MOON: LEARNING WHEN TO FORGET AND HOW TO MOVE ON IN VISION-AND-LANGUAGE NAVIGATION

Anonymous authors
Paper under double-blind review

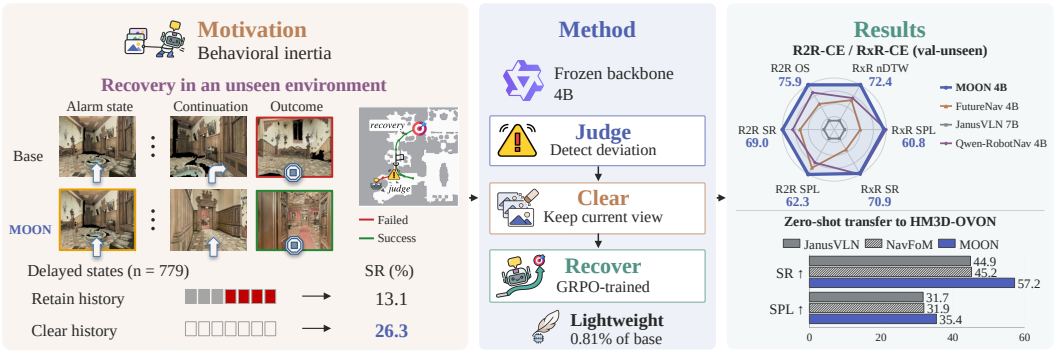


Figure 1: **MOON learns when to forget and how to move on.** Left: one paired episode in an unseen environment, in which the base fails and **MOON** recovers after the judge's alarm (amber border), and the effect of clearing, which raises pooled success from 13.1% to 26.3% across 779 delayed states of four frozen policies. Middle: a deviation judge triggers history clearing and a trained recovery policy on a frozen 4B backbone. Right: R2R-CE and RxR-CE val-unseen results against recent leading monocular RGB models of Table 1, 4B versions where a model comes in several sizes, and zero-shot HM3D-OVON results against JanusVLN and NavFoM. Each radar axis is scaled to the range of the plotted models; labeled values are **MOON**'s.

ABSTRACT

Recent vision-and-language navigation (VLN) policies act on their visual history, and an incorrect action can take them off the instructed route. Paired interventions on four VLN policies show that clearing the history doubles pooled success at the onset of a deviation but lowers it on route, while later in a deviation, removing only the off-route frames helps much less. After a deviation, the retained history keeps the policy from turning where a turn is needed, which we call *behavioral inertia*. We propose **MOON**, a framework that equips a frozen VLN policy with two lightweight modules, adding less than 1% of its parameters. A deviation judge decides when to forget, raising an alarm that clears the history when it predicts that the agent is off route. A recovery policy, trained with GRPO from alarm states, learns how to move on, turning more of the successes found only by sampling into a single greedy rollout. Experiments on R2R-CE and RxR-CE show that **MOON** reaches the highest R2R-CE success rate among the compared methods and raises the success rate of its frozen base by 5.2 and 6.0 points, and an ablation with a random trigger indicates that this gain depends on when the history is cleared. Without retraining, **MOON** also improves zero-shot object-goal navigation on HM3D-OVON and runs on a real robot. Together, these results suggest that deviation recovery depends on knowing when to forget, not only how to act.

1 INTRODUCTION

Vision-and-Language Navigation (VLN) requires an embodied agent to follow natural-language instructions and reach a target location in a 3D environment using egocentric visual observations

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

(Anderson et al., 2018b; Krantz et al., 2020). Recent VLN policies leverage pretrained vision-language models to predict navigation actions from the instruction, the current observation, and the visual history (Zhu et al., 2026; Cheng et al., 2025; Wei et al., 2025; Zhang et al., 2025). Yet long-horizon navigation remains vulnerable to action errors. An incorrect action can take the agent off the instructed route, causing a *deviation*. Unless corrected promptly, the agent must continue from an off-route state, where later errors can compound into task failure (Ross et al., 2011). Beyond avoiding deviations, reliable VLN thus requires correcting course and resuming the instructed route after a deviation, which we refer to as *deviation recovery*.

Existing approaches to deviation recovery primarily improve the policy itself. They expose the policy to off-route states and teach it to return to the instructed route, either by continuing training on corrective trajectories from failed episodes (Yu et al., 2026; Xin et al., 2026) or by reinforcement learning (Li et al., 2026b). These approaches improve overall success, but because they change the policy and evaluate it over complete episodes, which mix on-route and off-route states, they do not isolate the effect of retaining the visual history during recovery. Appendix A reviews related work, including detectors that hand control to a separate controller (Jeong et al., 2026). Work on human memory suggests that retaining everything is not always useful: forgetting reduces the influence of outdated information on decisions in changing environments (Richards & Frankland, 2017), and people segment their experience into a new event when their predictions fail (Zacks et al., 2007). In VLN, visual history may help before a deviation, when it records progress along the instructed route. Once the agent deviates, however, newly accumulated observations begin to describe an off-route trajectory that no longer corresponds to the instruction (Ke et al., 2019). The policy must therefore learn to recover while still conditioning on a history increasingly shaped by its own mistakes. This history could provide useful context for retracing the route or reinforce the deviation. We ask: ***How does visual history affect recovery after a deviation?***

To answer this question, we run paired interventions in the Habitat simulator (Savva et al., 2019) on R2R-CE with four VLN policies (NaVIDA, NaVILA, StreamVLN and Uni-NaVid) that realize three kinds of visual memory, without modifying their parameters. Following prior work (Yu et al., 2026; Xin et al., 2026), a deviation occurs when the agent's distance from the reference route exceeds a threshold; from each deviation onset, we run the policy twice, once with its accumulated visual history and once with the history cleared, leaving only the instruction and the current frame. Clearing the history raises the success rate from 13.7% to 28.6% across 915 onset states from the four policies and lowers it from 76.0% to 48.3% on route; both effects hold for every policy. A natural explanation for the harm is that the accumulated off-route frames mislead the policy. We test this explanation at states ten or more steps after the deviation by retaining only the history up to the onset, which removes the later off-route frames while keeping the agent's on-route context. On these states, doing so raises the success rate only from 13.1% to 16.3%, well below the 26.3% that clearing reaches (Figure 1, left). The harm therefore does not come mainly from the off-route frames.

After a deviation, the retained history keeps the policy from turning where the new state calls for a turn. We call this *behavioral inertia*. At the deviation onset, after a forward step, a shortest-path expert at the same pose continues without a turn in only 2% of states. The four policies continue without a turn through their next decision in 48% of states, and in 11% once their history is cleared. Further interventions on the visual content of a full-length retained history, shuffling it or swapping in one from another scene, lower the success rate on route but barely change it after a deviation. These findings are consistent with a known failure mode of policies trained on expert demonstrations. During training, features associated with expert histories reliably predict expert actions, leading the policy to depend on them. After the policy leaves the expert distribution, these associations may no longer hold, but the policy continues acting on them and compounds its errors (de Haan et al., 2019; Codevilla et al., 2019; Wen et al., 2020; Sobol Mark et al., 2026). We hypothesize that, in our setting, the history itself becomes such a feature through its presence and length more than its content: histories filled with current-frame copies or noise still cause most of the harm, and the harm grows with the retained length (Sections 2.2 and 2.3). Clearing the history reduces this inertia and doubles the recovery success rate. It does not reveal how to return to the route, however, and most deviations still end in failure. Yet among onset states where the retained greedy rollout fails, at least one of eight sampled rollouts after clearing succeeds in 61.6%, against 30.3% with the history retained (Section 2.4), so there exist many recoveries that a single greedy rollout misses.

Building on these insights, we propose **MOON** (Figure 1, middle), whose two components answer, respectively, when to clear the visual history and how to make recovery succeed after clearing.

Both are added to a frozen policy and together amount to less than 1% of its parameters. The first component, a *deviation judge*, clears the visual history when it predicts that the agent is off route, because clearing on route is harmful. The second component, a *recovery policy*, takes over after clearing. Because reinforcement learning with verifiable rewards is effective at making behaviors that a policy can already sample more reliable (Yue et al., 2025), we train it with Group Relative Policy Optimization (GRPO) (Shao et al., 2024) on cleared rollouts from alarm states.

We evaluate **MOON** on R2R-CE and RxR-CE (Figure 1, right). On R2R-CE it reaches the highest success rate among the compared methods, 2.1 points above the next best, Qwen-RobotNav-4B (Zhang et al., 2026b), and it raises the success rate of its frozen base by 5.2 points on R2R-CE and 6.0 on RxR-CE. A random trigger with as many alarms lowers success below the base, so the gain depends on when the history is cleared. Without retraining, the same components improve zero-shot open-vocabulary object-goal navigation on HM3D-OVON (Yokoyama et al., 2024b) and run on a real robot. Our contributions are threefold:

- **Analysis.** On four frozen VLN policies, paired interventions show that the visual history is beneficial on route and harmful after a deviation, that removing the off-route frames recovers only a small part of this harm, that the retained history keeps the policy from turning where a turn is needed (behavioral inertia), and that after clearing, sampling finds successes that a single greedy rollout misses (Section 2).
- **Method.** We propose **MOON**, two lightweight modules on a frozen policy: a deviation judge that learns when to clear the history and a recovery policy that learns how to recover (Section 3).
- **Experiments. MOON** reaches the highest R2R-CE success rate among the compared methods and competitive results on RxR-CE; its gain depends on when the history is cleared, and without retraining it improves zero-shot object-goal navigation and runs on a real robot (Section 4).

2 Motivation and Analysis

2.1 Preliminaries

Problem Formulation. We study vision-and-language navigation in continuous 3D environments. Given a natural-language instruction I, at each step t the agent receives an egocentric RGB frame o_t and executes an action a_t: move forward 0.25 m, turn 15° left or right, or stop. The policy reads its interaction history $H_t = \{(o_s, a_s)\}_{s=0}^{t-1}$ through a memory constructor M:

$$a_t \sim \pi_\theta\big(\cdot \mid I,\, o_t,\, M(H_t)\big). \tag{1}$$

In practice the policies act in chunks: at a *decision point* t, π_θ proposes a chunk c_t of atomic actions and executes it before the next decision point, and t always counts atomic steps. An episode succeeds if the agent stops within the standard success radius of the goal.

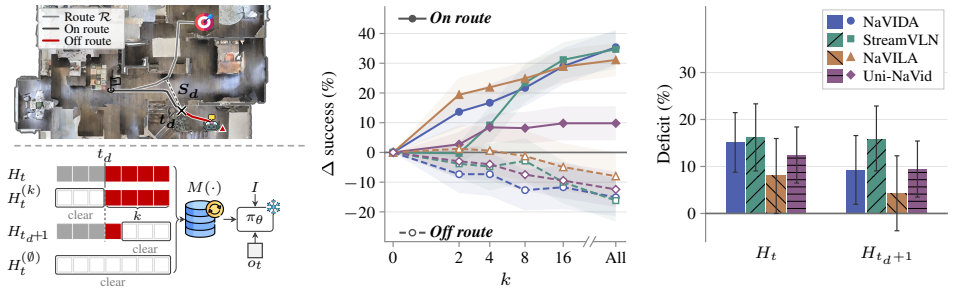
Deviation Criterion. The simulator provides the agent's position q_t and the reference route $\mathcal{R}$, which the instruction describes and the agent never observes. Following prior work (Yu et al., 2026; Xin et al., 2026), the agent deviates once it is more than $S_d = 2$ m from the route:

$$h_t = \min_{r \in \mathcal{R}} d_{\text{geo}}(q_t, r), \qquad t_d = \min\{\, t : h_t > S_d \,\}, \tag{2}$$

where d_{geo} is the geodesic distance and the deviation onset t_d is ∞ for an episode that never deviates.

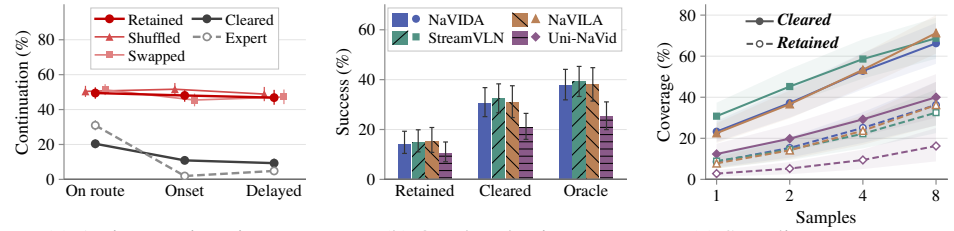
2.2 Two Opposing Roles of Visual History

We first ask whether, from the same simulator state, retaining the visual history helps or hurts the policy. All analyses in this section use the same 600 randomly sampled R2R-CE val-unseen episodes and four frozen policies that realize M in three ways (Appendix B). The *full-length retained history* denotes each policy's memory constructed from the complete interaction prefix H_t. *Clearing* replaces H_t with $\emptyset$ while preserving the instruction I and current frame o_t. Each intervention replaces only the history passed to M at a decision point and restarts the policy from the same simulator state with the same remaining step budget (Figure 2a).



(a) History interventions (b) History length (c) History content

Figure 2: **Visual history helps on route but can hinder recovery.** (a) Boxed steps are cleared. (b, c) Off route: delayed states. Bands and whiskers: 95% paired bootstrap intervals.



(a) Action continuation (b) Oracle selection (c) Sampling coverage

Figure 3: **Retained history keeps the policy from turning; after clearing, sampling finds more successes.** (a) After a forward step, four policies pooled. (b) Onset states; oracle: the successful rollout. (c) Onset states whose retained greedy rollout failed. 95% CIs: bootstrap (a, c), Wilson (b).

Clearing On and Off Route. An onset (delayed) state is the first decision point at or after t_d ($t_d + 10$). Pooled rates weight every state of the four policies equally. Across 915 paired onset states, clearing raises pooled success from 13.7% to 28.6% (95% CI on the difference $[11.8, 18.1]$) and raises success for every policy. On 1,465 on-route states nearest the midpoint of never-deviating episodes, clearing lowers pooled success from 76.0% to 48.3% and lowers success for every policy.

