

NAVGP-3: HARNESSING CONTEXT IN A HIERARCHICAL NAVIGATION RUNTIME

Gengze Zhou^{1,2†} Yicong Hong⁴ Jiazhao Zhang⁵ Xunyi Zhao¹ Jian Zhou¹
 Zixing Lei⁶ Zun Wang⁷ Chongyang Zhao⁸ Xionghui Chen⁵
 Stephen Gould^{2,3} Anton van den Hengel^{1,2} Qi Wu^{1,2†}
¹AIML, Adelaide University; ²Metacognition; ³ANU;
⁴Roblox; ⁵PKU; ⁶SJTU; ⁷UNC Chapel Hill; ⁸UNSW
[†]Correspondence to: {gengze.zhou, qi.wu01}@adelaide.edu.au
 Project Page: <https://metacognitionai.github.io/NavGPT3/>

ABSTRACT

Language models trained with long-horizon agentic reinforcement learning can generalize knowledge through reasoning, express precise actions, and pursue goals over many steps, raising the ceiling on what an embodied agent can understand and decide. Physical interaction, however, remains the domain of action policies, which provide dense, low-latency control. We present NAVGP-3, a harness that connects the two models, with an OS-like runtime built above it: reasoning, acting, and monitoring run as threads with their own context, tools, and permissions, while the runtime schedules them and decides which thread controls the robot’s motion, so that the robot can react to sudden real-world events through interruption and thread switching. Beneath it, our action policy NavGPT VLA, trained on 19.28M examples, allocates visual tokens using codec allocation, in proportion to scene change; its 8B model alone reaches 74.51 SR on R2R-CE and leads RxR-CE with 78.19 SR. With the complete harness, NAVGP-3 sets the state of the art on R2R-CE (81.51 SR) and, for the first time, brings an autonomous agent to human level: on RxR-CE it matches human followers in success (90.43 vs. 90.4 SR) and path fidelity (78.47 vs. 77.7 nDTW) at 1 min 22 s per episode, versus roughly 3 min for a human. We comprehensively ablate the harness design and the interaction between the two models, showing how tools and the action policy shape the path from language-model reasoning to physical control: when NavGPT VLA executes the route, the reasoning loop shortens and the system’s minimum reaction time falls from 3–19 s per language-model decision to 0.5–1 s per action-policy step (1–2 Hz). These results show that designing this embodied interface is central to connecting frontier language-model intelligence with low-level physical control. We will release all models, code, and evaluation records.

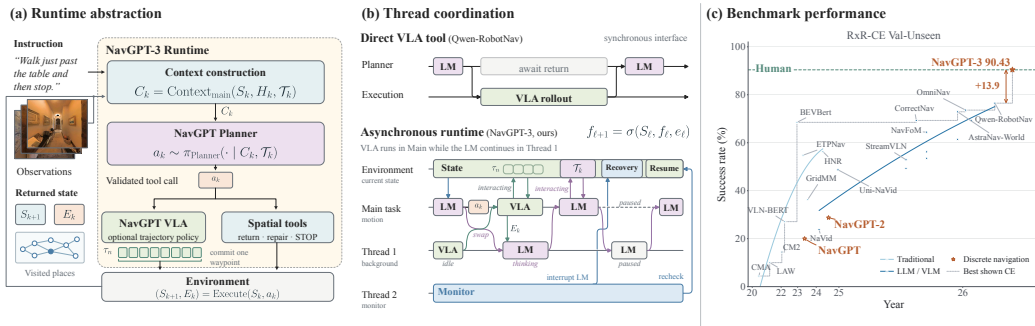


Figure 1: Overview of NAVGP-3: (a) runtime abstraction with the planner, VLA, and spatial tools; (b) synchronous tool use versus asynchronous thread coordination; and (c) benchmark performance. NAVGP-3 reaches human performance on RxR-CE.

1 INTRODUCTION

Language models trained with long-horizon agentic reinforcement learning generalize knowledge through reasoning, express actions as tool calls and code, and pursue goals over many steps (OpenAI, 2026; Anthropic, 2026b;a; Qwen Team, 2026a). In Vision-and-Language Navigation (VLN), successive generations of language models absorb more of what navigation systems once built as separate modules: commonsense grounding, spatial description, progress estimation, and long-horizon tool use. Execution is the exception. It runs on the world’s clock rather than the model’s, and control must proceed at a rate that step-by-step reasoning cannot sustain. The world also does not pause while a model thinks: observations arrive during inference, motion continues past the decision that began it, and a hazard may demand a stop before any reasoning finishes. An action policy is also far smaller and weaker at following instructions than the model that plans, so it drifts over a long rollout and must report back evidence the reasoner can check. Existing systems address part of this by combining reasoning and execution hierarchically, with a planner issuing subgoals to a learned controller (Gao et al., 2025; Hu et al., 2026; Chiang et al., 2024; Huang et al., 2026b), and recent agents add graph memory and backtracking (Liu et al., 2026) or verification and recovery skills (Wang et al., 2026b).

Connecting these abilities to physical control therefore requires a complete interaction loop: grounded context for deciding, spatial tools for acting, persistent state for remembering, and execution evidence for revising a route. A *harness* shapes model behavior through this loop, defining the context models receive, the tools they can call, and the state and feedback retained across calls (Zhong & Zhu, 2026; DeepSeek-AI, 2026). Real robots add a coordination requirement: reasoning, execution, and monitoring must be able to progress asynchronously. A *runtime* above the harness manages these activities as threads and controls which thread may move the robot.

We present NAVGPT-3, which combines a complete navigation harness with an OS-like runtime. The harness brings context management, spatial tools, persistent route state, and model execution into one embodied interface. The Planner can act through those tools or delegate execution of a navigation instruction to NavGPT VLA. Returned route images, execution feedback, an observed map, and numbered visited places make the resulting rollout inspectable and revisable: the Planner can return to a visited place, repair the route locally, delegate again, and explicitly decide when the task is complete. We developed these harness interfaces through recursive self-improvement (RSI), using automated research iterations over tool definitions and visual prompts (Section 4.1). The runtime builds on these interfaces to schedule threads and control *motion authority*: review can overlap execution, and monitoring can interrupt motion without waiting for reasoning (Figure 2). Tool-capability and visual-context studies examine the harness design alongside Planner–VLA interaction (Sections 4.2 and 4.3). Beneath the harness, NavGPT VLA predicts eight waypoints from the instruction and visual history. Because the visual context is bounded, we use codec allocation from video coding: the first observation of each view is a full reference (I-frame), and later observations (P-frames), receive visual tokens in proportion to how much they change (Tang et al., 2026; An et al., 2026).

Across benchmarks, the 8B NavGPT VLA alone leads published methods on RxR-CE (78.19 SR); under the runtime, NAVGPT-3 sets the state of the art on R2R-CE (81.51 SR) and, on RxR-CE, matches human followers in both success (90.43 vs. 90.4 SR) and path fidelity (78.47 vs. 77.7 nDTW). The same runtime transfers to object-goal navigation and embodied question answering (Tables 6 to 8). Ablating the harness tools and the VLA interaction shows how this interface governs the path from reasoning to control: given good tools, a strong language model navigates well by itself but decides only once every 3–19 s, whereas letting the VLA execute the route lowers the system’s minimum reaction time to 0.5–1 s per step (1–2 Hz; Table 3).

2 RELATED WORK

Agentic Vision-and-Language Navigation with LLMs. VLN requires an agent to follow natural-language instructions in unseen environments, first on discrete navigation graphs (Anderson et al., 2018; Ku et al., 2020) and then in continuous environments (Krantz et al., 2020). NavGPT initiated the line of work asking how much of navigation a LLM can carry on its own (Zhou et al., 2024b): given observations in text, the language model reasons about instruction progress and de-

cides each move zero-shot, without any training on action data. Later navigators extend this recipe with maps, discussion, and open-vocabulary grounding (Chen et al., 2024; Pan et al., 2023; Long et al., 2023; Qiao et al., 2025), and recent agents let the model act on the embodiment directly through code (Zhou et al., 2026b). A parallel line of work targets physical execution, tuning Vision-Language Models (VLMs) into VLA policies that provide dense, low-latency control while retaining the generalization of the VLM (Zhang et al., 2024; 2025b; Cheng et al., 2025; Wei et al., 2025b;a; InternVLA-N1 Team, 2025; Zhang et al., 2025a; AMAP CV Lab, 2026; Qwen Team, 2026b; Mistral AI, 2026). NavGPT-2 pursues both at once, emphasizing the role of reasoning in language models while training a policy that acts faster and more accurately, with the language and action heads decoupled over one shared representation (Zhou et al., 2024a). However, such training cannot perfectly align the reasoning and the actions. NavGPT-3 combines the two at the system level, connecting a reasoning model and a trained VLA policy through a harness with an OS-like runtime built above it.

Language models over learned policies. Hierarchical systems pair a planning model with a learned controller in navigation (Chiang et al., 2024; Huang et al., 2026b; Liu et al., 2026) and manipulation (Gao et al., 2025; Hu et al., 2026; Guo et al., 2026; Li et al., 2026; Yang et al., 2026; Wang et al., 2026b), and harness engineering treats context, tools, and state as the unit of design (Zhong & Zhu, 2026; DeepSeek-AI, 2026). These systems already cover graph memory, backtracking, verification, and recovery. The closest system, NavMCP, retains trajectory evidence across calls to a navigation foundation model in order to answer questions (Lei et al., 2026). NavGPT-3 instead uses returned evidence to revise an instruction-following route: after a delegated VLA rollout, the Planner can inspect the route, return to any visited place, repair locally, and decide completion.

Visual context in navigation policies. Navigation policies must fit long, multi-view observation histories into a bounded visual budget. Existing policies compress history by merging or pruning visual tokens (Zhang et al., 2025b; Wei et al., 2025b) or distribute a fixed budget across time and views (Qwen Team, 2026b). NavGPT VLA allocates tokens in proportion to scene change, as codec-aligned video encoders do (Tang et al., 2026; An et al., 2026).

3 METHOD: HARNESS, RUNTIME, AND VLA CO-DESIGN

NAVGP-3 co-designs a complete harness between a reasoning model and a navigation policy: context and state management, spatial tools, validated execution, structured returns, route repair, and an explicit decision to end the task. A runtime built above the harness controls its threads. Part I describes these two layers; Part II describes the VLA beneath them.

3.1 PART I: THE NAVGPT-3 HARNESS AND RUNTIME

3.1.1 HARNESS AND RUNTIME ABSTRACTION

The harness shapes model behavior through the context, state, and tool interfaces that connect the Planner, the VLA, and the environment (simulator or robot). It builds the observations and history the models see, keeps spatial references and annotations, checks tool requests, executes permitted calls, and returns evidence for the next decision. The Planner decides which step to take next and when the task is complete; the VLA optionally provides high-frequency execution. Built above the harness, the OS-like runtime controls logical threads, their lifecycle and permissions, and motion authority. Figure 2 illustrates this coordination through route review and a protective monitor.

3.1.2 CONTEXT, SPATIAL TOOLS, AND EXECUTION RETURNS

The harness turns observations and interaction history into decision context, and exposes spatial tools that make this context actionable. Because invoking the Planner at every sensorimotor update is impractical, the harness additionally exposes a VLA for extended navigation through a higher-frequency perception-action loop. Invoking the VLA is optional: the Planner decides whether to use it or make all navigation decisions itself through the available tools. At invocation k , it selects

$$a_k \sim \pi_{\text{Planner}}(\cdot \mid C_k, \mathcal{T}_k), \quad (1)$$

where C_k is the harness-provided context, $\mathcal{T}_k$ the permitted tool set, and a_k a call including its arguments. The harness checks the selected call and executes it if the runtime permits; within these limits, the Planner determines the navigation strategy.

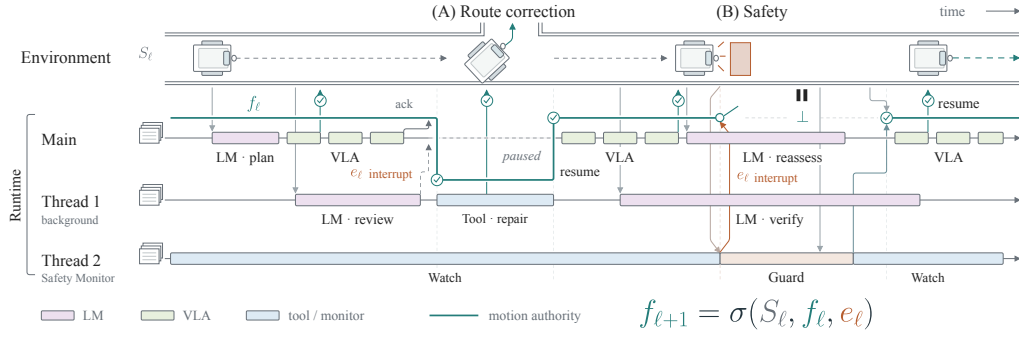


Figure 2: Illustrative foreground-control patterns in the NAVGPT-3 Runtime. Thin bars mark concurrent activity; the teal path denotes motion authority. (A) A route-review interrupt pauses main execution for correction, followed by resumption. (B) A safety interrupt revokes authority while inference continues; resumption requires fresh state and runtime authorization.

The interface groups operations into observation, instruction delegation, spatial revision, and semantic completion. Observation returns current views and observed geometry; delegation returns route keyframes and execution status; revision resolves visited-place references or local corrections into motion; annotation and termination preserve meaning and make completion explicit. These tools and their visual returns were iterated through RSI (Section 4.1). Their shared interface makes extended execution inspectable and revisable: a VLA stop ends one operation, while the Planner retains final task-stop authority (Appendix A).

3.1.3 EXECUTION COORDINATION AND MOTION AUTHORITY

Threads perform inference and receive observations at their own cadences, subject to their permissions, so activity on different rows of Figure 2 overlaps in time. The runtime controls which thread holds *motion authority*, the permission to issue task motion (teal path). Let $f_\ell \in \mathcal{T}_\ell \cup \{\perp\}$ denote the foreground thread at event index ℓ , where $\mathcal{T}_\ell$ is the thread set and $\perp$ denotes no holder. Control transfer follows

$$f_{\ell+1} = \sigma(S_\ell, f_\ell, e_\ell), \quad (2)$$

where S_ℓ is shared system state, including the environment (top row), e_ℓ an incoming event such as an interrupt (orange), and σ the runtime’s control-transfer rule, which uses preassigned event priorities to retain, transfer, or revoke authority. Protective events have highest priority; episode limits take precedence over routine review, and stale events cannot restore an obsolete operation (Appendix B.6). A handoff preserves thread contexts and takes effect only after runtime authorization and execution acknowledgement (checkmarks). Event order ℓ does not impose a common inference clock.

