatial Intelligence for Generalist Embodied Navigation*. arXiv:2608.30935, 2026. Paper.

[10] Light Origins Team. *INSIGHT-Bench v1*. Benchmark software and dataset, 2026. Official repository.

[11] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, et al. *Habitat-Matterport 3D Semantics Dataset*. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4927–4936, 2023. Paper.

[12] Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. *HM3D-OVON: A Dataset and Benchmark for Open-Vocabulary Object Goal Navigation*. arXiv:2409.14296, 2024. Paper.

[13] Peter Anderson, Angel Chang, Devendra Singh Chaplot, et al. *On Evaluation of Embodied Navigation Agents*. arXiv:1807.06757, 2018. Paper.

[14] Gabriel Ilharco, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. *General Evaluation for Instruction Conditioned Navigation using Dynamic Time Warping*. arXiv:1907.05446, 2019. Paper.

[15] Ziyuan Xia, Jingyi Xu, Chong Cui, et al. *Habitat-GS: A High-Fidelity Navigation Simulator with Dynamic Gaussian Splatting*. European Conference on Computer Vision (ECCV), pp. 306–323, 2026. Paper.

[16] Santhosh K. Ramakrishnan, Aaron Gokaslan, Erik Wijmans, et al. *Habitat-Matterport 3D Dataset (HM3D): 1000 Large-scale 3D Environments for Embodied AI*. arXiv:2109.08238, 2021. Paper.

[17] Intern Robotics, Shanghai AI Laboratory. *InternVLA-N1: An Open Dual-System Vision-Language Navigation Foundation Model with Learned Latent Plans*. Technical report, September 2025. Technical report.

[18] Alibaba Cloud. *Qwen3.8-Max Model Information*. Alibaba Cloud Model Studio documentation, 2026. Snapshot: `qwen3.8-max-0902`. Official model documentation, accessed October 26, 2026.

[19] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, et al. *Habitat: A Platform for Embodied AI Research*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019. Paper.

[20] Mayank Mittal, Pascal Roth, James Tigue, et al. *Isaac Lab: A GPU-Accelerated Simulation Framework for Multi-Modal Robot Learning*. arXiv:2511.04831, 2025. Paper.

[21] Adam Paszke, Sam Gross, Francisco Massa, et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. Advances in Neural Information Processing Systems (NeurIPS), vol. 32, 2019. Paper.

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, et al. *Efficient Memory Management for Large Language Model Serving with PagedAttention*. ACM Symposium on Operating Systems Principles (SOSP), 2023. Paper.