History Length. We vary history length by keeping only the k most recent steps,

$$H_t^{(k)} = \{(o_s, a_s)\}_{s=t-k}^{t-1}, \quad k \in \{0, \ldots, t\}, \tag{3}$$

so that $k = 0$ is clearing, written $H_t^{(\emptyset)}$, and $k = t$ is the full-length retained history. Figure 2b plots each length's success rate minus that with clearing (Δ success) for the on-route states and the 779 delayed states, each within its own population. On route, pooled success rises with k, from 9.3 points above clearing at $k = 2$ to 27.7 points with the full-length retained history. After a deviation, every retained length has lower pooled success than clearing, and this gap, which we call the *deficit*, widens from 3.5 points at $k = 2$ to 13.2 points with the full-length retained history.

History Content. If the off-route frames accumulated after a deviation mislead the policy, removing them should restore recovery. At the same 779 delayed states, we therefore retain only the history up to and including the deviation onset, H_{t_d+1} (Figure 2c). This raises pooled success from 13.1% to 16.3%, leaving a deficit of 10.0 points (95% CI $[6.5, 13.5]$). The deficit persists when the history ends before its first frame beyond 0.5 m from the route, and an on-route test suggests that the discontinuity between history and current frame does not explain it (Appendix B). Removing off-route frames recovers only a fraction of the loss: the harm lies mainly outside what they depict.

2.3 Behavioral Inertia

After a deviation, shuffling the full-length retained history or swapping in one from another scene leaves 13.0 and 12.5 of the 13.2-point deficit, whereas on route both lower success for every policy; length-matched copies of the current frame or noise leave 8.7 and 7.7 points (Appendix B).

Action Continuation. We take states right after a forward step and record whether the policy's next action chunk continues without a turn, counting a stop as no turn. We compare this continuation rate with that of a shortest-path expert queried at the same pose over the same number of steps (Figure 3a). On route, the expert continues in 31.0% of states, and the policy in 49.5% with retained history and 20.4% after clearing. After a deviation, the expert almost always turns, continuing in only 1.9% of onset and 4.8% of delayed states, whereas the policy with retained history still continues in 48.1% and 46.8%. After clearing, the continuation rate falls to 10.9% and 9.3%, and each policy shows the same pattern. Shuffled and swapped histories keep the pooled rate close to that of the full-length retained history; per policy they shift it by up to 25 points in either direction, but for every policy it stays above clearing and the expert (Appendix B). After a deviation the state calls for a turn, yet the retained history keeps the policy from turning; we call this behavioral inertia.

2.4 Beyond Clearing

Choosing correctly between retaining and clearing would not rescue most deviations: clearing rescues 195 of the 790 onset states where the retained rollout fails but breaks 58 of the 125 where it succeeds, and even a hindsight oracle leaves 65.0% of states unresolved (Figure 3b).

Sampling Coverage. On a separate subset of 320 onset states where the retained greedy rollout failed, 80 per policy, we draw eight rollouts per condition (Figure 3c; Appendix B). Coverage, written pass@k for k rollouts, is the share of states with at least one successful rollout. With eight samples, pooled coverage rises from 30.3% with retained history to 61.6% after clearing, yet 38.4% of states still have no successful cleared rollout. This offline coverage assumes, unlike deployment, that a success can be recognized among the candidates. Deciding when to clear, and turning successes that appear only under sampling into a single greedy rollout, are the two problems that the deviation judge and the recovery policy of Section 3 address.

3 Method

MOON answers the two questions left by Section 2: when to clear the history (§3.1), since clearing helps only off route (Section 2.2), and how to recover after clearing (§3.2), since sampling finds successes a greedy rollout misses (Section 2.4). The deviation judge f_ϕ and the recovery policy $\pi_{\theta,\psi}$ add LoRA adapters ϕ and ψ to the backbone of the frozen policy π_θ (Hu et al., 2022). At the judge's alarm, clearing empties the visual history and the recovery policy takes over (Figure 4).

3.1 Deviation Judge

Prediction Target and Input. The route $\mathcal{R}$ is unavailable at deployment, so the judge predicts deviation from the instruction and the agent's own observations and actions. At each decision point t, π_θ proposes the chunk c_t of Section 2.1. Brief excursions that return to the route occur mostly in episodes that succeed, so a LOST label requires the agent to be off route now and at half or more of a set of later steps V_t within a fixed horizon. On rollouts of π_θ in the training splits, each decision point receives a label from h_t and S_d of equation 2:

$$y_t = \begin{cases} \text{LOST}, & \text{if } h_t > S_d \text{ and } \big|\{\, u \in V_t : h_u > S_d \,\}\big| \geq \tfrac{1}{2}\,|V_t|, \\ \text{OK}, & \text{otherwise,} \end{cases} \tag{4}$$

where V_t is empty when too few steps remain (Appendix C). The judge reads $E_t = (I, o_t, H_t, c_t)$, taking sampled frames and recent actions from H_t. We include the proposed chunk because it shows where the agent is about to go. It outputs $p_\phi(y \mid E_t)$ for $y \in \{\text{OK}, \text{LOST}\}$ and is trained by cross-entropy on these labels. Stop actions carry no label, and chunks that contain a stop are not judged.

Score and Threshold. The deviation score $\bar{m}_t$ averages the judge's log-odds margin m_t over recent decision points to suppress single-point noise:

$$m_t = \log \frac{p_\phi(\text{LOST} \mid E_t)}{p_\phi(\text{OK} \mid E_t)}, \qquad \bar{m}_t = \frac{1}{|\mathcal{W}_t|} \sum_{s \in \mathcal{W}_t} m_s, \tag{5}$$

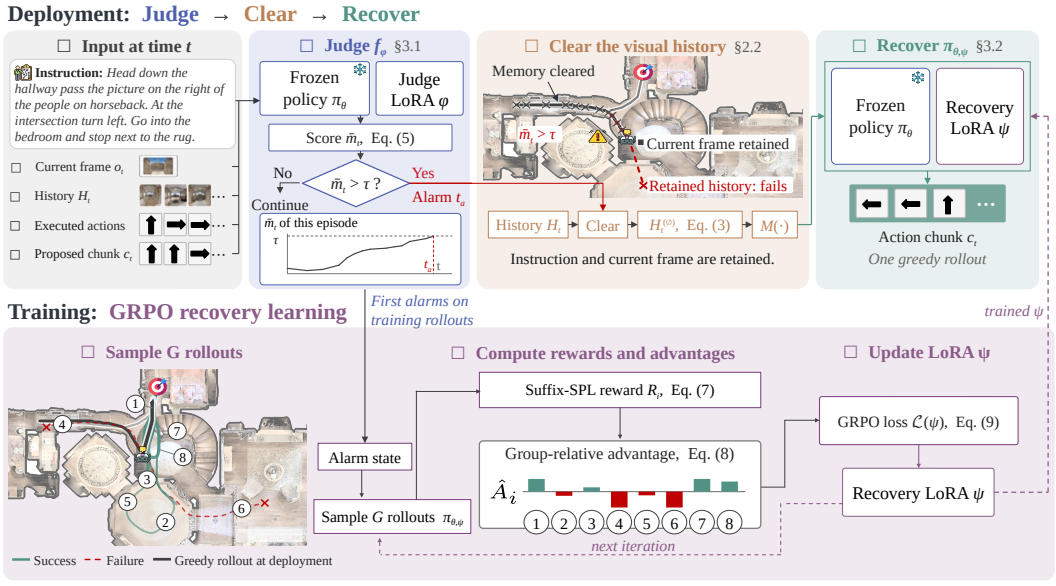


Figure 4: **Overview of MOON.** Top: at deployment, the deviation judge f_ϕ raises an alarm when the deviation score $\bar{m}_t$ exceeds the conformal threshold τ, clearing empties the visual history, and the recovery policy $\pi_{\theta,\psi}$ acts in one greedy rollout. Bottom: in training, G rollouts are sampled from each alarm state and scored by the suffix-SPL reward, and their group-relative advantages update only the LoRA ψ. Scenes and the score inset show one episode of the trained system, while the sampled rollouts are schematic and the advantage bars come from a group resampled after training.

where $\mathcal{W}_t$ holds the last W judged decision points up to t. Clearing on route lowers success (Figure 2b), so we use split conformal prediction (Vovk et al., 2005; Angelopoulos & Bates, 2021) to bound how often an episode that π_θ would complete raises an alarm. Each of n held-out calibration episodes that π_θ completes is scored by $Z_e = \max_t \bar{m}_t$, the closest it comes to an alarm. With the scores sorted as $Z_{(1)} \le \cdots \le Z_{(n)}$ and a level α, the threshold and the alarm step are

$$\tau = Z_{(\lceil (n+1)(1-\alpha) \rceil)}, \qquad t_a = \min\{\, t : \bar{m}_t > \tau \,\}, \tag{6}$$

where t ranges over judged decision points and $t_a = \infty$ if the score never exceeds τ. Until t_a the agent is exactly π_θ. A new episode that π_θ completes is therefore interrupted with probability at most α if it is exchangeable with the calibration episodes. Unlike t_d, the alarm step uses only what is available to the agent and can precede t_d.

Clearing and Handover. An alarm triggers clearing, which reduces the behavioral inertia of Section 2.3 (Figure 3a). At t_a, clearing replaces the retained history H_{t_a} with $H_{t_a}^{(\emptyset)}$ of equation 3 while keeping the instruction and the current frame. The recovery policy $\pi_{\theta,\psi}$ proposes the next chunk and acts greedily until the episode ends. Its history re-accumulates from t_a, so for $t > t_a$ it reads $H_t^{(t-t_a)}$. Clearing happens at most once per episode, as in training (§3.2). Algorithm 1 (Appendix C) summarizes the loop, and Section 4 compares this timing with other triggers.

3.2 Recovery Policy

Learning from Cleared States. After clearing, repeated sampling finds a success at far more states than a single rollout does (Figure 3c). This coverage assumes that a success can be recognized among candidates, which deployment cannot do, so the recovery policy must turn these successes into a single greedy rollout. Reinforcement learning with verifiable rewards can make sampled behaviors more reliable (Yue et al., 2025), so we train $\pi_{\theta,\psi}$ with GRPO (Shao et al., 2024). Training starts from the first alarm on rollouts of π_θ in the training splits, successful ones included, with the threshold given in Appendix C. From each alarm state we sample G rollouts $\zeta_1, \ldots, \zeta_G$ with $\pi_{\theta,\psi}$, each beginning with the same clearing as at deployment and running without the judge.

Suffix-SPL Reward. Only the suffix after the alarm can be credited to the recovery policy, so the reward applies SPL (Anderson et al., 2018a) to the suffix alone. With d_0 the geodesic distance from the alarm state to the goal, rollout ζ_i earns the suffix-SPL reward

$$R_i = \mathbf{1}[\mathrm{succ}_i] \left(\lambda + (1-\lambda)\,\frac{d_0}{\max(d_0,\, L_i)}\right), \tag{7}$$

where succ_i indicates that ζ_i stops within the success radius and L_i is its path length. Shorter successes earn more when $L_i > d_0$, and the floor $\lambda \in (0,1)$ keeps every success above every failure. GRPO normalizes rewards within the group, which gives a baseline without a learned critic:

$$\hat{A}_i = \frac{R_i - \mathrm{mean}(R_1,\ldots,R_G)}{\mathrm{std}(R_1,\ldots,R_G) + \epsilon}, \qquad \epsilon = 10^{-6}. \tag{8}$$

We update only the recovery adapter ψ by minimizing the loss $\mathcal{L}(\psi)$:

$$\mathcal{L}(\psi) = \frac{1}{N}\sum_{i,j} w_{i,j}\Big[\ell\big(c_{i,j}, \hat{A}_i\big) + \beta\,\hat{D}_{i,j}\Big], \tag{9}$$

where $c_{i,j}$ is the j-th chunk of ζ_i, ℓ is the clipped surrogate loss that GRPO inherits from PPO (Shao et al., 2024; Schulman et al., 2017), and $\hat{D}_{i,j}$ estimates $D_{\mathrm{KL}}(\pi_{\theta,\psi} \,\|\, \pi_\theta)$, with weights and normalization as in the appendix. Groups of equal rewards have zero advantage and are discarded. In practice almost all of these groups contain only failures, and clearing leaves fewer of them than retaining the history. Section 4 measures how clearing affects training.

4 Experiments

4.1 Experimental Setup

Implementation Details. We evaluate on the full val-unseen splits of R2R-CE (1,839 episodes) and RxR-CE (3,669 episodes) with NE, OS, SR, SPL and, on RxR-CE, nDTW. The frozen policy, NaVIDA-Qwen3VL, follows the official training recipe of NaVIDA (Zhu et al., 2026) and only replaces its backbone with Qwen3-VL-4B-Instruct. It is not among the four policies of Section 2, so Section 4.3 repeats the off-route clearing test on it. The judge and the recovery policy are trained on the training splits, the latter with GRPO for 20 iterations from alarm states. The threshold is calibrated per dataset at $\alpha = 10\%$ for R2R-CE and 7.5% for RxR-CE, the levels with the highest closed-loop SR of those evaluated on val-unseen. All policies decode greedily, and paired comparisons use the exact McNemar test on episode outcomes; Appendices C and D.3 give the details.

4.2 Main Results

Comparison with Prior Methods. On R2R-CE (Table 1), **MOON** is best in every column, with an SR of 69.0 against 66.9 for the next method, Qwen-RobotNav-4B, and an SPL tying CorrectNav at 62.3. On RxR-CE it has the lowest NE and second-highest nDTW, with SR and SPL within 2.7 points of the best, Qwen-RobotNav-8B. Over the frozen base, the adapters raise SR from 63.8 to 69.0 on R2R-CE and from 64.9 to 70.9 on RxR-CE, and SPL from 59.0 to 62.3 and 55.9 to 60.8.

Judge Alarms. Figure 5 shows where the judge alarms and what the alarm changes. The judge alarms in 56.3% of the episodes the base fails on R2R-CE and 45.8% on RxR-CE, but in only 13.0% and 8.0% of those it completes. Over all episodes, the full system rescues 24.6% of the base failures and breaks 5.8% of the base successes on R2R-CE (164 against 68 episodes, exact McNemar test, $p = 2\times10^{-10}$), and 24.7% against 4.1% on RxR-CE (317 against 97, $p = 2\times10^{-28}$). Appendix D.2 gives the counts, the recovery policy's effect on alarmed episodes and the alarm timing.

Zero-Shot Object Navigation. Zero-shot on HM3D-OVON val-unseen, with both adapters and the threshold unchanged, **MOON** raises the base's SR from 40.7 to 57.2 and SPL from 21.7 to 35.4, the best among the compared methods, trained on HM3D-OVON or not (Table 15, Appendix D).