For a task-motion operation a proposed by thread j , foreground eligibility is

$$\text{eligible}_\ell(j, a) = \mathbf{1}[j = f_\ell \wedge a \in \mathcal{P}_j], \quad (3)$$

where $\mathcal{P}_j$ is the thread’s permitted operation set. Eligibility is necessary, not sufficient: execution also checks the current state and permissions (Appendix B.7). Motion authority does not grant exclusive computation or sensing, and calling the Planner or VLA within a thread requires no switch. This mechanism supports both concurrent progress and synchronization. For example, in route correction (Figure 2A), the VLA holds authority while the background thread reviews its progress; the review raises e_ℓ , and after the VLA acknowledges, authority passes to a tool repair while the main thread pauses, then returns so the VLA resumes. In the safety case (B), the monitor’s guard revokes authority, $f_{\ell+1} = \perp$, without awaiting LLM inference or the routine acknowledgement, while reasoning continues; the main thread resumes only with fresh state and authorization.

3.2 PART II: NAVGPT VLA

We train NavGPT VLA as a *versatile navigation foundation model*: one vision-language-action backbone that learns from several navigation tasks, camera configurations, and embodiments. The

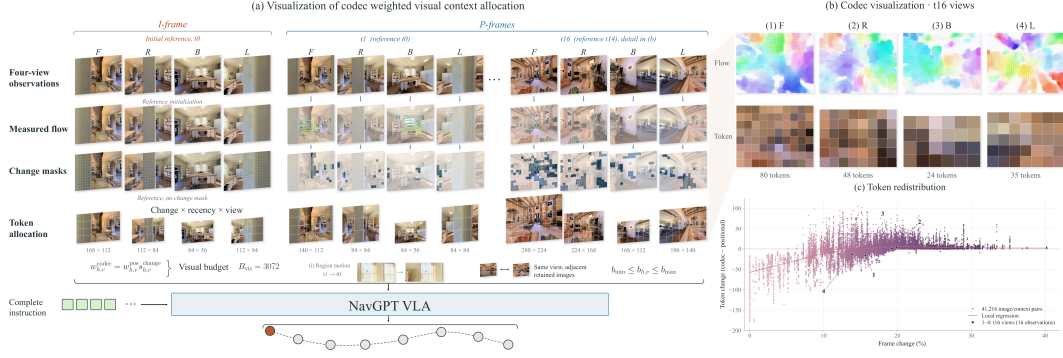


Figure 3: **Codec-weighted visual-context allocation.** (a) Change, recency, and view jointly determine whole-image resolution for a four-view history, with an initial I-frame and subsequent P-frames. (b) Optical-flow diagnostics and resulting token grids for the four views at t_{16} . (c) Allocation change relative to positional weighting versus RGB change $100\delta_{h,v}$ for 41,216 replayed image/context pairs from 13 recorded histories at $B_{\text{vis}} = 3072$. Curve: local regression; numbers match (b).

4B and 8B models are trained on instruction following, point-goal and object-goal navigation, target tracking, and outdoor driving, together with embodied question answering and visual grounding. Within NavGPT-3, this policy executes the navigation instructions issued by the Planner.

3.2.1 TRAJECTORY POLICY AND VISUAL-CONTEXT ALLOCATION

NavGPT VLA starts from Qwen3-VL-Instruct and adds an MLP action head. At VLA step n , it uses the provided instruction and up to 16 retained observations, each with front, right, back, and left views, to predict an ordered trajectory $\tau_n = ((x_{n,r}, y_{n,r}, \theta_{n,r}))_{r=1}^8$. Waypoint index r runs over eight cumulative targets in the current agent frame: x points forward and y points left (both in metres), and θ is relative heading in radians. Both 4B and 8B variants share this output format; execution commits only the first waypoint before the next prediction. Training combines waypoint regression with next-token supervision as detailed in Section 3.2.3.

The VLA sees up to N_{hist} retained observations of N_{view} views (F, R, B, L in Figure 3a) under a visual-token budget B_{vis} ; image (h, v) , view v of observation h , receives $b_{h,v}$ tokens. Our reimplementation of Qwen-RobotNav’s task-adaptive allocation (Qwen Team, 2026b) uses position alone: a weight $w_{h,v}^{\text{pos}}$ reflects recency and view direction, and an allocator converts the weights into integer counts with $b_{\min} \leq b_{h,v} \leq b_{\max}$ (Appendix C). Position records when and where an image was taken, but not whether it shows anything new.

Codec allocation adds a change score, inspired by I/P-frame video coding (Figure 3a). Within each view, we compare 64×64 RGB thumbnails of consecutive retained images: the first image (t_0) is an I-frame with full score, and later P-frames are scored by their mean absolute pixel difference $\delta_{h,v}$,

$$\delta_{h,v} = \frac{\|\text{thumb}(I_{h,v}) - \text{thumb}(I_{h-1,v})\|_1}{255 \cdot 3 \cdot 64^2}, \quad s_{h,v}^{\text{change}} = \begin{cases} 1, & \text{I-frame,} \\ \text{clip}(5\delta_{h,v}, 0.05, 1), & \text{P-frame,} \end{cases} \quad (4)$$

The codec weight $w_{h,v}^{\text{codec}} = w_{h,v}^{\text{pos}} s_{h,v}^{\text{change}}$ thus combines change, recency, and view (bottom row of Figure 3a); the allocator and token total are unchanged, and a constant score recovers the positional rule. Whether the score can act depends on the budget-to-capacity ratio $\rho_{\text{vis}} = B_{\text{vis}} / (N_{\text{hist}} N_{\text{view}} b_{\max})$: at $\rho_{\text{vis}} \geq 1$ every image can reach $b_{\max}$ and the two rules coincide, while below one tokens shift toward changing views. Training randomizes $B_{\text{vis}} \in [1024, 4096]$ and the allocation limits so that non-trivial histories enter this active regime (Table 13).

At t_{16} (Figure 3b), F, R, B, and L receive 80, 48, 24, and 35 tokens; the flow fields and change masks only visualize inter-frame differences, as allocation uses the thumbnail score in Eq. (4). Across recorded histories (Figure 3c), codec allocation takes tokens from low-change images and moves them to images that change.

3.2.2 TRAINING DATA TAXONOMY AND MIXTURE

A foundation action model inherits broad visual and linguistic knowledge from its pretrained backbone, but how reliably it acts depends on the range of situations its action data cover, and adding demonstrations of a single task does little to cover the edge cases a deployed policy meets. We therefore build the NavGPT VLA corpus around a taxonomy of coverage rather than the volume of any one benchmark. The taxonomy has four axes: the capability a task demands, the embodiment and observation interface through which it is performed, the conditions under which it occurs, and the visual-language understanding it presupposes. The first three are supervised through actions and the fourth through language. Every navigation source, whatever its origin, is converted to the same eight-waypoint target in the agent frame (Section 3.2.3), so coverage gained on any axis trains one shared action space rather than separate task-specific policies (Table 1 and Figure 4).

Capability. We span four capabilities that place progressively broader demands on the policy. Instruction following grounds guidance in motion, from step-by-step language to compact point goals that give only a direction or a relative coordinate. Active exploration, realized by object-goal navigation, removes the route altogether: the agent must search a partially observed scene and infer where a target is likely to be. Dynamic interaction, realized by target tracking, adds agents whose motion must be anticipated as it unfolds, while the target’s identity is kept among distractors. Cross-embodiment navigation, realized mainly by outdoor driving data from OpenScene (OpenScene Contributors, 2023) and nuScenes (Caesar et al., 2020), carries the same spatial planning to a vehicle in traffic, with a longer perception range, a different motion scale, and stricter safety constraints. Along this sequence, uncertainty about the environment, dependence on history, and embodiment diversity roughly increase. The capabilities are not disjoint: harder regimes reuse simpler skills while adding their own perception and control demands. Joint training over this overlap is intended to yield spatial planning that transfers across task families instead of being fit to each one separately.

Embodiment and observation interface. A policy that serves several platforms must act from whatever cameras a platform carries. The most economical way to cover this axis is to re-render existing episodes rather than collect new ones. Each R2R-CE and RxR-CE training episode (Krantz et al., 2020; Ku et al., 2020) therefore appears both as a four-view panorama from front, right, back, and left cameras and as a front-camera sequence, with identical waypoint targets, so the policy learns one behavior from surround and forward-only input. Front-view trajectories from VLNVerse (Lin et al., 2025), robot-navigation demonstrations from other platforms such as RoboCasa (Nasiriany et al., 2024), and the driving data (OpenScene Contributors, 2023; Caesar et al., 2020) extend the same interface to further simulators, bodies, and camera rigs. Within simulation, object-goal episodes start from random poses with randomized camera height and field of view, so a single source already spans several sensor placements.

Conditions and edge cases. Within a task, failures concentrate where the training distribution is thin: unfamiliar phrasing, appearance, and scenes; states off the expert path; and rare actions. We widen each of these while keeping the supervision fixed where possible. For language and appearance, we refine the instructions in R2R-CE and RxR-CE episodes with Gemini 3.1 Pro (Google DeepMind, 2026) and with subsets in VLN and tracking data whose rendered observations from habitat are refined with Qwen-Image-Edit (Wu et al., 2025) to look closer to real camera images. These variants change what the policy reads or sees but not the route it must take. For scenes, SRDF (Wang et al., 2025) supplies instruction–trajectory pairs in the HM3D environments (Wang et al., 2023), which span many more scenes than the R2R training split in MP3D. We rebuild these data in two ways. First, the original instructions are often inaccurate, so we regenerate all of them with Gemini 3.1 Pro (Google DeepMind, 2026), which rewrites each instruction from the trajectory video, the shortest-path video to the same goal, and the original instruction. Second, for state coverage we keep the DAgger rollouts alongside the shortest-path routes but change their supervision. The original data label each visited state with the rollout’s own next step, a pseudo action that need not move the agent toward the goal; we instead label every visited state with the direction of the shortest path from that position to the goal. The relabeled rollouts thus cover off-path states that shortest-path demonstrations never visit and supervise the recovery from each of them. For actions, geometry-aware resampling reduces the forward-motion bias of raw trajectories and raises the share of turns and stops (Section 3.2.3). The point-goal data additionally vary how fast the agent moves and how open the space around it is. Standard episodes advance about 0.25 m per step, whereas

low-speed episodes advance about 0.1 m, so the same displacement unfolds over more than twice as many observations and the policy must judge progress from much smaller changes between frames. Open-space episodes place goals at round metric offsets, such as 4 m ahead and 1 m to the left, in open areas of a scene, where walls and corridors no longer channel the route and the policy must reach the goal from the coordinates alone.

Visual-language understanding. Action supervision teaches where to move, but not what is in a scene or where a named object appears in the image. Language-supervised data, trained with next-token prediction only, cover this axis, and among them we found grounding data to be the most important. We draw it from LocateAnything-Data (Wang et al., 2026a), a corpus of 138M language queries with 785M boxes over 12M images. Our 19 subsets contribute 2.31M effective records, 12.0% of the complete mixture and second only to four-view instruction following. They span object detection (40.0% of these records, including egocentric, part-level, person, and driving scenes), referring-expression grounding (18.2%), point and affordance localization (13.7%), GUI element grounding (12.4%), text localization (11.0%), and document layout (4.6%). Every example pairs a language query with the boxes or points it refers to, which trains the backbone to map words to precise image locations across natural, egocentric, robotic, driving, and screen images. Navigation needs the same ability whenever an instruction names a landmark or a target object, and waypoint supervision alone does not provide it, since it ties language to motion rather than to explicit image positions. The remaining language-supervised data add spatial reasoning from LLaVA-OneVision-2 (An et al., 2026), question answering and dialogue on navigation episodes, and general vision-language instruction data that retain the backbone’s broader abilities while it learns to act.

Mixture composition. The complete mixture draws on 83 sources grouped into 11 families (Table 1) and contains approximately 19.28M effective records per epoch. As shown in Figure 4, 65.6% of these records supervise action regression and 34.4% supervise next-token prediction only. Spatial reasoning and grounding together contribute approximately 18% of the records, and general vision-language data 8.5%.

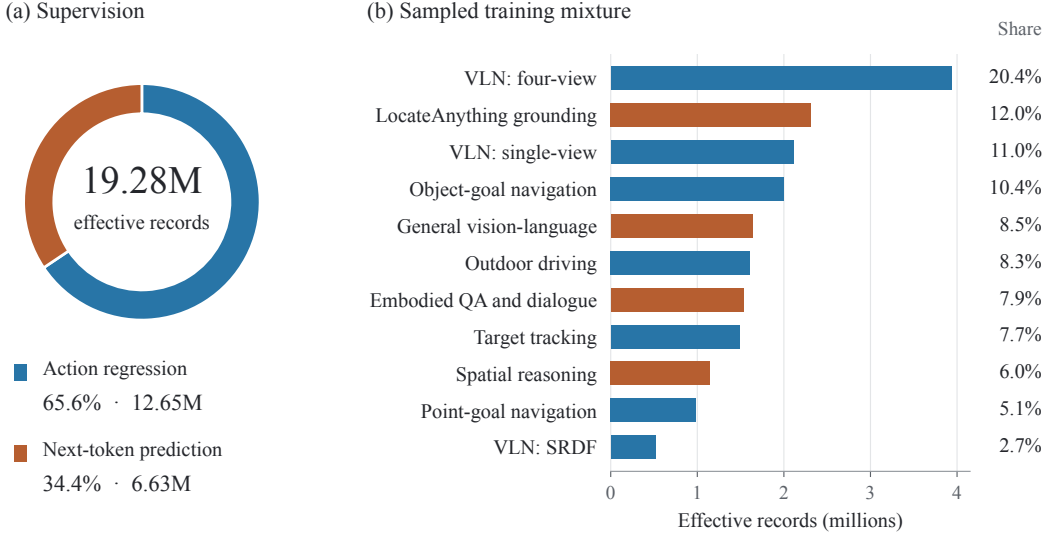


Figure 4: **NavGPT VLA training mixture.** Supervision split and effective records by data family.

Sampling and counting. Most sources are used in full, and several large sources are subsampled (Table 1): the SRDF navigation sources keep 15%, the SRDF-based dialogue sources 10%, the two largest general vision-language sources 30%, and the largest spatial-reasoning and LocateAnything sources between 5% and 60%. For source d with n_d records, sampling fraction $s_d \in (0, 1]$, and repetition count r_d , the effective size $\tilde{N}_d = n_d s_d r_d$ is the number of samples it contributes to one epoch, and a family’s share is its summed effective size divided by $\sum_d \tilde{N}_d$. Effective records count training samples, not unique trajectories, scenes, or datasets: the four-view and single-view

R2R/RxR data render the same episodes, the refined-instruction and refined-observation variants reuse their routes, and SRDF builds on HM3D environments.

Table 1: **Training-data mixture.** Sizes are effective records per epoch, in millions.

Family	Sources	Sampling	Size (M)
<i>Action regression</i>			
VLN: four-view	7	100%	3.938
VLN: single-view	5	100%	2.113
VLN: SRDF	2	15%	0.516
Point-goal navigation	5	100%	0.984
Object-goal navigation	1	100%	2.000
Target tracking	6	100%	1.486
Outdoor driving	5	100%	1.608
<i>Next-token prediction</i>			
Embodied QA and dialogue	12	10–100%	1.532
General vision-language	13	30–100%	1.640
Spatial reasoning	8	25–100%	1.149
LocateAnything grounding	19	5–100%	2.311
Total	83		≈19.28

3.2.3 DATA PROCESSING AND TRAINING

Navigation sources are converted to a common multimodal format containing an instruction, an ordered image sequence, and an eight-waypoint target. Four-view Habitat observations are rendered in front/right/back/left order, while single-view sources retain the front camera; the first and current observations are always retained and intermediate history is sampled up to 16 steps. Habitat discrete actions are simulated as 0.25 m forward motion, 15-degree turns, or padded stops and transformed into cumulative agent-local coordinates normalized by $(1.0, 0.433, 2.094)$ for (x, y, θ) . Geometry-aware resampling reduces the raw forward-action bias and increases turns and stops. Embodied navigation QA, general vision-language, spatial-reasoning, and grounding examples use the same image-and-conversation format but omit the trajectory target and train only the next-token objective.

Both model sizes are initialized from Qwen3-VL-Instruct and trained for one epoch on the complete mixture with the same global batch size, schedule, codec allocation, and action output; model and action-head learning rates are 5×10^{-6} and 10^{-3} , respectively (Table 2). The ablations in Sections 4.6 and 4.7 train additional models on smaller cumulative scopes of this mixture.

Table 2: **Matched VLA training configurations.** BS: global batch size; AH LR: action-head learning rate.

Backbone	GPUs	BS	Steps	Length	LR	AH LR
Qwen3-VL-4B	32	256	~75.3k	8,192	5×10^{-6}	10^{-3}
Qwen3-VL-8B	32	256	~75.3k	8,192	5×10^{-6}	10^{-3}

Supervision and masking. Training combines trajectory regression with vision-language co-training (Qwen Team, 2026b), using the coordinate normalization and training configuration above. The MLP action head predicts a 24-dimensional vector containing eight (x, y, θ) waypoints. For a minibatch with navigation examples $\mathcal{B}_{\text{nav}}$, its loss is

$$\mathcal{L}_{\text{nav}} = \frac{1}{24|\mathcal{B}_{\text{nav}}|} \sum_{i \in \mathcal{B}_{\text{nav}}} \sum_{r=1}^8 \|\hat{\mathbf{a}}_{i,r} - \mathbf{a}_{i,r}\|_2^2, \quad \mathcal{L} = \mathcal{L}_{\text{nav}} + \mathcal{L}_{\text{NTP}}. \quad (5)$$

Both losses have unit weight. Position and heading have equal weight after normalization; the coordinate scales therefore determine their relative weight in physical units. The fixed eight-waypoint

target includes padded stop positions near trajectory ends, and the MLP loss applies no additional terminal-waypoint mask. Navigation examples are excluded from next-token supervision. Conversely, QA and grounding examples have no action target and contribute only cross-entropy over labeled answer tokens; prompt and padding tokens are excluded from the loss. Their empty trajectory targets are masked out rather than learned as zero-motion targets. An absent example type contributes zero to its corresponding loss.

4 EXPERIMENTS

This section evaluates the NAVGPT-3 harness, the collaboration with the VLA, and the complete system. In Section 4.1, we first show the development process of the harness tools and of the visual context shown to the model by automated recursive self-improvement on a subset of training episodes. In Section 4.2, we evaluate the resulting tools: we add the VLA, Map, and Backtrack tool groups to the basic tools for two Planners and measure accuracy, cost, and reaction time. In Section 4.3, we vary how tool returns are presented to the Planner, and in Section 4.4, how often the Planner checks VLA execution. In Section 4.5, we compare the complete system with published methods on R2R-CE and RxR-CE and transfer it to object-goal navigation and embodied question answering. In Sections 4.6 and 4.7, we study NavGPT VLA on its own: its codec allocation and its scaling with data and model size. Finally, in Section 4.8, we deploy the system on a Unitree Go2 robot, and in Section 4.9, we show qualitative episodes.

For the navigation performance, we evaluate NAVGPT-3 on the full Val-Unseen splits of R2R-CE and RxR-CE (Krantz et al., 2020; Anderson et al., 2018; Ku et al., 2020), on object-goal navigation (Batra et al., 2020) in MP3D (Chang et al., 2017) and HM3D v2 (Ramakrishnan et al., 2021), and on embodied question answering (Ren et al., 2024; Zhai et al., 2025; Jiang et al., 2025); ablation studies use a fixed R2R subset of 100 episodes across ten MP3D scans (Qiao et al., 2025). We report success rate (SR), success weighted by path length (SPL), normalized dynamic time warping (nDTW), oracle success (OSR), and navigation error (NE), with accuracy and normalized steps for question answering. Unless varied, harness studies use the 8B NavGPT VLA. Each Planner runs inside its provider’s scaffold, Codex for GPT-6 Astra and Claude Code for Claude Opus 5, which assembles messages and manages the conversation, including automatic compression of long contexts. NAVGPT-3 replaces the system prompt, tools, observations, spatial state, and stopping rule; differences between the two frameworks are described in Appendix D.

4.1 HARNESS DEVELOPMENT THROUGH RECURSIVE SELF-IMPROVEMENT

We formulate recursive self-improvement (RSI) of the harness as a KDLoop-style tree search over executable harness versions (Zhou et al., 2026a). Each node specifies the Planner’s system prompt, the available tools, the persistent spatial state, and the formats in which observations and execution results are shown to the Planner; each edge is a candidate revision. THINK proposes edits using the current harness and accumulated episode evidence; CRITIC checks them against interface constraints and earlier failures; EXPERIMENT implements and scores the surviving branches; and DISTILL records confirmed findings, failed edits, and unresolved hypotheses. The highest-scoring candidate becomes the next harness, while REFLECT revisits whole categories of changes when progress stalls. Search operates at development time: the chosen interface is fixed during evaluation, without online Planner or VLA weight updates. Figure 5(a) summarizes the research loop; panel (b) illustrates three categories of changes, to tool interfaces, visual context, and execution results, which branch from a common harness, are revised, and merge the selected branch into the next best harness.

The search evaluates candidate harnesses with Claude Opus 5 on one fixed 100-episode subset sampled from the R2R-CE training split, whose scenes are disjoint from Val-Unseen, using success rate as the selection signal and episode traces to diagnose failures. Figure 5(c) shows the 14 harness configurations that were evaluated on all 100 episodes and reached $\text{SR} \geq 50\%$. The frontier is the best score reached so far, and dashed orange segments mark drops between consecutive candidates.

These configurations trace the harness’s development: front-view observation and primitive stepping score 66% SR; VLA delegation scores 72% with the 4B policy and 76% with the 8B policy; and combining VLA execution with spatial tools reaches a development best of 81%. An early place-graph interface scores 54%, and the consolidated interface scores 77%. One review found that the

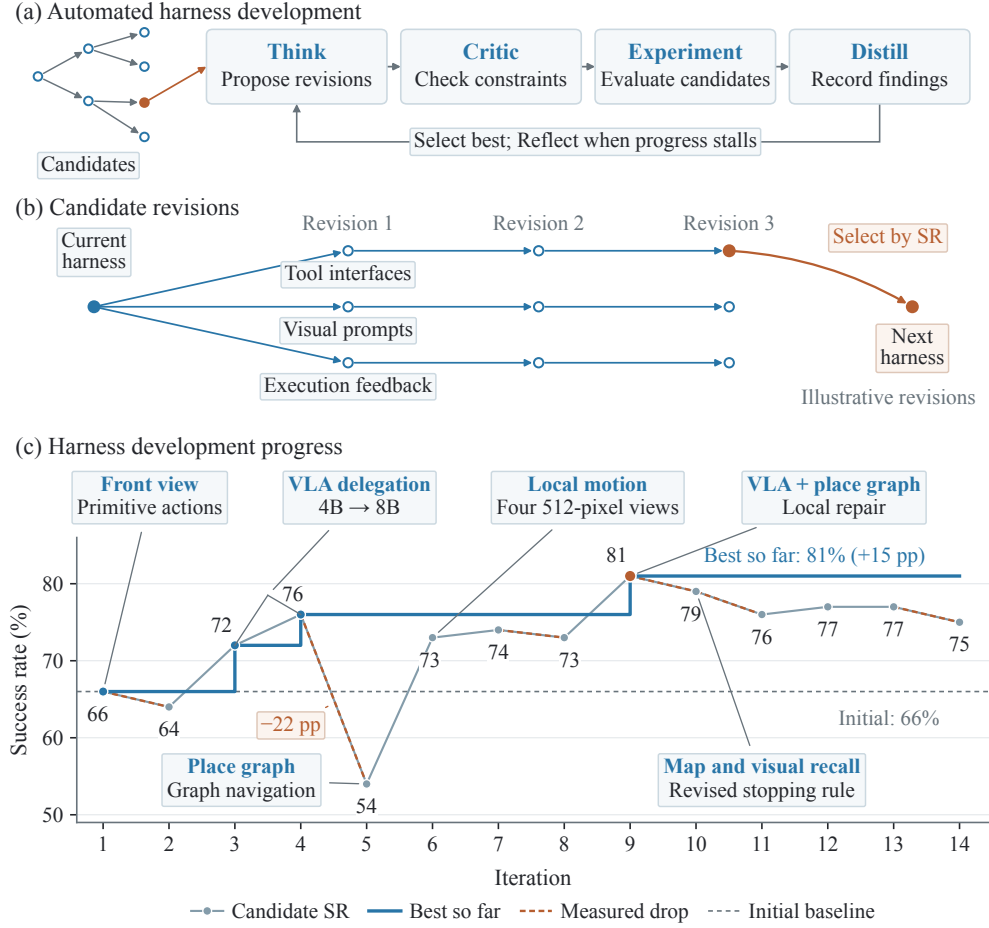


Figure 5: Harness development through recursive self-improvement: the research loop, candidate revisions, and success-rate progression.

task prompt described local movement, although no movement tool was available; later revisions added that tool and replaced the stitched panorama with four separate 512-pixel views. The resulting tool and visual-prompt studies are reported in Table 3 and Figure 6, and the resulting tool interface in Table 4.

4.2 HARNESSING AND VLA ABLATION

Table 3 measures how much each part of the harness contributes and what the VLA adds on top. Two language-model Planners, GPT-6 Astra and Claude Opus 5, navigate the same 100 R2R-CE episodes under the same episode limits and with the same 8B NavGPT VLA. Only the tools exposed to the Planner change between rows, and the task prompt describes only those tools. The tools and their visual returns were developed through RSI (Section 4.1).

What the Planner can do in each configuration. The eight tools fall into four groups (Table 4). Every configuration includes the four *basic* tools: `observe_forward` and `observe_panorama` to look at the current surroundings, `navigate_relative` to turn and move by a given angle and distance, and `terminate_episode` to end the episode. With the basic tools alone, the Planner moves the agent itself, so every environment action is a Planner decision. Three groups can be added. The *VLA* group, `navigate_by_instruction`, hands the full instruction, or the part that remains, to NavGPT VLA, which drives the agent until it signals a stop or

Table 3: **Tool and VLA ablation on the R2R-CE subset.** Resource use is averaged per episode. Opus costs are as reported by the provider’s SDK; Astra costs are estimated from recorded usage at list prices. Inference excludes communication and harness overhead. Reaction is the mean interval between environment actions: Planner time per turn without the VLA, VLA inference per step with it.

Planner	Tools			Navigation				Resource use				
	VLA	Map	Backtrack	NE↓	OSR↑	SR↑	SPL↑	Tokens (k)↓	Turns↓	Cost (\$) ↓	Infer. (s) ↓	React. (s) ↓
VLA only	✓	–	–	4.23	78	71	66.14	–	–	–	44.1	0.78
GPT-6 Astra	–	–	–	4.67	74	72	62.26	325.0	32.14	0.593	105.7	3.3
GPT-6 Astra	–	✓	✓	2.55	87	83	69.04	190.8	10.71	0.469	66.8	6.2
GPT-6 Astra	✓	–	–	4.58	79	75	67.83	227.9	3.37	0.469	84.4	0.78
GPT-6 Astra	✓	✓	–	4.01	79	76	67.35	201.4	3.42	0.443	80.4	0.78
GPT-6 Astra	✓	✓	✓	3.34	84	82	69.12	155.0	3.88	0.391	79.1	0.78
Opus 5	–	–	–	5.43	70	66	55.24	721.0	47.93	0.69	425.9	8.9
Opus 5	–	✓	✓	3.45	85	73	44.82	483.4	16.75	0.990	323.4	19.3
Opus 5	✓	–	–	3.78	80	76	61.85	199.8	11.19	0.353	201.5	0.78
Opus 5	✓	✓	–	4.04	85	81	69.29	211.8	8.96	0.393	199.4	0.78
Opus 5	✓	✓	✓	3.69	83	79	66.70	185.6	8.50	0.397	191.8	0.78

Group	Tool	Effect on state	Returned to the Planner
Basic	observe_forward()	None	Current forward RGB for fine semantic inspection.
	observe_panorama()	Updates observed local coverage, not pose	Four labelled views with bearings, clearance, pose displacement, and node information.
	navigate_relative()	Executes a validated local turn and translation	Realized motion status, endpoint panorama, and nearby node candidates.
	terminate_episode()	Optionally returns to a selected node, then stops	Stop or return status; this call ends the episode and cannot be undone.
VLA	navigate_by_instruction()	Runs one VLA rollout; the VLA keeps its history	Up to eight sampled route keyframes, endpoint panorama, optional observed map, and execution status.
Map	observe_map()	Updates observed coverage, not pose	Observed-only top-down map, map status, and route-ordered node table.
Backtrack	navigate_to_node()	Returns to a visited node or a way-point of the latest rollout	Endpoint panorama, observed map, return status, and updated node table.
	annotate_node()	Attaches a label to an existing node	Annotation status and updated node table.