Qualitative Results. Figure 7 shows an RxR-CE episode with an alarm; Appendix F shows more. Without retraining, **MOON** also runs on a wheeled robot (Appendix E).

Table 1: Comparison with prior methods on the R2R-CE and RxR-CE val-unseen splits. **Bold** and underline mark the best and second-best results in each column. Pano., Odo., Depth and S.RGB denote panoramic view, odometry, depth and single RGB input. †Recovery-oriented method.

Method	Params	Observation				R2R-CE Val-Unseen				RxR-CE Val-Unseen			
		Pano.	Odo.	Depth	S.RGB	NE↓	OS↑	SR↑	SPL↑	NE↓	SR↑	SPL↑	nDTW↑
Panoramic or depth input													
HPN+DN [ICCV21] (Krantz et al., 2021)	–	✓	✓	✓		6.31	40.0	36.0	34.0	–	–	–	–
CMA [CVPR22] (Hong et al., 2022)	–	✓	✓	✓		6.20	52.0	41.0	36.0	8.76	26.6	22.2	47.1
ETPNav [TPAMI24] (An et al., 2024)	–	✓	✓	✓		4.71	65.0	57.0	49.0	5.64	54.8	44.9	61.9
NavMorph [ICCV25] (Yao et al., 2025)	–			✓	✓	5.75	56.9	47.9	33.2	8.85	30.8	22.8	44.2
SC^2-WM† [ICML26] (Yao et al., 2026)	–	✓	✓	✓		4.25	70.0	66.0	54.0	5.72	60.0	50.0	67.0
Monocular RGB, recovery-oriented													
CorrectNav [AAAI26] (Yu et al., 2026)	7B				✓	4.24	67.5	65.1	**62.3**	4.09	69.3	63.3	**75.2**
BudVLN [arXiv26] (He et al., 2026)	7B				✓	4.74	65.6	57.6	51.1	5.79	56.1	46.6	63.2
SACA [arXiv26] (Li et al., 2026a)	8B				✓	4.19	69.3	64.7	56.9	4.75	62.1	51.7	66.0
DecoVLN [CVPR26] (Xin et al., 2026)	7B				✓	5.01	63.5	56.3	50.5	5.73	54.2	46.3	63.5
ReflectVLN [arXiv26] (Wang et al., 2026)	6B				✓	4.19	67.3	62.8	58.5	3.98	66.0	57.2	–
Monocular RGB, general-purpose													
NaVid [RSS24] (Zhang et al., 2024)	7B				✓	5.47	49.1	37.4	35.9	–	–	–	–
Uni-NaVid [RSS25] (Zhang et al., 2025)	7B				✓	5.58	53.3	47.0	42.7	6.24	48.7	40.9	–
NaVILA [RSS25] (Cheng et al., 2025)	8B				✓	5.22	62.5	54.0	49.0	6.77	49.3	44.0	58.8
StreamVLN [ICRA26] (Wei et al., 2025)	7B				✓	4.90	63.6	56.4	50.2	5.65	54.4	45.4	63.7
NavFoM [ICLR26] (Zhang et al., 2026a)	7B				✓	5.01	64.9	56.2	51.2	5.51	57.4	49.4	60.2
JanusVLN [ICLR26] (Zeng et al., 2026)	7B				✓	4.78	65.2	60.5	56.8	6.06	56.2	47.5	62.1
DualVLN [ICLR26] (Wei et al., 2026)	7B				✓	4.05	70.7	64.3	58.5	4.58	61.4	51.8	70.0
FutureNav-4B [arXiv26] (Zhang et al., 2026c)	4B				✓	4.24	70.2	65.4	61.3	4.60	60.9	52.8	68.0
FutureNav-8B [arXiv26] (Zhang et al., 2026c)	8B				✓	4.29	69.2	64.3	60.3	4.26	63.9	54.8	69.1
Qwen-RobotNav-4B [arXiv26] (Zhang et al., 2026b)	4B				✓	4.22	73.6	66.9	60.5	4.15	71.3	61.5	68.6
Qwen-RobotNav-8B [arXiv26] (Zhang et al., 2026b)	8B				✓	4.36	72.7	65.7	59.6	4.16	**73.4**	**63.5**	69.9
MOON (Ours)	4B				✓	**3.56**	**75.9**	**69.0**	**62.3**	**3.75**	70.9	60.8	72.4

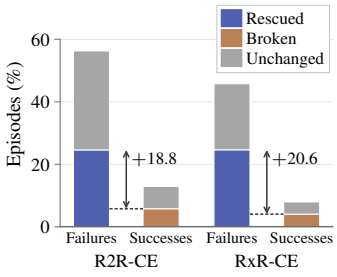


Figure 5: **Judge alarms and their outcomes.** Bars: share of base failures or successes with an alarm; colored: share rescued or broken; arrows: their difference, not an SR change.

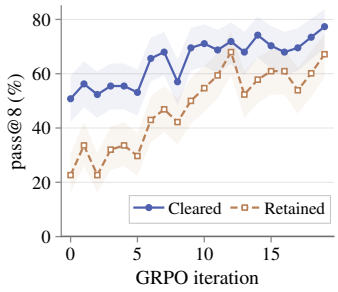


Figure 6: **Training with cleared and retained history.** pass@8 on the 128 alarm states of each GRPO iteration, the same states for both runs. Shaded: 95% Wilson intervals.

4.3 Ablation Study

Components. Tables 2–4 use R2R-CE, and Appendix D varies the alarm level and the judge query interval. In Table 2, without the judge, the recovery policy acts from the first step and SR falls to 59.9, below the base ($p = 3 \times 10^{-12}$ against the full system). Without the recovery policy, clearing at the alarm still raises SR above the base, to 65.6 ($p = 0.018$), but below the full system ($p = 1 \times 10^{-4}$): most of the gain comes from the recovery policy. Without clearing, the recovery policy takes over the retained history and reaches 67.4, not significantly below the full system ($p = 0.24$): a policy trained to recover copes with the retained history better than the untrained policies of Section 2, and clearing still matters in training (below).

Triggers. Table 3 changes when clearing starts, with and without the recovery policy. A random trigger with as many alarms at the same relative depths (Appendix D.2) lowers SR below the base, to 58.2 with clearing alone and 60.4 with the recovery policy. A ground-truth trigger at the first step with $h_t > S_d$ applies the intervention of Section 2.2 to the deployed base: clearing alone adds 5.2 points to SR, as for the policies of Section 2. With the recovery policy it reaches 71.2, as it catches 76.3% of base failures against 56.3% for the judge; the headroom lies in detection.

Table 2: Core components.

Judge	Clear	Recovery	SR↑	SPL↑
✗	✗	✗	63.8	59.0
✗	✗	✓	59.9	54.5
✓	✓	✗	65.6	59.0
✓	✗	✓	67.4	60.5
✓	✓	✓	69.0	62.3

Table 3: Alarm trigger, SR.

Trigger	Clear only	+ Recovery
Random	58.2	60.4
Judge	65.6	69.0
Ground truth	69.0	71.2

Table 4: Recovery training.

Training	SR↑	SPL↑
None (clear only)	65.6	59.0
SFT	62.9	57.6
$\text{GRPO}_{\text{start}}$	66.2	59.5
$\text{GRPO}_{\text{keep}}$	66.2	60.3
GRPO (ours)	69.0	62.3

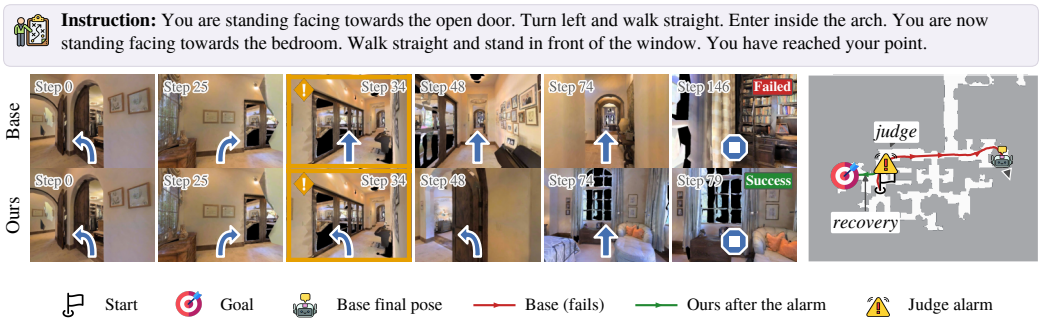


Figure 7: **Qualitative result on RxR-CE.** Rows share frames until the alarm (amber border), after which **MOON** clears the history and the recovery policy acts; last column: end of each run.

Recovery Training. After clearing, sampling finds more recoveries than a greedy rollout, as in Section 2.4: on the 527 alarmed R2R-CE episodes, at least one of eight samples succeeds in 56.0% against 35.1% for the greedy rollout, and GRPO raises greedy success to 47.1%. In Table 4, SFT on shortest-path demonstrations falls below the base (62.9), as the expert heads for a goal the cleared policy cannot see, while $\text{GRPO}_{\text{start}}$ (rollouts from the episode start) and $\text{GRPO}_{\text{keep}}$ (history retained in training and at deployment) both give 66.2. Clearing also makes training more efficient (Figure 6): at the first iteration, pass@8 is 50.8% with cleared rollouts against 22.7% with retained ones, so fewer groups carry no signal, although the two runs end at similar closed-loop SR (Table 11).

Efficiency. The two adapters increase the parameters of the frozen base by 0.81% and its time per action by 1.8% (Table 5). The judge is trained on 200,529 decision points and the recovery policy on 20,480 rollouts from 2,560 alarm states, against 4.92M samples for the base.

Table 5: Parameters and latency

Component	Params	Latency
Base (frozen)	4.44B	498 ms
+ Judge	+23.6M	–
+ Recovery	+12.4M	–
Full system	+36.0M ↑0.81%	507 ms ↑1.8%

5 Conclusion

We asked how visual history affects recovery after a deviation. Paired interventions on four general-purpose VLN policies show that the retained history helps on route but hinders recovery off route, where it keeps the policy from turning. After clearing, sampling often finds successes that a single greedy rollout misses. **MOON** adds two lightweight modules to a frozen policy: a deviation judge decides when to clear the history, and a GRPO-trained recovery policy turns more of these successes into a single greedy rollout. With 0.81% additional parameters, it raises the success rate of its base by 5.2 points on R2R-CE and 6.0 on RxR-CE, improves zero-shot object-goal navigation, and runs on a real robot without retraining. The appendix further reports the analysis protocol with per-policy results and history substitutions, ablations of the alarm level, the judge query interval and the training recipe, zero-shot object-goal navigation, the real-robot setup with its digital twin, and the prompts. Beyond navigation, many embodied policies act on a history of their own observations and actions, and an open question for future work is how such policies can learn when that history stops helping.

AI use statement

We used generative AI for language polishing, literature retrieval, illustrative graphics, and feedback on research methodology and interpretation. The authors take responsibility for the final content.

Ethics statement

We test navigation in a digital twin before real-world trials; broader deployment requires physical-safety and visual-privacy safeguards.

Reproducibility statement

Implementation details, evaluation protocols, and prompts are provided in Appendices B–G.

References

Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. ETPNav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024. doi: 10.1109/TPAMI.2024.3386695. arXiv:2304.03047.

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

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3674–3683, 2018b. arXiv:1711.07280.

Anastasios N. Angelopoulos and Stephen Bates. A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv preprint arXiv:2107.07511*, 2021.

Shizhe Chen, Pierre-Louis Guhur, Cordelia Schmid, and Ivan Laptev. History aware multimodal transformer for vision-and-language navigation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

An-Chieh Cheng, Yandong Ji, Zhaojing Yang, Zaitian Gongye, Xueyan Zou, Jan Kautz, Erdem Bıyık, Hongxu Yin, Sifei Liu, and Xiaolong Wang. NaVILA: Legged robot vision-language-action model for navigation. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2412.04453.

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

Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9328–9337, 2019. doi: 10.1109/ICCV.2019.00942. arXiv:1904.08980.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. arXiv:1905.11979.

Gang He, Zhenyang Liu, Kepeng Xu, Li Xu, Tong Qiao, Wenxin Yu, Chang Wu, and Weiying Xie. Nipping the drift in the bud: Retrospective rectification for robust vision-language navigation. *arXiv preprint arXiv:2602.06356*, 2026. doi: 10.48550/arXiv.2602.06356. URL `https://arxiv.org/abs/2602.06356`.

Keji He, Ya Jing, Yan Huang, Zhihe Lu, Dong An, and Liang Wang. Memory-adaptive vision-and-language navigation. *Pattern Recognition*, 153:110511, 2024. doi: 10.1016/j.patcog.2024.110511. URL `https://doi.org/10.1016/j.patcog.2024.110511`.

Yicong Hong, Zun Wang, Qi Wu, and Stephen Gould. Bridging the gap between learning in discrete and continuous environments for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. doi: 10.1109/CVPR52688.2022.01500. arXiv:2203.02764.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2106.09685.

Jaehwan Jeong, Evelyn Zhu, Jinying Lin, Emmanuel Jaimes, Tuan-Anh Vu, Jungseock Joo, Sangpil Kim, and M. Khalid Jawed. Your vision-language-action model already has attention heads for path deviation detection. *arXiv preprint arXiv:2603.13782*, 2026. URL https://arxiv.org/abs/2603.13782.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

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, 2020. arXiv:2004.02857.

Jacob Krantz, Aaron Gokaslan, Dhruv Batra, Stefan Lee, and Oleksandr Maksymets. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. arXiv:2110.02207.

Haoyuan Li, Rui Liu, Hehe Fan, and Yi Yang. Let's reward step-by-step: Step-aware contrastive alignment for vision-language navigation in continuous environments. *arXiv preprint arXiv:2603.09740*, 2026a.

Jiangyang Li, Cong Wan, Songlin Dong, Chenhao Ding, Qiang Wang, Zhiheng Ma, and Yihong Gong. Trajectory-diversity-driven robust vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9143–9154, 2026b. arXiv:2603.15370.