Table 4: Tools that implement the navigation capabilities, grouped as in the ablation of Table 3.

reaches the delegation limit; the Planner then receives up to eight keyframes of the executed route, the endpoint views, and the execution status. A VLA stop ends only the delegation, and the episode ends only when the Planner calls `terminate_episode`. The *Map* group, `observe_map`, returns a top-down map of the area observed so far, on which visited places are numbered in visiting order. The *Backtrack* group, `navigate_to_node` and `annotate_node`, lets the Planner return the agent to any numbered place and label places with what it has recognized there, so the map becomes something it can act on rather than only read.

Resource use. Each row reports per-episode averages. Tokens count all input, including cached input, and all output; turns count Planner responses separated by environment feedback; cost is the provider’s charge; and inference time is the total model inference time, excluding communication and harness overhead. Reaction time is the mean interval between environment actions: the Planner’s time per turn without the VLA, and the VLA’s time per step with it (Appendix D).

The harness sets how well the language model navigates. With only the basic tools and without specific harnesses, both Planners perform poorly and spend heavily: GPT-6 Astra reaches 72 SR at 325.0k tokens per episode and Claude Opus 5 reaches 66 SR at 721.0k (Table 3). A more robust, higher-quality harness changes this. Mapping tools lets the Planner assemble a much stronger spatial

context than the current view provides, and Backtrack lets it act on that context far more efficiently. Together they raise success to 83 and 73 SR while cutting tokens by 41% and 33%, and Astra’s tools-only row is the most accurate in the table.

Whether the VLA helps depends on the language model. The VLA always shortens the reasoning loop, but it does not always improve without a good harness. For Opus it does: turns fall from 16.75 to 8.50–11.19, success rises from 73 to 76–81 SR, and cost falls from \$0.99 to \$0.35–0.40 per episode. For Astra it does not: turns fall from 10.71 to 3.37, yet success drops to 75 SR and cost stays at \$0.469, as the tokens saved on navigation are spent supervising and correcting the VLA’s execution. A stronger language model thus absorbs more of the benefit the VLA would otherwise provide, and the upper bound on success is set by the language model itself: once it reads the environment well enough to decide accurately on its own, success is limited by its own navigation ability rather than by the interface. Once the VLA is given its own harness, with Mapping and Backtracking over its rollouts, token use reaches its minimum for both models (155.0k for Astra and 185.6k for Opus) with success close to each model’s best.

The VLA sets the minimum reaction time. Without the VLA, the system acts only as often as the language model decides, once every 3–19 s on average depending on the model and tools. With the VLA, the control loop runs at VLA inference rate instead, 0.78 s per step on average for the 8B model, which gives the whole system a basic guarantee of real-time response (Table 3). Section 4.4 examines how much of this higher interaction rate the Planner can exploit by checking VLA execution at increasing frequencies.

4.3 VISUAL PROMPT ORGANIZATION

The harness also decides how tool returns are shown to the Planner: as a visual prompt of camera views and an observed map, annotated with view labels, node IDs, heading, and turn cues. Figure 6 varies one presentation factor at a time against this grounded reference while keeping tools fixed (Appendix D.5): caption labels move the annotations into adjacent text, compact glyphs use a smaller digit font, heading-up rotates the map with the agent, and absolute bearings replace relative turn cues with fixed-frame directions. Both language models are robust to how the visual prompt is organized: SR stays within 76–78% for Astra and 73–77% for Opus 5 across all five versions.

What changes is how much reasoning a version demands, and that demand does not follow human readability. Absolute bearings, the hardest version for a person to read, use the fewest tokens for both models, while the most expensive version differs between them: compact glyphs for Astra and the grounded reference for Opus, within 160–210k and 139–173k mean cumulative tokens per episode, respectively. Strong language models are therefore robust to visual context, and what they read easily does not follow human readability.

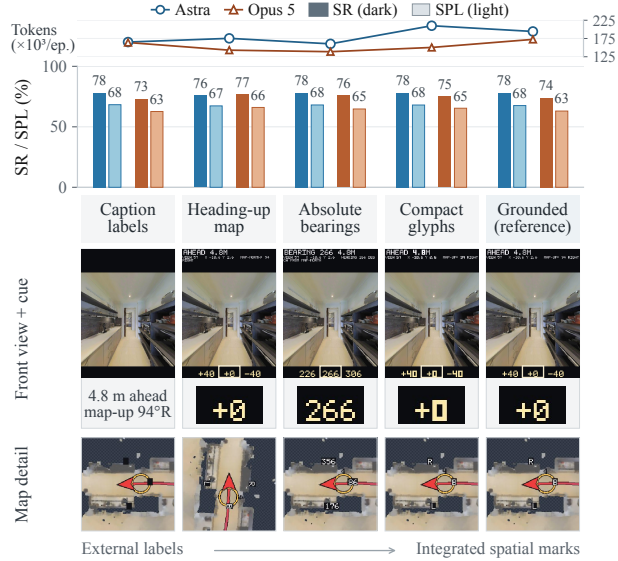


Figure 6: **Visual prompt organization.** SR/SPL and token usage for Astra and Opus 5.

Table 5: **Check frequency.** VLA steps per check.

Frequency	NE↓	OSR↑	nDTW↑	SR↑	SPL↑	Tokens (k)↓
None	3.82	81	16.74	77	67.51	179.2
32	3.56	81	16.66	78	68.88	1,529.9
16	2.01	85	18.72	83	77.36	2,319.5

Table 6: **Vision-and-language navigation.** Evaluation results on full R2R-CE and RxR-CE Val-Unseen splits. [†]RxR Val-Unseen English human followers (Ku et al., 2020), measured in DE. Both NAVGPT-3 systems use the 8B VLA.

Method	Model	R2R-CE Val-Unseen				RxR-CE Val-Unseen			
		NE↓	OSR↑	SR↑	SPL↑	NE↓	nDTW↑	SR↑	SPL↑
Human [†] (Ku et al., 2020)	–	–	–	–	–	1.32	77.7	90.4	–
NavFoM (Zhang et al., 2025a)	7B	4.61	72.1	61.7	55.3	4.74	65.8	64.4	56.2
ABot-N0 (AMAP CV Lab, 2026)	4B	3.78	70.8	66.4	63.9	3.83	–	69.3	60.0
AstraNav-World (Chen et al., 2026)	3B+5B	3.86	73.9	67.9	65.4	3.82	–	72.9	61.5
OmniNav (Xue et al., 2025)	3B	3.74	74.6	69.5	66.1	3.77	–	73.6	62.0
Qwen-RobotNav (Qwen Team, 2026b)	4B	3.80	77.2	69.5	63.6	3.80	71.9	75.2	65.0
Qwen-RobotNav (Qwen Team, 2026b)	8B	3.53	78.5	72.1	66.6	3.58	72.5	76.5	65.7
Robostrat Navigate (Mistral AI, 2026)	8B	3.20	81.3	77.4	74.2	3.47	–	75.1	68.7
<i>Ours: executable navigation policy</i>									
NavGPT VLA	4B	3.41	79.91	72.54	67.19	3.35	73.33	76.77	67.77
NavGPT VLA	8B	3.29	80.19	74.51	68.54	3.05	74.85	78.19	68.98
<i>Ours: complete navigation system</i>									
NAVGPT-3	Opus 5	2.77	84.07	79.01	70.16	2.27	75.09	84.80	69.85
NAVGPT-3	GPT-6 Astra	2.18	86.99	81.51	70.42	1.56	78.47	90.43	74.31

4.4 COMMUNICATION FREQUENCY AND COST

Table 5 varies how often the Planner checks VLA execution, measured in VLA forward passes per check; each pass predicts eight waypoints and executes the first (Section 3.2.1). Without periodic checks, the Planner regains control only when the VLA stops or reaches its delegation limit, after a median of 51 steps (90th percentile 123). At each check, the Planner reviews the route so far while the VLA keeps executing, and interrupts it only to revise or terminate the route. We test GPT-6 Astra with shared tools and instructions.

More frequent checks improve the route, but only at a steep cost. Checking every 32 steps gives most episodes at most one check and barely changes results (78 vs. 77 SR). Checking every 16 steps, about three times per episode, raises SR to 83 and SPL from 67.51 to 77.36, and cuts NE from 3.82 to 2.01 m. Tokens grow much faster than accuracy, because each check re-reads the accumulated route context: they rise from 179.2k to 1,529.9k per episode at 32 steps, and to 2,319.5k at 16. Periodic checks thus buy accuracy with costly supervision, which motivates event-triggered review for real-world deployment.

4.5 OVERALL NAVIGATION PERFORMANCE

Table 6 compares the complete NAVGPT-3 system with published methods on the full R2R-CE and RxR-CE Val-Unseen splits. Without any harness, NavGPT VLA is already at the state of the art on RxR-CE. The runtime then raises R2R-CE success from 74.51 to 81.51 SR with GPT-6 Astra, and gains more on RxR-CE, where success reaches 84.80 and 90.43 SR against 78.19 for the specialist alone and nDTW improves to 78.47: RxR instructions are longer and specify the route in far more detail, which gives the returned route evidence, returns to recorded places, and local corrections more to act on. NAVGPT-3 matches RxR human followers in success (90.43 vs. 90.4 SR) and path

Table 7: **Object-goal navigation.** Prior methods use HM3D v1, others HM3D v2. †: 500-episode subsets, scored with a stricter stop criterion. ‡: depth or odometry.

Model	MP3D		HM3D	
	SR↑	SPL↑	SR↑	SPL↑
<i>Prior methods: HM3D v1</i>				
VLFM [‡]	36.4	17.5	52.5	30.4
SG-Nav [†]	40.2	16.0	54.0	24.9
WMNav [‡]	45.4	17.2	58.1	31.2
CogNav [‡]	46.6	16.1	72.5	26.2
Uni-NaVid	–	–	73.7	37.1
Qwen-RobotNav 4B	52.2	16.0	75.6	30.6
Qwen-RobotNav 8B	48.8	17.7	71.2	33.0
NAVGPt-3 [†]	59.2	25.2	75.3	40.8

Table 8: **Embodied question answering.** Accuracy and normalized steps (Zhai et al., 2025) on HM-EQA (Ren et al., 2024) and MT-HM3D (Zhai et al., 2025); LLM score and path efficiency on EXPRESS-Bench (Jiang et al., 2025). *: from prior work. NAVGPt-3 uses a GPT-6 Astra judge.

System	HM-EQA		MT-HM3D		EXPRESS	
	Acc.↑	Steps↓	Acc.↑	Steps↓	Score↑	Path eff.↑
<i>Prior systems</i>						
Explore-EQA (Ren et al., 2024)	58.4	0.52	36.2*	0.64	–	–
Graph-EQA (Saxena et al., 2024)	63.5	0.20	45.6*	0.45	–	–
Memory-EQA (Zhai et al., 2025)	61.4	0.40	43.1	0.41	–	–
Fine-EQA (Jiang et al., 2025)	56.0	0.54	–	–	63.95	25.58
3D-Mem (Yang et al., 2025)	50.4	0.63	–	–	–	–
FAST-EQA (Zhang et al., 2026)	69.2	0.65	50.5	0.52	68.7	29.25
<i>Qwen-RobotNav 8B executor</i>						
NavMCP (Lei et al., 2026)	76.7	0.15	54.4	0.19	79.27	33.96
NAVGPt-3	77.38	0.16	55.27	0.18	80.43	34.26

fidelity (78.47 vs. 77.7 nDTW), the first autonomous agent to reach human-level performance on this benchmark (Ku et al., 2020).

Object navigation and embodied question answering. Table 7 evaluates NAVGPt-3 on 500-episode subsets of the MP3D and HM3D v2 validation splits against published full-split results, scoring success only when the agent stops within the distance threshold while facing the target. Both cross-task studies use the 8B NavGPT VLA under a Claude Opus 5 Planner. The two benchmarks locate the runtime’s contribution differently: MP3D spans 21 goal categories, many of them small or fine-grained (cushion, towel, picture), and the gain appears in success, whereas HM3D v2 uses six large, visually distinctive categories on which published systems already succeed in roughly three of four episodes, and the gain appears in path efficiency. On embodied question answering (Table 8), NAVGPt-3 is level with the strongest prior system, Qwen-RobotNav (Lei et al., 2026): accuracy is ahead on both datasets at comparable normalized steps (Appendix D.3).

4.6 CODEC ACROSS TRAINING SCOPES

Figure 7 compares *Without codec*, the positional allocation of our reimplement of the Qwen-RobotNav observation interface (Qwen Team, 2026b), with *With codec*, our change-aware allocation, for 4B and 8B policies trained on three cumulative scopes, labelled VLN, + Nav. QA (adding embodied navigation question answering), and + Cross-embod. (further adding cross-embodiment navigation). All policies are evaluated standalone on the full R2R-CE and RxR-CE Val-Unseen splits.