Weijia Liufu, Xiaoyu Guo, Ruiyi Chen, Jingzhi Liu, Kaidong Zhang, Xiwen Liang, Jianqi Lin, Dawei Sun, Yuze Wang, Rongtao Xu, Bingqian Lin, Bowen Yang, Tongtong Cao, Bowen Peng, Dongyu Zhang, Guangrun Wang, Min Wang, Liang Lin, and Xiaodan Liang. RePO-VLA: Recovery-driven policy optimization for vision-language-action models. *arXiv preprint arXiv:2605.09410*, 2026. URL https://arxiv.org/abs/2605.09410.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. Self-monitoring navigation agent via auxiliary progress estimation. In *International Conference on Learning Representations (ICLR)*, 2019a. URL https://openreview.net/forum?id=r1GAsjC5Fm. arXiv:1901.03035.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6725–6733, 2019b. doi: 10.1109/CVPR.2019.00689. arXiv:1903.01602.

Zhangyang Qi, Zhixiong Zhang, Yizhou Yu, Jiaqi Wang, and Hengshuang Zhao. VLN-R1: Vision-language navigation via reinforcement fine-tuning. *arXiv preprint arXiv:2506.17221*, 2025.

Botao Ren, Junjun Hu, Xinda Xue, Minghua Luo, Jintao Chen, Haochen Bai, Liangliang You, and Mu Xu. AstraNav-Memory: Contexts compression for long memory. *arXiv preprint arXiv:2512.21627*, 2025. doi: 10.48550/arXiv.2512.21627. URL https://arxiv.org/abs/2512.21627.

Blake A. Richards and Paul W. Frankland. The persistence and transience of memory. *Neuron*, 94 (6):1071–1084, 2017. doi: 10.1016/j.neuron.2017.04.037.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *PMLR*, pp. 627–635, 2011. arXiv:1011.0686.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9339–9347, 2019. arXiv:1904.01201.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Xiangyu Shi, Zerui Li, Wenqi Lyu, Jiatong Xia, Feras Dayoub, Yanyuan Qiao, and Qi Wu. SmartWay: Enhanced waypoint prediction and backtracking for zero-shot vision-and-language navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2025. arXiv:2503.10069.

Max Sobol Mark, Jacky Liang, Maria Attarian, Chuyuan Fu, Debidatta Dwibedi, Dhruv Shah, and Aviral Kumar. BPP: Long-context robot imitation learning by focusing on key history frames. *arXiv preprint arXiv:2602.15010*, 2026.

Samson Sun, Tianyi Yang, Tengyue Wang, Yikai Xue, Zhengjie Xu, Lingming Zhang, Qichen Zhang, Chao Liang, and Zhipeng Zhang. Beyond isolation: A unified benchmark for general-purpose navigation. In *Robotics: Science and Systems (RSS)*, 2026. arXiv:2605.09441.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer, New York, 2005. doi: 10.1007/b106715.

Jiahang Wang, Yirong Yang, Yanqing Zhu, Minghua Luo, Shichao Xie, Fei Liu, and Mu Xu. ReflectVLN: Training vision-language navigation agents with reflective reasoning. *arXiv preprint arXiv:2607.12680*, 2026.

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 (ICCV)*, pp. 11975–11986, 2023. doi: 10.1109/ICCV51070.2023.01103. arXiv:2307.15644.

Meng Wei, Chenyang Wan, Xiqian Yu, Tai Wang, Yuqiang Yang, Xiaohan Mao, Chenming Zhu, Wenzhe Cai, Hanqing Wang, Yilun Chen, Xihui Liu, and Jiangmiao Pang. StreamVLN: Streaming vision-and-language navigation via SlowFast context modeling. *arXiv preprint arXiv:2507.05240*, 2025.

Meng Wei, Chenyang Wan, Jiaqi Peng, Xiqian Yu, Yuqiang Yang, Delin Feng, Wenzhe Cai, Chenming Zhu, Tai Wang, Jiangmiao Pang, and Xihui Liu. Ground slow, move fast: A dual-system foundation model for generalizable vision-and-language navigation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2512.08186.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, pp. 2564–2575, 2020. arXiv:2010.14876.

Zihao Xin, Wentong Li, Yixuan Jiang, Bin Wang, Runmin Cong, Jie Qin, and Shengjun Huang. DecoVLN: Decoupling observation, reasoning, and correction for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 32410–32420, 2026. arXiv:2603.13133.

Xuan Yao, Junyu Gao, and Changsheng Xu. NavMorph: A self-evolving world model for vision-and-language navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. arXiv:2506.23468.

Xuan Yao, Yuze Zhu, Junyu Gao, Zongmeng Wang, and Changsheng Xu. SC^2-WM: A self-correcting world model with closed-loop feedback for vision-and-language navigation in continuous environments. In *International Conference on Machine Learning (ICML)*, 2026. arXiv:2608.07548.

Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. VLFM: Vision-language frontier maps for zero-shot semantic navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024a. arXiv:2312.03275.

Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. HM3D-OVON: A dataset and benchmark for open-vocabulary object goal navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024b. arXiv:2409.14296.

Zhuoyuan Yu, Yuxing Long, Zihan Yang, Chengyan Zeng, Hongwei Fan, Jiyao Zhang, and Hao Dong. CorrectNav: Self-correction flywheel empowers vision-language-action navigation model. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026. arXiv:2508.10416.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? *arXiv preprint arXiv:2504.13837*, 2025. NeurIPS 2025.

Jeffrey M. Zacks, Nicole K. Speer, Khena M. Swallow, Todd S. Braver, and Jeremy R. Reynolds. Event perception: A mind–brain perspective. *Psychological Bulletin*, 133(2):273–293, 2007. doi: 10.1037/0033-2909.133.2.273.

Shuang Zeng, Dekang Qi, Xinyuan Chang, Feng Xiong, Shichao Xie, Xiaolong Wu, Shiyi Liang, Mu Xu, Xing Wei, and Ning Guo. JanusVLN: Decoupling semantics and spatiality with dual implicit memory for vision-language navigation. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2509.22548.

Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and He Wang. NaVid: Video-based VLM plans the next step for vision-and-language navigation. In *Robotics: Science and Systems (RSS)*, 2024. arXiv:2402.15852.

Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-NaVid: A video-based vision-language-action model for unifying embodied navigation tasks. In *Robotics: Science and Systems (RSS)*, 2025. doi: 10.15607/RSS.2025.XXI.013. arXiv:2412.06224.

Jiazhao Zhang, Anqi Li, Yunpeng Qi, Minghan Li, Jiahang Liu, Shaoan Wang, Haoran Liu, Gengze Zhou, Yuze Wu, Xingxing Li, Yuxin Fan, Wenjun Li, Zhibo Chen, Fei Gao, Qi Wu, Zhizheng Zhang, and He Wang. Embodied navigation foundation model. In *International Conference on Learning Representations (ICLR)*, 2026a. arXiv:2509.12129.

Jiazhao Zhang, Gengze Zhou, Hale Yin, Yiyang Huang, Zixing Lei, Qihang Peng, Haoqi Yuan, Jie Zhang, Xudong Guo, Xiaoyue Chen, An Yang, Fei Huang, Zhibo Yang, Junyang Lin, Dayiheng Liu, Jingren Zhou, et al. Qwen-RobotNav technical report: A scalable navigation model designed for an agentic navigation system. *arXiv preprint arXiv:2606.18112*, 2026b.

Lingfeng Zhang, Zeying Gong, Xiaoshuai Hao, Haoxiang Fu, Qiang Zhang, Mingliang Zhou, Hangjun Ye, Xiaojun Liang, Junwei Liang, and Wenbo Ding. FutureNav: Unified world-action modeling for vision-and-language navigation. *arXiv preprint arXiv:2606.30367*, 2026c.

Weiye Zhu, Zekai Zhang, Xiangchen Wang, Hewei Pan, Teng Wang, Tiantian Geng, Rongtao Xu, and Feng Zheng. NaVIDA: Vision-language navigation with inverse dynamics augmentation. *arXiv preprint arXiv:2601.18188*, 2026.

Ziyu Zhu, Xilin Wang, Yixuan Li, Zhuofan Zhang, Xiaojian Ma, Yixin Chen, Baoxiong Jia, Wei Liang, Qian Yu, Zhidong Deng, Siyuan Huang, and Qing Li. Move to understand a 3D scene: Bridging visual grounding and exploration for efficient and versatile embodied navigation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. arXiv:2507.04047.

APPENDIX CONTENTS

A Related Work **16**

A.1 Vision-and-Language Navigation and Visual Memory 16

A.2 Deviation Detection and Recovery 16

A.3 Corrective Training and Reinforcement Learning 16

A.4 History Dependence in Imitation Learning 16

A.5 Calibrated Thresholds and Reinforcement Fine-Tuning 17

B Details of the Motivating Analyses **17**

B.1 State Populations and Uncertainty 17

B.2 History Constructions and Comparators 18

B.3 History Substitution 20

B.4 Action Continuation 20

B.5 Greedy Selection and Sampling Coverage 20

C Method Details **21**

C.1 Deployment Loop 21

C.2 Base Policy 21

C.3 Judge 21

C.4 Calibration 22

C.5 Training Start States 22

C.6 Counterfactual Stop Advantage 23

C.7 Recovery Objective 23

C.8 Training and Decoding 23

D Additional Results **24**

D.1 Training-Side Ablation over Iterations 24

D.2 Judge Counts 24

D.3 Alarm Level 24

D.4 Judge Query Interval 25

D.5 Zero-Shot Object Navigation 25

E Real-Robot Deployment **26**

E.1 Robot and Control Loop 26

E.2 Digital Twin Construction 26

E.3 Paired Runs 26

F Qualitative Examples **28**

G Prompts **35**

G.1 Navigation Policy 35
G.2 Deviation Judge 35

A Related Work

A.1 Vision-and-Language Navigation and Visual Memory

Vision-and-language navigation has progressed from path selection on predefined navigation graphs to action execution in continuous environments (Anderson et al., 2018b; Krantz et al., 2020). History-aware models such as HAMT incorporate past observations into cross-modal reasoning (Chen et al., 2021). More recent approaches, including NaVid and NaVILA, build on pretrained vision-language models to predict navigation actions from instructions, current observations, and visual history (Zhang et al., 2024; Cheng et al., 2025; Zhu et al., 2026). To keep this history compact, they sample frames, merge visual tokens online, or stream caches (Cheng et al., 2025; Zhu et al., 2026; Zhang et al., 2025; Wei et al., 2025): NaVILA and NaVIDA sample a fixed number of past frames uniformly, Uni-NaVid keeps fewer tokens for older frames, and StreamVLN pairs a key–value cache of recent steps with a slowly updated memory. Other methods select history frames by semantic relevance, visual diversity, and temporal coverage (Xin et al., 2026), decouple semantics and spatiality with a dual implicit memory (Zeng et al., 2026), suppress history noise, or compress context (He et al., 2024; Ren et al., 2025). However, memory ablations in several of these works retrain the policy (Zhang et al., 2025; Wei et al., 2025), and evaluation largely relies on full-episode navigation performance, without distinguishing the benefits and costs of history between on-route and off-route states. We instead keep the policies fixed, intervene only on the history they receive, and compare on-route states with states after a deviation.

A.2 Deviation Detection and Recovery

Progress monitors estimate how far an agent has advanced toward the goal (Ma et al., 2019a), and backtracking methods use such estimates, global signals, or history-aware planning to revisit earlier locations and reconsider the route (Ma et al., 2019b; Ke et al., 2019; Shi et al., 2025). The closest prior detector reads path deviation from attention heads of the frozen NaVILA checkpoint of Section 2 and hands control to a separate controller, leaving the VLA context untouched (Jeong et al., 2026). At an alarm, this controller drives the robot back to the last pose judged normal. Its authors list this unsynchronized context as a limitation. In ReflectVLN, an execution agent signals a deviation, and a separate intention agent reflects on it to produce a corrective subtask (Wang et al., 2026). **MOON** also raises its alarm on a frozen policy, using a trained adapter that also reads the chunk the policy proposes next, but it then leaves the agent at its current pose, clears the policy's visual history, and hands control to a recovery adapter on the same backbone.

A.3 Corrective Training and Reinforcement Learning

CorrectNav finds where its trained model still deviates on training trajectories and retrains on synthesized corrections, keeping earlier observations as history while supervising only the subsequent corrective trajectory (Yu et al., 2026). BudVLN identifies conflicts between corrective actions from error states and the original instruction, and returns to valid historical states to construct aligned supervision (He et al., 2026). DecoVLN builds corrective supervision for state–action pairs from geodesic deviation, keeping only samples within a trust region around the expert trajectory (Xin et al., 2026), and SACA resamples a repair from the divergence point of a failed trajectory (Li et al., 2026a). Reinforcement learning with task rewards also improves navigation robustness (Li et al., 2026b; Qi et al., 2025), and BudVLN and SACA combine corrective data with GRPO. These evaluations, however, do not isolate the effect of visual history on recovery under a fixed policy, so improved recovery does not show whether the history supplies useful cues for returning to the instructed route or interferes with adaptation to the current state. In manipulation, RePO-VLA clears the history only in recovery training data, so that corrective actions condition on the current state rather than on the earlier failure, with no reset or failure detector at deployment (Liufu et al., 2026). **MOON** instead keeps the base policy frozen, clears the history at deployment when its judge raises an alarm, and trains a recovery adapter with GRPO from the cleared alarm states.

A.4 History Dependence in Imitation Learning

Imitation policies conditioned on past observations and actions can rely on correlates of the expert's past behavior instead of the current observation (de Haan et al., 2019; Wen et al., 2020). Access

to more information can then worsen performance under distribution shift (de Haan et al., 2019), and copycat policies predict the expert's previous action instead of the next one (Wen et al., 2020). Codevilla et al. (2019) describe an inertia problem in end-to-end driving, where a policy conditioned on its own speed learns that a stopped car tends to stay stopped. Once a policy leaves the expert distribution, it meets histories that its training data did not cover, and its errors compound (Ross et al., 2011; Sobol Mark et al., 2026). Proposed remedies either change training, through targeted interventions (de Haan et al., 2019), removing information about the previous action (Wen et al., 2020), and auxiliary speed prediction (Codevilla et al., 2019), or replace the raw history with key frames in both training and deployment (Sobol Mark et al., 2026). The inertia we study arises from the retained visual history of navigation policies and becomes harmful after a deviation, when the state calls for a change of course. Most of this harm remains when the history is replaced by length-matched copies of the current frame or by noise (Section 2.3). Where these remedies reshape training or replace the history throughout an episode, **MOON** keeps the base policy frozen and clears its history only after an alarm.

A.5 Calibrated Thresholds and Reinforcement Fine-Tuning

Split conformal prediction sets a threshold from held-out calibration scores and bounds the error rate on new data that are exchangeable with them (Vovk et al., 2005; Angelopoulos & Bates, 2021). Jeong et al. (2026) instead choose attention heads on labeled training episodes and set their threshold by grid search under an empirical false-alarm constraint. GRPO normalizes rewards within a group of sampled outputs, which gives a baseline without a learned critic (Shao et al., 2024), and reinforcement learning with verifiable rewards is effective at making behaviors that a policy can already sample more reliable (Yue et al., 2025). **MOON** calibrates its threshold on episodes that the frozen policy completes, bounding how often such episodes raise an alarm, and trains its LoRA recovery adapter (Hu et al., 2022) with GRPO only from cleared alarm states, rewarding the part of each rollout after the alarm.

B Details of the Motivating Analyses

B.1 State Populations and Uncertainty

All interventions keep the navigation policy fixed and retain the current observation. Each episode is capped at 400 steps. NaVIDA samples eight frames uniformly from H_t and NaVILA seven, each alongside the current frame; StreamVLN keeps a key–value cache of its most recent 32 steps, generated actions included, and an eight-frame block for each earlier window; and Uni-NaVid merges visual tokens online. For NaVIDA and NaVILA, increasing k beyond eight spreads a fixed number of subsampled frames over a longer time span. Decoding is greedy except in the sampling study of Section 2.4, and the simulator is deterministic, so the rollout that retains the history is the remainder of the original trajectory. NaVIDA, StreamVLN and NaVILA run from the `NaVIDA-3B`, `StreamVLN_v1_3` and `navila-llama3-8b-8f` checkpoints. Uni-NaVid runs from `uninavid-7b-interim-from-dexbotic`, an interim 7B checkpoint from Dexbotic. We did not use the official Uni-NaVid weights. Deviation onset uses the route-deviation threshold $S_d = 2\,\text{m}$ of Section 2.1. The delayed condition uses the first available decision point at least 10 atomic steps after onset. At 749 of the 779 delayed states (96.1%), the agent is still more than $2\,\text{m}$ from the route. On-route controls are sampled from episodes that never deviate, at the decision point nearest the midpoint of each episode. Across the four policies, $1{,}465$ of the $2{,}400$ rollouts never deviate; of the 935 that deviate, 20 do so after their last decision point, which leaves 915 onset states. Table 6 records the distinct populations used in the figures. State-paired success differences use $2{,}000$ bootstrap resamples with seed 41. Within a policy, each of the on-route, onset and delayed populations holds at most one state per episode, so the state bootstrap is also an episode bootstrap. Pooled intervals resample the states of the four policies independently; resampling episodes jointly across the policies moves the bounds of the pooled success differences by less than one point. The 95% intervals are pointwise, not simultaneous bands. Equivalence comparisons use 90% paired bootstrap intervals and margins of ± 8 pp per policy and ± 5 pp pooled. We set these margins after reading the estimates of the shuffled and swapped success comparisons on the same 779 delayed states. An interval contained within the corresponding margin supports equivalence at that resolution, rather than exact equality.

Table 6: State counts for the analysis panels. After-deviation action observations include both onset and delayed states from some of the same episodes.

Population	NaVIDA	StreamVLN	NaVILA	Uni-NaVid
On-route history controls	359	337	402	367
Delayed history interventions	205	210	163	201
Greedy onset interventions	238	248	198	231
Sampling: retained failures	80	80	80	80
Action continuation, onset and delayed	279	302	292	391

B.2 History Constructions and Comparators

In Figure 2b, k counts past observations in addition to the current frame, with positions proportional to $\log_2(k+1)$. All denotes each policy's original history mechanism, including cached context where applicable, and is separated from numeric frame counts by an axis break. The full-history control in Table 7 replays the recorded frames for Uni-NaVid. For StreamVLN it also replays the frames and forces the recorded answers into the key–value cache at each recorded decision step. NaVIDA and NaVILA use the standard full-history condition. Shuffled and swapped paired contrasts use the native full-history reference. Figure 2c compares the full-history control with prefix + current, both relative to current-frame-only. The full-length retained history of the main text is this control in Figure 2c and Section 2.2, and the native reference in the shuffled and swapped contrasts; the two need not give identical estimates. Prefix + current retains the prefix up to deviation onset and the current observation, removing intervening post-onset history; it is the history H_{t_d+1} of Section 2.2, which ends with the onset frame. The 1.5, 1.0 and 0.5 m prefixes stop before the first frame farther than that distance from the route. They leave pooled deficits of 9.9, 8.2 and 7.4 points, and the deficit of the onset prefix is significant for three of the four policies (NaVILA: 4.3, $[-3.7, 12.3]$). On route, histories whose last frame lies 11 or 21 steps before the current frame raise success over current-frame-only by 27.4, 26.4, 23.5 and 8.8 points, and by 17.6, 10.1, 22.5 and 13.7 points (NaVIDA, StreamVLN, NaVILA, Uni-NaVid), with every 95% interval excluding zero. A swapped history is drawn from an on-route episode in another scene, while token-matched noise replaces visual content at matched token count. Expert + tail replays the dataset expert's actions along the reference route up to the waypoint nearest the current pose, then follows a shortest-path tail to the same pose. In 359 of the 779 delayed states that waypoint is the episode start, so the reconstruction contains no route segment. Because the tail must reach the current pose, which lies a median 3.85 m from the route, its final steps usually lie outside the S_d band. Expert + tail is therefore not a history free of off-route content, and the main text does not use it as one. Route-only history keeps only the route segment and is defined on the remaining 420 states. The own-history control matches the expert reconstruction's length. Table 7 includes all seventeen tested conditions, including comparisons whose intervals contain zero.

Table 7: Success-rate differences from current-frame-only, in points; a negative value is a deficit of the same size in the main text. Each cell gives the estimate and its pointwise 95% paired state-bootstrap interval. The default population is 779 delayed states. Route-only uses 420 eligible states. Pooled estimates weight states equally.

Construction	NaVIDA	StreamVLN	NaVILA	Uni-NaVid	Pooled
	−7.3	−14.8	−0.6	−5.5	−7.4
Prefix 0.5 m	[−14.2, 0.0]	[−22.4, −7.6]	[−8.6, 7.4]	[−11.4, 0.5]	[−10.8, −3.7]
	−10.7	−12.4	−3.1	−5.5	−8.2
Prefix 1.0 m	[−18.5, −3.4]	[−19.5, −5.7]	[−11.0, 4.9]	[−11.4, 0.0]	[−11.8, −4.6]
	−10.7	−12.4	−8.0	−8.0	−9.9
Prefix 1.5 m	[−18.1, −3.4]	[−19.5, −5.2]	[−16.0, 0.0]	[−13.9, −2.5]	[−13.2, −6.5]
	−9.3	−15.7	−4.3	−9.5	−10.0
Prefix + current	[−16.6, −2.0]	[−22.9, −9.0]	[−12.3, 3.7]	[−15.4, −3.5]	[−13.5, −6.5]
	−7.3	−3.8	1.2	−3.0	−3.5
Last 2	[−13.7, −1.0]	[−8.6, 1.0]	[−6.7, 9.2]	[−9.0, 2.5]	[−6.7, −0.3]
	−7.3	−4.8	0.6	−4.0	−4.1
Last 4	[−14.6, 0.0]	[−12.9, 2.9]	[−7.4, 8.6]	[−9.5, 1.5]	[−7.8, −0.5]
	−12.7	−2.9	−1.2	−7.5	−6.3
Last 8	[−19.5, −5.9]	[−10.0, 4.8]	[−9.8, 7.4]	[−12.9, −2.0]	[−9.9, −2.8]
	−11.7	−10.0	−4.9	−9.5	−9.2
Last 16	[−19.0, −4.4]	[−17.1, −2.9]	[−12.9, 3.1]	[−15.4, −3.5]	[−12.7, −5.6]
	−15.1	−16.2	−8.0	−12.4	−13.2
Full-history control	[−21.5, −8.8]	[−23.3, −9.0]	[−16.0, 0.0]	[−18.4, −6.5]	[−16.6, −9.6]
	−14.1	−17.6	−4.3	−13.9	−13.0
Shuffled	[−21.5, −7.3]	[−25.2, −10.5]	[−12.3, 3.7]	[−19.9, −8.5]	[−16.3, −9.4]
	−8.3	2.4	−1.8	2.0	−1.4
Swapped, last 4	[−15.1, −1.5]	[−4.3, 9.5]	[−9.2, 5.5]	[−4.0, 8.5]	[−4.9, 2.2]
	−16.6	−15.2	−7.4	−9.5	−12.5
Swapped, full	[−23.4, −9.8]	[−22.4, −8.6]	[−16.0, 0.6]	[−15.9, −3.5]	[−16.0, −8.9]
	−9.8	−8.6	−3.7	−11.9	−8.7
Filler current	[−16.1, −3.4]	[−15.7, −1.4]	[−11.7, 4.3]	[−17.9, −6.0]	[−12.2, −5.3]
	−13.7	−6.7	−2.5	−7.0	−7.7
Token-matched noise	[−20.5, −7.3]	[−14.3, 1.0]	[−10.4, 5.5]	[−13.9, 0.0]	[−11.2, −3.9]
	−16.6	−13.3	−4.9	−13.9	−12.6
Expert + tail	[−22.9, −10.2]	[−21.0, −5.7]	[−12.9, 3.1]	[−19.9, −8.5]	[−16.0, −9.0]
	−9.7	−8.0	1.1	−7.5	−6.4
Route only	[−20.4, 0.0]	[−16.8, 0.8]	[−10.1, 12.4]	[−17.2, 2.2]	[−11.4, −1.2]
	−14.1	−15.7	−6.7	−11.9	−12.5
Own, length-matched	[−21.0, −7.3]	[−23.3, −9.0]	[−14.7, 1.2]	[−17.9, −6.5]	[−15.9, −8.7]

B.3 History Substitution

We replace only the content of the history, keeping its length fixed. At the 779 delayed states, shuffling its (frame, action) pairs leaves a deficit of 13.0 points (95% CI $[9.4, 16.3]$), and swapping it for a full-length history from another scene leaves 12.5 points ($[8.9, 16.0]$), against 13.2 points for the full-length retained history. Relative to the full-length retained history, both stay within 5 points when pooled and 8 points per policy; these equivalence margins were set after these estimates had been read. On route, the same substitutions lower success for every policy by 5–11 and 6–20 points relative to the full-length retained history, whereas after a deviation they leave the deficit nearly unchanged. Filling the same length with copies of the current frame or with noise still leaves deficits of 8.7 and 7.7 points.

B.4 Action Continuation

Continuation is scored at states where the previously executed action is a forward step. A decision continues when its greedy action chunk contains no turn, so a chunk that ends in a stop without turning also continues. Scoring such chunks as non-continuing lowers the retained-history rate from 48.1% to 44.9% at onset and from 46.8% to 41.8% at delayed states, and leaves clearing and the expert unchanged. The shortest-path expert is queried at the same pose and continues when its first actions, as many as the retained-history chunk holds, contain no turn. States where the expert returns no action, all within the 3 m success radius, are excluded. Shuffled and swapped histories differ from the full-length retained history by less than 4 points in pooled point estimates (50.4% and 51.3% on route, 51.7% and 45.5% at the onset, 48.9% and 47.2% at the delayed states). Per-policy differences vary in sign, but for every policy and population both stay above clearing and the expert. Continuation intervals resample episodes separately within each population. Table 8 reports the rates per policy.

Table 8: Continuation rates per policy, in percent, with 95% episode-bootstrap intervals. States follow a forward step, and states where the expert returns no action are excluded. Pooled rows are the values in Figure 3a. Two on-route states each of NaVILA and Uni-NaVid have no swapped history, leaving 235 and 316 states (942 pooled).

Policy	n	Retained	Cleared	Shuffled	Swapped	Expert
On route						
NaVIDA	172	39.0 [32.0, 45.9]	22.7 [16.3, 29.1]	39.5 [32.6, 47.1]	37.8 [30.8, 45.3]	32.0 [25.0, 38.4]
StreamVLN	219	36.5 [30.1, 42.9]	20.1 [15.1, 25.6]	39.7 [33.3, 46.1]	37.0 [30.6, 43.4]	32.4 [26.5, 38.8]
NaVILA	237	76.8 [71.3, 82.3]	18.1 [13.5, 23.2]	51.5 [45.6, 57.8]	68.1 [62.1, 74.0]	35.4 [29.1, 41.8]
Uni-NaVid	318	43.7 [38.1, 49.1]	21.1 [17.0, 25.8]	62.9 [57.2, 68.2]	56.0 [50.3, 61.4]	26.1 [21.4, 30.8]
Pooled	946	49.5 [46.2, 52.7]	20.4 [17.9, 22.8]	50.4 [47.3, 53.6]	51.3 [48.3, 54.6]	31.0 [28.1, 33.9]
Onset						
NaVIDA	160	35.0 [27.5, 42.5]	11.9 [7.5, 16.9]	40.6 [32.5, 48.8]	24.4 [18.1, 31.2]	1.2 [0.0, 3.1]
StreamVLN	169	33.1 [26.0, 40.2]	13.6 [8.9, 18.9]	38.5 [31.9, 46.2]	33.1 [26.0, 40.2]	1.8 [0.0, 4.1]
NaVILA	182	68.7 [62.1, 75.3]	5.5 [2.2, 9.3]	48.4 [40.7, 55.5]	56.0 [48.4, 63.7]	1.1 [0.0, 2.7]
Uni-NaVid	215	52.1 [45.6, 59.1]	12.6 [8.4, 17.2]	73.0 [67.0, 79.1]	61.9 [55.3, 68.4]	3.3 [1.4, 5.6]
Pooled	726	48.1 [44.5, 51.8]	10.9 [8.8, 13.4]	51.7 [47.9, 55.2]	45.5 [41.9, 49.2]	1.9 [1.0, 3.0]
Delayed						
NaVIDA	119	40.3 [31.9, 49.6]	16.0 [10.1, 22.7]	39.5 [31.1, 48.7]	28.6 [21.0, 37.0]	3.4 [0.8, 6.7]
StreamVLN	133	33.8 [26.3, 42.1]	12.8 [7.5, 18.8]	36.8 [29.3, 45.1]	36.8 [28.6, 45.1]	2.3 [0.0, 5.3]
NaVILA	110	60.0 [50.9, 69.1]	1.8 [0.0, 4.5]	48.2 [39.1, 57.3]	63.6 [54.5, 72.7]	5.5 [1.8, 10.0]
Uni-NaVid	176	52.8 [46.0, 60.2]	6.8 [3.4, 10.8]	64.8 [58.0, 72.2]	57.4 [50.0, 64.8]	7.4 [4.0, 11.4]
Pooled	538	46.8 [42.8, 51.3]	9.3 [6.9, 11.9]	48.9 [44.8, 53.0]	47.2 [43.1, 51.5]	4.8 [3.2, 6.7]

B.5 Greedy Selection and Sampling Coverage

The greedy selection oracle takes the successful continuation whenever either retained or cleared history succeeds at the same onset state. On all 915 onset states it reaches 35.0% success, 6.3 points above always clearing (Figure 3b): clearing rescues 195 of the 790 states where the retained rollout fails and breaks 58 of the 125 where it succeeds, and both rollouts fail on 65.0% of the states.