Codec allocation helps in every 8B setting and on RxR-CE at both sizes. At 8B it raises R2R-CE SR by 1.7, 5.7, and 2.1 points across these three scopes, respectively, and RxR-CE SR by 1.6, 2.6, and 2.6 points; the large gain after adding navigation QA partly reflects a weaker positional baseline on that scope. At 4B it raises RxR-CE SR by 3.7, 0.5, and 1.4 points, whereas on R2R-CE it leaves SR within one point on the first two scopes (−0.1 and −0.8) and improves it only with cross-embodiment data (+1.1). With cross-embodiment data, codec allocation improves both SR and SPL on both benchmarks at both model sizes.

We see three likely reasons for this pattern. First, codec allocation matters most when images compete for a binding budget ($\rho_{\text{vis}} < 1$, Section 3.2.1). RxR-CE routes are longer and their instructions describe more landmarks along the way, so more steps run with a full history, and keeping detail on the views that change pays off; this is consistent with the gains on RxR-CE at both sizes. Second, the value of a change-aware rule grows with the visual diversity of the training data. In VLN data, every episode comes from one simulator with four fixed cameras and one motion scale, so how much a frame changes is largely predictable from its time and camera, and the positional rule is already close to the codec rule; Adding navigation QA brings question answering on the same

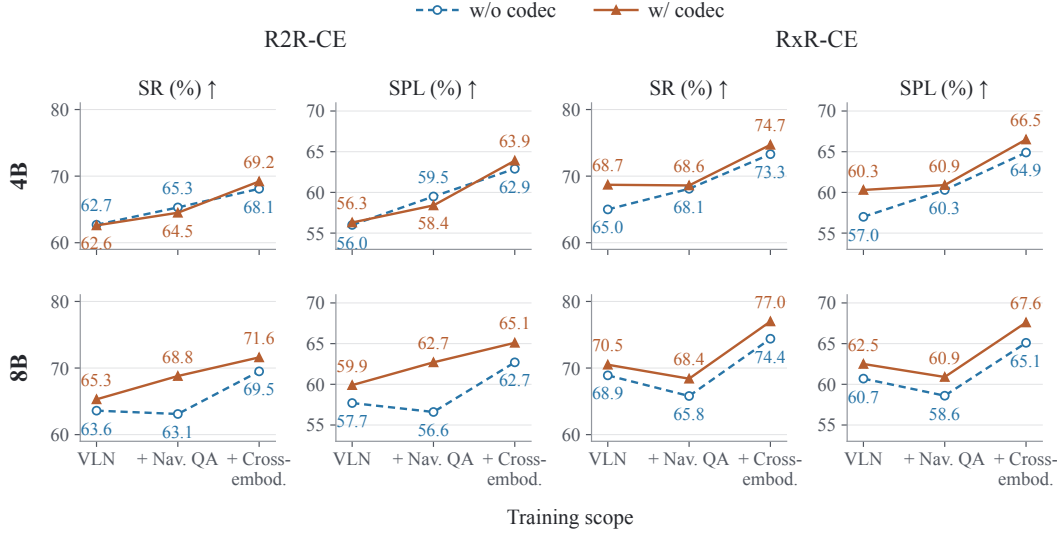


Figure 7: Codec comparison across training scopes.

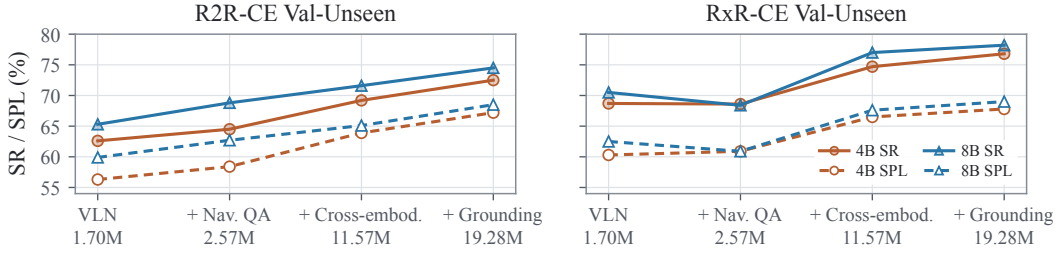


Figure 8: Model and data scaling.

episodes and therefore no new visual conditions. Cross-embodiment training adds single-camera views, slow and fast motion, moving targets, randomized camera heights and fields of view, and outdoor driving, where the amount of change between frames differs widely across sources. A fixed positional rule cannot adapt to these differences, whereas codec allocation follows the change in each sample. Third, codec allocation presents images at varying resolutions, which the model must learn to read. The 8B model exploits this on every scope, whereas the 4B model gains on R2R-CE only once cross-embodiment training supplies enough varied visual data.

4.7 MODEL AND DATA SCALING

Figure 8 plots 4B and 8B policies trained with codec across four cumulative training scopes on full Val-Unseen. Its first three scopes correspond to the *With codec* curves in Figure 7; the fourth adds visual grounding and is the complete approximately 19.28M-record mixture used in the main comparison.

NavGPT VLA improves steadily with both training data and model size. Data contributes most: growing the mixture elevenfold, from 1.70M VLN records to the complete 19.28M, raises R2R-CE SR from 62.6 to 72.54 at 4B and from 65.3 to 74.51 at 8B, and RxR-CE SR from 68.7 to 76.77 and from 70.5 to 78.19. R2R-CE SR improves at every step; on RxR-CE, adding navigation QA leaves SR flat at 4B and lowers it at 8B (70.5 to 68.4), and the large gain arrives with cross-embodiment data. Model size adds a smaller but consistent margin: the 8B model leads the 4B model in R2R-CE SR at every scope, by 2.0 to 4.3 points. The data effect therefore outweighs doubling the parameters. With cross-embodiment data, the 4B model (69.2 SR) surpasses the 8B model trained on VLN only (65.3), and the complete 4B model (72.54) surpasses the 8B model trained without the final grounding step (71.6). The data curve has not saturated: the final step, which adds visual grounding

and grows the mixture from 11.57M to 19.28M records, still raises R2R-CE SR by 3.3 and 2.9 points at 4B and 8B, and the 8B margin persists at the largest scale, suggesting further gains from more data and larger models. Because each step changes data volume and source composition together, these curves measure the cumulative recipe rather than the value of any single source.

4.8 REAL-WORLD DEPLOYMENT

We deploy NAVGPT-3 on a Unitree Go2 with RGB input, remote or onboard VLA inference, and a runtime LiDAR hazard monitor. In a matched comparison (Table 10), overlapping reasoning with priority monitoring raises SR from 73.3 to 83.3 over sequential queued handling and cuts p95 halt latency from 1840 to 180 ms.

4.8.1 DEPLOYMENT SETUP

Robot and sensing. We use the Unitree Go2 platform and onboard VLA inference. NavGPT VLA receives the robot’s front-facing RGB stream, with visual context allocated over the retained single-view history. This differs from the four-view simulator configuration in Section 3.2.1. The onboard LiDAR supplies geometric hazard detections to our runtime monitor; it is not an additional input to the VLA.

Remote and onboard inference. In the remote configuration, the robot compresses and transmits observations together with the Planner-selected navigation instruction to a server, which predicts waypoints and returns them for execution. The onboard configuration runs the 4B NavGPT VLA on an NVIDIA Jetson Thor with FP8 quantization and TensorRT acceleration, following [Qwen Team \(2026b\)](#). Both configurations expose the same waypoint interface to the harness. Onboard inference removes the network round trip for VLA predictions; the hosted Planner remains a separate service. Model inference, communication, and physical execution contribute separately to end-to-end response time and are measured separately below.

Harness and runtime integration. The harness manages observations, spatial state, tool calls, and execution returns, while the runtime above it coordinates independently paced reasoning, execution, and monitoring. Moving the VLA between the server and the robot does not change this division of responsibilities. On the robot, every task-motion command passes through a final command filter, which forwards it only while the issuing operation holds the runtime’s permission to move. The robot evaluation tests whether a LiDAR event can revoke that permission without waiting for the Planner, and whether execution resumes only after fresh sensing and a new permission (Appendix B.6).

4.8.2 GLASS-DOOR EXECUTION VISUALIZATION



Figure 9: Recorded glass-door execution. The outlined fourth panel highlights the near-door forward pause, followed by lateral correction and continued passage.

Approach and pause. Figure 9 follows a recorded instruction to enter a shop. Panels 1–3 show the approach from an indoor concourse toward the already-open glass doorway, beside a freestanding sign. Panel 4 (5.48 s) highlights the interruption of forward progress near the glass. Sensor poses place the pause and correction between 5.48 and 7.36 s; lateral movement continues, so the robot is not completely stationary.

Correction and continuation. Panel 5 (6.85 s) shows the lateral adjustment. Forward motion resumes in panel 6 (7.71 s), the robot passes the entrance in panel 7, and panel 8 shows it inside the shop without visible contact with the glass door. The connected points illustrate an interrupted

approach followed by revised paths through the opening: the task remains to enter the shop while the immediate route changes.

Connection to runtime control. This sequence illustrates the pause–revise–continue pattern targeted by the runtime: suspend the active motion operation, inspect fresh observations, and resume with an updated path under renewed authorization.

4.8.3 RUNTIME EVALUATION

Controlled inputs. Conditions share the forward RGB stream, LiDAR detector and timestamped events, speed limit, routes, and manufacturer safeguards. The VLA placement and precision are held fixed within each runtime comparison, together with the single-camera input configuration, VLA checkpoint, Planner prompt, detector threshold, and firmware. An independent operator stop remains active; operator intervention counts as a failed trial.

Table 9: Go2 scenarios and measurements.

Scenario	Controlled event	Primary measurements
A: Route revision	A detour or endpoint correction becomes necessary during an extended rollout; vary event timing relative to Planner inference.	Completion SR; event-to-authorized-repair time; route efficiency; Planner calls; execution idle time.
B: LiDAR protection	A soft obstacle enters a calibrated protected region during motion, including while the Planner is busy. Use matched harmless passages as negatives.	Missed and false stops; trigger-to-halt p50/p95; post-trigger distance; minimum clearance; operator interventions.
C: Priority and recovery	Trigger a hazard during repair or near VLA completion; delay a model reply, re-send an outdated command to the robot’s command filter, then clear the hazard.	Accepted stale commands; event ordering; halt latency; unauthorized resumption; correct resumption and task completion.

Comparisons. We compare VLA-only and Planner-with-tools baselines with a matched 2×2 runtime comparison: sequential versus overlapping Planner–VLA execution, crossed with queued versus priority monitor handling. All four runtime configurations share the same models, tool definitions, route evidence, and LiDAR detections. Queued handling processes a detected event at the next regular control step; priority handling immediately revokes the active permission at the command filter. Keeping a priority monitor in the sequential condition tests whether protection can be explained by a simple independent stop gate, while the overlap contrast tests progress during reasoning. The final runtime configuration combines overlap with priority handling. These controls separate detector quality, scheduling, and authority transfer.

Table 10: Go2 runtime results.

Condition	A: Revision		B: Protection		C: Recovery	
	SR	Repair (s)	Missed/ N	Halt p95 (ms)	Stale/ N	Resume SR
VLA-only	60.0	–	–	–	–	–
Planner + tools	70.0	14.2	9/30	2150	11/30	53.3
Sequential + queued	73.3	12.8	7/30	1840	8/30	60.0
Sequential + priority	76.7	11.5	2/30	420	3/30	70.0
Overlapping + queued	76.7	9.6	5/30	1310	6/30	66.7
Overlapping + priority	83.3	7.9	0/30	180	1/30	80.0

Results. Overlap and priority contribute separately (Table 10). Overlapping Planner reasoning with VLA execution shortens repair from 12.8 to 9.6 s under queued handling and from 11.5 to 7.9 s under priority handling, because execution continues while the Planner reviews the route. Priority handling reduces missed stops from 7/30 to 2/30 in the sequential runtime and from 5/30 to 0/30 with overlap, and lowers p95 halt latency from 1840 to 420 ms and from 1310 to 180 ms, respectively. Combining both yields the highest revision SR (83.3 vs. 73.3 for sequential queued handling), the fewest accepted stale commands (1/30), and the highest resume SR (80.0).

Event timing and outcomes. We log observation compression, network transfer, and VLA inference durations separately, and record sensor capture, detector trigger, review start and end, permission revocation, the last accepted nonzero command, stop-command dispatch, physical halt, renewed permission, and resumed motion on a single synchronized clock. A stop command accepted by the robot’s software interface does not by itself count as a halt: the robot has halted only when independently measured speed stays below a threshold, fixed before the trials, for a fixed interval. A stale command is one issued under a revoked permission version; *Stale/ N* counts trials in which such a command was accepted. Resume success requires fresh sensing, a new permission, and completion of the remaining task without operator intervention.

Trial allocation and uncertainty. Five pilot trials per condition and scenario validated logging and are excluded from the comparison. The evaluation uses five routes, three event timings, and two repetitions per condition and scenario (30 trials), randomized in matched blocks. Detection false negatives and false positives are annotated independently of the detector’s output.

4.9 QUALITATIVE RESULTS

Four selected successful episodes illustrate how returned observations guide the Planner after extended VLA execution. Claude Opus 5 recognizes that the VLA has passed the hallway table, retraces 8.54 m to node 1, and verifies a 1.2 m endpoint correction (Figure 10). On a longer route, it returns from a dead-end nook and directly completes the omitted cabinet and doorway clauses (Figure 11). GPT-6 Astra recovers from stalled execution: after a repeated VLA call and a blocked local move, it retreats to node 3 and takes the other aisle (Figure 12). Finally, Opus delegates again when local stair attempts leave the instruction unfinished; the next rollout reaches the upstairs bathroom (Figure 13).

Each case includes its complete instruction. Maps T1 onward show the position after each tool call returns; numbered views are sampled from the first VLA rollout. Solid lines mark the thread that holds motion authority, and dashed lines mark waiting. Token counts include cached input and output.