The per-policy bars in Figure 3b show pointwise Wilson intervals computed from the corresponding success counts. The oracle's 35.0% rate bounds selection between those two frozen greedy continuations, rather than recovery with a trained policy or a larger sampling budget. Sampling states are drawn with a fixed seed and route-distance stratification from retained-greedy failures, selecting 80 states per policy. Each condition produces eight continuations with temperature 1.0, top-$p = 0.95$, and paired random seeds. For a state with c successful samples among eight, the coverage estimate at budget k is $1 - \binom{8-c}{k} / \binom{8}{k}$, taking the numerator as zero when $8 - c < k$. Reported $\mathrm{pass@}k$ averages these estimates over states, with pointwise state-bootstrap intervals. These offline coverage estimates assume success can be recognized among sampled continuations and are not deployment success rates. Table 9 reports policy-level sampling coverage.

Table 9: Offline coverage after eight samples on retained-greedy failures (80 onset states per policy). Values are percentages with 95% state-bootstrap intervals. The final column is the fraction with no successful cleared continuation among eight samples.

Policy	Retained pass@8	Cleared pass@8	Cleared: all fail
NaVIDA	36.2 [26.2, 47.5]	66.2 [56.2, 76.2]	33.8
StreamVLN	32.5 [22.5, 43.8]	68.8 [58.8, 78.8]	31.2
NaVILA	36.2 [26.2, 47.5]	71.2 [61.3, 80.0]	28.7
Uni-NaVid	16.2 [8.8, 25.0]	40.0 [30.0, 51.2]	60.0

C Method Details

C.1 Deployment Loop

Algorithm 1 summarizes the deployment loop of Section 3.

Algorithm 1 MOON at deployment: alarm, history clearing and handover to the recovery policy

Require: instruction I, frozen policy π_θ, recovery policy $\pi_{\theta,\psi}$ (§3.2)
Require: judge f_ϕ, window W, threshold τ of equation 6
$\pi \leftarrow \pi_\theta$; $H \leftarrow \emptyset$
while the episode has not ended **do**
  $c_t \leftarrow$ greedy decoding of $\pi(\cdot \mid I, o_t, M(H))$
  if $\pi = \pi_\theta$ **and** c_t contains no stop **then**
    $E_t \leftarrow (I, o_t, H, c_t)$; compute m_t and $\bar{m}_t$ by equation 5
    if $\bar{m}_t > \tau$ **then**
      $H \leftarrow \emptyset$; $\pi \leftarrow \pi_{\theta,\psi}$ ▷ first alarm, $t = t_a$
      $c_t \leftarrow$ greedy decoding of $\pi_{\theta,\psi}(\cdot \mid I, o_t, M(\emptyset))$
    end if
  end if
  execute c_t, appending each step's (o, a) to H
end while

C.2 Base Policy

The frozen policy NaVIDA-Qwen3VL follows the official training recipe of NaVIDA (Zhu et al., 2026) with Qwen3-VL-4B-Instruct as the backbone, and Table 10 lists its configuration.

C.3 Judge

The lookahead set of equation 4 is $V_t = \{t+3, t+6, t+9, t+12, t+15\} \cap [0, T)$ in atomic steps, where T is the length of the episode. It is set to $\emptyset$ when it has fewer than three elements, so near the end of an episode a decision point is labeled LOST exactly when $h_t > S_d$. The input E_t contains up to nine frames spaced evenly over the history since the episode start, the last being o_t, together with the instruction, up to the last twelve executed atomic actions and the chunk c_t written as action names. The judge answers with one token. Its prompt also lists a third answer that no

Table 10: Training configuration of the base policy NaVIDA-Qwen3VL.

Configuration	Setting
Backbone	Qwen3-VL-4B-Instruct
Training data	0.82M (R2R-CE and RxR-CE), 0.76M (DAgger) and 3.35M (ScaleVLN); 4.92M samples in total (components rounded)
Stages	Stage I on R2R-CE and RxR-CE, DAgger collection, Stage II on all data
Epochs	1 per stage
Learning rate	2×10^{-5}, cosine decay
Global batch size	64 (Stage I), 128 (Stage II)
History frames	8, sampled uniformly
Image resolution	352×288
Precision and parallelism	BF16, FlashAttention-2, DeepSpeed ZeRO-1
Hardware	16 / 24 / 32 A100-80GB for Stage I / DAgger / Stage II (about 6 / 5 / 18 hours)

training row uses. The probability of that answer rescales $p_\phi(\mathrm{OK} \mid E_t)$ and $p_\phi(\mathrm{LOST} \mid E_t)$ by the same factor, so m_t is unchanged. Training rows come from rollouts of π_θ on the training splits of R2R-CE and RxR-CE and on ScaleVLN (Wang et al., 2023). On these rollouts, 64.5%, 55.4% and 56.0% of the steps that exceed S_d but are labeled OK lie in successful episodes for R2R-CE, RxR-CE and ScaleVLN, against less than 18% of the steps labeled LOST. Rows are drawn from every failed episode and an equal number of successful ones, at most 24 per episode, and the two larger datasets are then subsampled to the row count of R2R-CE. OK and LOST rows are drawn in equal proportion during training. The adapter ϕ is a LoRA of rank 32 with scaling factor 2 (LoRA alpha 64) on the query, key, value and output projections of the language model. Five of the checkpoints saved every 300 steps (900, 1,500, 2,100, 2,700 and 3,600) are compared by episode-level recall at $\alpha = 5\%$ on the selection half described below, and step 1,500 is used. The judge scores every decision point whose chunk contains no stop, with the window $W = 3$.

C.4 Calibration

The judge's training excludes 15% of the reference paths of the training splits, chosen by a hash of the path, and every scene contributes to both sides. The held-out paths are split into a selection half and a calibration half. For each dataset, the calibration episodes are the rollouts of π_θ on the calibration half that it completes with greedy decoding, and each dataset receives its own τ. The score Z_e is the maximum of $\bar{m}_t$ over all decision points of the episode whose chunk does not open with a stop, with the same window as at deployment, and $\tau = +\infty$ when $\lceil (n+1)(1-\alpha) \rceil > n$. Unlike deployment, calibration also scores the final decision point when its chunk contains a stop without opening with one; since that chunk ends the episode, this can only raise Z_e and hence τ, and so does not loosen the bound of Section 3.1. The simulator forces a stop once the distance to the goal has stayed unchanged for more than 25 consecutive atomic actions or more than 400 atomic actions have been executed. At a forced stop, the chunk scored at deployment can differ from the logged chunk used for calibration, so the bound of Section 3.1 can be exceeded by at most the probability that a completed episode ends in a forced stop. Forced stops end 0.15% and 0.27% of the successful rollouts of π_θ on the training splits of R2R-CE and RxR-CE. We use $\alpha = 10\%$ for R2R-CE ($n = 635$, $\tau = 2.54$) and $\alpha = 7.5\%$ for RxR-CE ($n = 1{,}103$, $\tau = 3.33$). These levels gave the highest closed-loop success among the levels we evaluated with the recovery policy on the val-unseen splits, four for R2R-CE and three for RxR-CE. They were therefore selected on the evaluation splits. Figure 4 shows R2R-CE val-unseen episode 959, rerun with the threshold calibrated at $\alpha = 7.5\%$ ($\tau = 3.08$) instead of the $\alpha = 10\%$ used for R2R-CE.

C.5 Training Start States

The judge is run offline on every decision point whose chunk does not open with a stop, over rollouts of π_θ on the training splits of R2R-CE and RxR-CE and on ScaleVLN, with $W = 3$ and per-dataset thresholds calibrated at $\alpha = 5\%$ ($\tau = 3.83$, 3.71 and 3.08 for R2R-CE, RxR-CE and ScaleVLN). The first alarm of each episode becomes a start state, and episodes on the judge's held-out paths are excluded. The pool holds 5,510 states. Of these, 24.2% come from episodes that π_θ completes, and

25.8% lie within S_d of the route. Because the offline run also scores chunks that contain but do not open with a stop, 283 of these states lie at a final decision point at which the deployed judge is not queried. A training rollout places the agent at the recorded pose with the remaining step budget, seeds the history with the current frame alone, and runs without the judge.

C.6 Counterfactual Stop Advantage

Training also uses a stop term. An outcome reward cannot tell at which chunk a rollout should have stopped, and because stops are rare, a group seldom contains both choices at one state. A stop ends the episode, however, so its return has a closed form. Let $c_{i,j}$ be the j-th chunk of ζ_i, $R^{\text{stop}}_{i,j}$ the reward of equation 7 had ζ_i stopped at the start of $c_{i,j}$, and $p_{i,j}$ the probability of opening $c_{i,j}$ with a stop under the policy that sampled ζ_i. We split the regret of not stopping between the two choices:

$$\Delta_{i,j} = R^{\text{stop}}_{i,j} - R_i, \qquad A^{\text{stop}}_{i,j} = (1 - p_{i,j})\,\Delta_{i,j}, \qquad A^{\text{cont}}_{i,j} = -p_{i,j}\,\Delta_{i,j}. \tag{10}$$

The pair has zero mean under that policy, $p_{i,j}A^{\text{stop}}_{i,j} + (1 - p_{i,j})A^{\text{cont}}_{i,j} = 0$, so its two terms need no relative weight, and they share one per-chunk weight. The advantage of $c_{i,j}$ adds $A^{\text{cont}}_{i,j}$ to $\hat{A}_i$, and the stop token receives $A^{\text{stop}}_{i,j}$. Removing this term has no measurable effect on success or on stopping behavior in our ablation (SR 66.3 against 66.2 with the term at iteration 3 and 66.6 against 67.0 at iteration 9, $p = 0.92$ and 0.53), so the main text does not treat it as a component.

C.7 Recovery Objective

The objective minimized over ψ is

$$\mathcal{L}(\psi) = \frac{1}{N}\sum_{i,j} w_{i,j}\Big[\ell\big(c_{i,j},\ \hat{A}_i + \omega_{i,j}A^{\text{cont}}_{i,j}\big) + \frac{1}{K_{i,j}}\,\ell\big(\text{stop},\ \omega_{i,j}A^{\text{stop}}_{i,j}\big) + \beta\,\hat{D}_{i,j}\Big], \tag{11}$$

where $\ell(x, A)$ is the clipped PPO loss for advantage A, averaged over the tokens of x; without the two stop terms it reduces to equation 9. For a chunk these are the unmasked tokens of its full generated text, $K_{i,j}$ counts them for $c_{i,j}$, and the mask removes sub-actions that parse to no atomic action and therefore execute a uniformly random movement. The ratio of $\pi_{\theta,\psi}$ to the sampling policy is clipped to $[0.8, 1.28]$, with a dual clip at 3 when $A < 0$. The term $\hat{D}_{i,j}$ estimates $D_{\text{KL}}(\pi_{\theta,\psi} \,\|\, \pi_\theta)$ on the same unmasked tokens by $\frac{\pi_\theta}{\pi_{\theta,\psi}} - \log\frac{\pi_\theta}{\pi_{\theta,\psi}} - 1$, and $\beta = 0.04$. There is no KL-based early stopping. The weight $w_{i,j} \propto 1/J_i$, where J_i counts the chunks of ζ_i used for training, is normalized to mean one over the chunks each worker collects in an iteration, so rollouts on one worker receive the same total weight. The weight $\omega_{i,j}$ is one for chunks that start within 5 m of the goal. Beyond 5 m, up to six chunks per rollout are drawn at random, two from each of the ranges $[5, 10)$, $[10, 15)$ and $[15, \infty)$ m where available and the rest from the remaining far chunks; they share one unit of weight, and the others get zero. This keeps the many far chunks, where $R^{\text{stop}}_{i,j} = 0$, from repeating the same signal. Because $\omega_{i,j}$ scales both terms of equation 10 alike, the pair keeps its zero mean. Here N is the number of chunks across all workers in one optimizer step, and the stop term is dropped for a chunk that opens with a stop. The stop probability $p_{i,j}$ is read from the sampling policy at the first generated position and the sampling temperature. All likelihoods in equation 11 and the stop probability $p_{i,j}$ are computed from temperature-scaled checkpoint logits without the repetition penalty of 1.05 used to sample rollouts. The standard deviation in equation 8 uses Bessel's correction.

C.8 Training and Decoding

Each iteration draws 128 alarm states and samples $G = 8$ rollouts per state at temperature 1.6 with top-$p = 1$. The reward uses $\lambda = 0.3$ and the environment's 3 m success radius, and L_i sums the displacements between the poses recorded after each step, which leaves out the displacement of the first step of the suffix; $R^{\text{stop}}_{i,j}$ uses the same convention. Optimization uses 16 samples per step and a learning rate that decays linearly from 5×10^{-5} toward 3×10^{-5}. The adapter ψ is a zero-initialized LoRA of rank 16 with scaling factor 2 (LoRA alpha 32) on the query, key, value and output projections of the language model, on the attention of four vision blocks and on the vision merger, so $\pi_{\theta,\psi} = \pi_\theta$ when training starts. Decoding is greedy before and after the alarm, at deployment and in every closed-loop evaluation.

D ADDITIONAL RESULTS

D.1 TRAINING-SIDE ABLATION OVER ITERATIONS

Table 11 extends Table 4 to all evaluated iterations. On the 527 alarmed R2R-CE episodes, the greedy success of 47.1% at iteration 15 (Section 4.3) includes 60 of the 205 base failures on which all eight samples had failed; eight samples give only a lower bound on what sampling reaches. Without the judge, $\text{GRPO}_{\text{start}}$ stays at the base level (62.3, $p = 0.21$). A binary success reward in place of suffix-SPL gives 66.5 at iteration 9, against 67.0 ($p = 0.39$).

Table 11: Training-side ablation on R2R-CE: SR at iterations 3, 9, 15 and 19, counted from 0. Start: where rollouts or demonstrations begin. Clear: whether they begin from cleared history; the row without it also retains the history at deployment. *No judge at deployment. $\text{GRPO}_{\text{binary}}$: binary success reward instead of suffix-SPL. –: does not apply or was not run.

Training	Start	Clear	SR↑ at iteration			
			3	9	15	19
None (clear only)	–	–		65.6		
SFT	alarm	✓	61.8	62.9	–	–
$\text{GRPO}^{*}_{\text{start}}$	step 0	–	–	62.3	–	–
$\text{GRPO}_{\text{start}}$	step 0	–	–	66.2	–	–
$\text{GRPO}_{\text{keep}}$	alarm	✗	65.5	66.2	67.4	67.8
$\text{GRPO}_{\text{binary}}$	alarm	✓	67.2	66.5	–	–
GRPO (ours)	alarm	✓	66.2	67.0	69.0	68.1

D.2 JUDGE COUNTS

Table 12 gives the counts behind Figure 5. The false-alarm rates of 13.0% and 8.0% compare with nominal levels of 10% and 7.5%; the calibration episodes come from held-out paths in the training scenes rather than from unseen scenes, so the rate can exceed the nominal level, as on R2R-CE. The other 43.7% of R2R-CE failures never receive an alarm, so detection bounds the gain. As in Section 2.4, clearing at an alarm can break an episode as well as rescue one: among alarmed episodes, the recovery policy raises the share of failures it completes from 29.1% with clearing alone to 43.7% and lowers the share of successes it breaks from 50.0% to 44.7% on R2R-CE. Rescued and broken episodes would cancel only if base successes made up 49.4% of the alarmed episodes on R2R-CE and 51.3% on RxR-CE, against the observed 28.8% and 24.4%, and allowing a second clearing per episode adds nothing. Alarms come a median of 13 atomic steps after the deviation onset t_d on R2R-CE and 21 on RxR-CE, partly because the window average of equation 5 delays the score, and some alarms come before t_d. The random trigger of Table 3 alarms on base failures only at the base failure rate (36.6% against 36.2%).

Table 12: Counts behind Figure 5. Brackets: Wilson 95% intervals. RxR-CE recall includes 99 base failures whose goal lies on a disconnected part of the navigation mesh, and is 589/1,187 without them.

	R2R-CE ($\alpha = 10\%$)	RxR-CE ($\alpha = 7.5\%$)
Alarm	527/1,839	779/3,669
Prec.	375/527	589/779
Recall	375/666	589/1,286
False alarm	152/1,173	190/2,383
Rescued	164/375 [38.8, 48.8]	317/589 [49.8, 57.8]
Broken	68/152 [37.1, 52.7]	97/190 [44.0, 58.1]

D.3 ALARM LEVEL

Table 13 varies the level α at which the threshold τ is calibrated, with the recovery policy of the main results. Every level improves on the base, and the reported levels give the highest success on

both datasets. In leave-one-scene-out cross-validation, which selects α on ten val-unseen scenes and scores the eleventh, ten of the eleven folds choose the reported level. The false-alarm rate follows α and exceeds it by at most 3.0 points on R2R-CE and 0.6 on RxR-CE.

Table 13: Alarm level. Alarms: share of all episodes; false alarms: share of base successes. †Reported level.

Dataset	α	τ	Alarms	False alarms	SR↑	SPL↑
R2R-CE	5.0%	3.83	16.3	5.5	66.9	60.3
	7.5%	3.08	23.3	9.4	67.9	60.4
	10.0%†	2.54	28.7	13.0	69.0	62.3
	12.5%	2.29	30.7	14.7	67.8	59.7
	Base	–	–	–	63.8	59.0
RxR-CE	2.5%	4.37	11.2	3.1	67.0	56.7
	7.5%†	3.33	21.2	8.0	70.9	60.8
	12.5%	2.71	26.7	11.7	67.8	56.4
	Base	–	–	–	64.9	55.9

D.4 Judge Query Interval

The deployed loop queries the judge at every decision point whose chunk contains no stop. Querying it at every second or fourth such decision point raises about as many alarms but later, and SR falls to 67.9 and 67.0 (Table 14).

Table 14: Judge query interval on R2R-CE. †Deployed interval.

Interval (decision points)	SR↑	SPL↑	Alarms
1†	69.0	62.3	527
2	67.9	59.9	535
4	67.0	58.8	540

D.5 Zero-Shot Object Navigation

We evaluate on the HM3D-OVON val-unseen split under its official protocol (Yokoyama et al., 2024b), as for the compared methods in Table 15, with 30° turns and success when the agent stops within 1 m of an instance of the goal category within 500 steps. The goal category is given to the policy as its instruction, `Walk to the <category> and stop by it.`, in the prompt of Appendix G. Neither the base nor the two adapters are trained on object-goal data, and both adapters run unchanged, with the threshold calibrated on R2R-CE. The route enters only the training labels of equation 4, and the judge reads only what is available to the agent (Section 3.1), so it can run on a task without a route. On HM3D-OVON the labels of equation 4 are undefined, so an alarm marks a state whose deviation score exceeds the R2R-CE threshold, not a measured deviation. The bound of Section 3.1 does not carry over either, because these episodes are not exchangeable with the R2R-CE calibration episodes, and the threshold acts as a fixed operating point. We therefore evaluate the alarm only through the SR and SPL of the full system (Table 15), against 40.7 and 21.7 for the base under the same protocol.

Table 15: Comparison on the HM3D-OVON val-unseen split. Results of other methods are taken from their papers, except VLFM, whose results are from Yokoyama et al. (2024b). **Bold** and underline mark the best and second-best results. *Zero-shot evaluation, without training on HM3D-OVON. VLFM, DAgRL+OD and MTU3D also use depth and odometry, ABot-N0 uses panoramic RGB and NavFoM four cameras. †360 randomly sampled episodes.

Method	SR↑	SPL↑
DAgRL+OD [IROS24] (Yokoyama et al., 2024b)	37.1	19.9
MTU3D† [ICCV25] (Zhu et al., 2025)	40.8	12.1
ABot-N0 [arXiv26] (Chu et al., 2026)	<u>54.0</u>	30.5
Qwen-RobotNav-4B [arXiv26] (Zhang et al., 2026b)	53.1	20.9
Qwen-RobotNav-8B [arXiv26] (Zhang et al., 2026b)	51.2	24.0
VLFM* [ICRA24] (Yokoyama et al., 2024a)	35.2	19.6
Uni-NaVid* [RSS25] (Zhang et al., 2025)	39.5	19.8
NavFoM* [ICLR26] (Zhang et al., 2026a)	45.2	<u>31.9</u>
JanusVLN* [ICLR26] (Zeng et al., 2026)	44.9	31.7
MOON (Ours)*	**57.2**	**35.4**

E Real-Robot Deployment

E.1 Robot and Control Loop

The robot is a Linkhou S2 mobile base with a forward-facing RGB camera. Navigation follows a cloud-to-robot loop. A server with an NVIDIA RTX 4090 GPU receives the RGB observation and the instruction, predicts a discrete action, and sends it back to the robot for execution. The trials cover corridors, a lounge and a kitchen, with instructions that involve passing doorways, turning and approaching named objects.

E.2 Digital Twin Construction

VLN policies are trained in simulators such as Habitat (Savva et al., 2019), whose scenes and dynamics differ from those of the deployment site, and real-world trials are hard to repeat and carry risk during iterative testing. We therefore build a digital twin of the deployment environment as an intermediate stage: the system trained in Habitat first runs in the twin under controlled, repeatable conditions and then in paired real-world trials on the same tasks. Figure 8 shows the three stages of the construction.

Mapping. We scan the deployment environment with a Lixel K2 Pro (XGrids), a handheld scanner that combines 360° LiDAR with multi-camera imagery. The resulting true-color point cloud, with centimeter-level relative accuracy, and the surface mesh give the photometric and geometric basis of the twin.

Digital Twin. We assemble the mapping outputs into a photorealistic twin that reproduces the layout and appearance of the functional areas, including the refreshment area and the lounge. Building on OmniNavBench (Sun et al., 2026), we implement the testbed in NVIDIA Isaac Sim: we import the reconstructed scene and our robot model, configure the onboard camera to match the real sensor, and specify start locations and navigation goals in the twin.

Egocentric Observation. The onboard camera renders photorealistic egocentric observations as the policy moves through the twin. Paired episodes in the twin and in the physical environment share start locations, goals and instructions, and the policy determines its own trajectory in each, so the two domains are compared under matched task conditions.

E.3 Paired Runs

Figure 9 shows two instructions that require passing a doorway and turning toward a named object; the robot reaches the goal in both runs, with the same sequence of turns and stops as in the twin.

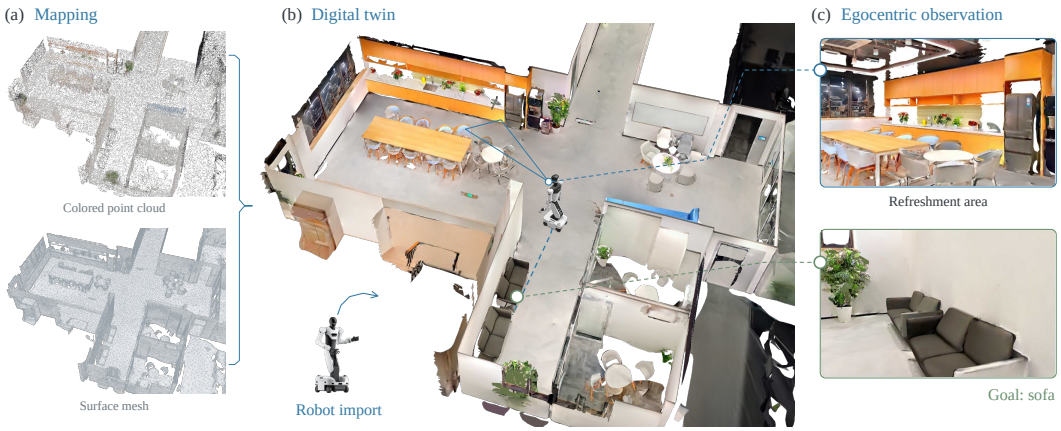


Figure 8: **Digital twin construction.** (a) The deployment site is scanned into a colored point cloud and a surface mesh. (b) The scene and the robot model are imported into NVIDIA Isaac Sim. (c) The onboard camera renders egocentric observations in the twin, here in the refreshment area and at a sofa goal.

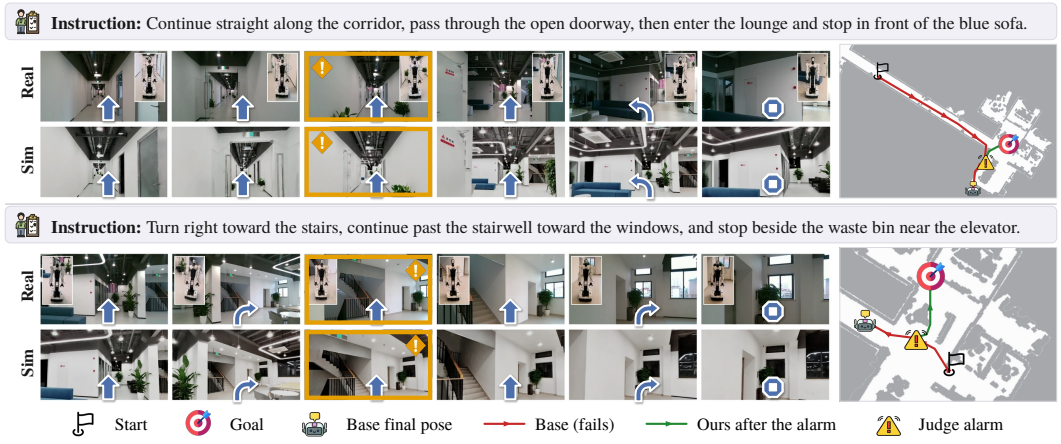


Figure 9: **Real-robot deployment without retraining.** Each block is one instruction. The top row shows first-person frames from the physical robot, with a third-person inset, and the bottom row shows the same instruction in a digital twin of the scene. Arrows mark the action executed at each frame (forward, turn or stop).

F QUALITATIVE EXAMPLES

The episode of Figure 7 and the six supplementary figures, three per dataset with three episodes each, were selected by hand among episodes that the frozen policy fails and **MOON** completes, so they illustrate recoveries rather than typical outcomes; Table 12 gives how often an alarm rescues or breaks an episode.