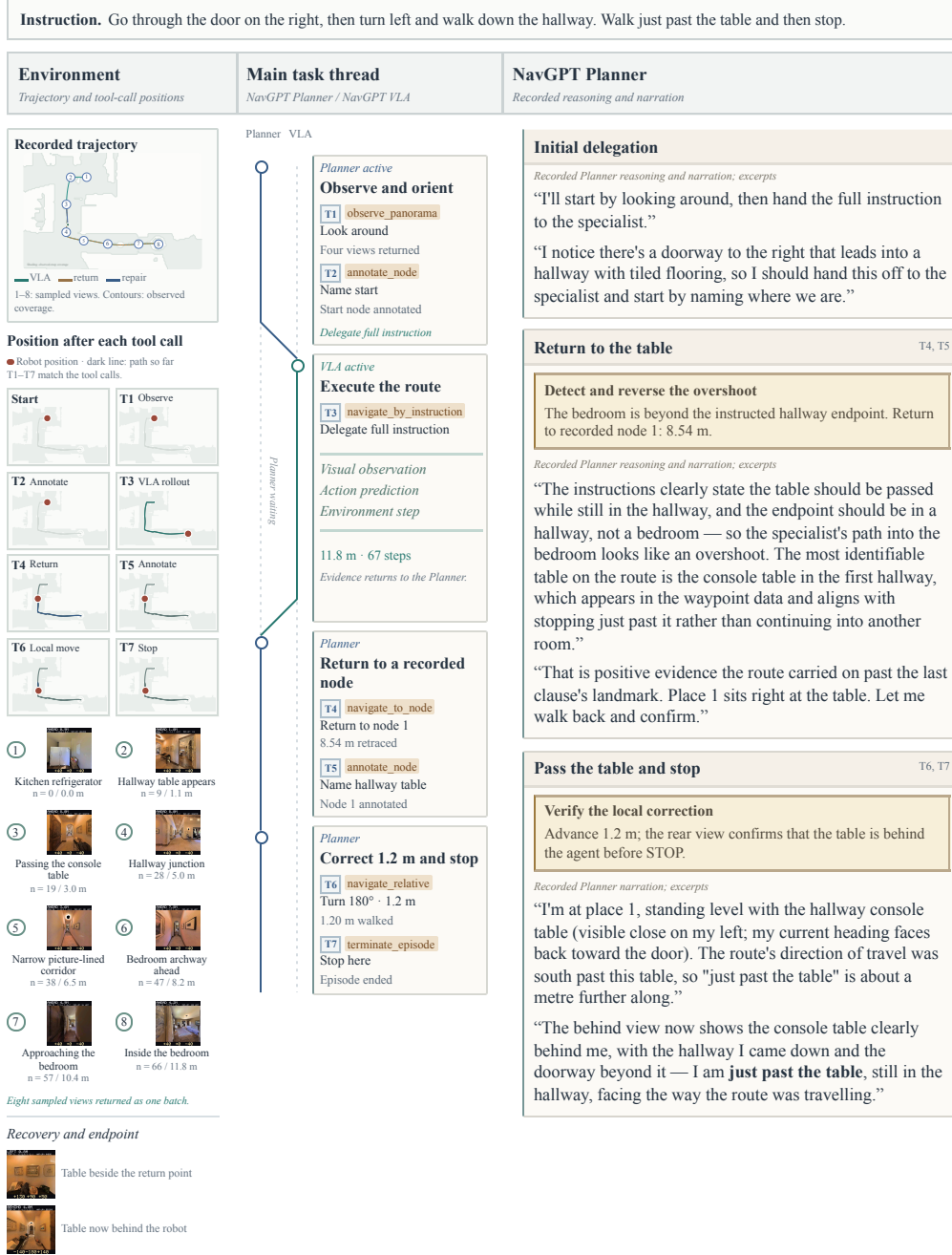


Figure 10: **Backtracking to an overshoot endpoint.** Claude Opus 5 on R2R-CE, episode 144. The Planner identifies the hallway table before the VLA’s bedroom stop, retraces 8.54 m to node 1, and verifies a 1.2 m local correction before stopping.



Figure 11: **Completing omitted route clauses.** Claude Opus 5 on RxR-CE, episode 2523. The Planner returns from a dead-end nook to node 8, then uses relative motion to reach the gray cabinet and the open doorway with a rising staircase.



Figure 12: **Recovering from a blocked aisle.** GPT-6 Astra on R2R-CE, episode 43. After repeated VLA execution and a direct motion attempt stall, the Planner returns to node 3, traverses the other aisle, and stops beyond the table beside the lamp.



Figure 13: **Resuming delegation after unsuccessful local correction.** Claude Opus 5 on RxR-CE, episode 5782. With the stair clause still unfinished, the Planner delegates the full instruction again. The second rollout reaches the upstairs bathroom, which the Planner verifies before stopping.

5 CONCLUSION

We presented NAVGPT-3, a complete navigation harness with a runtime layer above it. The harness organizes context and navigation state, exposes spatial tools, and returns execution evidence for repairing the route and deciding when the task is complete; the runtime coordinates these capabilities through thread control and motion authority. The standalone VLA and complete system attain strong full-split navigation results, with the latter matching human followers on RxR-CE under continuous rather than graph-based control. The harness studies show how spatial tools support Planner navigation, visual presentation affects processing cost, and VLA delegation provides frequent action updates with fewer reasoning turns. Together, these components establish a complete embodied interface through which reasoning can inspect, revise, and control extended physical execution. The harness provides the structure for integrating stronger reasoning and action models as they become available.

AI USE STATEMENT

Large language models are components of the studied system: GPT-6 Astra and Claude Opus 5 act as Planners, and a GPT-6 Astra judge scores EXPRESS-Bench answers, as described in Section 4 and Appendix D. Separately, we used generative AI coding assistants (Claude Code and Codex) to help implement the experiment and evaluation infrastructure, to create and revise scientific figures, and to draft, edit, and format parts of this paper. We have reviewed all AI-assisted work: AI-assisted code was tested before use, and all text, figures, and reported numbers were checked by the authors against the recorded experiment outputs. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

- AMAP CV Lab. ABot-N0: Technical report on the vla foundation model for versatile embodied navigation. *arXiv preprint arXiv:2602.11598*, 2026. 3, 13
- Xiang An, Yin Xie, Feilong Tang, Yunyao Yan, Huajie Tan, Didi Zhu, Changrui Chen, Xiuwei Zhao, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Kaichen Zhang, Wenkang Zhang, Zheng Cheng, Nansen Zhang, Chunsheng Wu, Chunjiang Ge, Zimin Ran, Dehua Song, Chunyuan Li, Shikun Feng, Ming Hu, Zhangquan Chen, Junbo Niu, Bo Li, Ziyong Feng, Ziwei Liu, Zongyuan Ge, and Jiankang Deng. LLaVA-OneVision-2: Towards next-generation perceptual intelligence. *arXiv preprint arXiv:2605.25979*, 2026. 2, 3, 7
- 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 *CVPR*, pp. 3674–3683, 2018. 2, 9
- Anthropic. Claude Fable 5.1 and Claude Mythos 5.1 system card. <https://www.anthropic.com/claude-fable-and-mythos-5-1>, 2026a. System card released 2026-09-01. 2
- Anthropic. Claude Opus 5 system card. <https://www-cdn.anthropic.com/b514064af1408018e64blad24e7d5e75850b4ffd/Claude%20Opus%205%20System%20Card.pdf>, 2026b. System card released 2026-07-24. 2
- Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects. In *arXiv:2006.13171*, 2020. 9
- Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11621–11631, 2020. 6
- Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niebner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. In *2017 International Conference on 3D Vision (3DV)*, pp. 667–676. IEEE, 2017. 9
- Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee K Wong. Mapgpt: Map-guided prompting for unified vision-and-language navigation. *arXiv preprint arXiv:2401.07314*, 2024. 3
- Jintao Chen, Junjun Hu, Haochen Bai, Minghua Luo, Xinda Xue, Botao Ren, Chengyu Bai, Shichao Xie, Ziyi Chen, Fei Liu, Zedong Chu, Xiaolong Wu, Mu Xu, and Shanghang Zhang. AstraNav-World: World model for foresight control and consistency. *arXiv preprint arXiv:2512.21714*, 2026. 13
- 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 *Robotics: Science and Systems (RSS)*, 2025. 3
- Hao-Tien Lewis Chiang, Zhuo Xu, Zipeng Fu, Mithun George Jacob, Tingnan Zhang, Tsang-Wei Edward Lee, Wenhao Yu, Connor Schenck, David Rendleman, Dhruv Shah, Fei Xia, Jasmine Hsu, Jonathan Hoech, Pete Florence, Sean Kirmani, Sumeet Singh, Vikas Sindhwani, Carolina Parada, Chelsea Finn, Peng Xu, Sergey Levine, and Jie Tan. Mobility VLA: Multi-modal instruction navigation with long-context VLMs and topological graphs. *arXiv preprint arXiv:2407.07775*, 2024. 2, 3
- DeepSeek-AI. DeepSeek Harness: Everything is a plugin. <https://github.com/deepseek-ai/deepseek-harness>, 2026. Accessed 2026-08-29. 2, 3

-
- Alexandre Défossez, Laurent Mazaré, Manu Orsini, Amélie Royer, Patrick Pérez, Hervé Jégou, Edouard Grave, and Neil Zeghidour. Moshi: a speech-text foundation model for real-time dialogue. *arXiv preprint arXiv:2410.00037*, 2024. URL <https://arxiv.org/abs/2410.00037>. 32
- Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, Zeyu Jiang, and Lin Shao. VLA-OS: Structuring and dissecting planning representations and paradigms in vision-language-action models. *arXiv preprint arXiv:2506.17561*, 2025. 2, 3
- Google DeepMind. Gemini 3.1 Pro model card. <https://deepmind.google/models/model-cards/gemini-3-1-pro/>, 2026. 6
- Pengyuan Guo, Zhonghao Mai, Zhengtong Xu, Kaidi Zhang, Quan Khanh Luu, Heng Zhang, Zichen Miao, Arash Ajoudani, Zachary Kingston, Qiang Qiu, and Yu She. PPlanAR: Planning-language-grounded agentic reasoning for robot manipulation. *arXiv preprint arXiv:2602.01662*, 2026. 3
- Jiaheng Hu, Mohit Shridhar, Caden Lu, Dhruv Shah, Hao-Tien Lewis Chiang, Jie Tan, and Annie Xie. What matters in orchestrating robot policies: A systematic study of hierarchical VLA agents. *arXiv preprint arXiv:2606.10267*, 2026. 2, 3
- Muye Huang, Lingling Zhang, Xingyu Yu, Lei Shi, Zhanyu Ma, Jun Xu, Jiuchong Gao, Jinghua Hao, Renqing He, and Jun Liu. DuplexOmni: Real-time listening, seeing, thinking, and speaking for full-duplex interaction. *arXiv preprint arXiv:2606.09186*, 2026a. URL <https://arxiv.org/abs/2606.09186>. 32
- Xiangyun Huang, Xiangchen Wang, Runfeng Lin, Yihao Xu, Kangyu Huang, Hengchen Jiang, Xiwang Dong, and Jiarong Lin. From routes to steps: Separating semantic progress from local execution in vision-and-language navigation. *arXiv preprint arXiv:2608.03143*, 2026b. 2, 3
- InternVLA-N1 Team. InternVLA-N1: An open dual-system navigation foundation model with learned latent plans. Technical report, Shanghai AI Laboratory, 2025. URL https://internrobotics.github.io/intervla-n1.github.io/static/pdfs/InternVLA_N1.pdf. 3
- Kaixuan Jiang, Yang Liu, Weixing Chen, Jingzhou Luo, Ziliang Chen, Ling Pan, Guanbin Li, and Liang Lin. Beyond the destination: A novel benchmark for exploration-aware embodied question answering. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9091–9101, 2025. 9, 14
- Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*, pp. 104–120. Springer, 2020. 2, 6, 9, 34
- 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 (EMNLP)*, pp. 4392–4412, 2020. 2, 6, 9, 13, 14, 34
- Zixing Lei, Gengze Zhou, Xiong-Hui Chen, Jiazhaoh Zhang, Yiyang Huang, Hang Yin, Haoqi Yuan, Qi Wu, Weixin Li, and Siheng Chen. Scaffolding foundation models into physical-world agents pushes the frontier of long-horizon navigation. *arXiv preprint arXiv:2608.30396*, 2026. 3, 14
- Ruiying Li, Yunlang Zhou, YuYao Zhu, Kylin Chen, Jingyuan Wang, Sukai Wang, Kongtao Hu, Minhui Yu, Bowen Jiang, Zhan Su, Jiayao Ma, Xin He, Yongjian Shen, Yang Yang, Guanghui Ren, Maoqing Yao, Wenhao Wang, and Yao Mu. RoboClaw: An agentic framework for scalable long-horizon robotic tasks. *arXiv preprint arXiv:2603.11558*, 2026. 3
- Sihao Lin, Zerui Li, Xunyi Zhao, Gengze Zhou, Liuyi Wang, Rong Wei, Rui Tang, Juncheng Li, Hanqing Wang, Jiangmiao Pang, Anton van den Hengel, Jiajun Liu, and Qi Wu. VLNVerse: A benchmark for vision-language navigation with versatile, embodied, realistic simulation and evaluation. *arXiv preprint arXiv:2512.19021*, 2025. 6