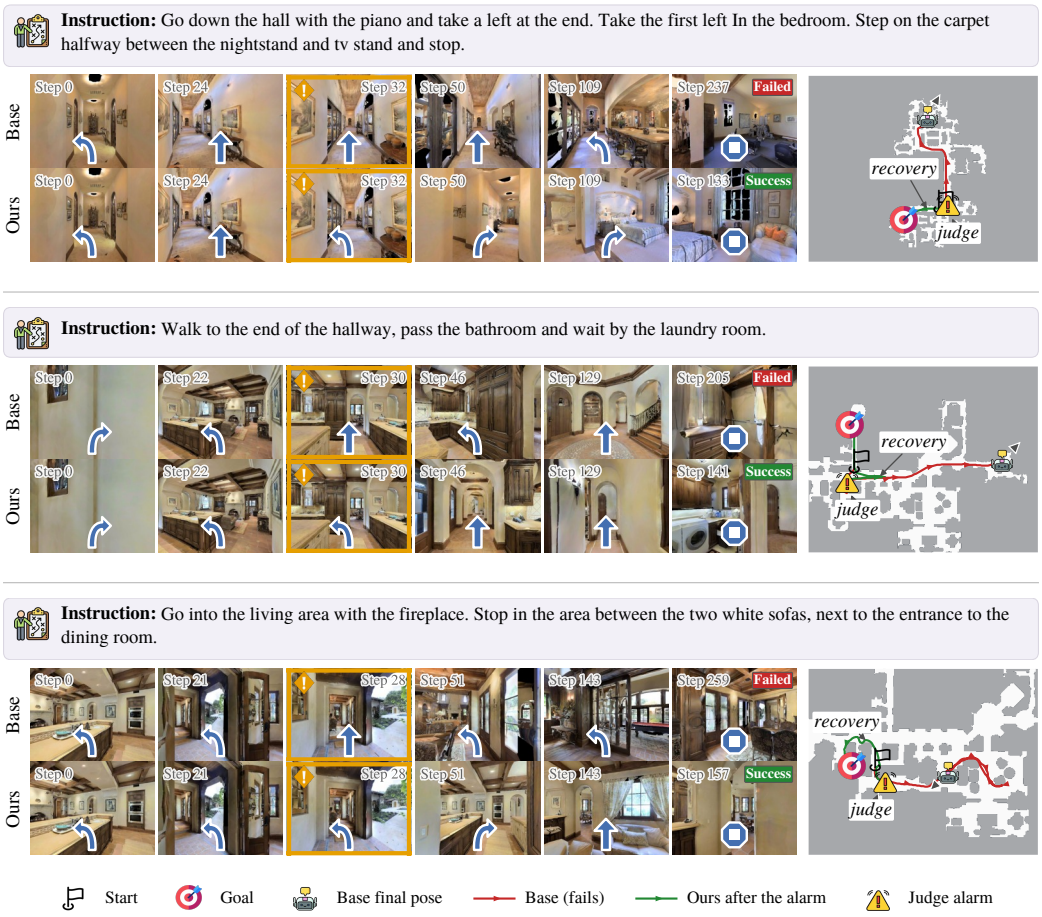


Figure 10: Additional R2R-CE episodes (1 of 3) in which the frozen policy fails and **MOON** succeeds, shown as in Figure 7.

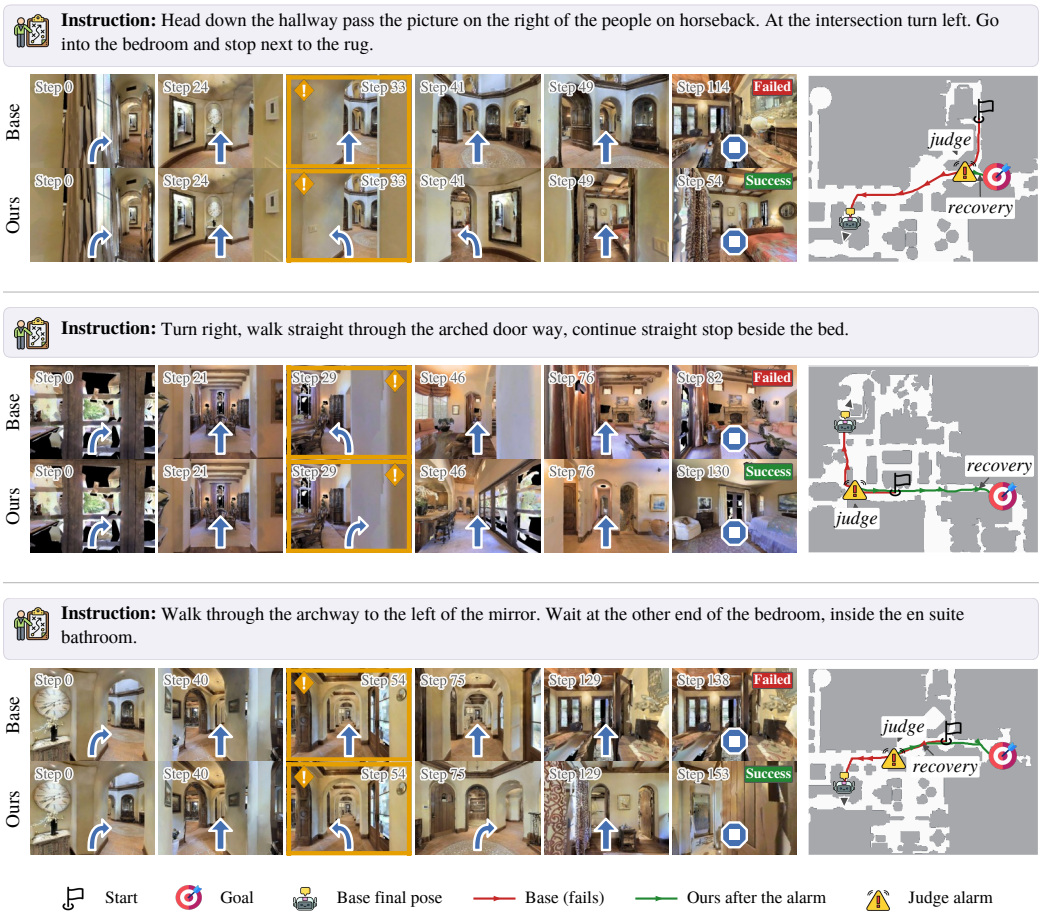


Figure 11: Additional R2R-CE episodes (2 of 3), shown as in Figure 7.

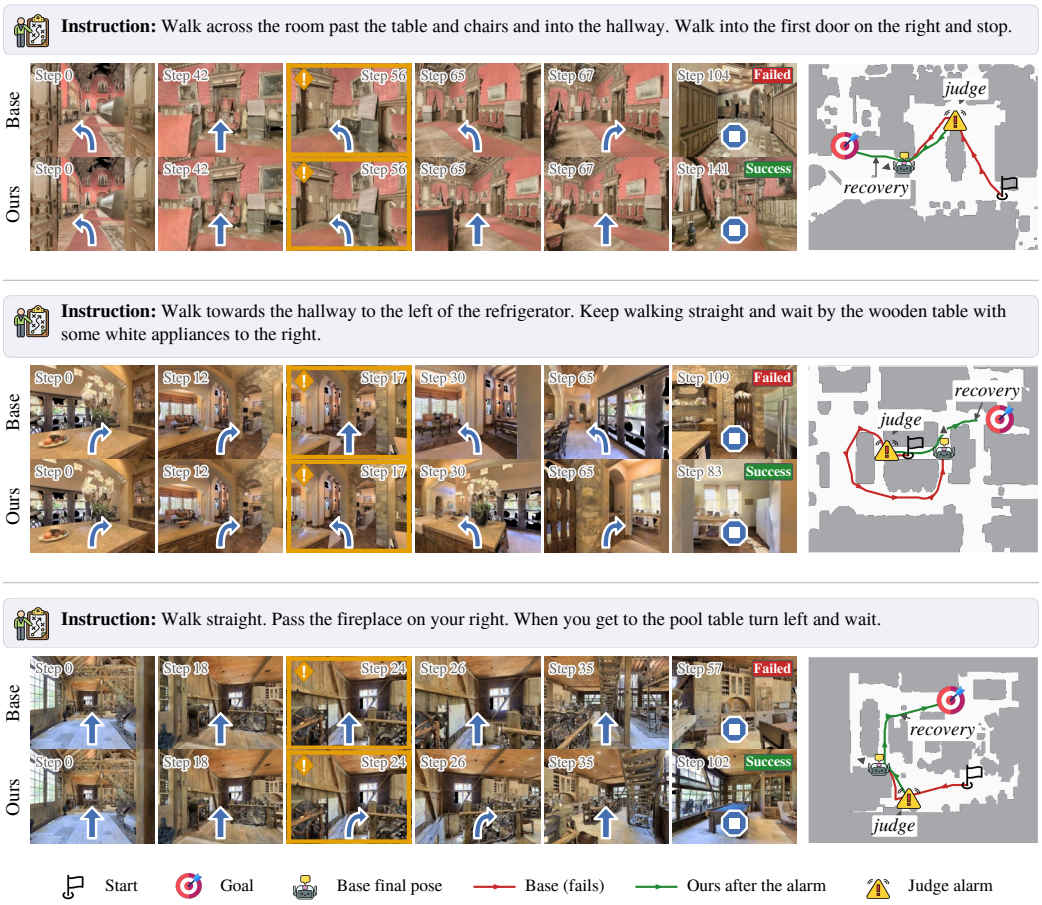


Figure 12: Additional R2R-CE episodes (3 of 3), shown as in Figure 7.

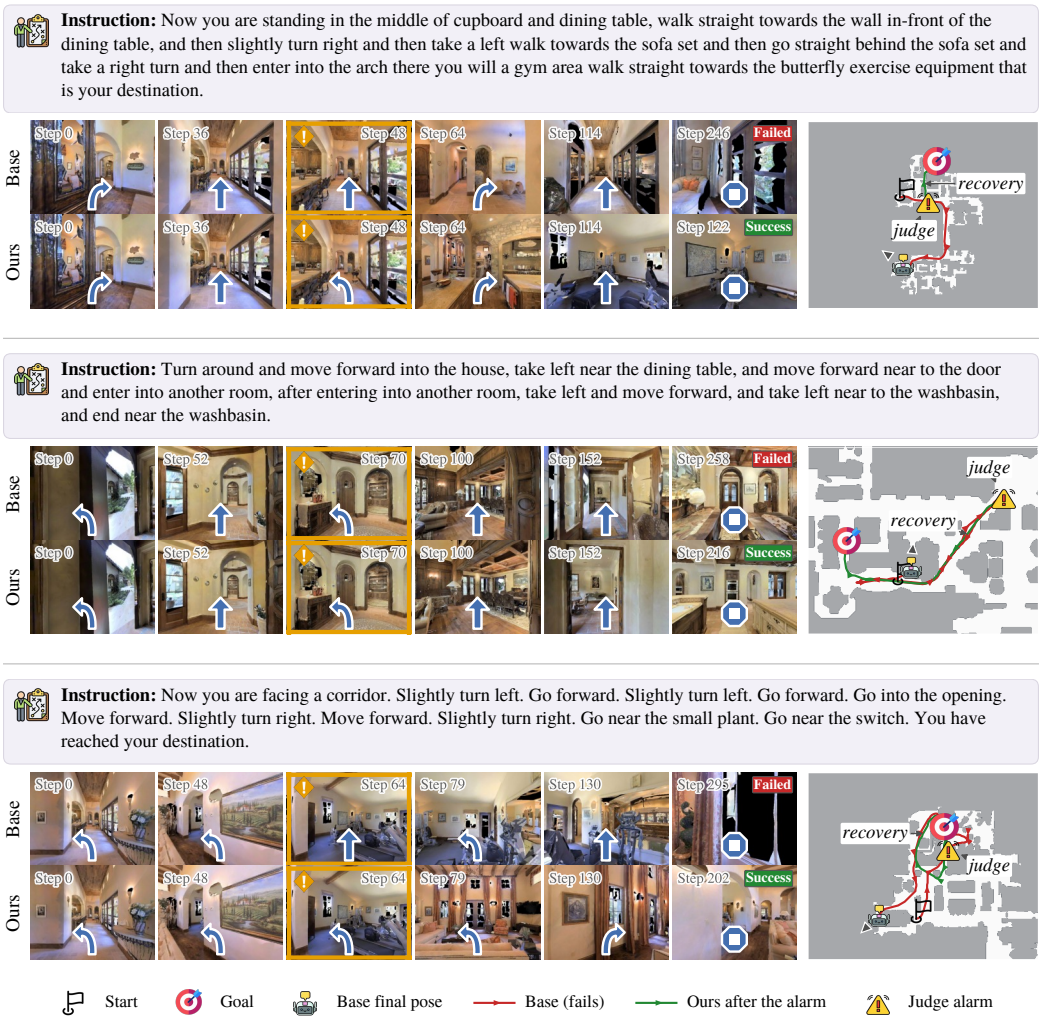


Figure 13: Additional RxR-CE episodes (1 of 3) in which the frozen policy fails and **MOON** succeeds, shown as in Figure 7.



Figure 14: Additional RxR-CE episodes (2 of 3), shown as in Figure 7.



Figure 15: Additional RxR-CE episodes (3 of 3), shown as in Figure 7.

G PROMPTS

Angle brackets mark inputs filled in at run time, frames enter as image tokens with no text between them, and within a turn each line of a box is one line of the prompt, with wrapped continuations indented.

G.1 NAVIGATION POLICY

The frozen base, the recovery policy at deployment and in GRPO training, and the SFT baseline all use the NaVIDA prompt below, with up to eight earlier frames sampled uniformly as `<frame 1>... <frame 8>` and the current frame as `<frame 9>`.

System

```
You are a helpful assistant.
```

User

```
Imagine you are a robot programmed for navigation tasks. You have
   been given a video of historical observations<frame 1>...<frame
   8>and an image of the current observation<frame 9>. Your assigned
   task is: '<instruction>'. Analyze this series of images to decide
   your next move, which could involve turning left or right by a
   specific degree or moving forward a certain distance.
```

Assistant

```
<answer>
```

At the first step of an episode and after a clearing, the history holds only the current frame, which then fills both frame slots. On HM3D-OVON, each episode fills `<instruction>` with `Walk to the <category> and stop by it.`, naming its goal category. The answer lists sub-actions separated by commas, each of the form `forward <d> cm`, `turn left <a> degree`, `turn right <a> degree` or `stop`, and the first two are executed.

G.2 DEVIATION JUDGE

The judge is trained and deployed with the prompt below, where `<frame 1>... <frame 8>` are up to eight frames spaced evenly over the history before the current frame `<frame 9>`, `<actions>` lists up to the last twelve executed atomic actions or reads `(none)` before the first one, and `<chunk>` is the proposed chunk; actions are written as `MOVE_FORWARD`, `TURN_LEFT` or `TURN_RIGHT` and separated by commas.

System

```
You are a helpful assistant.
```

User

```
You are a trajectory health critic for a vision-language navigation
   agent. Given the recent visual history, the instruction, and recent
   actions, judge whether the agent is on-track or off-track.
 This is your historical observation:<frame 1>...<frame 8>
 This is your current observation:<frame 9>
 Your task is to <instruction>
 Recent executed actions: [<actions>]
 The base policy proposes the next action chunk: [<chunk>]. Is the
   agent off-track or losing progress? Answer one word from {OK, lost,
   amb}, where 'OK' = making progress, 'lost' = off-track or
   stagnating, 'amb' = uncertain.
```

Assistant

```
<label>
```

Training supervises only the answer token, `lost` for LOST and `OK` for OK, and at deployment m_t in equation 5 is computed from the next-token probabilities of `lost` and `OK` at the answer position.