-
- An Liu, Bingxi Liu, Hongyu Ding, Yixuan Jiang, Yaran Chen, Fulin Tang, Cong Leng, Hong Zhang, and Jian Cheng. HAM-VLN: Harnessing hierarchical agentic memory for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2607.29600*, 2026. 2, 3
- Yuxing Long, Xiaoqi Li, Wenzhe Cai, and Hao Dong. Discuss before moving: Visual language navigation via multi-expert discussions. *arXiv preprint arXiv:2309.11382*, 2023. 3
- Mistral AI. Robostrat Navigate. *arXiv preprint arXiv:2607.20785*, 2026. 3, 13
- Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems (RSS)*, 2024. 6
- OpenAI. GPT-6 Astra system card. <https://deploymentsafety.openai.com/gpt-6-astra>, 2026. Released 2026-09-03. 2
- OpenScene Contributors. OpenScene: The largest up-to-date 3D occupancy prediction benchmark in autonomous driving. <https://github.com/OpenDriveLab/OpenScene>, 2023. 6
- Bowen Pan, Rameswar Panda, SouYoung Jin, Rogerio Feris, Aude Oliva, Phillip Isola, and Yoon Kim. Langnav: Language as a perceptual representation for navigation. *arXiv preprint arXiv:2310.07889*, 2023. 3
- Yanyuan Qiao, Wenqi Lyu, Hui Wang, Zixu Wang, Zerui Li, Yuan Zhang, Minghui Tan, and Qi Wu. Open-Nav: Exploring zero-shot vision-and-language navigation in continuous environment with open-source LLMs. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2025. URL <https://arxiv.org/abs/2409.18794>. 3, 9, 35
- Qwen Team. Qwen3.8-Max: A new bar for coding and cowork. <https://qwen.ai/blog?id=qwen3.8>, 2026a. Released 2026-08-03. 2
- Qwen Team. Qwen-RobotNav technical report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026b. 3, 5, 8, 13, 14, 16, 33, 34
- Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 9
- Allen Z. Ren, Jaden Clark, Anushri Dixit, Masha Itkina, Anirudha Majumdar, and Dorsa Sadigh. Explore until confident: Efficient exploration for embodied question answering. In *Robotics: Science and Systems (RSS)*, 2024. 9, 14
- Saumya Saxena, Blake Buchanan, Chris Paxton, Bingqing Chen, Narunas Vaskevicius, Luigi Palmieri, Jonathan Francis, and Oliver Kroemer. GraphEQA: Using 3D semantic scene graphs for real-time embodied question answering. *arXiv preprint arXiv:2412.14480*, 2024. 14
- Feilong Tang, Xiang An, Yunyao Yan, Yin Xie, Bin Qin, Kaicheng Yang, Yifei Shen, Yuanhan Zhang, Chunyuan Li, Shikun Feng, Changrui Chen, Huajie Tan, Ming Hu, Manyuan Zhang, Bo Li, Ziyong Feng, Ziwei Liu, Zongyuan Ge, and Jiankang Deng. OneVision-Encoder: Codec-aligned sparsity as a foundational principle for multimodal intelligence. *arXiv preprint arXiv:2602.08683*, 2026. 2, 3
- Shihao Wang, Shilong Liu, Yuanguo Kuang, Xinyu Wei, Yangzhou Liu, Zhiqi Li, Yunze Man, Guo Chen, Andrew Tao, Guilin Liu, Jan Kautz, Lei Zhang, and Zhiding Yu. LocateAnything: Fast and high-quality vision-language grounding with parallel box decoding. *arXiv preprint arXiv:2605.27365*, 2026a. 7
- Wei Wang, Wenqiao Zhang, Yutong Lin, Yuqian Yuan, Tianwei Lin, Jinhao Mao, Zhenxuan Fan, Mingjian Gao, Yang Dai, Wentong Li, Zheqi Lv, Zheng Dong, Yingjie Niu, Jiaqi Zhu, Jun Xiao, Chao Li, and Yueting Zhuang. EmbodiedSkills: A unified framework for orchestrating, training, and deploying VLA agents. *arXiv preprint arXiv:2609.01281*, 2026b. 2, 3

-
- Zun Wang, Jialu Li, Yicong Hong, Yi Wang, Qi Wu, Mohit Bansal, Stephen Gould, Hao Tan, and Yu Qiao. Scaling data generation in vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12009–12020, 2023. 6
- Zun Wang, Jialu Li, Yicong Hong, Songze Li, Kunchang Li, Shoubin Yu, Yi Wang, Yu Qiao, Yali Wang, Mohit Bansal, and Limin Wang. Bootstrapping language-guided navigation learning with self-refining data flywheel. In *International Conference on Learning Representations (ICLR)*, 2025. 6
- Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, et al. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. *arXiv preprint arXiv:2512.08186*, 2025a. 3
- 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. *arXiv preprint arXiv:2507.05240*, 2025b. 3
- Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-Image technical report. *arXiv preprint arXiv:2508.02324*, 2025. 6
- Xinda Xue, Junjun Hu, Minghua Luo, Shichao Xie, Jintao Chen, Zixun Xie, Kuichen Quan, Wei Guo, Mu Xu, and Zedong Chu. OmniNav: A unified framework for prospective exploration and visual-language navigation. *arXiv preprint arXiv:2509.25687*, 2025. 13
- Tianshuo Yang, Guanyu Chen, Yutian Chen, Zhixuan Liang, Yitian Liu, Zanxin Chen, Chunpu Xu, Haotian Liang, Jiangmiao Pang, Yao Mu, and Ping Luo. HiVLA: A visual-grounded-centric hierarchical embodied manipulation system. *arXiv preprint arXiv:2604.14125*, 2026. 3
- Yuncong Yang, Han Yang, Jiachen Zhou, Peihao Chen, Hongxin Zhang, Yilun Du, and Chuang Gan. 3D-Mem: 3D scene memory for embodied exploration and reasoning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 17294–17303, 2025. 14
- Mingliang Zhai, Zhi Gao, Yuwei Wu, and Yunde Jia. Memory-centric embodied question answering. *arXiv preprint arXiv:2505.13948*, 2025. 9, 14
- Haochen Zhang, Nirav Savaliya, Faizan Siddiqui, and Enna Sachdeva. FAST-EQA: Efficient embodied question answering with global and local region relevancy. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 1664–1673, 2026. 14
- Jiazhaohao 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. *arXiv preprint arXiv:2402.15852*, 2024. 3
- Jiazhaohao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, et al. Embodied navigation foundation model. *arXiv preprint arXiv:2509.12129*, 2025a. 3, 13
- Jiazhaohao 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. *Robotics: Science and Systems*, 2025b. 3
- Hailin Zhong and Shengxin Zhu. AI Harness Engineering: A runtime substrate for foundation-model software agents. *arXiv preprint arXiv:2605.13357*, 2026. 2, 3
- Gengze Zhou, Yicong Hong, Zun Wang, Xin Eric Wang, and Qi Wu. NavGPT-2: Unleashing navigational reasoning capability for large vision-language models. In *European Conference on Computer Vision (ECCV)*, pp. 260–278. Springer, 2024a. 3
- Gengze Zhou, Yicong Hong, and Qi Wu. NavGPT: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 7641–7649, 2024b. doi: 10.1609/aaai.v38i7.28597. 2

Jian Zhou, Sihao Lin, Jin Li, Shuai Fu, Gengze Zhou, and Qi Wu. Automating the design of embodied agent architectures. *arXiv preprint arXiv:2606.30111*, 2026a. 9

Jian Zhou, Xunyi Zhao, Gengze Zhou, Zerui Li, Sihao Lin, Jiajun Liu, and Qi Wu. Embodied agents take control: Minimal-interface zero-shot agents rival industrial-scale policies in vision-and-language navigation. *arXiv preprint arXiv:2607.26148*, 2026b. 3

A NAVIGATION INTERFACE IMPLEMENTATION

Section 3.1.2 describes the navigation capabilities without reference to specific tool names. The full NAVGPT-3 configuration provides them through eight tools (Table 4). The harness checks each call’s tool name and arguments before executing it, the environment executes the resulting motion, and the harness keeps the records of the route and of visited places (nodes). The Planner cannot add tools through an instruction or change the navigation state directly.

A.1 DIVISION OF RESPONSIBILITIES

Figure 14 lists, for each part of the NAVGPT-3 system, the state it owns and the decisions it makes. This division is independent of the runtime’s threads and of where each model is deployed. The main task thread calls Planner reasoning, VLA execution, and other tools as its current operation. The Planner’s context is assembled from several sources: captured observations, the maintained route record, formatted tool results, and the conversation history kept by the model provider.

Responsibility	Component and role
Thread coordination	Runtime: schedule threads, manage permissions, and control motion authority.
Navigation decisions	Planner: interpret evidence, choose tools, and decide when the goal is reached.
Context construction	Harness: select and render the images and state shown to the models.
Persistent navigation state	Harness: keep route references, nodes, and annotations.
Tool checking	Harness: validate requests and execute permitted tool calls.
Low-level navigation	VLA: predict waypoints from its visual history.
Physical execution	Environment: execute motion and report sensing and collisions.

Figure 14: Division of responsibilities in the NAVGPT-3 system.

A.2 CAPABILITIES AND EXECUTION

Capability	Navigation purpose	What it takes and returns
Local observation	Ground landmarks and inspect the current surroundings	Relate forward or panoramic appearance to the current viewing direction and pose.
Instruction delegation	Execute an extended route with the learned policy	Pass the requested instruction to the VLA, keep its history across calls, and return images of the executed route with its status.
Route inspection	Relate current progress to previously encountered places	Show the observed map and the ordered list of visited places, without marking unobserved space as free.
Spatial revision	Revisit a recorded place or correct a local endpoint error	Turn a reference to a visited place, or a relative motion request, into checked movement and report what was executed.
Semantic annotation	Retain an interpretation of a visited place	Attach a label to a recorded place without changing its recorded position.
Task termination	Accept the final navigation state	Make episode completion explicit and distinct from the end of a delegated rollout.

Table 11: Navigation capabilities and their interfaces.

Execution and returned results. Instruction delegation runs as the main task thread’s current operation: it passes the instruction chosen by the Planner to the VLA, which keeps its history across steps, and executes the VLA’s predictions until the VLA signals a stop, the step limit for one delegation is reached, or the episode limit is reached. A VLA stop ends only the delegation, not the episode. The instruction may be the full task instruction or one for a subtask. The observed map distinguishes unobserved space from observed free space, and numbered nodes give stable targets

for returning. Annotations attach names, captions, or instruction clauses to existing nodes without changing the recorded map. Because ending the episode may include a return to a selected node, any restriction on returning applies to that option as well as to the dedicated return tool. The full configuration has no primitive motion tools, no tool for recalling arbitrary past images, and no object detector.

B RUNTIME STATE AND TIMING

This appendix expands Eqs. (1) to (3), separating the harness interfaces that the models see from the thread control performed by the runtime above them.

B.1 LOGICAL THREADS AND SHARED STATE

We use j for a logical thread, k for an invocation, and t for wall-clock time. Shared system state $S(t)$ combines the instructions and spatial records kept by the harness, observations from the environment, and the runtime’s records of threads and running operations. Each thread carries

$$S_j^{\text{thread}}(t) = (\text{role}_j(t), H_j(t), \text{status}_j(t), \mathcal{R}_j(t), \mathcal{W}_j(t)), \quad (6)$$

where role_j is its responsibility, H_j its working history, status_j its lifecycle state, and $\mathcal{R}_j, \mathcal{W}_j$ its permitted shared-state reads and proposed writes. Shared spatial history records the route and places, whereas working history records a thread’s own interactions. The runtime tracks the status of each tool call separately, so the main thread can ask the Planner to reassess while its VLA rollout continues or is paused; finishing a tool call does not mean that the task is complete.

The control-transfer rule σ retains, transfers, or revokes motion authority in response to events. A routine transfer withdraws the previous operation’s permission to move and waits until its current motion step has finished before permitting new motion. Thread contexts survive a handoff, but outdated commands do not become valid again when execution resumes.

B.2 SYNCHRONOUS IMPLEMENTATION

In a sequential loop, at handoff time t_k , with $S_k = S(t_k)$, main-thread history H_k , and available tools $\mathcal{T}_k$, each tool-call cycle is

$$\begin{aligned} C_k &= \text{Context}_{\text{main}}(S_k, H_k, \mathcal{T}_k), \\ a_k &\sim \pi_{\text{Planner}}(\cdot \mid C_k, \mathcal{T}_k), \quad (S_{k+1}, E_k) = \text{Execute}(S_k, a_k). \end{aligned} \quad (7)$$

The harness constructs context C_k , validates call a_k , executes it if permitted, and returns updated state and evidence E_k . The runtime controls thread execution and motion authority; the environment executes physical motion. A long operation such as a VLA rollout may span many observations and motion steps between two Planner calls. VLA execution and motion chosen directly by the Planner use the same interface.

B.3 CONTEXT CONSTRUCTION AND STATE UPDATES

For thread j at its k th invocation, the general context and decision interface is

$$C_{j,k} = \text{Context}_j(S|_{\mathcal{R}_j}, H_j, \mathcal{E}_{j,k}, \mathcal{T}_j), \quad a_{j,k} \sim \pi_{j,k}(\cdot \mid C_{j,k}, \mathcal{T}_j), \quad (8)$$

evaluated when the invocation starts, where $\mathcal{E}_{j,k}$ contains the events delivered to the thread and $S|_{\mathcal{R}_j}$ restricts the visible state to permitted reads. The harness decides what information is gathered, what is kept, and where each item came from; Context_j assembles one model request from the state and tools permitted to thread j . Because the hosted model provider keeps the conversation history, replaying a request exactly without the provider would also require a complete record of each request. The harness maintains the navigation records, while the environment executes physical motion. The runtime accepts a proposed update only if it lies within $\mathcal{W}_j$ and the current task and delegation permissions, and the component that owns the target state checks its format and target.

B.4 MOTION AUTHORITY AND PROTECTIVE CONTROL

Let $G_o(t) \in \{0, 1\}$ indicate whether physical operation o holds the permission to move, and let $g(t)$ be a permission version number that increases whenever a permission is revoked or replaced. Only a permitted operation of the foreground thread may hold the permission (Eq. (3)), and each task command u_o carries the version number under which it was issued. A command is eligible only if

$$\sum_o G_o(t) \leq 1, \quad \text{eligible}(u_o, t) = G_o(t) \mathbf{1}[\text{version}(u_o) = g(t)] \mathbf{1}[\text{preconditions}(u_o, S(t))], \quad (9)$$

so late model outputs or waypoints queued before a pause cannot regain authority. With a protective-stop flag $p_{\text{stop}}(t)$ that stays set until it is explicitly cleared, and a low-latency filter Protect over fresh sensor observations $O_{\text{phys}}(t)$, the final command filter on the robot outputs

$$u_{\text{robot}}(t) = \begin{cases} u_{\text{stop}}, & p_{\text{stop}}(t) = 1 \text{ or no task command is eligible,} \\ \text{Protect}(O_{\text{phys}}(t), u_{\text{task}}(t)), & \text{otherwise.} \end{cases} \quad (10)$$

An obstacle detection or a control-loop timeout can set p_{stop} and revoke the current permission without waiting for the Planner; recovery requires the stop to be cleared, the state to be refreshed, and a new permission. Only the Planner decides that the goal has been reached; the protective path can block motion regardless of that decision.

B.5 SCHEDULING AND INTERACTION FREQUENCY

Activities may run at different update periods and on different events, as in full-duplex and asynchronous interaction-thinking models (Défossez et al., 2024; Huang et al., 2026a). The runtime design allows the Planner to review a fixed copy of the current state while VLA execution continues, interrupting only when the review asks for revision or termination. One VLA step predicts eight waypoints and executes the first; discarded predictions and failed retries are not counted as steps. The benefit of concurrent review is measured on the Go2 (Section 4.8). With a constant step duration P_{VLA} and a check every F steps, FP_{VLA} is the execution time between checks; total elapsed time also includes any time spent paused for review.

B.6 PRIORITY AND HANDOFF POLICY

The runtime assigns priorities when activities are created, from highest to lowest: protective stops and control-loop timeouts, episode limits, completion of an operation, and route-review requests. Each event carries its operation, permission version, timestamp, and sequence number; events of equal priority are handled in arrival order. Outdated review or completion events are discarded, and repeated review requests for the same operation are merged into one pending request. A protective stop immediately revokes motion authority and increases the version number; reasoning may continue, but neither a late model response nor a queued waypoint can move the robot under the old permission. Clearing an alert alone does not resume motion: fresh observations and a new permission are required. Episode limits also revoke motion; otherwise, an operation that finishes on its own takes precedence over a scheduled review arriving at the same time. A review that lets the route continue keeps the VLA’s history, whereas a correction waits for the current motion step to finish before a new operation is permitted.

This policy describes the full runtime design. The simulator implementation used for our ablations is simpler: it executes one motion operation at a time and implements the version checks, the precedence of operations that finish on their own, a single pending review request, and continuation that keeps the VLA’s history. Preemption by an independent safety monitor and its physical timing are measured on the Go2 (Section 4.8).

B.7 INTERACTION-FREQUENCY EVALUATION

The check schedule is defined in Appendix B.5. All frequencies use the same Planner, VLA, prompt, tools, route-summary format, and resumption procedure, which keeps the VLA’s history; a check may let the current route continue without choosing a recovery action. The execution loop also records a review request after two consecutive blocked moves; such requests are delivered at the

next scheduled check or when the VLA stops on its own, rather than triggering an extra check, so the frequency comparison contains no event-triggered reviews. A request cannot change navigation state or end the episode, and the runtime revokes the VLA’s permission to move before permitting replacement motion. Cost includes all model calls and waiting time; periodic checks are counted separately from returns that occur when the VLA stops on its own. Table 12 summarizes what each component may do at a handoff.

Table 12: What each component may do when execution is interrupted.

Operation	Planner	VLA	Enforced by
Interrupt the running VLA rollout	May request, if permitted	Stops after the current step	Runtime: revokes the permission to move
Request an early Planner review	Receives it, if permitted	Keeps executing	Runtime: queues or discards the request
Resume or revise the route	Chooses an available tool	Executes a valid delegation	Runtime: grants one operation the permission to move
Change physical or route state	Cannot change it directly	Proposes waypoints	Environment: executes motion; harness: updates route records
End the episode	Decides explicitly to stop	Its stop ends only the delegation	Runtime: enforces the stop and episode limits
Share information between Planner and VLA	Sends structured instructions only	Returns structured status and route evidence only	Runtime: controls visibility; harness: builds context

C VISUAL-CONTEXT ALLOCATION DETAILS

Baseline and codec rule. The positional baseline reimplements the Qwen-RobotNav observation interface (Qwen Team, 2026b): time and camera weights allocate a shared budget across retained images, and each image is resized to its assigned pixel budget. The time weight of the t th of T retained observations, counted from the oldest, is $\exp(\gamma t / (T - 1))$, so $\gamma \geq 0$ sets how strongly recent observations are favored. Codec adds the RGB-change multiplier defined in Eq. (4), retaining the same allocator and image-resizing path. The score measures visual change, including changes caused by camera motion and lighting; it is not a learned semantic-importance estimate or an optical-flow estimator.

Integer allocation. Flatten the retained history in time-major, front/right/back/left order. For $M = N_{\text{hist}}N_{\text{view}}$ images, initialize every allocation to $b_{\min}$, requiring $Mb_{\min} \leq B_{\text{vis}}$. Distribute the remaining budget proportionally to the positive joint weights, take the floor of each share, cap additions at $b_{\max}$, and redistribute any overflow among uncapped images. Assign the final integer remainder in descending weight order, filling each image’s remaining capacity before advancing. Ties between equal weights are broken by a fixed sort order. No tokens are assigned beyond an image’s capacity: when $B_{\text{vis}} \geq Mb_{\max}$ all images receive $b_{\max}$ and the excess budget is unused. Image resizing preserves aspect ratio and rounds dimensions to the encoder’s patch/merge grid, so realized image-token counts can be below their allocated limits.

Evaluation configuration. At evaluation, the four-view VLA uses $B_{\text{vis}} = 3072$, $\gamma = 2$, camera weights $(2, 1, 0.5, 1)$ in front/right/back/left order, $b_{\min} = 4$, and $b_{\max} = 196$. It retains at most 16 observations, always including the first and current ones, with the remaining indices sampled uniformly without replacement from intermediate history and sorted chronologically. With these settings, short histories of at most three observations saturate the per-image caps; allocation becomes budget-limited from four observations onward. These evaluation values differ from the randomized training budgets and from the settings of the synthetic check below.

Table 13 is a synthetic check of the allocator, using $b_{\min} = 4$, $\gamma = 2$, the camera weights above (weight one for a single view), and each row’s budget and cap. Change scores are drawn uniformly from $[0.05, 1)$ with a fixed random seed, and the first retained image of every view is given score

Table 13: **Allocation saturation.** Deterministic synthetic change-score diagnostic.

B_{vis}	N_{hist}	N_{view}	b_{max}	ρ_{vis}	Images differing	Mean $ \Delta b $ (tokens)
3072	2	1	196	7.84	0/2	0.0
3072	8	1	196	1.96	0/8	0.0
3072	15	1	196	1.04	0/15	0.0
4096	4	4	256	1.00	0/16	0.0
3072	16	1	196	0.98	3/16	8.0
3072	8	4	196	0.49	30/32	47.3
1024	16	1	196	0.33	14/16	31.4
2048	8	4	196	0.33	31/32	18.9
1024	16	4	196	0.08	57/64	7.9

one. The table reports how many images receive different allocations under the two rules and the mean per-image $|\Delta b|$, where $\Delta b = b^{\text{codec}} - b^{\text{pos}}$. All integer bounds and allocation totals hold, and the four saturated rows, where the two rules must agree, give identical allocations. We will release the score arrays, allocations, and allocator code. These are synthetic calculations, not navigation results; navigation performance is evaluated in Figure 7.

D ADDITIONAL EXPERIMENTAL DETAILS

D.1 EVALUATION SETTINGS

Table 14: Harness evaluation settings.

Setting	Value
RGB resolution	512 pixels
Camera order	Front, right, back, left
Returned route keyframes	At most 8, uniformly sampled along the rollout
Episode / delegation limit	500 / 200 environment steps
Planner-call limit	200 turns per episode
Episode timeout	2,400 s

Benchmark goal distances and success labels are available only to the offline evaluator.

D.2 HUMAN PERFORMANCE ON RxR-CE

RxR asks an agent to follow a route instruction to its endpoint. Human demonstrations use discrete graph edges, which abstract away low-level movement and obstacle avoidance; RxR-CE realizes navigation through continuous motion (Ku et al., 2020; Krantz et al., 2020). Our 90.43 SR and 78.47 nDTW match English human followers (90.4 SR and 77.7 nDTW) despite this additional control burden on the same instructions.

D.3 CROSS-TASK EVALUATION

The object-goal subsets are drawn in proportion to target categories and scenes. Reference values follow Table 4 of Qwen Team (2026b). We do not report per-category MP3D results, since proportional sampling leaves the rarest categories with about two episodes each. Question-answering metrics follow Table 7 of Qwen Team (2026b); prior EXPRESS-Bench scores used the now-deprecated GPT-4o-mini judge.

D.4 PLANNER SCAFFOLDS AND ACCOUNTING

We replace each scaffold’s default task prompt with the NAVGPT-3 navigation prompt, which describes the robot, the instruction, the tools, the movement budget, and the stopping rule, and we

provide the corresponding navigation tools. The scaffolds’ own task prompts are not used, and every reported Planner result is produced by the NAVGPT-3 harness running inside them. The scaffold remains responsible for assembling the conversation and managing its context window, including any automatic compression by the provider. The harness determines which observations, route evidence, and spatial state enter those messages, subject to the runtime’s visibility permissions. Replacing the task prompt therefore does not make Codex and Claude Code identical: their system instructions, built-in tools, and formatting of tool results may still differ. Ablations within one Planner keep the scaffold and prompt fixed; comparisons across Planners describe each complete Planner–scaffold combination rather than ranking the underlying models in isolation.

Tokens (k) sum all processed input tokens, including cached input, and generated output tokens across requests, divided by 1,000. Turns count Planner responses separated by environment feedback, including the initial response and final acknowledgement; text, reasoning, and parallel tool calls within a response count once. Cost is the total hosted inference charge in US dollars per attempted episode, including retries and cache charges. VLA latency is reported separately rather than priced. Opus costs are as reported by the provider’s SDK; Astra costs use recorded token usage at list prices. Inference timing excludes communication and harness overhead. The action-update interval in Table 3 is Planner time per turn without the VLA and VLA inference per step with it; it does not include the time for an interrupt to reach the robot or for the robot to brake.

D.5 VISUAL PROMPT VARIANTS

Every variant keeps the same eight tools, camera frames, route keyframes, pose readings, node IDs, annotations, observed geometry, and tool behavior. The grounded reference draws labels onto their images in a 5×7-pixel font and shows the map with the map frame’s north at the top, a heading arrow, and turn cues relative to the agent’s heading; north here refers to the map frame, not to geographic north. Compact glyphs use a 3×5-pixel digit font at the same height and contrast, the heading-up map keeps the same crop and scale, and absolute bearings provide bearing and pose information sufficient to derive the same directions. Labels stay outside the scene content so that no variant changes occlusion.

E EVALUATION COVERAGE AND VALIDATION

E.1 R2R ABLATION SUBSET

The 100-episode ablation set follows the public OpenNav subset (Qiao et al., 2025). All 100 episodes in our runs match the published subset in episode identifier, instruction, and scene, and each matches the R2R-CE v1.3 Val-Unseen release in instruction text, scene, start position, reference path, and geodesic distance. Table 15 compares these episode attributes with the full 1,839-episode split. Numeric entries give mean $\pm$ standard deviation. The absolute standardized mean difference (SMD) compares the subset with its disjoint 1,739-episode complement, using the square root of their equally weighted variances; D is the maximum difference between the subset and full empirical cumulative distributions. Directional mentions count `left`, `right`, and forms of `turn`; reference-path turns are horizontal direction changes of at least 30°, rather than executed primitive turns. These are interpretable task-complexity proxies, not a complete semantic-complexity measure.

Table 15: R2R ablation-subset coverage and distributions.

Statistic	Full	Subset	SMD	D
Episodes	1,839	100	–	–
Scenes	11	10	–	–
Unique reference trajectories	613	97	–	–
Geodesic distance (m)	8.90 $\pm$ 2.67	8.99 $\pm$ 3.06	0.033	0.072
Instruction length (words)	26.65 $\pm$ 11.48	24.68 $\pm$ 8.28	0.207	0.103
Directional mentions	2.14 $\pm$ 1.80	2.20 $\pm$ 1.75	0.034	0.040
Reference-path turns	1.56 $\pm$ 1.08	1.46 $\pm$ 1.21	0.095	0.100

Path distance and directional mentions have small mean differences; instructions in the subset are shorter and have a narrower length distribution. The subset covers 10 of 11 scenes, omitting a scene containing 18 full-split episodes, and its scene-frequency total-variation distance from the full split is 0.097. In particular, one scene accounts for 3% of the subset versus 9.63% of the full split. The Pearson χ^2 statistic of the subset’s scene frequencies is 8.14; among 10,000 random 100-episode subsets drawn without replacement, 56.7% have a statistic at least this large. The observed imbalance is therefore typical of random subsets of this size, but this does not establish that the subset is representative, especially since it was curated. The public OpenNav and v1.3 files also differ in the initial headings of all 100 selected episodes, and we fix them in our evaluation. Thus this comparison establishes coverage and similarity of the listed attributes, not full execution-protocol or performance equivalence. Full-split results remain the basis for benchmark comparisons.

E.2 MATCHED OBJECTNAV EVALUATION

The validation in Table 16 runs Qwen-RobotNav 8B and NAVGPT-3 on exactly the same 500 episodes for each dataset, with identical scene versions, sensing, motion budgets, collision handling, and stop/success rules. It is a system comparison; the complete system additionally uses its hosted Planner. Selection is fixed before observing either system’s outcomes. SR and SPL are percentages. Δ SR vs. full is the 500-episode SR minus the full-split SR published for Qwen-RobotNav 8B, in percentage points (−1.0 on MP3D, +1.2 on HM3D v2); it indicates how closely the selected episodes track the full split. NAVGPT-3 has no full-split run, so its entry is a dash.

Table 16: ObjectNav comparison on identical episodes.

Dataset	System	N	SR	SPL	Δ SR vs. full
MP3D	Qwen-RobotNav 8B	500	47.8	17.5	−1.0
MP3D	NAVGPT-3	500	59.2	25.2	—
HM3D v2	Qwen-RobotNav 8B	500	72.4	33.1	+1.2
HM3D v2	NAVGPT-3	500	75.3	40.8	—

E.3 EXPRESS JUDGE VALIDATION

We compare the GPT-6 Astra judge with the GPT-4o-mini judge used by prior EXPRESS-Bench results on the same saved answers and final observations, scored against human judgments. Table 17 reports agreement on 50 human-labeled answers: Astra agrees with the human label on 46 and GPT-4o-mini on 36, a 20.0-point difference. Without item-level pairing, the exact McNemar p depends on how the two judges’ errors overlap; for these totals it lies between 0.002 and 0.031. This issue applies to open-answer EXPRESS scoring; HM-EQA and MT-HM3D multiple-choice accuracy uses the reference option.

Table 17: EXPRESS judge agreement with human labels.

Judge	Labeled N	Agreement (%)
GPT-4o-mini	50	72.0
GPT-6 Astra	50	92.0
Astra — 4o-mini		+20.0